
NEXT GENERATION EQUATION-FREE MULTISCALE MODELLING OF CROWD DYNAMICS VIA MACHINE LEARNING

A PREPRINT

Hector Vargas Alvarez¹, Dimitrios G. Patsatzis¹, Lucia Russo², Ioannis Kevrekidis³, Constantinos Siettos^{4,*}

(¹) Modelling Engineering Risk and Complexity, *Scuola Superiore Meridionale*, Naples 80138, Italy

(²) Scienze e Tecnologie per l'Energia e la Mobilità Sostenibile Consiglio Nazionale delle Ricerche, Italy

(³) Dept. of Chemical and Biomolecular Engineering, Dept. of Applied Mathematics and Statistics,
Dept. of Urology, Johns Hopkins University, Baltimore, USA

(⁴) Dipartimento di Matematica e Applicazioni “Renato Caccioppoli”, *Università degli Studi di Napoli Federico II*,
Naples 80126, Italy

August 7, 2025

ABSTRACT

Bridging the microscopic/individual and the macroscopic/emergent modelling scales in crowd dynamics constitutes an important, open challenge for systematic numerical analysis, optimization, and control tasks. Here, we propose a combined manifold and machine learning approach to learn the discrete evolution operator for the emergent/collective crowd dynamics in latent spaces from high-fidelity individual/agent-based simulations. The proposed framework builds upon our previous works on the development of *next-generation Equation-free algorithms* for dealing with the “curse of dimensionality” when learning surrogate models for high-dimensional and multiscale systems. The proposed approach is a four-stage one, explicitly conserving the mass of the reconstructed dynamics in the high-dimensional space. In the first step, we derive continuous macroscopic fields (densities) from discrete microscopic data (pedestrians’ positions) using Kernel Density Estimation (KDE). In the second step, based on manifold learning, we construct a map from the macroscopic ambient space into an appropriate latent space parametrized by a few coordinates based on Proper-Orthogonal Decomposition (POD) of the corresponding density distribution. The third step involves learning reduced-order surrogate models (ROMs) in the latent space using machine learning techniques, particularly Long Short-Term Memory (LSTMs) networks and Multivariate Autoregressive models (MVARs). Finally, we reconstruct the crowd dynamics in the high-dimensional space in terms of macroscopic density profiles. We demonstrate that the POD reconstruction of the density distribution via SVD conserves the mass. With this “embed → learn in latent space → lift back to the high-dimensional space” pipeline, we create an effective solution operator of the unavailable (at the macroscopic scale) Partial Differential Equation (PDE) for the evolution of the density distribution. For our illustrations, we use the Social Force Model (SFM) to generate data in a corridor with an obstacle, imposing periodic boundary conditions. The numerical results demonstrate high accuracy, robustness, and generalizability, thus allowing for fast and accurate modelling/simulation of crowd dynamics from agent-based simulations. Notably, linear MVAR models surpass their non-linear counterparts, such as LSTMs, in predictive accuracy, while also offering significantly lower complexity and greater interpretability.

Keywords Crowd Dynamics, Multiscale modelling, Micro to Macro, Machine Learning, Numerical Analysis, Conservation laws, Reduced Order Models, Manifold Learning

*Corresponding author, email: constantinos.siettos@unina.it

1 Introduction

An important open challenge in the modelling of human crowd dynamics lies in bridging across the different modelling scales. Typically, models are formulated at three distinct levels: the microscopic (individual), mesoscopic (kinetic), and macroscopic (hydrodynamic) scales. Microscopic models describe the detailed behavior of individual pedestrians using agent-based approaches, such as the Social Force Model (SFM) [1] and its variants (see e.g. [2, 3]), cellular automata [4, 5, 6], etc. The mesoscopic scale serves as an intermediary between the microscopic and macroscopic approaches [7, 8, 9, 10, 11, 12, 13, 14, 15, 16]. At the macroscopic scale, modelling describes the emergent/collective behavior of crowds. For instance, in the seminal work of Hughes [17], the author models crowd dynamics by coupling a conservation law with an eikonal equation to represent how individuals choose paths based on a potential field. Moreover, in [18] the authors proposed nonlocal conservation laws for traffic and crowd dynamics, where the flux depends not only on local density but also on averaged information from the surrounding region. In [11] a framework for macroscopic crowd models was presented based on balance laws and kinetic theory, emphasizing how individual-level interactions give rise to complex collective behavior. More recently, in [16] are highlighted new trends such as multiscale coupling, behavioral heterogeneity, and nonlocal interaction kernels.

A major objective, when high-fidelity microsocpic/individual/agent-based simulators and/or detailed experimental data are available, is to derive mesoscopic or macroscopic models that can be then used for rigorous numerical analysis, optimization and control of the emergent/collective dynamics, for instance, in the optimal design of spaces, the reduction of evacuation times, and the management of crowd flows [11, 19, 20]. In this context, kinetic models serve as an intermediate bridge. They are typically derived analytically using statistical mechanics techniques, starting from microscopic interaction rules. These kinetic formulations, in turn, facilitate the derivation of macroscopic (continuum) models through asymptotic methods [10, 16].

The transition from high-dimensional agent-based representations to macroscopic or hydrodynamic descriptions is commonly achieved through closure techniques. These methods establish relationships between higher-order moments of the microscopic distributions and a limited, smaller set of lower-order moments that capture the observable emergent behavior [21, 22, 23]. However, such closures inherently rely on assumptions that introduce biases both for modelling and analyses tasks. Typically, these assumptions include infinitely large populations, homogeneous agents, and uniform and/or local interaction networks, but also unknown “behavioural” characteristics, which can significantly impact the accuracy and validity of collective-level analyses.

Recently, data-driven computational approaches based on physics-free machine learning (ML) have been employed for the modelling, numerical analysis, and control of pedestrian, crowd, and traffic dynamics. In this context, a wide range of ML architectures has been explored, including deep neural networks (DNNs) [24], Long Short-Term Memory (LSTM) networks [25, 26] for learning local movement decisions, Generative Adversarial Networks (GANs) [27, 28], Convolutional Neural Networks (CNNs) [29, 30, 31], and reinforcement learning methods [32, 33]. For a comprehensive review of these and related ML approaches, see [34]. These approaches have proven effective in forecasting individual and group trajectories within the framework of time-series modelling and forecasting, without relying on physics-based formulations. However, at the same time, such approaches lack interpretability and, importantly, do not incorporate information of the global “spatio-temporal” emergent dynamics, such as density fields, that are essential in modelling crowd dynamics.

An alternative data-driven paradigm that bypasses the need to explicitly learn surrogate models is the Equation-free multiscale framework [35]. By appropriately coupling on-demand microscopic simulations with coarse-graining (and fine-graining), this approach builds a coarse time-stepper for the evolution at the macroscopic scale, thus bypassing the need for analytically derived continuum equations [36, 37, 38]. While Equation-free framework facilitates powerful multiscale numerical analysis and control, it lacks an explicit, global macroscopic equation for the crowd/collective dynamics—that can be approximated, such as a mass-conserving PDE—limits interpretability, physical insight, and generalization for long-term simulations.

To address the aforementioned challenges, a few recent studies have utilized physics-informed neural networks (PINNs) [39] including physics-informed neural operators (NOs) [40, 41] for solving the inverse problem, i.e. that of constructing from data, hybrid models in the form of “grey-box” PDE models [42, 43, 21, 23, 41]. Such an approach, combines ML with underlying knowledge of physical principles, such as conservation laws [44] and closures [45], to create hybrid models that can both predict the spatio-temporal dynamics and ensure physical consistency, even under data scarcity. In the context of crowd dynamics, PINNs have been employed to construct both gray-box hydrodynamic PDEs [46] as well as for discovering the physics/rules governing interactions and motion at the microscopic level [47]. However, while PINNs provide an effective way to incorporate physical constraints and knowledge into machine learning models, they can also introduce biases due to their reliance on predefined forms of PDEs. Moreover, a key physical property in crowd modeling—mass conservation—is typically not enforced explicitly, but rather imposed

as a soft constraint through the loss function [48, 44, 49]. This indirect enforcement can limit the generalizability and physical reliability of PINN-based models, particularly in heterogeneous or unforeseen scenarios where the true dynamics deviate from the assumed PDE forms.

For learning classes of PDEs that incorporate conservation laws—such as those found in crowd dynamics—the use of black-box deep neural networks (DNNs) and black-box neural operators (NOs) [50, 51, 49, 52, 53, 23] is inherently problematic. These models often lack physical consistency, particularly in preserving conservation laws, which are usually enforced only as soft constraints in the loss function. In addition, their training suffers from low approximation accuracy due to the curse of dimensionality, potentially leading to unphysical predictions and numerical instabilities during long-term simulations. Thus, to deal with that problem, RiemannONets were recently presented in [54], which are NOs based on DeepONets to solve Riemann problems encountered in compressible flows, thus enforcing the preservation of conservation laws. Towards solving the inverse problem, we have recently proposed GoRINNs [55]: Godunov-Riemann informed neural networks for learning hyperbolic PDEs that respect the conservation laws explicitly.

It is important to note, however, that despite the long-standing tradition of modeling crowd dynamics through (hydrodynamic) PDEs, such formulations inherently rely on an “infinite-size” assumption. In realistic settings involving crowds of moderate or finite size, this assumption often breaks down. Finite-size effects—such as spatial heterogeneity, individual interactions, and boundary influences—can dominate the dynamics, rendering continuum PDEs insufficient for accurate, quantitative predictions of real-world crowd behavior.

Here, inspired by the Equation-free multiscale framework [35], and building on our previous recent works on what we call *next-generation Equation-free algorithms* based on both manifold learning and machine learning [56, 36, 57, 23, 58, 41], we present a computational framework, to learn the discrete evolution operator of the emergent dynamics in latent spaces from high-fidelity microscopic simulations of crowd dynamics. The proposed framework bridges the microscopic and macroscopic modelling scales, while explicitly preserving the underlying conservation law of mass in the ambient space. The methodology is deployed in four main stages. First, we map discrete microscopic distributions to continuous macroscopic representations (i.e., density fields). Second, we project the macroscopic fields into a low-dimensional latent space using Proper Orthogonal Decomposition (POD). Third, we learn surrogate reduced-order models (ROMs) in this latent space using discrete autoregressive maps: both linear Multivariate Autoregressive models (MVARs) and nonlinear LSTM networks. These autoregressive models with lags implicitly implement a delay coordinate embedding, aligning with Takens’ theorem [59] for reconstructing the phase-space structure (here in the latent space). The concept of the approach (i.e. that using POD embeddings and neural networks taking as inputs delayed time series for learning the dynamics of PDEs in latent spaces) can be traced back to the early ’90s [60, 61] (see also results and discussion in [61, 56, 62, 63, 57]). In crowd dynamics, behavioural characteristics such as pedestrian target paths and decision-making, or path optimization constitute the “unobservable” states, driving the observable positions/velocities and therefore the density field of the crowd; delay embeddings (as used implicitly in autoregressive models) have the potential, due to the Taken’s theorem, to reconstruct the effective phase space containing these (unobservable) latent dynamics. Finally, we lift the learned latent-space dynamics back to the high-dimensional macroscopic space via the POD basis. Crucially, this configuration ensures explicit mass conservation—a key physical constraint for any effective continuum description. For our numerical experiments, we employ the Social Force Model (SFM) [64] to simulate pedestrian flow within a rectangular corridor containing an obstacle, under periodic boundary conditions. The performance and generalization ability of the proposed framework are assessed on unseen initial conditions.

2 Methodology

2.1 Problem Statement

In this framework, we consider the problem of bridging the scales in crowd dynamics with particular focus on deriving ROMs from detailed, high-dimensional, agent-based simulations of pedestrian flows. This derivation involves reconstructing on demand the spatio-temporal dynamics under the strict constraint of mass conservation.

We utilize a detailed simulator that models the dynamics of a human crowd, consisting of N pedestrians moving within a domain $\Omega \subset \mathbb{R}^2$. The overall state of the pedestrian $i \in \{1, \dots, N\}$ is characterized by its position $\mathbf{x}_i = \mathbf{x}_i(t) = (x_i(t), y_i(t))$ and velocity $\mathbf{v}_i = \mathbf{v}_i(t) = (v_{ix}(t), v_{iy}(t))$, and also a set of “behavioral” variables $\mathbf{u}_i = \mathbf{u}_i(t) \in \mathbb{R}^b$, which may include phenomena such as decision-making strategies, herding behavior, but also panic and irrationality [65, 66, 67, 68, 69], where $b \in \mathbb{R}$ is the number of behavioral traits modeled for each pedestrian. Additionally, we introduce an auxiliary variable $\mathbf{z}_i \in \mathbb{R}^b$ to denote the rate of change of the behavioral variables \mathbf{u}_i . Then, the evolution of the position, velocity and behavior of each pedestrian is modelled by an individualistic/agent-based model, inspired

by pseudo-Newtonian/molecular/Brownian dynamics formulations [70, 16], expressed in the general form:

$$\frac{d\mathbf{u}_i}{dt} = \mathbf{z}_i, \quad \frac{d\mathbf{z}_i}{dt} = \sum_{j \in \Omega_i} \psi_i(\mathbf{x}_i, \mathbf{v}_i, \mathbf{u}_i, \mathbf{x}_j, \mathbf{v}_j, \mathbf{u}_j), \quad (1)$$

$$\frac{d\mathbf{x}_i}{dt} = \mathbf{v}_i, \quad \frac{d\mathbf{v}_i}{dt} = \sum_{j \in \Omega_i} \phi_i(\mathbf{x}_i, \mathbf{v}_i, \mathbf{u}_i, \mathbf{x}_j, \mathbf{v}_j, \mathbf{u}_j), \quad (2)$$

where j refers to all pedestrians within the interaction domain of the i -th pedestrian Ω_i , and $\psi_i(\cdot), \phi_i(\cdot)$ denote pseudo-acceleration terms incorporating interaction rules. The majority of models in this category are based on the celebrated social force model (SFM) for pedestrian dynamics [1, 16], which was proposed in the mid-1990s.

In contrast, at the macroscopic scale, crowd dynamics are typically described using phenomenological continuum models, most often formulated as partial differential equations (PDEs). These models capture the collective behavior of large-scale pedestrian flows and are grounded in the theories of hydrodynamics and continuum mechanics. Well-known examples include the so-called Hughes model [17] and multi-population nonlocal models [18]. However, such modelling approaches rely on several simplifying assumptions in order to derive algebraic closure relations that link individual-level motion with density-scale dynamics. These assumptions often include, for example, that pedestrian speed depends solely on the local total density, that individuals aim to minimize travel time while avoiding congestion, or that their motion is guided by a potential field directing them toward their destination.

To address the limitations of phenomenological continuum models, several data-driven computational methods have been introduced to construct macroscopic hybrid models in the form of “black-box” or “grey-box” PDE models [50, 51, 49, 52, 53, 23, 42, 43, 21]. These methods are predominantly based on DNNs/NOs, thus suffering from the “curse of dimensionality” that is inherent in their training. To circumvent this challenge, learning Reduced Order Models (ROMs) in well-constructed latent spaces is crucial, since it significantly lowers computational cost while capturing the essential dynamics. These ROMs must also inherently preserve fundamental principles like mass conservation, ensuring physical accuracy in the reconstructed space. Towards this goal, we aim at bridging high-fidelity microscopic/agent-based simulations and/or experimental/real-world data with macroscopic scale modelling, using ROMs that preserve mass conservation, within the framework of learning discrete operators/coarse-timesteppers [35].

The proposed methodology, presented in detail in the following section, is based on our previous efforts to develop what we call *next generation Equation-free* framework [56, 36, 57, 23, 58]. This framework deals with the “curse of dimensionality” by learning surrogate ML models in appropriate latent spaces emerging from the complex spatio-temporal dynamics of high-dimensional systems. It originates from the celebrated *Equation-free approach* [35], which enables high-dimensional simulations to perform system-level numerical analysis tasks on an appropriate latent space. This is achieved via the construction of a *lifting operator* \mathcal{L} and a *restriction operator* \mathcal{R} bridging the gap between states in a high-dimensional space, say $\mathbf{u}(t) \in \mathbb{R}^M$, and states in a latent space, say $\mathbf{y}_d(t) \in \mathbb{R}^d$, with $d \ll M$, as:

$$\mathbf{u}(t) = \mathcal{L}(\mathbf{y}_d(t)) \quad \text{and} \quad \mathbf{y}_d(t) = \mathcal{R}(\mathbf{u}(t)). \quad (3)$$

These operators are used to construct a coarse-timestepper, the discrete evolution operator at the latent space, $F_{\delta t} : \mathbb{R}^d \rightarrow \mathbb{R}^d$ defined as:

$$\mathbf{y}_d(t + \delta t) = \mathcal{R}\left(\Phi_{\Delta t}(\mathcal{L}(\mathbf{u}(t)))\right) := F_{\delta t}(\mathbf{y}_d(t)), \quad (4)$$

where $\Phi_{\Delta t} : \mathbb{R}^M \rightarrow \mathbb{R}^M$ is the evolution operator at the high-dimensional space. In this work, we learn the discrete evolution operator $F_{\delta t}$ using autoregressive ROMs, thus effectively learning $\Phi_{\Delta t}$ via the restriction and lifting operators, as:

$$\mathbf{u}(t + \Delta t) = \mathcal{L}\left(F_{\delta t}(\mathcal{R}(\mathbf{u}(t)), \dots, \mathcal{R}(\mathbf{u}(t - w\delta t)))\right), \quad (5)$$

where the learned ROMs account for w delayed embeddings $\mathcal{R}(\mathbf{u}(t - j\delta t))$ for $j = 1, \dots, w$.

A key distinction between our proposed *next generation EF* and the traditional EF framework is that, in the latter, one constructs local (numerical) maps and computes the quantities needed for bifurcation analysis on demand via short bursts of microscopic simulations to bridge detailed simulations and emergent dynamics (captured usually by a few moment distributions). In contrast, our proposed method learns a “global dynamical model” in Eq. (5) via machine learning from long-term, high-fidelity simulations spanning the entire phase space, in low-dimensional latent spaces. This global dynamical-systems-based perspective enables the use of both linear (such as POD) and nonlinear manifold-learning techniques (such as Diffusion maps [71, 72]), and techniques for the solution of the pre-image problem, such as geometric harmonics [73, 74] or autoencoders [75, 76], to construct maps between the high-dimensional and the latent space. The methodology is described analytically in the following section.

The key advantage of our *next generation EF framework* is its ability to circumvent the need for explicit derivation or closure assumptions in PDEs—a process that inherently introduces biases in modelling crowd density dynamic.

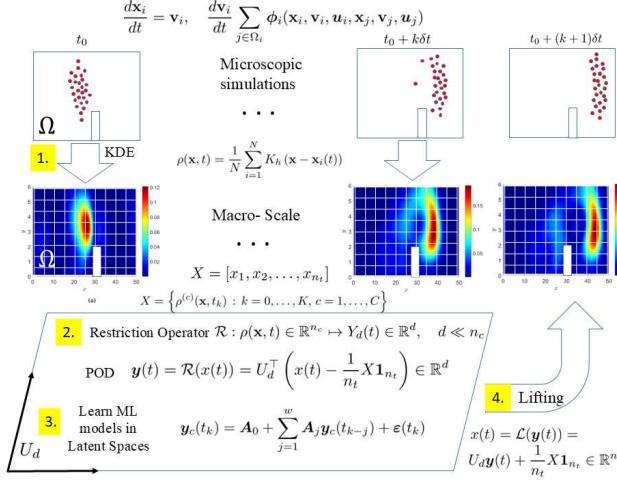


Figure 1: Schematic of the proposed methodology. First, microscopic distributions of pedestrian positions are mapped to macroscopic density fields (Step 1) and embedded into a latent space using POD (Step 2). Second, surrogate reduced-order models (MVAR and LSTM) are trained in this latent space to capture and forecast the complex dynamics (Step 3). Finally, the learned dynamics are lifted back to the high-dimensional density fields via linear projection with the constructed POD basis (Step 4).

Instead, it directly learns the “actual” solution operator in Eq. (5)—the map that advances observed macroscopic distributions (here the density) forward in time. This learned operator, embedded within computationally efficient linear or nonlinear autoregressive ROMs (MVAR or LSTM, respectively), effectively encodes the complex, often analytically intractable, dynamics of the crowd. By focusing on learning how densities evolve (the solution operator) rather than learning governing equations in the form of PDEs, the method circumvents the significant challenges of the “curse of dimensionality” associated with DNNs/NOs training of such PDEs. The resulting ROMs thus act as practical, data-driven surrogates that implicitly contain the action of the unknown PDE’s solution operator, enabling accurate prediction of future crowd states solely from observed density snapshots. Importantly, we demonstrate that for the proposed configuration, the POD method (used for identifying the latent space) conserves explicitly the mass of the reconstructed density field resulting from the ROMs. An explicit mass conservation of the continuum representations is an essential physical constraint for any credible surrogate model of crowd dynamics.

2.2 The proposed ML framework

The proposed methodology consists of four main steps. First, we map the discrete distributions of pedestrian positions into continuous density fields defined on a spatial grid using Kernel Density Estimation (KDE) [77]. Second, we construct a *restriction* operator to project the resulting density fields into an appropriate latent space. For this purpose, we employ the POD [78] method on normalized density profiles across the computational domain, ensuring that mass conservation is preserved in the reconstructed fields. Third, within this latent space, we train surrogate autoregressive ROMs using MVAR and LSTM time-series forecasting models to capture the complex dynamics present in the sequential data. Finally, in the fourth step, we lift the latent ROM dynamics back to the macroscopic density ambient space by solving the preimage problem. In the case of POD, this *lifting* step is computationally inexpensive, as it reduces to a simple linear projection onto the ambient space using the stored orthogonal basis. A schematic overview of the methodology is shown in Fig. 1.

2.2.1 From discrete pedestrian positions to continuous density fields

The initial step in our framework involves constructing a continuous density field from the discrete positions of pedestrians. Given microscopic observations, obtained by agent-based simulators in the form of Eq. (1, 2) or experimental data, macroscopic continuous quantities are typically derived via local averaging at each point of the spatial domain where the crowd moves. Here, we start from N pairs of pedestrian positions $\{\mathbf{x}_i(t)\}_{i=1}^N \subset \Omega$ in a rectangular domain $\Omega = [a, b] \times [c, d] \subset \mathbb{R}^2$, as obtained by the SFM microscopic simulator in Appendix A. We next employ *Multivariate Kernel Density Estimation* (KDE) [79] to estimate the continuous density fields along points on the spatial grid $\mathbf{x} \in \Omega$,

as:

$$\rho(\mathbf{x}, t) = \frac{1}{N} \sum_{i=1}^N K_h(\mathbf{x} - \mathbf{x}_i(t)), \quad (6)$$

where K_h is a smooth non-negative kernel function satisfying $\int_{\Omega} K(\mathbf{x}) d\mathbf{x} = 1$. Using a standard uniform spatial discretization $\delta x = (b-a)/n_x$, $\delta y = (c-d)/n_y$ with n_x/n_y grid points along the x/y -axis of the domain Ω , the resulting from Eq. (6) density fields are computed as $\rho(\mathbf{x}, t) = [\rho((x_1, y_1), t), \dots, \rho((x_{n_x}, y_{n_y}), t)]^\top \in \mathbb{R}^{n_x \times n_y}$, where $\rho((x_i, y_j), t)$ corresponds to the density computed at the grid point (x_i, y_j) , for $i = 1, \dots, n_x$, $j = 1, \dots, n_y$. Then, the total mass in the domain is given by $S(t) = \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \rho((x_i, y_j), t) \delta x \delta y$. While KDE can be used for extracting continuous macroscopic fields, such as momentum and energy, from any microscopic discrete quantities, here we only construct the density fields in Eq. (6). Additional details for the discretization of the computational domain Ω , the choice of kernel $K_h(\cdot)$, the consideration of obstacles in Ω (where a binary mask function is employed to consider zero density at the grid-points of the obstacle), and the treatment of periodic boundary conditions are provided in Appendix C.

2.2.2 The Restriction and Lifting Operators

Having obtained the macroscopic density fields, we now proceed to the second and fourth key elements of the approach, which, in analogy to the Equation-free framework, constitute the construction of an “embedding/restriction” and “lifting” operators to and from an appropriate low-dimensional latent space.

The restriction operator, say, \mathcal{R} , [35], is a map from density distributions $\rho(\mathbf{x}, t)$ to a low-dimensional latent space coordinates, say $Y_d(t)$, i.e.,:

$$\mathcal{R} : \rho(\mathbf{x}, t) \in \mathbb{R}^{n_c} \mapsto Y_d(t) \in \mathbb{R}^d, \quad d \ll n_c, \quad (7)$$

where $n_c = n_x \times n_y$. To construct the restriction operator, we apply *Proper Orthogonal Decomposition* (POD) [78] to a dataset consisting of macroscopic density “solutions” in time, obtained by microscopic simulations via Eq. (6), for different initial conditions; see Appendix B for details. Denoting the densities by $\rho^{(c)}(\mathbf{x}, t_0 + k\delta t) \equiv \rho^{(c)}(\mathbf{x}, t_k) \in \mathbb{R}^{n_c}$, where δt is a microscopic inner simulation step/observation sampling time of an experiment/real data, t_0 is the initial time, and the superscript (c) denotes the case $c = 1, \dots, C$ of the different initial conditions considered, the training dataset generated by microscopic runs is represented by the matrix $X \in \mathbb{R}^{n_c \times (C \cdot K)}$:

$$X = \left\{ \rho^{(c)}(\mathbf{x}, t_k) : k = 0, \dots, K, c = 1, \dots, C \right\}. \quad (8)$$

Compactly, we represent the above data set as:

$$X = [x_1, x_2, \dots, x_{n_t}] \in \mathbb{R}^{n_c \times n_t}, \quad (9)$$

with columns, say, $x_k \in \mathbb{R}^{n_c}$ being the normalized density field along cases and timesteps $k = 1, \dots, n_t = C \cdot K$.

To complete the restriction operator at the second step, we employ POD on X , implemented via Singular Value Decomposition (SVD) on the centered matrix:

$$\bar{X} = XH = U\Sigma V^T, \quad H = I_{n_t} - \frac{1}{n_t} \mathbf{1}_{n_t} \mathbf{1}_{n_t}^\top, \quad (10)$$

where $U \in \mathbb{R}^{n_c \times n_c}$ is the matrix collecting the left-singular vectors, $\Sigma \in \mathbb{R}^{n_c \times n_t}$ is the matrix collecting the singular vectors, $V \in \mathbb{R}^{n_t \times n_t}$ is the matrix collecting the right-singular vectors, I_{n_t} is the n_t -dim. unity matrix and $\mathbf{1}_{n_t}$ is the n_t -dim. unit column vector. The latent space is spanned by the first d left-singular vectors forming U_d . Thus, projecting X on the latent space we get, the low-dimensional embedding:

$$Y_d = U_d^T \bar{X}, \quad Y_d \in \mathbb{R}^{d \times n_t} \quad (11)$$

For the lifting operator that maps Y_d to the space of density profiles, we demonstrate that the POD reconstruction (aka the POD lifting operator) preserves the original density. This aligns with the general characteristic of POD in maintaining symmetries [80]. Here, in particular, we demonstrate the following proposition.

Proposition 1. Let us assume a matrix $X \in \mathbb{R}^{n_c \times n_t}$, with columns being normalized density fields. If the density is preserved along timesteps, i.e., if

$$[\sum_{i=1}^{n_c} x_{i,1} \quad \sum_{i=1}^{n_c} x_{i,2} \quad \dots \quad \sum_{i=1}^{n_c} x_{i,n_t}] = \mathbf{1}_{n_t}^\top, \quad (12)$$

then the reconstructed density field computed by the POD as:

$$\tilde{X} = U_d U_d^\top \bar{X} + X(I_{n_t} - H), \quad (13)$$

where $U_d = [u_1, u_2, \dots, u_d] \in \mathbb{R}^{n_c \times d}$ is the orthonormal basis formed by the first d left-singular vectors computed by the SVD, is also preserved, i.e.,:

$$[\sum_{i=1}^{n_c} \tilde{x}_{i,1} \quad \sum_{i=1}^{n_c} \tilde{x}_{i,2} \quad \dots \quad \sum_{i=1}^{n_c} \tilde{x}_{i,n_t}] = \mathbf{1}_{n_t}^\top. \quad (14)$$

Proof. As by hypothesis, the sum of each column of X is equal to one (that is $\mathbf{1}_{n_c}^\top X = \mathbf{1}_{n_t}^\top$), we have that:

$$\mathbf{1}_{n_c}^\top \bar{X} = \mathbf{1}_{n_c}^\top X H = \mathbf{1}_{n_t}^\top H = \mathbf{1}_{n_t}^\top - \frac{1}{n_t} \mathbf{1}_{n_t}^\top \mathbf{1}_{n_t} \mathbf{1}_{n_t}^\top = \mathbf{1}_{n_t}^\top - \mathbf{1}_{n_t}^\top = \mathbf{0}_{n_t}^\top. \quad (15)$$

Therefore, the sum of each column of the centered matrix \bar{X} is equal to zero.

This implies that $\mathbf{1}_{n_c}$ is orthogonal to the column space of \bar{X} , and therefore is orthogonal to the left singular vectors of the SVD decomposition of \bar{X} , which provide an orthogonal base for the column space, i.e.,:

$$\mathbf{1}_{n_c}^\top U_d = \mathbf{0}_d^\top, \quad (16)$$

where $d = 1, 2, \dots, r$, with $r = \text{rank}(X)$. Then, the multiplication of the reconstructed matrix in Eq. (13) by $\mathbf{1}_{n_c}^\top$ implies:

$$\mathbf{1}_{n_c}^\top \tilde{X} = \mathbf{1}_{n_c}^\top U_d U_d^\top \bar{X} + \mathbf{1}_{n_c}^\top X (I_{n_t} - H) = \mathbf{0}_{n_t}^\top + \frac{1}{n_t} \mathbf{1}_{n_t}^\top \mathbf{1}_{n_t} \mathbf{1}_{n_t}^\top = \mathbf{1}_{n_t}^\top, \quad (17)$$

thus getting:

$$\mathbf{1}_{n_c}^\top \tilde{X} = [\sum_{i=1}^{n_c} \hat{x}_{i,1} \quad \sum_{i=1}^{n_c} \hat{x}_{i,2} \quad \dots \quad \sum_{i=1}^{n_c} \hat{x}_{i,n_t}] = \mathbf{1}_{n_t}^\top. \quad (18)$$

Hence, the POD reconstruction preserves the original density. \square

In summary, given a density distribution $\rho(\mathbf{x}, t)$ obtained via Eq. (6), we obtain the normalized density along the n_c grid points of Ω as:

$$x(t) = \frac{1}{\sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \rho((x_i, y_j), t) \delta x \delta y} [\rho((x_1, y_1), t), \dots, \rho((x_{n_x}, y_{n_y}), t)]^\top \in \mathbb{R}^{n_c} \quad (19)$$

and define the *restriction* operator \mathcal{R} by the projection on the POD basis of Eq. (11) as:

$$\mathbf{y}(t) = \mathcal{R}(x(t)) = U_d^\top \left(x(t) - \frac{1}{n_t} X \mathbf{1}_{n_t} \right) \in \mathbb{R}^d, \quad (20)$$

where X is the dataset matrix in Eq. (8) and U_d is the matrix containing the first d left-singular vectors. On the other hand, the *lifting* operator \mathcal{L} that maps latent coordinates, say $\mathbf{y}(t)$, back to the density distributions is defined via the POD reconstruction as:

$$x(t) = \mathcal{L}(\mathbf{y}(t)) = U_d \mathbf{y}(t) + \frac{1}{n_t} X \mathbf{1}_{n_t} \in \mathbb{R}^{n_c}. \quad (21)$$

The above definitions of the restriction and lifting operators guarantee that the reconstructed, by the lifting operator, normalized density profiles preserve the total density within the domain Ω , as per Proposition 1. We highlight here that while the operators are constructed based on the dataset X , their employment applies to any seen or unseen observation $x(t) \in \mathbb{R}^{n_c}$ obtained from Eq. (19) and $\mathbf{y}(t) \in \mathbb{R}^d$.

2.2.3 Learning the dynamics in latent spaces using MVAR and LSTM

With the restriction operator constructed, which maps density fields to latent coordinates, the third step of the proposed approach involves training ROMs to learn the latent dynamics from sequential data. We note that while the restriction and lifting operators \mathcal{R} and \mathcal{L} are, in principle, defined over continuous time inputs, we apply them at a discrete set of uniformly spaced time instances $t_k = t_0 + k\delta t$ for $k = 1, 2, \dots, K$, consistent with the temporal sampling of our data; see Eq. (8). To consider lags in the latent coordinate embeddings $\mathbf{y}(t_k)$, we employ both linear and nonlinear autoregressive models, namely MVARs and LSTMs.

Let an MVAR model of order w (i.e., with w time lags) be expressed as:

$$\mathbf{y}_c(t_k) = \mathbf{A}_0 + \sum_{j=1}^w \mathbf{A}_j \mathbf{y}_c(t_{k-j}) + \boldsymbol{\varepsilon}(t_k), \quad (22)$$

where $\mathbf{y}_c(t_k)$ is the latent variable vector, evaluated at discrete times t_k for each of the $c = 1, \dots, C$ cases of different initial condition considered, $\mathbf{A}_0 \in \mathbb{R}^d$ is the intercept vector and $\mathbf{A}_j \in \mathbb{R}^{d \times d}$ are the regression coefficient matrices for each lag j . The predictor vectors $\mathbf{y}_c(t_{k-j}) \in \mathbb{R}^d$ include the w past lag latent variables evaluated at discrete times t_{k-1}, \dots, t_{k-w} , while the error term $\varepsilon(t_k) \in \mathbb{R}^d$ is assumed to follow a multivariate white noise process:

$$\mathbb{E}[\varepsilon(t_k)] = \mathbf{0}, \quad \mathbb{E}[\varepsilon(t_k)\varepsilon(t_k)^\top] = \Sigma, \quad \mathbb{E}[\varepsilon(t_k)\varepsilon(t_r)^\top] = \mathbf{0}, \quad \text{for } t_k \neq t_r, \quad (23)$$

for every $k, r = 1, \dots, K$.

Given sequential (time series) data for $\{\mathbf{y}_c(t_k)\}$, the unknown parameters of the MVAR in Eq. (22), i.e., the intercept vector \mathbf{A}_0 and the coefficient matrices \mathbf{A}_j , can be computed by the solution of the following least-squares problem:

$$\underset{\mathbf{A}_0, \{\mathbf{A}_j\}_{j=1}^w}{\operatorname{argmin}} \mathcal{L}_{\text{MVAR}} (\{\mathbf{y}_c(t_k)\}, \{\hat{\mathbf{y}}_c(t_k)\}; \mathbf{A}_0, \{\mathbf{A}_j\}_{j=1}^w), \quad (24)$$

where the MVAR loss function is:

$$\mathcal{L}_{\text{MVAR}} (\{\mathbf{y}_c(t_k)\}, \{\hat{\mathbf{y}}_c(t_k)\}; \mathbf{A}_0, \{\mathbf{A}_j\}_{j=1}^w) = \sum_{c=1}^C \sum_{k=w+1}^K \left\| \mathbf{y}_c(t_k) - \hat{\mathbf{y}}_c(t_k) \right\|_2^2, \quad (25)$$

with the predictions of the MVAR model given by:

$$\hat{\mathbf{y}}_c(t_k) = \mathbf{A}_0 + \sum_{j=1}^w \mathbf{A}_j \mathbf{y}_c(t_{k-j}), \quad (26)$$

for $c = 1, \dots, C$ and $k = 1, \dots, K$. The optimization problem in Eq. (24) can be solved using methods such as regularized least square methods; in particular, we used ridge regularization [81]. We note that MVARs require the trivial assumption of time series stationarity, which was the case for our illustrations. If the latent coordinate time series is non-stationary, pre-processing steps like differencing can be applied to achieve stationarity before solving the optimization problem. We highlight that MVAR model training further requires the selection of an optimal lag window size w , which determines the length of the time series history used to model the temporal dependencies. Here, we used two information-theoretic criteria for determining w , the details for which are provided in Appendix D.

As a non-linear autoregressive model alternative to linear MVARs, we further consider LSTM networks, a type of Recurrent Neural Networks (RNNs) characterized by gating mechanisms, which allow the network to retain and propagate information over long sequences selectively, but also effectively handle vanishing gradients; a common challenge in traditional RNNs [82, 83, 84, 85]. Let an LSTM model with past w latent states $\mathbf{y}_c(t_{k-j})$ for $j = 1, \dots, w$ be expressed as:

$$(\mathbf{h}_\tau, \mathbf{c}_\tau) = F_{\text{LSTM}}(\mathbf{y}_c(\tau), \mathbf{h}_{\tau-1}, \mathbf{c}_{\tau-1}; \boldsymbol{\theta}), \quad \text{for } \tau = t_{k-w}, \dots, t_{k-1}, \quad (27)$$

$$\hat{\mathbf{y}}_c(t_k) = \mathbf{W}_y \mathbf{h}_{t_{k-1}} + \mathbf{b}_y, \quad (28)$$

where $\mathbf{h}_{t_k} \in \mathbb{R}^{N_h}$ and $\mathbf{c}_{t_k} \in \mathbb{R}^{N_h}$ are the, evaluated at discrete times t_k , hidden and memory cell states, respectively, N_h corresponds to the number of hidden units and $\mathbf{W}_y \in \mathbb{R}^{d \times N_h}$, $\mathbf{b}_y \in \mathbb{R}^d$ define an output layer mapping the last hidden state $\mathbf{h}_{t_{k-1}}$ to the predicted latent variable vector $\hat{\mathbf{y}}_c(t_k) \in \mathbb{R}^d$. The gating mechanism of the LSTM is included in Eq. (27) at $F_{\text{LSTM}}(\cdot)$, which we avoid here for the conciseness of the presentation; a vanilla LSTM including 4 gates, contains the parameters $\boldsymbol{\theta} \in \mathbb{R}^{4N_h(d+N_h+1)}$.

Similarly to the sequence-to-one learning of MVARs, given sequential data for $\mathbf{y}_c(t_k)$ for a fixed lag window of width w , we train the LSTM model by solving the optimization problem:

$$\underset{\boldsymbol{\theta}, \mathbf{W}_y, \mathbf{b}_y}{\operatorname{argmin}} \mathcal{L}_{\text{LSTM}} (\{\mathbf{y}_c(t_k)\}, \{\hat{\mathbf{y}}_c(t_k)\}; \boldsymbol{\theta}, \mathbf{W}_y, \mathbf{b}_y), \quad (29)$$

where the LSTM loss function is the mean squared error (MSE) between the true value of the latent variable vector $\mathbf{y}_c(t_k)$ and the predicted one $\hat{\mathbf{y}}_c(t_k)$, as provided by Eqs. (27, 28), that is:

$$\mathcal{L}_{\text{LSTM}} (\{\mathbf{y}_c(t_k)\}, \{\hat{\mathbf{y}}_c(t_k)\}; \boldsymbol{\theta}, \mathbf{W}_y, \mathbf{b}_y) = \frac{1}{C(K-w)} \sum_{c=1}^C \sum_{t_k=w+1}^K \left\| \mathbf{y}_c(t_k) - \hat{\mathbf{y}}_c(t_k) \right\|_2^2. \quad (30)$$

To solve the optimization problem in Eq. (30), one typically uses gradient-based methods; in particular, we employed the Adam optimizer [86], which is a stochastic variant of gradient descent, known for its adaptive learning-rate capabilities.

We highlight that while MVAR models are theoretically simple and efficient, their strict reliance on stationary data poses a significant limitation, requiring explicit pre-processing that may distort dynamics or lose information [87]. In contrast, LSTMs as dynamical models can inherently handle non-stationarity through their gated memory cells and adaptive hidden states. This allows LSTM to learn complex, long-range temporal dependencies and nonlinear relationships directly from raw data. However, this flexibility comes at the cost of needing much larger datasets, greater computational resources, and often lacking interpretability compared to MVAR models.

Finally, the predicted latent coordinates $\hat{\mathbf{y}}_c(t_k)$ obtained from the trained MVARs or LSTMs models are subsequently mapped back to the high-dimensional density space using the lifting operator defined in Eq. (21).

3 Case study and methodology implementation

3.1 Configuration

The proposed methodology is employed on a crowd dynamics configuration where pedestrians are moving from left to right in a corridor of dimensions $48 \text{ m} \times 12 \text{ m}$ (x : length, y : width) in the presence of an obstacle, as shown in Fig. 2. The square obstacle of area 12.96 m^2 is centered at $(25.8 \text{ m}, 1.8 \text{ m})$, with the pedestrians navigating around it towards

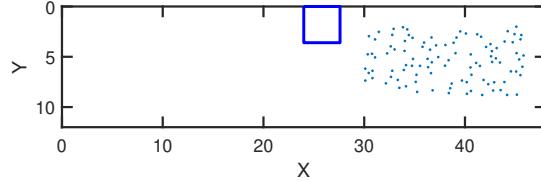


Figure 2: The case study: The SFM simulating pedestrians moving in a corridor with an obstacle.

reaching the corridor's right end. Hence, the computational domain is $\Omega = ([0, 48] \times [0, 12]) \setminus ([24, 27.6] \times [0, 3.6])$.

For modelling the pedestrians' movement, we use the SFM [1] as a microscopic agent-based simulator. SFM falls within the general form of Eqs. (1, 2) and the idea behind it is that the motion of individuals is governed by “virtual forces”. These forces include acceleration toward the desired destination, as well as repulsive forces “exerted” by obstacles, borders, and nearby individuals to maintain a safe distance. Additionally, one may also consider an attractive force due to mimetic behavior, which can be observed in scenarios such as emergency evacuations or collective movement toward objects of common interest [1]. The derivation of SFM is based on the Langevin-type equation that contains a frictional term, a deterministic force term, and a random force term. In the SFM, these terms translate into a goal-directed force (playing the role of friction), social and repulsive (from obstacles) deterministic forces, and a random term, reading:

$$m_i \frac{d\dot{\mathbf{x}}_i}{dt} = \underbrace{-\frac{1}{\tau_i}(v_{0,i}\mathbf{e}_i - \mathbf{v}_i)}_{\text{goal-directed force}} + \underbrace{\sum_{j \neq i} \mathbf{F}_{i,j}(\mathbf{x}_i, \mathbf{x}_j)}_{\text{social force}} + \underbrace{\sum_{\text{obstacles}} \mathbf{F}_{i,B}(\mathbf{x}_i)}_{\text{repulsive force}} + \mathbf{F}_{rand}, \quad (31)$$

where m_i is the mass of the i -th pedestrian, $v_{0,i}$ is their maximum walking speed, \mathbf{e}_i represents the desired direction (serving as a simplistic auxiliary/behavioral variable), and τ_i is a characteristic relaxation time. Here, we avoid the inclusion of random forces, since we are interested in the macroscopic behavior that often emerges from the deterministic pedestrian interactions [1]. Details on the SFM forces included in Eq. (31) and the related parameters that were considered in this work are provided in Appendix A.

We simulate a crowd of $N = 100$ pedestrians moving through a corridor, with initial spatial distributions ranging from homogeneous dispersions to clustered configurations (for more details, see in the Appendix B). Pedestrians are directed to move from left to right by setting their initial target at coordinates $(x, y) = (25 \text{ m}, 9.6 \text{ m})$, corresponding to the midpoint of the corridor in the horizontal direction and 80% of its vertical extent. Upon reaching this intermediate waypoint, their final destination is updated to $(x, y) = (48 \text{ m}, 4.2 \text{ m})$, which lies at the right end of the corridor and at 35% of its height. To maintain a constant population density, we implement periodic boundary conditions by re-spawning agents that reach the right boundary at $x = 48 \text{ m}$, at the left boundary at $x = 0 \text{ m}$.

Pedestrians are prohibited from passing the corridor's walls or entering the obstacle area by introducing strong wall forces in the SFM microscopic simulator. Such configuration ensures a controlled pedestrian flow through the simulated environment and also ensures the pedestrians avoid the obstacle each time they reenter the corridor.

3.2 Data generation of microscopic distributions

We generate synthetic data of pedestrian trajectories by numerically integrating the SFM with a first-order explicit Euler scheme using a small fixed time step of $\delta t_{micro} = 0.025$ s to ensure stability and accuracy. Each simulation has a time span of 275 s. To reduce computational storage costs while retaining essential temporal features, we subsample the resulting pedestrian trajectories at intervals of $\delta t = 0.25$ s. As already discussed, we generate pedestrian trajectories from various initial spatial distributions modelling diverse crowd scenarios. In particular, we consider $C = 10$ different initializations with zero velocities and position distributions including: (i) uniform distribution to provide a homogeneous baseline scenario, (ii) Gaussian clusters to model localized groups of pedestrians, (iii) piecewise linear distributions with higher densities near the wall boundaries to simulate characteristic wall-avoidance behavior, (iv) sinusoidal distributions to generate periodic macroscopic behavior, and (v) double Gaussian distributions to capture the formation of two distinct crowds. Details on the form and parameters of the above initializations are provided in Appendix B, distinguishing between those used for training and testing the proposed framework in Tables B.1 and B.2, respectively. The resulting datasets are comprised of 10 cases with different initializations, each case containing a matrix $\mathcal{X}^{(c)} = \{\mathbf{x}_i(t_k), i = 1, \dots, N, k = 1, \dots, K\} \in \mathbb{R}^{200 \times 1100}$, where $2N = 200$ corresponds to the pedestrians' positions and $K = 1100$ timesteps are sampled from each trajectory. Thus, the total datasets overall cases are $\mathcal{X}_{tr}, \mathcal{X}_{ts} \in \mathbb{R}^{200 \times (10 \cdot 1100)}$ used for training and testing, respectively.

3.3 Extraction of continuous density fields

For the first step of our framework, we derive macroscopic density fields from pedestrian positions via KDE, as described in Section 2.2.1. To achieve this, we discretize the computational domain Ω into $n_x \times n_y = 80 \times 20$ control volumes with uniform spacing $\delta x = \delta y = 0.6$ m; this resolution reflects a typical personal space for each pedestrian, allowing for capturing emergent collective behaviors, while minimizing artificial grid effects. For the KDE implementation on the cell centroids, we choose Gaussian kernels in Eq. (6) tuned with bandwidths $\mathbf{H} = diag(h_x, h_y) = (3, 2)$ for suppressing noise and preserving features of the microscopic data, while ensuring that local structures remain clear without introducing spurious artifacts. To account for the physical obstacle in Ω , a binary mask is applied to explicitly zero out density estimation on the cell centroids of the obstacle geometry. To account for the periodic boundary conditions, we further perform domain augmentation for the KDE support. All the details regarding KDE implementation are provided in Appendix C.

Using the above procedure, we derive the density fields $\rho^{(c)}(x_i, y_j, t_k)$ ($c = 1, \dots, C$, $i = 1, \dots, n_x$, $j = 1, \dots, n_y$ and $k = 1, \dots, K$) from the pedestrian positions included in the training \mathcal{X}_{tr} and testing data sets \mathcal{X}_{ts} . For conforming to the assumptions of Proposition 1, we finally normalize each density field following Eq. (19), thus constructing the normalized density matrices $X_{tr}, X_{ts} \in \mathbb{R}^{n_c \times n_t}$ in the form of Eq. (8), where $n_c = n_x \times n_y = 1600$ and $n_t = C \cdot K = 10 \cdot 1100$.

3.4 POD for the restriction and lifting operators

For constructing the restriction and lifting operators in Eqs. (7) and (21) respectively, we employ POD on the training dataset X_{tr} to determine the latent dimension d and the POD basis U_d . Using the economy-size SVD (which reduces computational complexity in thin matrices while maintaining exact reconstruction capability), we determine d by the best low-rank approximation of X_{tr} retaining 99% of the energy; thus, computing the minimum number d satisfying the criterion:

$$E_d = \frac{\sum_{i=1}^d \sigma_i^2}{\sum_{i=1}^r \sigma_i^2} \geq 0.99 \quad (32)$$

where $r = \min(n_c, n_t) = 1600$ is the rank of X_{tr} , and $\sigma_i > 0$ for $i = 1, \dots, r$ are the singular values in descending order. Next, using the first d left singular vectors of X_{tr} , we form U_d for completing the construction of the restriction and lifting operators in Eqs. (7, 21). To assess the accuracy of restriction and lifting operators, we evaluate the POD reconstruction error on the training and testing data sets by the relative L_2 error:

$$e_k^{2,rec} = \frac{\|x(t_k) - \mathcal{L}(\mathcal{R}(x(t_k)))\|_2}{\|x(t_k)\|_2}, \quad (33)$$

where $x(t_k) \in X_{tr}, X_{ts}$ with $k = 1, \dots, K$ is the normalized density defined in Eq. (19).

3.5 Autoregressive ROMs and their training

Having obtained the latent representations $y(t) \in \mathbb{R}^d$ from the restriction operator, the third step of the proposed approach is to construct autoregressive ROMs for learning the dynamics in the latent space. For this purpose, we learn

linear MVARs and non-linear LSTM models, as described in Section 2.2.3, for predicting the latent variable $\hat{y}_c(t_k)$ of the c case at timestep t_k given the past w lags $\{\mathbf{y}_c(t_{k-w}), \dots, \mathbf{y}_c(t_{k-1})\}$. To conform to this sequence-to-one learning, the features consist of the dataset $Y_f = \{\mathbf{y}_c(t_{k-w}), \dots, \mathbf{y}_c(t_{k-1})\} \in \mathbb{R}^{d \times w \times C \cdot (K-w)}$ and the targets consist of the data set $Y_t = \{\mathbf{y}_c(t_k)\} \in \mathbb{R}^{d \times C \cdot (K-w)}$, where each $\mathbf{y}_c(t_k)$ is computed via the restriction operator in Eq. (7) on the training data X_{tr} .

As a preliminary step, we first tune the lag window w —within a bound of $w_{max} = 100$ —using two information-theoretic criteria: the Bayesian Information Criterion (BIC) and the Akaike Information Criterion (AIC) [88, 89, 90], the details of which are described in Appendix D. Both criteria are likelihood-based, with AIC tending to select larger windows than BIC. We note that BIC and AIC are well-suited for MVARs models since they assume stationarity of the data and white noise residuals. For comparison, we adopt the same w for learning LSTMs, as the one tuned for MVARs by these criteria.

For the MVARs model training, we first verify that all input time series are stationary using the augmented Dickey-Fuller (ADF) test [91] with a threshold of significance level $p < 0.01$. Then, we solve the least-squares optimization problem in Eq. (24) using ridge regression with a regularization parameter $\lambda = 10^{-6}$ to compute the coefficients A_0 and A_j in Eq. (22).

For the LSTM model, we consider a single LSTM layer with 16 units and a hyperbolic tangent activation function in the hidden layers; the output layer is linear, as denoted in Eq. (28). To train the LSTM model, we solve the optimization problem in Eq. (29) using the Adam optimizer [86] with automatic differentiation for the gradients of the loss function, enabled through MATLAB’s Deep Learning Toolbox [92]. The model’s parameters are initialized with a Glorot uniform initialization [93], the initial learning rate is set to $l_r = 10^{-3}$, and the mini-batch size is set to 32; the latter hyperparameters were tuned on a trial-and-error basis, providing a good compromise between computational efficiency and generalization without exceeding memory constraints. The maximum number of epochs was set to 100 as no further improvement was achieved using more epochs. To further avoid overfitting of the LSTM, we split the training data into training and validation sets using a 80-20% ratio, and shuffle each data set without breaking the internal temporal structure of the sequences. We compute the loss function value on the validation set for every epoch and stop the training after 5 consecutive validation checks with no improvement.

To assess the convergence of both MVARs and LSTMs training, we report the values of the loss functions in Eqs. (25) and (30), respectively, at the end of training. Since the MVARs loss function is based on L_2 error, we further report the MSE:

$$MSE_{\text{MVAR}}(\{\mathbf{y}_c(t_k)\}, \{\hat{\mathbf{y}}_c(t_k)\}; A_0, \{A_j\}_{j=1}^w) = \frac{1}{C(K-w)} \sum_{c=1}^C \sum_{t_k=w+1}^K \|\mathbf{y}_c(t_k) - \hat{\mathbf{y}}_c(t_k)\|^2, \quad (34)$$

where $\hat{\mathbf{y}}_c(t_k)$ is the prediction in Eq. (26), to enable direct comparison with the MSE-based loss function of LSTMs. Additionally, we report the computational times required for training both ROMs; we expect MVARs to be much faster than LSTMs due to the lower number of parameters and the ridge regression employed for solving the related optimization problem.

3.6 Accuracy assessment of the proposed framework

Having constructed/trained each step of the proposed framework—construction of the restriction/lifting operators and training of the ROMs—, we finally assess the forecasting performance of the whole framework on the $c = 10$ different initializations of the testing set \mathcal{X}_{ts} ; see Appendix B for details. In particular, we first obtain the “ground truth” normalized densities for each initialization, say $x^{(c)}(t_0 + k \delta t) \equiv x^{(c)}(t_k) \in X_{ts}$ for all time steps $k = 1, \dots, K$. Then, following the proposed restrict-evolve with ROM-lift framework in Eq. (5), we forecast the normalized density at the k time step, using the w past normalized densities; i.e., computing:

$$\hat{x}^{(c)}(t_k) = \mathcal{L} \left(\Phi_{\text{ROM}} \left(\mathcal{R}(\hat{x}^{(c)}(t_{k-1})), \dots, \mathcal{R}(\hat{x}^{(c)}(t_{k-w})); \mathbf{p} \right) \right), \quad (35)$$

where \mathcal{R} and \mathcal{L} are the restriction and lifting operators in Eqs. (7) and (21) and Φ_{ROM} is the autoregressive ROM with parameters \mathbf{p} that can be either the trained MVARs or LSTMs model. Notice that while Eq. (35) could consider the restricted values of the normalized densities $\mathcal{R}(x^{(c)}(t_{k-j}))$ for $j = 1, \dots, w$ as inputs, it instead considers the predicted normalized densities; thus, evaluating the proposed framework in a closed-loop (autoregressive) manner. This allows us to feed subsequent predictions recursively into Eq. (35) to advance the forecast over time.

To quantify the prediction accuracy, we compute the relative L_1 , L_2 and L_∞ errors between the “ground truth” densities $x^{(c)}(t_k)$ and the predicted ones $\hat{x}^{(c)}(t_k)$ per case c and time step k as:

$$e_k^{1,(c)} = \frac{\|x^{(c)}(t_k) - \hat{x}^{(c)}(t_k)\|_1}{\|x^{(c)}(t_k)\|_1}, \quad e_k^{2,(c)} = \frac{\|x^{(c)}(t_k) - \hat{x}^{(c)}(t_k)\|_2}{\|x^{(c)}(t_k)\|_2}, \quad e_k^{\infty,(c)} = \frac{\|x^{(c)}(t_k) - \hat{x}^{(c)}(t_k)\|_\infty}{\|x^{(c)}(t_k)\|_\infty}. \quad (36)$$

We report these errors *overall cases and time steps*, also computing the 10-90 percentiles, and track down the evolution of these errors along time steps.

All numerical computations were executed on a PC equipped with an 11th Gen Intel(R) Core(TM) i7-1165G7 2.80 GHz CPU and 16 GB of system memory.

4 Numerical results

Here, we present the results of the proposed framework on predicting pedestrian flow following the configuration of a rectangular corridor with an obstacle described in Section 3.1. As already discussed in Section 3.2, we first collected pedestrian trajectory data using the SFM microscopic agent-based simulator under a wide range of initial conditions. The resulting training dataset \mathcal{X}_{tr} is used to train our framework, while the testing dataset \mathcal{X}_{ts} is used to assess its efficiency and prediction accuracy.

At the first step, we extracted the normalized density fields data X_{tr}, X_{ts} from the discrete pedestrian positions in $\mathcal{X}_{tr}, \mathcal{X}_{ts}$ using KDE, as described in Section 3.3. Next, we employed POD on \mathcal{X}_{tr} to discover the latent space dimension; see Section 3.4. Following the criterion in Eq. (32), the minimum latent dimension required for retaining more than 99% of the energy in the dataset is $d = 13$ modes, as shown in Fig. 3a). Using the discovered POD basis U_d , we

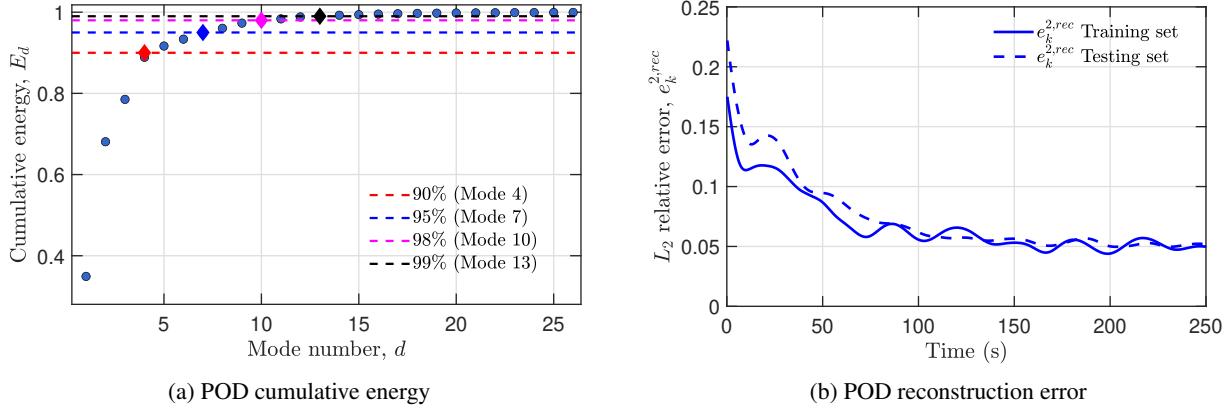


Figure 3: Accuracy of the restriction and lifting operators in Eqs. (7) and (21). Panel a) shows the minimum number of POD modes (colored diamonds) required for retaining the desired percentage of energy in the training data set. Panel b) depicts the average (over the $c = 10$ cases with different initial conditions) reconstruction error $e_k^{2,rec}$ in Eq. (33) with $d = 13$ POD modes, in time over the training and testing datasets.

next constructed the restriction and lifting operators in Eqs. (7) and (21). To assess their accuracy, we computed the relative L_2 reconstruction error $e_k^{2,rec}$ in Eq. (33) across the $k = 1, \dots, K$ time steps of each trajectory. Figure 3b) displays the average relative error *overall* $c = 10$ cases with different initial conditions included in either the training or testing datasets. While in the beginning, the reconstruction error is more than 10%, after the first 100 time steps, it reduces to 5% for both seen (training set) and unseen (testing set) data, indicating high accuracy of the restriction and lifting operators with a significantly reduced latent space dimension $d = 13$. We note that this level of reconstruction error serves as a baseline for the proposed framework, since the timeseries forecasting with ROMs in the latent space introduces additional approximation errors.

Having constructed the restriction and lifting operators (second and fourth step of the framework), we next train MVARs and LSTMs autoregressive ROMs to learn the latent dynamics; see Section 2.2.3. First, we obtained the latent variables $\mathbf{y}(t) \in \mathbb{R}^{13}$ from the density profiles of the training set, using the restriction operator. We then determined the lag window, using the BIC and AIC criteria described in Appendix D; BIC indicated a lag window of $w = 4$, whereas the AIC favored (a wider, as expected) $w = 9$. Both windows were used for training both MVARs and LSTMs

models, denoted from now on as MVAR(4), MVAR(9), LSTM(4) and LSTM(9) models. For performing sequence-to-one learning, we further divided the restricted training data set into features $Y_f = \{\mathbf{y}_c(t_{k-w}), \dots, \mathbf{y}_c(t_{k-1})\}$ and targets $Y_t = \{\mathbf{y}_c(t_k)\}$.

For the MVARs training, we first verified that the latent dynamics datasets Y_f, Y_t constitute stationary time series. Using the ADF test, we confirmed that all d variables are stationary processes with a significance level of $p = 0.001$. We next trained all MVAR and LSTM models using algorithms, hyperparameters and stopping criteria discussed in detail in Section 3.5. The training results are shown in Table 1 for the four ROMs, including the loss function values attained at the end of training and the computational time (in seconds) required. Since the MVARs loss function is L_2

Table 1: Training results of MVAR(4), MVAR(9), LSTM(4) and LSTM(9) models. Loss function values at the end of training for MVARs $\mathcal{L}_{\text{MVAR}}(\{\mathbf{y}_c(t_c)\}, \{\hat{\mathbf{y}}_c(t_k)\}; \mathbf{A}_0, \{\mathbf{A}_j\}_{j=1}^w)$ in Eq. (25) and for LSTMs $\mathcal{L}_{\text{LSTM}}(\{\mathbf{y}_c(t_c)\}, \{\hat{\mathbf{y}}_c(t_k)\}; \boldsymbol{\theta}, \mathbf{W}_y, \mathbf{b}_y)$ in Eq. (30), along with computational times (in seconds) required for training. For MVARs, the MSE in Eq. (34) is reported for comparison to the MSE-based loss $\mathcal{L}_{\text{LSTM}}(\cdot)$.

Model	$\mathcal{L}_{\text{MVAR}}(\cdot)$	$MSE_{\text{MVAR}}(\cdot)$	$\mathcal{L}_{\text{LSTM}}(\cdot)$	Comput. Time (s) ↓
MVAR(4)	1.66×10^{-3}	1.96×10^{-11}	-	0.0055
MVAR(9)	1.01×10^{-3}	7.38×10^{-12}	-	0.042
LSTM(4)	-	-	8.64×10^{-7}	180
LSTM(9)	-	-	3.51×10^{-7}	443

based, we further report the MSE, computed in Eq. (34), for MVARs to allow comparison with the MSE-based loss of the LSTMs. Table 1 indicates that all the optimization problems converge, as evident from the low loss function values attained. MVARs are able to reach much smaller loss function values than LSTMs, providing a hint for their better accuracy. As expected, LSTMs require significantly higher computational time for training in comparison to MVARs. Table 1 further shows that the training of both types of ROM converges to lower loss function values with increased lag window, albeit requiring significantly more computational time.

Having constructed and trained all components of our framework, we now evaluate its end-to-end performance, using the forecasting closed-loop time-stepper in Eq. (35) for long-horizon forecasting. In particular, we quantify the prediction accuracy via the relative L_1, L_2 and L_∞ errors, $e_k^{1,(c)}, e_k^{2,(c)}, e_k^{\infty,(c)}$ in Eq. (36) between the “ground truth” densities $x^{(c)}(t_k)$ and the predicted ones $\hat{x}^{(c)}(t_k)$ for the c , unseen in training, cases—of different initial conditions—included in the testing set X_{ts} across k time steps. Table 2 provides a summary of the relative errors $e_k^{1,(c)}, e_k^{2,(c)}, e_k^{\infty,(c)}$ for all four ROMs, including the median and the 10-90 percentiles, *overall* cases and time steps of the testing set. As shown

Table 2: End-to-end numerical accuracy of the closed-loop framework across the testing set, using the trained MVARs and LSTMs models for the latent dynamics. The relative L_1, L_2 and L_∞ errors of the high-dimensional density space, $e_k^{1,(c)}, e_k^{2,(c)}, e_k^{\infty,(c)}$ in Eq. (36) are reported, for the four MVAR(4), MVAR(9), LSTM(4), LSTM(9) latent dynamics models; median and 10-90 percentiles (in parenthesis) are shown *overall* $C = 10$ cases and $K = 1091$ time steps.

Model	L_1 error, $e_k^{1,(c)}$	L_2 error, $e_k^{2,(c)}$	L_∞ error, $e_k^{\infty,(c)}$
MVAR(4)	0.3599 (0.2408, 0.4535)	0.2536 (0.1609, 0.3098)	0.2251 (0.1361, 0.3435)
MVAR(9)	0.3783 (0.2865, 0.4655)	0.2765 (0.2040, 0.3367)	0.2569 (0.1935, 0.3330)
LSTM(4)	0.6323 (0.4313, 0.7941)	0.4160 (0.2889, 0.5269)	0.3372 (0.2532, 0.5037)
LSTM(9)	0.4646 (0.3172, 0.6820)	0.3185 (0.2151, 0.4626)	0.3010 (0.1993, 0.4393)

in Table 2, the prediction accuracy of the closed-loop time stepper with the use of MVAR models outperforms the one with LSTMs, demonstrating robust generalization with tighter 10-90 percentile ranges. In addition, while MVARs with shorter lag demonstrate slightly higher approximation accuracy, LSTMs significantly improve with higher lags, albeit remaining inferior to MVARs. We hereby note that, considering the baseline 5-10% L_2 relative error of POD reconstruction (see Fig. 3), the L_2 errors are 3-10 times higher for the end-to-end closed-loop framework, underscoring the compounding effect of autoregressive prediction.

To highlight the robust performance of the proposed closed-loop time-stepping framework over time, we further depict the relative L_2 and L_∞ errors, $e_k^{2,(c)}, e_k^{\infty,(c)}$ in time, across the multiple cases of different, unseen initial conditions in Figs. 4 and 5 for MVARs and LSTMs, respectively. In particular, Figs. 4,5 show the average relative errors $e_k^{2,(c)}, e_k^{\infty,(c)}$ and their 10-90 percentile ranges, over the $c = 10$ cases considered, across time steps $k = 1, \dots, 1100$.

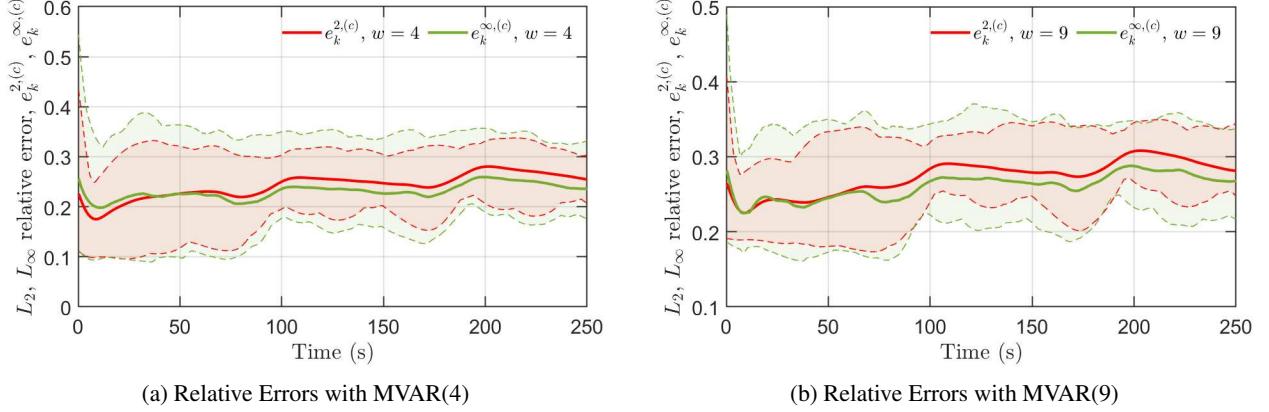


Figure 4: End-to-end numerical accuracy of the closed-loop framework across the testing set over time, using the trained MVAR models. The relative L_2 (red) and L_∞ (green) errors of the high-dimensional density space, $e_k^{2,(c)}$, $e_k^{\infty,(c)}$ in Eq. (36) are displayed for the MVAR(4) (panel a) and MVAR(9) (panel b) latent dynamics models; average (solid) and 10-90 percentiles (dashed) are shown overall $C = 10$ cases per time step.

Figures 4a,b) depict the results obtained with the MVAR(4) and MVAR(9) models, respectively, both indicating that, in spite of the long-time prediction, the relative error remains bounded in values below 30%; 5-10% is the baseline error of POD reconstruction in Fig. 3. In addition, the tight 10-90 percentile ranges indicate good generalization. Although both models exhibit comparable error profiles, the MVAR(4) model demonstrates slightly improved accuracy while using fewer parameters, as also reported in Table 2.

Similar bounded results are reported in Fig. 5 (a,b), depicting the employment of LSTM(4) and LSTM(9) models for long-time predictions. As expected by the results in Table 2, the use of LSTMs provides less accurate predictions in

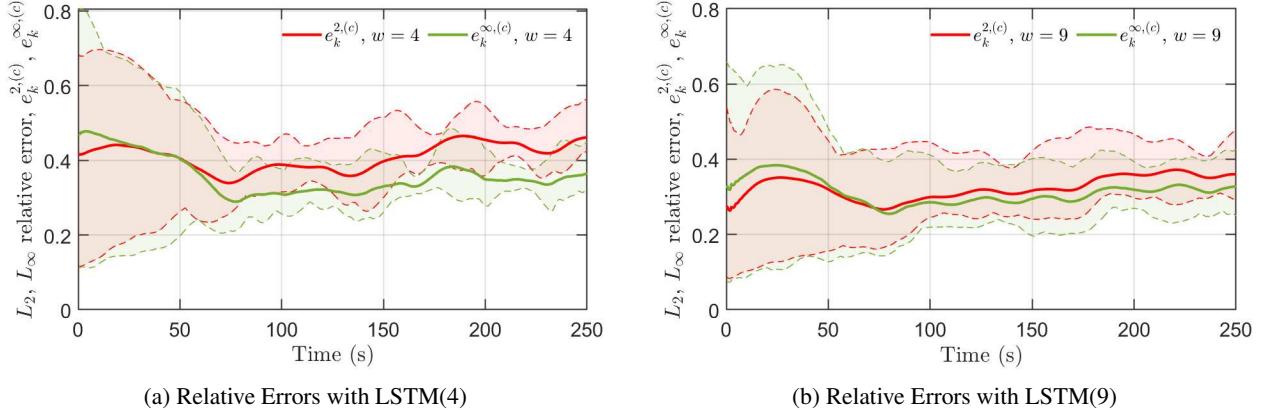


Figure 5: End-to-end numerical accuracy of the closed-loop framework across the testing set over time, using the trained LSTM models. The relative L_2 (red) and L_∞ (green) errors of the high-dimensional density space, $e_k^{2,(c)}$, $e_k^{\infty,(c)}$ in Eq. (36) are displayed for the LSTM(4) (panel a) and LSTM(9) (panel b) latent dynamics models; average (solid) and 10-90 percentiles (dashed) are shown overall $C = 10$ cases per time step.

comparison to MVARs. However, LSTMs provide tighter 10-90 percentile ranges towards the end of time integration, indicating a more robust behavior in comparison to MVARs. In addition, as also reported in Table 2, a wider lag window makes the scheme with LSTMs more accurate, while that of MVARs less accurate. Nevertheless, when comparing across model classes, the MVAR(4) model achieves slightly lower relative errors overall, while also requiring significantly fewer trainable parameters and hyperparameters. This makes MVAR(4) not only more accurate in this setting but also more computationally efficient.

For completeness, we provide a comparison of the “ground truth” normalized density fields $x^{(c)}(t_k)$ and the predicted ones $\hat{x}^{(c)}(t_k)$ derived based on the closed-loop time-stepper in Eq. (35) for a particular case, in the testing data set, of an initially spread pedestrian group; Gaussian initialization corresponding to the 6th row of Table B.2. Figure 6 displays such a comparison for the MVAR(4) model in four selected time steps ($k = 250, 500, 780, 930$) where the pedestrian crowd does not pass through the corridor boundary; similar comparisons are provided for the MVAR(9), LSTM(4) and LSTM(9) models in Figs. E.1, E.2, and E.3 of Appendix E.

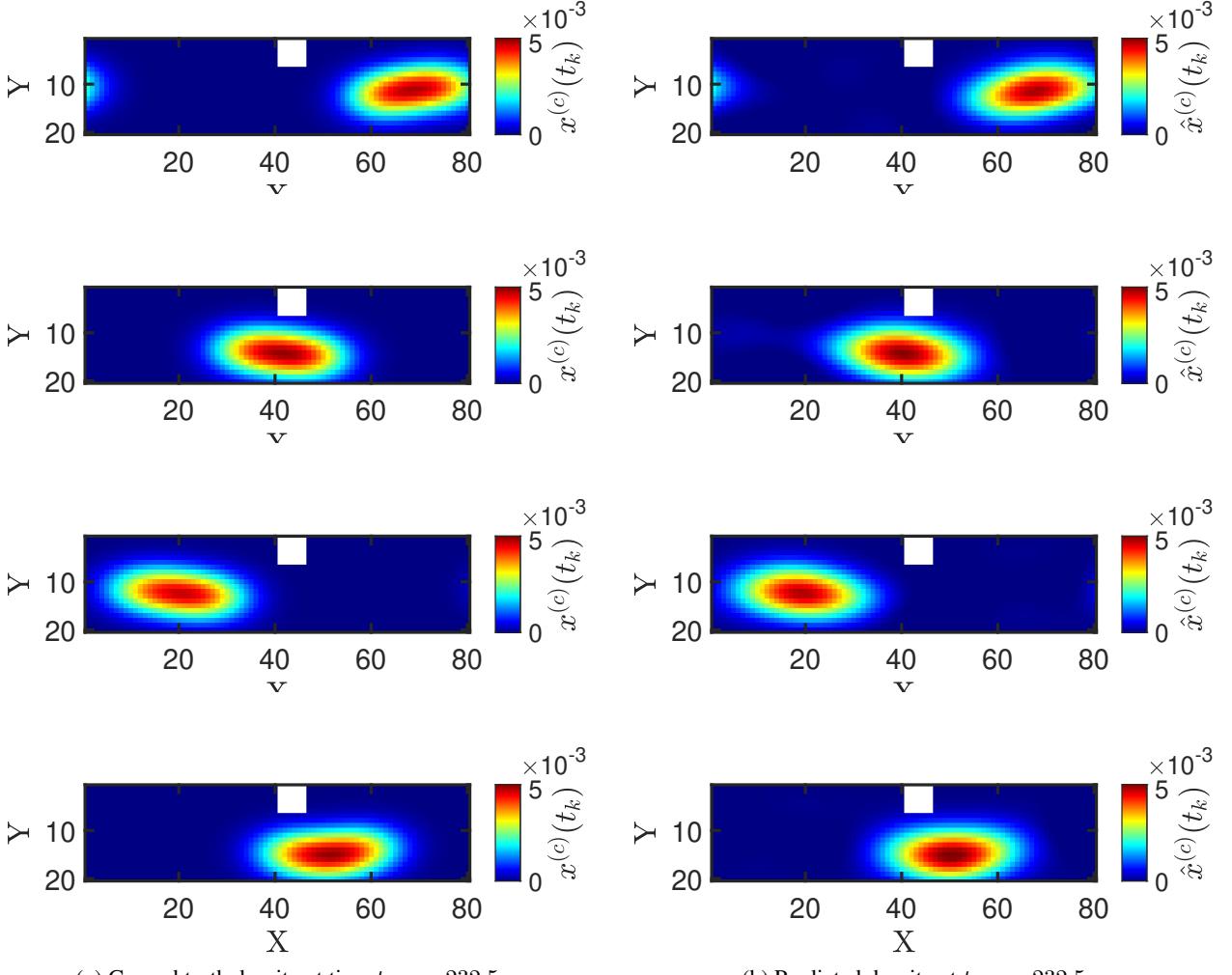


Figure 6: Comparison of the “ground truth” $x^{(c)}(t_k)$ and the predicted $\hat{x}^{(c)}(t_k)$ via the closed-loop time-stepper in Eq. (35) using the MVAR(4) model, normalized densities for an unseen case of the testing set; initialization at the 6th row of Table B.2. Panels a,c,e, and g show the “ground truth” density distribution in Ω , while panels b,d,f, and h, show the predicted one, for the time steps $t_{250} = 62.5$ s, $t_{500} = 125$ s, $t_{780} = 195$ s, and $t_{930} = 232.5$ s, respectively.

fields preserve the global structure and propagation direction observed in the “ground truth” density, demonstrating the framework’s ability to capture dominant crowd dynamics and to deliver meaningful long-horizon predictions. The predicted density fields additionally succeed in correctly respecting the obstacle region. Comparison with the other (less accurate) models in Figs. E.1, E.2, E.3 reveals that while the above two crucial for crowd flow characteristics are preserved, the LSTM(4) model (the least accurate one) tends to underestimate peak density values, introduce artificial density in low density regions (for coping with conservation of mass), and exhibit greater dissipation in regions of high density; these erroneous behaviors relax as more accurate models are used, i.e., the LSTM(9) and the MVAR(9) ones.

As shown, the proposed framework provides a balanced trade-off between prediction accuracy and computational efficiency for coarse-grained crowd dynamics. In particular, for the benchmark case of 100 pedestrians moving in the

corridor for more than 275 s, the conventional numerical integration requires a computational time of 20 s for the microscopic SFM simulator simulation plus 30 s for the extraction of the density fields (in a coarser time discretization), summing up to a total of 50 s. In contrast, our framework requires just 0.0063, 0.0161 s for the MVARs models and 6.75, 6.77 s for the LSTM ones to complete the predictions across the long-time horizon integration—representing a speed-up of 3,100x and 7.4x for the MVARs and LSTMs, respectively. Even when accounting for the offline costs (POD basis construction and ROM training), MVARs retain a significant advantage, with training times under 0.1 s (as shown in Table 1). The MVARs variants prove particularly efficient for real-time applications, while maintaining better accuracy than their LSTM counterparts. These gains are especially valuable for real-world scenarios, where the number of pedestrians renders the conventional approaches unfeasible.

5 Conclusion

In this work, we present a four-step data-driven framework that combines manifold learning and machine learning to construct a discrete evolution operator for emergent crowd dynamics. Our framework provides a highly computationally efficient and systematic approach for learning accurate ROMs for spatio-temporal emergent crowd dynamics. By first encoding the full high-dimensional state into a compact latent representation, we learn the system’s evolution operator using surrogate ROMs—sidestepping the prohibitive computational cost of learning PDEs via for example convolutional neural networks (CNNs) or other high-dimensional ML models such as DNNs. Working in latent space simplifies the enforcement of physical constraints (such as mass conservation) over long time horizons, since, as we demonstrate, the POD algorithm used to find an appropriate set of coordinates of the latent space explicitly conserves the mass of the reconstructed dynamics.

Thus, our approach does not attempt to learn any closed-form PDE, but instead learns the effective solution operator of such an unavailable PDE, mapping initial crowd density to its future evolution. The resulting ROMs trained in the latent space, in the form of linear (MVARs) or nonlinear multivariate autoregressive models (such as LSTMs), encode and reconstruct this operator numerically via the POD reconstruction. As a result, our framework allows for rapid and accurate “what-if” scenarios exploration and real-time forecasting of crowd dynamics without sacrificing accuracy or stability, making them an ideal tool for large-scale, real-world crowd-dynamics analysis and control.

Finally, we note that linear MVARs exhibit a better performance in terms of the median relative L_1 , L_2 , L_∞ errors when compared to LSTMs, thus offering a significantly lower complexity in their training and greater interpretability. This is in line with the results presented in other works [56, 94, 63, 57] which utilize PCA-based delay-coordinate embeddings to construct linear ROMs using in latent spaces to approximate complex nonlinear or chaotic dynamics.

References

- [1] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical Review E*, 51(5):4282–4286, 1995.
- [2] Alessandro Corbetta, Jasper A Meeusen, Chung-min Lee, Roberto Benzi, and Federico Toschi. Physics-based modeling and data representation of pairwise interactions among pedestrians. *Physical Review E*, 98(6):062310, 2018.
- [3] Xiaoxia Yang, Hairong Dong, Qianling Wang, Yao Chen, and Xiaoming Hu. Guided crowd dynamics via modified social force model. *Physica A: Statistical Mechanics and its Applications*, 411:63–73, 2014.
- [4] Kazume Nishidate, Mamoru Baba, and Richard J Gaylord. Cellular automaton model for random walkers. *Physical Review Letters*, 77(9):1675, 1996.
- [5] Victor J Blue and Jeffrey L Adler. Cellular automata microsimulation of bidirectional pedestrian flows. *Transportation Research Record*, 1678(1):135–141, 1999.
- [6] Felix Dietrich, Gerta Köster, Michael Seitz, and Isabella von Sivers. Bridging the gap: From cellular automata to differential equation models for pedestrian dynamics. *Journal of Computational Science*, 5(5):841–846, 2014.
- [7] Bouchra Aylaj, Nicola Bellomo, Livio Gibelli, and Alessandro Reali. A unified multiscale vision of behavioral crowds. *Mathematical Models and Methods in Applied Sciences*, 30(01):1–22, 2020.
- [8] Nicola Bellomo and Christian Dogbe. On the modelling crowd dynamics from scaling to hyperbolic macroscopic models. *Mathematical Models and Methods in Applied Sciences*, 18(supp01):1317–1345, 2008.
- [9] Emiliiano Cristiani, Benedetto Piccoli, and Andrea Tosin. *Multiscale modeling of pedestrian dynamics*, volume 12. Springer, 2014.

- [10] Nicola Bellomo and Abdelghani Bellouquid. On multiscale models of pedestrian crowds from mesoscopic to macroscopic. *Commun. Math. Sci.*, 13(7):1649–1664, 2015.
- [11] Nicola Bellomo and Christian Dogb  . On the modeling of traffic and crowds: A survey of models, speculations, and perspectives. *SIAM Review*, 53(3):409–463, 2011.
- [12] Christian Dogbe. On the modelling of crowd dynamics by generalized kinetic models. *Journal of Mathematical Analysis and Applications*, 387(2):512–532, 2012.
- [13] Daewa Kim and Annalisa Quaini. A kinetic theory approach to model pedestrian dynamics in bounded domains with obstacles. *arXiv preprint arXiv:1901.07620*, 2019.
- [14] Serge Hoogendoorn and Piet HL Bovy. Gas-kinetic modeling and simulation of pedestrian flows. *Transportation Research Record*, 1710(1):28–36, 2000.
- [15] Bouchra Aylaj, Nicola Bellomo, Livio Gibelli, and Dami  n Knopoff. *Crowd dynamics by kinetic theory modeling: complexity, modeling, simulations, and safety*. Morgan & Claypool Publishers, 2020.
- [16] Nicola Bellomo, Jie Liao, Annalisa Quaini, Lucia Russo, and Constantinos Siettos. Human behavioral crowds review, critical analysis and research perspectives. *Mathematical Models and Methods in Applied Sciences*, 33(08):1611–1659, 2023.
- [17] Roger L. Hughes. A continuum theory for the flow of pedestrians. *Transportation Research Part B: Methodological*, 36(6):507–535, 2002.
- [18] Rinaldo M Colombo and Nikolay Pogodaev. Nonlocal crowd dynamics models for several populations. *Journal of Nonlinear Science*, 22(4):511–547, 2012.
- [19] Giacomo Albi, Emiliano Cristiani, Lorenzo Pareschi, and Daniele Peri. Mathematical models and methods for crowd dynamics control. *Crowd Dynamics, Volume 2: Theory, Models, and Applications*, pages 159–197, 2020.
- [20] Ilias Panagiotopoulos, Jens Starke, and Wolfram Just. Control of collective human behavior: Social dynamics beyond modeling. *Physical Review Research*, 4(4):043190, 2022.
- [21] Seungjoon Lee, Yorgos M Psarella, Constantinos I Siettos, and Ioannis G Kevrekidis. Learning black-and gray-box chemotactic pdes/closures from agent based monte carlo simulation data. *Journal of Mathematical Biology*, 87(1):15, 2023.
- [22] Yorgos M. Psarella, Sangwoo Lee, Tapomoy Bhattacharjee, et al. Data-driven discovery of chemotactic migration of bacteria via coordinate-invariant machine learning. *BMC Bioinformatics*, 25(1):337, 2024.
- [23] Gianluca Fabiani, Nikolaos Evangelou, Tianqi Cui, Juan M Bello-Rivas, Cristina P Martin-Linares, Constantinos Siettos, and Ioannis G Kevrekidis. Task-oriented machine learning surrogates for tipping points of agent-based models. *Nature Communications*, 15(1):4117, 2024.
- [24] Xiao Song, Daolin Han, Jinghan Sun, and Zenghui Zhang. A data-driven neural network approach to simulate pedestrian movement. *Physica A: Statistical Mechanics and its Applications*, 509:827–844, 2018.
- [25] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016.
- [26] Yingfei Yu, Wei Xiang, and Xiaogang Jin. Multi-level crowd simulation using social lstm. *Computer Animation and Virtual Worlds*, 34(3-4):2180, 2023.
- [27] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2255–2264, 2018.
- [28] Koen Minartz, Fleur Hendriks, Simon Martinus Koop, Alessandro Corbetta, and Vlado Menkovski. Understanding complex crowd dynamics with generative neural simulators. *arXiv preprint arXiv:2412.01491*, 2024.
- [29] Nishant Nikhil and Brendan Tran Morris. Convolutional neural network for trajectory prediction. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018.
- [30] Gaurav Tripathi, Kuldeep Singh, and Dinesh Kumar Vishwakarma. Convolutional neural networks for crowd behaviour analysis: a survey. *The Visual Computer*, 35:753–776, 2019.
- [31] Simone Zamboni, Zekarias Tilahun Kefato, Sarunas Girdzijauskas, Christoffer Nor  n, and Laura Dal Col. Pedestrian trajectory prediction with convolutional neural networks. *Pattern Recognition*, 121:108–252, 2022.
- [32] Zhenzhen Yao, Guijuan Zhang, Dianjie Lu, and Hong Liu. Data-driven crowd evacuation: A reinforcement learning method. *Neurocomputing*, 366:314–327, 2019.

- [33] Kei Yoshida and William H. Warren. Visual influence networks in walking crowds. *Journal of Vision*, 23(9):5175–5175, 2023.
- [34] Shenshi Huang, Ruichao Wei, Liping Lian, Siuming Lo, and Shouxiang Lu. Review of the application of neural network approaches in pedestrian dynamics studies. *Heliyon*, 10(10), 2024.
- [35] C William Gear, James M Hyman, Panagiotis G Kevrekidis, Ioannis G Kevrekidis, Olof Runborg, and Constantinos Theodoropoulos. Equation-free, coarse-grained multiscale computation: Enabling microscopic simulators to perform system-level analysis. *Communications in Mathematical Sciences*, 1:715–762, 2003.
- [36] Dimitrios G. Patsatzis, Lucia Russo, Ioannis G. Kevrekidis, and Constantinos Siettos. Data-driven control of agent-based models: An equation/variable-free machine learning approach. *Journal of Computational Physics*, 478:111953, 2023.
- [37] Ilias Panagiotopoulos, Jens Starke, Jan Sieber, and Wolfram Just. Continuation with noninvasive control schemes: Revealing unstable states in a pedestrian evacuation scenario. *SIAM Journal on Applied Dynamical Systems*, 22(1):1–36, 2023.
- [38] T. Chin, J. Ruth, C. Sanford, and et al. Enabling equation-free modeling via diffusion maps. *Journal of Dynamic Differential Equations*, 36(Suppl 1):415–434, 2024.
- [39] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Numerical gaussian processes for time-dependent and nonlinear partial differential equations. *SIAM Journal on Scientific Computing*, 40(1):A172–A198, 2018.
- [40] Somdatta Goswami, Aniruddha Bora, Yue Yu, and George Em Karniadakis. Physics-informed deep neural operator networks. In *Machine Learning in Modeling and Simulation: Methods and Applications*, pages 219–254. Springer, 2023.
- [41] Gianluca Fabiani, Hannes Vandecasteele, Somdatta Goswami, Constantinos Siettos, and Ioannis G Kevrekidis. Enabling local neural operators to perform equation-free system-level analysis. *arXiv preprint arXiv:2505.02308*, 2025.
- [42] Seungjoon Lee, Mahdi Kooshbaghi, Konstantinos Spiliotis, Constantinos I. Siettos, and Ioannis G. Kevrekidis. Coarse-scale pdes from fine-scale observations via machine learning. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(1):013141, 2020.
- [43] Evangelos Galaris, Gianluca Fabiani, Ioannis Gallos, Ioannis Kevrekidis, and Constantinos Siettos. Numerical bifurcation analysis of pdes from lattice boltzmann model simulations: A parsimonious machine learning approach. *Journal of Scientific Computing*, 92(2):1–30, 2022.
- [44] Ameya D Jagtap, Ehsan Kharazmi, and George Em Karniadakis. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365:113028, 2020.
- [45] Seungjoon Lee, Yorgos M. Psarellis, Constantinos I. Siettos, and Ioannis G. Kevrekidis. Learning black- and gray-box chemotactic pdes/closures from agent based monte carlo simulation data. *Journal of Mathematical Biology*, 87(1):15, 2023.
- [46] Yongqing Guo, Hai Zou, Fulu Wei, Qingyin Li, Dong Guo, and Jahongir Pirov. Analysis of pedestrian second crossing behavior based on physics-informed neural networks. *Scientific Reports*, 14(1):21–278, 2024.
- [47] Guozhen Zhang, Zihan Yu, Depeng Jin, and Yong Li. Physics-infused machine learning for crowd simulation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2439–2449, 2022.
- [48] Guofei Pang, Lu Lu, and George Em Karniadakis. fpinns: Fractional physics-informed neural networks. *SIAM Journal on Scientific Computing*, 41(4):A2603–A2626, 2019.
- [49] Nazanin Ahmadi Daryakenari, Mario De Florio, Khemraj Shukla, and George Em Karniadakis. Ai-aristotle: A physics-informed framework for systems biology gray-box identification. *PLOS Computational Biology*, 20(3):1011916, 2024.
- [50] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.
- [51] Xiaofeng Li, Wei Wang, Qiang Liu, et al. Deep neural networks for simulating complex dynamical systems: A framework combining neural operators and recurrent neural networks. *arXiv preprint arXiv:2307.07331*, 2023.
- [52] Kamyar Azizzadenesheli, Nikola Kovachki, Zongyi Li, Miguel Liu-Schiavini, Jean Kossaifi, and Anima Anandkumar. Neural operators for accelerating scientific simulations and design. *Nature Reviews Physics*, 6(5):320–328, 2024.

- [53] Emanuele Zappala, Antonio Henrique de Oliveira Fonseca, Josue Ortega Caro, Andrew Henry Moberly, Michael James Higley, Jessica Cardin, and David van Dijk. Learning integral operators via neural integral equations. *Nature Machine Intelligence*, 6(9):1046–1062, 2024.
- [54] Ahmad Peyvan, Vivek Oommen, Ameya D Jagtap, and George Em Karniadakis. Riemannonets: Interpretable neural operators for riemann problems. *Computer Methods in Applied Mechanics and Engineering*, 426:116996, 2024.
- [55] Dimitrios G Patsatzis, Mario di Bernardo, Lucia Russo, and Constantinos Siettos. Gorinns: Godunov-riemann informed neural networks for learning hyperbolic conservation laws. *Journal of Computational Physics*, 534:114002, 2025.
- [56] Panagiotis G Papaioannou, Ronen Talmon, Ioannis G Kevrekidis, and Constantinos Siettos. Time-series forecasting using manifold learning, radial basis function interpolation, and geometric harmonics. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 32(8):083113, 2022.
- [57] Ioannis K Gallos, Daniel Lehmberg, Felix Dietrich, and Constantinos Siettos. Data-driven modelling of brain activity using neural networks, diffusion maps, and the koopman operator. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 34(1), 2024.
- [58] Alessandro Della Pia, Dimitrios G Patsatzis, Lucia Russo, and Constantinos Siettos. Learning the latent dynamics of fluid flows from high-fidelity numerical simulations using parsimonious diffusion maps. *Physics of Fluids*, 36(10), 2024.
- [59] Floris Takens. Detecting strange attractors in turbulence. In *Dynamical Systems and Turbulence, Warwick 1980: proceedings of a symposium held at the University of Warwick 1979/80*, pages 366–381. Springer, 2006.
- [60] K Krischer, R Rico-Martinez, IG Kevrekidis, HH Rotermund, G Ertl, and JL Hudson. Model identification of a spatiotemporally varying catalytic reaction. *AIChE Journal*, 39(1):89–98, 1993.
- [61] Stanislav Y Shvartsman and IG Kevrekidis. Low-dimensional approximation and control of periodic solutions in spatially extended systems. *Physical Review E*, 58(1):361, 1998.
- [62] Felix P Kemeth, Tom Bertalan, Thomas Thiem, Felix Dietrich, Sung Joon Moon, Carlo R Laing, and Ioannis G. Kevrekidis. Learning emergent partial differential equations in a learned emergent space. *Nature Communications*, 13(1):3318, 2022.
- [63] Joar Axås and George Haller. Model reduction for nonlinearizable dynamics via delay-embedded spectral submanifolds. *Nonlinear Dynamics*, 111(24):22079–22099, 2023.
- [64] Dirk Helbing, Illés Farkas, and Tamás Vicsek. Simulating dynamical features of escape panic. *Nature*, 407:487–490, Sep 2000.
- [65] Stamatina Th Rassia and Constantinos I Siettos. Escape dynamics in office buildings: using molecular dynamics to quantify the impact of certain aspects of human behavior during emergency evacuation. *Environmental Modeling & Assessment*, 15:411–418, 2010.
- [66] Nicola Bellomo and Livio Gibelli. Toward a mathematical theory of behavioral-social dynamics for pedestrian crowds. *Mathematical Models and Methods in Applied Sciences*, 25(13):2417–2437, 2015.
- [67] Nicola Bellomo, D Clarke, L Gibelli, P Townsend, and BJ Vreugdenhil. Human behaviours in evacuation crowd dynamics: From modelling to “big data” toward crisis management. *Physics of Life Reviews*, 18:1–21, 2016.
- [68] Giacomo Albi, Mattia Bongini, Emiliano Cristiani, and Dante Kalise. Invisible control of self-organizing agents leaving unknown environments. *SIAM Journal on Applied Mathematics*, 76(4):1683–1710, 2016.
- [69] Milad Haghani, Emiliano Cristiani, Nikolai WF Bode, Maik Boltes, and Alessandro Corbetta. Panic, irrationality, and herding: three ambiguous terms in crowd dynamics research. *Journal of Advanced Transportation*, 2019(1):9267643, 2019.
- [70] Nicola Bellomo, Seung-Yeal Ha, and Nisrine Outada. Towards a mathematical theory of behavioral swarms. *ESAIM: Control, Optimisation and Calculus of Variations*, 26:125, 2020.
- [71] Ronald R Coifman, Stephane Lafon, Ann B Lee, Mauro Maggioni, Boaz Nadler, Frederick Warner, and Steven W Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proceedings of the National Academy of Sciences*, 102(21):7426–7431, 2005.
- [72] Ronald R. Coifman and Stéphane Lafon. Geometric harmonics: A novel tool for multiscale out-of-sample extension of empirical functions. *Applied and Computational Harmonic Analysis*, 21(1):31–52, 2006.
- [73] Ronald R. Coifman and Stéphane Lafon. Geometric harmonics: A novel tool for multiscale out-of-sample extension of empirical functions. *Applied and Computational Harmonic Analysis*, 21(1):31–52, 2006. Special Issue: Diffusion Maps and Wavelets.

- [74] Felix Dietrich, Ioannis G. Kevrekidis, and Ronald R. Coifman. Manifold learning via double diffusion maps. *SIAM Journal on Applied Dynamical Systems*, 14(1):1–38, 2015.
- [75] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.
- [76] Diederik P Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.
- [77] Z. I. Botev, J. F. Grotowski, and D. P. Kroese. Kernel density estimation via diffusion. *The Annals of Statistics*, 38(5):2916–2957, 2010.
- [78] Lawrence Sirovich. Turbulence and the dynamics of coherent structures. I-III. *Quarterly of Applied Mathematics*, 45(3):561–590, 1987.
- [79] Luc Devroye. The equivalence of weak, strong and complete convergence in L^1 for kernel density estimates. *The Annals of Statistics*, pages 896–904, 1983.
- [80] Nadine Aubry, Wen-Yu Lian, and Edriss S Titi. Preserving symmetries in the proper orthogonal decomposition. *SIAM Journal on Scientific Computing*, 14(2):483–505, 1993.
- [81] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.
- [82] Mahmoud H. AbdElbaky, Mohammed H. Yacoub, Wafaa S. Sayed, and Lobna A. Said. High-performance fpga-accelerated lstm neural network for chaotic time series prediction. *AEU - International Journal of Electronics and Communications*, 199:155845, 2025.
- [83] Changwei Liu, Hao Ren, Guoqiang Li, Haojie Ren, Xiaojun Liang, Chunhua Yang, and Weihua Gui. Singular value decomposition-based lightweight lstm for time series forecasting. *Future Generation Computer Systems*, 174:107910, 2025.
- [84] Rohitash Chandra, Ayush Jain, and Divyanshu Singh Chauhan. Deep learning via lstm models for covid-19 infection forecasting in india. *PLOS ONE*, 17:1–28, 01 2022.
- [85] Kyongmin Yeo and Igor Melnyk. Deep learning algorithm for data-driven simulation of noisy dynamical system. *Journal of Computational Physics*, 376:1212–1231, 2019.
- [86] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.
- [87] Chris Brooks. *Introductory Econometrics for Finance*. Cambridge University Press, 4 edition, 2019.
- [88] Helmut Lütkepohl. *New Introduction to Multiple Time Series Analysis*. Springer, 2005.
- [89] Hirotugu Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.
- [90] Gideon Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.
- [91] David A. Dickey and Wayne A. Fuller. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, 74(366):427–431, 1979.
- [92] The MathWorks Inc. Matlab, 2024.
- [93] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.
- [94] Daniel Dylewsky, Eurika Kaiser, Steven L Brunton, and J Nathan Kutz. Principal component trajectories for modeling spectrally continuous dynamics as forced linear systems. *Physical Review E*, 105(1):015312, 2022.
- [95] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.
- [96] M. P. Wand and M. C. Jones. *Kernel Smoothing*. Chapman and Hall, 1995.
- [97] V. A. Epanechnikov. Non-parametric estimation of a multivariate probability density. *Theory of Probability and Its Applications*, 14(1):153–158, 1969.

A Social Force Model

Here, we provide the detailed expressions of the virtual forces included in the SFM Eq. (31) and the set of parameters selected for the microscopic simulations.

As a reminder, we consider the SFM including a goal-directed, a social, and a repulsive force term (the random force is neglected), reading:

$$m_i \frac{d\dot{\mathbf{x}}_i}{dt} = \underbrace{-\frac{1}{\tau_i}(v_{0,i}\mathbf{e}_i - \mathbf{v}_i)}_{\text{goal-directed force}} + \underbrace{\sum_{j \neq i} \mathbf{F}_{i,j}(\mathbf{x}_i, \mathbf{x}_j)}_{\text{social force}} + \underbrace{\sum_{\text{obstacles}} \mathbf{F}_{i,B}(\mathbf{x}_i)}_{\text{repulsive force}}, \quad (\text{A.1})$$

where m_i is the mass of the i -th pedestrian, $v_{0,i}$ is their maximum walking speed, \mathbf{e}_i represents the desired direction (serving as a simplistic auxiliary/behavioral variable), and τ_i is a characteristic relaxation time. The social force term in Eq. (A.1) introduces a short-range repulsive interaction between pedestrians, to prevent collisions. The social force between pedestrians i and j is given by:

$$\mathbf{F}_{ij}(\mathbf{x}_i, \mathbf{x}_j) = A_i \exp\left(-\frac{r_{ij} - d_{ij}}{B_i}\right) \mathbf{n}_{ij} + k g(r_{ij} - d_{ij}) \mathbf{n}_{ij} + \kappa g(r_{ij} - d_{ij}) \Delta v_{ij}^{\text{tang}} \mathbf{t}_{ij}, \quad (\text{A.2})$$

where $A_i \in \mathbb{R}^+$ is the strength of the repulsive interaction, $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$ is the distance between pedestrians, $r_{i,j} \in \mathbb{R}$ is the sum of their radii (minimal comfortable distance) [64], $B \in \mathbb{R}^+$ is the parameter controlling the spatial decay rate of the repulsive force, $\mathbf{n}_{ij} = (n_{i,j}^1, n_{i,j}^2) = \frac{\mathbf{x}_i - \mathbf{x}_j}{d_{ij}}$ is the unit vector pointing from pedestrian j to pedestrian i , $g(x)$ is a function which ensures that physical forces are only active when pedestrians overlap, i.e., when $(d_{i,j} < r_{i,j})$. Moreover, the constant $k > 0$ governs the strength of the linear contact (body) force, which causes repulsion between pedestrians when they are in contact. The third term is a sliding friction term that involves a coefficient $\kappa > 0$, which scales with the tangential component of the relative velocity $\Delta v_{ij}^{\text{tang}}$; this component is projected onto the unit tangential vector $\mathbf{t}_{i,j} = (-n_{i,j}^2, n_{i,j}^1)$. The repulsive force in Eq. (A.1), accounting for the repulsion that a pedestrian might experience due to static objects B , follows a similar, to Eq. (A.2), form, reading:

$$\mathbf{F}_{iB}(\mathbf{x}_i) = C_i \exp\left(-\frac{d_{iB}}{D_i}\right) \mathbf{n}_{iB} + k g(r_i - d_{iB}) \mathbf{n}_{iB}. \quad (\text{A.3})$$

Here, $C_i \in \mathbb{R}^+$ is the strength of the repulsive interaction against a wall or an obstacle and D_i is its corresponding decay parameter. Unlike pedestrian–pedestrian interactions, no tangential (sliding) friction is typically included for static obstacles, as they do not exert dynamic resistance.

For our simulations, the model parameters were calibrated based on established values from previous studies [1, 64]. These works describe pedestrians with a standard average mass of 80kg, for which a relaxation time $\tau = 0.5$ s provides adequate behavior under normal conditions. The complete set of parameters governing repulsive potentials, including agent-agent and agent-obstacle interactions, along with the simulation environment specifications, are detailed in Table A.1. This parameter selection, grounded in empirical crowd studies, enables the simulation of diverse scenarios, including the present case study while maintaining consistency with validated bottleneck evacuations scenarios, and even though the parameters should vary individually, in the literature the parameters are often chosen equally for all pedestrians to maintain a reasonable number of parameters calibration [64].

Table A.1: Simulation parameters.

Simulation Parameters	Value
Number of Pedestrians	100
Corridor length (L_x)	48 m
Corridor width (L_y)	12 m
Time Step δ_t	2.5×10^{-2} s
Simulation time	275 s
Repulsion coefficient (k)	$1.2 \times 10^5 \frac{\text{kg}\cdot\text{m}}{\text{s}^2}$
Sliding friction coefficient (κ)	$2.4 \times 10^5 \frac{\text{kg}\cdot\text{m}}{\text{s}^2}$
Interaction strength (A_i)	2×10^3 N
Interaction range (B_i)	0.08 m
Interaction strength (C_i)	2×10^3 N
Interaction range (D_i)	0.08 m
Mass	80 kg
Relaxation time (τ)	0.5 s
Pedestrian radius	0.2 m
Initial velocity in x -direction (v_x)	0 m/s
Initial velocity in y -direction (v_y)	0 m/s
Obstacle width	3.6 m
Obstacle height	3.6 m
x -coordinate of center	25.8 m
y -coordinate of center	1.8 m

B Initial Conditions of Microscopic Distributions

Here, we describe the initial conditions that we considered in the SFM to construct the datasets, upon which the proposed framework is trained and tested. We considered zero initial velocities for all pedestrians, while we used the following distributions for initializing the position $x_i = (x_i, y_i) \in \Omega$ of the i -th pedestrian.

- Positions are sampled from a two dimensional normal distribution with mean (μ_x, μ_y) and standard deviations (σ_x, σ_y) :

$$p(x_i, y_i) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left(-\frac{(x_i - \mu_x)^2}{2\sigma_x^2} - \frac{(y_i - \mu_y)^2}{2\sigma_y^2}\right), \quad (\text{B.1})$$

- Uniform density distribution: Positions are sampled uniformly over the domain Ω

$$p(x_i, y_i) = \mathcal{U}([x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]). \quad (\text{B.2})$$

- Cosine distribution;

$$p(x_i, y_i) = \frac{1}{\pi\sigma_x\sigma_y} \cos\left(\frac{x_i - \mu_x}{\sigma_x}\right) \cos\left(\frac{y_i - \mu_y}{\sigma_y}\right). \quad (\text{B.3})$$

- Piecewise linear density distribution:

To initialize particle positions in a rectangular domain $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$, we assume a uniform distribution in x and a non-uniform, piecewise linear distribution in y . This approach allows control over the particle concentration along the vertical axis.

Particles are sampled uniformly in the x -direction over the interval $[x_{\min}, x_{\max}]$. In the y -direction, particle positions are sampled from a custom piecewise linear distribution designed to increase density near the top and bottom boundaries. Specifically, we use a triangular distribution defined over the interval $[0, 12]$ by

$$p(y_i) = \begin{cases} \frac{2}{12} \left(1 - \frac{y_i}{6}\right), & 0 \leq y_i \leq 6, \\ \frac{2}{12} \left(\frac{y_i - 6}{6}\right), & 6 < y_i \leq 12, \end{cases} \quad (\text{B.4})$$

which has peaks at $y = 0$ and $y = 12$, and a minimum at $y = 6$.

To sample y -coordinates from this distribution, we use inverse transform sampling. First, a uniform random number $u \sim \mathcal{U}(0, 1)$ is generated. If $u \leq 0.5$, the corresponding y value is computed by solving $u = \frac{y}{12}(2 - \frac{y}{6})$, yielding

$$y = 6 \left(1 - \sqrt{1 - 2u}\right). \quad (\text{B.5})$$

If $u > 0.5$, we solve $u = \frac{1}{2} + \frac{(y-6)^2}{72}$, resulting in

$$y = 6 + \sqrt{72(u - 0.5)}. \quad (\text{B.6})$$

This method produces particle positions that are uniformly distributed along the horizontal axis and more concentrated near the top and bottom edges along the vertical axis.

For generating the training/testing data set, we used a wide range of initial conditions, based on the described distributions; in particular, for the gaussian sampling, the means and standard deviations were selected relative to the domain size to ensure spatial coverage while avoiding boundary artifacts. Specifically, the means were placed at the geometric center or fractional positions within the domain (e.g., 0.5 or 0.25 of the domain width), while standard deviations were set as fractions of the domain dimensions (e.g., one-fifth or one-half of the width or height). This ensures that the pedestrian positions lie within the domain, while allowing for spatial clustering or multimodal patterns. The parameters were chosen heuristically to reflect plausible crowd configurations and to introduce variety in the training and testing datasets. the ones used to construct the training are shown in Table B.1, while the ones for the testing data set are shown in Table B.2.

Table B.1: Microscopic distribution configurations and parameters for training the LSTM and MVAR models.

Distribution Type	Parameters
Data set 1: Gaussian	$\mu_x = 7.5, \mu_y = 5.0,$ $\sigma_x = 2.2, \sigma_y = 2.0$
Data set 2: Uniform	$\mathcal{U}([2, 15] \times [3, 6])$
Data set 3: Gaussian	$\mu_x = 8.4, \mu_y = 4.0,$ $\sigma_x = 1.8, \sigma_y = 1.3$
Data set 4: Gaussian	$\mu_x = 12.5, \mu_y = 4.0,$ $\sigma_x = 7, \sigma_y = 1.1$
Data set 5: Double Gaussian	$\mu_{x1} = 17.3, \mu_{x2} = 13.6,$ $\sigma_{x1} = \sigma_{x2} = 0.7,$ $\sigma_y = 1.2; \mu_y = 7.0$
Data set 6: Double Gaussian	$\mu_{x1} = 19.7, \mu_{x2} = 13.7,$ $\sigma_{x1} = \sigma_{x2} = 0.62,$ $\sigma_y = 4.0; \mu_y = 2.33$
Data set 7: Double Gaussian	$\mu_{x1} = 32.2, \mu_{x2} = 27.75,$ $\sigma_{x1} = \sigma_{x2} = 0.35,$ $\sigma_y = 4.0; \mu_y = 1.0$
Data set 8: Piecewise Linear	$x_i \sim \mathcal{U}(30, 47), y_i \sim p(y_i)$ in Eq. (B.4)
Data set 9: Piecewise Linear	$x_i \sim \mathcal{U}(15, 35), y_i \sim p(y_i)$ in Eq. (B.4)
Data set 10: Cosine	$\mu_x = 25, \mu_y = 3.5,$ $L_x = 10, L_y = 2.5$

Table B.2: Microscopic distribution configurations and parameters for testing the LSTM and MVAR models.

Configuration	Parameters
Dataset 1: Double Gaussian	$\mu_{x1} = 41.2, \mu_{x2} = 33.7,$ $\sigma_{x1} = \sigma_{x2} = 0.5,$ $\sigma_y = 2.0, \mu_y = 9.0$
Dataset 2: Double Gaussian	$\mu_{x1} = 11.7, \mu_{x2} = 5.2,$ $\sigma_{x1} = \sigma_{x2} = 0.4,$ $\sigma_y = 1.0, \mu_y = 9.0$
Dataset 3: Double Gaussian	$\mu_{x1} = 42.5, \mu_{x2} = 37.5,$ $\sigma_{x1} = \sigma_{x2} = 0.3,$ $\sigma_y = 2.0, \mu_y = 5.0$
Dataset 4: Gaussian	$\mu_x = 36.0, \mu_y = 4.0,$ $\sigma_x = 2.4, \sigma_y = 1.5$
Dataset 5: Gaussian	$\mu_x = 23.0, \mu_y = 8.5,$ $\sigma_x = 2.0, \sigma_y = 2.5$
Dataset 6: Gaussian	$\mu_x = 7.85, \mu_y = 8.0,$ $\sigma_x = 4.3, \sigma_y = 2.0$
Dataset 7: Cosine	$\mu_x = 40, \mu_y = 8.0,$ $L_x = 5, L_y = 2.0$
Dataset 8: Uniform	$\mathcal{U}([10, 20] \times [4, 9])$
Dataset 9: Piecewise Linear	$x_i \sim \mathcal{U}(30, 44), y_i \sim p(y_i)$ in Eq. (B.4)
Dataset 10: Cosine	$\mu_x = 39, \mu_y = 6.5,$ $L_x = 7, L_y = 2.5$

C Kernel Density Estimation

Here, we provide additional details for the first step of the proposed framework, regarding the extraction of continuous density profiles from discrete pedestrians' positions. As already discussed in Section 2.2.1, the Kernel Density Estimation (KDE) is used for this task, following the general form in Eq. (6). Here, we elaborate on the kernel selection, bandwidth tuning, and adjustments for boundaries and obstacles.

The choice of kernel function directly influences the smoothness and continuity of the density estimate. Common options include the uniform, triangular, Epanechnikov, and Gaussian kernels [95, 96]. The Epanechnikov kernel is theoretically optimal for minimizing mean integrated squared error (MISE) [97], but its bounded support can limit flexibility in practice. In contrast, the Gaussian kernel is widely preferred due to its smoothness, infinite support, and analytical tractability. Its multivariate form is given by:

$$K_h(\mathbf{x} - \mathbf{x}_i(t)) = \frac{1}{2\pi \det(\mathbf{H})^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}_i(t))^\top \mathbf{H}^{-1}(\mathbf{x} - \mathbf{x}_i(t))\right), \quad (\text{C.1})$$

where \mathbf{H} is the symmetric positive-definite bandwidth matrix, $\mathbf{x} \in \mathbb{R}^2$ is the spatial domain and $\mathbf{x}_i(t) \in \mathbb{R}^2$ is the position of the i -th pedestrian.

The choice of the bandwidth matrix, which in this case takes the form $\mathbf{H} = \text{diag}(h_x, h_y)$, plays a central role in the KDE estimate. Small values of h_x/h_y result in undersmoothing and high variance, while large values can over-smooth the density and obscure relevant features. Silverman's rule of thumb [95], originally formulated for univariate distributions, extends naturally to multivariate settings, indicating an optimal bandwidth:

$$h_i = \left(\frac{4}{N(d+2)}\right)^{1/(d+4)} \sigma_i \quad (i = x, y). \quad (\text{C.2})$$

where N is the number of data points, d is the number of spatial dimensions, in this case $d = 2$ and σ_i is the standard deviations of the pedestrian positions. However, in our case, Silverman's rule resulted in wide bandwidths, which did not allow us to consider sufficient detail in the density profiles. We thus tuned h_x/h_y to the values reported in Section 2.2.1.

The standard symmetric kernels are known to introduce bias near domain boundaries, when handling cases where *periodic boundary conditions* are imposed, such as in the x -direction of pedestrian flow in our configuration. These artifacts arise because the kernel's support extends beyond the domain, failing to reflect the periodic structure of the system. To mitigate these artifacts, we define an extended support for the KDE in the x -direction. Let the physical domain boundaries be $x = a$ and $x = b$, and introduce a small displacement $\Delta x = \frac{b-a}{n}$, where n defines the extension of the domain along x -direction; here, we tuned $n = 5$. We then define the extended domain as:

$$\Omega^{\text{ext}} = [a - \Delta x, b + \Delta x] \times [c, d], \quad (\text{C.3})$$

while the support in the y -direction remains unbounded, i.e., $\Omega_y = (-\infty, \infty)$. To enforce continuity and reduce boundary-induced artifacts in the x -direction, we further augment the observed positions with “ghost” particles near the domain edges. Let $\epsilon = \Delta x$ define a threshold for proximity to the boundary. For each position $\mathbf{x}_i(t) = (x_i(t), y_i(t))$ such that $x_i(t) \leq a + \epsilon$, we introduce a “mirrored” pedestrians at

$$\mathbf{x}_i^{\text{aug}1}(t) = (x_i(t) - (b - a), y_i(t)).$$

Similarly, for every $\mathbf{x}_i(t)$ such that $x_i(t) \geq b - \epsilon$, we introduce a “mirrored” pedestrians at

$$\mathbf{x}_i^{\text{aug}2}(t) = (x_i(t) + (b - a), y_i(t)).$$

The resulting augmented set of pedestrian positions used for KDE is then given by

$$\{\mathbf{x}'_i(t)\} = \{\mathbf{x}_i(t)\} \cup \{\mathbf{x}_i^{\text{aug}1}(t)\} \cup \{\mathbf{x}_i^{\text{aug}2}(t)\}. \quad (\text{C.4})$$

A pseudo-code for the estimation of the density in $\Omega \subset \mathbb{R}^2$ is given in Algorithm C.1.

Algorithm C.1 Kernel Density Estimation (KDE) for extracting density fields from pedestrian positions.

Require: Number of pedestrians N , computational domain $\Omega = [a, b] \times [c, d]$, spatial discretization n_x/n_y , obstacle subdomain $\Gamma = [a_o, b_o] \times [c_o, d_o] \subset \Omega$ and pedestrian positions $\{\mathbf{x}_i(t)\}_{i=1}^N \in \Omega \setminus \Gamma$

- 1: Set bandwidth $\mathbf{H} = \text{diag}(h_x, h_y)$.
- 2: Set augmentation on x -direction $\Delta x = (b - a)/n$
- 3: Define extended support Ω^{ext} to handle periodic boundary conditions ▷ Use Eq. (C.3)
- 4: Discretize extended domain Ω^{ext} in $(n_x + 2n_x/n) \times n_y$ cells
- 5: Introduce “mirrored” pedestrians to form $\{\mathbf{x}'_i(t)\}$ ▷ Use Eq. (C.4)
- 6: Initialize zero density field $\rho((x_i, y_j), t)$ across the extended domain Ω^{ext}
- 7: **for** each pedestrian $p = 1, 2, \dots, N$ **do** ▷ Account for “mirrored” pedestrians
- 8: Compute kernel contribution $K_h(\mathbf{x} - \mathbf{x}_p(t))$ over the extended support Ω^{ext} ▷ Use Eq. (C.1)
- 9: Add contribution to density field $\rho((x_i, y_j), t)$
- 10: **end for**
- 11: Normalize $\rho((x_i, y_j), t)$ with total number of pedestrians, N and “mirrored” ones
- 12: Retrieve density $\rho((x_i, y_j), t)$ in computational domain Ω
- 13: Employ binary mask function to ensure zero density in obstacle subdomain Γ , by setting ▷ See Eq. (C.5)
- 14: $\rho((x_i, y_i), t) \leftarrow \mathcal{M}(x_i, y_i) \cdot \rho((x_i, y_i), t)$
- 15: Compute total estimated density over the entire domain $S(t) = \sum_{i,j=1}^{n_x, n_y} \rho((x_i, y_j), t) \delta x \delta y$
- return Normalized density field $\rho((x_i, y_j), t) \leftarrow \rho((x_i, y_j), t)/S(t)$

In the presence of obstacles, formally described as a subdomain $\Gamma \subset \Omega$, the standard KDE must be adjusted to ensure that the density vanishes within and along the boundaries of these regions. This is typically achieved by introducing a binary mask function $\mathcal{M}(x, y)$:

$$M(x, y) = \begin{cases} 0, & \text{if } (x, y) \in \Gamma \subset \Omega, \\ 1, & \text{otherwise.} \end{cases} \quad (\text{C.5})$$

which is then applied to the initially estimated densities to enforce zero density in Γ , implying:

$$\rho((x_i, y_j), t) = \mathcal{M}(x_i, y_j) \rho((x_i, y_j), t), \quad \forall (x_i, y_j) \in \Omega. \quad (\text{C.6})$$

D Lag window size selection via the Bayesian and the Akaike Information Criteria

As discussed in Section 3.5, to determine the optimal lag window size w for the MVAR model, we used two information-theoretic criteria: the Bayesian Information Criterion (BIC) and the Akaike Information Criterion (AIC) [88, 89, 90]. These criteria provide a balance between model fit and complexity, and are used to avoid underfitting or overfitting the temporal dynamics. Both tests work under the assumption of data stationarity. The BIC is defined as:

$$\text{BIC}(w) = -2 \ln(\mathcal{L}(w)) + k \ln(n_t), \quad (\text{D.1})$$

where \mathcal{L} is the maximized likelihood of the model. For an MVAR model with d latent dimensions and w order of the model, the total number of parameters is then $k = d^2 \cdot w$. The AIC, which provides a different trade-off between model complexity and fit, is given by:

$$\text{AIC}(w) = -2 \ln(\mathcal{L}(w)) + 2k. \quad (\text{D.2})$$

While AIC tends to favor more complex models and is better suited for short-term predictive accuracy in smaller datasets, BIC imposes a stronger penalty on complexity, making it preferable for larger sample sizes. The log-likelihood $\ln(\mathcal{L}(w))$ is computed under the assumption of Gaussian noise, and is derived from the residual covariance matrix of the MVAR model.

Finally, to select the optimal lag window w^* , we evaluated the BIC and AIC for all candidate lags $w \in \{1, 2, \dots, w_{\max}\}$. The chosen lag w^* corresponds to the value of w that yielded the smallest BIC or AIC in this range, i.e.:

$$w_{\text{BIC}}^* = \arg \min_{w \in \{1, \dots, w_{\max}\}} \text{BIC}(w), \quad w_{\text{AIC}}^* = \arg \min_{w \in \{1, \dots, w_{\max}\}} \text{AIC}(w). \quad (\text{D.3})$$

E Comparison of the “ground truth” density fields with the predicted ones, using MVAR(9), LSTM(4) and LSTM(9) models

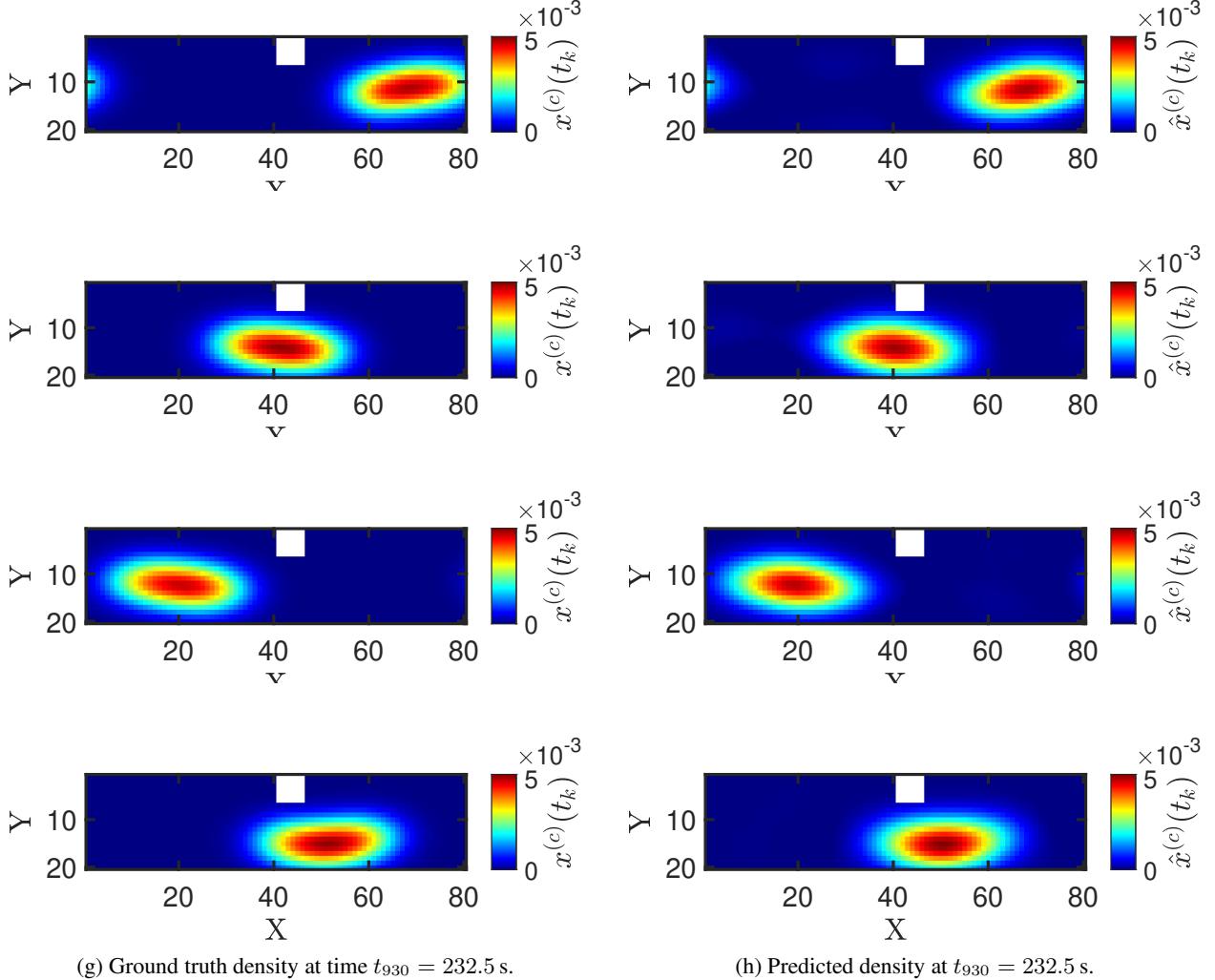


Figure E.1: Comparison of the “ground truth” $x^{(c)}(t_k)$ and the predicted $\hat{x}^{(c)}(t_k)$ via the closed-loop time-stepper in Eq. (35) using the MVAR(9) model, normalized densities for an unseen case of the testing set; initialization at the 6th row of Table B.2. Panels a,c,e, and g show the “ground truth” density distribution in Ω , while panels b,d,f, and h, show the predicted one, for the time steps $t_{250} = 62.5$ s, $t_{500} = 125$ s, $t_{780} = 195$ s, and $t_{930} = 232.5$ s, respectively.

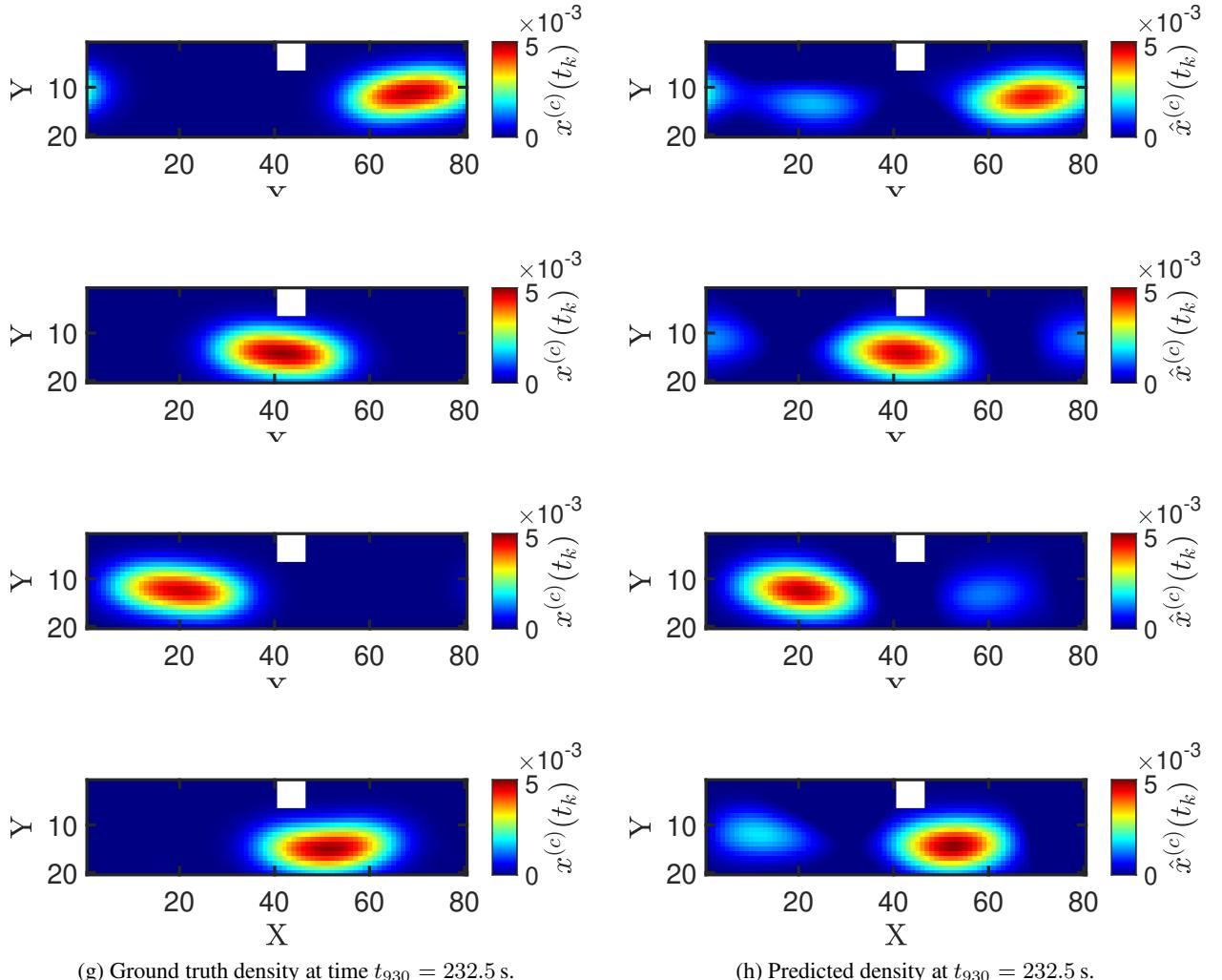


Figure E.2: Comparison of the “ground truth” $x^{(c)}(t_k)$ and the predicted $\hat{x}^{(c)}(t_k)$ via the closed-loop time-stepper in Eq. (35) using the LSTM(4) model, normalized densities for an unseen case of the testing set; initialization at the 6th row of Table B.2. Panels a,c,e, and g show the “ground truth” density distribution in Ω , while panels b,d,f, and h, show the predicted one, for the time steps $t_{250} = 62.5$ s, $t_{500} = 125$ s, $t_{780} = 195$ s, and $t_{930} = 232.5$ s, respectively.

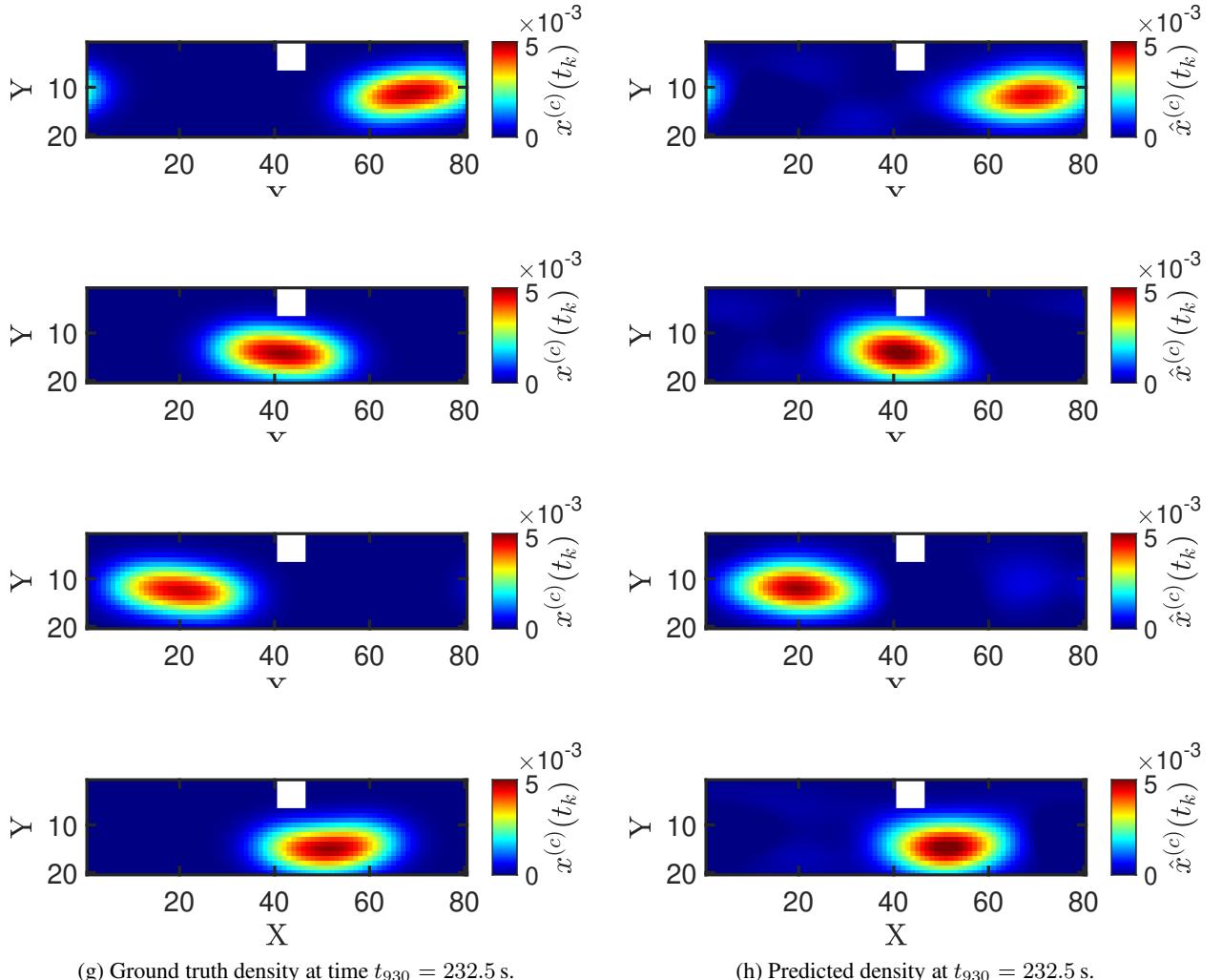


Figure E.3: Comparison of the “ground truth” $x^{(c)}(t_k)$ and the predicted $\hat{x}^{(c)}(t_k)$ via the closed-loop time-stepper in Eq. (35) using the LSTM(9) model, normalized densities for an unseen case of the testing set; initialization at the 6th row of Table B.2. Panels a,c,e, and g show the “ground truth” density distribution in Ω , while panels b,d,f, and h, show the predicted one, for the time steps $t_{250} = 62.5$ s, $t_{500} = 125$ s, $t_{780} = 195$ s, and $t_{930} = 232.5$ s, respectively.