Multi-domain encoder–decoder neural networks for latent data assimilation in dynamical systems

Sibo Cheng, Yilin Zhuang, Lyes Kahouadji, Che Liu, Jianhua Chen, Omar K Matar, Rossella Arcucci

Multi-domain encoder-decoder neural networks for latent data assimilation in dynamical systems

Sibo Cheng*[1,2], Yilin Zhuang[3], Lyes Kahouadji[3], Che Liu[4], Jianhua Chen[3,5], Omar K. Matar[3], Rossella Arcucci[4]

[1] *CEREA, École des Ponts and EDF R&D, Île-de-France, France*
[2] *Data Science Institute, Department of Computing, Imperial College London, UK*
[3] *Department of chemical engineering, Imperial College London, UK*
[4] *Department of Earth Science & Engineering, Imperial College London, UK*
[5] *State Key Laboratory of Multiphase Complex Systems, Institute of Process Engineering, Chinese Academy of Sciences, China*
**corresponding: sibo.cheng@enpc.fr*

Abstract

High-dimensional dynamical systems often require computationally intensive physics-based simulations, making full physical space data assimilation impractical. Latent data assimilation methods perform assimilation in reduced-order latent space for efficiency but struggle with complex, nonlinear state-observation mappings. Recent solutions like Generalised Latent Data Assimilation (GLA) and Latent Space Data Assimilation (LSDA) address heterogeneous latent spaces by incorporating surrogate mapping functions but introduce computational costs and uncertainties. Furthermore, current algorithms that integrate data assimilation and deep learning still face limitations when it comes to handling non-explicit mapping functions. To address these challenges, this paper introduces a novel deep-learning-based data assimilation scheme, named Multi-domain Encoder-Decoder Latent Data Assimilation (MEDLA), capable of handling diverse data sources by sharing a common latent space. The proposed approach significantly reduces the computational burden since the complex mapping functions are mimicked by the multi-domain encoder-decoder neural network. It also enhances assimilation accuracy by minimizing interpolation and approximation errors. Extensive numerical experiments from three different test cases assess MEDLA's performance in high dimensional dynamical systems, benchmarking it against state-of-the-art latent data assimilation methods. The numerical results consistently underscore MEDLA's superiority in managing multi-scale observational data and tackling intricate, non-explicit mapping functions.

Keywords: Data assimilation; Deep learning; Data fusion; Dynamical systems

Main Notations

Latent data assimilation methodology

$\mathbf{x}, \tilde{\mathbf{x}}$	state vector in the full and the reduced space
$\mathbf{y}, \tilde{\mathbf{y}}$	observation vector in the full and the reduced space
$\mathbf{X}, \tilde{\mathbf{X}}$	a set of state vectors in the full and the reduced space
$\mathbf{Y}, \tilde{\mathbf{Y}}$	a set of observation vectors in the full and the reduced space
T_{y}	a set of time steps where observations are available for assimilation
$\mathcal{E}, \mathcal{D}$	encoder and decoder neural networks
$\mathcal{E}_x, \mathcal{D}_x$	state and observation encoders
$\tilde{\mathbf{x}}_x, \tilde{\mathbf{x}}_y$	latent state vector encoded using state and observation encoders
J^{AE}	loss function of encoder-decoders
J_t	loss function of data assimilation at time t
$\mathbf{L}_{\mathbf{X},q}$	POD projection operator with truncation parameter q
$m_{\mathrm{in}}, m_{\mathrm{out}}$	input and output sequence of the predictive model
t_F	number of total time steps
$\tilde{\mathbf{x}}_{b,t}, \tilde{\mathbf{x}}_{a,t}$	background and analysis state vectors in the latent space at time t
$\tilde{\mathbf{y}}_t$	observation vector in the latent space at time t
$\tilde{\mathcal{H}}_t, \tilde{\mathbf{H}}_t$	state-observation mapping function and its linearization
$\tilde{\mathbf{B}}_t, \tilde{\mathbf{R}}_t$	background and observation error covariance matrices in the latent spaces

2D Burgers' equation test case

u, v	velocity field
Re	Reynolds number
$\mathbf{S}_{y,t}, \mathbf{C}_y^e$	observation error covariance and correlation matrices
$\sigma_{y,t}^{(i)}$	observation error deviations at vector coordinate i
Δ_t	time lag between background and true states

Multiphase flow test case

$\alpha_k, \rho_k, \boldsymbol{U}_{\mathrm{k}}$	concentration, density and velocity fields for oil/water phase
$\boldsymbol{M}_{\mathrm{k}}$	rate of momentum transfer per unit volume
U_{m}	initial mixture velocity
$f_{\mathcal{H}}(.)$	marginal mapping function

Microfluidic drop test case

$\mathbf{u}, \mathbf{F}$	velocity and pressure fields
ρ	density
$\mathbf{F}$	local surface tension force at the interface of drops

1. Introduction

For high-dimensional dynamical systems, running high-fidelity physics-based simulations can be time-consuming. To circumvent the computational burden, Machine Learning (ML)-based low-dimensional surrogate models have been widely applied in engineering problems, including climate forecasting [43], air pollution modelling [8], computational fluid dynamics (CFD) [30, 54], nuclear reactor physics [23, 22] and ocean engineering [28]. To adjust the surrogate model prediction, real-time information, for instance, from local sensors or satellites, can be employed through Data Assimilation (DA) algorithms [7]. However, due to the high-dimensionality and the complexity of the transformation function, implementing DA in the full physical space can be computationally difficult, if not infeasible. Some recent works also use generative models such as Generative Adversarial Networks (GANs) [20] or diffusion models [17] to link the state space to the observation space. However, the combination of generative models and data assimilation techniques is still under exploration. Many recent research efforts [49, 8, 24, 56, 15, 39, 37, 12, 33, 34, 38, 2, 11] have been given to reduce the computational burden by developing reduced order surrogate models and performing DA in low-dimensional spaces, issued from Proper Orthogonal Decomposition (POD), dynamic mode decomposition or ML-based auto-encoders. Such algorithms, known as Latent data Assimilation (LA), can benefit from the efficiency of reduced-order surrogate modelling and the accuracy of DA. In [8], the authors suggest using Recurrent Neural Network (RNN) within a reduced space to improve future predictions by learning assimilated results. A related concept is presented in [5], which introduces an iterative DA-ML scheme. However, it is worth noting that when implementing this algorithm, retraining the Neural Network (NN) is necessary whenever new observations are obtained.

In recent two years, online LA has raised much research attention. The methods proposed can be broadly categorised into two groups [13]: LA [39, 34, 36] where the full observations are used to correct the reduced-order models; LA$^+$ [1, 12, 33] where the state variables and observations are encoded into a common latent space. The former methods are more suitable for chaotic dynamical systems with limited observation data where it is difficult to construct a low-dimensional latent space for the system's observations. The LA$^+$ approaches enable more efficient assimilation, especially for dense observation mappings. However, it requires the states and observations to share the same encoding (compression) function, necessitating them to be defined in the same physical space. This becomes challenging with highly non-linear state-observation transformation mappings, which are common in real-world DA problems [7]. As a result, different autoencoders are needed for the states and observations, leading to heterogeneous latent spaces. To address this challenge, recent approaches [11, 37] utilize local surrogate functions, such as polynomial functions [11] and Multi layer percepton (MLP) [37], to bridge the two latent spaces. These methods are known as Generalised Latent Assimilation (GLA) and Latent Space Data Assimilation (LSDA), respectively. By employing surrogate functions, variational DA becomes feasible by solving a local optimization problem. These approaches significantly enhance the accuracy of the surrogate models in practical applications.

It is important to note, however, that the computation of local surrogate functions around the predicted latent variables must be performed online, which can result in relatively high computational costs. More importantly, considerable uncertainties can be introduced when mapping the two latent spaces [11] especially when the choice of the approximation range for the surrogate function is inappropriate. The latter can be difficult since the prediction error of the surrogate model is often out of reach. Recent

research [53] addresses the difficulty of complex state-observation mapping by proposing a new DA scheme, named Deep Data Assimilation (DDA), which employs an observation-domain encoder and a state-domain decoder. Using this method allows for direct transfer of observation data to the state space. However, it is important to note that DA is still necessary in the full physical space. In conclusion, conducting reduced-order assimilation with multi-domain or multi-scale data sources remains a major challenge in current DA schemes.

In this paper, we propose a novel LA scheme with a multi-domain encoder-decoder which can perform both state-in-state-out and observation-in-state-out encoding-decodings. More precisely, it consists of training two encoders which share the same decoder on an alternating basis with separate loss functions. To ensure alignment in the latent space, fine-tuning can be performed on the observation-domain encoder using encoded state variables as output. The idea of multi-domain or multi-modal encoding-decoding has been introduced for computer vision [52], nature language processing [42] and transfer learning [59]. To the best of the authors' knowledge, using multi-domain encoders to adjust the prediction of dynamical systems has not been presented in the literature before. A surrogate predictive model can then be trained in the common latent space. As for the DA step, similar to LA$^+$ [1, 33], only linear LA is required since the observation and the state can be encoded to the same latent space. The novel approach, named Multi-domain Encoder-Decoder Latent data Assimilation (MEDLA), can combine the efficiency of LA$^+$ and the generalizability of GLA/LSDA. Furthermore, MEDLA also aims to improve the LA accuracy by avoiding/reducing interpolation (e.g., in LA$^+$) or approximation (e.g., in GLA/LSDA) errors. Different data compression/transformation strategies employed in LA$^+$, GLA/LSDA, DDA and MEDLA are illustrated in Figure 1. A qualitative comparison of different approaches, regarding the novel MEDLA method, is depicted in Table 1.

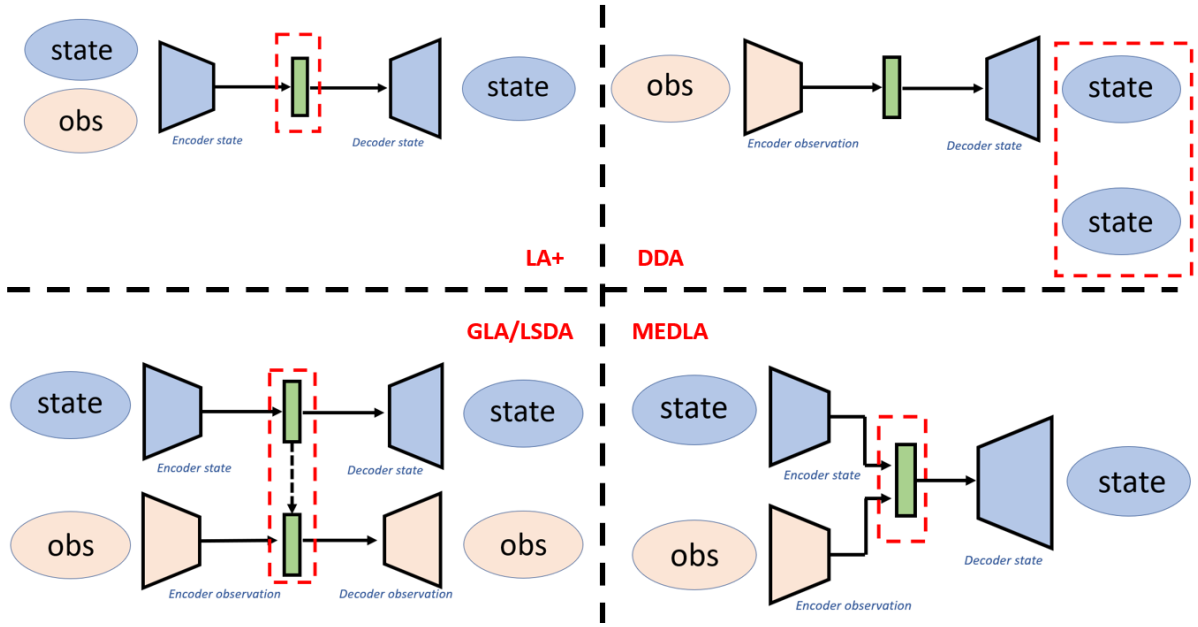


Figure 1: Workflows of different latent spaces in LA approaches. The dashed red rectangles indicate the space where DA takes place

To test the performance of MEDLA in comparison with the state-of-the-art LA approaches, three numerical experiments are designed in this work. The first one involves

solving the two-dimensional Burgers' equation [6] on squared meshes, where both states and observations are the velocity field on a different scale. MEDLA is compared against LA$^+$ [1] in terms of assimilation accuracy with different levels of observation errors. The second test involves CFD simulations of a multiphase flow systems in a pipe with two non-linear state-observation transformation functions, and the performance of MEDLA is compared to GLA [11]. The last test case involves drop interactions in a microfluidics device with multi-modal data comprising CFD and camera observations, for which no explicit transformation function could be identified. Existing LA approaches that rely on explicit transformation functions are unable to handle this scenario. However, thanks to the multi-domain encoder-decoder, the proposed MEDLA can successfully assimilate the reduced-order state variables using non-explicit observations.

In summary, in this paper, we make the following main contributions:

- We introduce a novel latent data assimilation scheme, called MEDLA, that leverages multi-domain encoding-decoding to enhance the efficiency and accuracy of DA for high-dimensional dynamical systems, addressing key challenges in current DA schemes.
- We demonstrate that MEDLA is capable of handling complex and non-explicit mapping functions efficiently.
- We perform extensive numerical experiments with both synthetic and physical assimilation problems to highlight the advantage of MEDLA in terms of accuracy and efficiency compared to the state-of-the-art latent data assimilation algorithms.

The rest of this paper is organized as follows. Section 2 presents the state-of-the-art LA approaches. The novel method MEDLA is introduced in Section 3, followed by the numerical experiments of three test cases in Section 4. We end the paper with a conclusion and a future work discussion in Section 5.

Table 1: Comparison of existing deep learning-assisted assimilation approaches in relation to the experiments in this paper

Methods	Reduced state	Reduced observation	Non-linear mapping	Non-explicit mapping
RODDA [8]	✓	✗	✓	✗
LA [39, 34]	✓	✗	✗	✗
LA$^+$ [1, 12, 33]	✓	✓	✗	✗
GLA [11]	✓	✓	✓	✗
LSDA [37]	✓	✓	✓	✗
DDA [53]	✗	✗	✓	✓
MEDLA	✓	✓	✓	✓
Experiments				
Burgers equation				
Multiphase flow				
Microfluidic drops				

2. Latent data assimilation for high-dimensional systems

In this section, we introduce the technical background of LA algorithms, including reduced-order modelling, RNN-based surrogate models, and data assimilation in low-dimensional latent spaces.

2.1. Reduced-order modelling

Here, we introduce two types of Deep Learning (DL)-based reduced-order modellings (ROMs), namely the Convolutional Autoencoder (CAE) and the Singular Value Decomposition (SVD) Autoencoder (AE).

2.1.1. Convolutional autoencoder

Autoencoding as an unsupervised ML approach and has been widely applied for data compression, especially in high-dimensional systems. A DL-based autoencoder consists of two neural networks: an encoder $\mathcal{E}$ which maps the input variables $\mathbf{x} = [x_1, x_2, ..., x_n] \in \mathbb{R}^n$ to a low-dimensional vector $\tilde{\mathbf{x}}$, and a decoder $\mathcal{D}$ for reconstructing variables $\mathbf{x}'$ in the full physical space. More precisely,

$$\tilde{\mathbf{x}} = \mathcal{E}(\mathbf{x}) \quad \text{and} \quad \mathbf{x}' = \mathcal{D}(\tilde{\mathbf{x}}). \tag{1}$$

The encoder $\mathcal{E}$ and decoder $\mathcal{D}$ must be trained jointly. Since the objective here is to minimize the mismatch between the original and the reconstructed state variables in the full physical space, the training loss function could be defined as, for instance, the mean square error (MSE),

$$J^{\mathrm{AE}}(\mathcal{E}, \mathcal{D}) = \frac{1}{N_{\mathrm{train}}} \sum_{j=1}^{N_{\mathrm{train}}} ||\mathbf{x}_j - \mathcal{D} \circ \mathcal{E}(\mathbf{x}_j)||^2. \tag{2}$$

where $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{N_{\mathrm{train}}}\}$ is the training dataset. Fully-connected MLP can be employed to construct $\mathcal{E}$ and $\mathcal{D}$. However, the large number of parameters of MLP can be cumbersome for high-dimensional systems. Furthermore, local patterns are ignored since MLP treats each input pixel/mesh in a parallel way. By using CAE, we can overcome these drawbacks thanks to the convolutional layers, which capture spatial patterns in the original system with a much smaller number of parameters. More precisely, multi-dimensional filters are employed in convolutional layers to recognize local patterns by sliding the layer inputs. Many research works have demonstrated that CAE outperforms classical linear compression methods in terms of reconstruction accuracy on a variety of applications [48].

2.1.2. SVD autoencoder

Despite the efficiency of CAE, difficulties can be found when facing unstructured/unsquared data since the convolutional filter can only capture structured (in most cases squared) pixels/nodes. Much effort has been devoted to tackling this bottleneck. Proposed solutions include, for example, space-filling curves [26], spatially varying kernels [60] or graph-based networks [57]. In this work, one of the test cases involves non-squared cylinder meshes. Thus, we make use of a recently developed training-efficient ROM, named SVD AE [40, 41]. As the first step of dimension reduction, we apply SVD to obtain the full set of principal components (PCs) $\mathbf{L}_{\mathbf{x}}$ of the training dataset $\mathbf{X}$, i.e.,

$$\mathbf{X} = \Big[\mathbf{x}_1|\mathbf{x}_2, ..., |\mathbf{x}_{N_{\text{train}}}\Big],$$
$$\mathbf{C}_\mathbf{x} = \mathbb{E}(\mathbf{X}\mathbf{X}^T) = \mathbf{L}_\mathbf{X}\mathbf{D}_\mathbf{X}{\mathbf{L}_\mathbf{X}}^T, \tag{3}$$

where $\mathbf{C}_\mathbf{x}$ represents the covariance matrix of $\mathbf{X}$ and $\mathbf{D}_\mathbf{X}$ is a diagonal matrix, containing the eigenvalues of $\mathbf{C}_\mathbf{x}$. By keeping the first q eigenvectors (which correspond to the q largest eigenvalues), we obtain the SVD-compressed vector $\mathbf{L}_{\mathbf{X},q}^T\mathbf{X}$. Here q is also known as the truncation parameter in POD or truncated SVD. To further reduce the system dimension, a dense autoencoder $(\mathcal{E}', \mathcal{D}')$ with fully connected layers is involved,

$$\tilde{\mathbf{x}} = \mathcal{E}'\Big({\mathbf{L}_{\mathbf{X},q}}^T\mathbf{x}\Big), \quad \text{while} \quad \mathbf{x}' = \mathbf{L}_{\mathbf{X},q}\mathcal{D}'(\tilde{\mathbf{x}}). \tag{4}$$

Since the SVD AE first maps the physical fields to their principal components, it can handle both structured and unstructured meshes. Recent research [11] has also numerically demonstrated the advantage of SVD AE compared to standard POD, especially when the latent space is of extremely small dimension. A more advanced approach that combines POD and deep learning for reduced order modelling has been introduced in [18] where the principle components are also engaged in the training process.

2.2. Low-dimensional surrogate model

Once the ROM is performed, it is crucial to understand the dynamics in the low-dimensional latent space. Much effort has been given to predict latent variables via machine learning approaches, such as Random Forest (RF) [23], RNN[1, 12] or Transformers [21]. LA techniques can be implemented in combination with all these mentioned predictive models to enhance forecasting. Since we are aiming for long-term predictions of physical systems, long short-term memory (LSTM) neural netowk [27], a variant of RNN is chosen in this paper to build the surrogate model. Capable of dealing with long-term time dependencies, LSTM can address the vanishing gradient problem [27] which can be crumblesome for other variants of RNNs. LSTM can also deliver sequence-to-sequence predictions (i.e., m_{in} time steps as input and m_{out} time steps as output), which can decrease the online computational time, and more importantly, reduce the accumulated prediction error. For a time series of encoded latent variables $\tilde{\mathbf{X}}_{\text{train}} = [\tilde{\mathbf{x}}_1^{\text{train}}, \tilde{\mathbf{x}}_2^{\text{train}}, ..., \tilde{\mathbf{x}}_{T_{\text{train}}}^{\text{train}}]$, the training of LSTM can be carried out by shifting the starting time step:

$$\begin{aligned}
[\tilde{\mathbf{x}}_1^{\text{train}}, \tilde{\mathbf{x}}_2^{\text{train}}, ..., \tilde{\mathbf{x}}_{m_{in}}^{\text{train}}] &\xrightarrow{\text{LSTM} \quad \text{train}} [\tilde{\mathbf{x}}_{m_{in}+1}^{\text{train}}, \tilde{\mathbf{x}}_{m_{in}+2}^{\text{train}}, ..., \tilde{\mathbf{x}}_{m_{in}+m_{out}}^{\text{train}}], \\
[\tilde{\mathbf{x}}_2^{\text{train}}, \tilde{\mathbf{x}}_3^{\text{train}}, ..., \tilde{\mathbf{x}}_{m_{in}+1}^{\text{train}}] &\xrightarrow{\text{LSTM} \quad \text{train}} [\tilde{\mathbf{x}}_{m_{in}+2}^{\text{train}}, \tilde{\mathbf{x}}_{m_{in}+3}^{\text{train}}, ..., \tilde{\mathbf{x}}_{m_{in}+m_{out}+1}^{\text{train}}] \\
&\vdots \\
[\tilde{\mathbf{x}}_{\text{train}-m_{in}-m_{out}+1}^{\text{train}}, ..., \tilde{\mathbf{x}}_{\text{train}-m_{out}}^{\text{train}}] &\xrightarrow{\text{LSTM} \quad \text{train}} [\tilde{\mathbf{x}}_{\text{train}-m_{out}+1}^{\text{train}}, ..., \tilde{\mathbf{x}}_{T_{\text{train}}}^{\text{train}}].
\end{aligned} \tag{5}$$

Different loss functions, such as MSE or mean absolute error (MAE), can be employed in the training phase by measuring the mismatch between predicted and true latent variables. As for the online prediction of $\tilde{\mathbf{X}}_{\text{test}} = [\tilde{\mathbf{x}}_1^{\text{test}}, \tilde{\mathbf{x}}_2^{\text{test}}, ..., \tilde{\mathbf{x}}_{T_{\text{test}}}^{\text{test}}]$, an iterative process can be involved for long-term forecasting,

$$\begin{aligned}
[\tilde{\mathbf{x}}_1^{\text{test}}, \tilde{\mathbf{x}}_2^{\text{test}}, ..., \tilde{\mathbf{x}}_{m_{in}}^{\text{test}}] &\xrightarrow{\text{LSTM} \quad \text{predict}} [\tilde{\mathbf{x}}_{m_{in}+1}^{\text{test}}, \tilde{\mathbf{x}}_{m_{in}+2}^{\text{test}}, ..., \tilde{\mathbf{x}}_{m_{in}+m_{out}}^{\text{test}}], \\
[\tilde{\mathbf{x}}_{m_{in}+1}^{\text{test}}, \tilde{\mathbf{x}}_{m_{in}+2}^{\text{test}}, ..., \tilde{\mathbf{x}}_{m_{in}+m_{out}}^{\text{test}}] &\xrightarrow{\text{LSTM} \quad \text{predict}} [\tilde{\mathbf{x}}_{m_{in}+m_{out}+1}^{\text{test}}, ..., \tilde{\mathbf{x}}_{m_{in}+2m_{out}}^{\text{test}}] \\
&\vdots
\end{aligned} \tag{6}$$

When applying such an iterative model, minimising accumulated prediction error is of the most importance.

2.3. Latent data assimilation

LA techniques are developed to perform efficient DA in a low-dimensional latent space. It can thus be used for real-time forecasting corrections for dynamical systems [13]. At a given time step t, the prediction result (also known as the background state) in the latent space is denoted as $\tilde{\mathbf{x}}_{b,t}$. The available observations, either in the full or the reduced space, and the state-observation mapping (also known as the transformation function) are denoted as $\tilde{\mathbf{y}}_t$ and $\tilde{\mathcal{H}}_t$, respectively. The objective function of latent data assimilation reads

$$J_t(\tilde{\mathbf{x}}) = \frac{1}{2}(\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_{b,t})^T \tilde{\mathbf{B}}_t^{-1}(\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_{b,t}) + \frac{1}{2}(\tilde{\mathbf{y}}_t - \tilde{\mathcal{H}}_t(\tilde{\mathbf{x}}))^T \tilde{\mathbf{R}}_t^{-1}(\tilde{\mathbf{y}}_t - \tilde{\mathcal{H}}_t(\tilde{\mathbf{x}})), \tag{7}$$

where $\tilde{\mathbf{B}}_t$ and $\tilde{\mathbf{R}}_t$ represent the error covariance related to $\tilde{\mathbf{x}}_{b,t}$ and $\tilde{\mathbf{y}}_t$, respectively. The assimilated latent state $\tilde{\mathbf{x}}_{a,t}$ can be obtained via the minimization of the objective function,

$$\tilde{\mathbf{x}}_{a,t} = \underset{\tilde{\mathbf{x}}}{\operatorname{argmin}}\Big(J_t(\tilde{\mathbf{x}})\Big). \tag{8}$$

In the works of [1, 12, 33], the latent transformation function is supposed to be linear. In other words, a linear operator $\tilde{\mathbf{H}}_t$ is used to replace the transformation function $\tilde{\mathcal{H}}_t$ in Equation (8). Therefore, the minimization of Equation (7) can be performed via the Best Linear Unbiased Estimator (BLUE), and the analysed state $\tilde{\mathbf{x}}_{a,t}$ can be calculated explicitly as:

$$\tilde{\mathbf{x}}_{a,t} = \tilde{\mathbf{x}}_{b,t} + \tilde{\mathbf{K}}_t(\tilde{\mathbf{y}}_t - \tilde{\mathbf{H}}_t\tilde{\mathbf{x}}_{b,t}) \tag{9}$$

where the latent Kalman gain matrix $\tilde{\mathbf{K}}_t$ is defined as:

$$\tilde{\mathbf{K}}_t = \tilde{\mathbf{B}}_t\tilde{\mathbf{H}}_t^T(\tilde{\mathbf{H}}_t\tilde{\mathbf{B}}_t\tilde{\mathbf{H}}_t^T + \tilde{\mathbf{R}}_t)^{-1}. \tag{10}$$

However, in a wide range of DA applications, it is almost infeasible to compress state variables and observations into the same latent space. To perform LA with complex transformation function, recent works [11, 37] proposed novel algorithms, named GLA and LSDA, to build simplified surrogate functions $\tilde{\mathcal{H}}_t^s$ linking the state and the observation latent spaces. More precisely, in GLA local polynomial regressions are used to build $\tilde{\mathcal{H}}_t^s$ in a neighbourhood of $\tilde{\mathbf{x}}_{b,t}$. Thanks to the smoothness of polynomial functions, quasi-Newton methods can be employed to minimize the approximated objective function,

$$J_t^s(\tilde{\mathbf{x}}) = \frac{1}{2}(\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_{b,t})^T \tilde{\mathbf{B}}_t^{-1}(\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_{b,t}) + \frac{1}{2}(\tilde{\mathbf{y}}_t - \tilde{\mathcal{H}}_t^s(\tilde{\mathbf{x}}))^T \tilde{\mathbf{R}}_t^{-1}(\tilde{\mathbf{y}}_t - \tilde{\mathcal{H}}_t^s(\tilde{\mathbf{x}})). \tag{11}$$

In LSDA, multi-layer perceptron (MLP) neural networks are used to build $\tilde{\mathcal{H}}_t^s$ instead of polynomial functions. Ensemble DA techniques [14] are then applied to overcome the difficulties of inverting deep learning functions. The DA accuracy of these approaches has been demonstrated in high-dimensional systems. However, the construction of $\tilde{\mathcal{H}}_t^s$, as well as the minimization or the ensemble approximations, must be performed online, leading to considerable computational time. The complexity of such algorithms can be increased when observations of various resources/scales exist since different local surrogate functions need to be established online. More importantly, the choice of the sampling range and the polynomial degree in GLA is crucial and might require careful tuning. As discussed in [11], a too small sampling range will decrease the generalizability while an excessively large one may lead to inaccurate approximations.

3. Multi-domain encoder-decoder for latent data assimilation (MEDLA)

3.1. Joint encoder-decoder

The general workflow of MEDLA is presented in Figure 2, which can include different types of encoders/decoders and predictive models. The essential idea of MEDLA is to encode both the state and the observations into the same latent space, as shown in Figure 2, thus no surrogate functions are required to link them, as performed in GLA and LSDA. Therefore, the encoder of observations needs to be trained jointly with the decoder of state variables, that is,

$$\tilde{\mathbf{x}}_x = \mathcal{E}_x(\mathbf{x}), \quad \tilde{\mathbf{x}}_y = \mathcal{E}_y(\mathbf{y}), \quad \mathbf{x'}_x = \mathcal{D}(\tilde{\mathbf{x}}_x), \quad \text{and} \quad \mathbf{x'}_y = \mathcal{D}(\tilde{\mathbf{x}}_y), \tag{12}$$

where $\mathcal{E}_x, \mathcal{E}_y$ are state and observation encoders, respectively, sharing the same state decoder $\mathcal{D}$. The concept is to create latent vectors capable of capturing the characteristics of both state variables and observations. Therefore, we attempt to reduce the mismatch $||\mathbf{x}-\mathbf{x'}_x||, ||\mathbf{x}-\mathbf{x'}_y||$, and $||\tilde{\mathbf{x}}_x-\tilde{\mathbf{x}}_y||$ where $||.||$ represents a vector distance measure. The minimization of the former two residual terms can be performed using available/historical data via the following loss functions,

$$\begin{aligned} J_x(\mathcal{E}_x, \mathcal{D}) &= \frac{1}{N^x_{\text{train}}} \sum_{j=1}^{N^x_{\text{train}}} ||\mathbf{x}_j - \mathcal{D} \circ \mathcal{E}_x(\mathbf{x}_j)||^2, \\ J_y(\mathcal{E}_y, \mathcal{D}) &= \frac{1}{N^y_{\text{train}}} \sum_{j=1}^{N^y_{\text{train}}} ||\mathbf{x}_j - \mathcal{D} \circ \mathcal{E}_y(\mathbf{y}_j)||^2, \end{aligned} \tag{13}$$

where $N^x_{\text{train}}, N^y_{\text{train}}$ denotes the size of the training dataset for $(\mathcal{E}_x, \mathcal{D})$ and $(\mathcal{E}_y, \mathcal{D})$ respectively. The two encoder-decoders must be trained separately because the encoder cannot read the state and observation simultaneously. In this work, we propose to train $(\mathcal{E}_x, \mathcal{D})$ and $(\mathcal{E}_y, \mathcal{D})$ on an alternating basis as summarised in Algorithm 1. In particular, since the parameters in the decoder $\mathcal{D}$ are trained jointly both with $\mathcal{E}_x$ and $\mathcal{E}_y$, decreasing learning rates r_x, r_y (for $(\mathcal{E}_x, \mathcal{D})$ and $(\mathcal{E}_y, \mathcal{D})$ respectively) as defined in Algorithm 1, could assist the convergence of $\mathcal{D}$.

Fine tunings could be required to ensure the latent domain alignment (i.e., $\tilde{\mathbf{x}}_x \approx \tilde{\mathbf{x}}_y$) since the decoder function $\mathcal{D} : \mathbb{R}^{\dim(\tilde{\mathbf{x}})} \to \mathbb{R}^n$ is not necessarily bijective, thus

$$\mathcal{D}(\tilde{\mathbf{x}}_x) = \mathcal{D}(\tilde{\mathbf{x}}_y) \not\longrightarrow \tilde{\mathbf{x}}_x = \tilde{\mathbf{x}}_y. \tag{14}$$

In the field of domain adaption and transfer learning, much effort has been devoted to improving the latent domain alignment for information from different resources. The state-of-the-art approaches, involving, for instance, discriminative neural network [51], contrastive learning [58] or meta-learning [55], attempting to build a common latent space which can represent the information in different input spaces. Compared to classical domain alignment tasks, the temporal pattern is extremely important for dynamical systems. In other words, the correspondence between encoded states $\tilde{\mathbf{x}}_x$ and encoded observations $\tilde{\mathbf{x}}_y$ at the same time step needs to be established. To ensure this temporal synchronisation, we propose to perform fine-tunings for $\mathcal{E}_y$ via the loss function,

$$\begin{aligned} J^{\text{FT}}_y(\mathcal{E}_y) &= \frac{1}{N^{x,y}_{\text{train}}} \sum_{j}^{N^{x,y}_{\text{train}}} ||\tilde{\mathbf{x}}_{x,j} - \tilde{\mathbf{x}}_{y,j}||^2, \\ &= \frac{1}{N^{x,y}_{\text{train}}} \sum_{j}^{N^{x,y}_{\text{train}}} ||\mathcal{E}_x(\mathbf{x}) - \mathcal{E}_y(\mathbf{y})||^2, \end{aligned} \tag{15}$$

Algorithm 1: Training of multi-domain encoder-decoder

Parameters:
Epoch size: $N_{\text{epoch}}, N_{\text{epoch}}^{FT}$
Batch size: $L_{\text{batch}}^{x}, L_{\text{batch}}^{y}, L_{\text{batch}}^{x,y}$

Inputs:
Train/Validation state dataset:
$\mathbf{X}_{\text{train}} = \left[\mathbf{x}_1|\mathbf{x}_2, ..., |\mathbf{x}_{N_{\text{train}}^{x}}\right], \mathbf{X}_{\text{val}} = \left[\mathbf{x}_1|\mathbf{x}_2, ..., |\mathbf{x}_{N_{\text{val}}^{x}}\right]$
Train/Validation observation dataset:
$\mathbf{Y}_{\text{train}} = \left[\mathbf{y}_1|\mathbf{y}_2, ..., |\mathbf{y}_{N_{\text{train}}^{y}}\right], \mathbf{Y}_{\text{val}} = \left[\mathbf{y}_1|\mathbf{y}_2, ..., |\mathbf{y}_{N_{\text{val}}^{y}}\right]$
Initial learning rate: $r_x, r_y, r_{x,y}$
Initial weight parameters for encoder-decoders: $\mathbf{W}_{\mathcal{E}_x}, \mathbf{W}_{\mathcal{E}_y}, \mathbf{W}_{\mathcal{D}}$

Algorithm:
while $n_{\text{epoch}} \leq N_{\text{epoch}}$ **do**
  for L_{batch}^{x} in 1 to $N_{\text{train}}^{x}/L_{\text{batch}}^{x}$ **do**
    for iter in 1 to L_{batch}^{x} **do**
      $\text{train\_loss}_x = J_x(\mathcal{E}_x, D)$
      $\mathbf{W}_{\mathcal{E}_x}, \mathbf{W}_{\mathcal{D}} \leftarrow Adam(\text{train\_loss}_x, r_x)$
    end
  end
  for L_{batch}^{y} in 1 to $N_{\text{train}}^{y}/L_{\text{batch}}^{y}$ **do**
    for iter in 1 to L_{batch}^{y} **do**
      $\text{train\_loss}_y = J_y(\mathcal{E}_y, D)$
      $\mathbf{W}_{\mathcal{E}_y}, \mathbf{W}_{\mathcal{D}} \leftarrow Adam(\text{train\_loss}_y, r_y)$
    end
  end
  compute $\text{val\_loss}_x, \text{val\_loss}_y$
  if $\text{val\_loss}_x < \text{min\_val\_loss}_x$ (resp. $\text{val\_loss}_y < \text{min\_val\_loss}_y$) **then**
    $\text{min\_val\_loss}_x = \text{val\_loss}_x$ (resp. $\text{min\_val\_loss}_y = \text{val\_loss}_y$)
    $n\_\text{patience}_x = n\_\text{patience}_y = 0$
  end
  else
    $n\_\text{patience}_x += 1$ (resp. $n\_\text{patience}_y += 1$)
    if $n\_\text{patience}_x == N_{\text{patience}}$ (resp. $n\_\text{patience}_y == N_{\text{patience}}$) **then**
      Reduce r_x (resp. r_y)
    end
  end
  $n_{\text{epoch}} += 1$
end
while $n_{\text{epoch}}^{FT} \leq N_{\text{epoch}}^{FT}$ **do**
  for iter in 1 to $L_{\text{batch}}^{x,y}$ **do**
    $\text{train\_loss}_{x,y} = J_y^{FT}(\mathcal{E}_y)$
    $\mathbf{W}_{\mathcal{E}_y} \leftarrow Adam(\text{train\_loss}_{x,y}, r_{x,y})$
  end
  $n_{\text{epoch}}^{FT} += 1$
end

outputs: $\mathbf{W}_{\mathcal{E}_x}, \mathbf{W}_{\mathcal{E}_y}, \mathbf{W}_{\mathcal{D}}$

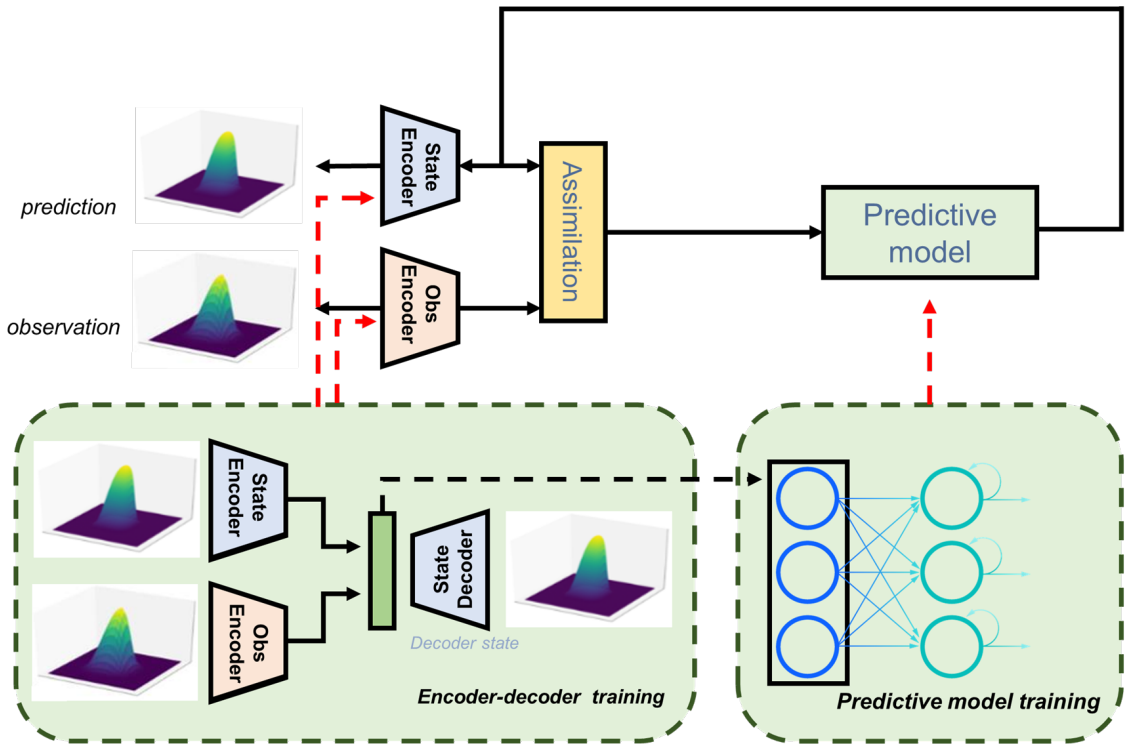


Figure 2: Flowchart of the proposed MEDLA approach

where $N_{\text{train}}^{x,y}$ denotes the size of the joint training set, and the state encoder $\mathcal{E}_x$ is frozen during the fine tuning as shown in Algorithm 1. It is important to note that a joint training using a combination of Equations (13) and (15) would also be an alternative solution to train the multi-domain encoder-decoder. However, one will then only be able to use the snapshots where observations are available for training. Given the sparse nature of the observation data, this will lead to a significant reduction in the amount of available training data.

3.2. Assimilation for dynamical systems

After completing the joint encoding-decoding process, the predictive model can be trained using encoded state vectors, as described in Equation (5). When real-time observations become available during prediction, LA techniques can be applied to adjust the system's prediction directly in the latent space by encoding the observations. In fact, by applying $\mathcal{E}_y$, the observations can be easily compressed into the same latent space of state variables. The transformation operator, which maps the full state space to the full observation space is thus included in the encoder-decoder functions. Therefore only linear DA is required in MEDLA as described in Algorithm 2. The assimilated latent state $\tilde{\mathbf{x}}_{a,t}$ can be used as the starting point for the next-level prediction on an iterative basis as depicted in Figure 2. Compared to GLA or LSDA approaches, MEDLA can considerably reduce the computational cost for online LA mainly because i) the DA is linear; ii) no online computation of local surrogate functions is required.

4. Numerical experiments

In this section, we present three numerical experiments as shown in Table 1 aimed at assessing the effectiveness of the novel method MEDLA in comparison to state-of-the-art LA methods. We provide two implementations of MEDLA in Python using the two most

Algorithm 2: MEDLA with trained encoder-decoders

Parameters:
Number of time-steps: t_{F}
Predictive model input/ouput length: $m_{\mathrm{in}}, m_{\mathrm{out}}$

Inputs:
Initial states: $\mathbf{x}_t$, for $t \in \{0..t_{\mathrm{x}}\}$
Observation data: $\mathbf{y}_t$, for $t \in T_{\mathrm{y}}$
Estimated latent covariance matrices: $\{\hat{\mathbf{B}}_t\}, \{\hat{\mathbf{R}}_t\}$
Predictive function: f^p

Algorithm:
Encoding $\tilde{\mathbf{x}}_{x,t} = \mathcal{E}_x(\mathbf{x}_t), t \in \{0..t_{\mathrm{x}}\}$
Initialization: $t = t_x$
while $t < t_{\mathrm{F}}$ **do**
  $\{\tilde{\mathbf{x}}_{x,t+1}, \tilde{\mathbf{x}}_{x,t+2}, ..., \tilde{x\mathbf{x}}_{x,t+m_{\mathrm{out}}}\} = f^p\Big(\tilde{\mathbf{x}}_{x,t-m_{\mathrm{in}}}, \tilde{\mathbf{x}}_{x,t-m_{\mathrm{in}}+1}, ..., \tilde{\mathbf{x}}_{x,t}\Big)$
  for j from $t+1$ to $t+m_{\mathrm{out}}$ **do**
    if $j \in T_y$ **then**
      $\tilde{\mathbf{x}}_{y,j} = \mathcal{E}_y(\mathbf{y}_j)$
      $J_j(\tilde{\mathbf{x}}) = \frac{1}{2}(\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_{x,j})^T \tilde{\mathbf{B}_j}^{-1}(\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_{x,j}) + \frac{1}{2}(\tilde{\mathbf{x}}_{y,j} - \tilde{\mathbf{x}})^T \tilde{\mathbf{R}_j}^{-1}(\tilde{\mathbf{x}}_{y,j} - \tilde{\mathbf{x}})$
      $\tilde{\mathbf{x}}_{a,j} = \underset{\tilde{\mathbf{x}}}{\mathrm{argmin}}\Big(J_j(\tilde{\mathbf{x}})\Big).$
      $\tilde{\mathbf{x}}_{x,j} \longleftarrow \tilde{\mathbf{x}}_{a,j}$
    end
  end
  $t \longleftarrow t + m_{\mathrm{in}}$
end

outputs: $\{\tilde{\mathbf{x}}_t, \mathcal{D}(\tilde{\mathbf{x}}_t), t \in \{0..t_F\}\}$

widely adopted deep learning libraries, namely TensorFlow and PyTorch. The numerical experiments of the 2D Burgers' equation and the shallow water models are performed using Tensorflow while the experiments of microfluidic drop interactions are done with Pytorch.

4.1. 2D Burger's equation

4.1.1. Experiment setup

The first test case consists of a 2D viscous Burgers' equation problem where the governing equation (in a 2D space with (x, y) as coordinate) reads

$$\begin{aligned} \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} &= \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \\ \frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} &= \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right), \end{aligned} \tag{16}$$

where Re is a parameter called the Reynolds number, a measure of the flow inertia; u, v in Equation (16) denote the velocity components, and t represents the time of the dynamical system. These physical quantities are all nondimensionalised. Introduced by [3], Burgers'

equation can be considered as a simplification of the Navier-Stokes equations without pressure gradients. It has been widely applied in modelling fluid dynamics, traffic flows, and mass transport [31]. As shown in Figure 3 (a), the initial condition of this numerical test is chosen to be in the form of a square column of liquid of a certain radius that is released at $t = 0$. The initial and boundary conditions of the velocity field are set as

$$u_{t=0} = v_{t=0} = u_{\text{boundary}} = v_{\text{boundary}} = 1. \tag{17}$$

Equation (16) is solved numerically using a first-order finite difference method with a backward difference scheme. Two simulation scales (128×128) and (32×32) are used to form the state variables and observations, respectively. For the sake of consistency, the time interval between two consecutive steps of the state simulations is also four times finer than the one of the observation simulations. In this test case, we attempt to correct the model prediction by assimilating coarse grid simulations in the same velocity field.

Since the simulations are carried out using square meshes, CAE is chosen to perform the reduced-order modelling. The training and test datasets for the encoder-decoders are generated by simulations using different Reynolds numbers. All the snapshots in the simulations of the training dataset are used to train the multi-domain encoder-decoder. We compare numerically the assimilated results of LA$^+$ [1] and MEDLA with different interpolation methods. The former uses the state auto-encoder to encode the observation data after interpolation while the latter compresses both states and interpolated observations into the same latent space thanks to a joint encoder-decoder.

To further inspect the robustness of the proposed approach, Gaussian noises have been added to the observations, i.e.,

$$\mathbf{y}_t^{\text{noisy}} = \mathbf{y}_t + \epsilon_{y,t} \quad \text{and} \quad \epsilon_{y,t} \sim \mathcal{N}(0, \mathbf{S}_{y,t}), \tag{18}$$

where $\mathbf{S}_{y,t}$ denotes the observation error covariance matrix in the full velocity field

$$\mathbf{S}_{y,t} = (\Sigma_{y,t})\mathbf{C}_y^e(\Sigma_{y,t}), \tag{19}$$

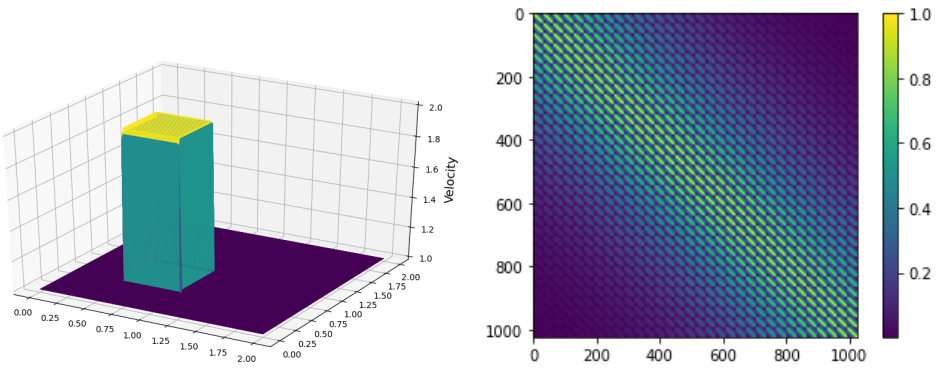
$\Sigma_{y,t}$ represents the marginal error standard deviation of each observation point in the 2D space,

$$\Sigma_{y,t} = diag(\sigma_{y,t}^{(1)}, \sigma_{y,t}^{(2)}, ..., \sigma_{y,t}^{\dim(\mathbf{y})}) \tag{20}$$

and $\mathbf{C}_y^e$ is the error correlation matrix in the observation space i.e., (32×32). In this experiment, both spatially-independent and correlated errors have been tested. The former makes use of an identity correlation matrix, i.e., $\mathbf{C}_y^e = \mathbf{I}_{\dim(\mathbf{y})}$ while correlated observation errors are generated using second-order auto-regressive functions,

$$\mathbf{C}_y^e(i,j) = (1 + \frac{r(i,j)}{L}) \exp(-\frac{r(i,j)}{L}), \quad \forall\{i,j\} \in \{1, 2, ..., \dim(\mathbf{y})\} \tag{21}$$

where $r(.)$ represents the spatial distance of two nodes in the 2D space; L is known as the correlation scale length, set to $L = 4$ in this example. Such a correlation matrix is presented in Figure 3(b). The error deviations $\{\sigma_{y,t}^{(i)}\}, i \in \{1, 2, ..., \dim(\mathbf{y})\}$ are set to be proportional to the exact observation values.



(a) Initial (b) loss

Figure 3: (a): The velocity field u at $t = 0$; (b): Error correlation matrix ($32^2 \times 32^2$) with second-order autoregressive function

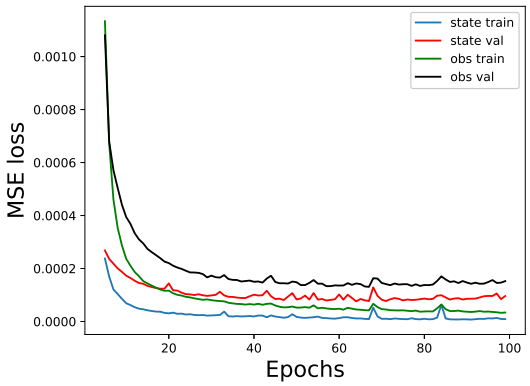


Figure 4: Training loss (MSE) for the multi-domain encoder-decoder from the 5th Epoch

4.1.2. Results

In this twin experiment, both the sate-in-state-out and observation-in-state-out encoder-decoders are trained over 100 epochs, as shown in Figure. 4 where the loss functions of training and validation datasets are displayed. The validation dataset consists of 10% snapshots randomly chosen among the training data. Both encoder-decoders achieve stable loss values after 100 epochs despite the fact that some oscillations can be observed. To test the performance of MEDLA, the objective of this twin experiment is to reconstruct the velocity field $u_{t=800}$. Simulations at different time steps, namely $t = 400,\ 600,$ and 1000 are viewed as background states (i.e., model predictions). In other words, a time lag of $\Delta_t = -400, -200$ or $+200$ is imposed between predictions and the ground truth. Numerical results of MEDLA are compared against those of LA$^+$ [1] using either linear or spline cubic interpolations [35] in the full observation space. We illustrate in Figure 5, the averaged assimilation error $\bar{\epsilon}_{\Delta_t}$ against the setted relative observation error deviation, varying from 0% to 45%. For a fair comparison, Monte Carlo tests are employed with an ensemble size of $N_{ens} = 50$. The assimilation error $\bar{\epsilon}_{\Delta_t}$ can then be

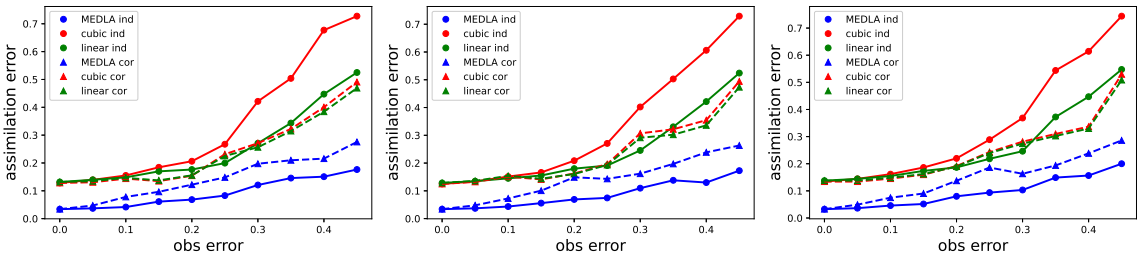
estimated,

$$\mathbf{y}^{\text{noisy}}_{t,(j)} = \mathbf{y}_t + \epsilon_{y,t,(j)} \quad \text{with} \quad \epsilon_{y,t,(j)} \sim \mathcal{N}(0, \mathbf{S}_{y,t}), \quad \forall j \in 1, ..., N_{ens} \tag{22}$$

$$\tilde{\mathbf{x}}_{b,\Delta_t} = \mathcal{E}_x(u_{t=800+\Delta_t}), \quad \tilde{\mathbf{y}}_{t,(j)} = \mathcal{E}_y(\mathbf{y}^{\text{noisy}}_{t,(j)}) \tag{23}$$

$$\tilde{\mathbf{x}}_{a,\Delta_t,(j)} = \text{DA}(\tilde{\mathbf{x}}_{b,\Delta_t}, \tilde{\mathbf{y}}_{t,(j)}), \quad \text{where} \quad \text{DA} \in \{\text{LA}^+, \text{MEDLA}\} \tag{24}$$

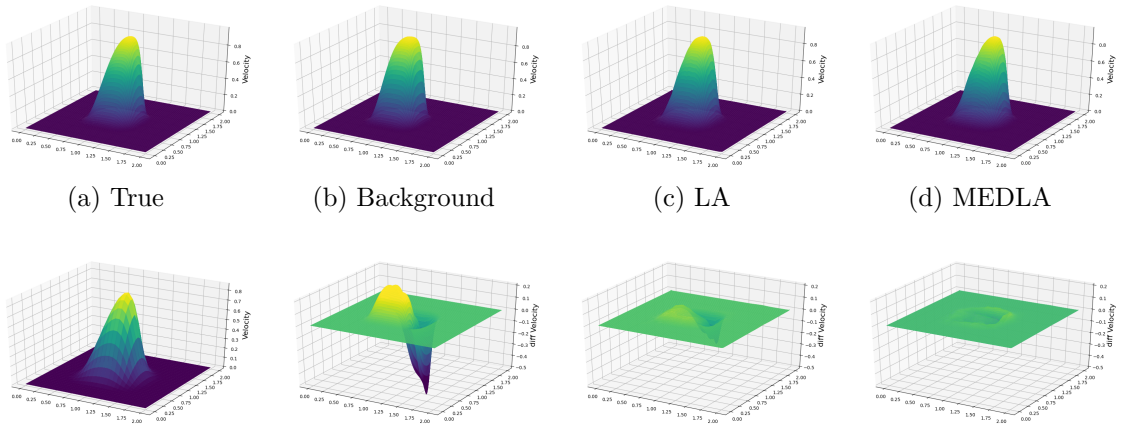
$$\bar{\epsilon}_{\Delta_t} = \frac{1}{N_{ens}} \sum_{j}^{N_{ens}} ||u_{t=800} - \mathcal{D}\big(\tilde{\mathbf{x}}_{a,\Delta_t,(j)}\big)||_2 \big/ ||u_{t=800}||_2. \tag{25}$$



(a) $\Delta_t = -400$ (b) $\Delta_t = -200$ (c) $\Delta_t = +200$

Figure 5: Assimilation error against observation error level estimated using Monte Carlo methods with 50 experiments for each error deviation

As shown by Figure 5, the reconstruction error of MEDLA is considerably lower compared to LA$^+$ methods with both independent or correlated observation noises. The advantage of MEDLA is more substantial when observation error deviation increases, showing the strong robustness of the proposed approach. Compared to existing LA approaches, MEDLA avoids performing interpolation on noisy data, which might introduce extra uncertainties. On the other hand, MEDLA performs slightly better with independent observation errors in comparison to correlated errors since denoising independent errors is easier for the convolutional layer by capturing local patterns.



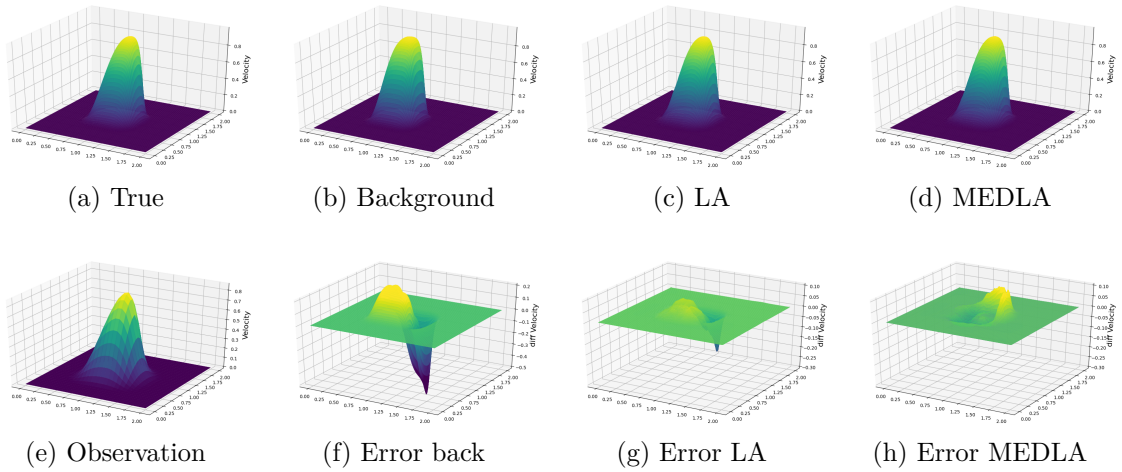
(a) True (b) Background (c) LA (d) MEDLA

(e) Observation (f) Error back (g) Error LA (h) Error MEDLA

Figure 6: Results of different latent data assimilation approaches without observation errors

To further inspect the assimilation performance, we display in Figures 6 and 7, the assimilated velocity fields either without or with correlated observation noises, respectively.

Two assimilation methods are compared: MEDLA and LA$^+$ with linear interpolation. In both cases, the background time lag is fixed as $\Delta_t = -200$. What can be clearly observed in Figures 6 and 7 is that the MEDLA approach can significantly better adjust this time lag, regardless of the level of observation noise.



(a) True (b) Background (c) LA (d) MEDLA

(e) Observation (f) Error back (g) Error LA (h) Error MEDLA

Figure 7: Results of different latent data assimilation approaches with correlated observation errors. The relative error deviation is set to be 20%

4.2. Multiphase flow modelling

4.2.1. Flow system setup

To evaluate the performance of MEDLA with highly non-linear observations, a high-dimensional multiflow CFD model is implemented in this study. Current LA methods face challenges in handling such complex scenarios [11, 13]. The CFD modelling consists of the two-phase flow of silicone oil and water in a pipe with a length of 4 m and a diameter 26 mm, as shown in Figure 8. The experimental flow rig [50] was simulated by using a cylindrical mesh of $\dim(\mathbf{x}) = 180,000$ cells, as also shown in Figure 8. Eulerian-Eulerian simulations are performed through the open-source CFD platform OpenFOAM (version 8.0), and population balance models [32] are used to model the droplet size and coalescence behaviour.

The governing equations of the Eulerian framework are given as below:

$$\frac{\partial}{\partial t}\left(\alpha_{\mathrm{k}}\rho_{\mathrm{k}}\right) + \nabla\cdot\left(\alpha_{\mathrm{k}}\rho_{\mathrm{k}}\,\boldsymbol{U}_{\mathrm{k}}\right) = 0, \tag{26}$$

$$\frac{\partial}{\partial t}\left(\alpha_{\mathrm{k}}\rho_{\mathrm{k}}\boldsymbol{U}_{\mathrm{k}}\right) + \nabla\cdot\left(\alpha_{\mathrm{k}}\rho_{\mathrm{k}}\boldsymbol{U}_{\mathrm{k}}\boldsymbol{U}_{\mathrm{k}}\right) = -\alpha_{\mathrm{k}}\nabla p + \nabla\cdot\left(\alpha_{\mathrm{k}}\boldsymbol{\tau}_{\mathrm{k}}\right) + \alpha_{\mathrm{k}}\rho_{\mathrm{k}}\boldsymbol{g} + \boldsymbol{M}_{\mathrm{k}}, \tag{27}$$

where the subscript k represents the phases of water and oil, respectively, and $\boldsymbol{\tau}$ is the stress tensor expressed as

$$\boldsymbol{\tau}_{\mathrm{k}} = \mu_{\mathrm{eff}}\left[\nabla\boldsymbol{U}_{\mathrm{k}} + \left(\nabla\boldsymbol{U}_{\mathrm{k}}\right)^{\mathrm{T}} - \frac{2}{3}\left(\nabla\cdot\boldsymbol{U}_{\mathrm{k}}\right)I\right]. \tag{28}$$

In Equation (27), α_k, ρ_k and $\boldsymbol{U}_{\mathrm{k}}$ represents the concentration, density, and velocity of each phase, respectively; $\boldsymbol{M}_{\mathrm{k}}$ denotes the rate of momentum transfer per unit volume. More details of the CFD models can be found in our recent work [9]. As shown in

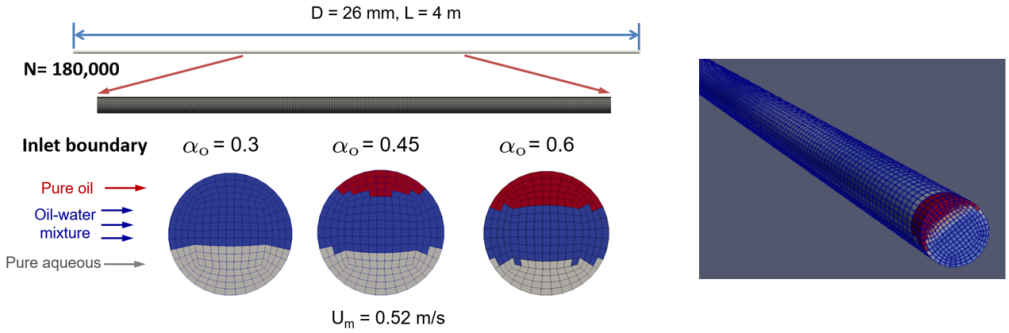


Figure 8: Dimension and parameters of the pipe and the two-phase flow

Table 2, the two test cases explored in this work have an initial mixture velocity of U_{m} = 1.04m/s. Each CFD simulation is performed with a uniform time step of 0.005s and the flow time is set to 10s, ensuring convergence at the current mesh resolution.

The objective here is to predict the oil concentration $\alpha_{\mathrm{oil},t}$ via a low-dimensional surrogate model, which can be updated using LA techniques. Since unsquared meshes are used in the CFD modelling, the ROM is performed using SVD AE in this test case. Once encoded latent vectors are computed, a LSTM neural network is trained on a training dataset (as described in Equation (5)) to build the surrogate model.

In terms of real-time observations $\{\mathbf{y}_t = [y_{1,t}, ..., y_{m,t}]\}$, synthetic data is used. Following the set up of [11], the transformation operator $\mathcal{H}$ in the full space consists of a selection operator $\mathbf{H}$ and a marginal non-linear function $f_{\mathcal{H}}$:

$$\mathbf{y}_t = \begin{bmatrix} y_{1,t} \\ y_{2,t} \\ \vdots \\ y_{\dim(\mathbf{y}),t} \end{bmatrix} = \mathcal{H}(\mathbf{x}_t) = \mathbf{H} f_{\mathcal{H}}(\mathbf{x}_t) = \begin{bmatrix} \mathbf{H}_{1,1}, \dots \mathbf{H}_{1,\dim(\mathbf{x})} \\ \vdots \\ \mathbf{H}_{\dim(\mathbf{y}),0}, \dots \mathbf{H}_{\dim(\mathbf{y}),\dim(\mathbf{x})} \end{bmatrix} \begin{bmatrix} f_{\mathcal{H}}(x_{1,t}) \\ f_{\mathcal{H}}(x_{2,t}) \\ \vdots \\ f_{\mathcal{H}}(x_{\dim(\mathbf{x}),t}) \end{bmatrix} \tag{29}$$

$$\text{with} \quad \mathbf{H}_{i,j} = \begin{cases} 0 & \text{with probability} \quad 1-P \\ 1 & \text{with probability} \quad P \end{cases},$$

$$\text{where} \quad \{i,j\} \in \{1, ..., \dim(\mathbf{y})\} \times \{1, ..., \dim(\mathbf{x})\}.$$

The dimension of the observation vectors are fixed as $\dim(\mathbf{y}) = 30000$ in this example. A randomly-generated selection operator is commonly used for testing the performance of DA algorithms (e.g., [10, 16]). In this work, we choose a sparse representation with $P = 0.1\%$. As in [11], two marginal non-linear functions $f_{\mathcal{H}}$ are employed,

- quadratic function: $f_{\mathcal{H}}(x) = x^2$
- reciprocal function: $f_{\mathcal{H}}(x) = 1/(x + 0.5)$.

Both transformation functions are tested to evaluate the performance of MEDLA.

The training of encoder-decoders is implemented using both CFD simulations, including 1600 snapshots, with different initial conditions, as presented in Table 2. The dataset is homogeneously split into a training dataset (including 10% of validation data) and a test dataset with 800 snapshots each. After ROM, a LSTM surrogate model with $m_{in} = m_{out} = 10$ (see Equation (5)) is built to predict the evolution of latent variables. Following Equation (5), the sequence-to-sequence LSTM uses the same training

dataset as the encoder-decoders. To further examine the performance of MEDLA under varying levels of prediction errors, DA with two LSTM models, namely LSTM100 and LSTM1000, are evaluated. LSTM100 is trained for 100 epochs, representing a prediction model with some level of noise, while LSTM1000 is trained for 1000 epochs, representing a more accurate prediction model.

Table 2: Operating parameters of CFD simulations

U_{m} (m s^{-1})	ϵ_{o}	$h_{\mathrm{C0}}^{+} = h_{\mathrm{C0}}/D$	${h_{\mathrm{O0}}}^{+} = h_{\mathrm{O0}}/D$	${h_{\mathrm{P0}}}^{+} = h_{\mathrm{P0}}/D$	d_{320} (mm)
1.04	0.15	0.405	0.997	0.92	1.14
1.04	0.3	0.189	0.997	0.92	1.27

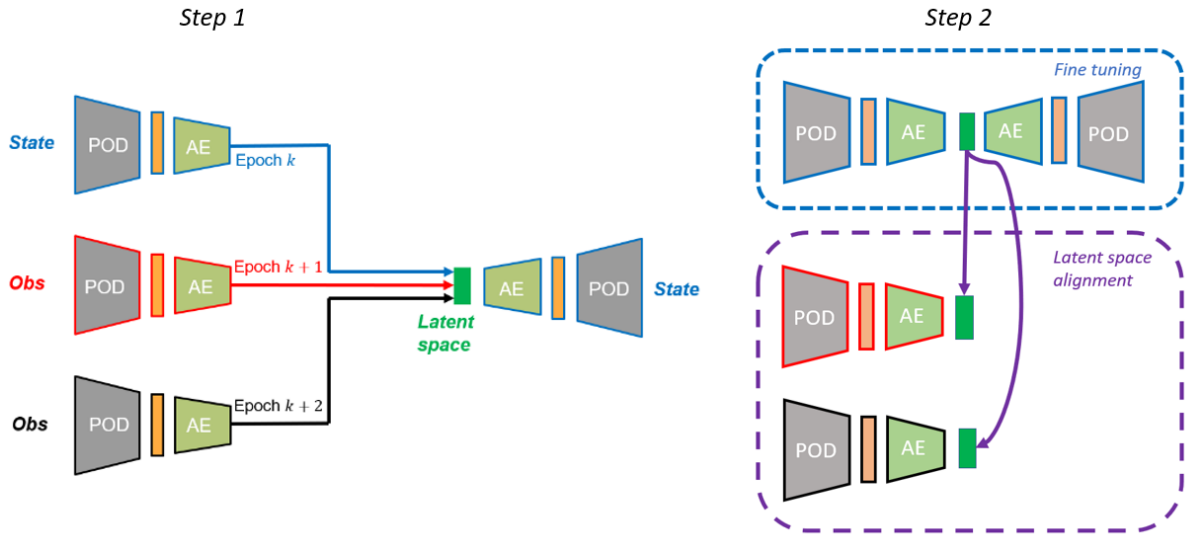


Figure 9: Workflow of multi-domain encoder-decoder with SVD AE and two observation resources for the multiphase flow modelling

4.2.2. Results

Since assimilation algorithms need to incorporate both observations (see Section 4.2.1), three different encoders are required in this test case, as shown in Figure 9 (step 1). In other words, the state-decoder should manage to reconstruct the full physical field by reading one of the observation quantities as input. During the training procedure, an alternate among these three encoders is necessary. The loss functions of this alternating training (300 epochs for each encoder-decoder) with an adaptive decreasing learning rate $r_x, r_y \in [10^{-2}, 10^{-3}, 10^{-4}]$ are shown in Figure 10(a). Both training and validation losses decrease significantly against the number of epochs, despite the fact that more oscillations can be found compared to Figure 4. This is mainly due to the complexity of the transformation function (Equation (29)) with two different observation mappings. Therefore, following Equation (15), fine tuning is implemented here to ensure the latent space alignment as shown in Figure 9 (step 2).

We evaluate the L^2 reconstruction error on both the training and the test datasets as illustrated in Table 3. For all different encoder-decoders, the error on the test dataset is very close to the one on the training dataset, showing the robustness of the joint encoder-decoders with a low level of overfitting.

Table 3: Relative reconstruction error of encoder-decoders

	PC space		Full space	
	train	test	train	test
state	3.15%	3.22%	3.21%	3.28%
square	3.16%	3.21%	3.22%	3.26%
Recip	5.22%	5.31%	5.26%	5.34%

The MSE loss of LSTM training is shown in Figure 10 (b) and the vertical line corresponds to the 100th epoch where the training of LSTM100 terminates. We apply both LSTM models on the CFD simulation of $\epsilon_o = 0.3$ (i.e., the second row of Table 2). The forecasting starts at the 100th time step (i.e., $t = 1s$). The predicted evolution of latent variables against encoded CFD (considered as ground truth in the latent space) is shown in Figure 11 (a-d) and 12 (a-d), respectively. As expected, LSTM1000 considerably outperforms LSTM100 for all four latent variables (LV1 - LV4) presented. MEDLA is then performed with both LSTM100 and LSTM1000 using either square or reciprocal transformation mapping. The assimilations take place every 100 time steps for 10 consecutive snapshots. In other words,

$$T_y = \{150, ..., 159, 250, ..., 259, ..., 950, ..., 959\} \tag{30}$$

in Algorithm 2. Assimilated latent features are shown in Figure 11 (e-l) and 12 (e-l). Thanks to MEDLA, the mismatch between encoded CFD and predicted latent variables can be significantly reduced in all cases. This fact highlights the robustness of MEDLA regarding different levels of prior noises. It enhances not only the assimilated steps (where observations are available) but also next-level predictions regardless of the observation operator and prediction error level. These results are consistent with the observation from the decoded full physical space as shown in Figure 13.

We compare the performance of MEDLA against the GLA algorithm, which has been previously implemented in this multiphase flow test case [11], in Table 3. For the hyperparameters of GLA, we have chosen $d^p = 4$ for the degree of the surrogate polynomial function and $r_s = 0.3$ for the relative sampling range. These parameters have shown the best performance of GLA on the same application in the previous work of [11]. When incorporating with the '*noisy*' predictive model LSTM100, both MEDLA and GLA effectively reduce prediction errors. Nevertheless, MEDLA consistently outperforms GLA, achieving a relative MSE reduction of over 5%. When employed alongside the '*accurate*' predictive model LSTM100, GLA struggles to further reduce prediction errors significantly. This limitation arises from the approximation function (detailed in Section 2), which connects the two latent spaces and introduces an additional layer of error, as previously observed in [11]. On the other hand, MEDLA succeeds in further enhancing the prediction accuracy with both observation functions thanks to its great robustness. The readers are referred to [11] for the implementation details of GLA. The averaged inference time, including prediction, assimilation and decoding, is also presented in Table 4. Due to its well-designed encoder-decoder structure, MEDLA seamlessly integrates multi-domain physics data, resulting in exceptionally rapid inference time that closely aligns with the original LSTM predictions. This test case vividly illustrates MEDLA's proficiency in effectively assimilating multi-domain data characterized by complex and nonlinear mapping functions.

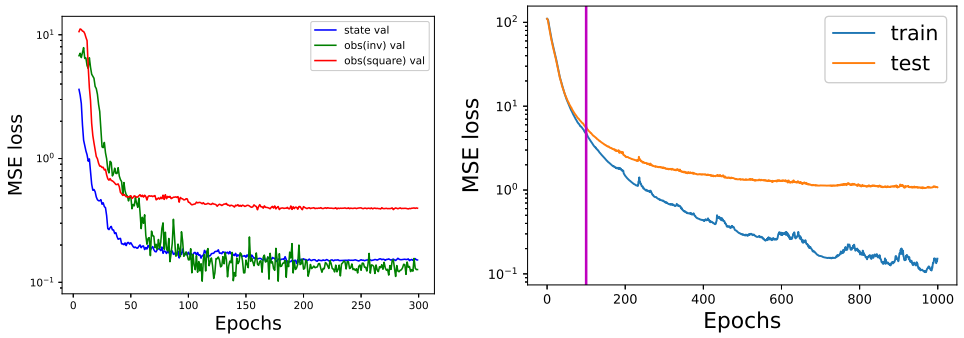
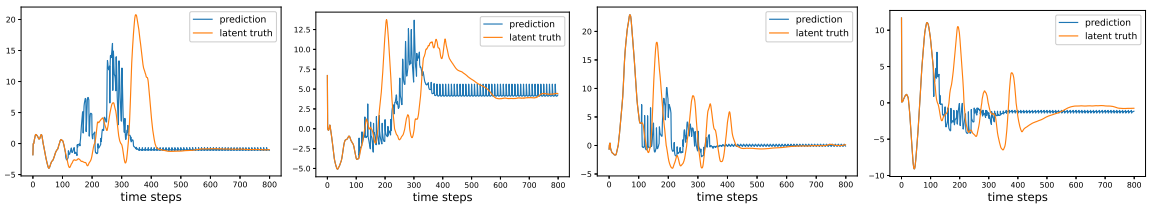
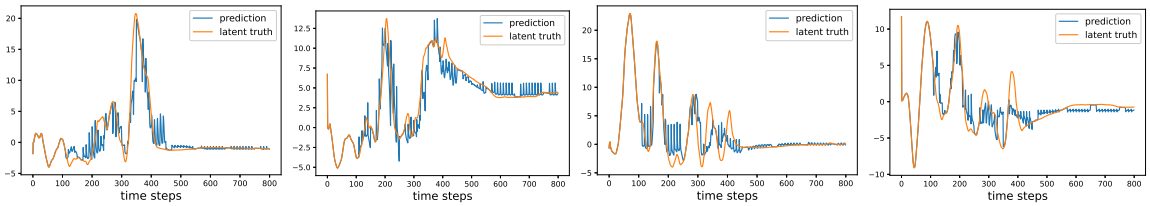


Figure 10: Training loss (MSE) for (a) the multi-domain encoder-decoder from the 5th Epoch; (b) the two forward model LSTM100 (until the verticle line) and LSTM1000

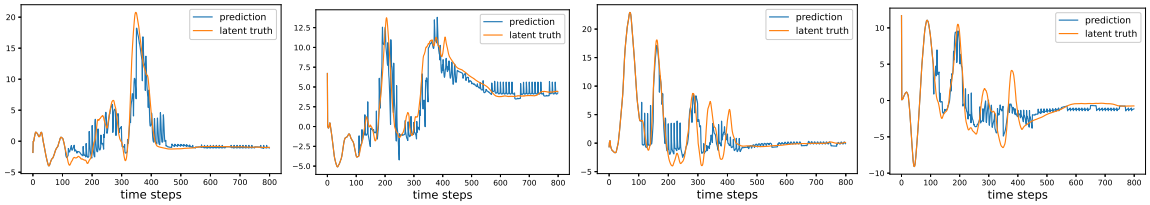
.



(a) LSTM100 (LV 1) (b) LSTM100 (LV 2) (c) LSTM100 (LV 3) (d) Model (LV 4)

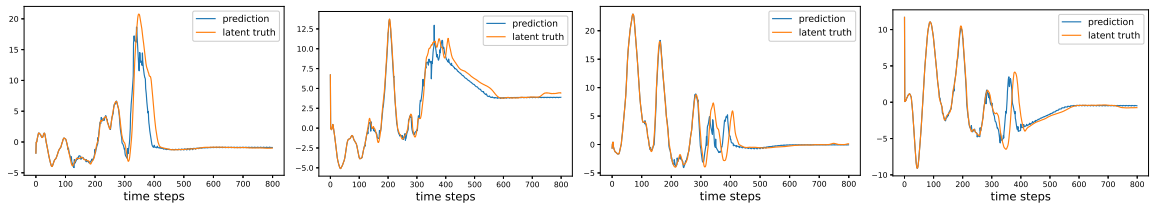


(e) Square (LV 1) (f) Square (LV 2) (g) Square (LV 3) (h) Square (LV 4)

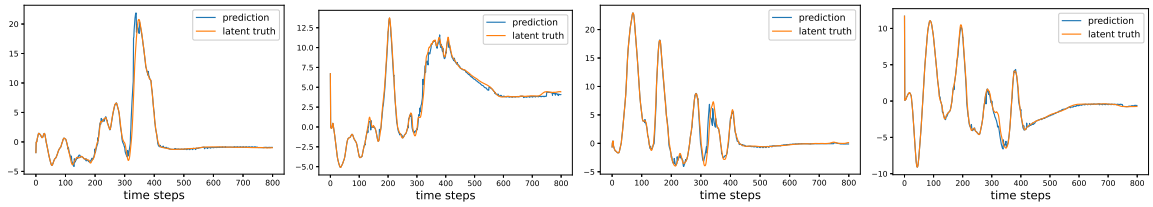


(i) Recip (LV 1) (j) Recip (LV 2) (k) Recip (LV 3) (l) Recip (LV 4)

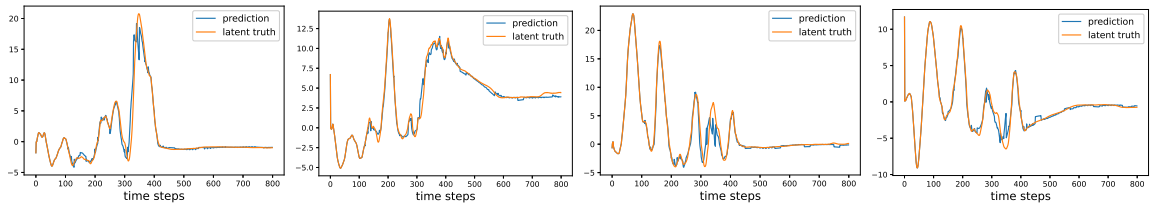
Figure 11: Effects of MEDLA on latent variables (LV) of a '*noisy*' predictive model LSTM100



(a) LSTM1000 (LV 1) (b) LSTM1000 (LV 2) (c) LSTM1000 (LV 3) (d) LSTM1000 (LV 4)

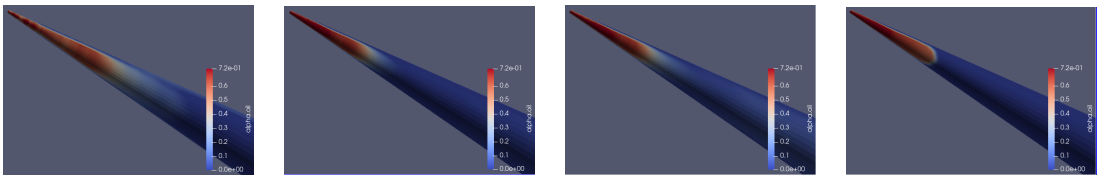


(e) Square (LV 1) (f) Square (LV 2) (g) Square (LV 3) (h) Square (LV 4)

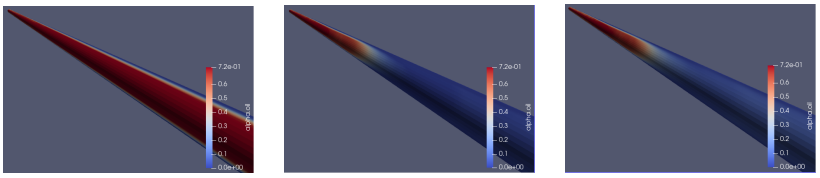


(i) Recip (LV 1) (j) Recip (LV 2) (k) Recip (LV 3) (l) Recip (LV 4)

Figure 12: Effects of MEDLA on latent variables (LV) of an '*accurate*' predictive model LSTM1000



(a) LSTM 1000 (b) Square 1000 (c) Recip 1000 (d) CFD



(e) LSTM 100 (f) Square 100 (g) Recip 100

Figure 13: Results of different latent data assimilation approaches with correlated observation errors

4.3. Microfluidic drop interactions

4.3.1. Simulation and observation system setup

In order to assess MEDLA's performance in handling observations without explicit state-observation mapping function, we conduct numerical experiments within a case study involving microfluidic drop interactions. The state variables are derived from experimental recordings, while real-time corrections are made using simulated CFD data in this study. We note that this example problem involves so-called interfacial 'singularities', which are associated with an interfacial value going to zero; this occurs in this case when the thin film between the two interacting drops drains completely, allowing

Table 4: Averaged relative prediction error and computational time (inference time/step) for GLA and MEDLA in the test dataset consisting of 3 trajectories

	LSTM100		LSTM1000	
	relative error	inference time	relative error	inference time
Original	36.70%	2.34s	9.18%	2.32s
GLA square	22.52%	78.30s	8.02%	74.52s
GLA Recip	20.63%	79.48s	10.63%	76.17s
MEDLA square	15.21%	2.51s	3.20%	2.44s
MEDLA Recip	15.62%	2.40s	4.98%	2.45s

them to coalesce. Capturing these singularities numerically via CFD simulations is a well-known challenging task [44].

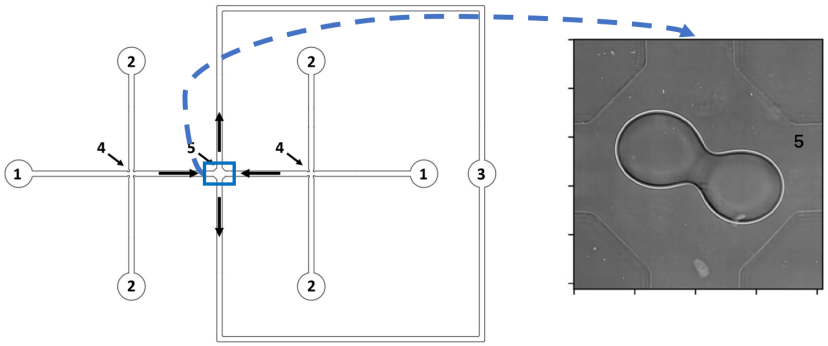


Figure 14: Scheme of the microfluidics device: 1 – inlets for dispersed phase, 2 – inlets for continuous phase, 3 – outlet, 4 – X-junctions for drop formation, 5 – coalescence chamber

.

Recording video data as states. This study uses the same observation data in [61] to examine droplet dynamics in a microfluidic device made of polydimethylsiloxane (see Figure 14). The device has distinct channels and a coalescence chamber for droplet interaction. A high-speed video camera attached to an inverted microscope records the droplet behavior. The processed dataset includes 47 trajectories, each with 195 frames captured from video recordings and all the frames are processed into grey-scale images with each pixel varying from 0 to 1. Our study focuses exclusively on the frames preceding the coalescence or non-coalescence events, thereby eliminating any distinction between the two outcomes. This strategy enables us to examine the fundamental droplet dynamics independently of the final coalescence event. The video recordings document droplet behavior at two superficial velocities: 2.09 mm/s (26 trajectories) and 1.57 mm/s (21 trajectories). In total, 7 trajectories, 3 with velocity 2.09 mm/s and 4 with velocity 1.57 mm/s have corresponding CFD simulations. In this study, the trajectories with 1.57 mm/s (resp. 2.09 mm/s) are used as training (resp. test) datasets for the surrogate model to evaluate the robustness of MEDLA with unseen initial conditions.

CFD simulations as observations. The CFD framework used here to mimic the experimental observations of the coalescing chamber is based on the solution of the fully

three-dimensional Navier-Stokes equation given by

$$\begin{aligned} \nabla \cdot \mathbf{u} &= 0 \\ \rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}\right) &= -\nabla P + \nabla \cdot [\mu\left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right)] + \mathbf{F} \end{aligned} \tag{31}$$

where $\mathbf{u}$ is the velocity, P the pressure, $\mathbf{F}$ is the local surface tension force at the interface, ρ the density, and μ the dynamical viscosity; the fluids are assumed to be incompressible, and the surface tension constant. This equation is solved on a structured Eulerian-grid structure, and the interface uses a Lagrangian non-structured adaptive mesh via a hybrid Front-tracking/level-set technique [45, 46, 47]. First, the construction of the chamber and the four branches is done via a module that we use for the creation of solid objects. Using a similar approach as that in [29] which circumvents the need for time-consuming construction, meshing, and remeshing. Thus, a static distance function, which is positive for the fluid part and negative for the solid part, is set for this purpose. Thus, the shape of the coalescence chamber and its 4 branches is the zero iso-value of that distance function (see the geometry of the chamber in Figure 15).

A crucial initialisation is the exact location of the two drops inside the coalescing chamber: 7 cases were chosen experimentally at a precise time and this has been used to mimic the experiments.

The pressure fields obtained from the CFD simulations are used to assimilate the predictions of the model trained using the recorded videos. The initial pressure field highlighted in Figure 15 (a) is a result of the projection method used to solve the entire Navier-Stokes method. Contrary to the velocity field which is zero everywhere in the domain, except at the boundary condition, the pressure field adjusts itself (higher at the inlets and lower and the outlet) in such a way to satisfy the free divergence condition $\nabla.\mathbf{u} = 0$.

A comparison between the interface and the pressure field obtained from the experimental image is shown in Figure 16. It is important to note that deriving an explicit numerical function from recorded experimental videos to map them to the CFD pressure field is exceptionally challenging. Consequently, current DA or LA methods face significant difficulties when attempting to employ multi-domain data for correcting predictive models of drop dynamics.

4.3.2. Surrogate model and data assimilation setup

The resolutions of the experimental images and pressure field are both set to 256×256 (see Figure 16). Using the method delineated in Section 3, two distinct training phases are implemented. Initially, both the state and observation encoders are trained for 1,000 epochs exclusively on the 4 trajectories that possess corresponding CFD simulations. Subsequently, the state encoder is fine-tuned for another 1,000 epochs, but this time on the 40 trajectories without CFD simulations. To align the two encoders, we fine-tune the state encoder (see Algorithm 1), while keeping the weights of the observation encoder constant. Given that the recorded frames represent the drop edges in a binary form for each pixel after pre-processing (see Figure 16), we employ Binary Crossentropy (BCE) loss for training both CAE and LSTM models.

After training the predictive model, MEDLA is tested on the 3 video datasets with a superficial velocity 2.09 mm/s. DA is performed at every step to correct the prediction as the LSTM is trained with a different flow rate. The new method MEDLA allows us to integrate the high-fidelity CFD simulations with the LSTM model predictions, thereby

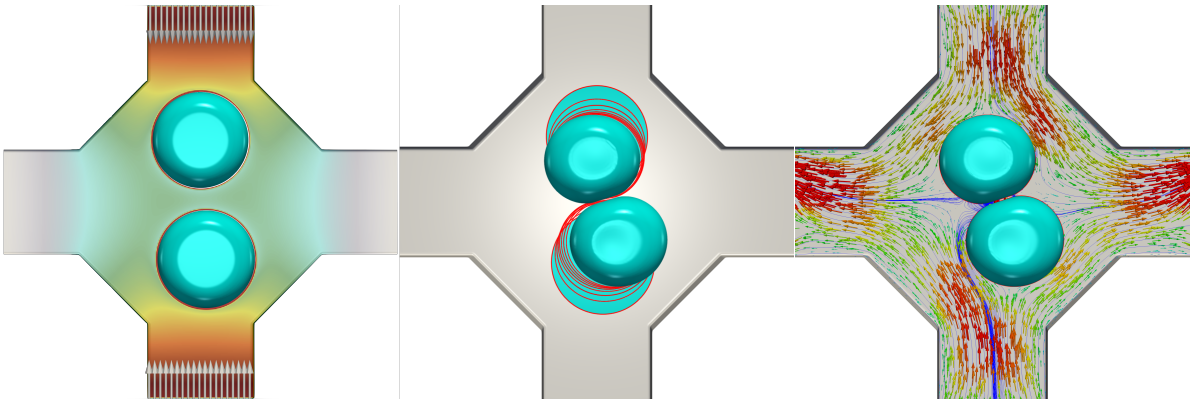


Figure 15: (a) Numerical initialisation of two drops inside the coalescing chamber. (b) Numerical snapshots of the interface from its initial state $t = 0$ s until 0.2 s. (c) Final state highlighting the velocity field and the streamlines.

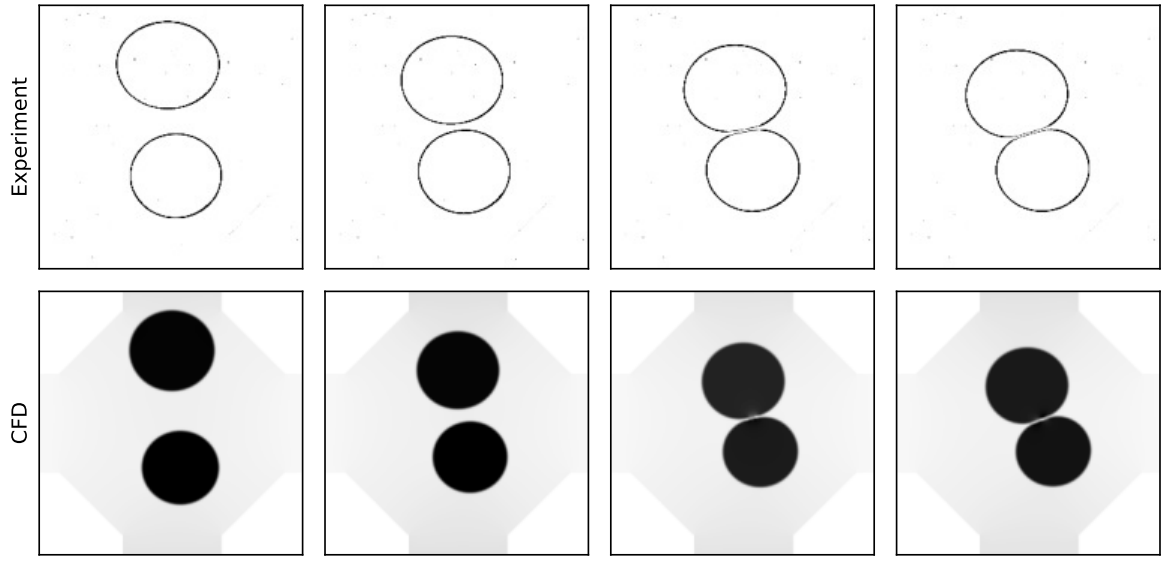


Figure 16: Left to right: From the frame where two drops enter the coalescence chamber to the frame where drops coalesce or drift apart. The experimental image is the extracted interface from video recordings and the CFD is the snapshot of the pressure field.

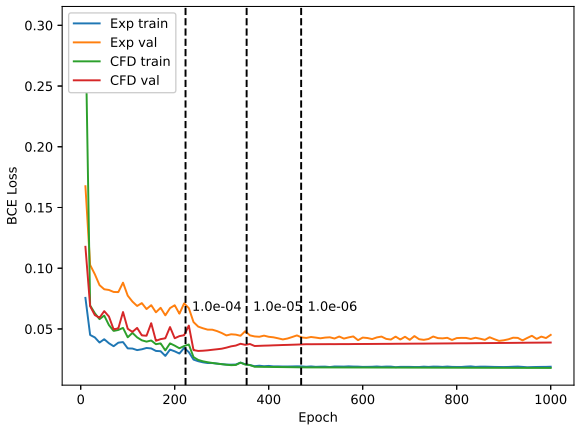


Figure 17: Training loss (BCE) for the multi-domain encoder-decoder from the 1st epoch, vertical dashed lines mark the step of learning rate decay.

.

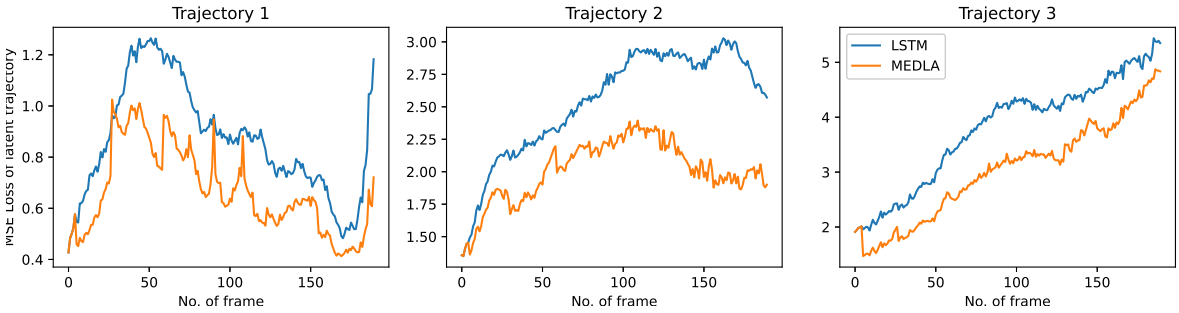


Figure 18: MSEs of three trajectories compared to the encoded experimental trajectories on the three test dataset

enhancing the model's robustness and accuracy. This is difficult to achieve using the current DA and LA approaches due to the absence of an explicit mapping function.

4.3.3. Results

We present the BCE loss during the training of multi-domain encoders in Figure 17. Consistent with the findings from the preceding two test cases (see Section 4.1 and 4.2), the losses for both the training and test datasets consistently decrease and stabilize after approximately 300 epochs. This indicates the successful encoding of multi-domain data into a shared latent space. However, a gap between training and validation loss can be observed for both CFD and experimental data due to the different flow rates in the training and the test dataset.

In this case study, we employ the MSE as our evaluation metric instead of a relative error. This choice is made due to the presence of numerous empty pixels in the background images, as shown in Figure 16. Figure 18 depicts the evolution of MSE with and without MEDLA in the 3 trajectories in the test dataset. It is important to highlight that the prediction errors encompass both compression errors as shown in Figure 17, and prediction errors from the application of LSTM. Our goal in this study is to diminish the latter type of errors by integrating real-time CFD data through MEDLA. It can be clearly seen in Figure 18 that the assimilation helps with long-term stability with a significant reduction of the prediction error. This finding is consistent with the averaged MSE shown in Table 5.

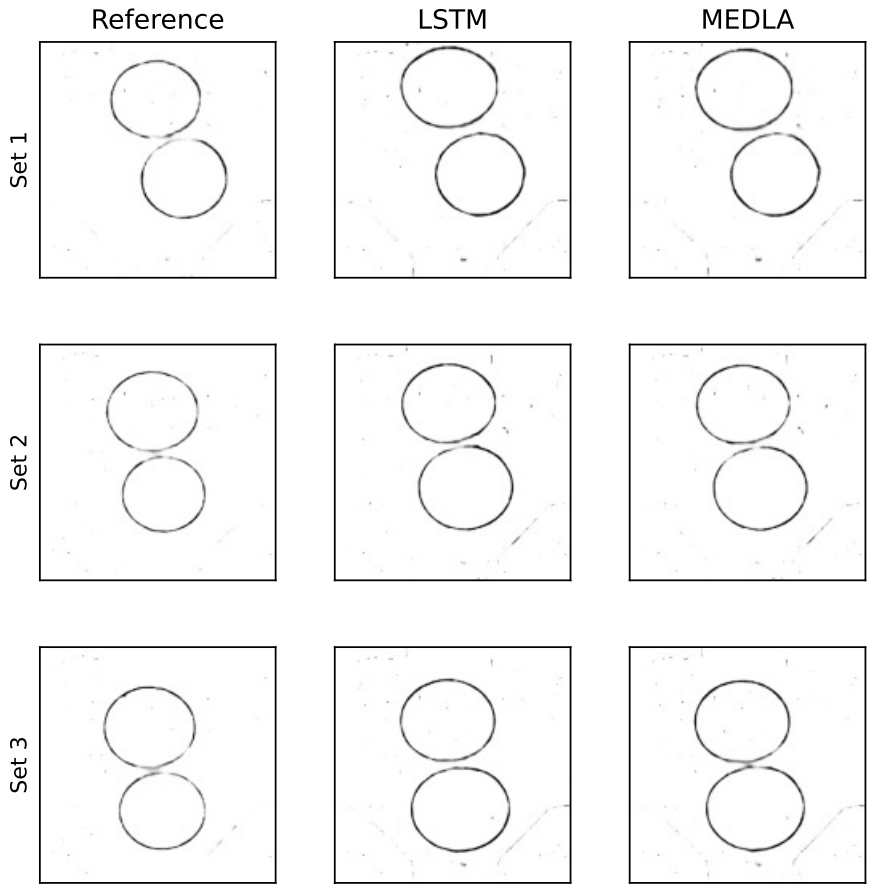


Figure 19: Comparison of the decoded images of the three trajectories at the last frame

Table 5: MSE and online inference time (including decoding) with and without MEDLA

	Error			Time
	Trajectory 1	Trajectory 2	Trajectory 3	Inference time per step
LSTM-MSE	0.867	2.540	3.781	$0.23s$
MEDLA-MSE	0.666	2.009	3.019	$0.31s$

A comparison of the decoded images of the three trajectories at the last frame is shown in Figure 19. While the LSTM predictions already exhibit a high degree of proximity to the reference frames, it becomes evident that the prediction results move even closer to the reference frames following assimilation in the test dataset. In fact, the original and predicted images of drops are sparse, which can lead to a double penalty bias when evaluating using mean square error, as discussed in [4]. Addressing this issue remains an open and challenging question within the data assimilation community. Nevertheless, within MEDLA, the assimilation process can make use of observation data even lacking an explicit mapping function—a challenge that typically cannot be handled by conventional DA algorithms.

In terms of computational cost, the averaged online inference time of 100 repetitions is also shown in Table 5. Thanks to its unique structure of encoder-decoder, MEDLA manages to efficiently integrate multi-domain data in the assimilation mechanism.

5. Conclusions and future work

Current deep-learning assisted DA algorithms encounter significant challenges when dealing with multi-domain observation data and complex or non-explicit mapping functions. In response to these challenges, this paper proposes a novel deep-learning-based DA scheme called MEDLA, which uses an encoder-decoder that can perform both encoding and decoding with multi-domain data. The new approach benefits from the efficiency of deep neural networks while aiming to improve DA accuracy by reducing interpolation/approximation errors. Comprehensive numerical experiments are conducted to evaluate the efficacy of the MEDLA scheme, comparing it to state-of-the-art LA methods. The results consistently highlight MEDLA's superiority in handling multi-scale observation data and addressing complex, non-explicit mapping functions. Figure 20 presents a qualitative assessment, emphasizing MEDLA's advantages over existing methods, especially when confronted with dense observation data (multi-scale or multi-domain) and situations where explicit mapping functions are elusive. As discussed in Section 3 and summarized in Figure 20, sparse observations may introduce difficulties when training the multi-domain encoder-decoder due to the non-existence of a state-observation bijective function. Our future works aim to broaden MEDLA's applicability, particularly to scenarios with sparse observation data and diverse sensor configurations. This extension may involve integrating MEDLA with techniques such as Voronoi-tessellated Convolutional Neural Network (CNN) [19] or masked autoencoders [25]. As discussed in Section 4.3.3, efforts will be directed towards optimally designing the training loss function and evaluation metrics to account for extreme events. Future work will also explore the robustness of MEDLA in handling observation errors with non-explicit and highly nonlinear observation operators.

It is also important to note that some recent work [36] used probabilistic modelling, such as variational autoencoders, to perform latent data assimilation. This approach

may lead to a more controlled and smooth posterior distribution. However, the inference of probabilistic models typically requires ensemble modelling, which reduces algorithm efficiency. We plan to further explore the advantages and disadvantages of using probabilistic models in latent data assimilation algorithms quantitatively in our future work. In summary, this research holds significant applicability across various dynamical sys-

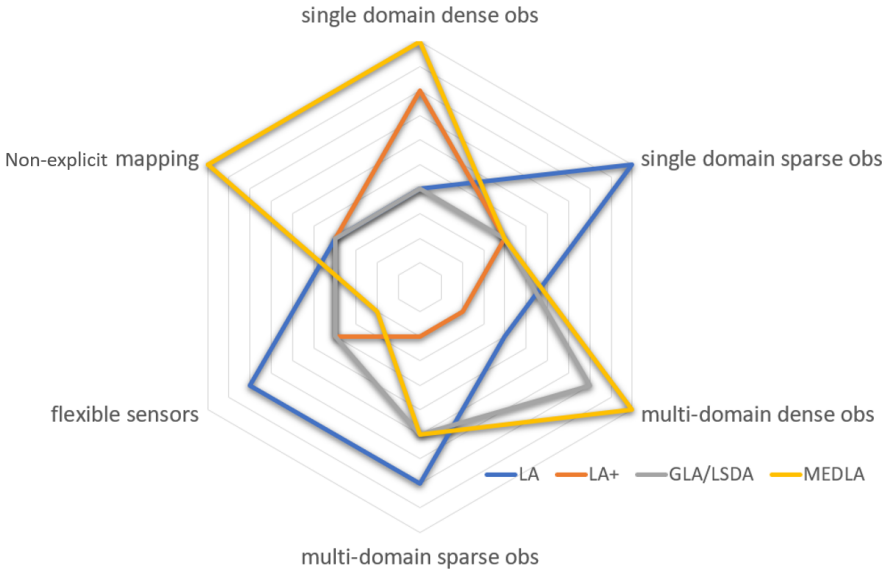


Figure 20: Qualitative comparison of different latent data assimilation algorithm

tems, notably in domains such as climate forecasting, natural hazard prediction, and nuclear engineering. In these fields, timely and precise predictions are paramount, and the observational data frequently exhibits a multi-domain nature. Additionally, obtaining an explicit state-observation function can be challenging in some scenarios. Moreover, we envision that the concept of a multi-domain encoder-decoder can be expanded to other numerical techniques, such as physics-informed machine learning, facilitating the incorporation of multi-domain physical constraints.

Acknowledgement

The authors would like to thank Dr. Nina Kovalchuck and Prof. Mark Simmons from University of Birmingham for providing the experiment data for microfluidic drop interactions. This work is supported by the EP/T000414/1 PREdictive Modelling with Quantification of UncERtainty for MultiphasE Systems (PREMIERE). This work is also partially supported by the Leverhulme Centre for Wildfires, Environment and Society through the Leverhulme Trust, grant number RC-2018-023.

Data and code availability

The code of this study is available at
`https://github.com/DL-WG/MEDLA-Multi-domain-encoder-decoder-neural-networks-for-late`

Appendix

Computational resources for different test cases

Test case 1: The CFD simulations and the neural network training were performed using the Google Colab platform with an Intel Xeon CPU @2.30 GHz and a Tesla T4

GPU.

Test case 2: The CFD simulations were performed on the high performance computing system (Intel Xeon(R) CPU E5-2620, 2.00 GHz, RAM 64 GB). The neural networks were trained on a workstation equipped with an Intel i9-12900K processor and an RTX A6000 GPU.

Test case 3: For the CFD simulation, $11 \times 11 \times 1$ processor cores were used to carry the simulations of the coalescing chamber with code parallelization based on an algebraic domain decomposition technique. The CFD code is written in the computing language Fortran 2008 and communications are managed by data exchange across adjacent subdomains via the Message Passing Interface (MPI) protocol. The neural network for microfluidic droplet interactions runs on a laptop equipped with an Intel i7-11800H processor and an RTX 3080 GPU.

Structure of the neural networks in different test cases

The exact structure of the neural networks used in the three test cases in this paper are shown in Tables 6, 7, 8 respectively. It is worth mentioning that in test case 2 (multiphase flow) we first compress the data using the first 1000 principle components of POD then simple fully connected neural networks are employed for the autoencoder.

To ensure effective learning and prevent stagnation, learning rate scheduler is used during the training of multi-domain encoder-decoder. For test case 1 and test case 2, a customized scheduler is designed where, after the first 100 epochs, the learning rate decreases from 1×10^{-3} to 1×10^{-5} for the alternating training. We kept the learning rate at 1×10^{-5} during the alignment fine-tuning. In test case 3, the ReduceLROnPlateau scheduler is used, with a patience of 50 steps and minimum learning rates of 1×10^{-6} for the alignment phase and 1×10^{-8} for the fine-tuning phase.

Table 6: MEDLA Neural Network Architecture for the Burger's equation (Test case 1)

Component	Layer	Output Shape	Activation
State Encoder			
Input	-	(1, 128, 128)	-
Conv2D	4 channels, 8x8 kernel	(4, 128, 128)	ReLU
MaxPooling	4x4 pool	(4, 32, 32)	-
Conv2D	8 channels, 4x4 kernel	(8, 32, 32)	ReLU
MaxPooling	2x2 pool	(8, 16, 16)	-
Conv2D	8 channels, 4x4 kernel	(8, 16, 16)	ReLU
MaxPooling	2x2 pool	(8, 8, 8)	-
Flatten	-	512	-
Dense	-	15	LeakyReLU
Observation Encoder			
Input	-	(1, 32, 32)	-
Conv2D	4 channels, 4x4 kernel	(4, 32, 32)	ReLU
MaxPooling	2x1 pool	(4, 16, 16)	-
Conv2D	4 channels, 4x4 kernel	(4, 16, 16)	ReLU
MaxPooling	2x2 pool	(4, 8, 8)	-
Flatten	-	256	-
Dense	-	15	LeakyReLU
Decoder (Shared Decoder)			
Input	-	15	-
Dense	-	512	-
Reshape	-	(8, 8, 8)	-
Conv2D	8 channels, 4x4 kernel	(8, 8, 8)	ReLU
Upsampling	2x scale	(8, 16, 16)	-
Conv2D	8 channels, 4x4 kernel	(8, 16, 16)	ReLU
Upsampling	2x scale	(8, 32, 32)	-
Conv2D	4 channels, 4x4 kernel	(4, 32, 32)	ReLU
Upsampling	4x scale	(4, 128, 128)	-
Conv2D	1 channel, 4x4 kernel	(1, 128, 128)	Sigmoid

Table 7: MEDLA Neural Network Architecture for the mutliphase flow (Test case 2)

Component	Layer	Output Shape	Activation
State Encoder			
Input	-	1000	-
Dense	-	128	LeakyReLU
Dense	-	30	LeakyReLU
Observation Encoder			
Input	-	1000	-
Dense	-	128	LeakyReLU
Dense	-	30	LeakyReLU
Decoder (Shared Decoder)			
Input	-	30	-
Dense	-	128	LeakyReLU
Dense	-	1000	LeakyReLU

Table 8: MEDLA Neural Network Architecture for Drop Interactions (Test case 3)

Component	Layer	Output Shape	Activation
State Encoder			
Input	-	(1, 256, 256)	-
Conv2D	8 channels, 16x16 kernel	(8, 256, 256)	ReLU
MaxPooling	4x4 pool	(8, 64, 64)	-
Conv2D	16 channels, 8x8 kernel	(16, 64, 64)	ReLU
MaxPooling	4x4 pool	(16, 16, 16)	-
Conv2D	32 channels, 4x4 kernel	(32, 16, 16)	ReLU
MaxPooling	4x4 pool	(32, 4, 4)	-
Flatten	-	512	-
Dense	-	16	-
Observation Encoder			
Input	-	(1, 256, 256)	-
Conv2D	8 channels, 16x16 kernel	(8, 256, 256)	ReLU
MaxPooling	4x4 pool	(8, 64, 64)	-
Conv2D	16 channels, 8x8 kernel	(16, 64, 64)	ReLU
MaxPooling	4x4 pool	(16, 16, 16)	-
Conv2D	32 channels, 4x4 kernel	(32, 16, 16)	ReLU
MaxPooling	4x4 pool	(32, 4, 4)	-
Flatten	-	512	-
Dense	-	16	-
Decoder (Shared Decoder)			
Input	-	16	-
Dense	-	512	-
Reshape	-	(32, 4, 4)	-
Conv2D	32 channels, 4x4 kernel	(32, 4, 4)	ReLU
Upsampling	4x scale	(32, 16, 16)	-
Conv2D	16 channels, 8x8 kernel	(16, 16, 16)	ReLU
Upsampling	4x scale	(16, 64, 64)	-
Conv2D	8 channels, 16x16 kernel	(8, 64, 64)	ReLU
Upsampling	4x scale	(8, 256, 256)	-
Conv2D	1 channel, 4x4 kernel	(1, 256, 256)	Sigmoid

Acronyms

NN	Neural Network
ML	Machine Learning
LA	Latent data Assimilation
DA	Data Assimilation
AE	Autoencoder
CAE	Convolutional Autoencoder
BLUE	Best Linear Unbiased Estimator
BCE	Binary Cross Entropy
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
LSTM	long short-term memory
POD	Proper Orthogonal Decomposition
PC	principal component
SVD	Singular Value Decomposition
ROM	reduced-order modelling
CFD	computational fluid dynamics
MSE	mean square error
BCE	Binary Crossentropy
MAE	mean absolute error
DL	Deep Learning
RF	Random Forest
GLA	Generalised Latent Assimilation
GAN	Generative Adversarial Network
MLP	Multi layer percepton
DDA	Deep Data Assimilation
LSDA	Latent Space Data Assimilation
MEDLA	Multi-domain Encoder-Decoder Latent data Assimilation

Bibliography

[1] M. Amendola, R. Arcucci, L. Mottet, C. Q. Casas, S. Fan, C. Pain, P. Linden, and Y.-K. Guo. Data assimilation in the latent space of a convolutional autoencoder. In *International Conference on Computational Science*, pages 373–386. Springer, 2021.

[2] J. Bao, L. Li, and A. Davis. Variational autoencoder or generative adversarial networks? a comparison of two deep learning methods for flow and transport data assimilation. *Mathematical Geosciences*, pages 1–26, 2022.

[3] H. Bateman. Some recent researches on the motion of fluids. *Monthly Weather Review*, 43(4):163–170, 1915.

[4] M. Bocquet, P. J. Vanderbecken, A. Farchi, J. Dumont Le Brazidec, and Y. Roustan. Bridging classical data assimilation and optimal transport. *EGUsphere*, 2023:1–36, 2023.

[5] J. Brajard, A. Carrassi, M. Bocquet, and L. Bertino. Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: a case study with the Lorenz 96 model. *Geoscientific Model Development Discussions*, 2019:1–21, 2019.

[6] J. M. Burgers. A mathematical model illustrating the theory of turbulence. *Advances in applied mechanics*, 1:171–199, 1948.

[7] A. Carrassi, M. Bocquet, L. Bertino, and G. Evensen. Data assimilation in the geosciences: An overview of methods, issues, and perspectives. *Wiley Interdisciplinary Reviews: Climate Change*, 9(5):e535, 2018.

[8] C. Q. Casas, R. Arcucci, P. Wu, C. Pain, and Y.-K. Guo. A reduced order deep data assimilation model. *Physica D: Nonlinear Phenomena*, 412:132615, 2020.

[9] J. Chen, C. Anastasiou, S. Cheng, N. M. Basha, L. Kahouadji, R. Arcucci, P. Angeli, and O. K. Matar. Computational fluid dynamics simulations of phase separation in dispersed oil-water pipe flows. *Chemical Engineering Science*, 267:118310, 2023.

[10] S. Cheng, J.-P. Argaud, B. Iooss, D. Lucor, and A. Ponçot. Background error covariance iterative updating with invariant observation measures for data assimilation. *Stochastic Environmental Research and Risk Assessment*, 33(11):2033–2051, 2019.

[11] S. Cheng, J. Chen, C. Anastasiou, P. Angeli, O. K. Matar, Y.-K. Guo, C. C. Pain, and R. Arcucci. Generalised latent assimilation in heterogeneous reduced spaces with machine learning surrogate models. *Journal of Scientific Computing*, 94(1):11, 2023.

[12] S. Cheng, I. C. Prentice, Y. Huang, Y. Jin, Y.-K. Guo, and R. Arcucci. Data-driven surrogate model with latent data assimilation: Application to wildfire forecasting. *Journal of Computational Physics*, page 111302, 2022.

[13] S. Cheng, C. Quilodrán-Casas, S. Ouala, A. Farchi, C. Liu, P. Tandeo, R. Fablet, D. Lucor, B. Iooss, J. Brajard, et al. Machine learning with data assimilation and uncertainty quantification for dynamical systems: a review. *IEEE/CAA Journal of Automatica Sinica*, 10(6):1361–1387, 2023.

[14] G. Evensen. Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *Journal of Geophysical Research: Oceans*, 99(C5):10143–10162, 1994.

[15] R. Fablet, L. Drumetz, and F. Rousseau. Joint learning of variational representations and solvers for inverse problems with partially-observed data. *arXiv preprint arXiv:2006.03653*, 2020.

[16] A. Farchi, P. Laloyaux, M. Bonavita, and M. Bocquet. Using machine learning to correct model error in data assimilation and forecast applications. *Quarterly Journal of the Royal Meteorological Society*, 147(739):3067–3084, 2021.

[17] T. S. Finn, L. Disson, A. Farchi, M. Bocquet, and C. Durand. Representation learning with unconditional denoising diffusion models for dynamical systems. *EGUsphere*, 2023:1–39, 2023.

[18] S. Fresca and A. Manzoni. Pod-dl-rom: Enhancing deep learning-based reduced order models for nonlinear parametrized pdes by proper orthogonal decomposition. *Computer Methods in Applied Mechanics and Engineering*, 388:114181, 2022.

[19] K. Fukami, R. Maulik, N. Ramachandra, K. Fukagata, and K. Taira. Global field reconstruction from sparse sensors with voronoi tessellation-assisted deep learning. *Nature Machine Intelligence*, 3(11):945–951, 2021.

[20] F. Gatti and D. Clouteau. Towards blending physics-based numerical simulations and seismic databases using generative adversarial network. *Computer Methods in Applied Mechanics and Engineering*, 372:113421, 2020.

[21] N. Geneva and N. Zabaras. Transformers for modeling physical systems. *Neural Networks*, 146:272–289, 2022.

[22] H. Gong, Z. Chen, Y. Maday, and Q. Li. Optimal and fast field reconstruction with reduced basis and limited observations: Application to reactor core online monitoring. *Nuclear Engineering and Design*, 377:111113, 2021.

[23] H. Gong, S. Cheng, Z. Chen, and Q. Li. Data-enabled physics-informed machine learning for reduced-order modeling digital twin: Application to nuclear reactor physics. *Nuclear Science and Engineering*, pages 1–26, 2022.

[24] H. Gong, Y. Yu, X. Peng, and Q. Li. A data-driven strategy for xenon dynamical forecasting using dynamic mode decomposition. *Annals of Nuclear Energy*, 149:107826, 2020.

[25] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[26] C. E. Heaney, Y. Li, O. K. Matar, and C. C. Pain. Applying convolutional neural networks to data on unstructured meshes with space-filling curves. *arXiv preprint arXiv:2011.14820*, 2020.

[27] S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[28] S. C. James, Y. Zhang, and F. O'Donncha. A machine learning framework to forecast wave conditions. *Coastal Engineering*, 137:1–10, 2018.

[29] L. Kahouadji, E. Nowak, N. Kovalchuk, J. Chergui, D. Juric, S. Shin, M. J. Simmons, R. V. Craster, and O. K. Matar. Simulation of immiscible liquid–liquid flows in complex microchannel geometries using a front-tracking scheme. *Microfluidics and nanofluidics*, 22:1–12, 2018.

[30] D. Kumar and S. Srinivasan. Indicator-based data assimilation with multiple-point statistics for updating an ensemble of models with non-gaussian parameter distributions. *Advances in Water Resources*, 141:103611, 2020.

[31] M. Landajuela. Burgers equation. *BCAM Internship-summer*, 2011.

[32] Y. Liao, R. Oertel, S. Kriebitzsch, F. Schlegel, and D. Lucas. A discrete population balance equation for binary breakage. *International Journal for Numerical Methods in Fluids*, 87(4):202–215, 2018.

[33] C. Liu, R. Fu, D. Xiao, R. Stefanescu, P. Sharma, C. Zhu, S. Sun, and C. Wang. Enkf data-driven reduced order assimilation system. *Engineering Analysis with Boundary Elements*, 139:46–55, 2022.

[34] R. Maulik, V. Rao, J. Wang, G. Mengaldo, E. Constantinescu, B. Lusch, P. Balaprakash, I. Foster, and R. Kotamarthi. Efficient high-dimensional variational data assimilation with machine-learned reduced-order models. *Geoscientific Model Development*, 15(8):3433–3445, 2022.

[35] S. McKinley and M. Levine. Cubic spline interpolation. *College of the Redwoods*, 45(1):1049–1060, 1998.

[36] B. Melinc and Ž. Zaplotnik. 3d-var data assimilation using a variational autoencoder. *Quarterly Journal of the Royal Meteorological Society*, 2023.

[37] S. Mohd Razak, A. Jahandideh, U. Djuraev, and B. Jafarpour. Deep learning for latent space data assimilation in subsurface flow systems. *SPE Journal*, pages 1–21, 2022.

[38] S. Pawar and O. San. Equation-free surrogate modeling of geophysical flows at the intersection of machine learning and data assimilation. *Journal of Advances in Modeling Earth Systems*, 14(11):e2022MS003170, 2022.

[39] M. Peyron, A. Fillion, S. Gürol, V. Marchais, S. Gratton, P. Boudier, and G. Goret. Latent space data assimilation by using deep learning. *Quarterly Journal of the Royal Meteorological Society*, 147(740):3759–3777, 2021.

[40] T. R. Phillips, C. E. Heaney, P. N. Smith, and C. C. Pain. An autoencoder-based reduced-order model for eigenvalue problems with application to neutron diffusion. *International Journal for Numerical Methods in Engineering*, 122(15):3780–3811, 2021.

[41] C. Quilodrán-Casas, R. Arcucci, C. Pain, and Y. Guo. Adversarially trained lstms on reduced order models of urban air pollution simulations. *arXiv preprint arXiv:2101.01568*, 2021.

[42] A. Ramponi and B. Plank. Neural unsupervised domain adaptation in nlp—a survey. *arXiv preprint arXiv:2006.00632*, 2020.

[43] S. Ravuri, K. Lenc, M. Willson, D. Kangin, R. Lam, P. Mirowski, M. Fitzsimons, M. Athanassiadou, S. Kashem, S. Madge, et al. Skilful precipitation nowcasting using deep generative models of radar. *Nature*, 597(7878):672–677, 2021.

[44] R. Scardovelli and S. Zaleski. Direct numerical simulation of free-surface and interfacial flow. *Annual review of fluid mechanics*, 31(1):567–603, 1999.

[45] S. Shin, J. Chergui, and D. Juric. A solver for massively parallel direct numerical simulation of three-dimensional multiphase flows. *J. Mech. Sci. Tech.*, 31:1739–1751, 2017.

[46] S. Shin, J. Chergui, D. Juric, L. Kahouadji, O. K. Matar, and R. V. Craster. A hybrid interface tracking – level set technique for multiphase flow with soluble surfactant. *J. Comp. Phys.*, 359:409–435, 2018.

[47] S. Shin and D. Juric. Modeling three-dimensional multiphase flow using a level contour reconstruction method for front tracking without connectivity. *J. of Comp. Phys.*, 180:427–470, 2002.

[48] K. Siwek and S. Osowski. Autoencoder versus pca in face recognition. In *2017 18th International Conference on Computational Problems of Electrical Engineering (CPEE)*, pages 1–4. IEEE, 2017.

[49] M. Tang, Y. Liu, and L. J. Durlofsky. A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems. *Journal of Computational Physics*, 413:109456, 2020.

[50] V. Voulgaropoulos. *Dynamics of spatially evolving dispersed flows*. PhD thesis, University College London, 2018.

[51] J. Wang, J. Chen, J. Lin, L. Sigal, and C. W. de Silva. Discriminative feature alignment: Improving transferability of unsupervised domain adaptation by gaussian-guided latent alignment. *Pattern Recognition*, 116:107943, 2021.

[52] M. Wang and W. Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018.

[53] Y. Wang, X. Shi, L. Lei, and J. C.-H. Fung. Deep learning augmented data assimilation: Reconstructing missing information with convolutional autoencoders. *Monthly Weather Review*, 2022.

[54] Z. Wang, D. Xiao, F. Fang, R. Govindan, C. C. Pain, and Y. Guo. Model identification of reduced order fluid dynamics systems using deep learning. *International Journal for Numerical Methods in Fluids*, 86(4):255–268, 2018.

[55] G. Wei, C. Lan, W. Zeng, and Z. Chen. Metaalign: Coordinating domain alignment and classification for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16643–16653, 2021.

[56] D. Xiao, J. Du, F. Fang, C. Pain, and J. Li. Parameterised non-intrusive reduced order methods for ensemble kalman filter data assimilation. *Computers & Fluids*, 177:69–77, 2018.

[57] M. Xu, S. Song, X. Sun, and W. Zhang. Ucnn: A convolutional strategy on unstructured mesh. *arXiv preprint arXiv:2101.05207*, 2021.

[58] J. Yang, J. Duan, S. Tran, Y. Xu, S. Chanda, L. Chen, B. Zeng, T. Chilimbi, and J. Huang. Vision-language pre-training with triple contrastive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15671–15680, 2022.

[59] Y. Yao, Y. Zhang, X. Li, and Y. Ye. Heterogeneous domain adaptation via soft transfer network. In *Proceedings of the 27th ACM international conference on multimedia*, pages 1578–1586, 2019.

[60] Y. Zhou, C. Wu, Z. Li, C. Cao, Y. Ye, J. Saragih, H. Li, and Y. Sheikh. Fully convolutional mesh autoencoder using efficient spatially varying kernels. *arXiv preprint arXiv:2006.04325*, 2020.

[61] Y. Zhuang, S. Cheng, N. Kovalchuk, M. Simmons, O. K. Matar, Y.-K. Guo, and R. Arcucci. Ensemble latent assimilation with deep learning surrogate model: application to drop interaction in a microfluidics device. *Lab on a Chip*, 22(17):3187–3202, 2022.