A APPENDIX

A.1 RELATED WORK

Diffusion Probabilistic Model. Diffusion Probabilistic Models (DM) Ho et al. (2020) have emerged as the leading approach in density estimation Kingma et al. (2021) and have also demonstrated superior sample quality Dhariwal & Nichol (2021). These models leverage the inherent characteristics of image-like data by employing a UNet as their underlying neural backbone Ronneberger et al. (2015); Ho et al. (2020); Dhariwal & Nichol (2021). Notably, the use of a reweighted objective Ho et al. (2020) during training typically leads to the highest synthesis quality. Another research line for image generation is GAN-based methods Creswell et al. (2018); Chen et al. (2016b). A representative study of this research line is infoGAN Chen et al. (2016b), which is a type of generative adversarial network that not only generates realistic samples but also maximizes the mutual information between a select few latent variables and the generated output. InfoGAN allows for the discovery of meaningful representations. However, these studies cannot provide explanations for the generated samples.

Denosing Diffusion Probabilistic Model (DDPM) for AI for Science. Denoising diffusion probabilistic models have demonstrated their ability to predict dynamic evolution in a wide range of domains, including fluid dynamics Cachay et al. (2023), weather forecasting Price et al. (2023), and molecular dynamics Wu et al. (2022). They have also proven effective in inverse design tasks, facilitating the optimization of airfoils Wu et al. (2024) and proteins Watson et al. (2023). Additionally, diffusion models have shown promise in tackling complex inverse problems Holzschuh et al. (2023). These examples represent just a fraction of the diverse applications where diffusion models have been successfully employed. In the field of biology, researchers have utilized denoising diffusion probabilistic models (DDPM) to model diffusion processes in biological networks, enabling the analysis of protein-protein interactions and gene regulatory networks Fu et al. (2023); Best & Hummer (2011); Gao et al. (2023); Xu et al. (2022). In physics, the DDPM has been applied to study particle diffusion in complex systems, such as the propagation of heat in materials. Furthermore, in the realm of chemistry, the DDPM has been employed to gain insights into the diffusion of molecules and reactions in chemical systems. These studies highlight the versatility and effectiveness of the DDPM in capturing and analyzing diffusion dynamics across various scientific disciplines Xu et al. (2022). Ongoing research aims to further explore the potential of DDPMs in solving complex problems in the field of AI for Science. For additional related work on diffusion models, please refer to Appendix A.1.

A.2 PRELIMINARY

Diffusion Probabilistic Models: The Denoising Diffusion Probabilistic Model (DDPM) Ho et al. (2020) comprises two fundamental processes: the forward process (or diffusion process) and the reverse process. Let's begin by describing the forward process. In a diffusion model, the forward process approximates the posterior distribution $q(x_{1:t}|x_0)$, which represents the sequence of latent variables $x_{1:t}$ given an initial value x_0. This approximation is achieved by iteratively applying a Markov chain that gradually adds Gaussian noise over time.

The forward process is represented as follows:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \mu_t(x_{t-1}), \beta_t I)$$

Here, x_t denotes the latent variable at time step t, and x_{t-1} is the variable at the previous time step. The distribution $q(x_t|x_{t-1})$ is modeled as a Gaussian distribution with mean $\mu_t(x_{t-1})$ and variance $\beta_t I$, where β_t is the variance parameter at time step t, and I represents the identity matrix. The mean $\mu_t(x_{t-1})$ can depend on the previous latent variable x_{t-1} and is typically modeled using neural networks or other parameterized functions.

By sequentially applying the distribution $q(x_t|x_{t-1})$ for each time step, starting from the initial value x_0, we obtain an approximation of the posterior distribution $q(x_{1:t}|x_0)$ that captures the temporal evolution of the latent variables via $q(x_{1:t}|x_0) := \prod_{t=1}^{T} q(x_t|x_{t-1})$.

Consider a diffusion model with T time steps. Given an observed data point x_t at the final time step, the goal is to generate a sample from the initial distribution $p(x_0)$.

The reverse process in a diffusion model can be formulated as follows: 1. Initialization: Set x_t as the observed data point. 2. Iterative Sampling: Starting from $t = T - 1$ and moving backwards until

$t = 0$, sample x_t from the distribution $p(x_t|x_{t+1})$, where $p(x_t|x_{t+1})$ represents the reverse diffusion process.

The distribution $p(x_t|x_{t+1})$ in the reverse process is typically modeled as a Gaussian distribution, similar to the forward process. However, the mean and variance parameters are adjusted to account for the reverse direction. The specific form of $p(x_t|x_{t+1})$ is defined as follows:

$$p_{t+1}(x_t; \mu_{t+1}(x_{t+1}, t+1), \Sigma_{t+1}(x_{t+1}, t+1))$$

By iteratively sampling from the reverse process, we can generate a sequence of latent variables $x_{0:t}$ that follows the reverse diffusion process. This reverse sequence represents a sample from the initial distribution $p(x_0)$. The reverse process is crucial for training the diffusion model. During training, the model learns to approximate the reverse process by minimizing the discrepancy between the generated samples and the observed data points. This training procedure ensures that the model captures the underlying data distribution and can generate realistic samples.

The optimization objective of the diffusion model is conducted via the following negative log-likelihood:

$$\begin{aligned} E[-\log p_{t+1}(x_0)] &\leq E_{p_{t+1}}[-\log(p_{t+1}(x_{0:t})/q(x_{1:t}|x_0))] \\ &= E_{p_{t+1}}[-\log p_{t+1}(x_t) - \sum_{t=1}^{T} \log(p_{t+1}(x_{t-1}|x_t)/q(x_t|x_{t-1}))] = L \end{aligned}$$

Cahn-Hilliard Function: Wetting phenomena and interfacial tension play significant roles in numerous scientific and engineering fields, ranging from fluid dynamics to materials science. In recent years, phase-field methods have emerged as powerful computational tools for studying and simulating wetting processes. These methods employ a phase-field variable, a continuous function that describes the local composition or wetting state, enabling the realistic modeling of complex interfacial dynamics. By incorporating the concept of interfacial tension, phase-field models can capture the intricate interplay between fluids and solid surfaces.One of the key equations used in phase-field modeling of wetting phenomena is the Cahn-Hilliard equation, which governs the evolution of the phase-field variable. This equation takes into account the interfacial energy associated with the fluid-solid interface and the interfacial tension between the two phases. The interfacial tension term is crucial for accurately simulating the contact angle, adhesion, and spreading behavior. The Cahn-Hilliard equation provides a mathematical framework to capture the dynamics of phase separation by considering the free energy of the system. It takes the following general form:

$$\frac{\partial \phi}{\partial t} = \nabla \cdot \left(M \nabla \left(\frac{\delta F}{\delta \phi} \right) \right) \tag{5}$$

where ϕ is the phase-field variable or order parameter representing the local composition. t is time. M is the mobility coefficient, controlling the rate of diffusion of the phase-field variable. F is the free energy functional of the system with respect to the phase-field variable ϕ. The free energy functional F typically consists of two terms: the bulk free energy term and the gradient energy term. The bulk free energy accounts for the thermodynamic properties of the system, including the interfacial energy between the two phases and the potential energy associated with phase separation. The gradient energy term penalizes sharp variations or spatial gradients in the order parameter, promoting smoother phase transitions.

Functional Formulations for Modeling Tension Phenomena via Phase-Field: To accurately represent the shape of an arbitrary object, we utilize a phase-field variable ϕ, where $\phi = 1$ denotes the interior of the cell and $\phi = 0$ denotes the exterior. The transition from $\phi = 1$ to $\phi = 0$ occurs gradually within a width defined by the parameter ϵ. The system's total energy is denoted by $H(\phi)$, and the time evolution of ϕ is determined by the following equation $\frac{\partial \phi}{\partial t} = -\frac{\delta H(\phi)}{\delta \phi}$. This equation describes the rate of change of ϕ with respect to time. The right-hand side represents the derivative of the total energy $H(\phi)$ with respect to ϕ, indicating the force or driving mechanism that governs the evolution of the phase-field variable ϕ. By minimizing the energy functional $H(\phi)$, the system tends to reach an equilibrium state that corresponds to the desired shape of the object. For the tension dataset, the surface tension is characterized by the tension per unit length multiplied by the total surface area. To ensure proper normalization, we express it as follows:

$$H_{\text{ten}}(\phi) = \gamma \int d^2 r \left(\frac{\epsilon}{2} |\nabla \phi|^2 + \frac{G(\phi)}{\epsilon} \right) \tag{6}$$

Here, γ denotes the coefficient of surface tension, while ϵ represents the characteristic width of the interface. The term $|\nabla\phi|^2$ quantifies the gradient of ϕ in space, while $G(\phi)$ is a function proportional to the perimeter of the object and is given by $G(\phi) = 18\phi^2(1-\phi)^2$. The integral in Equation 6 accounts for the total energy associated with surface tension. The time evolution under tension is described by the following equation, known as the Allen-Cahn equation, which is a reaction-diffusion equation:

$$\frac{\partial\phi}{\partial t} = -\frac{\delta H_{\text{ten}}(\phi)}{\delta t} = -\gamma\left(\epsilon\nabla^2\phi + \frac{G'(\phi)}{\epsilon}\right) \tag{7}$$

In practice, the term $G(\phi)$ yields similar results to the $|\nabla\phi|^2$ term. Therefore, when explicitly calculating the total tension, we can use the following simplified form:

$$H'_{\text{ten}}(\phi) = \gamma\int d^2r\left(\frac{2G(\phi)}{\epsilon}\right) \tag{8}$$

Equation 8 provides an alternative expression for the total tension, which considers only the $G(\phi)$ term. This formulation allows for efficient computation of the tension without explicitly calculating the gradient term.

Functional Formulations of Wets via Phase-field: The interaction of an object with a substrate involves adhesion, which pulls the object towards the substrate, and a repellent force that prevents the object from penetrating into the substrate. To express the total energy in a physically consistent manner, we define it as follows:

$$H_{\text{sub}}(\phi) = \gamma\int d^2r\left(-A\frac{2G(\phi)G(\varphi)}{\epsilon^2} + B\phi\varphi\right) \tag{9}$$

Here, γ represents the coefficient of adhesion, A is the adhesion strength (with $A > 0$), and B is the repellent strength (with $B > 0$). The field φ corresponds to the substrate. In our simulation, the substrate is considered a fixed function given by:

$$\varphi(r) = \frac{1}{2}\left\{1 + \tanh\left[3\times\left(\frac{y_0 - y}{\epsilon}\right)\right]\right\} \tag{10}$$

Here, the substrate is positioned at a specific vertical location, $y = y_0$. In our simulation, we set $y_0 = -10$. The function φ describes the spatial profile of the substrate, with a smooth transition from high to low values as y increases from the substrate position. During this period, the evolution function, which includes the tension part, is given by:

$$\frac{\partial\phi}{\partial t} = -\gamma(\epsilon\nabla^2\phi + \frac{G'(\phi)}{\epsilon}) - A\frac{G'(\phi)G(\varphi)}{\epsilon^2} + B\varphi \tag{11}$$

Equation 11 represents the time derivative of ϕ, where the first term on the right-hand side accounts for the tension contribution, the second term describes the adhesion interaction between the object and the substrate, and the third term represents the repellent force due to the substrate. This evolution equation governs the dynamics of the phase-field variable ϕ in the presence of adhesion and substrate effects.

A.3 Datasets Generation

Tension Datasets Generation: (1) The configuration of the grid and domain: The parameters m and n determine the quantity of grid points along the x and y axes, respectively. The dimensions of the domain are specified by L_x and L_y in the x and y directions. Vectors x and y denote the coordinates of the grid points along the x and y axes, correspondingly. The parameters k and ϵ regulate the bending and tension within the system. (2) ten_{vec} is a vector that iterates over tension values for the simulation. Inside the loop for each tension value: γ is the tension parameter. dt, x_i, M_v are time steps, smoothing parameter, and viscosity parameters, respectively. N_{max} is the maximum number of iterations. $record_{num}$ determines how often the simulation results are recorded. $x_{radi}, y_{radi}, r_{radi}$ are parameters defining the initial shape of the system. ϕ_0 is the initial condition of the simulation. (3) Change m, n, L_x, L_y for a finer or coarser grid or a larger/smaller domain. Modify k, ϵ, and other parameters to explore different physical scenarios. Adjust parameters inside the tension loop (dt, x_i, M_v, etc) for different simulation characteristics. Change the initial shape parameters ($x_{radi}, y_{radi}, r_{radi}$) to explore different starting configurations.

Wets Datasets Generation: (1) The vector ten_{vec} represents tension values used in the simulation, while adh_{vec} represents adhesion values. For each combination of tension and adhesion values, denoted by ten_i and adh_j iterating through the indices of ten_{vec} and adh_{vec} respectively, the following actions take place. The parameter γ is set to the current tension value, and adh is set to the current adhesion value. The variable rep is calculated as a multiple of tension. At the beginning of each iteration, the initial shape parameters $(x_{radi}, y_{radi}, r_{radi})$ and the shape initialization (ϕ_0) are redefined to ensure unique initial conditions for each combination of tension and adhesion values. The position of the substrate is represented by y_0, and the substrate's initial shape sub_0 is initialized using a hyperbolic tangent function. (2) By altering these parameters, particularly adjusting tension, adhesion, and other physical factors, you have the ability to generate datasets that depict wetting in various scenarios. Depending on your requirements, you can experiment with different levels of tension, adhesion, initial shapes, and simulation parameters to examine how the system's morphology evolves under different conditions.

Jellyfish Datasets Generation: To generate our training and testing datasets, we employ the Lily-Pad simulator Weymouth (2015). The 2D flow field has a resolution of 128×128, assuming an infinite extension. For the jellyfish, the fixed coordinates for the head are set as $(25.6, 64)$. The wings are represented as ellipses with an identical shape, with a fixed ratio of 0.15 between their shorter and longer axes. Symmetry is maintained across the central horizontal line defined by $y = 64$. To delineate the boundaries of the wings, we sample a total of $M = 20$ points along each wing. In this 2D experiment, the key control signal is the opening angle of the wings, denoted as w. This angle is defined as the deviation between the longer axis of the upper wing and the horizontal line.

Each trajectory commences with the widest possible opening angle and follows a periodic cosine curve with a period of $T' = 200$. The trajectories differ in their initial angle (w_0), angle amplitude, and phase ratio (τ). To determine the initial angle w_0, a two-step process is employed. Firstly, a random mean angle $w^{(m)}$ is sampled from the range of $[20°, 40°]$. Then, a random angle amplitude $w^{(a)}$ is sampled from the interval $[10°, \min(w^{(m)}, 60° - w^{(m)})]$. The resulting initial angle is computed as $w_0 = w^{(m)} + w^{(a)}$, constrained within the range of $[10°, 60°]$. The phase ratio τ is randomly selected from the range of $[0.2, 0.8]$. The opening angle w_t at step t adheres to a specific pattern: it decreases from $w^{(m)} + w^{(a)}$ to $w^{(m)} - w^{(a)}$ as t progresses from 0 to $\tau T'$, and then it increases from $w^{(m)} - w^{(a)}$ to $w^{(m)} + w^{(a)}$ as t advances from $\tau T'$ to T'. Beyond T', w_t exhibits periodic variations. This configuration aligns with previous studies on jellyfish's propulsive performance Kang et al. (2023). Each trajectory is simulated for 600 steps, equivalent to 3 periods. Only the segment of the trajectory from $T' = 200$ to $3T' = 600$ steps is saved, with a step size of 10. This decision is made to conserve space, as the simulation from $t = 0$ to $T' = 200$ is primarily used for initializing the flow field. Consequently, each trajectory is stored as a sequence consisting of $\tilde{T} = (600 - 200)/10 = 40$ discrete steps.

In addition to tracking the positions of the wing boundary points and the opening angles w, we incorporate an image-like representation of the wing boundaries. This representation contains spatial information that can be efficiently integrated with the PDE states (fluid field) through convolutional neural networks. The image-like boundary representation seamlessly aligns with the shape of the PDE states. At each time step, the boundaries of the two wings are combined and transformed into a tensor with dimensions [3, 64, 64]. This tensor represents the spatial information in a grid-like format. Each cell in the tensor contains three features: a binary mask indicating whether the cell is part of a wing boundary (1) or within the fluid (0), and the relative position $(\Delta x, \Delta y)$, which denotes the distance from the cell center to the nearest boundary point. For each trajectory, the following components are saved: - PDE states u: This captures the fluid field states for each time step and has a shape of $[\tilde{T}, 3, 64, 64]$. It includes the velocity components in the x and y directions as well as the pressure. The resolution is downsampled from 128×128 to 64×64. - Velocity: $[\tilde{T}, 2, 64, 64]$. - Pressure: $[\tilde{T}, 1, 64, 64]$. - Opening angles w: This stores the opening angle in radians for each step and has a shape of $[\tilde{T}]$. - Boundary points: This records the boundary points for both the upper and lower wings and has a shape of $[\tilde{T}, 2, M, 2]$. Each wing consists of $M = 20$ points, and each point is represented by its coordinates in the x and y directions. The coordinates are scaled accordingly to fit within the grid dimensions.

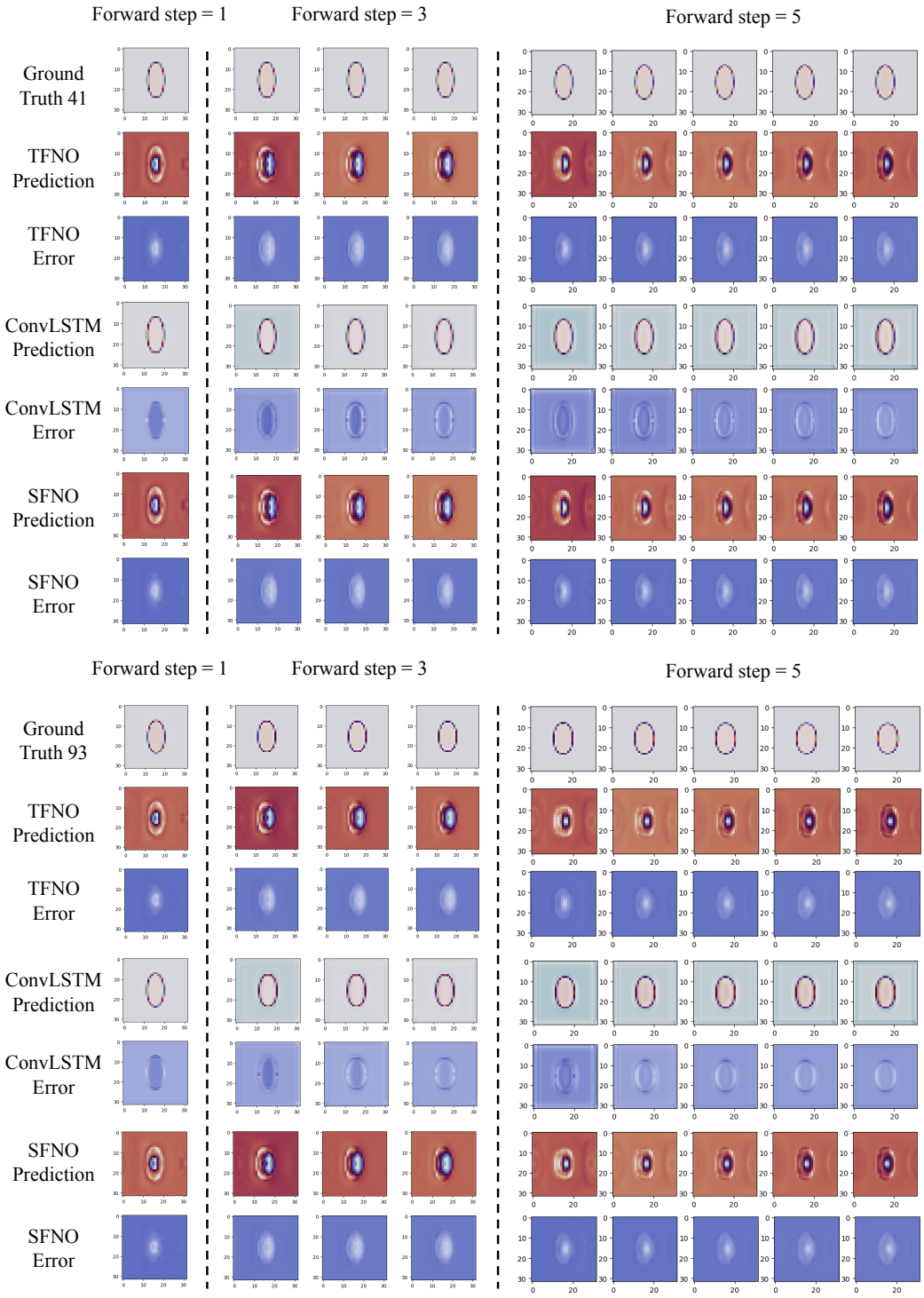
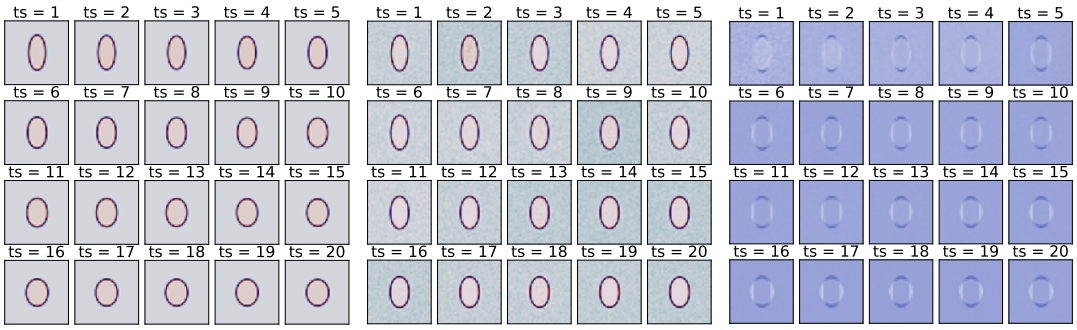
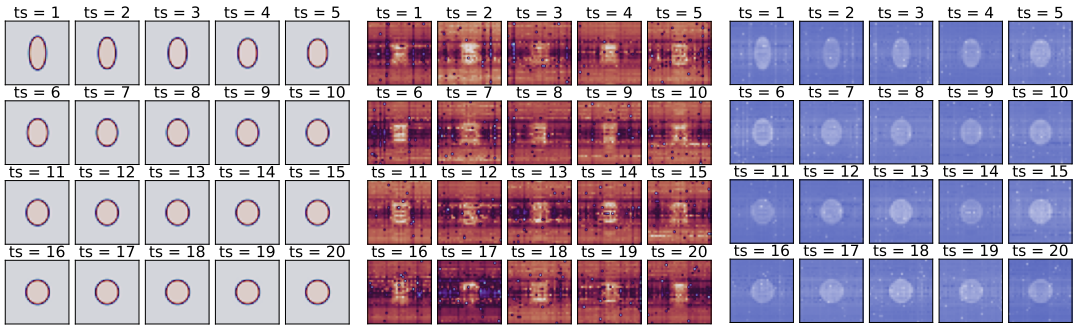


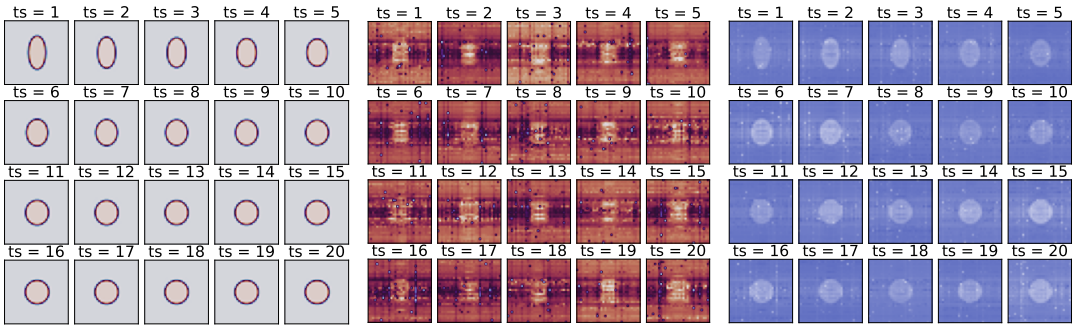
Figure 4: In our study, we offer visualizations that compare benchmark methods, namely TFNO, SFNO and ConvLSTM, using the tension dataset. The first row portrays the authentic cell scenario, which serves as the ground truth. The subsequent rows illustrate the cells generated by the benchmark methods and the corresponding error bar.



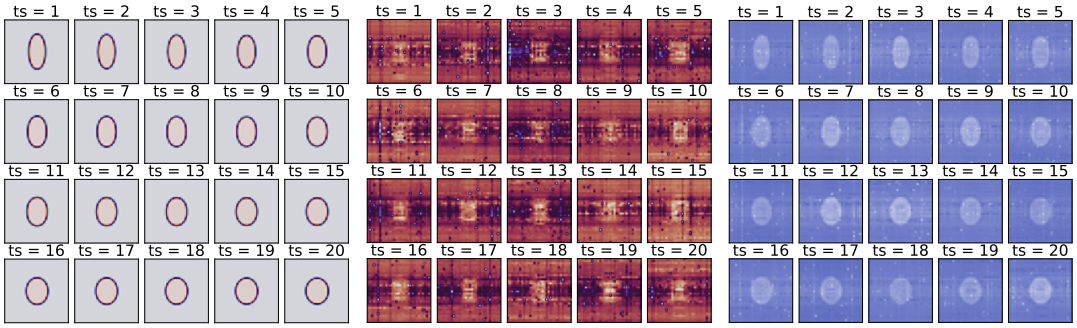
(a) Ground truth (sample 68) (b) DDPM (sample 68) (c) Error (sample 68)



(d) Ground truth (sample 74) (e) DDOs-FNO (sample 74) (f) Error (sample 74)

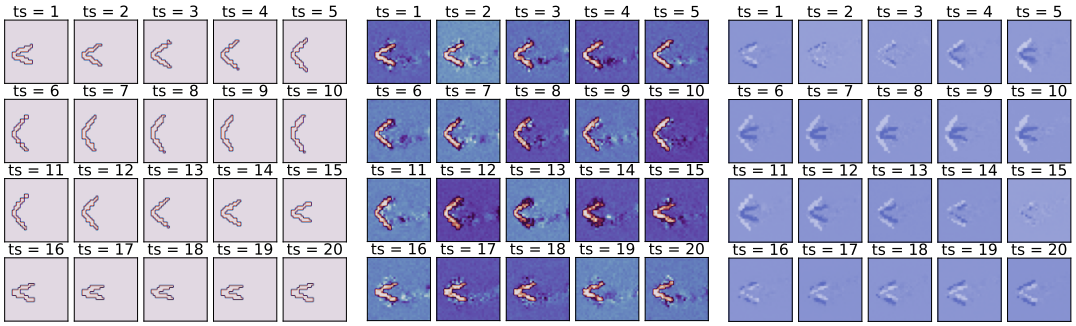


(h) Ground truth (sample 86) (i) DDOs-SFNO (sample 86) (j) Error (sample 86)

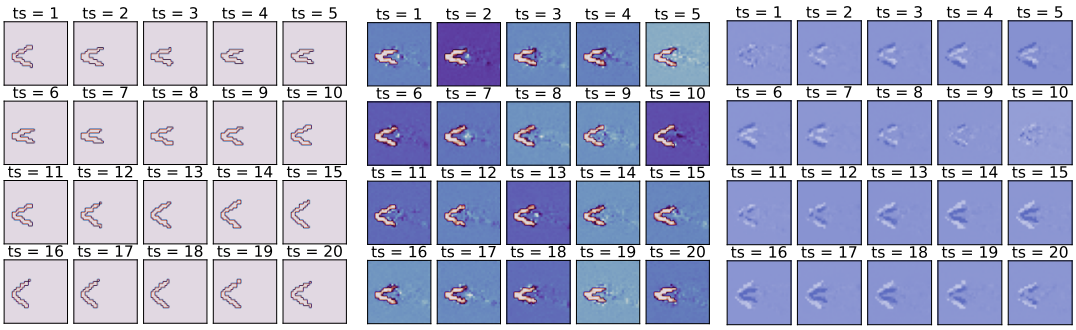


(k) Ground truth (sample 93) (l) DDOs-TFNO (sample 93) (m) Error (sample 93)

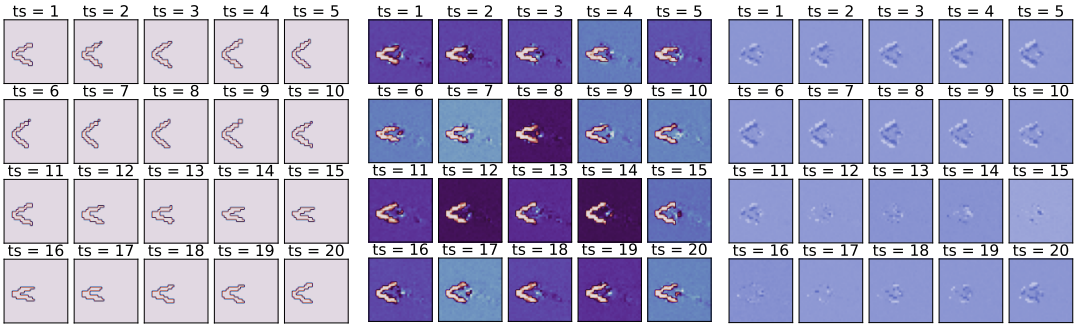
Figure 5: In our study, we present visualizations using DDPM, including the ground truth, simulation results, and error analysis. The first column subfigure represents the authentic cell scenario, which serves as the reference. The subfigures in the second column showcase the cells generated by DDPM. The subfigures in the third column illustrate the error figures, highlighting the differences between the ground truth and the simulation results obtained through DDPM.



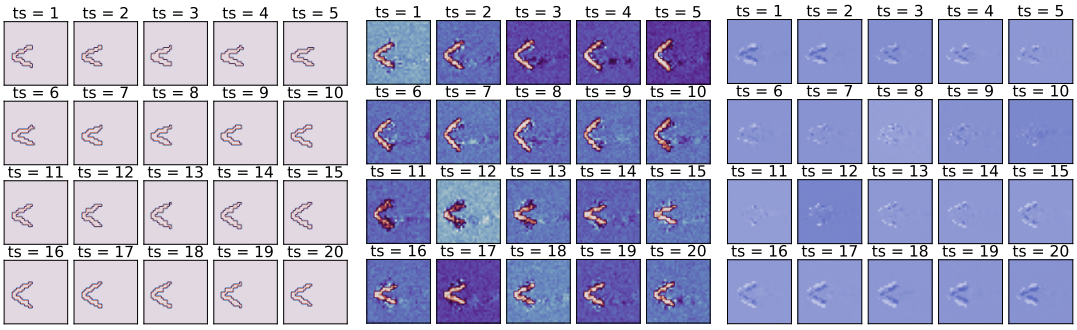
(a) Ground truth (sample 31) (b) DDPM (sample 31) (c) Error (sample 31)



(d) Ground truth (sample 61) (e) DDPM (sample 61) (f) DDPM (sample 61)



(h) Ground truth (sample 71) (i) DDPM (sample 71) (j) Error (sample 71)



(k) Ground truth (sample 81) (l) DDPM (sample 81) (m) DDPM (sample 81)

Figure 6: Our research includes visualizations utilizing DDPM to analyze the Jellyfish (Fluid) dataset, specifically focusing on the ground truth, simulation results, and error analysis. In the first column, the initial subfigure presents the accurate depiction of the boundary for a jelly-like robot. Transitioning to the second column, the subfigures demonstrate the pressure fields generated through the implementation of DDPM. Lastly, in the third column, the error figures provide a visual representation of the disparities between the ground truth and the simulated results obtained using DDPM.

A.4 Detailed Illustration and Setup of Benchmark Methods

Detailed Illustrations of Benchmark Methods: In this part, we aim to provide detaield illustrations on baselines for simulating cell evolving.

FNO Li et al. (2020): The Fourier Neural Operator is a groundbreaking approach that utilizes Fourier space to learn weights, enabling resolution-invariant global convolutions. This influential work has been further expanded upon by numerous other neural operators. However, one notable drawback is the substantial memory requirements. Specifically, each weight matrix in the Fourier domain consumes $\mathcal{O}(H^2M^D)$ memory, where H represents the hidden size, M denotes the number of Fourier modes used after truncating high frequencies, and D signifies the problem dimension.

SFNO Bonev et al. (2023): Spherical Fourier Neural Operators utilizes spherical harmonics to transform data into the frequency domain. This is analogous to the way FNO uses the Fourier transform for Cartesian grids but is specifically designed for spherical data. This approach allows for handling data on the entire sphere without the distortions introduced by map projections. While the traditional FNO faces substantial memory demands, SFNO optimizes memory usage through spherical harmonics. Despite this, the memory requirement for SFNO is still $\mathcal{O}(H^2M^2)$, where H is the hidden size and M is the number of retained spherical harmonic modes.

TFNO: Li et al. (2020) We improve the previous FNO model by simply using a Tucker Tensorized FNO with just a few parameters. This will use a Tucker factorization of the weights. The forward pass will be efficient by contracting directly the inputs with the factors of the decomposition. The Fourier layers will have 5% of the parameters of an equivalent, dense FNO.

UNet Ronneberger et al. (2015): The UNet architecture is known for its U-shaped design, which resembles an encoder-decoder structure. It is particularly effective for tasks that require precise localization and segmentation of objects within images. UNet has achieved remarkable performance in various applications, including biomedical image segmentation, satellite image analysis, and more. The Unet consists of the forward and backward passes. the time complexity is mainly driven by the size of the input image or volume, denoted as $\mathcal{O}(N)$, where N is the number of pixels or voxels. It is important to note that the specific implementation details and variations of UNet may introduce additional computational complexities.

DDPM: Ho et al. (2020) DDPM gradually transforms a noise-corrupted data version into the original distribution through iterative diffusion. Training involves two components: sampling and loss function computation. Sampling complexity is $\mathcal{O}(S * N)$, where S is the diffusion steps and N is the step size. The loss function compares generated samples with the original data, with complexity $\mathcal{O}(L * N)$, where L is the number of layers. Overall training complexity is approximately $\mathcal{O}(S * N + L * N)$. DDPM models complex data distributions using denoising diffusion. Training time complexity is typically approximated as $\mathcal{O}(S * N + L * N)$, where S is diffusion steps, L is layers, and N is layer size.

ConvLSTM Shi et al. (2015) postulates the prognostication of forthcoming spatiotemporal precipitation patterns as an endeavor entailing the anticipation of sequential data in spatial and temporal dimensions.

ViT Dosovitskiy et al. (2020) asserts that the reliance on convolutional neural networks (CNNs) is dispensable, as the direct application of transformers to sequences of image patches yields exceptional performance in the classification of visual data.

MLP-Mixer Tolstikhin et al. (2021) is exclusively constructed upon the foundation of multi-layer perceptrons (MLPs). It encompasses two distinct layer types: one that individually applies MLPs to image patches, while the other employs MLPs across multiple patches, promoting enhanced representation learning.

DDOs-FNO Lim et al. (2023) advances a robust mathematical framework meticulously tailored for the training of diffusion models within the realm of function space.

DDOs-SFNO Lim et al. (2023) derives inspiration from the fusion of DDPM and FNO methodologies, forging a cohesive amalgamation of their respective strengths.

DDOs-TFNO Lim et al. (2023) adopts a hybrid approach, drawing inspiration from both DDPM and TFNO methods, thereby capitalizing on their synergistic potential.

Setups of Benchmark Methods:

Setting of **FNO**: The modes and width of FNO was set to 12 and 32, providing a suitable level of complexity for the specific task at hand. The number of channels for the two cell dynamics datasets and the fluid dataset was determined as $[1, 3, 5, 7, 9]$, taking into consideration the unique characteristics of each dataset. All three datasets were standardized to an image size of 32×32 pixels. In order to accelerate convergence, a learning rate of 1×10^{-3} was adopted.

Setting of **SFNO**: The SFNO modes were set to 16, and the hidden channel was configured to 64. The channel numbers for both cell dynamics datasets and the fluid dataset were set to $[1, 3, 5, 7, 9]$, reflecting the unique characteristics of each dataset. All three datasets were standardized to an image size of 32×32 pixels. To accelerate convergence, we implemented a learning rate of 1×10^{-3}.

Setting of **TFNO**: The TFNO modes were set to 16, and the hidden channel was configured to 64. We employed Tucker factorization with a rank of 0.05. The channel numbers for both cell dynamics datasets and the fluid dataset were set to $[1, 3, 5, 7, 9]$, reflecting the unique characteristics of each dataset. All three datasets were standardized to an image size of 32×32 pixels. To accelerate convergence, we implemented a learning rate of 1×10^{-3}.

Setting of **UNet**: The hidden size of the UNet convolutional neural network was set to 32, providing a suitable level of complexity for the specific task at hand. The number of channels for the two cell dynamics datasets and the fluid dataset was determined as $[1, 3, 5, 7, 9]$, taking into consideration the unique characteristics of each dataset. All three datasets were standardized to an image size of 32×32 pixels. In order to accelerate convergence, a learning rate of 1×10^{-4} was adopted.

Setting of **DDPM**: To ensure equity and uniformity in our experimental procedures, the concealed dimension of the U-Net convolutional neural network was established at 32, furnishing an apt level of intricacy for the given undertaking. For the Gaussian diffusion process, an extensive 1000 diffusion steps were executed, facilitating comprehensive exchange of information. The channel size multiplier of the U-Net neural networks was stipulated as $[1, 2, 4, 8]$, ensuring efficacious extraction of features across diverse scales. The number of channels for the two cell dynamics datasets and the fluid dataset were defined as 20, accommodating the distinctive attributes of each dataset. The dimensions of all three datasets were standardized to an image size of 32×32 pixels. To expedite convergence, a learning rate of 8×10^{-5} was embraced.

Setting of **ConvLSTM**: The ConvLSTM model is initialized with input dimension and hidden dimension are both set to 1, indicating a single-channel input and a single-channel output per hidden layer. The model is designed to process sequences of length determined by $[1, 3, 5, 7, 9]$, representing the time dimension parameter. The convolution operation within the LSTM utilizes a kernel size of 3×3, which allows the model to capture spatial relationships within the data effectively.

Setting of **ViT**: ViT uses the temporal positional encoding method to handle the sequence length for positional encoding, applied across combined time steps and patches (with size equals to 4). A linear layer transforms the patch embeddings back into pixel values, ensuring the reconstruction of the original image or the generation of future frames in the biological trajectory. The channel numbers for both cell dynamics datasets and the fluid dataset were set to $[1, 3, 5, 7, 9]$, reflecting the unique characteristics of each dataset. All three datasets were standardized to an image size of 32×32 pixels. To accelerate convergence, we implemented a learning rate of 1×10^{-3}.

Setting of **MLP-Mixer**: The MLP-Mixer processes input data through two primary stages: channel-mixing and token-mixing, utilizing a patch size of 4×4, hidden dimensions of 32, and 4 layers. The number of channels for both cell dynamics datasets and the fluid dataset was set to $[1, 3, 5, 7, 9]$, tailored to the unique characteristics of each dataset. All three datasets were standardized to an image size of 32×32 pixels. To enhance convergence, a learning rate of 1×10^{-3} was adopted.

Setting of **DDOs-FNO/DDOs-SFNO/DDOs-TFNO**: The hidden size of the FNO neural network in DDOs was set to 32, providing an appropriate level of complexity for the specific task at hand. To enable thorough information exchange, a substantial number of 1000 diffusion steps were performed for the Gaussian diffusion process. The number of channels for the two cell dynamics datasets and the fluid dataset was determined as 20, taking into account the unique characteristics of each dataset. All three datasets were standardized to an image size of 32×32 pixels. In order to accelerate convergence, a learning rate of 8×10^{-5} was adopted.

A.5 RESULTS OF BENCHMARK METHODS

We have presented the results of our experiments in Table 4 and Table 5. Upon careful examination of the tables, it becomes evident that ConvLSTM outperforms other models, primarily due to its inherent capability to capture long sequences effectively. Additionally, our observations reveal that DDOs-FNO, DDOs-SFNO, and DDOs-TFNO do not exhibit satisfactory performance across the three datasets. This can be attributed to the inherent challenges faced by these models in terms of convergence when applied to biological datasets.

Table 4: Comparison of benchmarks in terms of MSE and Relative L2 norm on two datasets.

Time Step	Metrics	ConvLSTM	ViT	MLP-Mixer	DDOs-FNO	DDOs-SFNO	DDOs-TFNO
Tension							
TS = 1	MSE	0.0165 ± 0.0003	0.0869 ± 0.0002	0.0342 ± 0.0001	0.1288 ± 0.0010	0.1012 ± 0.0020	0.1622 ± 0.0020
	L2	0.2982 ± 0.0002	0.7431 ± 0.0001	0.4065 ± 0.0002	0.7218 ± 0.0020	0.6022 ± 0.0030	0.6407 ± 0.0010
TS = 3	MSE	0.0212 ± 0.0002	0.0842 ± 0.0002	0.0497 ± 0.0002	0.1295 ± 0.0030	0.1124 ± 0.0010	0.1734 ± 0.0040
	L2	0.3564 ± 0.0001	0.7289 ± 0.0001	0.4760 ± 0.0002	0.7418 ± 0.0030	0.6228 ± 0.0010	0.6538 ± 0.0010
TS = 5	MSE	0.0805 ± 0.0001	0.0839 ± 0.0003	0.0519 ± 0.0001	0.1306 ± 0.0020	0.1206 ± 0.0020	0.1801 ± 0.0030
	L2	0.6086 ± 0.0003	0.7319 ± 0.0001	0.5369 ± 0.0001	0.7512 ± 0.0030	0.6322 ± 0.0030	0.6622 ± 0.0010
TS = 7	MSE	0.0858 ± 0.0002	0.0849 ± 0.0003	0.1562 ± 0.0003	0.1311 ± 0.0020	0.1278 ± 0.0030	0.1846 ± 0.0020
	L2	0.6455 ± 0.0003	0.8217 ± 0.0002	0.7913 ± 0.0002	0.7715 ± 0.0010	0.6401 ± 0.0030	0.6703 ± 0.0030
TS = 9	MSE	0.0860 ± 0.0002	0.0884 ± 0.0002	0.1006 ± 0.0001	0.1387 ± 0.0020	0.1304 ± 0.0020	0.1887 ± 0.0020
	L2	0.6688 ± 0.0003	0.7738 ± 0.0002	0.7754 ± 0.0002	0.7815 ± 0.0020	0.6517 ± 0.0010	0.6806 ± 0.0010
Wet							
TS = 1	MSE	0.0798 ± 0.0002	0.1354 ± 0.0001	0.1282 ± 0.0002	0.1477 ± 0.0030	0.1319 ± 0.0030	0.1192 ± 0.0030
	L2	0.3612 ± 0.0001	0.6897 ± 0.0002	0.6019 ± 0.0002	0.6209 ± 0.0040	0.6028 ± 0.0030	0.6011 ± 0.0020
TS = 3	MSE	0.0843 ± 0.0002	0.1512 ± 0.0002	0.1267 ± 0.0001	0.1501 ± 0.0020	0.1489 ± 0.0010	0.1243 ± 0.0020
	L2	0.3861 ± 0.0001	0.6958 ± 0.0001	0.5989 ± 0.0002	0.6235 ± 0.0030	0.6172 ± 0.0010	0.6224 ± 0.0030
TS = 5	MSE	0.0899 ± 0.0002	0.1743 ± 0.0002	0.1325 ± 0.0001	0.1566 ± 0.0020	0.1512 ± 0.0010	0.1304 ± 0.0010
	L2	0.3872 ± 0.0001	0.7341 ± 0.0001	0.6057 ± 0.0001	0.6308 ± 0.0030	0.6172 ± 0.0010	0.6406 ± 0.0030
TS = 7	MSE	0.0965 ± 0.0002	0.1765 ± 0.0002	0.1369 ± 0.0001	0.1602 ± 0.0020	0.1599 ± 0.0030	0.1368 ± 0.0020
	L2	0.3946 ± 0.0002	0.7355 ± 0.0001	0.6129 ± 0.0001	0.6405 ± 0.0020	0.6509 ± 0.0030	0.6594 ± 0.0020
TS = 9	MSE	0.0992 ± 0.0002	0.1862 ± 0.0002	0.1653 ± 0.0001	0.1624 ± 0.0030	0.1665 ± 0.0020	0.1403 ± 0.0030
	L2	0.3970 ± 0.0002	0.7543 ± 0.0001	0.6408 ± 0.0001	0.6532 ± 0.0020	0.6673 ± 0.0030	0.6622 ± 0.0010

Table 5: Comparison of benchmarks in terms of MSE and Relative L2 norm on Jellyfish (Fluid).

Time Step	Metrics	ConvLSTM	ViT	MLP-Mixer	DDOs-FNO	DDOs-SFNO	DDOs-TFNO
TS = 1	MSE	0.0569 ± 0.0002	0.2440 ± 0.0003	0.1841 ± 0.0001	0.5012 ± 0.0020	0.5019 ± 0.0020	0.5563 ± 0.0010
	L2	0.4719 ± 0.0002	0.8754 ± 0.0001	0.8065 ± 0.0003	0.9019 ± 0.0020	0.9314 ± 0.0020	0.9209 ± 0.0040
TS = 3	MSE	0.0989 ± 0.0001	0.2608 ± 0.0003	0.1908 ± 0.0001	0.5218 ± 0.0030	0.5367 ± 0.0020	0.5678 ± 0.0020
	L2	0.5953 ± 0.0002	0.9027 ± 0.0001	0.7217 ± 0.0001	0.9215 ± 0.0010	0.9461 ± 0.0010	0.9287 ± 0.0030
TS = 5	MSE	0.1428 ± 0.0002	0.2773 ± 0.0001	0.2548 ± 0.0002	0.5517 ± 0.0030	0.5466 ± 0.0010	0.5466 ± 0.0030
	L2	0.6991 ± 0.0001	0.9247 ± 0.0002	0.7897 ± 0.0001	0.9484 ± 0.0030	0.9566 ± 0.0010	0.9309 ± 0.0010
TS = 7	MSE	0.1784 ± 0.0002	0.2887 ± 0.0003	0.2756 ± 0.0002	0.5674 ± 0.0010	0.5581 ± 0.0010	0.5522± 0.0030
	L2	0.7792 ± 0.0002	0.9373 ± 0.0001	0.8124 ± 0.0001	0.9518 ± 0.0010	0.9617 ± 0.0010	0.9455 ± 0.0010
TS = 9	MSE	0.2496 ± 0.0001	0.2965 ± 0.0001	0.2843 ± 0.0002	0.5718 ± 0.0020	0.5718 ± 0.0020	0.5609 ± 0.0030
	L2	0.8843 ± 0.0002	0.9513 ± 0.0001	0.8249 ± 0.0001	0.9645 ± 0.0030	0.9634 ± 0.0030	0.9534 ± 0.0030

A.6 BROADER IMPACTS AND LIMITATIONS

Broader Impact: The introduction of comprehensive large-scale datasets, namely Tension, Wets, CellDivision and Jellyfish, has significant implications for the field of biological fluid simulation. These datasets address the challenges faced by the community, including the lack of dynamic biological process capture and limited scale of existing datasets. By providing a standardized evaluation framework and incorporating physical modeling techniques, the datasets empower researchers to objectively assess and compare data-driven methodologies. This fosters advancements in the field and promotes the development of accurate and efficient models for simulating complex fluid dynamics within biological systems. The availability of benchmark datasets also enhances reproducibility and comparability of results across studies, facilitating knowledge sharing and collaboration within the research community.

Limitations: While the introduced datasets offer valuable resources for data-driven biological fluid simulation, they may have some limitations. First, the datasets are designed based on specific biological scenarios and may not encompass the full range of biological fluid dynamics. Researchers should be cautious when extrapolating findings to other systems. Second, the datasets rely on physical modeling techniques such as the phase-field method, which may introduce certain simplifications and assumptions that could impact the accuracy and applicability of the results. It is important to consider

the limitations and assumptions of the underlying models when interpreting the data. Finally, the scale and complexity of the datasets may pose computational and resource challenges for researchers with limited access to high-performance computing infrastructure.