Inferring micro-bubble dynamics with physics-informed deep learning

Hanfeng Zhai and Guohui Hu*

Shanghai Institute of Applied Mathematics and Mechanics,
Shanghai Key Laboratory of Mechanics in Energy Engineering,
Shanghai University, Shanghai 200072, PR China

*Corresponding author: ghhu@staff.shu.edu.cn

Abstract

Micro-bubbles and bubbly flows are widely observed and applied to medicine, involves deformation, rupture, and collision of bubbles, phase mixture, etc. We study bubble dynamics by setting up two numerical simulation cases: bubbly flow with a single bubble and multiple bubbles, both confined in the microtube, with parameters corresponding to their medical backgrounds. Both the cases have their medical background applications. Multiphase flow simulation requires high computation accuracy due to possible component losses that may be caused by sparse meshing during the computation. Hence, data-driven methods can be adopted as a useful tool. Based on physics-informed neural networks (PINNs), we propose a novel deep learning framework BubbleNet, which entails three main parts: deep neural networks (DNN) with sub nets for predicting different physics fields; the physics-informed part, with the fluid continuum condition encoded within; the time discretized normalizer (TDN), an algorithm to normalize field data per time step before training. We apply the traditional DNN and our BubbleNet to train the simulation data and predict the physics fields of both the two bubbly flow cases. Results indicate our framework can predict the physics fields more accurately, estimating the prediction absolute errors. The proposed network can potentially be applied to many other engineering fields.

Keywords: Physics-informed neural networks, machine learning, bubble dynamics, multiphase flow, microfluids.

1 Introduction

Machine learning (ML) has achieved tremendous success in the last decade due to the availability of big data and computer resources. ML is the study of computer algorithms that allow computer programs to automatically improve through experience [1]. The success of AlphaGo

burst the public's interest by showing the huge potential of machine learning and artificial intelligence [2, 3]. The ML techniques shows promising results, specifically in genomics [4, 5], public health [6, 11, 12, 13, 14], and medicine [7, 8, 9, 10], both are becoming more significant in our aging societies. Deep neural networks (DNNs), as one of the most prominent tools of ML, have been adopted to tackle various physics problems including turbulence [15], flow control [16], heat transfer [17], and combustion [18]. These deep learning applications have grown drastically in recent years, mainly on learning physical equations and inferring dynamics. Numerous novel frameworks has henceforth been proposed: SINDy [19] and PDE-FIND [20], using sparse regression to identify the governing equations for nonlinear dynamic systems; Graph Kernel Network [22], Fourier Neural Operator [21] and MeshfreeFlowNet [23], using convolutional neural networks to learn image mapping the physics fields; Deep potential [24], DeePMD [25] and DeePCG [26], using deep neural nets to map the molecular potentials at the microscale. In 2018, Raissi et al. [27, 28, 29] proposed a deep learning framework called physics-informed neural networks (PINNs) for identifying and inferring dynamics of physical systems governed by partial differential equations (PDEs). The strategy of PINN can be simplified as encoding governing PDEs into the loss function as a soft physics constraint, namely the 'physics-informed' part. Based on PINN, Lu et al. then proposed DeepXDE [30] and DeepONet [31], the refined versions of PINNs, for learning and inferring nonlinear operators of PDEs, which were later applied to electroconvection multiphysics [32] and hypersonics [33].

The PINN series has been developed to solve numerous problems, including Fractional PINNs for predicting fractional PDEs; Conservative PINNs for nonlinear conservation laws [35]; Extended PINNs, a PINN approach for space-time domain decomposition [36]; Parareal PINNs, a PINN solver decomposing a long-time problem into many independent short-time problems supervised by an inexpensive/fast coarse-grained (CG) solver [37]; and many related PINNs. Besides, PINNs has achieved great success for predicting laminar flows [38] and high speed flows [41], heat transfer [39] and turbulence [40]. Notably, Lin et al. [42] have used the prementioned DeepONet to predict bubble dynamics adopting the ideal bubble models.

Bubbly flows is a classic fluid mechanics problem, specifically at the mesoscale. The bubble pinch-off effect confined in microtube is one of the most studied problems in fluid mechanics [43, 44, 45, 46, 47], depicting deformation and movement of single bubble dynamics. Single bubble movement has also been studied for investigating blood-brain barrier [48, 49, 50]. Bubbly flow with multiple bubbles displays complexity due to the interactions between bubbles [51, 52], which are also widely studied for drug delivery [53, 54].

Acknowledging the importance and numerous applications of bubbly flow, and realizing the powerful tool of physics based machine learning, we here apply machine learning to infer and predict the bubbly flow. Our work is inspired by Raissi et al. [27] and Lin et al. [42] using physics-informed deep learning to infer bubble dynamics. Different from Lin et al.'s approach, we do not adopt the theoretical model optimized for modeling bubble (i.e., the Rayleigh-Plesset equation). Instead, we directly compute how bubble moves through numerical simulations of multiphase flow. We also propose a new deep learning architecture, called BubbleNet, for

predicting such.

The paper is arranged as follows: in Part I we introduce our approach of study bubbly flow and set up two numerical cases for computing the bubble flow in microfluidics: the single bubble flow and multiple bubbles flow confined in the microtube. We thence estimate the bubble(s) motion and validate the accuracy of simulations. In Part II we introduce the traditional DNNs and our BubbleNet algorithms, respectively. And we train the data from the data of numerical simulations on each NN for predictions. We hence obtain and analyze different errors from machine learning. We eventually conclude the paper from the DNN and BubbleNet results in Part III.

2 Problem formulation

To investigate how bubble flows in the microtube, we apply computational fluid dynamics (CFD) for 2D bubbly flow simulations, adopting the time-dependent level set algorithms for modeling bubbles, using COMSOL Multiphysics®.

In the computations, for level set algorithms, the Navier-Stokes equation takes the form:

$$\frac{\partial \phi}{\partial t} + \nabla \phi \cdot \mathbf{u} = 0$$

where the level set phase function, ϕ, varies between 0 and 1 across the free surface and is constant at 0 or 1 in the bulk of the two fluids. Here, ϕ is 0 for the liquid and 1 for the gas phase. And $\mathbf{u}$ is the velocities, where $\mathbf{u} = (u,\ v)$ in our 2D simulations.

For the time-dependent level set simulations, we set ρ_1 as the water density and ρ_2 as the air density; μ_1 as the water viscosity and μ_2 as the air viscosity. Their relationships can be connected through the level set function, as follows:

$$\rho = \rho_1 + \phi(\rho_2 - \rho_1)$$

$$\mu = \mu_1 + \phi(\mu_2 - \mu_1)$$

To describe the interface between phases, specifically the liquid-gas interfaces, the level set computation method obeys:

$$\frac{\partial \rho}{\partial t} + \mathbf{u} \cdot \nabla \phi = \gamma \nabla \cdot \left(\epsilon_{ls} \nabla \phi - \phi(1-\phi) \frac{\nabla \phi}{|\nabla \phi|} \right)$$

where γ is the reinitialization parameter, which equals 1 in our cases. ϵ_{ls} is the parameter controlling the interface thickness, equals 0.43 in our cases.

Two cases were set up for investigating the bubble flow: [1]single bubble flow and [2]multiple bubbles flow, confined in a microtube. For the single bubble case: the microbubble diameter

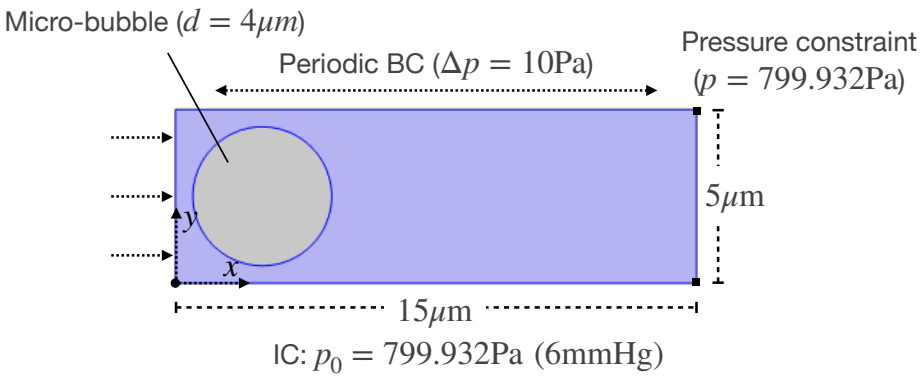

Figure 1. Modeling of the single bubble flow case.

is set to be $d = 4\mu$m; the micro-tube has a length of 15μm and diameter of 5μm; we set up periodic boundary conditions (BCs) as $\Delta p = 10$Pa, and the tube is confined at the end as constant pressure $p = 799.932$Pa (6mmHg), corresponding to the pressure in human interstitial fluid; the initial conditions (ICs) is set as the pressure $p_0 = 799.932$Pa, with room temperature as 293.15K; shown as in figure 1.

For the multiple bubbles flow: 60 micro-bubbles are randomly distributed in a 2D micro-tube with a length of 100μm and a diameter of 50μm, with each micro-bubble of the diameter of 3μm. The BCs and ICs are same with the single bubble case: $\Delta p = 10$Pa, constant pressure at tube end $p = 799.932$Pa and $p_0 = 799.932$Pa, with room temperature as 293.15K. Both the meshing of the two bubbly flows are set to be extremely fine. For a single bubble case, the simulation is run for 5000μs, and for the multiple bubbles case, the simulation is run for 3000μs.

The single bubble motion in microflow is shown in figure 3, where the bubble is reported at 9 different time steps, starting from 400μs to 3600μs. The motion shows that the right bubble side flows outwards in a parabolic shape, while the left bubble side flows inwards with two side bubbles trending to be ruptured from the main bubble. The motion of the main bubble corresponds to the deformation of red blood cells confined in micro-tube, by Tomaiuolo et al. The multiple bubbles motion indicates bubbles tend to collide and ruptured from each other, resulting to form bigger bubbles, as shown in figure 4.

In multiphase flow simulations, multiple factors (i.e., meshing, BCs, ICs, solvers) could cause component losses, leading to inaccurate results. To validate our simulations, we test the components ratio, namely the liquid-gas ratio (L-G ratio), during the whole computation process, shown in figure 5 and 6 for both the single and multiple bubbles cases. Here, for our cases, if we use λ to signify the component ratio of one element, using A to signify the area of the 2D modeling. Thence, we could compute component ratio of air, for derivation of the L-G ratio: $\lambda_{air} = A_{air}/\left(A_{water} + A_{air}\right)$. The L-G ratio can therefore be computed by $\lambda_{water}/\lambda_{air}$.

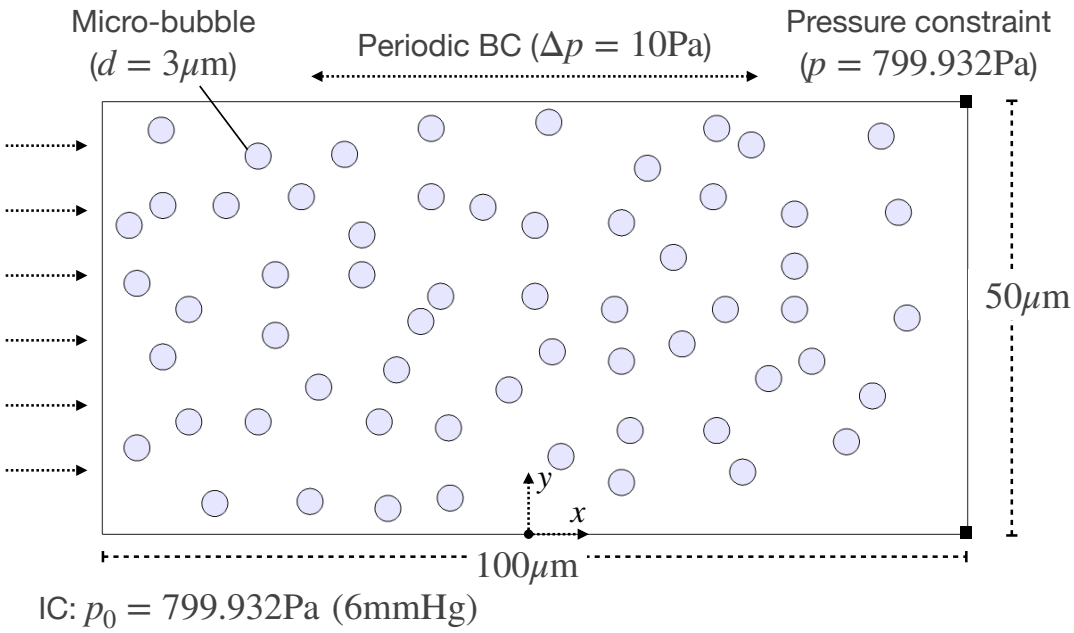

Figure 2. Modeling of the bubbly flow with multiple bubbles.

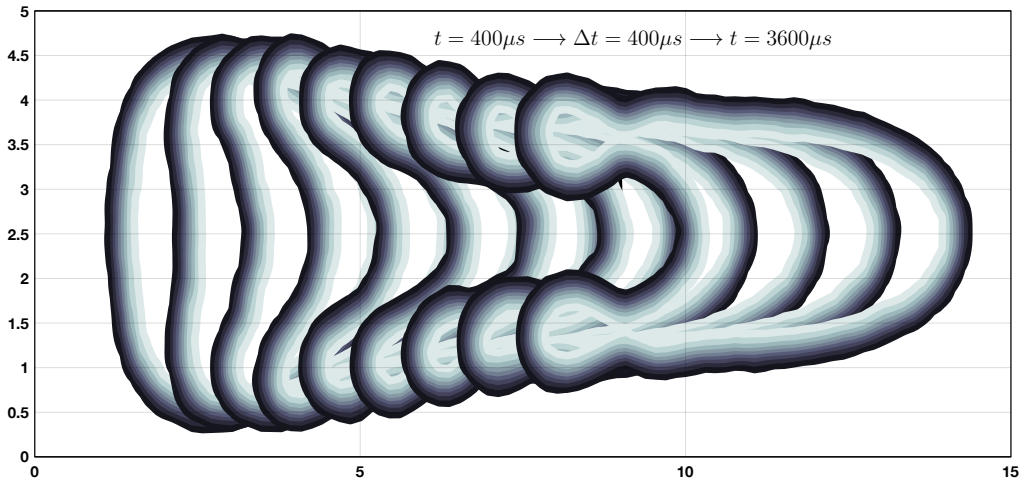

Figure 3. Simulation results of the single bubble flow. The bubble deformation is reported at 9 different time step, starting from 400μs to 3600μs.

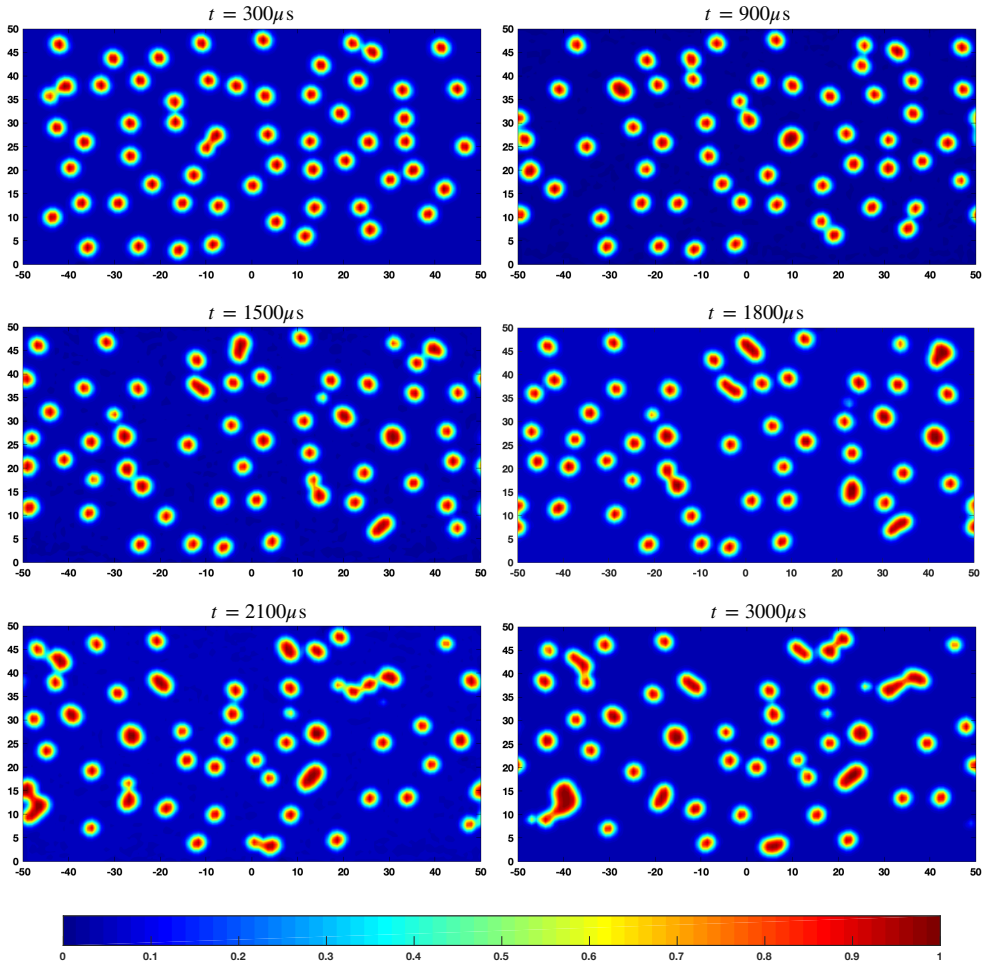

Figure 4. Simulation results of the multiple bubbles flow. The flow is reported at 6 different time steps, starting from 300μs to 3000μs.

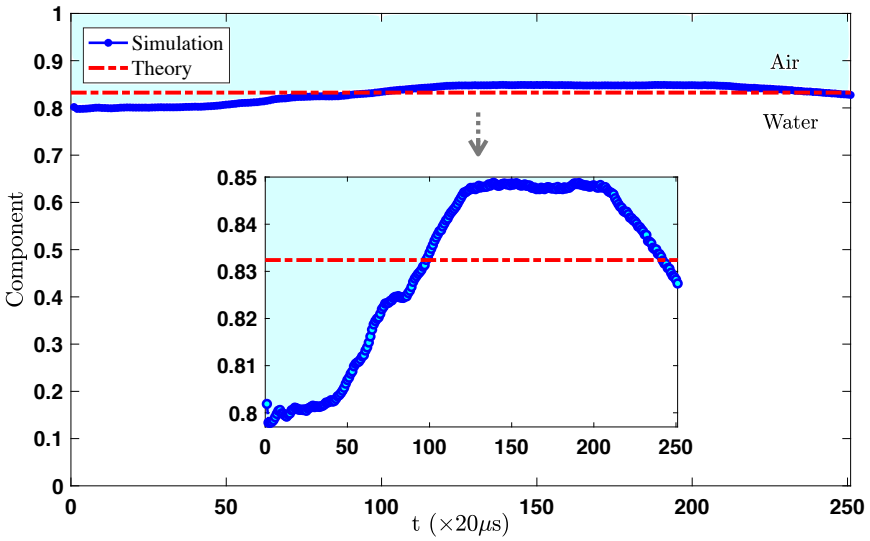

Figure 5. The liquid-gas ratio of the single bubble simulation process.

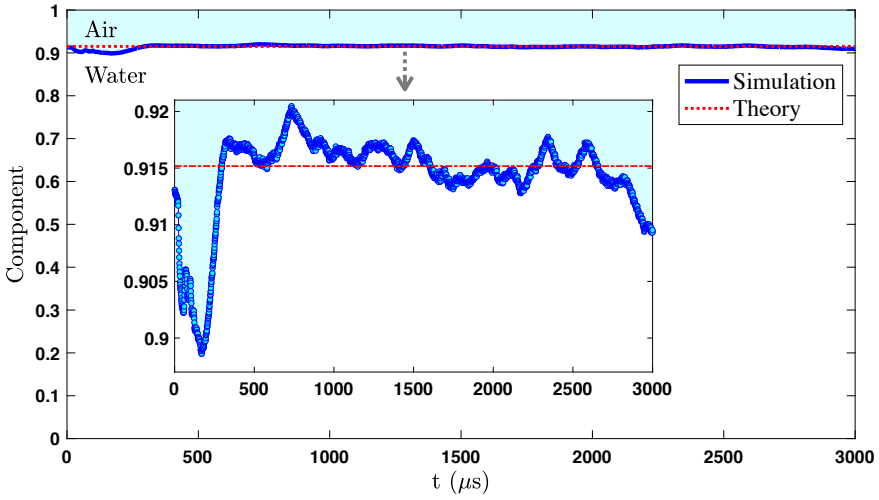

Figure 6. The liquid-gas ratio of the multiple bubbles simulation process.

Both the figures show that the L-G ratio, or more specifically the water-air ratio, generally agrees with the theoretical results, with relative errors smaller than 5%, validates our accurate simulations.

3 Deep learning algorithms

In this section, we briefly introduce the basis of deep learning and neural networks; then we propose our approach for using deep neural networks (DNNs) and our framework BubbleNet to predict bubbly flows physical fields. Both the DNN and BubbleNet are trained on our simulation data and the bubble motion is predicted respectively.

3.1 Traditional DNNs

A traditional neural net (NN) consists of input layers, hidden layers, and output layers, fully connected within. DNNs are NNs with multiple hidden layers that are able to approximate nonlinear operators and mapping between two Banach spaces.

Here we apply a DNN with four sub nets, Net_u, Net_v, Net_p, Net_ϕ, for predicting the physics fields u, v, p, ϕ, respectively. Each sub net consists of 9 layers with 30 neurons for each layer. The input data are the field data in the space-time domain, namely x, y, t. For the training process, we adopt the Adam optimizer and the 'L-BFGS-B' optimization method. Each neuron is activated by the *tanh* activation function. The maximum iteration for 'L-BFGS-B' optimization is 500000. To reduce the computation resources, we rescale our training data: the single bubble simulation data is extracted every 10 points in the space domain and 40 points on the time domain; the multiple bubbles case data is extracted every 20 points on the space domain and 30 points on the time domain. For the single bubble case, the DNNs are trained for 10000 iterations on the normalized data, and we aim to predict the physics fields on 2000μs. For the single bubble case, the DNNs are trained for 200000 iterations on the normalized data, and we aim to predict the physics fields on 1500μs. The algorithm for our DNN is shown in **Algorithm 1**.

3.2 Physics-informed neural networks

Physics-informed neural networks (PINNs), as introduced, encode physics equations into the loss function, making the NN approximate the real physics equations during trainings. Our algorithm BubbleNet encode the continuum equation of incompressible fluids in the inference

process, eliciting the latent function ψ for predicting the velocity fields $u,\ v$:

$$u = \frac{\partial \psi}{\partial y},\ v = -\frac{\partial \psi}{\partial x}$$

when $u,\ v$ is computed from ψ, the continuum condition is automatically satisfied.

In our training, we use mean squared error (MSE) for computing the loss function, as the deviation of predictions and training data. If we use U to represent training data for the NN containing the physics fields, then the loss function takes the form:

$$Loss = MSE(U_{pred} - U_{train})$$

where U_{pred} is the predictions of the NN training and U_{train} is the normalized training data.

Here, if the original simulation data pertaining to the physics fields of bubbly flow is $\mathcal{U}$, where $\mathcal{U} = (u,\ v,\ p,\ \phi)$, then the training data U can be obtained through normalization, signified by the normalization function $\mathbf{N}$:

$$U = \mathbf{N}(\mathcal{U})$$

wherein BubbleNet $\mathbf{N}$ is TDN; while in DNNs $\mathbf{N}$ is the MaxMinScaler. The algorithm of BubbleNet are shown in **Algorithm 2**. The parameters of NN are same as **Algorithm 1**.

Algorithm 1 DNN for predicting bubble dynamics

function DEEPNEURALNET($self,\ x,\ y,\ t,\ u,\ v,\ p,\ \phi,\ layers$)
  $(\hat{x},\ \hat{y},\ \hat{t},\ \hat{u},\ \hat{v},\ \hat{p},\ \hat{\phi}) =$ UPDATE($x,\ y,\ t,\ u,\ v,\ p,\ \phi$)
  $(\hat{weights},\ \hat{biases},\ \hat{layers}) = self.$INITIALIZENN($weights,\ biases,\ layers$)
  $self.$Loss = MSE$[(u - u_{pred}) + (v - v_{pred}) + (p - p_{pred}) + (\phi - \phi_{pred})]$
  $u_{pred} = self.\text{Net}_\text{u}(x,\ y,\ t)$
  $v_{pred} = self.\text{Net}_\text{v}(x,\ y,\ t)$
  $p_{pred} = self.\text{Net}_\text{p}(x,\ y,\ t)$
  $\phi_{pred} = self.\text{Net}_\phi(x,\ y,\ t)$
  Optimization method 'L-BFGS-B' & Optimizer: Adam
  def INITIALIZENN($self,\ layers$)
    Initialize all the *weights* & *biases* for Net_u, Net_v, Net_p, Net_ϕ.
  def NEURALNET($self,\ weights,\ biases$)
    Build NN for $u,\ v,\ p,\ \phi$ with four sets of *weights* & *biases*.
  def $\{\text{Net}_\text{u}, \text{Net}_\text{v}, \text{Net}_\text{p}, \text{Net}_\phi\}$ ($self,\ x,\ y,\ t$)
    $\{u,\ v,\ p,\ \phi\} = self.$NEURALNET($x,\ y,\ t,\ weights,\ biases$)
  def TRAIN($self,\ iterations$)
    Obtain training time & Losses; train the NN with Adam optimizer.
  def PREDICT $\{u,\ v,\ p,\ \phi\}$ ($self,\ iterations$)
    $\{u_{pred},\ v_{pred},\ p_{pred},\ \phi_{pred}\} = self.$ sess.run($x,\ y,\ t$)
end function
Input = $\{x,\ y,\ t\}$, Output = $\{u,\ v,\ p,\ \phi\}$
Hidden layers = [30 neurons $\times$ 9 layers]
Load fields data of micro-bubble system dynamics simulation.
Set training sets = $\{x_{train},\ y_{train},\ t_{train},\ u_{train},\ v_{train},\ p_{train},\ \phi_{train},\ layers\}$
  = MaxMinScaler(Simulation Data)
model = DEEPNEURALNET(training sets)
model.TRAIN($\#Iterations$)
Set target prediction time as t_{pred}
Obtain $\{u_{pred},\ v_{pred},\ p_{pred},\ \phi_{pred}\}$ = model.PREDICT($x,\ y,\ t$) at t_{pred}.
Save all the data & post-processing.

Algorithm 2 BubbleNet: physics-informed neural network for bubble dynamics

function BUBBLENET($self,\ x,\ y,\ t,\ u,\ v,\ p,\ \phi,\ layers$)
  $(\hat{x},\ \hat{y},\ \hat{t},\ \hat{u},\ \hat{v},\ \hat{p},\ \hat{\phi}) =$ UPDATE($x,\ y,\ t,\ u,\ v,\ p,\ \phi$)
  $(\hat{weights},\ \hat{biases},\ \hat{layers}) = self.$INITIALIZENN($weights,\ biases,\ layers$)
  $self.$Loss $=$ MSE$[(u - u_{pred}) + (v - v_{pred}) + (p - p_{pred}) + (\phi - \phi_{pred})]$
  $\{u_{pred}, v_{pred}, p_{pred}, \phi_{pred}\} = self.\{\text{Net}_\psi, \text{Net}_\text{p}, \text{Net}_\phi\}(x,\ y,\ t)$
  Optimization method 'L-BFGS-B' & Optimizer: Adam
  def INITIALIZENN($self,\ layers$)
    Initialize all the $weights$ & $biases$ for Net_ψ, Net_p, Net_ϕ.
  def NEURALNET($self,\ weights,\ biases$)
    Build NN for $\psi,\ p,\ \phi$ with four sets of $weights$ & $biases$.
  def $\{\text{Net}_\psi, \text{Net}_\text{p}, \text{Net}_\phi\}$ ($self,\ x,\ y,\ t$)
    $\{\psi,\ p,\ \phi\} = self.$NEURALNET($x,\ y,\ t,\ weights,\ biases$)
    $u = \partial_y \psi$ & $v = -\partial_x \psi$
  def TRAIN($self,\ iterations$)
    Obtain training time & Losses; train the NN with Adam optimizer.
  def PREDICT $\{u,\ v,\ p,\ \phi\}$ ($self,\ iterations$)
    $\{u_{pred},\ v_{pred},\ p_{pred},\ \phi_{pred}\} = self.$ sess.run($x,\ y,\ t$)
end function
Set training sets $= \{x_{train},\ y_{train},\ t_{train},\ u_{train},\ v_{train},\ p_{train},\ \phi_{train},\ layers\}$
  $=$ TimeDiscretizedNormalization(Simulation Data, $timestep$)
model $=$ BUBBLENET(training sets)
model.TRAIN($\#Iterations$)
Rest procedures same as **Algorithm 1**

The schematic for our proposed PINN architecture BubbleNet is shown in figure 7, where we use three sub nets: $\text{Net}_\psi,\ \text{Net}_p,\ \text{Net}_\phi$, for predicting $\psi,\ p,\ \phi$ and compute the velocities through automatic differentiations. We also elicit Time discretized normalizer (TDN), a function that are able to normalize the training data per time steps.

The DNNs and BubbleNet's loss - iterations diagrams for both the single bubble and multiple bubbles simulation cases are shown in figure 8 and 9, respectively. Figure 8 indicates traditional DNNs exhibits lower losses and with longer iterations for single bubble case, whilst BubbleNet stops training at earlier stage with higher losses. Figure 9 shows different trends: BubbleNet exhibits lower losses with more iterations & training. Yet both DNNs displays similar fluctuating losses (blue line & red line in the down side in figure 8 & 9).

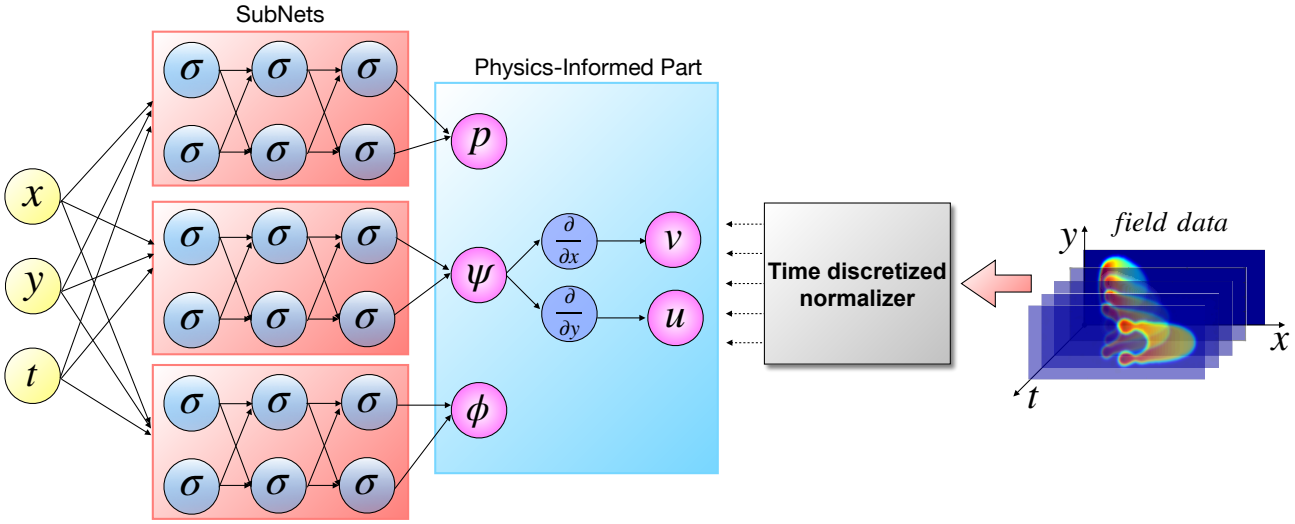

Figure 7. The schematic view of our proposed deep learning framework BubbleNet, consisting of a normal deep neural net and the physics-informed part that entails physics equation, and Time discretized normalizer for data preprocessing.

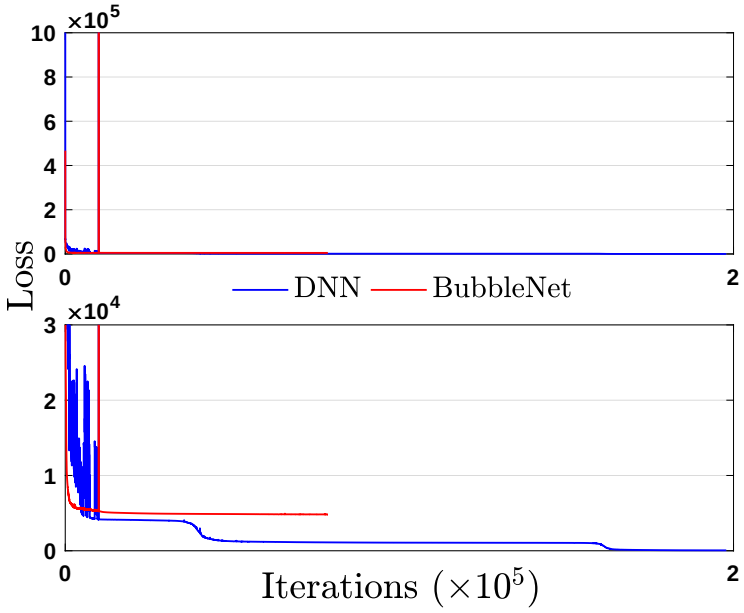

Figure 8. Loss function during training for the single bubble case.

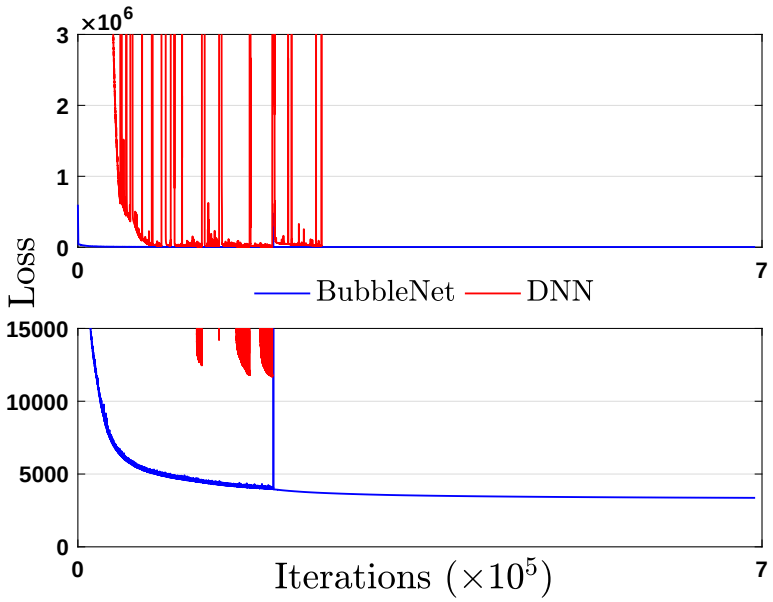

Figure 9. Loss function during training for the multiple bubbles case.

4 Results and discussion

4.1 Predictions

For the single bubble flow case, we use DNN and BubbleNet to predict bubbly flow's physics fields, compared with our benchmark by 2D CFD simulations. The predicted physics fields u, v, p, ϕ comparison are shown in figures 10, 11, 12, 13. From the velocities predictions in figures 10 & 11, BubbleNet evidently outperforms traditional DNN on approximating the physical trends. Here, two factors may account for the DNN's inaccurate predictions: I. The velocities data are in the 10^{-3} scale at our targeted prediction time (2000μs), which is comparably small among all the u, v magnitudes in the space-time domain. Hence, normalization on the whole space-time domain causes the velocities at $t = 2000\mu$s to lose their features due to the small scale. Such operations cause the NN to be 'cheated' during training approximating the data on the whole space-time domain, making those time steps with velocities of small magnitudes (i.e., $t = 2000\mu$s) are neglected. II. To show how our BubbleNet can approximate the physics more accurately, we only train the NNs for 10000 iterations. Such training is not enough to make DNNs approximating the training data. The inaccurate predictions may be refined if trained with more iterations. Here, we only apply the relatively small training steps to show how the deep learning structures vary on the predictions.

Also, for pressure p and level set phase function ϕ, both the DNN and BubbleNet displays good accuracy. In figure 12, BubbleNet has better prediction accuracy on the numerical magni-

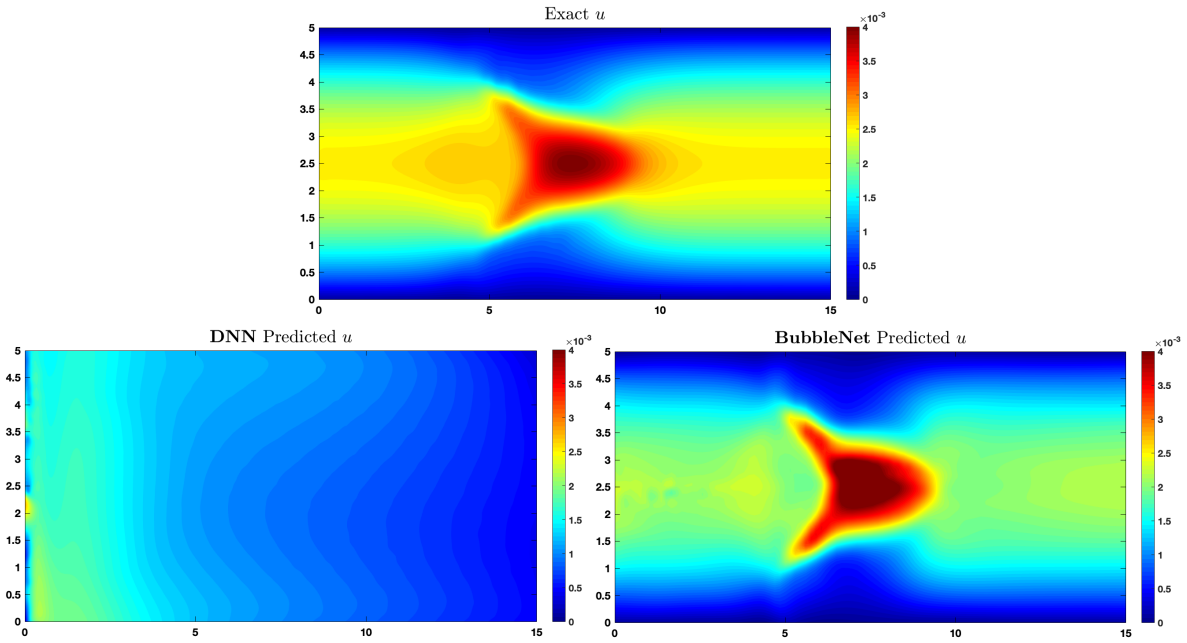

Figure 10. The comparison between the simulation results, DNN and BubbleNet predicted velocity fields u for the single bubble flow case.

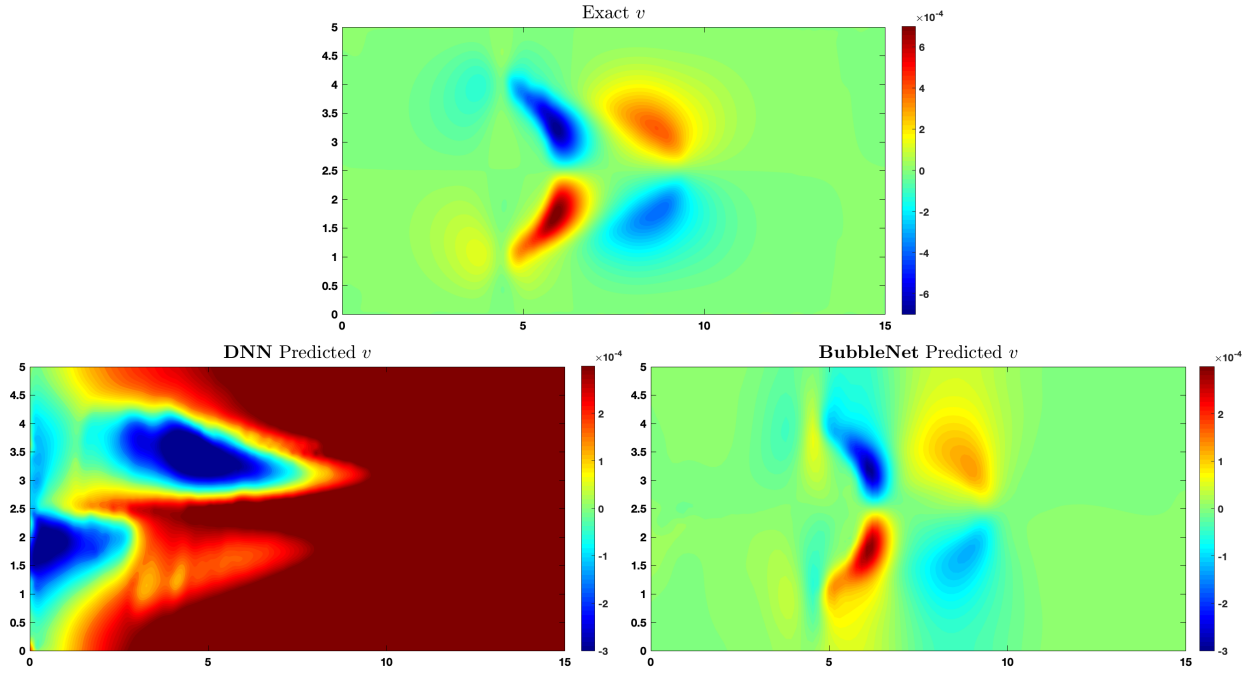

Figure 11. The comparison between the simulation results, DNN and BubbleNet predicted velocity fields v for the single bubble flow case.

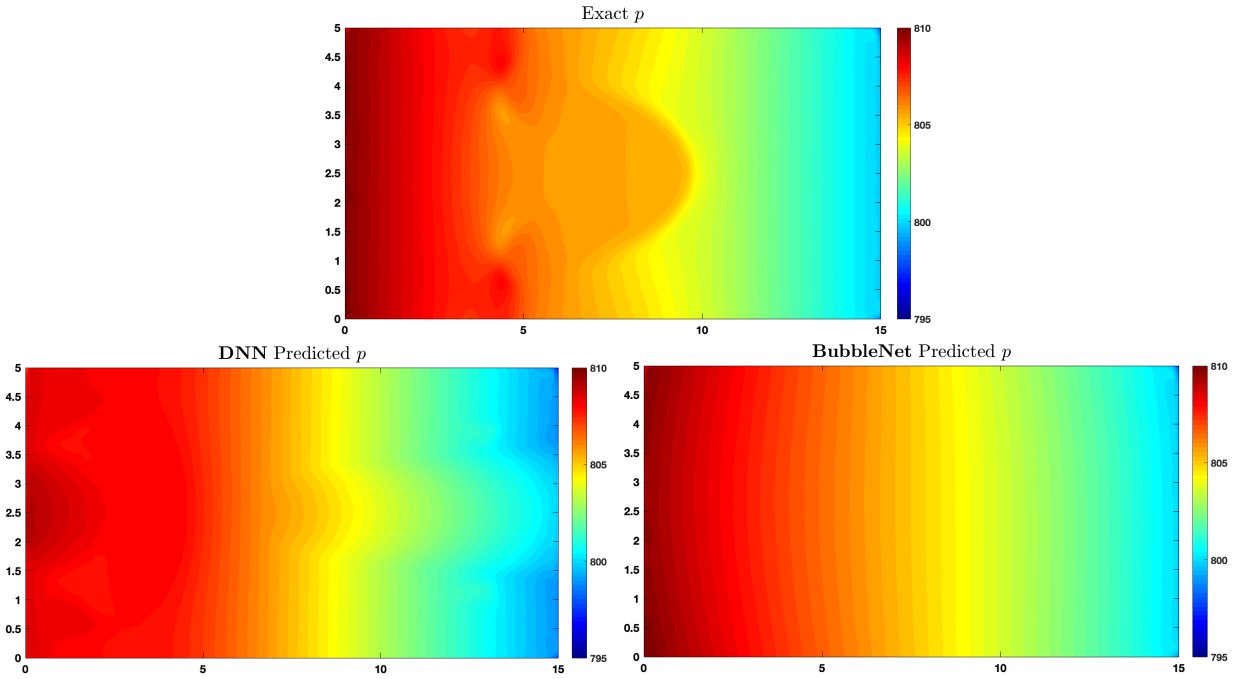

Figure 12. The comparison between the simulation results, DNN and BubbleNet predicted pressure fields p.

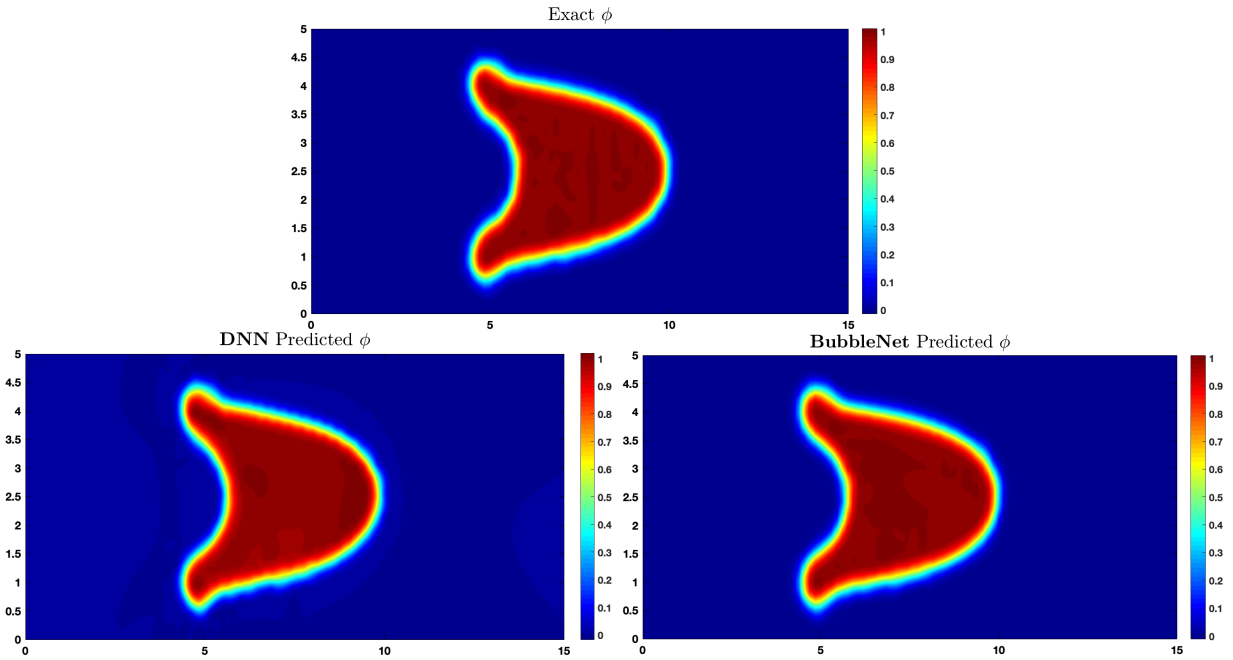

Figure 13. The comparison between the simulation results, DNN and BubbleNet predicted phase fields ϕ for the single bubble flow case.

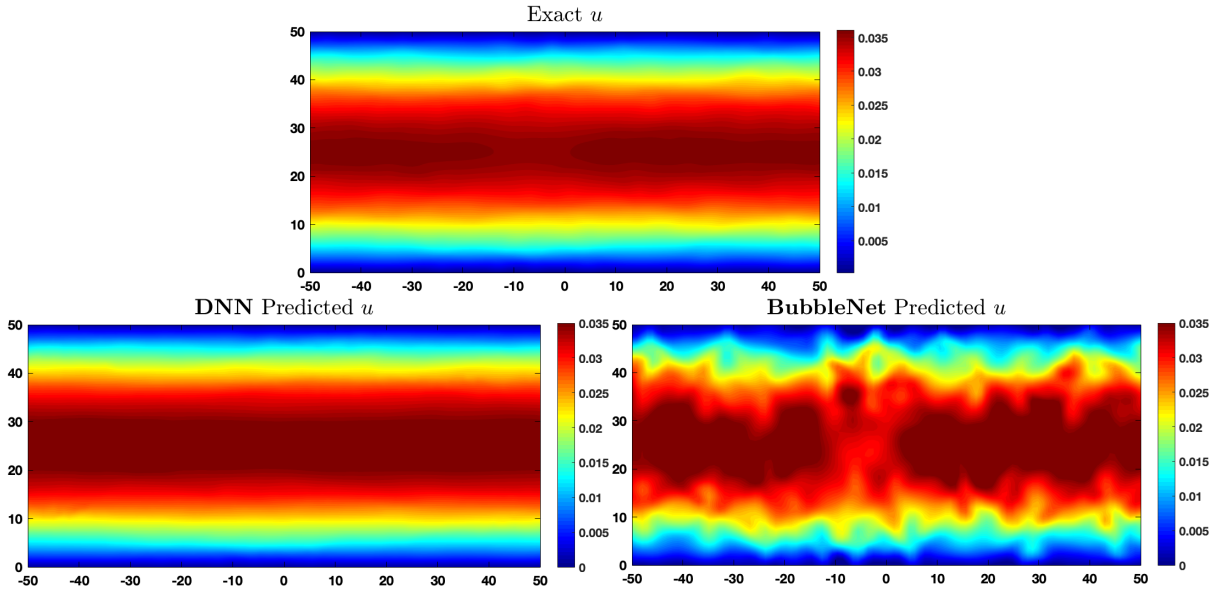

Figure 14. The comparison between the simulation results, DNN and BubbleNet predicted velocity fields u for the multiple bubbles flow case.

tudes, yet both the DNN and BubbleNet do not successfully capture the bubble shape feature details in the pressure field. We explain such by: I. The NN training requires more iterations. II. The subtle variance of pressure magnitude depicting the bubble shape weighs too small compared with the large pressure range ($[795, 800]$ shown in figure 12). The NN approximate mainly on the large pressure difference within limited iterations, neglecting the small variance, during the training. In figure 13 both NN structures exhibits good accuracy on predicting ϕ. Here, ϕ varies between $[0, 1]$, depicting the phase of two fluids. The relatively large data variance is easier to be detected by the NN, accounting for both the accurate predictions.

For the bubbly flow with multiple bubbles, more characteristics are involved in the data, i.e., more variations within the same data length caused by the phase transitions of multiple small bubbles; hence we trained the data with more iterations. The predictions by DNN and BubbleNet are shown from figure 14 to 17. For the bubbly flow with multiple bubbles, velocity u is not accurately predicted by BubbleNet in figure 14. Two factors may be accounted for: I. BubbleNet mainly focuses on velocity v while training, due to its more complex field distribution. The continuum condition we encoded connects velocities u and v. While v is more accurately approximated, the u field distribution is hence 'compromised' by the v field. II. The v field is not perfectly approximated, as the v predictions focus more on the y-axial features, in figure 15. Such inaccurate approximation also contributes to the inaccurately predicted u field.

Both the DNN and BubbleNet displays good p field predictions, in figure 16. The x-axial pressure difference is evident and simple, making approximating such a trend not a complex

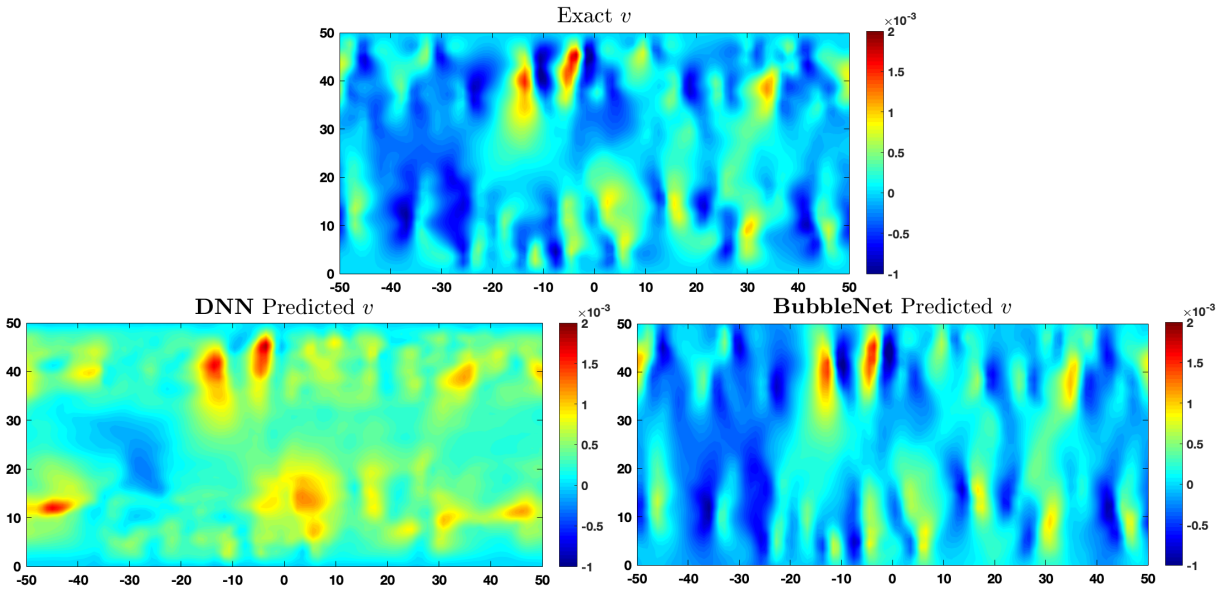

Figure 15. The comparison between the simulation results, DNN and BubbleNet predicted velocity fields v for the multiple bubbles flow case.

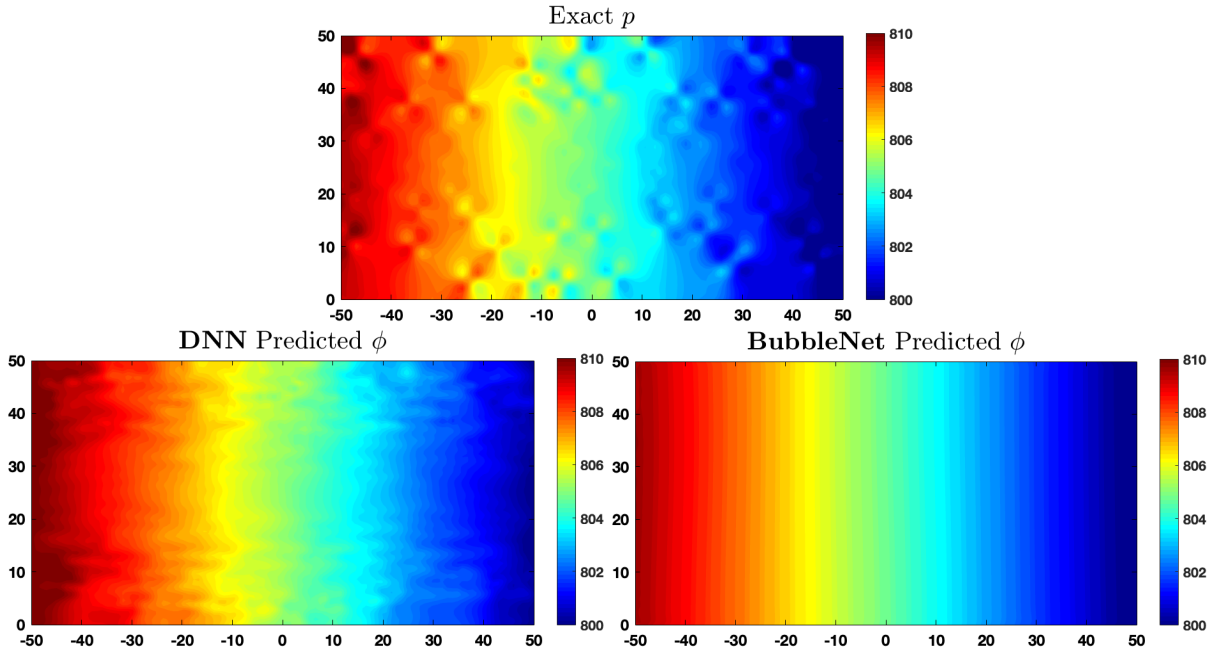

Figure 16. The comparison between the simulation results, DNN and BubbleNet predicted pressure fields p for the multiple bubbles flow case.

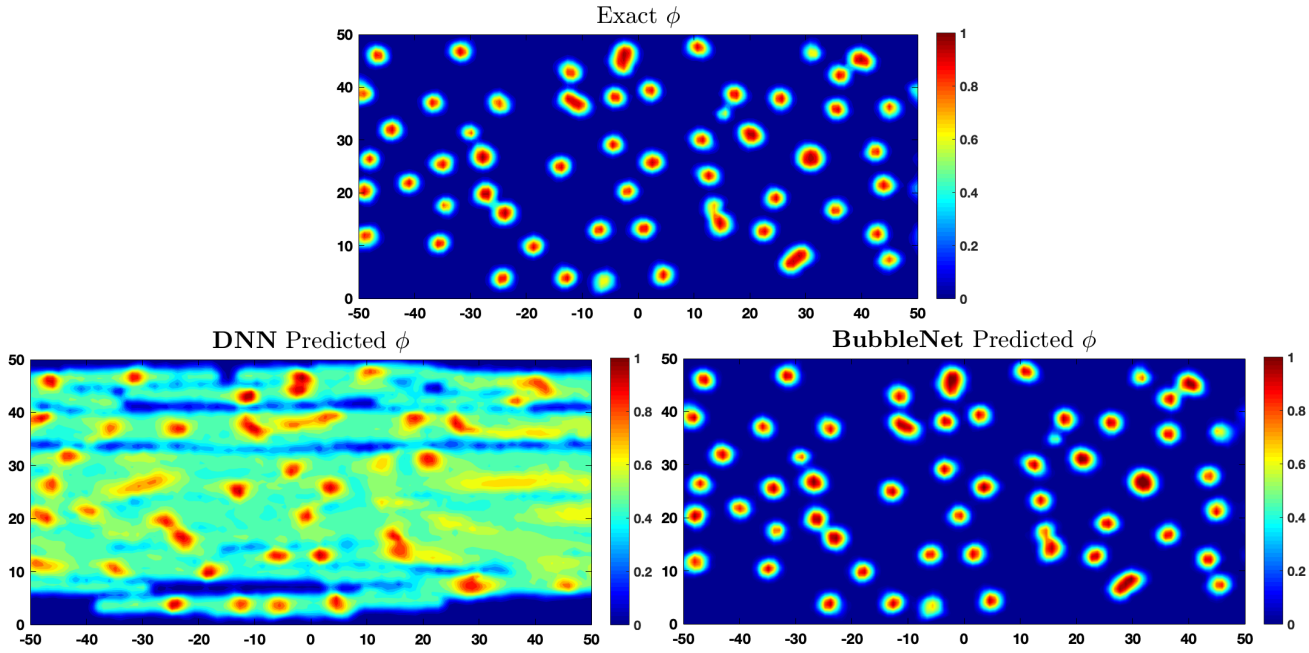

Figure 17. The comparison between the simulation results, DNN and BubbleNet predicted phase fields ϕ for the multiple bubbles flow case.

task. Notably, BubbleNet approximate the numerical magnitude more accurately, as can be judged from the color difference. The level set phase function ϕ, depicts how bubbles move and interact, can be adopted as the key role for describing bubble dynamics. From the predictions shown in figure 17, we see BubbleNet shows accurate approximation on ϕ field yet DNN cannot accurately predict such. The ϕ field data requires no normalization since all the data ranges $[0, 1]$. Here we explain the DNN inaccuracy by mentioning our DNN structure: A main neural net consisting of four sub nets for predicting the four physics fields. Since Net_ϕ is only one of the sub net constructing the NN, it may not be fully trained during the whole 200k iterations. The NN train the sub nets by our preset orders: Net_u, Net_v, Net_p, Net_ϕ, as shown in **Algorithm 1**. While the previous physics fields not accurately approximated during training, ϕ field data is hence not fully trained on.

4.2 Error analysis

For detailed comparison and estimation for different deep learning algorithms, we here propose two different errors, the relative error $\bar{\epsilon}$, of the NN training, and the absolute error of predictions $|\epsilon|$, for estimating the training process and the actual prediction variance.

The training error, specifically the relative error $\bar{\epsilon}$, shows how well the NN approximate the

training data of the network input and supervision data, takes the form:

$$\overline{\epsilon} = \frac{U_{pred} - U_{train}}{U_{train}}$$

where U_{pred} is the NN prediction by iterations during training and U_{train} is the training data.

As previously defined, $\mathcal{U}$ is the real physics computed by simulations and U is the training data for NN. Hence, we compute the predictions by $\mathcal{U}_{pred} = \mathbf{N}^{-1}U_{pred}$, where $\mathbf{N}$ is the normalizer for preprocessing the training data. In our BubbleNet, $\mathbf{N}$ = TDN. We use $\mathcal{U}_{exact}$ to denote the computation benchmark ($u,\ v,\ p,\ \phi$) by simulations. The absolute errors hence takes the form:

$$|\epsilon| = |\mathcal{U}_{pred} - \mathcal{U}_{exact}|$$

which are used to describe how well the NNs performs on real physics predictions.

The relative errors of both the DNN and BubbleNet training on the physics fields $u,\ v,\ p,\ \phi$, for both the two bubbly flow cases are shown in figures 18 and 19, in which the upper side views $\overline{\epsilon}$ are in larger data scale adjusting the NN with higher $\overline{\epsilon}$, and in the lower views $\overline{\epsilon}$ are in the data scale adjusting the NN with lower $\overline{\epsilon}$. From both the two figures it's hard to conclude or determine which NN structure performs better during the training since $\overline{\epsilon}$ varies from different physics fields. For the single bubble case, BubbleNet displays higher $\overline{\epsilon}$ on $u,\ v$, and ϕ fields, indicating less approximation accuracy on the training data. For the multiple bubbles flow case, BubbleNet displays lower $\overline{\epsilon}$ on p and ϕ, contending more accurate training approximations.

The absolute errors, $|\epsilon|$, of the four physics fields $u,\ v,\ p,\ \phi$, (or $\mathcal{U}$) for the single bubble case by both the DNN and BubbleNet, are shown in figures 20, 21, 22 and 23. The results indicate that only the ϕ field BubbleNet exhibits a bit higher $|\epsilon|$, less than 0.3. Such a phenomenon may be accounted for the less focus on Net_ϕ while training the other sub nets, since the ϕ field is already approximated in good shape in figure 13. Notwithstanding, BubbleNet still shows less $|\epsilon|$ distribution on other fields and the ϕ field is also predicted obeying the physical trends.

The $|\epsilon|$ for $\mathcal{U}$, for the bubbly flow with multiple bubbles by both the DNN and BubbleNet, are shown in figures 24, 25, 26 and 27. Figure 24 contends BubbleNet exhibits higher $|\epsilon|$ on u field, corresponds with figure 14. Such a result is already explained previously. Overall, for the multiple bubbles case, BubbleNet also displays lower $|\epsilon|$ distribution generally, indicating good predition accuracy. In short, BubbleNet displays lower $|\epsilon|$ on most physics fields for both the two bubbly flow cases, proving the accuracy of this framework.

5 Summary and conclusion

In this study, we apply the traditional DNN and proposed a novel deep learning framework, BubbleNet, for inferring and predicting the physics information of bubbly flow. We first study

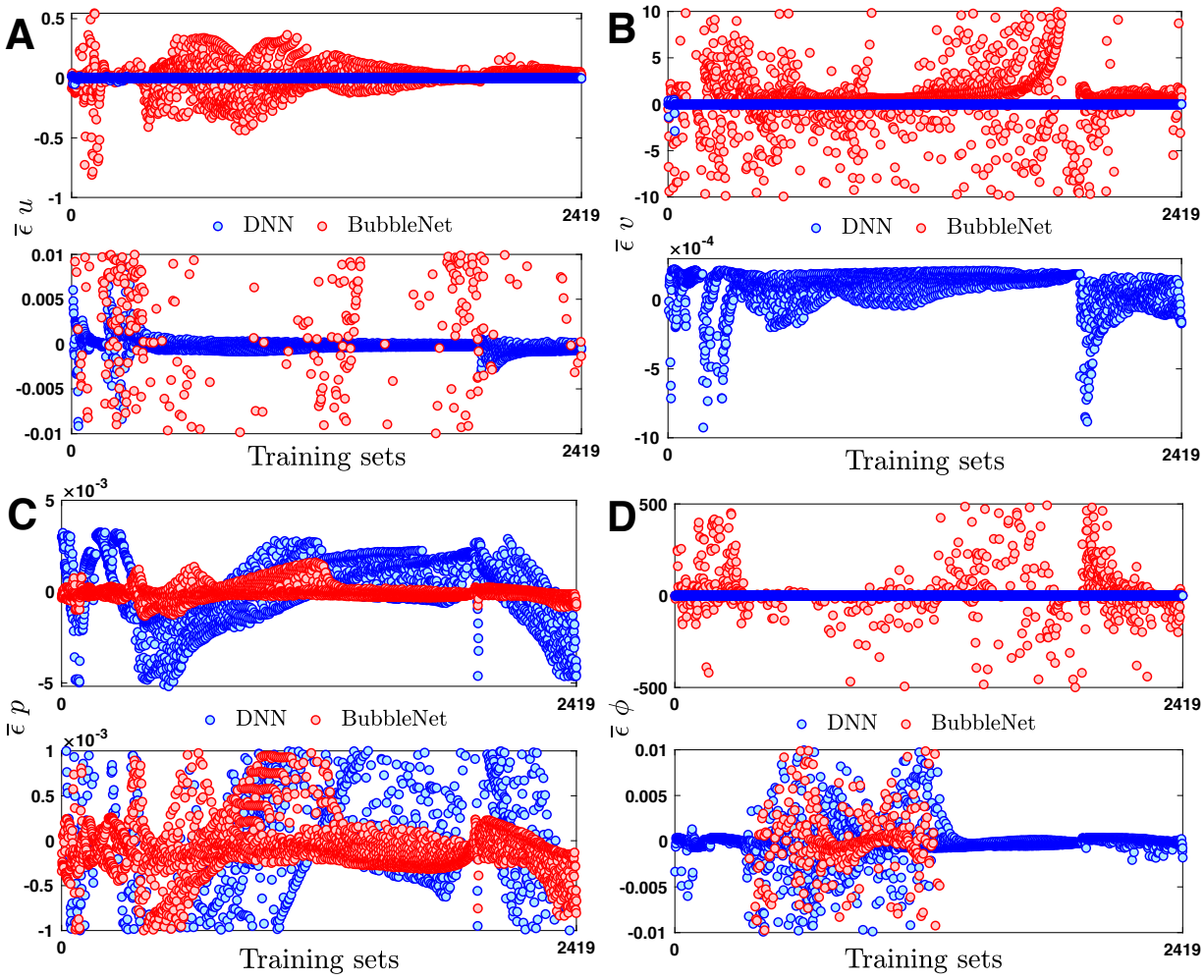

Figure 18. The relative error $\bar{\epsilon}$ for both the DNN and BubbleNet predicted physics field u, v, p, ϕ, where the blue part indicates DNN's training results and red part indicates BubbleNet's training results.

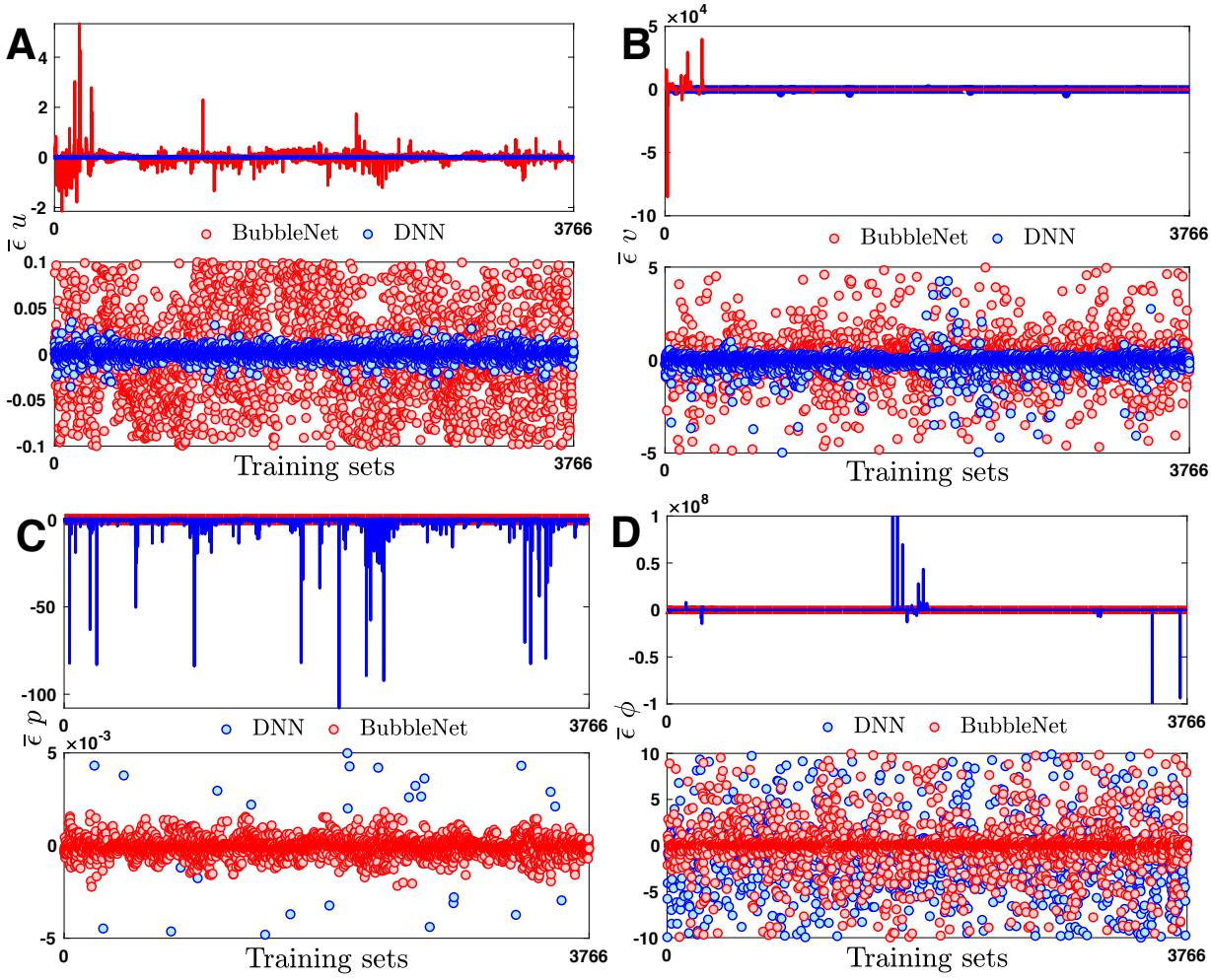

Figure 19. The relative error $\bar{\epsilon}$ for both the DNN and BubbleNet predicted physics field u, v, p, ϕ, where the blue part indicates DNN's training results and red part indicates BubbleNet's training results.

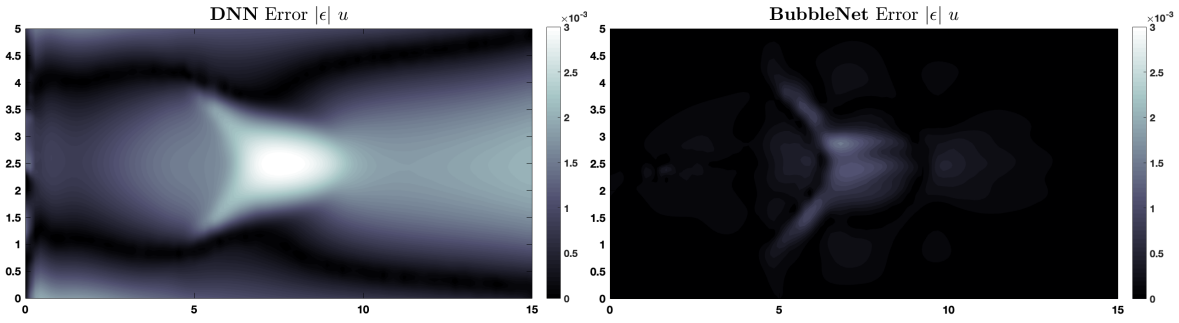

Figure 20. The absolute error $|\epsilon|$ comparison for DNN and BubbleNet predicted velocity field u for the single bubble flow case.

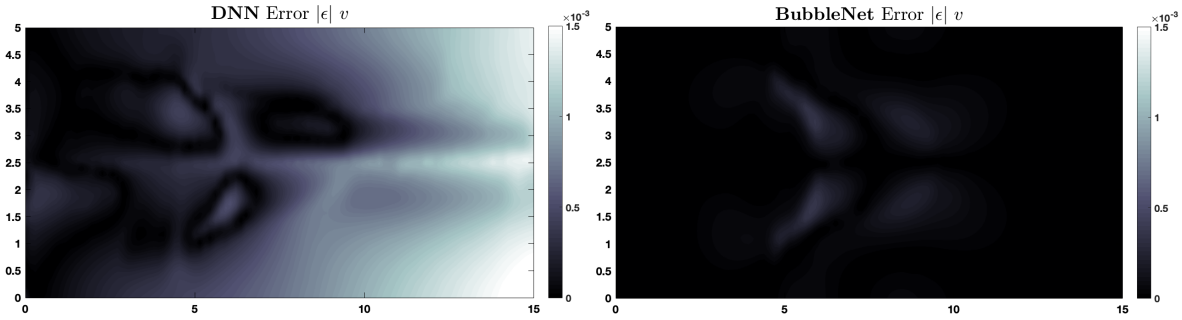

Figure 21. The absolute error $|\epsilon|$ comparison for DNN and BubbleNet predicted velocity field v for the single bubble flow case.

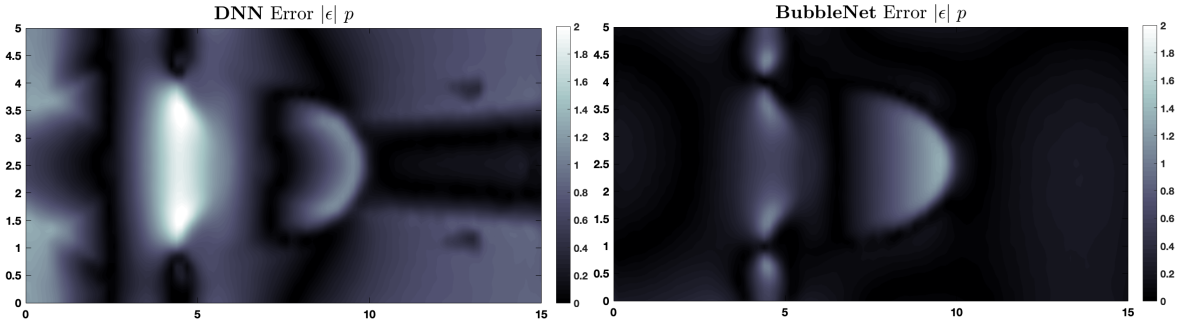

Figure 22. The absolute error $|\epsilon|$ comparison for DNN and BubbleNet predicted pressure field p for the single bubble flow case.

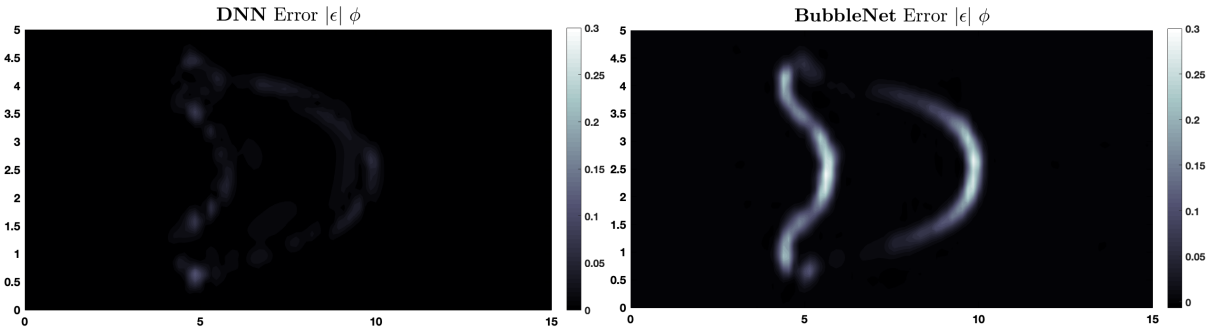

Figure 23. The absolute error $|\epsilon|$ comparison for DNN and BubbleNet predicted phase field ϕ for the single bubble flow case.

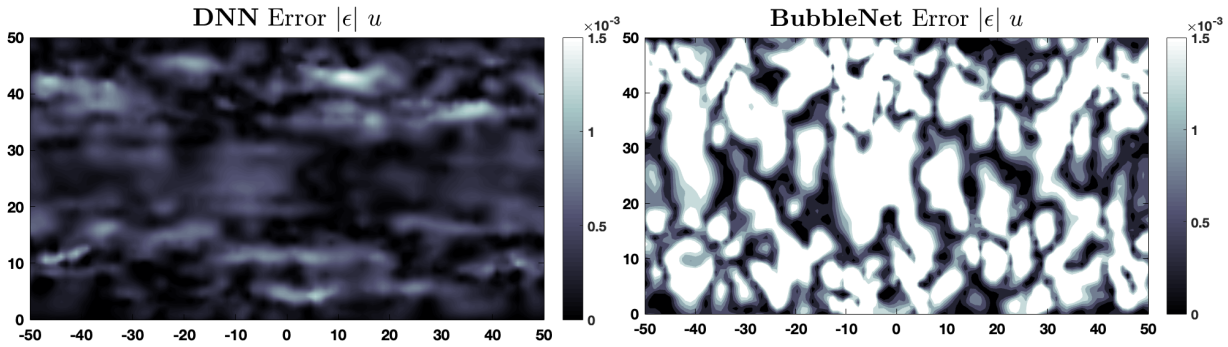

Figure 24. The absolute error $|\epsilon|$ comparison for DNN and BubbleNet predicted velocity field u for the multiple bubbles flow case.

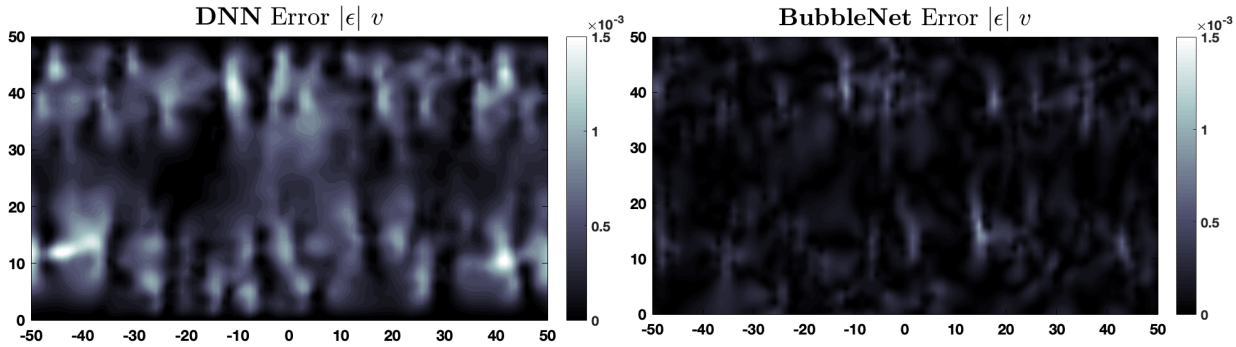

Figure 25. The absolute error $|\epsilon|$ comparison for DNN and BubbleNet predicted velocity field v for the multiple bubbles flow case.

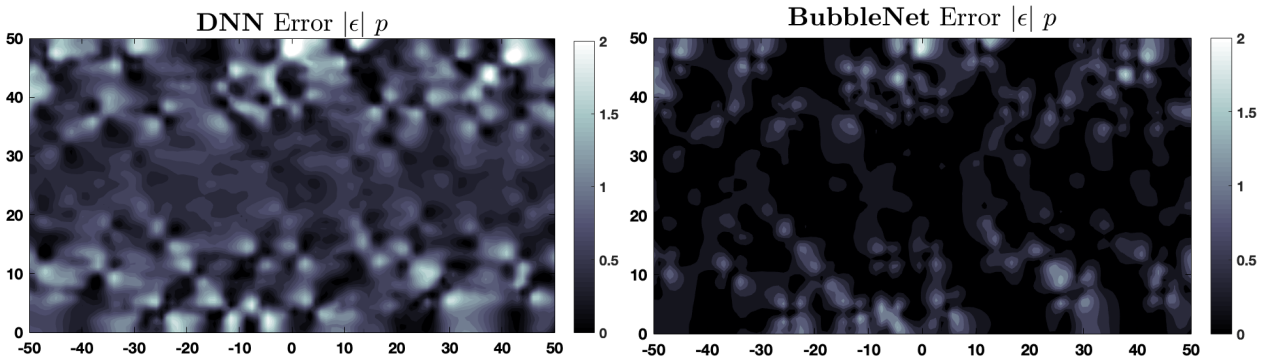

Figure 26. The absolute error $|\epsilon|$ comparison for DNN and BubbleNet predicted pressure field p for the multiple bubbles flow case.

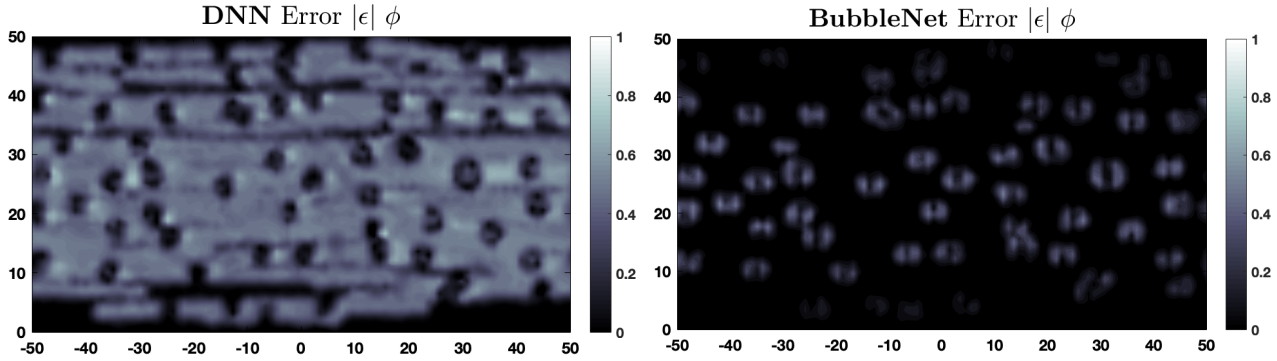

Figure 27. The absolute error $|\epsilon|$ comparison for DNN and BubbleNet predicted phase field ϕ for the multiple bubbles flow case.

bubbly flow from two 2D simulation cases, the single bubble flow, and the multiple bubbles' flow, using the time-dependent level set algorithms for multiphase flow. The single bubble case results indicate that a single bubble front interface will flow outwards as an increasing parabolic shape while the tale gets flow inwards till two attached smaller bubbles get ruptured from the main bubble. The multiple bubbles case indicates grouped bubbles collide to be fused or ruptured as time increases. For both cases, we analyze the liquid-gas ratio during the whole computation to ensure correct calculation of simulations and estimate the possible errors.

We introduce our approach of using DNNs to predict the physics fields of bubbly flows, constructing four sub nets for predicting the velocities, pressure, and level set phase functions u, v, p, ϕ. We, therefore, propose our own deep learning framework, the BubbleNet, including a DNN with three sub nets for predicting different physics fields, specifically ψ, p, ϕ; the physics-informed part, with the fluid continuum condition, encoded within; the time discretized normalizer (TDN), an algorithm to normalize field data per time step before training. We demonstrate the effectiveness of BubbleNet from training the data obtained from the two bubbly flow cases. Predictions indicate our BubbleNet can predict the physics more accurately than DNNs. From estimating the errors we conclude BubbleNet does not evidently outperform DNNs with regards to the training relative errors $\bar{\epsilon}$, yet exhibits higher accuracy on predicting the real physics fields, considering the predictions absolute errors $|\epsilon|$. Such results indicate the effectiveness of TDN, supporting data re-scale plays an important role in NN predictions. The proposed deep learning framework can not only be applied to bubbly flow, but also to combustion, interface engineering, chemical reactions, and many related engineering and scientific fields.

6 Acknowledgments

The data and code used in this paper can be downloaded at `https://github.com/hanfengzhai/BubbleNet`. Details of this project can be viewed at `https://hanfengzhai.net/BubbleNet`.

References

[1] Mitchell, T. (1997). Machine Learning. New York: McGraw Hill. ISBN 0-07-042807-7.

[2] Silver, D., Huang, A., Maddison, C., et al. (2016) Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489.

[3] Silver, D., Schrittwieser, J., Simonyan, K., et al. (2017) Mastering the game of Go without human knowledge. *Nature* **550**, 354–359.

[4] Senior, A.W., Evans, R., Jumper, J., et al. (2020) Improved protein structure prediction using potentials from deep learning. *Nature* **577**, 706–710.

[5] Koumakis, L., Kanterakis, A., Kartsaki, E., et a;. (2016). MinePath: Mining for Phenotype Differential Sub-paths in Molecular Pathways. *PLoS Computational Biology* **12**(11): e1005187.

[6] Waring, J., Lindvall, J., and Umeton, R. (2020) Automated machine learning: Review of the state-of-the-art and opportunities for healthcare. *Artificial Intelligence in Medicine* **104**, 101822.

[7] Jo, T., Nho, K., and Saykin, A.J. (2019) Deep Learning in Alzheimer's Disease: Diagnostic Classification and Prognostic Prediction Using Neuroimaging Data. *Front. Aging Neurosci.* **11**, 40, 220.

[8] Woldaregay, A.Z., Årsand, E., Walderhaug, S., et al. (2019) Data-driven modeling and prediction of blood glucose dynamics: Machine learning applications in type 1 diabetes. *Artificial Intelligence in Medicine* **98**, 109-134.

[9] Harmon, S.A., Sanford, T.H., Xu, S., et al. (2020) Artificial intelligence for the detection of COVID-19 pneumonia on chest CT using multinational datasets. *Nat. Commun.* **11**, 4080.

[10] Chassagnon, G., Vakalopoulou, M., Battistella, E., et al. (2021) AI-driven quantification, staging and outcome prediction of COVID-19 pneumonia. *Medical Image Analysis* **67**, 101860.

[11] Benke, K., and Benke, G. (2018) Artificial Intelligence and Big Data in Public Health. *International Journal of Environmental Research and Public Health.* **15**(12):2796.

[12] Panch, T., Pearson-Stuttard, and J., Greaves, F., et al. (2019) Artificial intelligence: opportunities and risks for public health. *The Lancet Digital Health* **1**, 1, 13-14.

[13] Li, Y., Shang, K., Bian, W., et al. (2020) Prediction of disease progression in patients with COVID-19 by artificial intelligence assisted lesion quantification. *Sci. Rep.* **10**, 22083.

[14] Punn, N.S., Sonbhadra, S.K., and Agarwal, S. (2020) COVID-19 Epidemic Analysis using Machine Learning and Deep Learning Algorithms. `medRxiv:10.1101/2020.04.08.20057679`.

[15] Corbetta, A., Menkovski, V., Benzi, R., et al. (2021) Deep learning velocity signals allow quantifying turbulence intensity. *Science Advances*. **7**: 12.

[16] Bieker, K., Peitz, S., Brunton, S.L., et al. Deep model predictive flow control with limited sensor data and online learning. *Theor. Comput. Fluid Dyn.* **34**, 577–591 (2020).

[17] Edalatifar, M., Tavakoli, M., Ghalambaz, M., et al. (2020). Using deep learning to learn physics of conduction heat transfer. Journal of Thermal Analysis and Calorimetry. 10.1007/s10973-020-09875-6.

[18] Wang, Z., Song, C., and Chen, T. (2017) Deep learning based monitoring of furnace combustion state and measurement of heat release rate, Energy. **131**: 106-112.

[19] Brunton, S.L., Proctor, J.L., and Kutz, J.N. (2016) Sparse identification of nonlinear dynamics. *PNAS* **113** (15): 3932-3937;

[20] Rudy, S.H., Brunton, S.L., and Proctor, J.L., et al. (2017) Data-driven discovery of partial differential equations. *Science Advances* **3**, 4.

[21] Li, Z., Kovachki, N., Azizzadenesheli, K., et al. (2020) Fourier Neural Operator for Parametric Partial Differential Equations. *arXiv*. `arXiv:2010.08895`.

[22] Li, Z., Kovachki, N., Azizzadenesheli, K., et al. (2020) Neural Operator: Graph Kernel Network for Partial Differential Equations. *arXiv*. `arXiv:2003.03485`.

[23] Jiang, C., Esmaeilzadeh, S., Azizzadenesheli, K., et al. (2020) MeshfreeFlowNet: A Physics-Constrained Deep Continuous Space-Time Super-Resolution Framework. *arXiv*. `arXiv:2005.01463`.

[24] Zhang, L., Han, J., Wang, H., et al. (2018) Deep Potential Molecular Dynamics: A Scalable Model with the Accuracy of Quantum Mechanics. *Phys. Rev. Lett.* **120**, 143001.

[25] Wang, H., Zhang, L., Han, J., et al. (2018) DeePMD-kit: A deep learning package for many-body potential energy representation and molecular dynamics. *Comp. Phys. Comm.*. **228**: 178-184.

[26] Zhang, L., Han, J., Wang, H., et al. (2018) DeePCG: Constructing coarse-grained models via deep neural networks. *J. Chem. Phys.* **149**, 034101.

[27] Raissi, M., Perdikaris, P., and Karniadakis, G.E. (2019) Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics* **378**, 686-707.

[28] Raissi, M., Perdikaris, P., and Karniadakis, G.E. (2019) Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. *arXiv.* `arXiv:1711.10561`

[29] Raissi, M., Perdikaris, P., and Karniadakis, G.E. (2019) Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations. *arXiv.* `arXiv:1711.10566`

[30] Lu, L., Meng, X., Mao, Z., et al. (2020) DeepXDE: A deep learning library for solving differential equations. *arXiv.* `arXiv:1907.04502`.

[31] Lu, L., Jin, P., and Karniadakis, G.E. (2019) DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv.* `arXiv:1910.03193`.

[32] Cai, S., Wang, Z., Lu, L., et al. (2020) DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks. *arXiv.* `arXiv:2009.12935`.

[33] Mao, Z., Lu, L., Marxen, O., et al. (2020) DeepM&Mnet for hypersonics: Predicting the coupled flow and finite-rate chemistry behind a normal shock using neural-network approximation of operators. *arXiv.* `arXiv:2011.03349`.

[34] Pang, G., Lu, L., and Karniadakis, G.E. (2019) fPINNs: Fractional Physics-Informed Neural Networks. *SIAM J. Sci. Comput.* **41**(4), A2603–A2626.

[35] Jagtap, A.D., Karazmi, E., and Karniadakis, G.E. (2019) Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering.* **365**: 113028.

[36] Jagtap, A.D., and Karniadakis, G.E. (2020) Extended Physics-Informed Neural Networks (XPINNs): A Generalized Space-Time Domain Decomposition Based Deep Learning Framework for Nonlinear Partial Differential Equations. *Commun. Comput. Phys.*. **28**: 2002-2041.

[37] Meng, X., Li, Z., Zhang, D., et al. (2020) PPINN: Parareal Physics-Informed Neural Network for time-dependent PDEs. *Computer Methods in Applied Mechanics and Engineering.* **370**: 113250.

[38] Rao, C., Sun, H., and Liu, Y (2020) Physics-informed deep learning for incompressible laminar flows. *Theoretical and Applied Mechanics Letters.* **10**(3): 207-212.

[39] Cai, S., Wang, Z., Wang, S., et al. (2021) Physics-Informed Neural Networks for Heat Transfer Problems. *J. Heat Transfer.* **143**(6): 060801

[40] Wang, R., Kashinath, K., Mustafa, M., et al. (2020) Towards Physics-informed Deep Learning for Turbulent Flow Prediction. KDD '20: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. August 2020. pp 1457–1466.

[41] Mao, Z., Jagtap, D., Karniadakis, G.E., et al. (2020) Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering.* **360**: 112789.

[42] Lin, C., Li, Z., Lu, L., et al. (2020) Operator learning for predicting multiscale bubble growth dynamics. *J. Chem. Phys.* **154**, 104118.

[43] Dollet, B., van Hoeve, W., Raven, J.-P., et al. (2008) Role of the Channel Geometry on the Bubble Pinch-Off in Flow-Focusing Devices. *Phys. Rev. Lett.* **100**, 034504.

[44] Herrada, M.A., Montanero, J.M., Ferrera, C., et al. (2010) Analysis of the dripping–jetting transition in compound capillary jets. *J. Fluid Mech.* **649**, 523–536.

[45] van Hoeve, W., Dollet, B., Versluis, M., et al. (2011) Microbubble formation and pinch-off scaling exponent in flow-focusing devices. *Physics of Fluids* **23**, 092001.

[46] Vega, E. J., Acero, A. J., Montanero, J. M., et al. (2014) Production of microbubbles from axisymmetric flow focusing in the jetting regime for moderate Reynolds numbers. *Phys. Rev. E* **89**, 063012.

[47] Zhao, B., Pahlavan, A.A., Cueto-Felgueroso, L., et al. (2014) Forced Wetting Transition and Bubble Pinch-Off in a Capillary Tube. *Phys. Rev. Lett.* **120**, 084501.

[48] Miao, H., Gracewski, S.M., and Dalecki, D. (2008) Ultrasonic excitation of a bubble inside a deformable tube: Implications for ultrasonically induced hemorrhage *J. Acoust. Soc. Am.* **124**, 2374–2384.

[49] Hosseinkhah, N., Chen, H., Matula, T.J., et al. (2013) Mechanisms of microbubble–vessel interactions and induced stresses: A numerical study. *J. Acoust. Soc. Am.* **134**, 1875-1885.

[50] Hosseinkhah, N., Goertz, D.E., and Hynynen, K. (2015) Microbubbles and Blood Brain Barrier Opening: A Numerical Study on Acoustic Emissions and Wall Stress Predictions. *IEEE Transactions on Biomedical Engineering.* **62**(5): 1293–1304.

[51] Talu, E., Hettiarachchi, K., Powell, R.L., et al. (2008) Maintaining Monodispersity in a Microbubble Population Formed by Flow-Focusing. *Langmuir.* **24**(5): 1745-1749.

[52] Tenjimbayashi, M., Doi, K., and Naito, M. (2019) Microbubble flows in superwettable fluidic channels. *RSC Advances.* **9** 21220.

[53] Peyman, S.A., Abou-Saleh, R.H., McLaughlan, J.R., et al. (2012) Expanding 3D geometry for enhanced on-chip microbubble production and single step formation of liposome modified microbubbles. *Lab Chip* **12**, 4544–4552.

[54] Papadopoulou, V., Tang, M.-X., Balestra, C., et al. (2014) Circulatory bubble dynamics: From physical to biological aspects. *Advances in Colloid and Interface Science* **206**, 239–249.

[55] Kim, Y., and Park, H. (2021) Deep learning-based automated and universal bubble detection and mask extraction in complex two-phase flows. *Sci. Rep.* **11**: 8940.

[56] Rajendra, P., and Brahmajirao, V. (2021) Modeling of dynamical systems through deep learning. *Biophys. Rev.* **12**(6):1311-1320.