NEURALDEM: REAL-TIME SIMULATION OF INDUSTRIAL PARTICULATE FLOWS

Benedikt Alkin[†,*,1,2] **Tobias Kronlachner**[†,*,1,3] **Samuele Papa**[†,‡,1,4,5]
Stefan Pirker[3] **Thomas Lichtenegger**[1,3] **Johannes Brandstetter**[1,2]

[1]Emmi AI GmbH [2]ELLIS Unit Linz, Institute for Machine Learning
[3]Department of Particulate Flow Modelling, JKU Linz
[4]University of Amsterdam [5]The Netherlands Cancer Institute
[†] core contributor [*] equal contribution [‡] work done during internship

ABSTRACT

Advancements in computing power have made it possible to numerically simulate large-scale fluid-mechanical and/or particulate systems, many of which are integral to core industrial processes. The discrete element method (DEM) provides one of the most accurate representations of a wide range of physical systems involving granular materials. Additionally, DEM can be integrated with grid-based computational fluid dynamics (CFD) methods, enabling the simulation of chemical processes taking place, e.g., in fluidized beds. However, DEM is computationally intensive because of the intrinsic multiscale nature of particulate systems, restricting either the duration of simulations or the number of particles that can be simulated. Towards this end, NeuralDEM presents a first end-to-end approach to replace slow and computationally demanding DEM routines with fast deep learning surrogates. NeuralDEM treats the Lagrangian discretization of DEM as an underlying continuous field, while simultaneously modeling macroscopic behavior directly as additional auxiliary fields using "multi-branch neural operators", enabling fast and scalable neural surrogates. NeuralDEM will open many new doors to advanced engineering and much faster process cycles.
Project page: https://emmi-ai.github.io/NeuralDEM/.

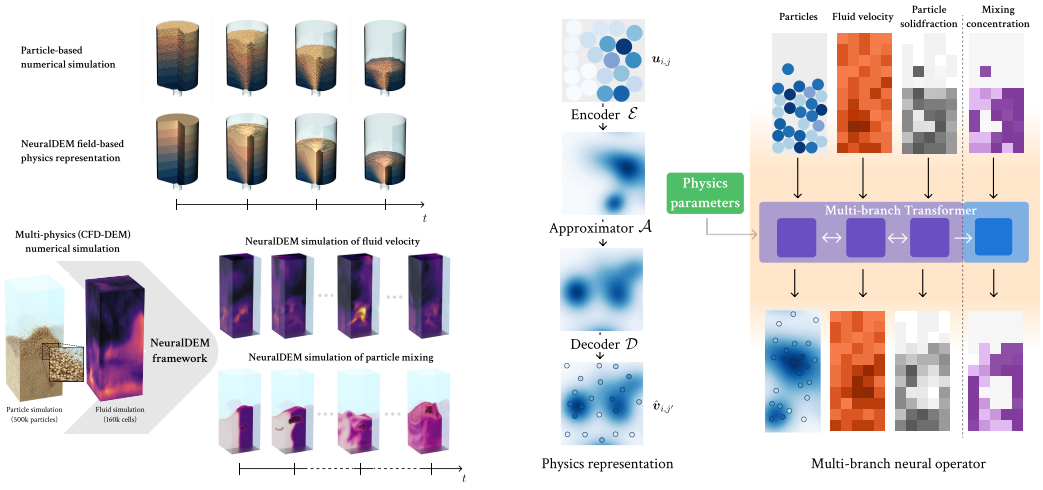


Figure 1: **NeuralDEM** presents an end-to-end approach to replace discrete element method (DEM) routines and coupled multiphysics simulations with deep learning surrogates. **Left top**: Hopper simulations. NeuralDEM treats inputs and outputs as continuous fields, while modeling macroscopic behavior directly as additional auxiliary fields. **Left bottom**: Fluidized bed reactors. Air enters the domain from the bottom plane (CFD problem) and pushes the particles up (DEM problem). NeuralDEM is built to model complex multiphysics simulations. **Right**: Both parts and the interaction thereof is modeled via the new multi-branch neural operator approach.

1 INTRODUCTION

In recent years, real-time numerical simulations (Chen et al., 2020; Ali et al., 2024; Guillaud et al., 2015; Yan et al., 2023; Javaid et al., 2023), have emerged as new modeling paradigm, enabling immediate analysis and decision-making based on live data and conditions. Unlike traditional simulations, which may take hours or even days to run, real-time simulations provide instantaneous feedback, allowing users to interact with and adjust parameters on the fly. Moreover, in engineering, fast simulations are driving the design of safer and more efficient structures and machines by accurately predicting their behavior under different conditions, thereby allowing extensive scans of vast parameter spaces, and eliminating the need for expensive physical prototypes.

Among the different numerical methods available for particle mechanics, the discrete element method (DEM) (Cundall & Strack, 1979) provides one of the most accurate representations of a wide range of physical systems, by tracking and computing the behavior of each particle. Consequently, DEM has become a widely accepted approach for tackling engineering problems in granular flows and powder mechanics. However, the inherent multiscale nature of particulate systems makes DEM computationally costly. (i) Large-scale granular flows consist of a huge number of particles, each interacting with the surrounding ones. (ii) The high material stiffness of solid particles severely limits the numerical timestep that can be used in the DEM. Often, its value is in the range of microseconds, whereas process-relevant durations may be minutes or hours.

Issue (i) is usually mitigated by employing coarse-graining techniques that replace many small particles with a large parcel (Bierwisch et al., 2009; Sakai & Koshizuka, 2009). If the interaction parameters of these parcels are chosen appropriately – either using scaling rules or a calibration routine – the accuracy impairments compared to the fine-grained ground truth are often acceptable. However, the limitation of small timesteps and the need for parameter calibration persist and make DEM slow and sometimes too cumbersome for a quick application within engineering workflows.

We present **NeuralDEM**, the first end-to-end deep learning alternative for modeling industrial processes. NeuralDEM introduces *multi-branch neural operators* inspired by multi-modal diffusion transformers (MMDiT) (Esser et al., 2024) and is scalable to real-time modeling of industrially-sized relevant scenarios. In NeuralDEM, we introduce two key components. The first is modeling the Lagrangian discretization given by DEM simulations directly from a compressed Eulerian perspective, i.e., a *field-based point of view*. Interestingly, this new modeling point of view aligns well with recent findings that a DEM-simulated system's effective degrees of freedom are orders of magnitude less than the microscopic degrees of freedom (Lichtenegger, 2018). This has the benefit of allowing direct modeling of macroscopic processes, e.g., mixing or transport processes, via additional auxiliary continuous fields. The second key component is *multi-physics modeling* via repeated interactions between physics phases. Multi-physics is prevalent when modeling the interaction of fluid dynamics and particulate systems in fluidized bed simulations. NeuralDEM, additionally, learns to stably simulate the system using longer timesteps, addressing issue (ii). Together, these properties allow tackling all common issues with DEM, making systems with large numbers of particles computationally feasible and allowing the use of longer timesteps.

We test NeuralDEM and demonstrate its capability to picture various transport processes, e.g., mass, species and mixing, in two scenarios, both of which are simulated with several 100k DEM particles: (i) slow and pseudo-steady hoppers with varying hopper angles, internal friction angles and flow regimes, and (ii) fast and transient fluidized bed reactors with varying inflow velocities. In all scenarios, we investigate the correct physics modeling of, e.g., outflow ratios, mixing ratio, and others. We observe that NeuralDEM generalizes to unseen parameter choices, and, that NeuralDEM produces faithful physics simulations for long time horizons. Most notably, our largest NeuralDEM model is able to physically-correctly model coupled CFD-DEM fluidized bed reactors of 160k CFD cells and 500k DEM particles for trajectories of 28 s, which amounts to 2800 machine learning timesteps. These findings will open many new doors to advanced engineering and much faster process cycles.

To provide extensive visualization of the temporally evolving systems considered in this paper, we created github project page: https://emmi-ai.github.io/NeuralDEM/

2 BACKGROUND

In this section, we introduce the relevant core phenomena of DEM. A more in-depth explanation can be found, e.g., in the review article form Blais et al. (2019) or the textbook from Norouzi et al. (2016). Further, we introduce neural operators and discuss their applicability to model particulate systems.

2.1 DISCRETE ELEMENT METHOD

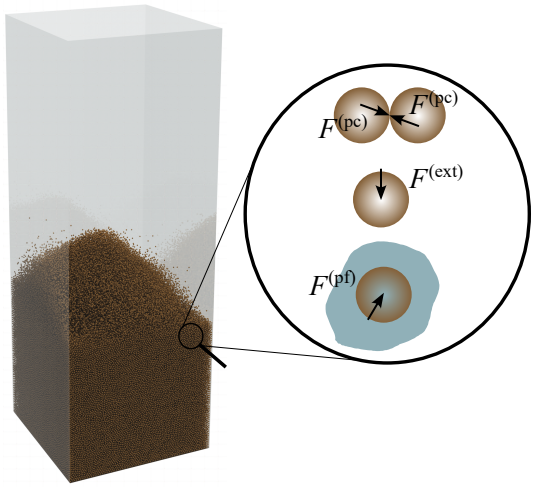


Figure 2: *Discrete element method.* The force on a particle consists of particle-particle contacts $\boldsymbol{F}_i^{(\mathrm{pc})}$, the external force $\boldsymbol{F}_i^{(\mathrm{ext})}$, and the interaction with a surrounding fluid phase $\boldsymbol{F}_i^{(\mathrm{pf})}$.

In a system of solid particles with masses m_i, radii r_i, positions $\boldsymbol{r}_i$, and velocities $\boldsymbol{v}_i$, each of them has to obey Newton's second law

$$\frac{\mathrm{d}}{\mathrm{d}t} m_i \boldsymbol{v}_i = \boldsymbol{F}_i^{(\mathrm{ext})} + \boldsymbol{F}_i^{(\mathrm{pc})} + \boldsymbol{F}_i^{(\mathrm{pf})}. \tag{1}$$

Particle i experiences forces of external origin, most important gravity $\boldsymbol{F}_i^{(\mathrm{ext})} \approx m_i \boldsymbol{g}$, contact forces with the nearby grains and walls $\boldsymbol{F}_i^{(\mathrm{pc})} = \sum_{j \neq i} \boldsymbol{F}_{i,j}$, and the influence of a surrounding fluid phase $\boldsymbol{F}_i^{(\mathrm{pf})}$ if present and relevant.

The contact force between solid particles i and j is commonly approximated with spring-dashpot models for both the normal $\boldsymbol{F}_{i,j}^{(\mathrm{n})}$ and tangential component $\boldsymbol{F}_{i,j}^{(\mathrm{t})}$,

$$\boldsymbol{F}_{i,j}^{(\mathrm{n})} = -k^{(\mathrm{n})} \delta_{i,j}^{(\mathrm{n})} \boldsymbol{n}_{i,j} + \gamma^{(\mathrm{n})} \boldsymbol{v}_{i,j}^{(\mathrm{n})} \tag{2}$$

$$\boldsymbol{F}_{i,j}^{(\mathrm{t})} = \min\Big[-k^{(\mathrm{t})} \delta_{i,j}^{(\mathrm{t})} \boldsymbol{t}_{i,j} + \gamma^{(\mathrm{t})} \boldsymbol{v}_{i,j}^{(\mathrm{t})}, \mu |\boldsymbol{F}_{i,j}^{(\mathrm{n})}| \boldsymbol{t}_{i,j} \Big], \tag{3}$$

where the tangential force is limited by Coulomb's friction law. Material properties enter these expressions in terms of the spring stiffnesses $k^{(\mathrm{n,t})}$, damping coefficients $\gamma^{(\mathrm{n,t})}$ and sliding friction μ. $k^{(\mathrm{n,t})}$ give rise to the reaction against normal and tangential overlap $\delta_{i,j}^{(\mathrm{n,t})}$ between two grains in the respective directions $\boldsymbol{n}_{i,j}$ and $\boldsymbol{t}_{i,j}$, and $\gamma^{(\mathrm{n,t})}$ account for viscous dissipation caused by the normal and tangential relative velocities $\boldsymbol{v}_{i,j}^{(\mathrm{n,t})}$ during contact. While simple geometric shapes such as perfect spheres allow computing values for the material parameters from measurable properties like Young's and shear modulus (Johnson, 1985), actual, imperfect grains necessitate calibration towards characterization experiments (Coetzee, 2017).

The magnitude of the numerical timestep to solve Equation (1) is limited by the requirement to properly resolve contacts between grains. More specifically, the timestep needs to be significantly smaller than the duration of contact of colliding particles ("Hertz time") and the time it takes density waves generated upon impact to travel over the grain surface ("Rayleigh time"). For stiff materials, this often amounts to steps in the range of microseconds.

2.1.1 COUPLED PARTICLE-FLUID SIMULATIONS

Particles will also experience a force from a surrounding fluid phase. While it may be neglected if no significant relative velocities occur, it can be a crucial factor for particle dynamics otherwise. The dominant contributions are usually caused by gradients of the pressure and by the drag force, i.e., the resistance against relative velocity between fluid and grain, so that

$$\boldsymbol{F}_i^{(\mathrm{pf})} \approx -V_i \nabla p + \beta\big(\boldsymbol{u}_\mathrm{f} - \boldsymbol{v}_i\big). \tag{4}$$

The drag coefficient β depends on the particle size and the local flow conditions. A multitude of empirical correlations can be found in the literature to take into account the impact of Reynolds number, particle volume fraction α_p, size distribution, etc. (Kieckhefen et al., 2020).

The fluid velocity itself is governed by the filtered Navier-Stokes equations (Anderson & Jackson, 1967)

$$\frac{\partial}{\partial t}\alpha_\mathrm{f} + \nabla \cdot \alpha_\mathrm{f} \boldsymbol{u}_\mathrm{f} = 0 \tag{5}$$

$$\frac{\partial}{\partial t}\alpha_\mathrm{f}\boldsymbol{u}_\mathrm{f} + \nabla \cdot \alpha_\mathrm{f}\boldsymbol{u}_\mathrm{f}\boldsymbol{u}_\mathrm{f} = \nabla \cdot \boldsymbol{\sigma}_\mathrm{f} - f^{(\mathrm{pf})} \tag{6}$$

which differ from their single-phase counterpart in two regards. The presence of particles reduces the locally available volume to a fraction $\alpha_\mathrm{f} = 1 - \alpha_\mathrm{p}$, and the density of force Equation (4) exerted by the fluid on the particles is felt by the fluid in opposite direction because of Newton's third law. The coupled solution of the CFD Equations (5) and (6) and the DEM Equation (1) gives rise to CFD-DEM simulations.

For a proper definition of the field quantities α_p and $f^{(\mathrm{pf})}$, Lagrangian particle information needs to be mapped onto Eulerian fields. To this end, a filter function $g_l(r)$, e.g., a Gaussian with width l, is employed in terms of

$$\alpha_\mathrm{p}(\boldsymbol{r}) \equiv \sum_i g_l(|\boldsymbol{r} - \boldsymbol{r}_i|)V_i \tag{7}$$

$$f^{(\mathrm{pf})}(\boldsymbol{r}) \equiv \frac{\sum_i g_l(|\boldsymbol{r} - \boldsymbol{r}_i|)V_i \boldsymbol{F}_i^{(\mathrm{pf})}}{\sum_i g_l(|\boldsymbol{r} - \boldsymbol{r}_i|)V_i}. \tag{8}$$

An analogous definition as Equation (8) can be invoked to define the *spatial field distribution* of any particle property. As a matter of fact, the target quantities of most particle simulations are not necessarily connected to single-particle properties located exactly at the positions of each grain. Instead, one might be interested in the spatial distribution of, e.g., particle volume fraction, residence time or temperature. Two strategies are available to obtain these fields: (i) One carries out a DEM simulation and postprocesses particle data according to Equation (8). The trajectory and properties of each grain are only needed as an intermediate step for the DEM simulation. (ii) One can try to directly formulate particle EOMs in an Eulerian fashion by filtering the Lagrangian ones and solving them disregarding discrete properties. As demonstrated by the two-fluid model (Gidaspow, 1994), this can significantly reduce computational costs but can come with a serious degree of uncertainty (Chen & Wang, 2014) because not all particle properties lend themselves to a straight-forward formulation in terms of fields.

Even if the resulting inaccuracies are acceptable, such simulations are still cumbersome because of the restriction to small timesteps (which is also present in an Eulerian formulation) and the lack of a direct relationship between macroscopic behavior and microscopic parameters. An attractive solution to this predicament might be offered by neural operators that can be trained with detailed particle data to predict any underlying field quantities in a highly efficient way.

2.2 NEURAL OPERATORS LEARNING FOR SCIENTIFIC AND ENGINEERING APPLICATIONS

In recent years, deep learning tools have been extensively integrated into scientific modeling, and have resulted in breakthroughs in, e.g., protein folding (Jumper et al., 2021; Abramson et al., 2024), material discovery (Batzner et al., 2022; Batatia et al., 2022; Merchant et al., 2023; Zeni et al., 2023), or weather modeling (Pathak et al., 2022; Bi et al., 2023; Lam et al., 2023; Bodnar et al., 2024; Nguyen et al., 2023). Driven by applications in CFD (Vinuesa & Brunton, 2022; Guo et al.,

2016; Li et al., 2021; Kochkov et al., 2021; Gupta & Brandstetter, 2023), deep neural network based surrogates, most importantly neural operators (Li et al., 2021; Lu et al., 2021; Kovachki et al., 2023), have emerged as a computationally efficient alternative (Zhang et al., 2023). In addition to computational efficiency, neural operators offer the potential to introduce generalization capabilities across phenomena, as well as generalization across characteristics such as boundary conditions or coefficients (McCabe et al., 2023; Herde et al., 2024).

Neural operators (Lu et al., 2021; Li et al., 2020; 2021; Kovachki et al., 2023) are formulated with the aim of learning a mapping between function spaces, enabling outputs that remain consistent across varying input sampling resolutions. Following the framework of Kovachki et al. (2023), we assume $\mathcal{U}, \mathcal{V}$ to be Banach spaces of functions defined on compact domains $\mathcal{X} \subset \mathbb{R}^{d_x}$ or $\mathcal{Y} \subset \mathbb{R}^{d_y}$, respectively, which map into $\mathbb{R}^{d_u}$ or $\mathbb{R}^{d_v}$. A neural operator $\hat{\mathcal{G}} : \mathcal{U} \to \mathcal{V}$ approximates the ground truth operator $\mathcal{G} : \mathcal{U} \to \mathcal{V}$.

When training a neural operator $\hat{\mathcal{G}}$, a widely adopted approach is to construct a dataset of N discrete data pairs $(\boldsymbol{u}_{i,j}, \boldsymbol{v}_{i,j'})$, $i = 1, \ldots, N$, which correspond to $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ evaluated at spatial locations $j = 1, \ldots, K$ and $j' = 1, \ldots, K'$, respectively. Note that K and K' can, but need not be equal, and can vary for different i, which we omit for notational simplicity. Figure 3 (first row) shows the operator learning problem, i.e., the mapping of an input function $\boldsymbol{u}_i$ to an output function $\boldsymbol{v}_i$ via an operator $\mathcal{G}$. The functions are given via K and K' discretized input and output points, respectively. On this dataset, $\hat{\mathcal{G}}$ is trained to map $\boldsymbol{u}_{i,j}$ to $\boldsymbol{v}_{i,j'}$ via supervised learning, as sketched in Figure 3 (bottom row), where $\hat{\mathcal{G}}$ is composed of three maps (Seidman et al., 2022; Alkin et al., 2024): $\hat{\mathcal{G}} := \mathcal{D} \circ \mathcal{A} \circ \mathcal{E}$, comprising the encoder $\mathcal{E}$, the approximator $\mathcal{A}$, and the decoder $\mathcal{D}$. First, the encoder $\mathcal{E}$ transforms the discrete function samples $\boldsymbol{u}_{i,j}$ to a latent representation of the input function. Then, the approximator $\mathcal{A}$ maps the latent representation to a representation of the output function. Lastly, the decoder evaluates the output function at spatial locations j'. The neural network $\hat{\mathcal{G}}$ is then trained via gradient descent, using the gradient of, e.g., a mean squared error loss in the discretized space $\mathcal{L}_i = \frac{1}{K'} \sum_{j'} \|\hat{\boldsymbol{v}}_{i,j'} - \boldsymbol{v}_{i,j'}\|_2^2$, where $\|_2$ is the Euclidean norm.

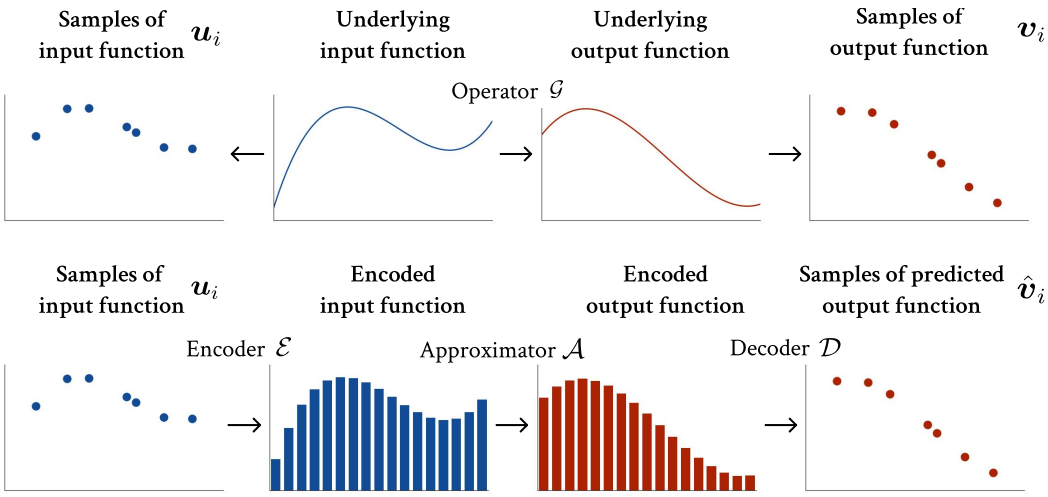


Figure 3: Neural operator learning. Neural operators aim to learn a mapping between function spaces, enabling outputs that remain consistent across varying input sampling resolutions. The neural operator $\hat{\mathcal{G}}$ approximates the ground truth operator $\mathcal{G}$ with three maps, composing encoder $\mathcal{E}$, approximator $\mathcal{A}$, and decoder $\mathcal{D}$. The approximation of $\hat{\mathcal{G}}$ is ideally independent from the number of sampled input points, and approximates the output function for an arbitrary number of points.

For temporally evolving systems, we assume the Banach spaces of functions to be equal, i.e., $\mathcal{U} = \mathcal{V}$, where models are trained on next-step prediction $\boldsymbol{u}_{i,j}^t \to \boldsymbol{v}_{i,j'}^{t'}$. To fully evolve a system after training, the model is then applied repeatedly in autoregressive fashion, i.e., every prediction serves as new input.

3 METHOD

NeuralDEM presents the first end-to-end solution for replacing computationally intensive numerical DEM routines and coupled CFD-DEM simulations with fast and flexible deep learning surrogates. NeuralDEM introduces two conceptually novel modeling paradigms:

1. *Physics representation*: We model the Lagrangian discretization of DEM as an underlying continuous field, while simultaneously modeling macroscopic behavior directly as additional auxiliary fields. NeuralDEM encodes different physics inputs which are representative for DEM dynamics and/or multi-physics scenarios. Examples are particle displacement, particle mixing, solid fraction, or particle transport.
2. *Multi-branch neural operators*: We introduce multi-branch neural operators scalable to real-time modeling of industrial-size scenarios. Multi-branch neural operators build on the flexible and scalable "Universal Physics Transformer" (Alkin et al., 2024) framework by enhancing encoder, decoder, and approximator components using multi-branch transformers to allow for modeling of multi-physics systems. The system quantities fundamental to predicting the evolution of the state in time are modeled in the main-branches, where they are tightly coupled. Additionally, auxiliary *off-branches* can be added to directly model macroscopic quantities by retrieving information from the main-branch state and further refining the prediction using relevant inputs.

3.1 PHYSICS REPRESENTATION

As common when training deep learning surrogates, we train on orders of magnitude coarser timescales than what a classical solver requires to be stable and accurate. For the numerical experiments in this paper, the timescale relation is at least $1000\Delta t_{\text{DEM}} = \Delta t_{\text{ML}}$. Additionally, for learning the dynamics of particle movement, we use the *particle displacement*, which is defined as the difference between the position $\boldsymbol{r}_i \in \mathbb{R}^3$ of particle i at timestep t_{ML} and the position of the same particle at timestep $t_{\text{ML}} + \Delta t_{\text{ML}}$. Finally, we use the term *transport* to denote the particle movement integrated over multiple timesteps Δt_{ML}.

NeuralDEM models the Lagrangian discretization of DEM as a continuous field in a compressed latent space, leveraging the insight that the effective degrees of freedom of physical systems is often much smaller than its input dimensionality (Lichtenegger, 2018). Therefore, we assume that there exists some underlying field that describes the particle displacements in a DEM simulation and learn this underlying field over the whole domain instead of a displacement per particle. However, particle displacements can fluctuate depending on their exact position within the bulk of the material. Such fine-grained details are lost when going to a field-based representation which smoothes out these variations. This makes field-based models unable to move particles accurately around in space, which would be required to get macroscopic insights into the simulation dynamics.

To circumvent this issue, we introduce additional auxiliary fields that model the macroscopic insights directly instead of calculating them in post-processing from the particle locations. For example, by modeling the accumulated particle movement over a long period of time via a "transport" field, we can learn macroscopic properties directly instead of integrating short-term movements which would require precise prediction of the fluctuations thereof. This is visualized on the right side of Figure 1.

Even over such a large timestep, the evolution of a flow and its properties at each point is mainly determined by the field values in a nearby, bounded subdomain which grows with the the step size, and hardly influenced by very distant points (Lichtenegger, 2024). This behavior can be resembled by the attention mechanism of transformer networks (Vaswani et al., 2017).

3.2 MULTI-BRANCH NEURAL OPERATORS

Macroscopic modeling via auxiliary fields. An important observation regarding the systems we model is that the insights that a classical solver can provide into the physical dynamics are rarely on a microscopic level and more often on a macroscopic level, where the macroscopic properties are extracted from the microscopic results of the classical solver. Motivated by this intuition, we introduce additional off-branches, which are trained to model macroscopic processes such as particle mixing or particle transport directly during training. Similar to classical solvers, where the

macroscopic process does not influence the microscopic updates, off-branches do not influence any of the main-branches. Instead, each off-branch creates its predictions by repeatedly processing its own data, as well as retrieving information from the microscopic state of the main-branches (without influencing them).

Multi-branch transformers. The central neural network component of NeuralDEM are multi-branch transformers. Multi-branch transformers, as the name suggests, consist of multiple branches: main-branch(es) and off-branch(es). Each branch is a stack of transformer (Vaswani et al., 2017) blocks where weights are not shared between branches. Each branch operates on a set of so-called tokens, which are obtained by embedding the input into a compressed latent representation. Main-branch(es) concatenate all tokens before each attention operation along the set dimension, allowing interactions between them, followed by splitting tokens again into the different branches, akin to multi-modal diffusion transformer (MMDiT) blocks (Esser et al., 2024). Additionally, multi-branch transformers can include arbitrarily many off-branches, where the self-attention is replaced by a cross-attention which uses only its own off-branch tokens as queries and concatenates its own off-branch tokens with the main-branch tokens to use as keys and values. This roughly corresponds to simultaneous self-attention between the off-branch tokens and cross-attention between off-branch and main-branch tokens. No gradient flows through the cross-attention back to the main-branch tokens. Off-branches are implemented via a modified diffusion transformer block (Peebles & Xie, 2023). A schematic sketch is shown on the right side of Figure 1 and a detailed architecture visualization is provided in Appendix Figure 12.

In our numerical experiments, we consider temporally evolving systems of multiple fields. Each input at time t $\boldsymbol{u}_i^t$ consists of $h = 1, \ldots, M$ fields, where the hth field at timestep t is denoted as $\boldsymbol{u}_i^{h,t}$. Each field is modeled by one branch of the multi-branch transformer. We create datasets of function pairs that are evaluated at K and K' input and output positions $(\boldsymbol{u}_{i,j}^{h,t}\ \boldsymbol{v}_{i,j'}^{h,t+\Delta t})$ and train all M branches in parallel to map $\boldsymbol{u}_{i,j}^{h,t}$ to the target $\boldsymbol{v}_{i,j'}^{h,t+\Delta t}$. Each branch of the multi-branch transformer consists of M encoders $\mathcal{E}^h$, M approximators $\mathcal{A}^h$, and M decoders $\mathcal{D}^h$.

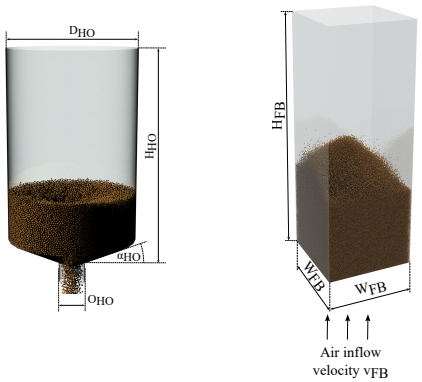
$$\mathcal{E}^h : \boldsymbol{u}_{i,j=1,\ldots,K}^{h,t} \in \mathbb{R}^{K\times d} \xrightarrow{\text{embed}} \mathbb{R}^{K\times d_{\text{hidden}}} \xrightarrow{\text{multi-branch transformer}} \boldsymbol{z}_i^{h,t} \in \mathbb{R}^{n_{\text{latent}}\times d_{\text{hidden}}}$$

$$\mathcal{A}^h : \boldsymbol{z}_i^{h,t} \in \mathbb{R}^{n_{\text{latent}}\times d_{\text{hidden}}} \xrightarrow{\text{multi-branch transformer}} \boldsymbol{z}_i^{h,t+\Delta t} \in \mathbb{R}^{n_{\text{latent}}\times d_{\text{hidden}}}$$

$$\mathcal{D}^h : (\boldsymbol{z}_i^{h,t+\Delta t}, \boldsymbol{y}_{i,j'=1,\ldots,K'}^{h}) \xrightarrow{\text{perceiver decoder}} \hat{\boldsymbol{v}}_{i,j'=1,\ldots,K'}^{h,t+\Delta t} \in \mathbb{R}^{K'\times d}\ .$$

4 Numerical Experiments

We test the NeuralDEM framework on two industrially relevant use cases: hoppers and fluidized bed reactors, both visualized in Figure 4 and described in Table 1. Additional details to the datasets are outlined in Appendix Section D and training details in Appendix Section E.



(a) Hopper (b) Fluidized bed

Parameter	Value
Hopper diameter D_{HO}	0.24 m
Hopper height H_{HO}	0.40 m
Hopper outlet diameter O_{HO}	0.05 m
Hopper angle α_{HO}	0-60°
Hopper particle diameter d	0.004 m
FB width w_{FB}	0.096 m
FB height h_{FB}	0.24 m
FB particle diameter d_{FB}	0.0012 m

(c) Dimensions of the cases

Figure 4: Schematic of the two numerical experimental cases and the associated dimensions.

Table 1: Overview of dataset properties used in the experiments.

Name	Simulation method	#Particles	#CFD cells	Variability
Hopper	DEM	250k	-	Hopper angle, material friction
Fluidized bed	CFD-DEM	500k	160k	Fluid inlet velocity

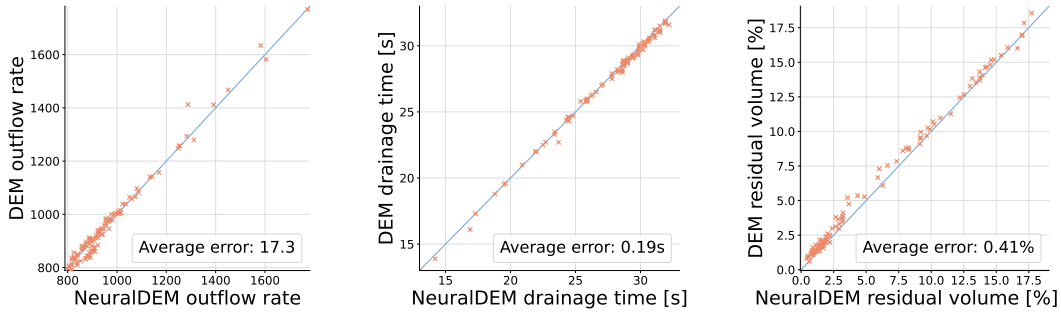
We are interested in developing a model for industrially relevant use-cases and therefore only consider our UPT (Alkin et al., 2024) based architecture, employing reduced order modeling and a field-based decoding for computationally efficiency and scalability to large systems. Other neural operator architectures would (i) be often not applicable to particle simulations as many are restricted to regular grids (ii) be computationally infeasible to train on such a scale and (iii) require additional components to introduce field-based modeling into their architecture.

4.1 HOPPER

With its outlet closed, the hopper is initially filled with particles, to roughly 250k grains on average (particle counts can vary based on the outlet slope α_{HO}). Then the outlet at the bottom of the hopper is opened and grains start to flow out. We create a dataset of 1000 simulations with a train/validation/test split of 800/100/100. Each simulation is run to cover 40 physical seconds at most (the simulation is stopped if no particles remain in the hopper). Snapshots are stored in $0.1\,\text{s}$ intervals (resulting in 400 ML timesteps Δt_{ML}) which is the data that NeuralDEM models are trained on. The DEM solver requires 10k timesteps per 0.1 physical seconds. Simulating 40s of particle flow with DEM requires 3 hours on 16 CPUs whereas NeuralDEM takes only $8\,\text{s}$ on the same hardware.

The following macroscopic properties of hopper simulations are of interest to practitioners: (i) Peak outflow rate: How many particles exit the hopper within a certain timeframe at most? (ii) Drainage time: How long does it take to empty the hopper? (iii) Residual material: How much material got stuck inside the hopper after full drainage? (iv) Flow regime: Does the material exhibit mass flow or funnel flow? (v) Visualization: How does the simulation look like?

NeuralDEM is able to accurately model tasks (i) to (iii) as shown in Figure 5. Additionally, NeuralDEM produces faithful flow regimes and visualizations as shown in Appendix A.



(a) Outflow rate (b) Drainage time (c) Residual volume

Figure 5: Macroscopic simulation insights from the predicted occupancy field. The NeuralDEM model can accurately capture macroscopic measurements from the learned dynamics.

4.2 FLUIDIZED BED REACTORS

We consider the highly dynamic system of fluidized bed reactors, where particles and air interact, necessitating a multi-physics modeling. These processes are described through a coupled CFD-DEM simulation, where DEM is used to handle particles, while the surrounding fluid is simulated by CFD. The physical duration of one DEM simulation amounts to $5\,\text{s}$, comprising 2M DEM timesteps and 20k CFD timesteps. To train the NeuralDEM model, each trajectory is sub-sampled to 300 timesteps, starting from $2\,\text{s}$ of the original simulation and sampling every $\Delta t_{\text{ML}} = 0.01\,\text{s}$. We select 60 inlet velocities at random for the training set and the remaining 16 are left for validation.

Practitioners are often interested to evaluate how well particles are mixed within the fluidized bed reactor. To this end, we train a NeuralDEM model to model the mixing behavior in addition to the particle and fluid phase. Particle mixing by definition is a particle-associated quantity where each particle either belongs to group A or B. In our numerical experiments, we label particles belonging to group A those that start in the left half of the reactor, and particles belonging to group B those that start in the right half. However, when using field-based representations, a faithful modeling of particle mixing is not feasible. Therefore, we introduce the particle mixing concentration field, which defines – for a given spatial location and time – the concentration of particles that belong to group B. Consequently, we discretize the domain using a hexahedron mesh and map the particle information using a Gaussian kernel on that mesh, as shown in Figure 6.

We use the Lacey mixing index (Lacey, 1954) to compare the model prediction for the mixing concentration field and ground truth simulation data. The Lacey mixing index represents the current state of mixing and can be plotted over time to compare the time evolution of the process. As the particles get more mixed, the Lacey mixing index goes towards 1. As visualized in Figure 7, NeuralDEM predictions match the characteristics of the ground truth DEM-CFD trajectories over long time horizons.

Additionally, we evaluate the accuracy of modeling the fluid and particle phases as well as time extrapolation capabilities of our model in Appendix B. Numerically, when using traditional CFD and DEM methods, these simulation of a fluidized bed reactor of 500k particles, with a trajectory spanning 3 s amounts to 12k CFD and 1.2M DEM timesteps and requires 6 hours on 64 cores of high-performance CPUs. In contrast, on a single state-of-the-art GPU, the fastest NeuralDEM inference model faithfully reproduces the same physical behavior in just 11 s.

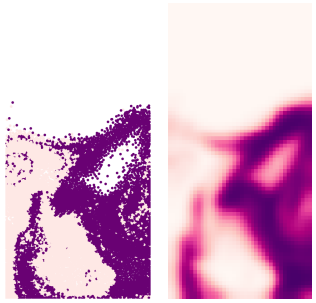
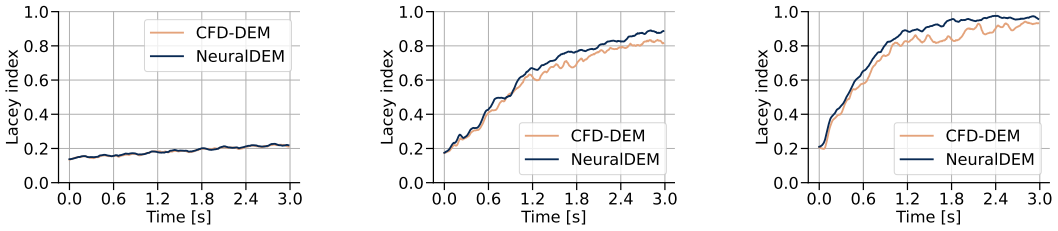


Figure 6: **Left**: *Particle-based mixing* where the dark particles started from the right reactor half. **Right**: *Field-based concentration* obtained using a Gaussian kernel.



(a) Low inlet velocity (b) Medium inlet velocity (c) High inlet velocity

Figure 7: Temporal evolution of the Lacey mixing index for different inlet velocities. NeuralDEM predictions (blue) match the characteristics of the ground truth DEM-CFD trajectories (orange) over long time horizons. In the low inlet velocity simulation, mixing is slow and the model can very accurately predict the mixing rate. For higher inlet velocities mixing is faster and the exact behavior looks more random because the behavior is more chaotic. However, the model correctly captures the exact trend in the mixing, starting fast and later slowing down.

5 Conclusion

The present work has introduced NeuralDEM, a multi-branch neural operator framework that can learn the complex behavior of particulate systems over a wide range of dynamic regimes: from dense, pseudo-steady motion in hoppers to dilute, highly unsteady flow in fluidized bed reactors. NeuralDEM treats the Lagrangian discretization of DEM as an underlying continuous field while, simultaneously, modeling macroscopic behavior directly as additional auxiliary fields. Long-term rollouts of our data-driven model show a high degree of stability and lead to accurate predictions regarding various target quantities such as residence times or mixing indices even for unseen conditions like different wall geometries, material properties, or boundary values. Evaluation times are several orders of magnitude faster than the underlying (CFD-)DEM simulations and demonstrate the real-time capability of NeuralDEM.

ACKNOWLEDGEMENTS

We would like to sincerely thank Dennis Just, Miks Mikelsons, Robert Weber, Bastian Best, the whole NXAI team, and the whole Emmi AI team for ongoing help and support. We are grateful to Sepp Hochreiter, Phillip Lippe, Maurits Bleeker, Patrick Blies, Andreas Fürst, Andreas Mayr, Andreas Radler, Behrad Esgandari, Daniel Queteschiner for valuable inputs. Samuele Papa and Johannes Brandstetter thank Efstratios Gavves and Jan-Jakob Sonke for the help to make Samuele's research exchange smooth and mutual beneficial.

We acknowledge EuroHPC Joint Undertaking for awarding us access to Karolina at IT4Innovations, Czech Republic, MeluXina at LuxProvide, Luxembourg, LUMI at CSC, Finland and Leonardo at CINECA, Italy. The ELLIS Unit Linz, the LIT AI Lab, the Institute for Machine Learning, are supported by the Federal State Upper Austria.

REFERENCES

Abramson, J., Adler, J., Dunger, J., Evans, R., Green, T., Pritzel, A., Ronneberger, O., Willmore, L., Ballard, A. J., Bambrick, J., et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, pp. 1–3, 2024.

Ali, Z., Biglari, R., Denil, J., Mertens, J., Poursoltan, M., and Traoré, M. K. From modeling and simulation to digital twin: evolution or revolution? *SIMULATION: Transactions of The Society for Modeling and Simulation International*, 100(7):751–769, 2024.

Alkin, B., Fürst, A., Schmid, S., Gruber, L., Holzleitner, M., and Brandstetter, J. Universal physics transformers. *arXiv preprint arXiv:2402.12365*, 2024.

Anderson, T. B. and Jackson, R. O. Y. A fluid mechanical description of fluidized beds. *Ind. Eng. Chem. Fundam.*, 6(4):527–539, 1967.

Batatia, I., Kovacs, D. P., Simm, G., Ortner, C., and Csányi, G. Mace: Higher order equivariant message passing neural networks for fast and accurate force fields. *Advances in Neural Information Processing Systems*, 35:11423–11436, 2022.

Batzner, S., Musaelian, A., Sun, L., Geiger, M., Mailoa, J. P., Kornbluth, M., Molinari, N., Smidt, T. E., and Kozinsky, B. E (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature communications*, 13(1):2453, 2022.

Bi, K., Xie, L., Zhang, H., Chen, X., Gu, X., and Tian, Q. Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, 619(7970):533–538, 2023.

Bierwisch, C., Kraft, T., Riedel, H., and Moseler, M. Three-dimensional discrete element models for the granular statics and dynamics of powders in cavity filling. *J. Mech. Phys. Solids*, 57(1): 10–31, 2009.

Blais, B., Vidal, D., Bertrand, F., Patience, G. S., and Chaouki, J. Experimental methods in chemical engineering: Discrete element method—DEM. *Can. J. Chem. Eng.*, 97(7):1964–1973, 2019.

Bodnar, C., Bruinsma, W. P., Lucic, A., Stanley, M., Brandstetter, J., Garvan, P., Riechert, M., Weyn, J., Dong, H., Vaughan, A., et al. Aurora: A foundation model of the atmosphere. *arXiv preprint arXiv:2405.13063*, 2024.

Chen, X. and Wang, J. A comparison of two-fluid model, dense discrete particle model and CFD-DEM method for modeling impinging gas–solid flows. *Powder Technol.*, 254:94–102, 2014.

Chen, X., Liang, C., Huang, D., Real, E., Wang, K., Pham, H., Dong, X., Luong, T., Hsieh, C., Lu, Y., and Le, Q. V. Symbolic discovery of optimization algorithms. *Advances in Neural Information Processing Systems*, 36:49205–49233, 2023.

Chen, Y., Yang, O., Sampat, C., Bhalode, P., Ramachandran, R., and Ierapetritou, M. Digital twins in pharmaceutical and biopharmaceutical manufacturing: A literature review. *Processes*, 8(9), 2020. ISSN 2227-9717.

Coetzee, C. J. Calibration of the discrete element method. *Powder Technol.*, 310:104–142, 2017.

Cundall, P. A. and Strack, O. D. L. A discrete numerical model for granular assemblies. *Géotechnique*, 29(1):47–65, 1979.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

Esser, P., Kulal, S., Blattmann, A., Entezari, R., Müller, J., Saini, H., Levi, Y., Lorenz, D., Sauer, A., Boesel, F., Podell, D., Dockhorn, T., English, Z., and Rombach, R. Scaling rectified flow transformers for high-resolution image synthesis. In *ICML*, volume 235 of *Proceedings of Machine Learning Research*, pp. 12606–12633. PMLR, 2024.

Gidaspow, D. *Multiphase Flow and Fluidization: Continuum and Kinetic Theory Descriptions*. Elsevier Science, 1994. ISBN 9780122824708.

Goniva, C., Kloss, C., Deen, N. G., Kuipers, J. A. M., and Pirker, S. Influence of rolling friction on single spout fluidized bed simulation. *Particuology*, 10:582–591, 2012.

Guillaud, X., Faruque, M. O., Teninge, A., Hariri, A. H., Vanfretti, L., Paolone, M., Dinavahi, V., Mitra, P., Lauss, G., Dufour, C., Forsyth, P., Srivastava, A. K., Strunz, K., Strasser, T., and Davoudi, A. Applications of real-time simulation technologies in power and energy systems. *IEEE Power and Energy Technology Systems Journal*, 2(3):103–115, 2015.

Guo, X., Li, W., and Iorio, F. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 481–490, 2016.

Gupta, J. K. and Brandstetter, J. Towards multi-spatiotemporal-scale generalized PDE modeling. *Trans. Mach. Learn. Res.*, 2023, 2023.

Herde, M., Raonić, B., Rohner, T., Käppeli, R., Molinaro, R., de Bézenac, E., and Mishra, S. Poseidon: Efficient foundation models for pdes. *arXiv preprint arXiv:2405.19101*, 2024.

Javaid, M., Haleem, A., and Suman, R. Digital twin applications toward industry 4.0: A review. *Cognitive Robotics*, 3:71–92, 2023. ISSN 2667-2413.

Johnson, K. L. *Contact Mechanics*. Cambridge University Press, Cambridge, UK, 1985.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

Kieckhefen, P., Pietsch, S., Dosta, M., and Heinrich, S. Possibilities and limits of computational fluid dynamics–discrete element method simulations in process engineering: a review of recent advancements and future trends. *Annu. Rev. Chem. Biomol. Eng.*, 11(1):397–422, 2020.

Kloss, C., Goniva, C., Hager, A., Amberger, S., and Pirker, S. Models, algorithms and validation for opensource DEM and CFD-DEM. *Prog. Comput. Fluid Dyn.*, 12:140–152, 2012.

Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., and Hoyer, S. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118 (21):e2101784118, 2021.

Kovachki, N. B., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A. M., and Anandkumar, A. Neural operator: Learning maps between function spaces with applications to pdes. *J. Mach. Learn. Res.*, 24:89:1–89:97, 2023.

Lacey, P. M. C. Developments in the theory of particle mixing. *Journal of Applied Chemistry*, 4(5): 257–268, 1954.

Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Alet, F., Ravuri, S., Ewalds, T., Eaton-Rosen, Z., Hu, W., et al. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.

Li, Z., Kovachki, N. B., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A. M., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. In *ICLR*, 2021.

Lichtenegger, T. Local and global recurrences in dynamic gas-solid flows. *Int. J. Multiphase Flow*, 106:125 – 137, 2018.

Lichtenegger, T. Data-assisted, physics-informed propagators for recurrent flows. *Phys. Rev. Fluids*, 9:024401, Feb 2024.

Lippe, P., Veeling, B., Perdikaris, P., Turner, R., and Brandstetter, J. Pde-refiner: Achieving accurate long rollouts with neural pde solvers. *Advances in Neural Information Processing Systems*, 36: 67398–67433, 2023.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3): 218–229, 2021.

McCabe, M., Blancard, B. R.-S., Parker, L. H., Ohana, R., Cranmer, M., Bietti, A., Eickenberg, M., Golkar, S., Krawezik, G., Lanusse, F., et al. Multiple physics pretraining for physical surrogate models. *arXiv preprint arXiv:2310.02994*, 2023.

Merchant, A., Batzner, S., Schoenholz, S. S., Aykol, M., Cheon, G., and Cubuk, E. D. Scaling deep learning for materials discovery. *Nature*, pp. 1–6, 2023.

Nguyen, T., Brandstetter, J., Kapoor, A., Gupta, J. K., and Grover, A. Climax: A foundation model for weather and climate. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pp. 25904–25938. PMLR, 2023.

Norouzi, H. R., Zarghami, R., Sotudeh-Gharebagh, R., and Mostoufi, N. *Coupled CFD-DEM modeling: formulation, implementation and application to multiphase flows*. John Wiley & Sons, 2016.

Pathak, J., Subramanian, S., Harrington, P., Raja, S., Chattopadhyay, A., Mardani, M., Kurth, T., Hall, D., Li, Z., Azizzadenesheli, K., et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

Peebles, W. and Xie, S. Scalable diffusion models with transformers. In *ICCV*, pp. 4172–4182. IEEE, 2023.

Sakai, M. and Koshizuka, S. Large-scale discrete element modeling in pneumatic conveying. *Chem. Eng. Sci.*, 64(3):533–539, 2009.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. Learning to simulate complex physics with graph networks. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8459–8468. PMLR, 2020.

Seidman, J., Kissas, G., Perdikaris, P., and Pappas, G. J. Nomad: Nonlinear manifold decoders for operator learning. *Advances in Neural Information Processing Systems*, 35:5601–5613, 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Vinuesa, R. and Brunton, S. L. Enhancing computational fluid dynamics with machine learning. *Nature Computational Science*, 2(6):358–366, 2022.

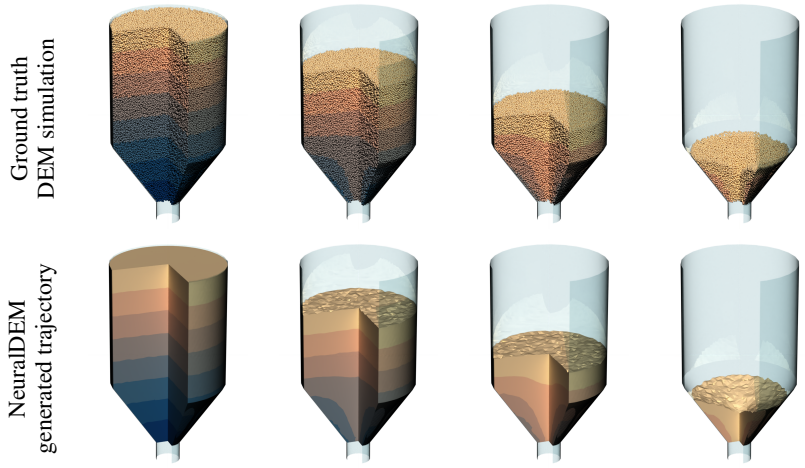
Weller, H. G., Tabor, G., Jasak, H., and Fureby, C. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computer in Physics*, 12(6):620–631, 11 1998. ISSN 0894-1866.

Yan, W., Wang, J., Lu, S., Zhou, M., and Peng, X. A review of real-time fault diagnosis methods for industrial smart manufacturing. *Processes*, 11(2), 2023. ISSN 2227-9717.

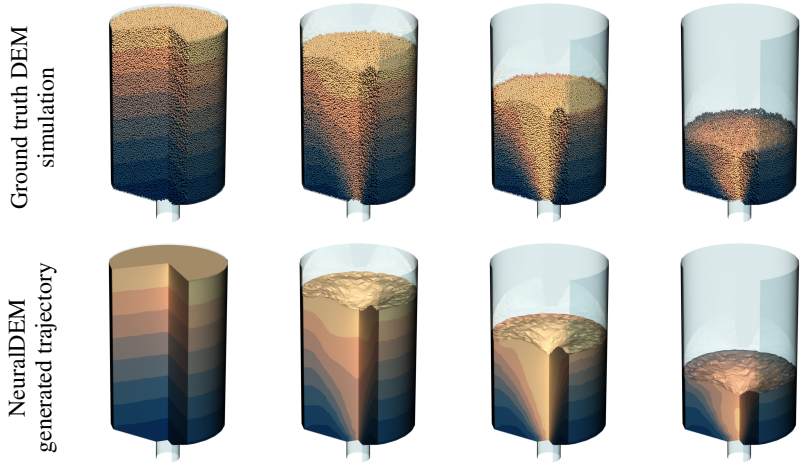
Zeni, C., Pinsler, R., Zügner, D., Fowler, A., Horton, M., Fu, X., Shysheya, S., Crabbé, J., Sun, L., Smith, J., et al. Mattergen: a generative model for inorganic materials design. *arXiv preprint arXiv:2312.03687*, 2023.

Zhang, X., Wang, L., Helwig, J., Luo, Y., Fu, C., Xie, Y., Liu, M., Lin, Y., Xu, Z., Yan, K., et al. Artificial intelligence for science in quantum, atomistic, and continuum systems. *arXiv preprint arXiv:2307.08423*, 2023.

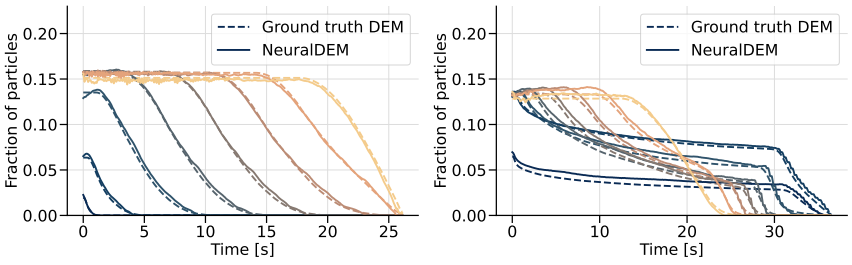
A EXTENDED HOPPER RESULTS



(a) Mass flow regime



(b) Funnel flow regime



(c) Mass flow regime

(d) Funnel flow regime

Figure 8: Visualization of a NeuralDEM generated trajectory vs the DEM simulation. Different colors indicate different initial particle layers. Emerging funnel and mass flow regimes are clearly visible. In mass flow (a), mass moves uniformly down towards the outlet. In funnel flow (b), particles primarily move down a funnel above the outlet. The fractions of colored material over time are shown for (c) mass flow regime and (d) funnel flow. In the funnel flow regime particle layer inversion happens, i.e., particles from higher layers overtake particles from the lower layers through the funnel. This emerging macroscopic phenomena is perfectly modeled by NeuralDEM.

B EXTENDED FLUIDIZED BED RESULTS

B.1 MODELING THE DYNAMICS OF THE SYSTEM

A fluidized bed reactor simulation exhibits fast and transient dynamics with many physically possible trajectories, i.e., fluidized bed reactor trajectories are chaotic. This means that – also for numerical solvers – starting a fluidized bed simulation with different initial particle packings will yield different trajectories. Still, in the limit of long simulation trajectories, different initial packings do not affect the temporal statistics. Due to this phenomenon, we use time-averaged statistics for quantitative comparisons, see Figure 10 and Figure 11. Precise step-by-step comparison is not feasible due to numerical differences that naturally arise during a rollout. In Figure 8, we show a visual comparison of the iso-surface of solid fraction, solid fraction field, and fluid velocity for two different inlet velocities. When the two time series are compared to the ground truth snapshots they look very similar, especially the bubble structure visualized by the iso surface on the solid fraction field.

Notice how, with low inlet velocity, a single bubble of air forms inside the dense particle bed. When the bubble reaches the surface, a ripple-like structure is formed by the particles, clearly displaying their fluidization. With high inlet velocity, the behavior is much more unorganized and a complicated bubble structure inside the particle bed arises. NeuralDEM handles both regimes successfully, modeling the organized and structured low-velocity bubble dynamics as well as the chaotic, high-velocity particle interactions, accurately capturing the transitions in flow patterns and complex fluidization behavior across different regimes.

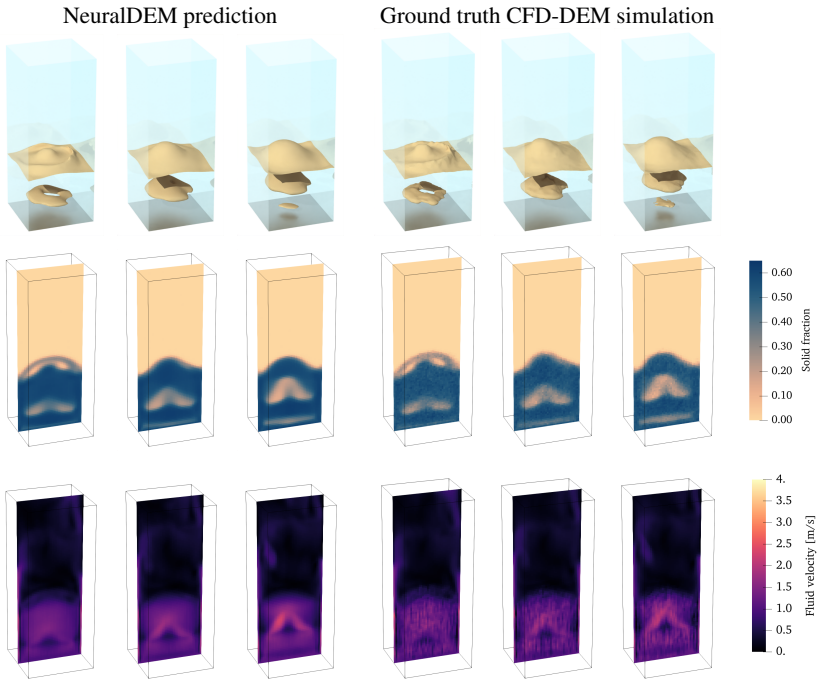
The fluid velocity within the particle bed shows distinct patterns influenced by particle motion and fluid-particle interactions. At low inlet velocity, the fluid flow is relatively stable, with modest spatial gradients around the rising air bubble. In contrast, with high inlet velocity, the fluid velocity across the bed is higher compared to the previous case, not only due to the high inlet velocity but also because of the dynamic movement of particles within the bed. These variations result in complex flow fields, with strong fluctuations that are challenging to model. NeuralDEM accurately captures these dynamics and replicates the different velocity profiles for all inlet velocity regimes between the scenarios shown here.

B.2 PHYSICS EVALUATION

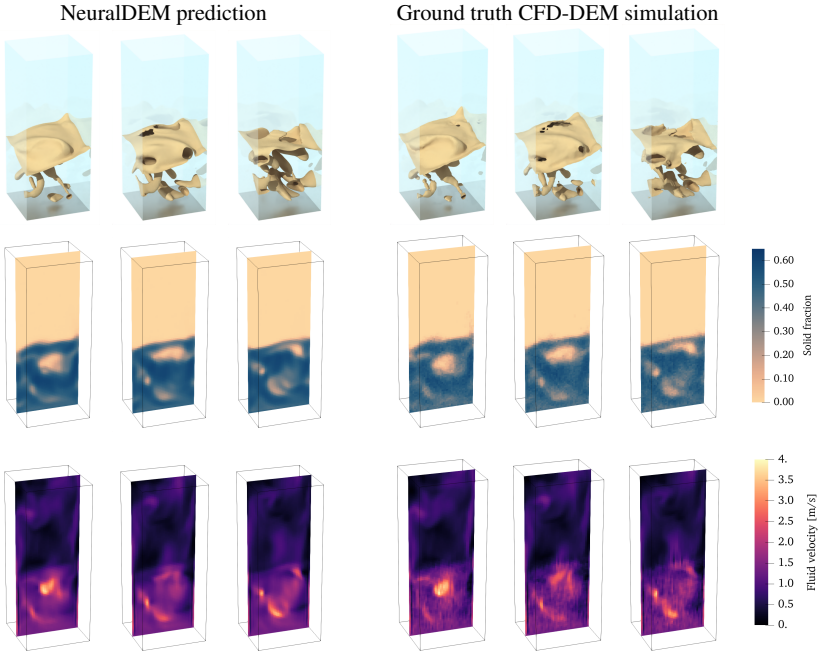
NeuralDEM's capability to maintain stability and accuracy over extended simulation periods is essential for realistic modeling of particle-fluid systems. We demonstrate the long-term rollout stability by analyzing the time average solid fraction and fluid velocity field for 2.8k steps, representing $28\,\mathrm{s}$ of physical simulation. The length of the trajectory resulted from a $30\,\mathrm{s}$ CFD-DEM simulation, sufficient to determine the long-term statistics. NeuralDEM is not limited to this length and did not show any stability problem up to $100\,\mathrm{s}$. To the best of our knowledge, these are the longest reported stable rollouts in the deep learning-based 3D physical simulation field (Lippe et al., 2023; Kochkov et al., 2021; Sanchez-Gonzalez et al., 2020).

In Figure 10 we show the mean and variance of the magnitude of the velocity and the solid fraction over time for a given mesh cell. In the figure, we display the central slice along the y-axis (aka the depth of the reactor). NeuralDEM successfully captures the different long-term behavior of the particles and the fluid when different inlet velocities are used.

Possibly the most crucial property of a physics simulation is mass conservation. In a CFD-DEM simulation, as long as the particles stay within the domain, the overall mass will not change. In Figure 11 we show how NeuralDEM, although it does not model each particle independently, maintains very good mass conservation for all timesteps, showing no drift throughout the predicted trajectory.

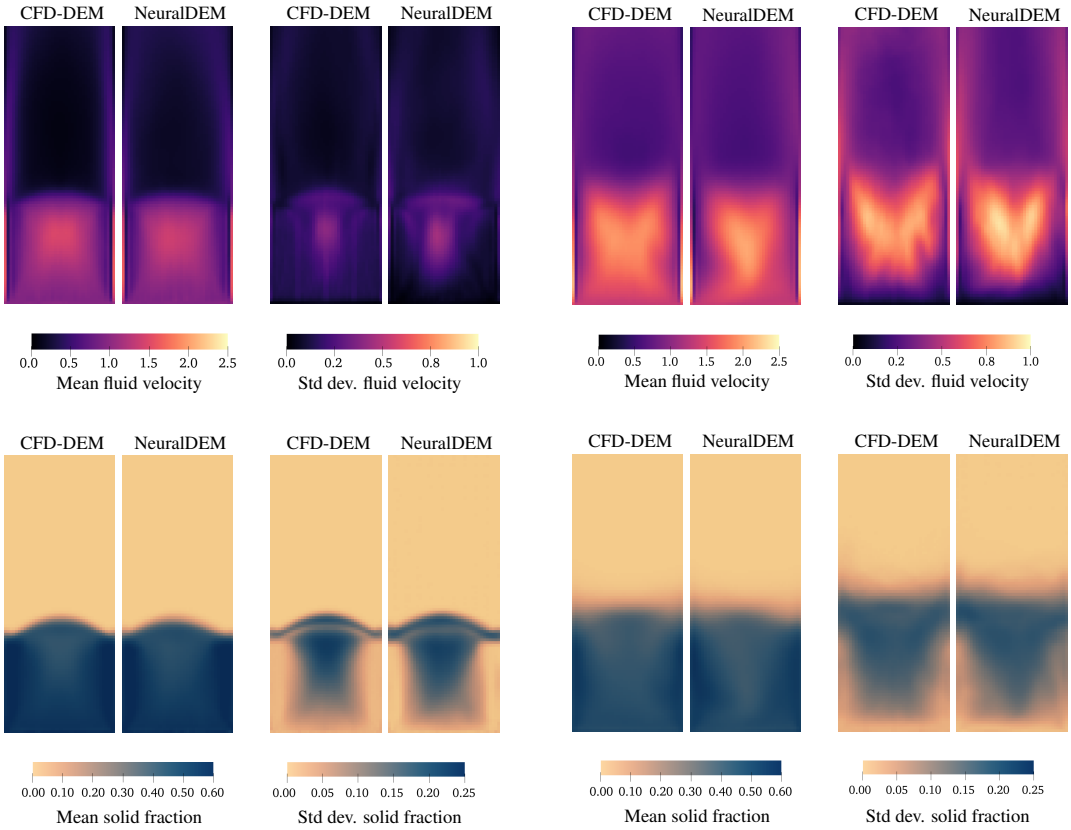


(a) Low inlet velocity



(b) High inlet velocity

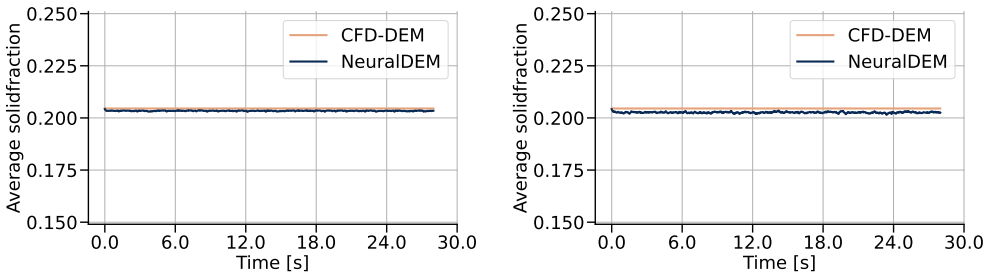
Figure 9: Visualization of three snapshots taken at $0.06\,\mathrm{s}$, $0.09\,\mathrm{s}$ and $0.12\,\mathrm{s}$ for two different inlet velocities (a) $0.45\,\mathrm{m\,s^{-1}}$ and (b) $0.7\,\mathrm{m\,s^{-1}}$. The first row shows iso-surfaces of solid fraction at 0.35 uncovering the emerging bubble structure of fluidized bed reactors. The second and the third show the central slice along the y-axis for the solid fraction and the magnitude of fluid velocity, respectively. Here, NeuralDEM uses the same initial conditions as the CFD-DEM simulation and faithfully reproduces the evolution of the system in the first few steps of the simulation.



(a) Low inlet velocity

(b) High inlet velocity

Figure 10: Comparison of long term temporal averaging statistics for two fluid inlet velocities. Slices along the y-axis of the mean and the standard deviation of the magnitude of the velocity field and the solid fraction field shown. Each pairs shows the CFD-DEM simulation and the NeuralDEM model, respectively. NeuralDEM predictions closely match the long-term statistics of both slow and fast inlet velocity regimes, where spatial gradients vary from small to very high.



(a) Low inlet velocity

(b) High inlet velocity

Figure 11: The average solid fraction predicted by the NeuralDEM model is stable over extremely long rollouts that are almost $10\times$ the length of the $3\,\mathrm{s}$ sequences used for training. Notably, although the output of NeuralDEM models a field, the mass of the entire system is preserved almost perfectly. The y-axis ranges from 0.15 to 0.25 to show more detail but the solid fraction can range from 0 to 1.

C ARCHITECTURE DETAILS

Figure 12 shows the inner workings of a multi-branch neural operator block. The primary quantities are processed via a multi-modal diffusion transformer (MMDiT) architecture (Esser et al., 2024) and additional secondary quantities use an adapted MMDiT design with a stop gradient before concatenating the sequences for the k and v matrices of the attention.

For encoding, we use a supernode pooling layer (Alkin et al., 2024) for particles and a patch embedding layer (Dosovitskiy et al., 2021) for the fluid phase (as it is discretized to a regular grid). Similarly, we employ a UPT position-wise decoder (Alkin et al., 2024) that decodes a particle/patch by using its embedded location in the domain as query to the decoder.

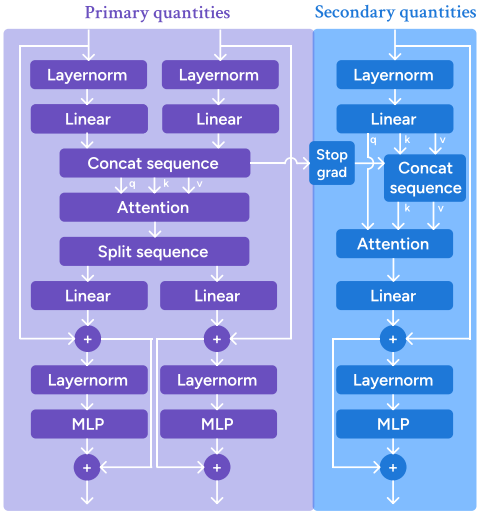


Figure 12: Schematic architecture of a *multi-branch transformer block*. DiT (Peebles & Xie, 2023) modulation is applied to each attention and MLP block but is omitted for visual clarity.

D EXTENDED DATASET DETAILS

D.1 HOPPER

Hoppers are industrially used for short as well as long term storage of particulate material, showcasing slow and pseudo-steady macroscopic behavior. DEM is the preferred method since the air around the particles can usually be neglected due to the slow velocities in the system. In our experiments, the hopper geometry, as shown in Figure 4a, is filled with 250k particles, which gradually exit the domain over the simulation duration when the hopper empties. Timestepping of DEM solvers strongly depends on particle size as well as particle properties. A timestep of $\Delta t_{\text{DEM}} = 10\,\mu\text{s}$ is required for the tested numerical experiments with LIGGGHTS (Kloss et al., 2012).

We do not refill any new particles into the hopper. Different simulations in the dataset vary hopper geometry and particle friction as specified in Table 2. This variability in simulation parameters results in different flow regimes ("funnel flow" or "mass flow") where the dynamics are slow and pseudo-steady. In funnel flow, particles primarily move down a funnel above the outlet, whereas in mass flow, material moves uniformly down towards the outlet, see Figure 8.

Simulating a granular flow of 250k particles through a hopper, with a trajectory spanning or 4M numerical timesteps, requires 3 hours on 16 cores of high-performance CPUs when using traditional DEM. In contrast, on a single state-of-the-art GPU, the fastest NeuralDEM inference model faithfully reproduces the physics rollout in just . Notably, NeuralDEM requires significantly fewer timesteps compared to the numerical simulation. Further acceleration is achieved through NeuralDEM's field-based output representation, which reduces the number of required output points while still capturing the macroscopic bulk behavior accurately. When processing all 250k output points

on a single GPU, NeuralDEM inference takes $8\,\mathrm{s}$. Running NeuralDEM on the same 16 CPUs used for the numerical simulation – notably leveraging the benefits of reduced outputs due to field-based representations – results in a trajectory rollout of $41\,\mathrm{s}$. Those inference times are shorter (GPU) than or in the same order of magnitude (16 CPUs) as the trajectory duration of $40\,\mathrm{s}$, highlighting the feasibility of real-time simulations. While traditional DEM could theoretically be further parallelized, for a certain thread number hardware communication bottlenecks present a limiting factor, and thus further underlines the potential of NeuralDEM.

Name	Description	Range	Sampling
α_{HO}	Angle of the slope towards the outlet	[0°, 60°]	LHS
μ_{s}	Particle sliding friction	[0.05, 1.00]	LHS
μ_{r}	Particle rolling friction	[0.00, 0.50]	LHS
θ	Angle of internal friction	evaluated	-
ffc	Flow function coefficient	evaluated	-

Table 2: Parameters of the hopper dataset.

D.2 Fluidized bed

Fluidized bed reactors are characterized by fast and transient phenomena and are widely used in industry for a variety of processes. Fluidized bed reactors showcase strong interactions of the particles with the surrounding fluid, necessitating an accurate modeling of particles, the gas phase, as well as particle-gas interactions. Thus, modeling approaches need to combine DEM parts with simulations of the surrounding fluid. For data generation, we use a coupled CFD-DEM approach (Goniva et al., 2012) which is built upon LIGGGHTS (Kloss et al., 2012) and OpenFOAM (Weller et al., 1998). The geometry of the setup is sketched in Figure 4b and the dimensions for both cases can be found in Table 4c. The reactor is filled with 500k particles and the fluid, i.e., air, that is uniformly pushed into the reactor from the bottom is modeled on a grid of 160k hexahedral cells.

Numerically, the reactor is filled with 500k particles, and the air that is uniformly pushed into the reactor from the bottom is modeled on a grid of 160k hexahedral cells. The following numerical experiments are carried out on a fixed-geometry reactor with varying inlet velocities, and we evaluate short-term behavior and long-term statistics. In total, 76 difference inlet velocities sampled uniformly from $0.337\,\mathrm{m\,s^{-1}}$ to $0.842\,\mathrm{m\,s^{-1}}$ were used. Additionally, 6 different initial random particle packings per inlet velocity were sampled, resulting in 456 CFD-DEM trajectories.

The classical solver requires a much finer time resolution where $\Delta t_{\mathrm{ML}} = 4000\Delta t_{\mathrm{DEM}} = 40\Delta t_{\mathrm{CFD}}$.

For testing the long rollout performance of NeuralDEM (see Figure 11, we further generate 4 sequences with a physical duration of $28\,\mathrm{s}$ and different inlet velocities, where new random initial packings are applied. Those trajectories result in sequences of 2800 timesteps.

E Implementation details

E.1 Hopper

The considered hopper simulations exhibit slow and pseudo-steady dynamics, resulting in a state that is well-defined from the scalar input parameters (timestep, particle friction, hopper angle) alone. Therefore, it is neither necessary to encode the previous state nor to have interactions between branches, as the scalar input parameters already provide full information about the state to all branches. Therefore, we opt for a decoder-only architecture consisting of a single transformer block per branch that starts from 32 static learnable latent tokens and prepares them for decoding via a perceiver cross-attention block that takes positional embeddings as queries and uses the latent tokens as keys and values. Both the transformer and the cross-attention block use DiT (Peebles & Xie, 2023) modulation to incorporate the scalar parameters (timestep, α_{HO}, μ_s and μ_r). The whole model consists of 50M parameters.

We train models in the hopper setting for 10k updates using a batchsize of 256, a peak learning rate of 10^{-4} which is warmed up for 1k updates followed by a decaying cosine schedule afterwards. The loss is summed for all branches and LION (Chen et al., 2023) is used as optimizer.

E.2 FLUIDIZED BED

Modeling a fluidized bed reactor requires interaction between fluid and particles. To this end, we utilize both grid data and particle data as input to the model. As described in Section C, we use specialized designs for different physics phases. Namely, ViT (Dosovitskiy et al., 2021) patch embedding for grid data and UPT supernode pooling (Alkin et al., 2024) for particle data. The number of supernodes and patches can be changed after training, enabling stable training and a flexible choices for generating trajectory rollouts. Rollouts are carried out in autoregressive fashion, where each timestep is used as input for the next timestep. During training, the model receives data from a random timestep t, is conditioned on the inlet velocity, and is supervised with the quantities from timestep $t + \Delta t_{\text{ML}}$. The inlet velocity conditioning is performed analogously to the hopper setting, with a DiT (Peebles & Xie, 2023) modulation. The particle mixing is modeled as an off-branch quantity.

The final model has roughly 850M trainable parameters in total where we use a standard DiT (Peebles & Xie, 2023) conditioning mechanism which adds a lot of parameters that do not contribute a significant amount of FLOPS to the model. We use 12 multi-branch transformer blocks in total with hidden dimension 768, where each of the 3 branches would correspond to a ViT-Base (Dosovitskiy et al., 2021) (86M parameters) if no DiT modulation was applied. The model processes sequences with length of 2048 tokens for the particle displacement, 2500 for the fluid velocity and particle solidfraction, and 2500 for the particle mixing, both with $4 \times 4 \times 4$ patches. The model is trained 126k updates using a batch size of 128, a peak learning rate of 4×10^{-5} which is warmed up for 13k updates followed by a decaying cosine schedule to 10^{-7} using LION (Chen et al., 2023) as optimizer with 0.5 as weight decay. The loss is summed over all branches.