Liquid–Liquid Dispersion Performance Prediction and Uncertainty Quantification Using Recurrent Neural Networks

Fuyue Liang, Juan Valdes, Sibo Cheng, Lyes Kahouadji, Seungwon Shin, Jalel Chergui, Damir Juric, Rossella Arcucci, Omar Matar

Liquid-liquid dispersion performance prediction and uncertainty quantification using recurrent neural networks

Fuyue Liang,[*,†] Juan P. Valdes,[†] Sibo Cheng,[‡] Lyes Kahouadji,[†] Seungwin Shin,[¶] Jalel Chergui,[§] Damir Juric,[§,‖] Rossella Arcucci,[⊥] and Omar K. Matar[†]

[†]*Department of Chemical Engineering, Imperial College London, London, UK*

[‡]*Data Science Institute, Imperial College London, London, UK*

[¶]*Department of Mechanical and System Design Engineering, Hongik University, Seoul, Republic of Korea*

[§]*Université Paris Saclay, Centre National de la Recherche Scientifique (CNRS), Laboratoire Interdisciplinaire des Sciences du Numérique (LISN), Orsay, France*

[‖]*Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Cambridge, UK*

[⊥]*Department of earth science & engineering, Imperial College London, London, UK*

E-mail: fuyue.liang18@imperial.ac.uk

Abstract

We demonstrate the application of a Recurrent Neural Network to perform multi-step and multivariate time-series performance predictions for stirred and static mixers as exemplars of complex multiphase systems. We employ two Long-Short-Term Memory (LSTM) frameworks in this study which are trained on high-fidelity, three-dimensional, computational fluid dynamics simulations of the mixer performance, in the presence

and absence of surfactants, in terms of drop size distributions and interfacial areas as a function of system parameters; these include physico-chemical properties, mixer geometry, and operating conditions. Our results demonstrate that whilst it is possible to train a LSTM with a single fully-connected layer more efficiently than a LSTM Encoder-decoder, the latter is shown to be more capable of learning the dynamics underlying dispersion metrics. Details of the methodology are presented, which include data pre-processing, LSTM model exploration, methods for model performance visualisation; an ensemble-based procedure is also introduced to provide a measure of the model uncertainty. The workflow is designed to be generic and can be deployed to make predictions in other industrial applications with similar time-series data.

Introduction

Multiphase dispersion processes, and in particular liquid-liquid (L-L) mixing, are of central importance to a broad range of industrial applications, ranging from microscopically-manufactured ('structured') emulsions in the manufacturing of fast-moving consumer goods and pharmaceuticals, to chemical reactions (e.g., nitration, sulfonation, etc.) in the energy sector.[1–3] These operations greatly depend on several key performance indicators such as the interfacial area of the dispersed phase governing mass transfer-controlled reaction rates, as well as the droplet size distribution (DSD) and count determining the stability and physical properties of emulsions. Consequently, numerous studies have focused on developing predictive (semi)-empirical correlations to estimate metrics such as the mean/maximum drop size based on a given set of flow conditions and fluid properties[4–8] and a broad range of mixing devices, designs, and flow regimes.[9–14] Lately, robust numerical frameworks have been developed aiming to improve the predictive capabilities of previous empirical models by providing a physics-based understanding of the governing phenomena via Computational Fluid Dynamics (CFD) simulations and Population-Balance Modelling (PBM).[15–18]

While it is true that substantial ground has already been covered, both experimentally and

numerically, multiple challenges still exist when modelling complex and industrially relevant mixing processes. A prevalent scenario in L-L systems is the presence of surface-active agents (surfactants), originating either as contaminants or additives. Such systems require a much more comprehensive computational framework to accurately describe the dispersion dynamics unfolding, such as the inclusion of equations of state and surfactant mass transport modelling to account for the intertwined effect between interfacial tension and surfactant concentration. Analogous non-idealities in multiphase mixing flows, such as highly concentrated, turbulent or non-Newtonian systems, lead to a similar challenge. Consequently, this situation gives rise to a challenging trade-off between model robustness and accuracy on the one hand, and resource consumption on the other. Therefore, it is unsurprising that no general model has been established thus far to provide sufficiently accurate predictions of the dispersion performance under analogous scenarios.

Fortunately, the rapid development of Artificial Intelligence (AI) and data-driven techniques has granted us access to a powerful and cost-effective toolkit of computational alternatives to circumvent the challenges set out above. In recent years, with the increase in available simulation data, Machine Learning (ML) has become a popular method to accelerate CFD simulations in multiple fields, such as chemical engineering,[19] built environment[20] and so forth. Among the different types of ML-based techniques, Recurrent Neural Networks (RNNs) stand out due to their capability to handle sequential data. Specifically, its variant, Long Short-Term Memory (LSTM), which is designed to solve the so-called vanishing problem[21,22] seen in the conventional RNNs, has been commonly used in ranges of applications. For example, neural networks with LSTM embedded have been trained using simulation data to perform prediction of scenarios including material leakage position[23,24] and bio-oil yield of fluidized bed.[25] Moreover, the LSTM architecture has been coupled with Reduced Ordered Modelling (ROM) to model the key features underlying turbulent flows.[26] Similarly, combination of the two frameworks has been applied to carry out transonic aeroelastic analysis.[27] More recently, novel models based on LSTM have been proposed to forecast the hydrodynamics of

submarine prototypes[28] and the behaviour of ocean waves.[29]

Inspired by the prior work reviewed in the foregoing, this study seeks to develop an inexpensive time-series model using RNNs by capitalising on a comprehensive set of high-fidelity three-dimensional CFD simulations, some of which have been exploited in recent works[30–33] to unravel the fundamental governing mechanisms underlying extensively utilised mixing systems handling L-L dispersions across a range of industrially-relevant scenarios. These simulations have been conducted with a state-of-the-art code, which comprises a hybrid Front-tracking/Level-set interface-tracking algorithm, embedded along a well-validated multiphase solver for surfactant transport at the interface and in the bulk phase.[34] This framework unlocks an unprecedented level of detail on the interfacial dynamics unfolding and thus provides an accurate, physics-based estimation of the temporal evolution of key performance metrics, such as interfacial area growth, drop generation, and DSD. The works by Liang at al. 2022, 2023 explored a pitch-blade stirred vessel mixer with varying impeller speeds and different surfactant profiles, whereas those by Valdes et al. 2023a, 2023b ran simulations with a SMX static mixer considering different inlet configurations and types of surfactants.

The time-series model implemented in this work aims to supersede the CFD model in predicting the key dispersion metrics previously mentioned based solely on their initial behaviour during the early stages of the process. More importantly, we intend to investigate the model's capability to identify different initial performance signatures, inherently linked to mixer design, operational conditions or surfactant physicochemical profile, and extrapolate their future behaviour accordingly. To achieve this, we develop a general predictive workflow around three mixing performance metrics, with the implementation of LSTM at its core, as shown in Figure 3. The trained model is then initialised with early-stage CFD data and set to self-iterate on its output in a 'rollout' procedure to generate future performance predictions. Finally, we perform an uncertainty quantification analysis on the trained model via ensemble perturbation to track the evolving model uncertainty and its performance

through the prediction propagation via 'rollout' .

The rest of this paper is organised as follows: Section 2 covers the main theoretical concepts of RNNs and LSTM relevant to our work; Section 3 presents the overall framework development and deployment, from CFD data acquisition, pre-processing and re-conditioning, to model training, tuning, and sequence generation via rollout methodology, discussing two separate model architectures and exploring their accuracy and uncertainty. The predictions vs. CFD, which are treated as the ground truth data generated for both stirred vessels and static mixers and the corresponding discussion around the model's explainability, are presented in Section 4. Finally, concluding remarks are given in Section 5.

Theoretical background: RNNs and LSTM

We aim to perform sequence prediction via deep learning, which is different from the other types of learning problems since it imposes an order on the observations that must be preserved for model training and deployment. Recurrent neural networks (RNNs) are specifically designed to address such problems where they have loops that allow the information from one-time-step to be passed to the next. As shown in Figure 1a, when an RNN is trained to predict a dynamical quantity in the future from the past, the network incorporates the information from the input observations $\mathbf{x}$ into the hidden state $\mathbf{h}$, which is passed forward through time. The circuit depicts that the current state is fed back into the network influencing its future state, and the black square indicates that such interaction takes place with a delay of one-time-step. Once the whole historical sequence, $\mathbf{x}_t$, is scanned, the RNN learns a summary, $\mathbf{h}_t$, of the relevant aspects in the past up to time t. Following this, an extra architecture layer, known as *output layer*, will be added to read information out of the $\mathbf{h}_t$ to make predictions.

A traditional RNN has only one hidden state, $\mathbf{h}_t$, which is sensitive to short-term inputs. However, when the sequence length grows, it becomes unable to learn the summary that connects the current state with the state further back in time.[35] To address this issue,

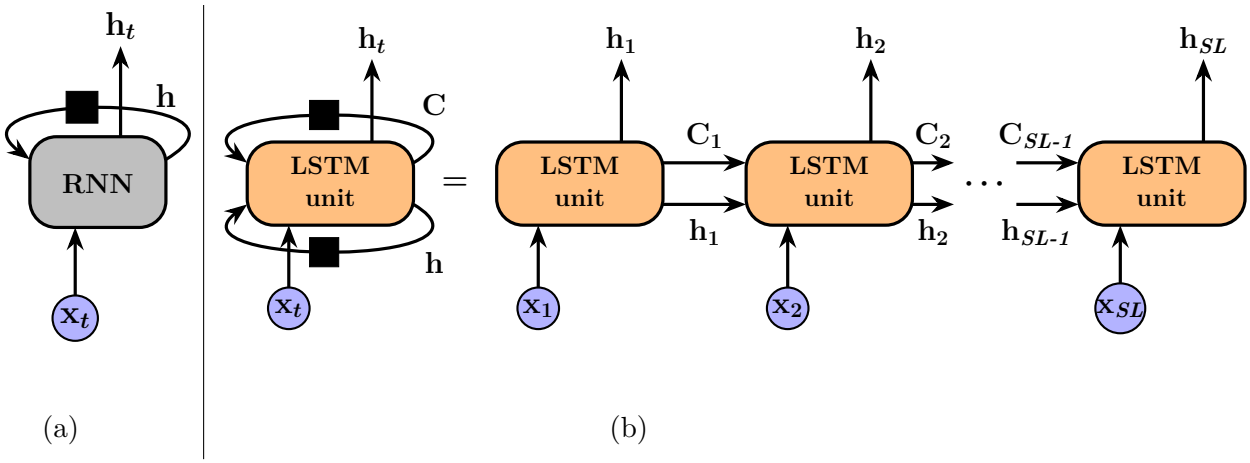


Figure 1: (a) Circuit diagram of a RNN with no output layers; (b) Circuit diagram (left) and its unrolled view (right) until a final time-step labelled as sequence length (SL) of a LSTM network.

Long Short-Term Memory (LSTM) networks harness a cell state $\mathbf{C}_t$ to store the long-term information, as shown in Figure 1b. As with RNNs, the information flows through time in LSTM networks, but both short-term and long-term memory are carefully carried via hidden state and cell state, respectively. Several variants of the LSTM unit have been developed[36,37] since the LSTM unit was first proposed.[38] The *standard* LSTM[36] is used in the current study as it has been proved to outperform other variants[39] and has been used extensively over a broad range of applications.[40,41]

The core advantage of LSTM is its ability to control information deletion from or addition to the cell state. As depicted in Figure 2, the cell state runs through the LSTM unit at the top of the diagram with some minor interactions, indicating the information stored in the cell state could flow along with slight or no changes. These interactions are carefully regulated by different gates.[38] The role of the first one, the *Forget Gate*, is to decide how much information from $\mathbf{C}_{t-1}$ should be discarded. This decision is made by a Sigmoidal activation function $\sigma(x) = 1/(1+e^{-x})$ looking at the previous hidden state, $\mathbf{h}_{t-1}$, and the current input, $\mathbf{x}_t$:

$$f_t = \sigma(\mathbf{W}_f \mathbf{h}_{t-1} + \mathbf{W}_f \mathbf{x}_t + b_f). \tag{1}$$

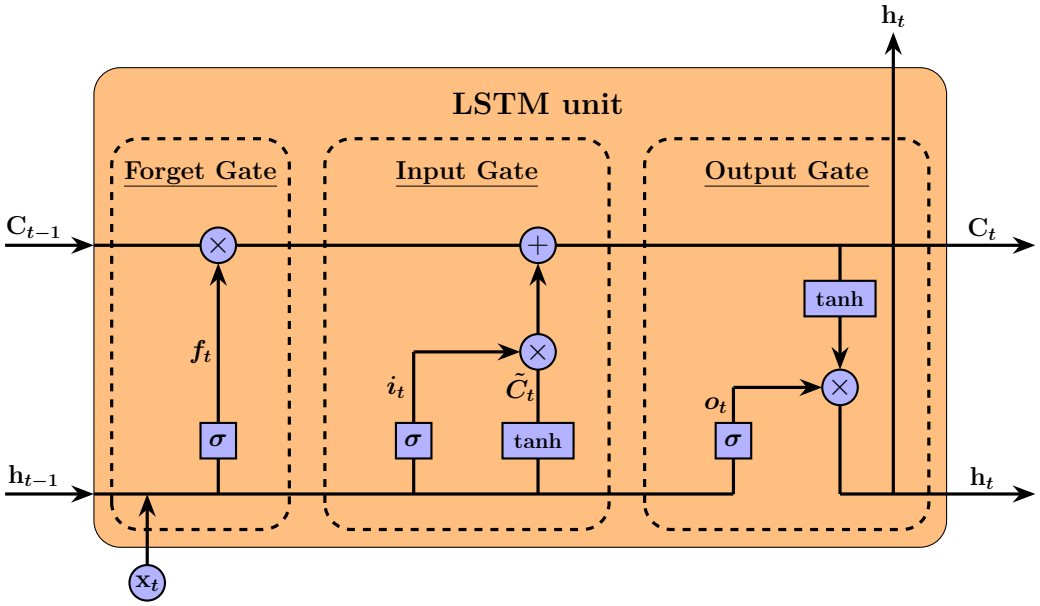


Figure 2: Diagram of different gates in one *standard* LSTM unit.

The subsequent *Input Gate* determines the new information to be memorised in $\mathbf{C}_t$:

$$\begin{aligned} i_t &= \sigma(\mathbf{W}_i \mathbf{h}_{t-1} + \mathbf{W}_i \mathbf{x}_t + b_i), \\ \tilde{C}_t &= \tanh(\mathbf{W}_C \mathbf{h}_{t-1} + \mathbf{W}_C \mathbf{x}_t + b_C). \end{aligned} \tag{2}$$

Herein, the candidate cell state, $\tilde{C}_t$, is introduced to describe the current input. Later, the cell state is updated by:

$$\mathbf{C}_t = f_t \odot \mathbf{C}_{t-1} + i_t \odot \tilde{C}_t, \tag{3}$$

where $\odot$ denotes the Hadamard product of vectors and matrices. Lastly, the *output gate* is applied to settle the final output of the LSTM cell:

$$\begin{aligned} o_t &= \sigma(\mathbf{W}_o \mathbf{h}_{t-1} + \mathbf{W}_o \mathbf{x}_t + b_o), \\ \mathbf{h}_t &= o_t \odot \tanh(\mathbf{C}_t), \end{aligned} \tag{4}$$

which contains the $\mathbf{h}_t$, a cache of the recent information from $\mathbf{h}_{t-1}$, and long-period aspects from $\mathbf{C}_t$. In the equations above, $\mathbf{W}$ and b are trainable parameters of the LSTM unit,

representing the weight vectors and the offset term for different gates, respectively. These equations are computed for each time-step, and hence with a subscript t denoting the time-step. As described above, LSTM is capable of learning aspects in long-time sequences and thus has been used in modelling dynamical systems.[23–25] Detailed architecture of the model employed in the current study will be presented in the following section.

Methodology

The computational framework presented in this work adheres to a three-stage workflow consisting of 1) Data acquisition, 2) Data re-conditioning, and 3) Model training, deployment and explainability, as illustrated in Figure 3 and detailed throughout this section. At its core, we constructed a multivariate LSTM model capable of carrying out multi-step predictions of the temporal evolution of key multidimensional dispersion performance metrics, whilst remaining agnostic to the specifics around the mixing process itself. It is worth noting that two separate LSTM models were trained, one for each mixing device. This separation stemmed from a disparity in the time-series dimensions between mixing systems (stirred vessel cases have three times as many time-steps in comparison), which poses a challenge for the model to handle, as we will discuss further in this section.

Data acquisition

Problem statement: CFD simulations of mixing systems

This study utilises high-fidelity CFD data of two widely employed mixing systems in the industry: stirred tanks and static mixers—handling two-phase L-L flows. Each mixer operates in a completely different flow regime, with the former handling transitional/turbulent flows ($\mathrm{Re} = \rho N D_r^2/\mu \approx [9000, 18000]$), and the latter operating under laminar conditions ($\mathrm{Re} = \rho U_r D_r/\mu = 1.63$). For brevity, specifics on the problem formulation of each system will not be described herein, but readers are encouraged to refer to previous publications detailing

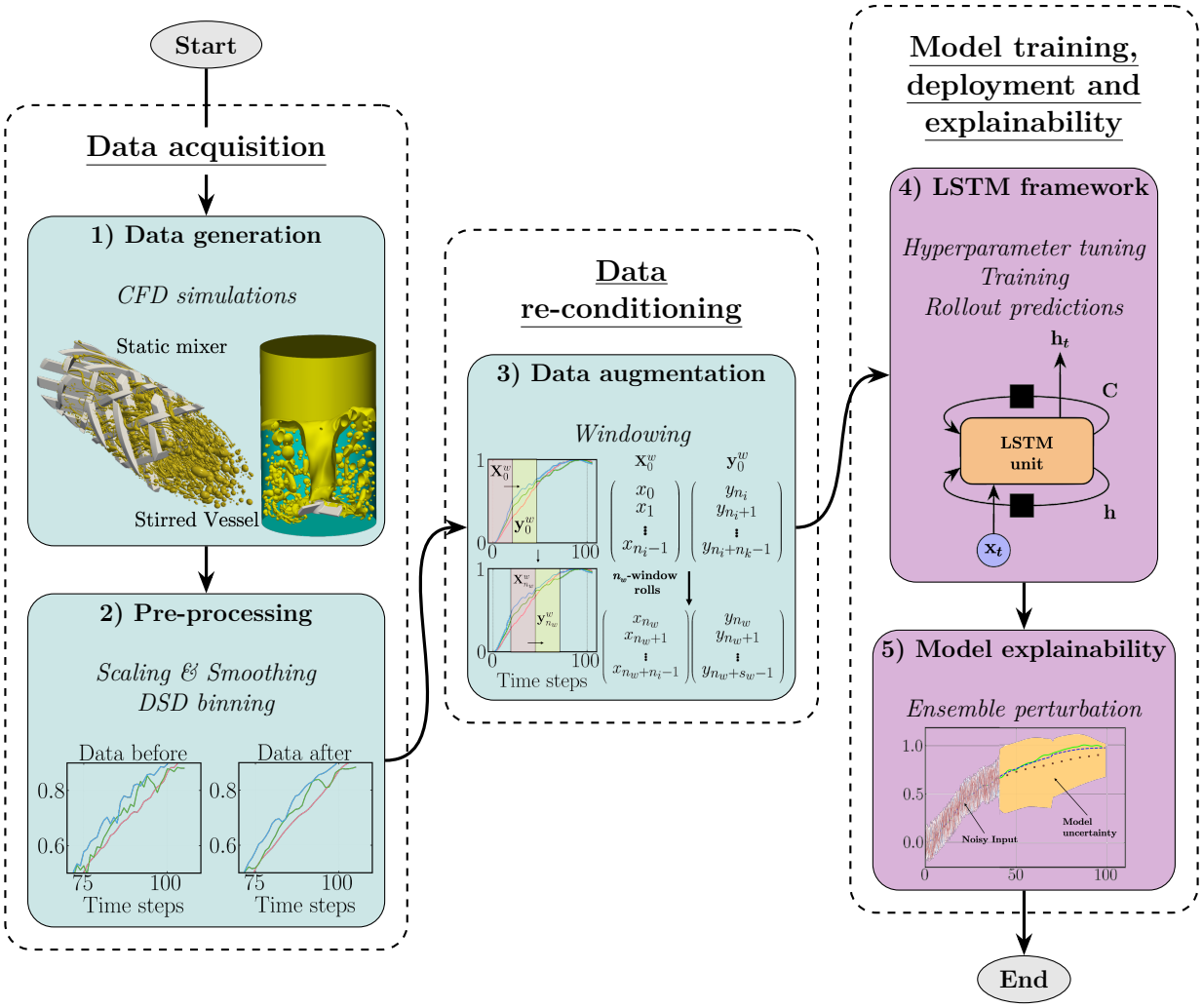


Figure 3: Flowchart detailing the overall framework architecture developed to train and deploy a multivariate multi-step LSTM model with two-phase mixing performance data from high fidelity CFD simulations. Colour-coded blocks refer to general data pre-processing (teal) and model operations and exploration (violet).

the geometrical and operational specifications, fluid properties, numerical considerations (e.g., grid refinement) and validation.[30–33] The extracted datasets comprise multidimensional time-series data encompassing three key metrics integral to the dispersion performance: interfacial area growth (IA), drop count (ND) and droplet size distribution (DSD), calculated as the approximate volume of cells resolving a fully-detached structure or 'drop'. The choice of these parameters capitalises on the explicit and robust nature of the interface-tracking scheme (Level-Contour Reconstruction Method, LCRM) embedded in the CFD code used, which

furnishes a more accurate and well-resolved representation of the intricate interfacial dynamics compared to other traditional schemes (e.g., level-set methods).[42]

A comprehensive set of 43 simulations was performed: fourteen cases involved stirred vessels, exploring various rotational speeds (N_{rot}) and surfactant profiles, while the remaining 29 cases focused on static mixers, investigating different inlet configurations, geometry arrangements, and types of surfactants. While the former 14 cases are divided between clean (varying N_{rot}) and surfactant-laden systems, the latter 29 overlap inlet setup and geometry arrangement with clean and contaminated scenarios. The specific parameters, combinations, and value ranges explored for each set of cases are detailed in Table 1.

Simulations were carried out using in-house code BLUE,[34,42,43] which considers a three-dimensional single-field formulation of the Navier-Stokes equations in a Cartesian domain:

$$\nabla \cdot \mathbf{u} = 0, \tag{5}$$

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = -\nabla P + \rho \mathbf{g} + \nabla \cdot \left[\mu (\nabla \mathbf{u} + \nabla \mathbf{u}^T) \right] + \mathbf{F}, \tag{6}$$

where t, P, $\mathbf{u}$, $\mathbf{g}$, and $\mathbf{F}$ denote time, pressure, velocity, gravity, and a local surface force. For clean systems, this term follows a hybrid formulation of the form,

$$\mathbf{F} = \sigma \kappa_H \nabla \mathcal{H}, \tag{7}$$

where σ is a constant interfacial tension coefficient, κ_H is twice the mean interface curvature field, and $\mathcal{H}(\mathbf{x}, t)$ stands for a numerical Heaviside function, generated through a vector distance function from the interface $\varphi(\mathbf{x})$, and solved numerically with a smooth 3-4 grid cells transition.[32,43] This same function is used to define density and viscosity throughout the domain, which are respectively given by

$$\begin{aligned} \rho(\mathbf{x}, t) &= \rho_a + (\rho_o - \rho_a)\, \mathcal{H}(\mathbf{x}, t), \\ \mu(\mathbf{x}, t) &= \mu_a + (\mu_o - \mu_a)\, \mathcal{H}(\mathbf{x}, t), \end{aligned} \tag{8}$$

Table 1: Simulation cases considered in this study, categorised into three broad groups according to the focus of each study: varying surfactant properties, modifications in the operating conditions and different mixing element arrangements, specific to static mixers.

Mixer/Case	**Stirred Mixer**	**# Cases**	**Static Mixer**	**# Cases**
Surfactant-laden	Bi = [0.001,1]	5	3-drop inlet	
	β = [0.5,0.9]	3	Bi = [0.01,1]	3
			β = [0.3,0.9]	3
			Da = [0.01,1]	3
Operational configuration	Rotational speed (Cl)		Clean pre-mix (pm)	
			Inlet = [Coarse, Fine, 3-drop]	3
	N_{rot} (Hz) = [5,10]	6	Surfactant-Laden (Coarse)	
			Bi_{pm} = [0.01,0.1]	2
			β_{pm} = [0.6,0.9]	2
			Da_{pm} = [0.1]	1
Geometrical arrangement	N/A	N/A	Clean	
			Coarse pm = [Alt1, Alt2, Alt3]	3
			Fine pm = [Alt4]	1
			Surfactant-Laden	
			· 3-drop inlet (Alt 4):	
			β_{alt4} = [0.3,0.9]	3
			Bi_{alt4} = [0.01,1]	3
			· Pre-mixed inlet:	
			$\text{Coarse}_{\beta=0.9}$ = [Alt1, Alt4]	2

where $\mathcal{H}(\mathbf{x}, t)$ has a value of 1 for the oil phase (subscript o) and zero for the aqueous phase (subscript a). Additional terms are added to the formulation in Equation 6 when dealing with stirred tanks. Firstly, a Smagorinsky-Lilly LES turbulence model is implemented, adding the filter $\left(\mu_{\text{vis}} + \rho C_s^2 \Delta^2 \left|\bar{S}\right|\right)$ to the dissipation term, and secondly a Direct Forcing Method is added with the inclusion of a fluid-solid interaction force, $\mathbf{F}_{\text{fsi}}$.[30]

In the presence of surfactants, the force $\mathbf{F}$ is decomposed into its normal ($\sigma\kappa\mathbf{n}$) and

tangential components ($\nabla_s\sigma$), as shown in Equation 9,

$$\mathbf{F} = \int_A [\sigma\kappa\mathbf{n} + \nabla_s\sigma]\delta(\mathbf{x} - \mathbf{x}_f)dA, \tag{9}$$

where κ denotes the interface curvature, ∇_s stands for the surface gradient operator, and $\mathbf{n}$ is the normal unit vector pointing away from the interface. The 3D Dirac delta function $\delta(\mathbf{x} - \mathbf{x}_f)$ is set to 0 everywhere except at the interface, which is located at $\mathbf{x} = \mathbf{x}_f$.[33] In surfactant-laden cases, σ is no longer constant but is modelled through a Langmuir EoS of the form $\sigma = \sigma_{\text{cl}} + \mathcal{R}T\Gamma_\infty \ln\left(1 - \frac{\Gamma}{\Gamma_\infty}\right)$, where Γ refers to the surfactant surface concentration. The chemical nature of the surfactant is parameterised by the following dimensionless numbers

$$\beta = \frac{\mathcal{R}T\Gamma_\infty}{\sigma_{cl}}, \ \text{Bi} = \frac{k_d L_r}{U_r}, \ \text{Da} = \frac{\Gamma_\infty}{L_r C_\infty}, \tag{10}$$

where β, Bi, and Da stand for the elasticity, Biot, and Damkohler numbers, characterising surfactant 'strength' (interfacial tension sensitivity on concentration), desorptive capability vs. convective surface transport, and adsorption depth into the bulk, respectively.[33] These parameters are used to label the surfactant-laden cases employed herein and broadly describe the nature of each surfactant modelled (e.g., highly/weakly adsorptive/desorptive, etc). Further details into the equations governing surfactant transport and the specifics behind the interface-tracking algorithm (i.e., LCRM) are found in previous publications.[31,33,34,42]

Data pre-processing: Scaling, smoothing and DSD binning

The initial set of features considered in this study consists of two scalar quantities, namely IA and ND, and a variable-sized list of scalars (i.e., drop volumes) representing the DSD. To maintain dimensional consistency in the former two features across all cases sharing the same mixing system, a post-sequence truncation is implemented up to a final time-step τ_f (see table illustrations in Figure 4), which corresponds to the length of the shortest sequence within a

given mixer's pool of cases. However, the length of the DSD feature remains dependent on the number of drops at each time-step t (ND_t), which is inherently linked to the governing physics of each case. This inconsistency effectively renders a three-feature input sequence with a case-specific, uneven feature size of $(1, 1, ND_t)$ per time-step t.

Handling input sequences with varying feature lengths represents a challenge for the model architecture implemented here, as standard LSTM networks are designed to have a fixed topology a priori (i.e., invariant parameter size),[44] thus requiring to be fed with fixed-sized input sequences of equal feature lengths.[45,46] This fixed-size requirement aims to prevent potential issues such as information loss and limitations on the model's capability to learn meaningful long-term dependencies. Previous works have implemented techniques to circumvent this flaw, such as padding, truncation, or 'attention' mechanisms. The former two are common approaches implemented in text recognition[47] and image processing,[46] where the length of the longest (padding) or shortest (truncation) sequence is set as the standard, and each sequence is either filled with zeros or has data removed accordingly.[46] Despite the benefits of longer padded sequences (Khotijah et al. demonstrated consistently higher model accuracy when handling longer sequences) or the easiness of dealing with truncating datasets, other studies have suggested alternatives (e.g., nearest neighbour interpolation) arguing that padding can be computationally demanding and naive truncation methods can lead to critical information loss.[48] More sophisticated methods such as *attention layers*[49] have shown remarkable potential in adequately filtering relevant input subsets. Still, they have been seen to fail when handling long-time-step predictions.[50]

Based on the above, we explored an application-specific approach to address the fluctuating DSD feature size without compromising the integrity of the datasets or substantially increasing the model's complexity and resource requirements. Our proposed method, depicted in Figure 4, consists of transforming drop volume data into discrete drop counts for different size ranges. These counts can be then partitioned into a fixed number of M bins ($[B_0, B_{M-1}]$), serving as individual features which monitor the number of drops entering or exiting a given size range

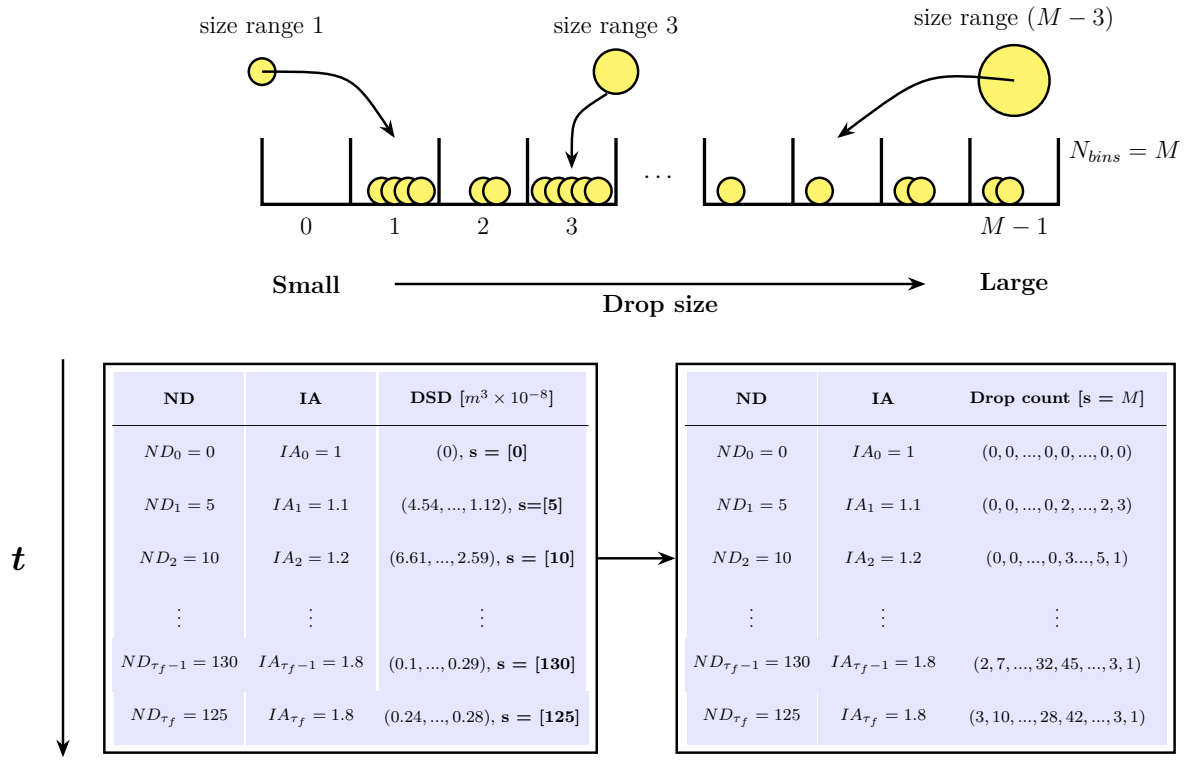


Figure 4: Schematic detailing DSD binning transformation. Top diagram showcases the drop size sorting and counting process in each size range, while the bottom tables show the transition from variable-sized volume lists (s = ND_t) to discrete drop counts in a fixed number of bins (s = M).

over time. These features are subsequently scaled between 0 and 1 through a normalised probability density estimation procedure, using the count of each bin as a density estimate as showcased in Equation 11 and Equation 12 for a bin $B_l \in [B_0, B_{M-1}]$ at a time t,

$$\hat{p}_{l,t} = \hat{p}(C_{l,t}) = \frac{C_{l,t}}{ND_t \times w_l}, \tag{11}$$

$$\hat{n}_{l,t} = \hat{n}(C_{l,t}) = \frac{\hat{p}_l(C_{l,t})}{\sum_{j=0}^{M-1} \hat{p}_l(C_{j,t})}, \tag{12}$$

where $\hat{p}_{l,t}$ and $\hat{n}_{l,t}$ denote the probability and normalised density estimators, respectively, for a drop count value $C_{l,t}$ corresponding to B_l at time t; ND_t stands for total drop count at time t, and w_l denotes the width of B_l. Our proposed approach introduces M additional features to the LSTM model, thus increasing the computational resources needed for training. However,

the resolution of the DSD (M) can be adjusted depending on the system studied, thus acting as a refinement parameter that balances accuracy and computational cost. In this work, M was initially set to 20 and 12 for the stirred and static mixing cases, respectively, aiming to provide sufficient resolution for the DSD data based on statistical analyses conducted in previous publications.[31,33] To eliminate outliers and uninteresting features (i.e., bins that mostly remain as 0 throughout the time domain), M was ultimately cut to 10 bins for both mixers ($B_0 - B_9$), dropping the boundary bins at each end of the size range proportionally, yielding 12 features overall for the LSTM to handle.

Finally, the ND and IA features were scaled and smoothed to mitigate potential biases arising from their substantially different scales ($\sim O(10^2)$ vs. $\sim O(1)$), and the sharply fluctuating trend noticed for the ND data. The scaling was executed through a linear sklearn MinMaxScaler method, applied to a range of [0,1]. Subsequently, three smoothing techniques were tested, namely moving average, Savitzky-Golay filter and Locally Weighted Scatterplot Smoothing, or 'Lowess'. After preliminary tests, the Lowess method, with a fraction of $\delta = 0.06$, and the Savitzky-Golay filter, with a window size of 5 and a third-order poly fit, yielded the best behaviour for the stirred and static mixer datasets, respectively. The former method conducts a weighted linear regression at each point using a cubic weight function $W(x) = (1 - |x|)^3$, based on the nearest $N \times \delta$ data points. Meanwhile, the latter performs a k-order polynomial fitting within a specified window length based on the least squares principle.[51] The normalised density estimations from the binned DSD data were not subjected to smoothing due to their extremely noisy behaviour (see Figure 12 and Figure 13 (e),(f)). The methods probed herein proved insufficient when attempting to retain the most relevant temporal trends, particularly for the boundary bins (i.e., largest or smallest sizes).

Data re-conditioning: Augmentation through windowing

Data augmentation is the process of generating synthetic data that incorporates existing knowledge about invariant properties of the original data against specific transformations. This

procedure lowers the risk of model over-fitting and enhances its accuracy and generalisation capabilities by providing a diverse yet realistic dataset.[52] Data augmentation has been actively probed for classification models in the field of computer vision (e.g., flipping, rotating, scaling images, adding Gaussian noise/distortion), but has been less explored in time-series scenarios given their vulnerability to transformation procedures.[52,53] Methods used for image data augmentation are not well generalised with time-series data, since some of their intrinsic properties (i.e., temporal dependency) are not fully leveraged by such methods and there is no assurance that the meaning of the raw data will remain unchanged.[53,54]

While it is possible to perform a frequency domain transformation to the time-series data to facilitate the application of some of these methods, additional complications emerge when dealing with complex or inherently intertwined multivariate time series data,[54] as is the case for the features studied herein. In such scenarios, it is advised to develop a tailored augmentation methodology that conserves the original data semantics.[53] Considering the above, and given the limited number of simulation runs available for training, we devised a basic time domain sequence-to-sequence (S2S) rolling window augmentation technique, which aims to expand the training data available while preserving their original semantics. Let us first examine the characteristics of the data and the training procedure without augmentation.

Let $\mathbf{X}$ be the original time-series array of a case under consideration with dimensions $D = (\tau_f + 1, M + 2)$, where $\tau_f + 1$ and $M + 2$ correspond to the total number of time-steps and features. A schematic of the time-series array is shown in Equation 13,

$$\mathbf{X} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{\tau_f} \end{bmatrix} = \begin{bmatrix} ND_0 & IA_0 & \hat{n}_{0,0} & \cdots & \hat{n}_{M-1,0} \\ ND_1 & IA_1 & \hat{n}_{0,1} & \cdots & \hat{n}_{M-1,1} \\ \vdots & \vdots & \vdots & & \vdots \\ ND_{\tau_f} & IA_{\tau_f} & \hat{n}_{0,\tau_f} & \cdots & \hat{n}_{M-1,\tau_f} \end{bmatrix}, \tag{13}$$

where $x_t \in \mathbb{R}^{M+2}$. The LSTM model training requires the original time series, $\mathbf{X}$, to be divided into input ($\mathbf{x}$) and target ($\mathbf{y}$) arrays, the latter acting as the ground truth for the

model to evaluate its predictions. Accordingly, we define n_i as the number of input steps aimed to be fed into the LSTM model and n_k as the number of target steps to predict into the future. Assuming $\tau_f = n_i + n_k - 1$, we can then define subsets $\mathbf{x}^{(n_i)},\ \mathbf{y}^{(n_k)} \in \mathbf{X}$, denoting the input and target sequences from the original time-series, with dimensions $D_{\mathbf{x}} = (n_i, M+2)$ and $D_{\mathbf{y}} = (n_k, M+2)$, respectively, as given by Equation 14.

$$\mathbf{x}^{(n_i)} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n_i-1} \end{bmatrix},\ \mathbf{y}^{(n_k)} = \begin{bmatrix} x_{n_i} \\ x_{n_i+1} \\ \vdots \\ x_{n_i+n_k-1} \end{bmatrix}. \tag{14}$$

This method would render an insufficient 10 or fewer training datasets for either mixing system (1 set per case). Therefore, the case-wise windowing approach implemented here seeks to augment the number of input-target sequence pairs $\{\mathbf{x}^{(n_i)}, \mathbf{y}^{(n_k)}\}$ from a single data-set, as explained below and shown schematically in Figure 5:

1. **Window creation:** The aim is to split the original data-set, $\mathbf{X}$, into a set of n_w+1 input and target sequences, denoted as $\left\{\mathbf{X}_j^w, \mathbf{y}_j^w\right\}_{j=0}^{n_w}$, with arbitrarily reduced dimensions $D_{\mathbf{X}^w}$ and $D_{\mathbf{y}^w}$, following the same definitions introduced earlier for n_i and n_k, but now considering $n_i + n_k << \tau_f$. The input/target pair is referred to as 'window', and its size along the temporal axis is defined as $s_w = n_i + n_k$. The procedure starts by building Window 0 from t_0 to t_{s_w-1}, as seen in Figure 5. This can be expressed as:

$$\mathbf{X}_0^w = \mathbf{X}[0 : n_i - 1, :], \quad \mathbf{y}_0^w = \mathbf{X}[n_i : s_w - 1, :].$$

2. **Window rolling:** Window 0 is then rolled forward to generate Window 1:

$$\mathbf{X}_1^w = \mathbf{X}[1 : n_i, :], \quad \mathbf{y}_1^w = \mathbf{X}[n_i + 1 : s_w, :],$$

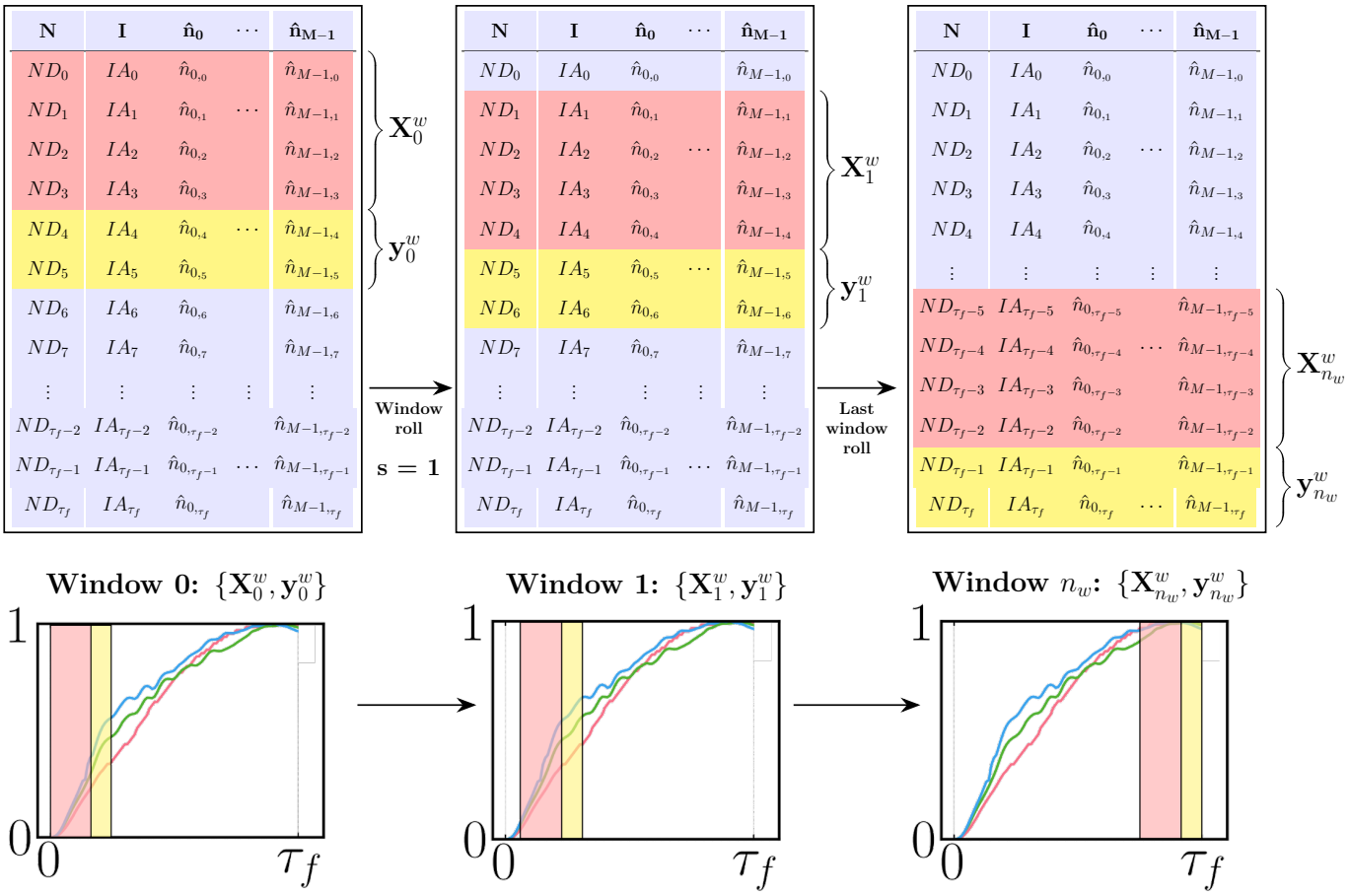


Figure 5: Schematic representation of the time-domain S2S windowing procedure implemented in this study. Top part provides a tabular representation of the windowed datasets, while bottom figures illustrate the rolling window procedure on top of a generic feature. Here, the window size is set to $s_w = 6$, where the number of input and target steps is $n_i = 4$ and $n_k = 2$, respectively. The window is shown to be rolled with a stride of $s = 1$.

where the rolling obeys a user-defined stride s, which was set to $s = 1$ in this work (see Figure 5). In general, the window for the next iteration $j + 1$ with $s = 1$ is:

$$\mathbf{X}_{j+1}^w = \mathbf{X}[j+1 : j+n_i, :], \quad \mathbf{y}_{j+1}^w = \mathbf{X}[j+n_i+1 : j+s_w, :].$$

3. **Data stacking:** The rolling procedure in step 2 is repeated n_w times, where n_w is determined by the expression $n_w = \tau_f + 1 - s_w$, taking $s = 1$. After each roll, the generated input/target sequences $\{\mathbf{X}_j^w, \mathbf{y}_j^w\}$ in each window are stacked separately into

new tensors $\mathbf{W_X}$, $\mathbf{W_y}$, respectively, as shown in Equation 15:

$$\mathbf{W_X} = \{\mathbf{X}_0^w, \cdots, \mathbf{X}_j^w, \cdots, \mathbf{X}_{n_w}^w\} = \begin{bmatrix} x_0 & x_1 & \cdots & x_{n_i-1} \\ x_1 & x_2 & \cdots & x_{n_i} \\ \vdots & \vdots & \cdots & \vdots \\ x_{\tau_f+1-s_w} & x_{\tau_f+2-s_w} & \cdots & x_{\tau_f-n_k} \end{bmatrix},$$
$$\mathbf{W_y} = \{\mathbf{y}_0^w, \cdots, \mathbf{y}_j^w, \cdots, \mathbf{y}_{n_w}^w\} = \begin{bmatrix} x_{n_i} & x_{n_i+1} & \cdots & x_{n_i+n_k-1} \\ x_{n_i+1} & x_{n_i+2} & \cdots & x_{n_i+n_k} \\ \vdots & \vdots & \cdots & \vdots \\ x_{\tau_f-n_k+1} & x_{\tau_f-n_k+2} & \cdots & x_{\tau_f} \end{bmatrix}, \quad (15)$$

where $x_t = (ND_t,\ IA_t,\ \{\hat{n}_{l,t}\}_{l=0}^{M-1}) \in \mathbb{R}^{M+2}$, $\mathbf{W_X} \in \mathbb{R}^{n_w \times n_i \times (M+2)}$ and $\mathbf{W_y} \in \mathbb{R}^{n_w \times n_k \times (M+2)}$. In this work, n_i and n_k constitute user-defined parameters in the LSTM architecture. These parameters serve a dual function: instructing the network on the number of input and output steps it will handle and constraining the number of windows created. On the other hand, τ_f is an intrinsic characteristic of the original time-series dataset. It is worth mentioning that $n_i,\ n_k$, and thus n_w, may affect the outcome of the LSTM training and rollout procedure, as they determine how many times the model iterates over its output, and consequently how much error propagation the results are exposed to (as we will discuss further on). However, this aspect of the model was not treated as a tunable hyperparameter as it would entail re-processing the entire raw dataset for every model training variation tested. Consequently, this would impose a substantial increase in computational expenditure.

Model training, deployment and explainability

LSTM model architectures & Training procedure

As mentioned at the onset of this section, we aim to construct a multivariate LSTM network to perform multi-step time-series predictions. The LSTM layers implemented herein were

built using Python package PyTorch, since its framework inherently supports multi-step input/target sequences with a constant number of features. The first architecture tested in this study is a simple neural net composed of a single LSTM layer and one fully-connected layer (LSTM-FC), where the latter reads the hidden state, $\mathbf{h}_t$, from the former to yield a predicted sequence. As seen in Figure 6, the LSTM processes a complete input sequence of n_i elements sequentially. Each element, $x_t \in \mathbb{R}^{M+2}$, contains all $M+2$ features per time-step, as previously introduced. Subsequently, the hidden state from the last time-step, $\mathbf{h}_{n_i-1}$, is fed into the FC layer to generate the predicted output sequence. This output sequence comprises n_k elements, each holding the same dimensions as those assigned to the elements within the input sequence. Connecting with the prior discussion, Figure 6 showcases the LSTM-FC network handling Window 0, where it reads the first element $\mathbf{W}_{\mathbf{X}}^{\mathbf{0}} = \{\mathbf{X}_0^w\}$ as an input sequence and predicts $\widehat{\mathbf{y}}_0^w$, corresponding to the target sequence pair $\mathbf{W}_{\mathbf{y}}^{\mathbf{0}} = \{\mathbf{y}_0^w\}$.

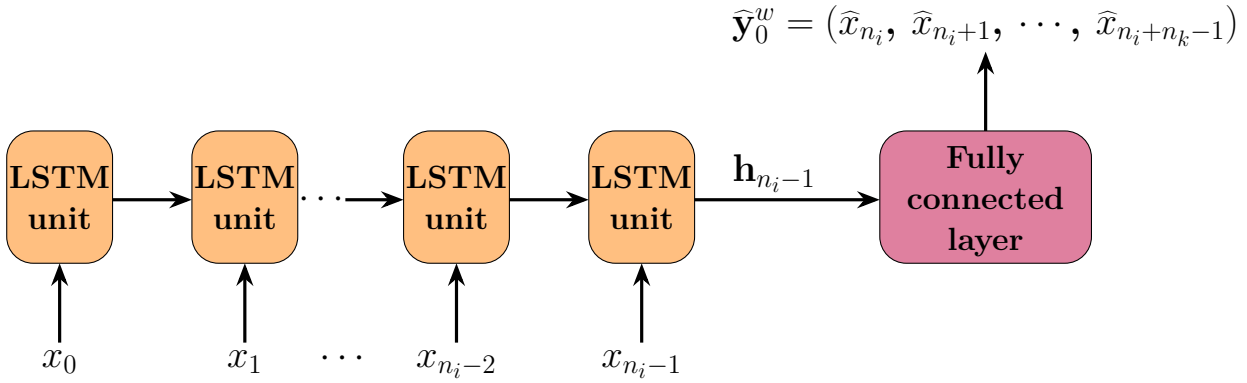


Figure 6: Diagram of a LSTM-FC neural network architecture.

Despite the simplicity of the previous neural net, more specialised architectures have been specifically designed to tackle multi-step forecasting. A common example is the LSTM Encoder-Decoder architecture, which comprises two LSTM sub-networks. The input network, referred to as the *Encoder*, derives a compressed representation of the input sequence, referred to as the encoded state. The second sub-network, referred to as the *Decoder*, interprets the encoded representation and exploits it to generate the predicted target sequence. Unlike the previous model, predictions in this approach are computed sequentially from the hidden

states of the LSTM units themselves, instead of coming from a single re-shaping layer at the end of the network. As depicted in Figure 7, the last element of the input sequence, x_{n_i-1}, and the encoded state of the last LSTM unit in the Encoder layer (corresponding to its hidden and cell state, $\mathbf{c}_{n_i-1}$, $\mathbf{h}_{n_i-1}$, respectively) are sent into the Decoder layer, which is the one responsible for generating a time-step prediction per LSTM unit.

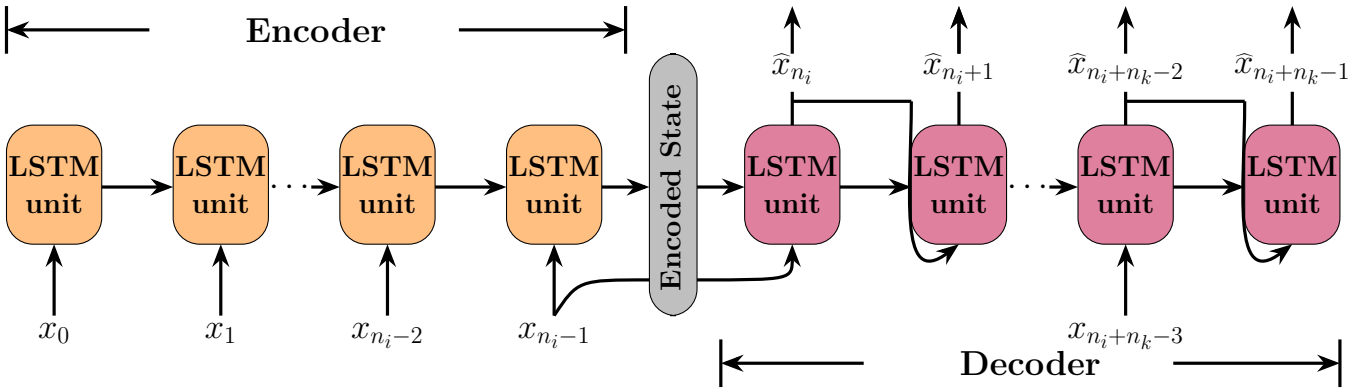


Figure 7: Diagram of a LSTM Encoder-decoder architecture, implementing mixed teacher forcing training.

A noteworthy advantage of this more intricate architecture over the LSTM-FC is its flexibility when it comes to training methodologies. Firstly, the predicted output of each LSTM unit in the Decoder layer can be recursively fed back into the following unit, until an output of the desired length is generated; this process is generally referred to as *recursive* prediction. Alternatively, true/target data can be exclusively fed into the LSTM Decoder units to make computations, similar to the Encoder layer, which is known as prediction via *teacher forcing*. Finally, both predicted outputs and true data can be alternately fed throughout the Decoder layer; in such a scenario, the model is said to be generated via *mixed teacher forcing*. For the latter method, a *teacher forcing ratio* parameter or *t.f.* ratio can be defined, which determines how much true data will be fed to the LSTM units in the Decoder layer vs. how much the model will re-use its predicted outputs to forecast the next step. Moreover, a boolean *dynamic teacher forcing* (*d.t.f.*) parameter can be introduced. When this parameter is set as 'True', the teacher forcing ratio is gently reduced at each epoch (i.e., one complete pass of the training dataset during model training). In this way, the model is

trained to learn patterns from the target data at the early times but it gradually acquires knowledge from its output, relying more on it to generate future predictions.

In this study, a shuffled batch-wise training methodology was adopted for both LSTM architectures. This procedure consists of dividing the windowed tensors, $\mathbf{W}_{\mathbf{X},\mathbf{y}}$, into smaller subsets or 'batches', thereby limiting the number of samples shown to the network during training before each weight update, and thus enhancing computational efficiency. In this approach, a set of input/target sequence pairs or windows, packed together into a batch $\mathbf{W}_{batch} = \{\mathbf{X}_j^w, \mathbf{y}_j^w\}_j^{\text{batch size}}$, are indexed and randomly shuffled before being fed into the network. Batching and randomising sequences enhance the model's ability to generalise effectively across different datasets, improves its capability to discern common patterns, and prevents it from learning biases associated with the order of the data.

In addition, a custom loss function was introduced during the training procedure, consisting of a standard Mean Square Error (MSE) loss function with an added penalty term that adopts the expression $w_p * \frac{1}{N}\sum_{i=1}^{N}(\text{ReLU}(-\hat{\mathbf{y}}_j^w))$, where w_p stands for a user-defined penalty weight, $\text{ReLU}(x)$ denotes the Rectified Linear Unit activation function, defined as $\max(0, x)$, and $\hat{\mathbf{y}}_j^w$ denotes a predicted sequence. This penalty term is meant to avoid negative predictions, which, in the context of this work, would yield non-physical results (i.e., negative drop count). This is done by isolating and averaging all initially estimated negative values by the network and then adding a fraction of this average as penalty. In this way, loss decreases when negative predictions are minimised. Furthermore, two regularisation terms $L1$ and $L2$, also known as Lasso and Ridge regressions, were added to the loss function to help manage overfitting. $L1$ introduces a penalty term based on the absolute value of the coefficient magnitudes, expressed as $l_1 \sum |\beta|$, while $L2$ adds a penalty based on the square magnitude of the coefficients, given by $l_2 \sum \beta^2$. Both penalty terms are regulated by coefficients $l_1,\ l_2$, respectively.

In the final stage of model training, we introduced two key strategies: an early-stopping procedure and a '*ReduceLROnPlateau*' scheduler. The early-stopping mechanism continuously monitors improvements in the validation loss score and saves the best-performing model before

any degradation occurs, effectively preventing overfitting. The scheduler optimises model convergence by dynamically reducing the learning rate once the model performance stagnates (i.e., validation loss stops improving). It is worth noting that the scheduler implemented here uses the well-known Adam (Adaptive Moment Estimation) optimiser, which is also employed for the backpropagation process throughout training.

Hyperparameter tuning

Before initiating the training procedure to estimate the network's learnable parameters (i.e., *weights* and *biases*), we carried out a comprehensive parametric sweep to optimise the framework's user-defined parameters, also known as 'hyperparameters'. As highlighted earlier, the input and target sequence sizes ($n_i,\ n_k$) were not included in this tuning step due to computational constraints. The remaining hyperparameters tuned in this study and shared by both architectures include the hidden size (hidden state dimension), learning rate (step size for adjusting model weights during training), and batch size (number of windows fed during training before weight update), as well as loss-related weight parameters such as the custom penalty weight w_p, and the l_1, l_2 coefficients controlling the corresponding regression penalty terms described prior. On top of the above, the LSTM Encoder-Decoder model considers an additional hyperparameter related to the training method adopted, namely the *t.f.* ratio. The *d.t.f.* feature was fixed as 'True' for all Encoder-Decoder tuning cases tested to reduce the size of the sample space and prioritise other hyperparameters.

We performed four separate exhaustive searches of over 2000 parameter combinations, one for each model and mixing system. These parametric explorations were carried out through PyTorch package Ray Tune, in search of each specific case's optimal configuration. The hyperparameter sample space explored is detailed in Table 2, along with each case's best-performing model configuration. The parameter candidate values were selected based on early sensitivity trials and their relevance in the context of this study. For instance, a finer batch size sample space was explored given the relatively modest dataset available for

Table 2: Hyperparameter search space and best-performing model configuration for each corresponding architecture and mixing system studied.
* *batch size range lies between 8 and 40, with a step of 4, rendering 9 parameter values.*

Hyperparameters	**Value ranges**	**Best performing model**			
		LSTM-FC		**Encoder-decoder**	
		Stirred	Static	Stirred	Static
Hidden size	64, 128, 256	256	64	256	64
Learning rate	0.002, 0.005, 0.01	0.002	0.01	0.002	0.005
Batch size	(8∼40, 4)*	36	36	40	24
Penalty weight (w_p)	0.01, 0.1, 1, 10	0.1	0.1	0.1	0.1
Prediction type	Recursive, t.f., mixed	—		Mixed	Mixed
t.f. ratio	0.02, 0.1, 0.2, 0.4	N/A		0.1	0.02
l_1 coefficient (Lasso)	0, 1×10^{-5}, 1×10^{-4}	0	0	0	0
l_2 coefficient (Ridge)	0, 1×10^{-5}, 1×10^{-4}	0	1×10^{-5}	0	0

training and validation, which causes noise effects introduced from varying batch sizes to be more impactful in the generalisation performance of the model. This is reflected in the widely different sizes obtained for each scenario. Other settings yielded expected results, such as larger hidden sizes for the stirred mixer given its substantially larger datasets, or a constant penalty weight, w_p, given the similar nature of all features and datasets. Similarly, the $L1$, $L2$ penalty terms were essentially deemed unnecessary, as overfitting has already been mitigated in various other ways, as previously described (e.g., batch randomisation). On the other hand, the Encoder-Decoder architecture always preferred a mixed teacher forcing training method with small *t.f.* ratios, implying that the model mostly does not rely on the ground truth to carry out predictions throughout the Decoder layer, but still prefers to have some access early on to improve its performance, rather than running exclusively on its output. Future work is suggested to explore the inclusion of the *d.t.f* feature in the hyperparameter tuning exercise, as it might show the true dependency of the model on the ground truth.

Prediction via sequence generation (Rollout)

As explained earlier with the introduction of the windowing process and the model architectures, the LSTM network is designed to map an input sequence of length n_i to an output sequence of fixed length, n_k, where $n_k + n_i << \tau_f$. Recall that our objective is to predict the entire temporal evolution of the dispersion features up to the final time-step, τ_f, where simulations are terminated. To achieve this, a rollout procedure is implemented, wherein the trained model is iteratively reused until the desired time-step is reached. During each iteration, the previous output sequence is fed back into the trained model to predict the next n_k sequence. The number of iterations or 'rollouts' is therefore determined by the length of the output sequence, following the expression $r = (\tau_f - n_i)/n_k$.

The selection of n_i and n_k is not a trivial task since these two values determine the size and number of data samples available for the model to be trained with (refer to subsection *Data re-conditioning*), which naturally has a direct effect on model performance and uncertainty. Considering the differing lengths of the time-series datasets for each simulation case, acknowledging that they have been truncated for each mixing system ($\tau_f = 385$ and $\tau_f = 98$ for the stirred and static mixer, respectively), the values for $n_i,\ n_k$ were fixed to (50, 50) and (40, 30) for the stirred and static mixer, respectively. These values were subjected to an early sensitivity test, but a full-scale tuning process would be required to discover the optimal configuration for each mixing case study. Accordingly, the number of rollouts, r, is computed as 7 and 2 for the stirred and static mixer, respectively.

Uncertainty quantification from an ensemble of perturbed inputs

After adequately training the model, the next logical step is to estimate its prediction uncertainty. Regarding this, many researchers have contributed to understanding and quantifying the uncertainty in a neural network's prediction (see the comprehensive review by Gawlikowski et al.). Yet, an approach to uncertainty quantification applicable in our current work is required to cope with the fact that the trained model iterates over its output

numerous times, as described above through the rollout procedure. In other words, we require a method that is capable of tracking the evolving model uncertainty and granting us access to the model performance through the propagation. To achieve this, an ensemble-based procedure, adapted from the ensemble forecasting technique that has been extensively used in numerical weather prediction,[56–58] is proposed herein.

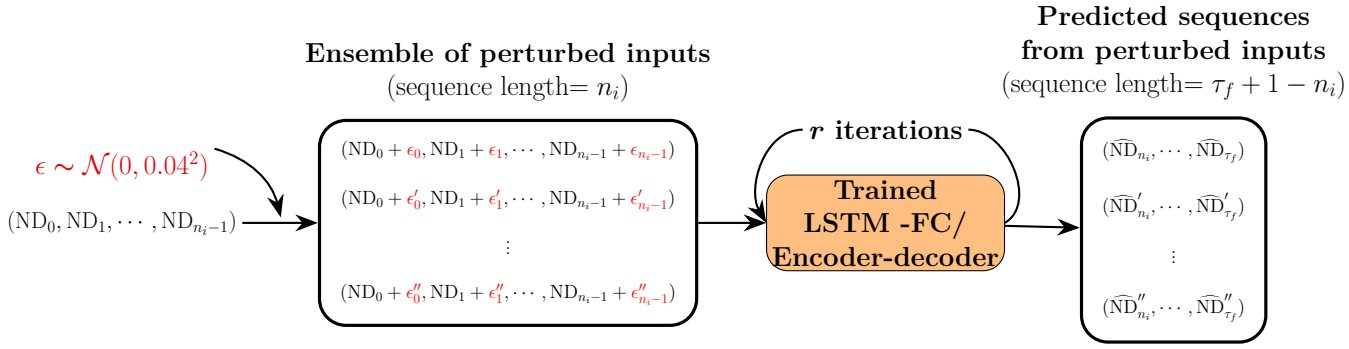


Figure 8: Diagram showcasing the procedure of ensemble-based uncertainty quantification. Take ND as an example, perturbations are introduced to the input sequence by adding noises, ϵ, drawn from a Gaussian distribution, $\mathcal{N}(0, 0.04^2)$, at each time-step, giving rise to the ensemble of perturbed inputs. All of the ensemble members are fed into the trained model which progressively produces the ensemble of predicted sequences (with a sequence length of $\tau_f + 1 - n_i$). r is the number of iterations determined by the target sequence length, $r = (\tau - n_i)/n_k$.

As showcased in Figure 8, we start by introducing perturbations parameterised by ϵ to the input sequence. This involves adding noise, drawn from a distribution, to the feature values at each time-step (Figure 8 takes ND as an example). Herein, a Gaussian distribution with mean $\mu = 0$ and standard deviation (referred to as *std. dev.* henceforth) $\sigma_{s.d.} = 0.04$ is used, namely $\epsilon \sim \mathcal{N}(0, 0.04^2)$, such that the feature values in the perturbed input sequence represent a probable uncertainty arising from observed feature values at early dispersion. For instance, for a given mixing system, the measured dispersed drop count at early times could be a few drops off if measurement errors are considered. It is worth noting here that the added noise could lead to negative feature values in the perturbed input sequence, which would be physically unrealistic; nonetheless, the purpose of their inclusion is to examine the capability of the trained model to handle any type of disturbance in the input data. In this

way, an arbitrary perturbed input ensemble containing 200 members is generated, which subsequently enters the trained model to produce an ensemble of corresponding predicted sequences.

With the sequences above, we first compute the *std. dev.* across the 200 ensemble members at each time-step using the empirical *std. dev.* expression per sample:

$$\sigma_{s.d.} = \sqrt{\frac{\sum(b_i - \bar{b})^2}{N_e - 1}},$$

where b_i, $\bar{b}$, and N_e denote the i^{th} member in the ensemble, the ensemble average, and the ensemble size ($N_e = 200$), respectively. Then, a prediction interval is calculated to give an overview of the range wherein the model prediction is likely to occur (with a 95% confidence) given the unperturbed input sequence, which could be written as:

$$[\mu - z\sigma_{s.d.}, \mu + z\sigma_{s.d.}], \quad z = 1.96.$$

Lastly, we made use of the prediction interval to suggest a fair indicator of the model performance, namely the absolute residual between the targeted perturbed evolution and its predicted counterpart from the trained model using the unperturbed input sequence. Corresponding results and relevant discussion are presented in the ensuing section.

Results and discussion

This section is subdivided as follows: firstly, a brief discussion on the model's generalisation based on its performance on both training and validation datasets. Secondly, we present an in-depth exploration of the framework's performance on the testing datasets. This involves a thorough analysis of the predicted temporal evolution for all features, namely ND, IA, and DSD, across both mixing systems and network architectures. In particular, the DSD predictions are interpreted through selected bins, i.e., B_3, B_5, B_6, and B_8, for both mixers,

containing the normalised density estimators, $\hat{n}_{l,t}$, introduced for the binning procedure in the Methodology section. The drop size is recovered from the density estimators as a log-scale normalised drop volume, $\log_{10}(V_d/V_{\text{cap}})$, where V_d is the volume of the dispersed drop and V_{cap} denotes the volume of a spherical drop whose diameter corresponds to the capillary length scale, $\lambda_c = \sqrt{\frac{\sigma_{cl}}{(\rho_a - \rho_o)g}}$. Accordingly, the drop sizes covered in this work lie in a range of $\log_{10}(V_d/V_{\text{cap}}) = [-4.5, 0.0]$ and $[-7.25, -0.5]$ for the stirred and static mixer, respectively. Following the binning procedure described earlier, the actual volumes corresponding to each of the bins treated in this section are listed in Table 3. Lastly, the ensemble-based approach implemented to quantify the model's uncertainty is showcased using features ND and IA from one testing case.

Table 3: Volumes of the dispersed entities corresponding to the size ranges presented in Figure 12 and Figure 13.

Bin	**Stirred mixer** $\lambda_c = 0.0045[m]$		**Static mixer** $\lambda_c = 0.0030[m]$	
	Bin edge $\log_{10}(V_d/V_{\text{cap}})$	**Actual size** $[m^3 \times 10^{-8}]$	**Bin edge** $\log_{10}(V_d/V_{\text{cap}})$	**Actual size** $[m^3 \times 10^{-8}]$
B_3	[-3.50, -3.00]	$[1.54 \times 10^{-3}$, $4.86 \times 10^{-3}]$	[-5.75, -5.00]	$[2.54 \times 10^{-6}, 1.43 \times 10^{-5}]$
B_5	[-2.50, -2.00]	[0.02, 0.05]	[-4.25, -3.50]	$[8.04 \times 10^{-5}, 4.52 \times 10^{-4}]$
B_6	[-2.00 -1.50]	[0.05, 0.15]	[-3.50, -2.75]	$[4.52 \times 10^{-4}, 2.54 \times 10^{-3}]$
B_8	[-1.00, -0.50]	[0.49, 1.54]	[-2.00, -1.25]	[0.01, 0.08]

Model generalisation: performance on training and validation data

Prior to analysing model performance on the testing sets, we first compare the performance of both trained networks on the training and validation datasets, depicted in Figure 9 and Figure 10. As displayed, the predicted values relevant to the training dataset align reasonably well with the true data; in contrast, a deviation between predictions and the ground truth slightly develops when looking at the validation set. An evident deviation at low values (data point < 0.3) can be observed, not only for train and validation but also for testing

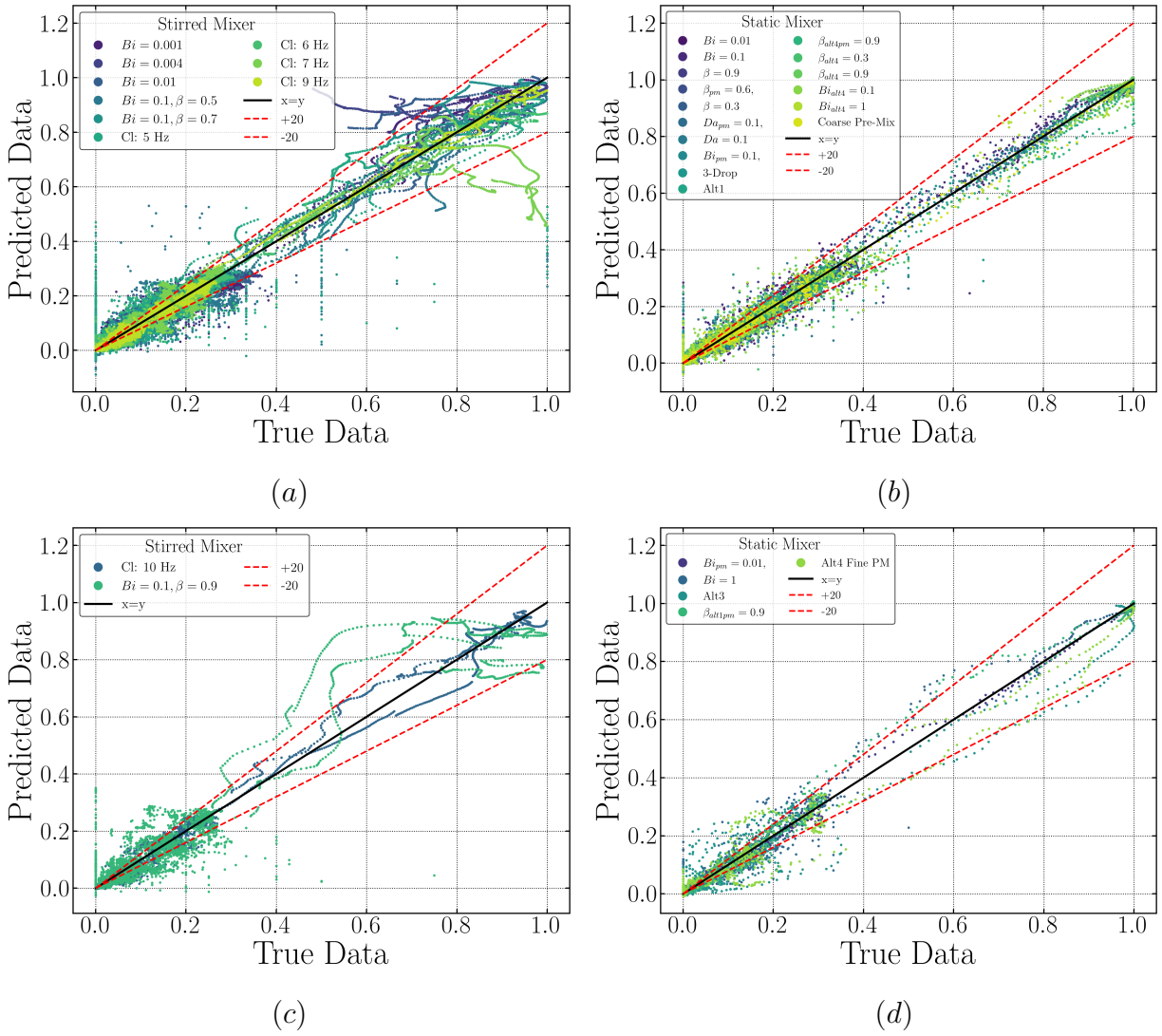


Figure 9: LSTM-FC predicted vs. true data error dispersion plots for all 12 features considered in this study. A ± 20 % deviation area is included. Sub-figures to the left ((*a*), (*c*)) showcase training and validation data for stirred mixers, while those to the right ((*b*), (*d*)) illustrate training and validation data for static mixers, respectively.

predictions (refer to Figure 11). Therefore, relevant discussion on this observation will be included up next to avoid redundancy. In addition, to give a clear overview, the root mean squared error (RMSE) and the coefficient of determination, R^2, are computed and listed in Table 4 to evaluate the training and validation performance. The consistently small values of RMSE for both the stirred mixer (0.0571 training, 0.0577 validation) and static mixer (0.0321 training, 0.0467 validation), coupled with their marginal increase relative to the training

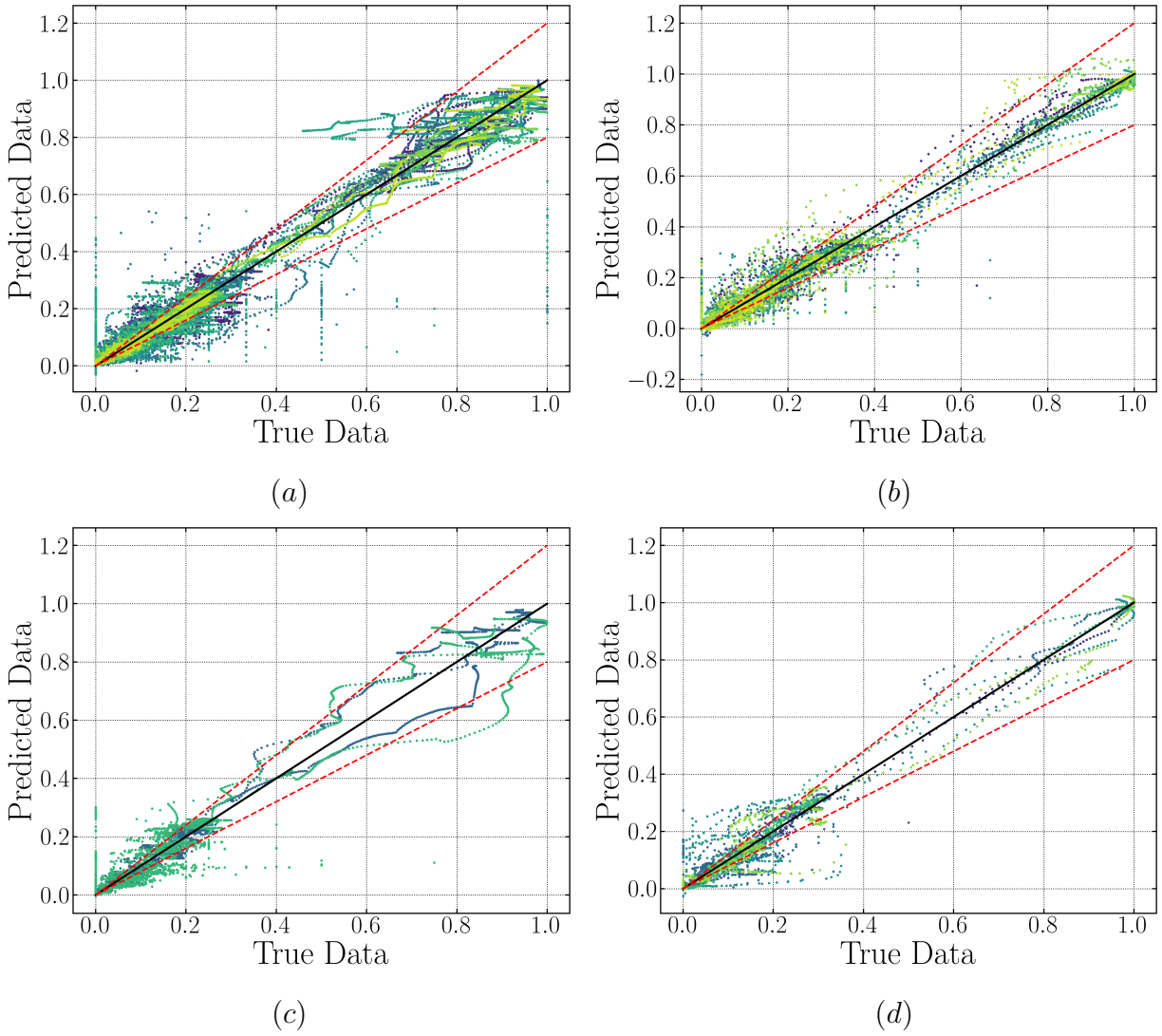


Figure 10: LSTM Encoder-decoder predicted vs. true data error dispersion plots for all 12 features considered in this study. A ± 20 % deviation area is included. Sub-figures to the left ((*a*), (*c*)) showcase training and validation data for stirred mixers, while those to the right ((*b*), (*d*)) illustrate training and validation data for static mixers, respectively. Plot legends are shared with Figure 9, and thus not included here to avoid redundancy.

dataset, suggest that the trained LSTM-FC model adeptly captures underlying patterns in the dispersion dynamics and can generalise this knowledge to new data, indicating robustness against overfitting and ensuring reliable predictions on unseen data. Similar performance can be seen from the RMSE values for the LSTM Encoder-decoder, which is also supported by the corresponding values of R^2. From the dispersion clouds displayed in Figure 9 and Figure 10, no major differences are apparent between the two architectures when examining the static

mixer. On the contrary, the LSTM Encoder-decoder seems to provide a better-trained state for the stirred vessel, as most of the sequences above 20% deviation are eliminated, especially for the clean and low $Bi < 0.1$ cases. This observation holds for the validation set, where only a small section is underpredicted beyond a 20% deviation for the encoder-decoder structure, unlike the LSTM-FC which heavily over-predicts the surfactant-laden case.

Table 4: Evaluation metrics for model performance on training and validation data. The values in the bracket refer to the corresponding metrics for datasets: (training, validation).

Metrics	**Stirred mixer**		**Static mixer**	
	LSTM-FC	**LSTM Encoder-decoder**	**LSTM-FC**	**LSTM Encoder-decoder**
RMSE	(0.0571,0.0577)	(0.0500,0.0502)	(0.0321,0.0467)	(0.0424,0.0491)
R^2	(0.9516,0.9513)	(0.9629,0.9630)	(0.9853,0.9678)	(0.9745,0.9644)

Model performance: prediction via rollout

This section dives into the model rollout predictions on the testing datasets for both mixing systems and LSTM architectures. Similar to what we have shown for the training and validation datasets, Figure 11 presents error dispersion plots for the testing cases, which contain all 12 target scaled features (ND, IA and 10 bins). This figure offers a broad overview of each model's performance for all testing cases. However, it falls short when pinpointing the specific features associated with a particular error cloud or region. This association can only be achieved by correlating it with the subsequent figures showcasing the rollout predictions for each feature (Figure 12 and Figure 13).

Starting with the LSTM-FC model in Figure 11$-(b)$, the dots corresponding to the surfactant-free case of Fine Pre-Mix (coloured light yellow) are mostly found above the black solid parity line ($y = x$). This implies that the model is inclined to overestimate the dispersion metrics for this particular setup, predicting a larger interfacial area, and more, larger dispersed drops overall. We believe the overestimation is due to the trained LSTM failing to learn the

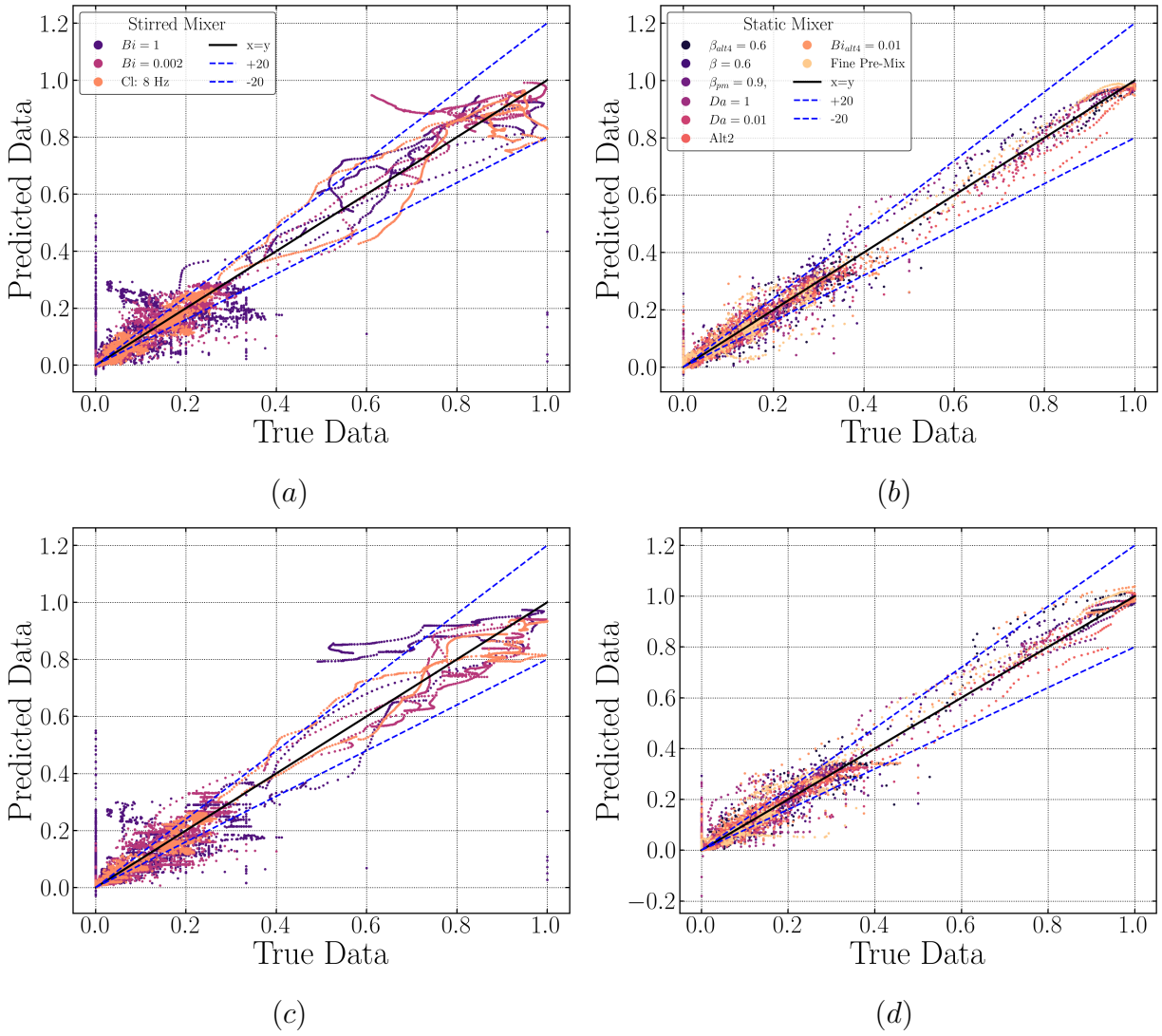


Figure 11: Predicted vs true data error dispersion plots for testing datasets, with a ± 20 % deviation area included. Sub-figures a) and c) showcase rollout prediction data dispersion for the stirred mixer via FC and Encoder-Decoder architecture, while subfigures b) and d) illustrate rollout prediction data for the static mixer via FC and Encoder-Decoder, respectively.

relationship between surfactant and the distribution of ND and IA, whereby the absence of surfactant in the Fine Pre-Mix case naturally leads to markedly smaller ND and IA values due to higher interfacial tension when compared to other surfactant cases.

Another example is the Alt 2 static mixer case in Figure 11−(b), where its predicted values (coloured red-orange) are entirely underestimated, signifying that the distinct dispersion performance patterns induced by an alteration in the mixer's geometry, rather than a change

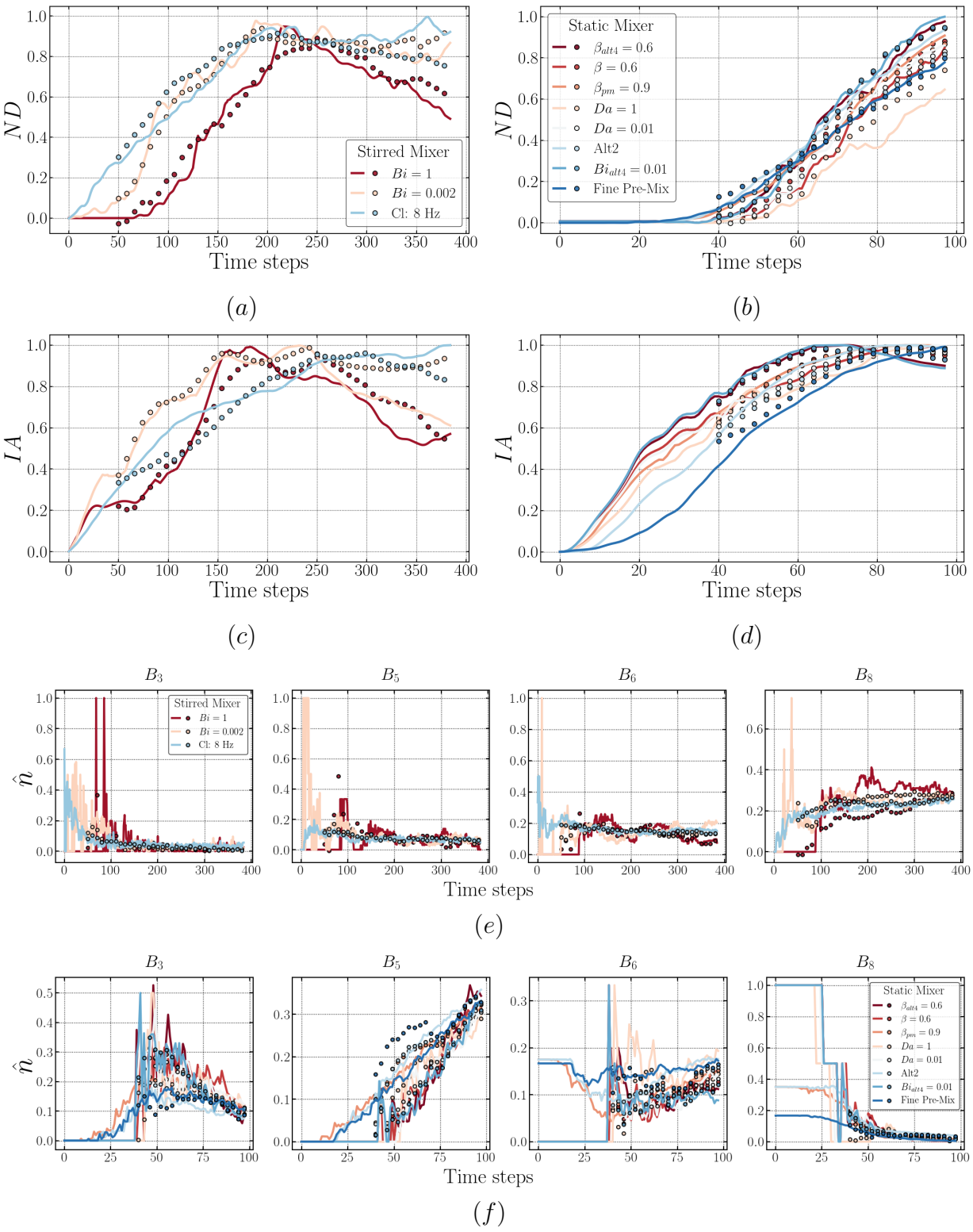


Figure 12: Plots comparing the model target sequences (lines) and predicted sequences via rollout procedure (dots) of LSTM-FC for both mixers. The results of predicted drop size distribution are exemplified using B_3, B_5, B_6 and B_8 for both mixers ($(e), (f)$).

in the chemical nature of the species (e.g., surfactant profile), is not learned properly. In contrast, the predicted values via LSTM-FC for the stirred mixer cases are distributed over both sides of the parity line $y = x$, as shown in Figure 11$-(a)$, implying that there isn't a defined bias in the accuracy of the model. Figure 12 offers a detailed perspective on the rollout predictions concerning each feature. In all cases, encompassing both mixers, noticeable variations beyond $\pm 20\%$ are observed at low true data values (data point < 0.3). These discrepancies predominantly align with early time-step predictions for any feature. This deviation is most likely due to the sensitivity of DSD to perturbations in the total drop count during the early stages of dispersion formation, especially at the edges of the distribution (i.e., small or large drops). More discussion relevant to this will be presented along with subsequent figures.

As mentioned previously, Figure 12 compares the target and predicted values for each feature via LSTM-FC. The initial observation drawn from this figure indicates that the predicted values for features ND and IA (at early times) exhibit a much better agreement with the targets compared to that seen for the selected bins exemplified herein. This corroborates our earlier assertion regarding the origin of the large deviations associated with low-value true data. Moreover, the trained LSTM-FC correctly captures the hierarchy of the stirred mixer's cases for all features (see Figure 12$-(a), (c), (e)$), particularly at the early time-steps of the predicted sequence. Taking ND as an example, the order, Cl: 8 Hz $>$ $Bi = 0.002 > Bi = 1$ is then reproduced for time-steps $\sim 50 - 200$. However, the trained LSTM-FC is unable to predict this ranking further down the time range, as displayed in Figure 12$-(a)$: the predicted ND for Cl: 8 Hz is lower than that for $Bi = 0.002$. This can be attributed to the model's inability to extract the correct physical knowledge from the Cl: 8 Hz case, failing to recognise that this particular case has a higher impeller speed, thus resulting in an extended duration of dispersed drop generation. A similar scenario can be seen for the static mixer (see Figure 12-$(b), (d), (f)$) wherein the trained LSTM-FC is capable of recovering the hierarchy at early times, but since the gap between the prior input information

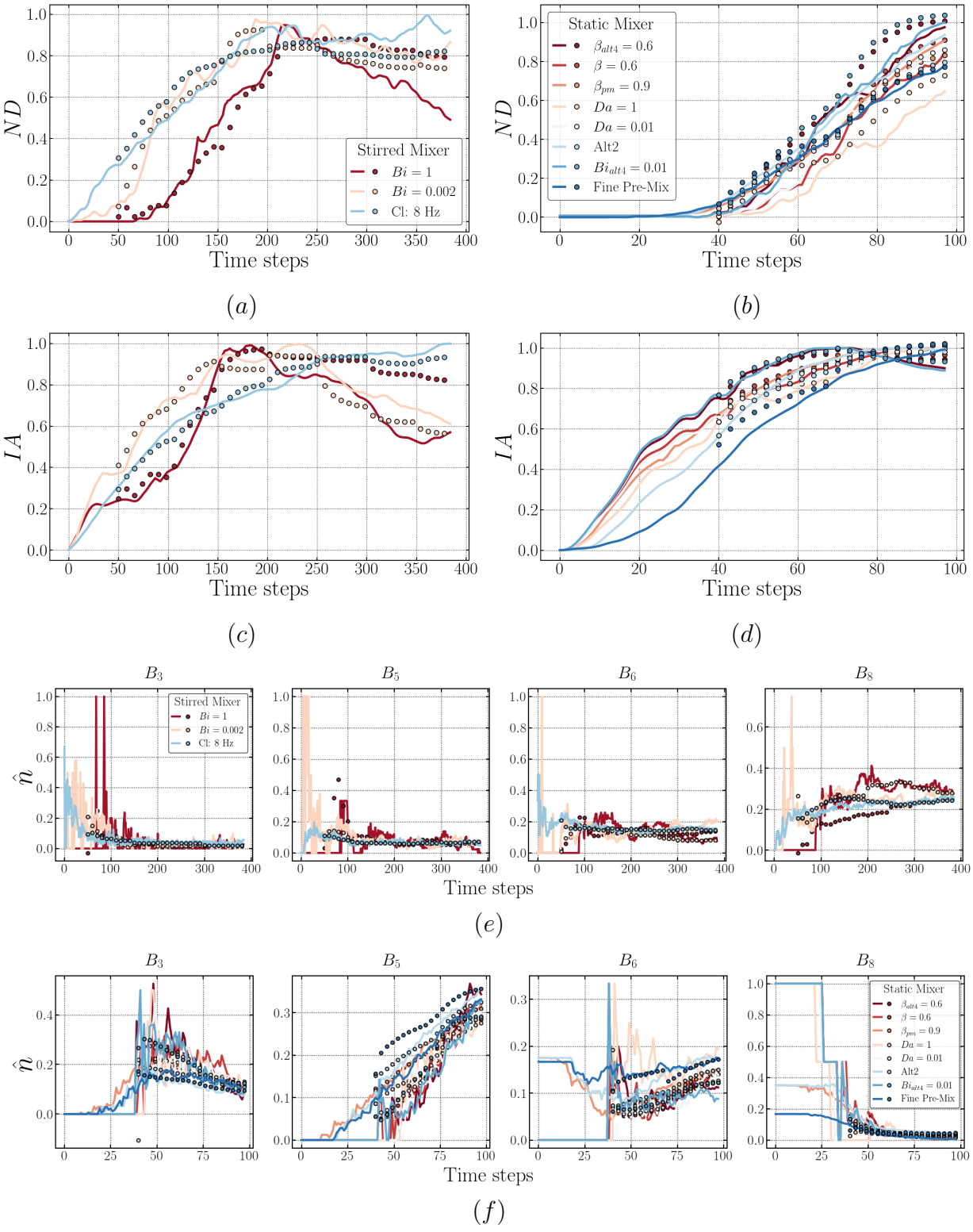


Figure 13: Plots comparing the model target sequences (lines) and predicted sequences via rollout procedure (dots) of LSTM-FC for both mixers.The results of predicted drop size distribution are exemplified using B_3, B_5, B_6 and B_8 for both mixers ($(e), (f)$).

and the point where it is needed becomes larger, the model starts to perform poorly to some degree.

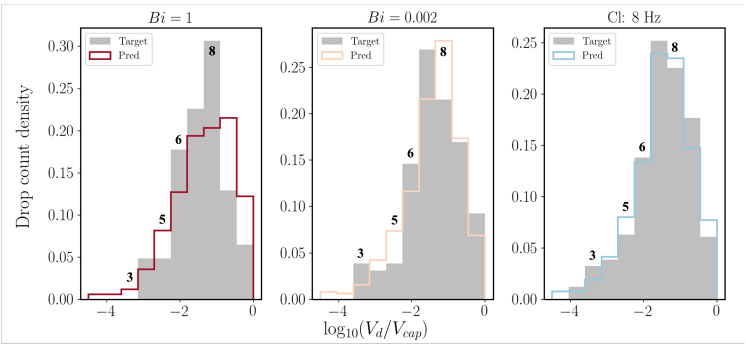
The overall performance of the LSTM Encoder-decoder in terms of the stirred mixer is comparable to that of the LSTM-FC (see Figure 11$-(c)$): except for the scattered cloud seen at the initial steps, where most of the predicted values are located within the 20% deviation area from the parity line $y = x$. However, obvious discrepancies arise from the Fine Pre-Mix and $Da = 1$ (coloured purple) cases when dealing with the static mixer, whereas the predicted values for other cases remain closer to the parity line $y = x$ (see Figure 11$-(d)$). As for the performance on each feature, Figure 13 proves that the capability of the LSTM Encoder-decoder is similar to that of the LSTM-FC, correctly capturing the case hierarchy at the early times. Furthermore, Figure 13$-(b), (d), (f)$ clearly show the deviation highlighted above for the static mixer (the case of Fine Pre-Mix) in terms of each feature. This provides evidence to support the fact that the deviation is mainly caused by the poor performance on the bins prediction. From these figures, we infer that the underestimated B_3 and the overestimated B_5 and B_6 correspond to the dots occur beyond $\pm 20\%$ in Figure 11.

From the discussion presented above, it is clear that the prediction accuracy is high at early times and deteriorates subsequently for both the LSTM-FC and LSTM Encoder-decoder; however, this occurs for different reasons related to the different architectures. As described in the previous section, LSTM-FC is trained to map an entire output sequence with its input, while LSTM Encoder-decoder learns to progressively generate the output sequence; this could enable the LSTM Encoder-decoder to have a superior performance when dealing with sequence generation (i.e., prediction via rollout herein) since the LSTM-FC has not been trained to utilise the previous prediction for the generation of the next sequence. Consequently, the LSTM Encoder-decoder is likely to perform particularly well given that the dispersion dynamics underlying the data have been learned. For instance, as presented in Figure 13$-(c)$, the trained LSTM Encoder-decoder accurately predicts the increase in the case of Cl: 8 Hz as well as the descending trend for $Bi = 0.002$ and $Bi = 1$.

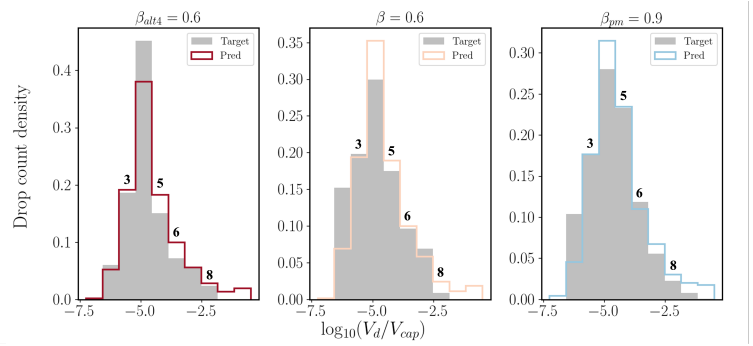
Nonetheless, additional observations from Figure 13 indicate that the trained LSTM Encoder-decoder performs relatively poorly on some features (or cases), e.g., the ND predictions exhibit deviations from their target values for various stirred mixer cases. Similarly, for static mixers (e.g., see B_5 in Figure 13$-(f)$), the case of Fine Pre-Mix deviates substantially from its target, while the trend for $Bi_{alt4} = 0.01$ is properly captured. These observations suggest that the trained LSTM Encoder-decoder must be improved to achieve a better understanding of the features and the mechanisms underlying the data presented in this work; this could be achieved by training on larger datasets and further tuning the hyperparameters. Although simply mapping between input and output sequences makes it comparatively easier to train an LSTM-FC, its performance would be globally inferior to that of a well-trained and finely-tuned LSTM Encoder-decoder; this is due to the feature evolution embedded in the sequence in the latter, which is absent in LSTM-FC.

Figure 14 presents examples of the predictions of the DSD in the form of histograms containing all the bins involved in the predictions. This figure, along with the rollout predictions relevant to bins presented above, forms an integrated visualisation of model performance concerning DSD prediction. For the case of stirred vessels, it can be seen from Figure 12$-(e)$ and Figure 13$-(e)$ that the two models perform reasonably well on the bins of Cl: 8 Hz, which agrees with what is shown in Figure 14$-(a), (c)$ that the predicted distribution of Cl: 8 Hz fairly matches its target. Likewise, from the latter figure, B_8 for the case of $Bi = 1$ is underestimated which is consistent with that displayed in the previous plots.

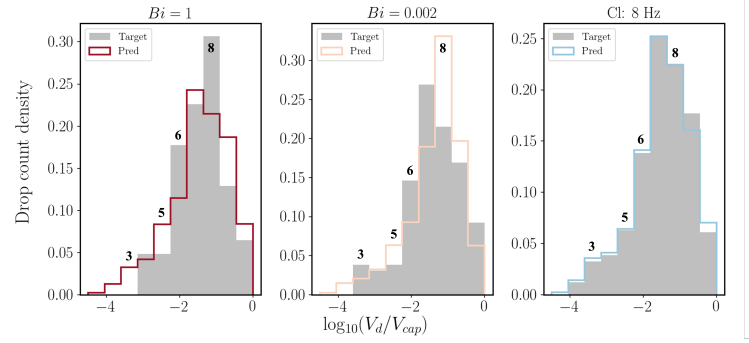
To give an overview of the difference between the targeted and predicted distributions, we computed the Wasserstein distance using the built-in functions from SciPy library in Python, which is a measurement originating from studying optimal transport problems.[59] Intuitively, if each distribution is treated as piles of soils, this metric quantifies the masses and corresponding distance that must be moved around to turn one distribution into the other; hence, this metric is also known as earth mover's distance (EMD). It is ubiquitous



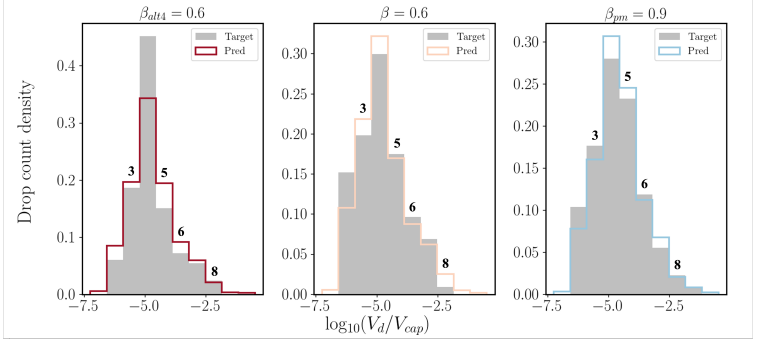
(a) LSTM-FC: Stirred mixer



(b) LSTM-FC: Static mixer



(c) LSTM Encoder-decoder: Stirred mixer

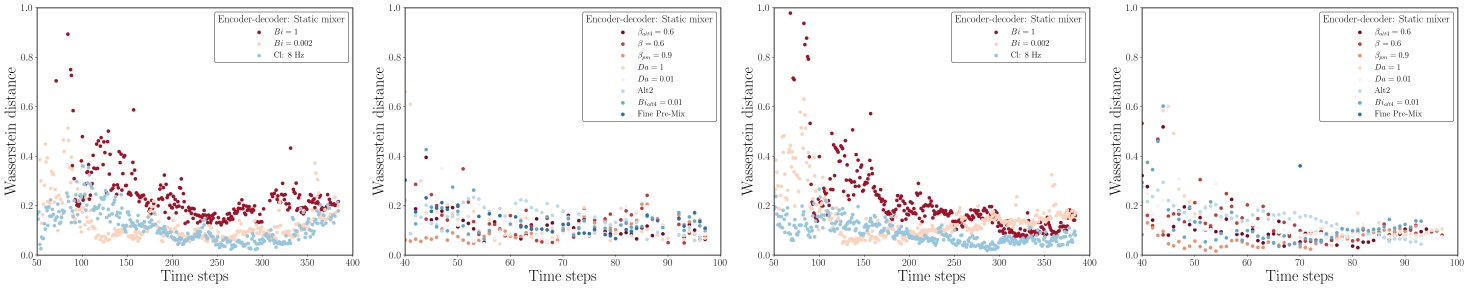


(d) LSTM Encoder-decoder: Static mixer

Figure 14: Exemplified histograms presenting the predicted drop size distribution via LSTM-FC ($(a), (b)$) and LSTM Encoder-decoder ($(c), (d)$) for both mixers. In the case of stirred mixer, three test cases are shown for time-step $t = 280$: $Bi = 1, Bi = 0.002$ and Cl: 8 Hz, whereas the results of static mixer are demonstrated using test cases, $\beta_{\text{alt4}} = 0.6, \beta = 0.6$ and $\beta_{\text{pm}} = 0.9$ at time-step $t = 74$. The numeric labels in the figure denote the corresponding bins (B_3, B_5, B_6 and B_8) shown in Figure 12 and Figure 13.

in the field of statistical analysis to address the dissimilarity quantification between two probability distributions (see a comprehensive review by Panaretos and Zemel).

In general, the EMD can be considered as the distance (or divergence) between two distributions; hence, lower values of this score indicate higher similarity. With this in mind, the information drawn from Figure 15 is that, initially, the predicted DSDs via both LSTM-FC and LSTM Encoder-decoder tend to significantly deviate from the target; the deviation then slowly diminishes with increasing time-step. This strongly supports our previous hypothesis that the small amount of dispersed drops produced during the early stages of dispersion formation gives rise to a "statistically unstable" distribution that is rather sensitive to small changes in the drop count of each bin. In addition, the statistical instability could be



(a) (b) (c) (d)

Figure 15: Temporal plots presenting the divergence between targeted and predicted drop size distribution via LSTM-FC ($(a), (b)$) and Encoder-decoder ($(c), (d)$) for both mixers, using metrics, Wasserstein distance.

inherently linked to the grid resolution of our simulations, which indicates that non-physical drops are appearing or disappearing from the domain, causing stochastic irregularity in the data. Recall that as shown in Figure 11, the poor performance related to bins at the early stages of dispersion formation is mirrored by the widely spread dots in the lower left corner of the figure. Nevertheless, with an increase in drop count, the trained models capture the DSD more faithfully with a low level of dissimilarity as reflected by the EMD scores (< 0.2) in Figure 15.

Uncertainty quantification

In this section, we present the uncertainty quantification of the two trained models, by demonstrating the procedure using two features ND and IA of the test case $Bi = 0.002$ for a stirred mixer and $\beta = 0.6$ for a static mixer.

The results shown in Figure 16 correspond to a comparison among the model target, model prediction from an unperturbed input sequence, and the prediction interval region from a perturbed input ensemble. The first observation from this figure is the different spread sizes of the prediction interval region for the two model architectures. As illustrated previously, perturbation noise is sampled from the distribution $\mathcal{N}(0, 0.04^2)$, which generates a spread of the input ensemble with corresponding *std. dev.*, $\sigma_{s.d.} \approx 0.04$. However, the spread of the

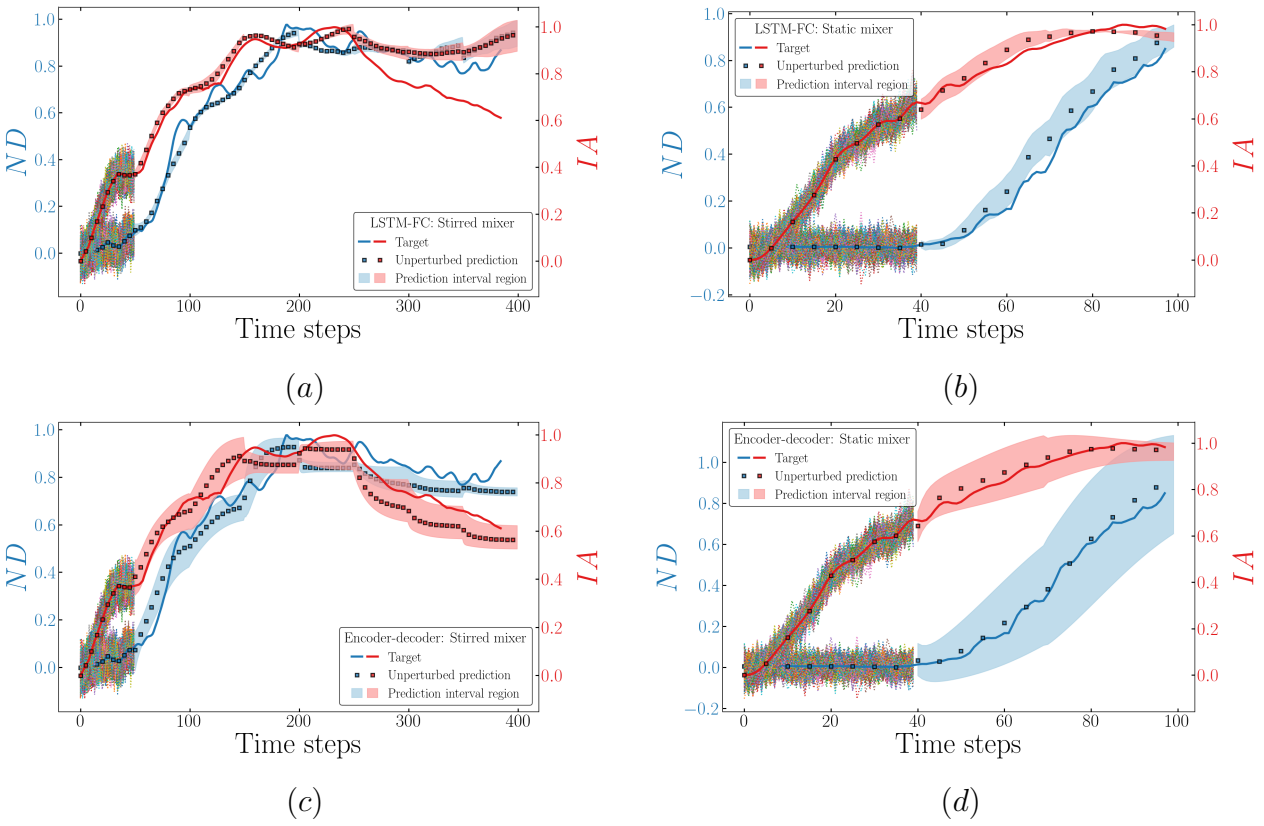


Figure 16: Uncertainty quantification plots, comparing the model target, model predictions from unperturbed input and the ensemble prediction interval region, for LSTM-FC ($(a), (b)$) and Encoder-decoder ($(c), (d)$). The dots cloud displayed at the early times represents the ensemble of perturbed input sequences.

prediction ensemble via LSTM-FC is narrower than that of the input ensemble. This suggests a low level of sensitivity of the LSTM-FC to input noise. Meanwhile, an apparent deviation of the target (solid line) from the spread is seen, particularly in the case of the stirred mixers; in comparison, the spread relating to the static mixer captures the target trends for longer periods.

A possible explanation of LSTM-FC's improved performance for static mixers is that more simulation cases and shorter data lengths are involved, which forms a relatively less challenging learning task. As for the case of stirred mixers, wherever LSTM-FC must deal with longer sequences, either larger training datasets and/or more a sophisticated model is required for a better performance. In contrast, the LSTM Encoder-decoder presents

a wider prediction interval region for both mixers, which mirrors a higher sensitivity to the added noise. Nonetheless, this wider spread correctly captures the target evolution of features relevant to stirred mixers. Such an improvement in performance as well as the higher sensitivity could be attributed to the sequential procedure of the LSTM Encoder-decoder when generating an output sequence: the model is trained to learn the trend step-by-step; therefore, fluctuations in feature values have a profound effect on the model prediction for the next time-step.

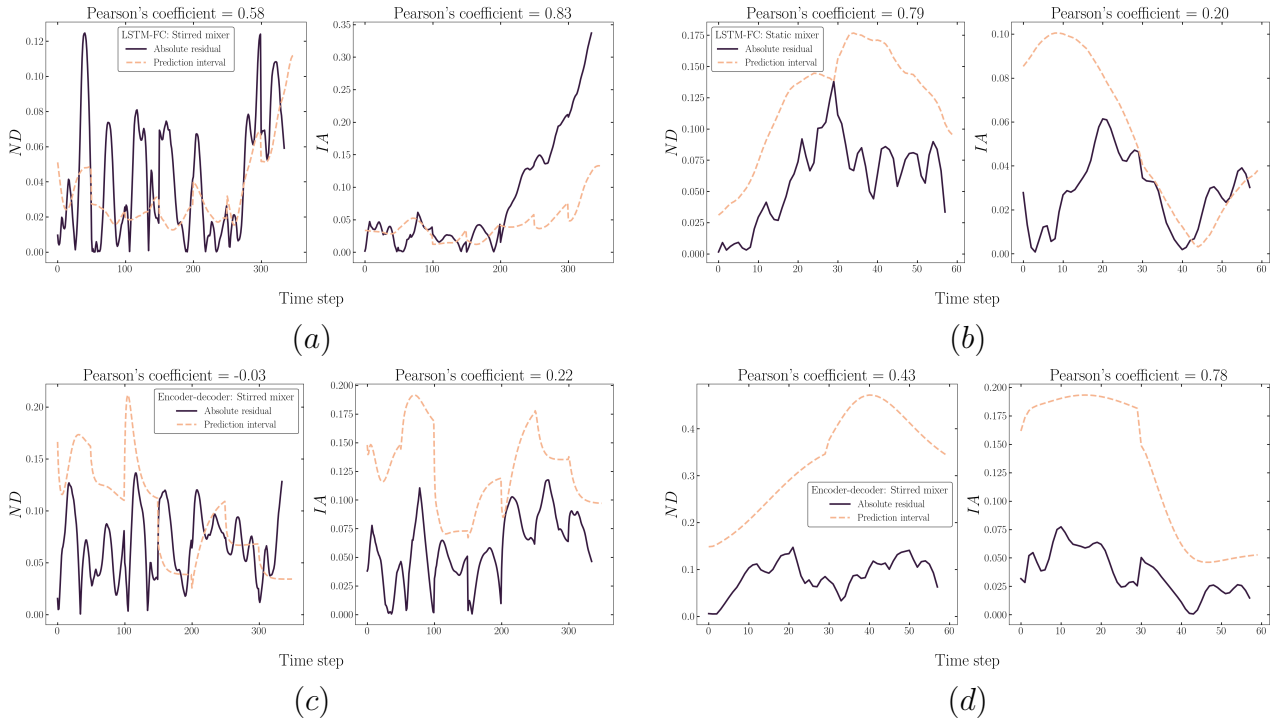


Figure 17: Temporal plots comparing the evolution of spread size and model performance (i.e., absolute residual between model prediction and target at each time-step, $|\widehat{y}_t - y_t|$).

Furthermore, we investigate the correlation between the spread size of the prediction interval region and the model performance along the time axis; here, the model performance is expressed in terms of the divergence of the prediction from its corresponding target at each time-step, which is simply calculated as $|\widehat{y}_t - y_t|$. Pearson's Correlation Coefficient (PCC)[61] is also computed to compare the strength and direction of the relationship between the prediction and the associated target. As presented in Figure 17, in terms of the PCC

(≥ 0.4), the LSTM-FC-based model performance for the features obtained from both mixers is well captured by the spread size. Similarly, the spread size generated via the LSTM Encoder-decoder in the case of the static mixer presents a fairly strong positive correlation with the model performance indicated by a PCC ≥ 0.4. However, the PCC values relevant to the stirred mixer are relatively low, especially for the feature ND, PCC=-0.03. Nonetheless, related studies have suggested that researchers should not rely on the correlation coefficients for identifying the relation between datasets,[62,63] instead, it is essential to plot the data for visual inspection.[61,64] From this perspective, plots shown in Figure 17 provide an insight into the temporal similarity between the spread size and the model performance. It can be seen that, though the spread size does not mirror the exact amplitude of the model performance, it can correctly capture the core trend of the latter. For instance, the exponential increase in IA of stirred mixer (see Figure 17$-(a)$), and the subsequent increase following the reduction in IA of static mixer (see Figure 17$-(b)$). More importantly, we note that the local extrema of the model performance and spread size align (see Figure 17$-(c)$). Thus, the extrema of the spread size signals the deviation of model predictions from the target. This is beneficial when performing sequential prediction since the spread size could help to identify the point where the accuracy of the model prediction begins to deteriorate.

From the above, we suggest that the spread size of the prediction interval region provides an appropriate indicator of model performance and reliability along the sequence generation. This is particularly valid in the case where the trained model is deployed to predict mixing systems that had not been included in the training dataset and for which no access to the model target, or 'truth', is possible.

Conclusions

The current study has demonstrated the application of Recurrent Neural Networks (RNN) to predict the temporal behaviour of key dispersion performance metrics for complex multiphase

mixing systems. Specifically, two Long-Short-Term Memory network architectures, LSTM-FC and LSTM Encoder-decoder, have been finely trained and deployed to predict interfacial area growth, drop count, and their size distribution for future time-steps. Taking advantage of our previous work, the LSTM models developed herein have been trained with data obtained from high-fidelity, three-dimensional, CFD simulations of two mixing systems, carried out for stirred[30,31] and static[32,33] mixers. In particular, the LSTM model has been assigned the task of learning and capturing the influence of varying physicochemical properties, operational conditions, and mixer geometrical characteristics on dispersion performance. Concerning the network architecture, the LSTM-FC has been taught to map a single long output sequence from the input sequence alone, while the LSTM Encoder-decoder has learned to progressively generate future output sequences based on its previous predictions.

We have presented the model performance in terms of the prediction of each dispersion metric obtained for stirred and static mixers. For both model architectures, the prediction accuracy is high at early times and subsequently deteriorates. We have reasoned this to the nature of rollout procedure, wherein the entire dispersion metrics evolution are extrapolated based solely on their initial behaviour during the early stages of the process. In addition, we have demonstrated that it is easier to train an LSTM-FC due to its simplicity, despite, inferior model performance has been seen, which is cause by that the feature evolution of embedded in the sequence is absent. By contrast, the trained LSTM Encoder-decoder is capable of recovering the dispersion dynamics that is incorrectly predicted when the trained LSTM-FC is applied. However, we have suggested that the trained LSTM Encoder-decoder in the current work must be improved since non-trivial deviation on some features is seen. In particular, we present different methods to visualise the model performance relevant to the drop size distribution. These methods include the temporal drop counts in each bin representing a specific drop size range and the histogram of all bins at one time-step. Additionally, Wasserstein distances have been computed to track the similarity between the predicted and targeted distributions. With this score, we have shown the evident discrepancies occur at

the early times, where the corresponding DSD is sensitive to the perturbation in the total drop count due to its small value initially; hence, the deviation gradually diminishes with increase in drop numbers. Lastly, we have proposed an ensemble-based procedure to quantify the model uncertainty where we have further investigated the correlation between the spread size of the prediction interval region and the model performance during prediction via rollout. We have illustrated the high sensitivity of the LSTM Encoder-decoder when concerning the added noise drawn from the same Gaussian distribution as LSTM-FC; meanwhile, the prediction interval via the former verifies its robustness by correctly capturing the true data. Moreover, throughout the correlation investigation, we suggest that the spread size could serve as an appropriate indicator of the evolution of model reliability through the rollout.

Throughout this work, we have delineated a baseline procedure for data pre-processing, re-conditioning and LSTM RNN deployment in the field of complex multi-phase mixing. More importantly, we have implemented a system-agnostic framework capable of seamlessly handling any time-series data with similar structures from other applications, regardless of their origin (i.e., numerical data or experimental measurements), and performing analogous temporal evolution predictions of key metrics. Thus, this study could significantly benefit the chemical engineering community and industrial practitioners by introducing a cost-effective yet robust alternative to the standard CFD modelling tools used nowadays. A spectrum of additional hyperparameters not considered in this work could also affect the model performance, such as the number of LSTM layers, different learning rate schedulers, or dynamic teacher forcing ($d.t.f.$). Future work is suggested to keep exploring the optimal configuration for the LSTM network and implementing variations in the model's architecture.

Acknowledgement

This work is supported by the Engineering and Physical Sciences Research Council, United Kingdom, through the EPSRC MEMPHIS, United Kingdom (EP/K003976/1) and

PREMIERE (EP/T000414/1) Programme Grants. O.K.M. acknowledges the Royal Academy of Engineering for a Research Chair in Multiphase Fluid Dynamics. This work is also supported by the Colombian Ministry of Science, Technology and Innovation, MINCIENCIAS, Colombia, through a doctoral studentship for J.P.V. Simulations have been performed using code BLUE.

References

(1) Valdes, J. P.; Kahouadji, L.; Matar, O. K. Current advances in liquid–liquid mixing in static mixers: A review. *Chemical Engineering Rsearch & Design* **2022**, *177*, 694–731.

(2) Leng, D. E.; Calabrese, R. V. *Handbook of Industrial Mixing - Science and Practice*; John Wiley & Sons, 2004; Chapter 12, pp 639–755.

(3) Håkansson, A. Emulsion Formation by Homogenization: Current Understanding and Future Perspectives. *Annual Review of Food Science and Technology* **2019**, *10*, 239–258.

(4) Middleman, S. Drop Size Distributions Produced by Turbulent Pipe Flow of Immiscible Fluids through a Static Mixer. *Industrial and Engineering Chemistry Process Design and Development* **1974**,

(5) Calabrese, R. V.; Chang, T.; Dang, P. Drop breakup in turbulent stirred-tank contactors. Part I: Effect of dispersed-phase viscosity. *AIChE Journal* **1986**, *32*, 657–666.

(6) Zhou, G.; Kresta, S. M. Correlation of mean drop size and minimum drop size with the turbulence energy dissipation and the flow in an agitated tank. *Chemical Engineering Science* **1998**, *53*, 2063–2079.

(7) Kraume, M.; Gäbler, A.; Schulze, K. Influence of physical properties on drop size distribution of stirred liquid-liquid dispersions. *Chemical Engineering & Technology: Industrial Chemistry-Plant Equipment-Process Engineering-Biotechnology* **2004**, *27*, 330–334.

(8) Rama Rao, N. V.; Baird, M. H.; Hrymak, A. N.; Wood, P. E. Dispersion of high-viscosity liquid-liquid systems by flow through SMX static mixer elements. *Chemical Engineering Science* **2007**,

(9) Zhou, G.; Kresta, S. M. Evolution of drop size distribution in liquid–liquid dispersions for various impellers. *Chemical Engineering Science* **1998**, *53*, 2099–2113.

(10) Pacek, A.; Chamsart, S.; Nienow, A.; Bakker, A. The influence of impeller type on mean drop size and drop size distribution in an agitated vessel. *Chemical Engineering Science* **1999**, *54*, 4211–4222.

(11) Giapos, A.; Pachatouridis, C.; Stamatoudis, M. Effect of the number of impeller blades on the drop sizes in agitated dispersions. *Chemical Engineering Research and Design* **2005**, *83*, 1425–1430.

(12) Lemenand, T.; Dupont, P.; Della Valle, D.; Peerhossaini, H. Turbulent mixing of two immiscible fluids. *Journal of Fluids Engineering, Transactions of the ASME* **2005**,

(13) Sechremeli, D.; Stampouli, A.; Stamatoudis, M. Comparison of mean drop sizes and drop size distributions in agitated liquid–liquid dispersions produced by disk and open type impellers. *Chemical Engineering Journal* **2006**, *117*, 117–122.

(14) Theron, F.; Sauze, N. L. Comparison between three static mixers for emulsification in turbulent flow. *International Journal of Multiphase Flow* **2011**, *37*, 488–500.

(15) Zainal Abidin, M. I. I.; Abdul Raman, A. A.; Mohamad Nor, M. I. Mean drop size correlations and population balance models for liquid—liquid dispersion. *AIChE Journal* **2015**, *61*, 1129–1145.

(16) Lebaz, N.; Sheibat-Othman, N. A population balance model for the prediction of breakage of emulsion droplets in SMX+ static mixers. *Chemical Engineering Journal* **2019**,

(17) Ghotli, R. A.; Abbasi, M. R.; Bagheri, A.; Raman, A. A. A.; Ibrahim, S.; Bostanci, H. Experimental and modeling evaluation of droplet size in immiscible liquid-liquid stirred vessel using various impeller designs. *Journal of the Taiwan Institute of Chemical Engineers* **2019**, *100*, 26–36.

(18) Vikhansky, A. CFD modelling of turbulent liquid–liquid dispersion in a static mixer. *Chemical Engineering and Processing - Process Intensification* **2020**, *149*.

(19) Zhu, L.-T.; Chen, X.-Z.; Ouyang, B.; Yan, W.-C.; Lei, H.; Chen, Z.; Luo, Z.-H. Review of machine learning for hydrodynamics, transport, and reactions in multiphase flows and reactors. *Industrial & Engineering Chemistry Research* **2022**, *61*, 9901–9949.

(20) Calzolari, G.; Liu, W. Deep learning to replace, improve, or aid CFD analysis in built environment applications: A review. *Building and Environment* **2021**, *206*, 108315.

(21) Bengio, Y.; Frasconi, P.; Simard, P. The problem of learning long-term dependencies in recurrent networks. IEEE international conference on neural networks. 1993; pp 1183–1188.

(22) Bengio, Y.; Simard, P.; Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* **1994**, *5*, 157–166.

(23) Kim, H.; Park, M.; Kim, C. W.; Shin, D. Source localization for hazardous material release in an outdoor chemical plant via a combination of LSTM-RNN and CFD simulation. *Computers & Chemical Engineering* **2019**, *125*, 476–489.

(24) Selvaggio, A. Z.; Sousa, F. M. M.; da Silva, F. V.; Vianna, S. S. Application of long short-term memory recurrent neural networks for localisation of leak source using 3D computational fluid dynamics. *Process Safety and Environmental Protection* **2022**, *159*, 757–767.

(25) Zhong, H.; Wei, Z.; Man, Y.; Pan, S.; Zhang, J.; Niu, B.; Yu, X.; Ouyang, Y.; Xiong, Q. Prediction of instantaneous yield of bio-oil in fluidized biomass pyrolysis using long short-term memory network based on computational fluid dynamics data. *Journal of Cleaner Production* **2023**, *391*, 136192.

(26) Mohan, A. T.; Gaitonde, D. V. A deep learning based approach to reduced order modeling for turbulent flow control using LSTM neural networks. *arXiv preprint arXiv:1804.09269* **2018**,

(27) Li, W.; Gao, X.; Liu, H. Efficient prediction of transonic flutter boundaries for varying Mach number and angle of attack via LSTM network. *Aerospace Science and Technology* **2021**, *110*, 106451.

(28) Hou, Y.; Li, H.; Chen, H.; Wei, W.; Wang, J.; Huang, Y. A novel deep U-Net-LSTM framework for time-sequenced hydrodynamics prediction of the SUBOFF AFF-8. *Engineering Applications of Computational Fluid Mechanics* **2022**, *16*, 630–645.

(29) Zhao, L.; Li, Z.; Qu, L.; Zhang, J.; Teng, B. A hybrid VMD-LSTM/GRU model to predict non-stationary and irregular waves on the east coast of China. *Ocean Engineering* **2023**, *276*, 114136.

(30) Liang, F.; Kahouadji, L.; Valdes, J. P.; Shin, S.; Chergui, J.; Juric, D.; Matar, O. K. Numerical study of oil–water emulsion formation in stirred vessels: effect of impeller speed. *Flow* **2022**, *2*, E34.

(31) Liang, F.; Kahouadji, L.; Valdes, J. P.; Shin, S.; Chergui, J.; Juric, D.; Matar, O. K. Numerical simulation of surfactant-laden emulsion formation in an un-baffled stirred vessel. *Chemical Engineering Journal* **2023**, *472*, 144807.

(32) Valdes, J. P.; Kahouadji, L.; Liang, F.; Shin, S.; Chergui, J.; Juric, D.; Matar, O. K. Direct numerical simulations of liquid–liquid dispersions in a SMX mixer under different inlet conditions. *Chemical Engineering Journal* **2023a**, *462*, 142248.

(33) Valdes, J. P.; Kahouadji, L.; Liang, F.; Shin, S.; Chergui, J.; Juric, D.; Matar, O. K. On the dispersion dynamics of liquid–liquid surfactant-laden flows in a SMX static mixer. *Chemical Engineering Journal* **2023b**, *475*, 146058.

(34) Shin, S.; Chergui, J.; Juric, D.; Kahouadji, L.; Matar, O. K.; Craster, R. V. A hybrid interface tracking–level set technique for multiphase flow with soluble surfactant. *Journal of Computational Physics* **2018**, *359*, 409–435.

(35) Hochreiter, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* **1998**, *6*, 107–116.

(36) Gers, F. A.; Schmidhuber, J.; Cummins, F. Learning to forget: Continual prediction with LSTM. *Neural computation* **2000**, *12*, 2451–2471.

(37) Yu, Y.; Si, X.; Hu, C.; Zhang, J. A review of recurrent neural networks: LSTM cells and network architectures. *Neural computation* **2019**, *31*, 1235–1270.

(38) Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural computation* **1997**, *9*, 1735–1780.

(39) Greff, K.; Srivastava, R. K.; Koutník, J.; Steunebrink, B. R.; Schmidhuber, J. LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems* **2016**, *28*, 2222–2232.

(40) Cheng, S.; Prentice, I. C.; Huang, Y.; Jin, Y.; Guo, Y.-K.; Arcucci, R. Data-driven surrogate model with latent data assimilation: Application to wildfire forecasting. *Journal of Computational Physics* **2022**, *464*, 111302.

(41) Cheng, S.; Chen, J.; Anastasiou, C.; Angeli, P.; Matar, O. K.; Guo, Y.-K.; Pain, C. C.; Arcucci, R. Generalised latent assimilation in heterogeneous reduced spaces with machine learning surrogate models. *Journal of Scientific Computing* **2023**, *94*, 11.

(42) Shin, S.; Juric, D. A hybrid interface method for three-dimensional multiphase flows based on front tracking and level set techniques. *International Journal for Numerical Methods in Fluids* **2009**, *60*, 753–778.

(43) Shin, S.; Chergui, J.; Juric, D. A solver for massively parallel direct numerical simulation of three-dimensional multiphase flows. *Journal of Mechanical Science and Technology* **2017**, *31*, 1739–1751.

(44) Staudemeyer, R. C.; Morris, E. R. Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks. 2019.

(45) Gers, F. A.; Schraudolph, N. N.; Schmidhuber, J. Learning precise timing with LSTM recurrent networks. *Journal of machine learning research* **2002**, *3*, 115–143.

(46) Dwarampudi, M.; Subba Reddy, N. V. Effects of padding on LSTMs and CNNs. *CoRR* **2019**, *abs/1903.07288*.

(47) Khotijah, S.; Tirtawangsa, J.; Suryani, A. A. Using LSTM for Context Based Approach of Sarcasm Detection in Twitter. Proceedings of the 11th International Conference on Advances in Information Technology. New York, NY, USA, 2020.

(48) Wang, J.; Xue, M.; Culhane, R.; Diao, E.; Ding, J.; Tarokh, V. Speech Emotion Recognition with Dual-Sequence LSTM Architecture. ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2020; pp 6474–6478.

(49) Chen, S.; Ge, L. Exploring the attention mechanism in LSTM-based Hong Kong stock price movement prediction. *Quantitative Finance* **2019**, *19*, 1507–1515.

(50) Xiao, Y.; Yin, H.; Zhang, Y.; Qi, H.; Zhang, Y.; Liu, Z. A dual-stage attention-based Conv-LSTM network for spatio-temporal correlation and multivariate time series prediction. *International Journal of Intelligent Systems* **2021**, *36*, 2036–2057.

(51) Dai, Y.; Wang, Y.; Leng, M.; Yang, X.; Zhou, Q. LOWESS smoothing and Random Forest based GRU model: A short-term photovoltaic power generation forecasting method. *Energy* **2022**, *256*, 124661.

(52) Rashid, K.; Louis, J. Window-Warping: A Time Series Data Augmentation of IMU Data for Construction Equipment Activity Identification. 36th International Symposium on Automation and Robotics in Construction. 2019; pp 651–657.

(53) Oh, C.; Han, S.; Jeong, J. Time-Series Data Augmentation based on Interpolation. The 17th International Conference on Mobile Systems and Pervasive Computing. 2020; pp 64–71.

(54) Wen, Q.; Sun, L.; Song, X.; Gao, J.; Wang, X.; Xu, H. Time Series Data Augmentation for Deep Learning: A Survey. *CoRR* **2020**, *abs/2002.12478*.

(55) Gawlikowski, J.; Tassi, C. R. N.; Ali, M.; Lee, J.; Humt, M.; Feng, J.; Kruspe, A.; Triebel, R.; Jung, P.; Roscher, R.; others A survey of uncertainty in deep neural networks. *Artificial Intelligence Review* **2023**, *56*, 1513–1589.

(56) Zhu, Y. Ensemble forecast: A new approach to uncertainty and predictability. *Advances in atmospheric sciences* **2005**, *22*, 781–788.

(57) Leutbecher, M.; Palmer, T. N. Ensemble forecasting. *Journal of computational physics* **2008**, *227*, 3515–3539.

(58) Parker, W. S. Ensemble modeling, uncertainty and robust predictions. *Wiley interdisciplinary reviews: Climate change* **2013**, *4*, 213–223.

(59) Kantorovich, L. V. On the translocation of masses. *Journal of mathematical sciences* **2006**, *133*, 1381–1382.

(60) Panaretos, V. M.; Zemel, Y. Statistical aspects of Wasserstein distances. *Annual review of statistics and its application* **2019**, *6*, 405–431.

(61) Asuero, A. G.; Sayago, A.; González, A. The correlation coefficient: An overview. *Critical reviews in analytical chemistry* **2006**, *36*, 41–59.

(62) Armstrong, R. A. Should Pearson's correlation coefficient be avoided? *Ophthalmic and Physiological Optics* **2019**, *39*, 316–327.

(63) Derrick, T. R.; Bates, B. T.; Dufek, J. S. Evaluation of time-series data sets using the Pearson product-moment correlation coefficient. *Medicine and science in sports and exercise* **1994**, *26*, 919–928.

(64) Anscombe, F. J. Graphs in statistical analysis. *The american statistician* **1973**, *27*, 17–21.

TOC Graphic

Some journals require a graphical entry for the Table of Contents. This should be laid out "print ready" so that the sizing of the text is correct.
Inside the `tocentry` environment, the font used is Helvetica 8 pt, as required by *Journal of the American Chemical Society*.
The surrounding frame is 9 cm by 3.5 cm, which is the maximum permitted for *Journal of the American Chemical Society* graphical table of content entries. The box will not resize if the content is too big: instead it will overflow the edge of the box.
This box and the associated title will always be printed on a separate page at the end of the document.