processes

MDPI

Article

Underwater Gas Leak Quantification by Convolutional Neural Network Using Images

Gustavo Luís Rodrigues Caldas [1,2,*], Roger Matsumoto Moreira [3] and Maurício B. de Souza, Jr. [1,2]

[1] Chemical and Biochemical Process Engineering Program, School of Chemistry, Universidade Federal do Rio de Janeiro, Rio de Janeiro 21941909, Rio de Janeiro, Brazil; mbsj@eq.ufrj.br
[2] Chemical Engineering Program, Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa em Engenharia (COPPE), Universidade Federal do Rio de Janeiro, Rio de Janeiro 21941914, Rio de Janeiro, Brazil
[3] School of Engineering, Universidade Federal Fluminense, Niterói 24210240, Rio de Janeiro, Brazil; rmmoreira@id.uff.br
* Correspondence: gustavo.caldas@eq.ufrj.br

Abstract: Exploration and production activities in deep-water oil and gas reservoirs can directly impact the surrounding ecosystems. Thus, a tool capable of measuring oil and gas leaks based on surveillance images, especially in pre-mature stages, is of great importance for ensuring safety and environmental protection. In the present work, a Convolutional Neural Network (U-Net) is applied to leak images using transfer learning and hyperparameter optimization, aiming to predict bubble diameter and flow rate. The data were extracted from a reduced model leak experiment, with a total of 77,676 frames processed, indicating a Big Data context. The results agreed with the data obtained in the laboratory: for the flow rate prediction, coefficients of determination by transfer learning and hyperparameter optimization were, respectively, 0.938 and 0.941. Therefore, this novel methodology has potential applications in the oil and gas industry, in which leaks captured by a camera are measured, supporting decision-making in the early stages and building a framework of a mitigation strategy in industrial environments.

Keywords: subsea leaks; U-Net; bubble diameter; computer vision; deep learning

check for updates

Academic Editor: Chunhui Zhao

Received: 1 December 2024
Revised: 21 December 2024
Accepted: 23 December 2024
Published: 5 January 2025

Citation: Caldas , G.L.R.; Moreira, R.M.; de Souza, M.B., Jr. Underwater Gas Leak Quantification by Convolutional Neural Network Using Images. *Processes* **2025**, *13*, 118. https://doi.org/10.3390/pr13010118

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The drilling and the exploration of wells in the subsea environment make use of pipelines to transport oil and gas. Due to corrosion, structural failure, or third-party damage, subsea pipelines are subject to material leaks [1,2]. The risks increase especially if the pipeline is in a decommissioning state and/or after a long time of service. The consequences can be significant to ecosystems, which also incur financial losses for all stakeholders [3]. Therefore, monitoring tools are an essential part of a safety culture during the process operation. Several methods have been developed for detecting leakages in different domains [4]: acoustic [5–7], chemical [8], pressure [9], and others. Some of these sensors can be attached to remotely operated vehicles (ROV), introduced internally via pigging/crawling robots, or installed on seabeds.

An ROV equipped with an onboard camera can provide real-time filming for monitoring purposes during the inspection of a leak suspicion or routine surveying throughout the production. Since the operation of an ROV can be costly, a fixed camera could be set permanently in strategic positions for continuous surveillance [10]. Nonetheless, in both cases, the images just provide information on whether there is a leak. One could be interested in extracting more information regarding the nature of the leakage, e.g., the flow rate or the type

of fluid leaked. This type of information will influence whether there is a need to intervene in the well. A successful intervention demands trained personnel and financial resources. Hence, for a better assessment, it is of great interest to increase the ROV's capabilities by developing a quantitative tool to support the intervention decision-making process.

In this sense, computer vision is a field dedicated to the interpretation of images by a computer that leads to actions or inferences [11]. Image processing techniques can aid the oil and gas industry in solving problems in various applications on refineries and upstream plants [12–15]. With respect to bubbly flow, Chen et al. [16] developed an algorithm for the processing of bubbles originated from high-speed image experiments and Generative Adversarial Networks (GANs); they coupled background subtraction, binarization (using Otsu's method), boundary extraction, and arc segmentation.

Regarding subsea leaks, Wang and Socolofsky [17] developed a stereoscopic image system coupled to an ROV to study natural seeps' bubbles and droplets in the deep-sea waters of the Gulf of Mexico. In this case, the laboratory apparatus consisted of air bubbles released in a 1 m tap water tank at 20 °C and atmospheric pressure. According to the authors, the air is considered a good representation of methane fluid dynamics for image testing under environmental conditions. In short, they employed traditional image-processing techniques, such as a Sobel gradient filter, canny edge detection, thresholding, and a watershed algorithm. Important leak features were calculated after all bubbles were identified, such as the bubble size distribution and the flow rate. The volumetric gas flow rate was calculated based on the average total volume of bubbles in a frame and the average bubble displacement time in the vertical direction. After an initial application of this method in natural seeps in the Gulf of Mexico, Wang et al. [18] and Razaz et al. [19] expanded this method to other observations in the same ocean basin. In another study regarding natural seeps, but in the South China Sea, Di et al. [20] calculated the bubble diameter distribution and release rate employing a semi-automated algorithm that relied on manually determined thresholds. She et al. [21] proposed the gas flow quantification on the marine environment with the aid of median filters, canny edge detection, and ellipse fitting. However, as Liu et al. [22] stated, this traditional image processing approach cannot deal with the variety and complexity of different scenarios. The main advantage of learning-based methods over traditional image-processing techniques is their ability to leverage data. This enables better generalization across varying conditions, reducing the need for specific, manually crafted procedures for each situation and minimizing human intervention. For instance, in the work of Wang and Socolofsky [17], the image analysis sequence performed depended upon the bubble flow type.

Therefore, despite the widespread applications of the traditional image processing approaches, the computer vision field can also rely on new tools arising from Artificial Intelligence (AI), which are driven by learning from data accumulation and have a superior capacity to meet stringent demands (accuracy, application transfer, efficiency) [22]. Convolutional Neural Networks (CNNs), as a subarea of AI, have gained notorious attention in the scientific community for this purpose [23–25], since they can efficiently handle data organized in tensors, including images [26–29], time-domain information [30–32], radars [33], pressure signal [34], acoustic information collected from sonars [7], and flow information [35,36], together with Computational Fluid Dynamics (CFD) [37–41]. Furthermore, due to the scarcity of data or model complexity, transfer learning may be employed for time-saving and model enhancement purposes, e.g., training a CNN architecture on pre-trained weights. Transfer learning was performed on other chemical engineering applications, for instance, in fault detection [42,43], quantum chemistry [44], and thermodynamics [45]. Another common question concerns the architecture hyperparameters, which are defined a priori of the training. Neural networks are adaptable to handle various types

of data. However, tuning and optimizing different hyperparameters, which is essential for achieving optimal performance, can be time-intensive. The goal of hyperparameter optimization methods is to minimize the generalization error predicted by a machine learning model across different potential configurations. Sadoune et al. [46] combined Bayesian optimization with a tree-structured Parzen estimator to forecast biogas production using real data from wastewater treatment plants.

Hence, CNNs can be applied to study fluid flow, which includes flow regime identification, detection of two-phase flow, and the indication of leaks. Concerning identifying bubbles in a water column, Caldas et al. [39] employed CNNs to detect bubbles of simulated data. The algorithm consisted of a classification problem, yielding 99% of accuracy. Poletaev et al. [35] applied CNNs to identify patterns in a two-phase bubbly jet. They generated 30,000 synthetic images comprising a broad range of overlapping, blurriness, and volume gas fractions. Their methodology is described as follows: (1) classify the probability of bubble presence in the image after comparing 60 CNNs' different configurations; (2) calculate the bubble center employing an algorithm; (3) separate bubbles from the noise background using a denoising autoencoder (a multilayer perceptron network), which facilitates the determination of geometric parameters. The NNs were validated against experimental data. Ilonen et al. [36] compared several bubble-detecting methods, among them CNNs (whose accuracy was superior to the others), in an oxygen delignification process for a pulp production line. The authors also described a bubble size estimator that does not require prior bubble detection. Khaydarov et al. [47] suggested that the bubble size distribution in a stirred bioreactor can be a target variable for enabling smart equipment because it can indicate mixing quality and gas holdup, thus saving operational time. For this purpose, they used a You Only Look Once (YOLO) network to detect air bubbles. The data could be used to complete a CFD model. Another work that used YOLO was made by Xiang et al. [48], who applied it as a bubble detector in artificially generated bubbly flows. Rutkowski et al. [49] also carried out image processing by the YOLO network and Faster R-CNN in microfluidic droplets.

Some studies were limited to detecting the plume as a whole, not individual bubbles. Shi et al. [26] also applied Faster R-CNN to leak detection in ethane cracker plants. Similarly, for subsea gas leak monitoring purposes, YOLO and Faster R-CNN networks were employed by Zhu et al. [50] and Hu et al. [51]. The mentioned authors did not proceed with a quantification of the leaks, focusing on detection and classification and finding their location.

Nevertheless, all cited applications focused solely on object detection, which, in general, (1) demands intensive human labor on manually labeling the dataset during the training phase; (2) is only a task to predict if some object is present in the image, creating a bounding box close to the object; and (3) consequently, does not dictate the precise object boundaries, requiring further work on region delimitation and contour definition for precise bubble size calculation. Therefore, a more simple and promising alternative is to use semantic segmentation instead of object detection. Semantic segmentation is a classification process on the pixel level, assigning each one a label. Thus, it predicts if some pixel belongs to the category, e.g., to an object or the background. It can be a strategy for multi-phase flow processes in chemical engineering that require knowledge of particle diameter distribution. One of the CNNs developed for this task is the U-Net created by Ronneberger et al. [52] for cell identification in biomedical applications, a field where computer vision is well established. Other networks commonly used for semantic segmentation are DeepLab series, SegNet, and PSPNet [22]. Schäfer et al. [53] employed U-Net to segment oil droplets dispersed in a water column. The post-processing phase enabled a straightforward calculation of the particle size distribution. Furthermore, Caldas

et al. [40] employed U-Net for measuring bubble diameter using synthetic images generated by Computational Fluid Dynamics in different conditions of simulated leaks. Cerqueira et al. [54] generated a Particle Tracking Velocimetry (PTV) algorithm using U-Net for identifying and highlighting drop contours of oil in water on a centrifugal pump impeller. Bergau et al. [55] proposed quantifying small methane leaks employing a 3D CNN model but using artificial laser spectroscopy, not real images.

In this work, we propose an innovative methodology that contributes to the literature as follows:

1. We demonstrate an easy way to automatically create a ground truth dataset (77,676 images) for CNN training purposes;
2. We show how to perform hyperparameter optimization on U-Net in Big Data situations;
3. We apply the U-Net network for gas flow rate quantification and bubble diameter calculation in a water column employing a laboratory model of subsea leaks. In this step, a comparison between transfer learning and hyperparameter optimization is carried out. In the former, the network architecture and the weights from the step based on the synthetic images serve as the starting point. In the latter, a hyperparameter optimization on the U-Net is developed to find the best architecture without any prior knowledge. In this step, the training is performed in a Big Data context. The results are validated against the experimental values obtained in the laboratory.

2. Theory

This section aims to briefly overview the techniques used in this work and already cited in Section 1. First, we will shed light on more traditional image techniques in Section 2.1. Then, Machine Learning and more specifically Convolutional Neural Networks, will be discussed in Section 2.2.

2.1. Image Processing Techniques

2.1.1. Thresholding

Thresholding is a segmentation technique in which the classification is based on the histogram of a property of an image, which is usually its grayscale. It aims to find a value that can divide the image into a group with less than this value and another one with greater than or equal to this value. It can be viewed as a statistical decision process in which the probability of belonging to a specific class is searched. Hence, the histogram of the image property is treated as an approximation of its probability density function [11].

Suppose an image whose histogram of grayscale intensity $f(x, y)$ presents a bimodal distribution. An example is an image with bright objects on a black background. Therefore, extracting objects in this image is straightforward by finding the value T that divides the pixels into two modes, and consequently, the background and foreground are found. An intuitive approach is to determine a threshold T whose output $g(x, y)$ is classified into [11]

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) \geq T \\ 0 & \text{if } f(x,y) < T \end{cases} \quad (1)$$

The output $g(x, y)$ is called a binarized image whose background is denoted as zero, and the foreground is marked as one. If T is a single value valid for the entire image, thresholding is denoted as global. If T depends on the position (x, y), selected independently for a group of pixels, they are called local (or dynamic/adaptive) thresholding [11,56]. This work will discuss only the main global thresholding method: Otsu's. This method aims to maximize the interclass variance present in the histogram. In this sense, the threshold

T should give the best outcome possible between classes regarding grayscale intensity values [11]. The method is described below, as introduced by Otsu [57].

Supposing that a threshold $T(k) = k$ $(0 < k < L-1)$ is selected so that the image is divided into two classes, C_1 and C_2, C_1 consists of pixels with the levels $[0, k]$, and C_2 consists of pixels with the level $[k, L-1]$. Using the threshold, the probabilities of each class's occurrence are β_1 and β_2, respectively; the mean values of each class are calculated by μ_1 and μ_2, respectively, and the class variances are σ_1^2 and σ_2^2, respectively. In this sense, σ_T^2 is the global variance, and μ_T^2 is the mean value for the whole image histogram. Adopting a discriminant criterion $\eta = \sigma_B^2/\sigma_T^2$ that evaluates the goodness of the threshold at the level k, where the subscripts BC and WC represent the between-class and within-class variances, respectively:

$$\sigma_{WC}^2 = \beta_1\sigma_1^2 + \beta_2\sigma_2^2 \tag{2a}$$

$$\sigma_{BC}^2 = \beta_1(\mu_1 - \mu_T)^2 + \beta_2(\mu_2 - \mu_T)^2 \tag{2b}$$

$$\sigma_T^2 = \sigma_{WC}^2 + \sigma_{BC}^2 \tag{2c}$$

Therefore, the method aims to maximize the between-class variance σ_{BC}^2, which is a measure of the dissociation between classes. In this sense, σ_1^2 and σ_2^2 should be the farthest from each other. Since σ_T is a constant $(\sigma_T > 0)$, η is also maximized. Thus, the algorithm should search for the optimum threshold k^*, such that [11,57]

$$\eta = \frac{[\mu_T\beta(k) - \mu(k)]^2}{\beta(k)[1-\beta(k)]} \tag{3a}$$

$$\sigma_{BC}^2(k^*) = \max_{0<k<L-1} \sigma_{BC}^2(k) \tag{3b}$$

2.1.2. Sobel Gradient Filter

The main use of the Sobel gradient filter is to find edges on an image by computing their derivatives. The first-order derivatives indicate an edge because maximum or minimum points of intensity represent the sudden change in pixels. The gradient vector of an image reads:

$$\nabla f(x,y) = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x}(x,y) \\ \frac{\partial f}{\partial y}(x,y) \end{bmatrix} \tag{4a}$$

The magnitude of the gradient vector of an image is called the gradient image

$$M(x,y) = \sqrt{{G_x}^2 + {G_y}^2} \tag{4b}$$

Since images have a discrete representation, it is not possible to proceed as a continuous differentiation. Therefore, the Sobel operators try to approximate the computation of the derivatives by convolving the Sobel kernel with the image, represented as a two-dimensional array A (where $*$ is the convolution operator):

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * A \tag{4c}$$

$$G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A \tag{4d}$$

Other types of gradient operators are Roberts and Prewitt [11,56].

2.1.3. Fast Fourier Transform

The Fourier transform $\mathfrak{F}$ (FT) for an image $f(x,y)$ with height H and width W is given by Equation (5). The Fourier transform computes the magnitude and phase shift response at each frequency ξ and repeats this procedure all over the domain [58].

$$\mathfrak{F}\{f(x,y)\} = F(\omega,v) = \frac{1}{HW}\sum_{y=0}^{H-1}\sum_{x=0}^{W-1} f(x,y)e^{-j2\pi(\frac{\omega x}{W}+\frac{vy}{H})}, \quad x = 0,1,2\ldots W-1, y = 0,1,2\ldots H-1. \tag{5}$$

ω and v are the frequency indices in the horizontal and vertical directions, respectively. The component $F(0,0)$ represents the average intensity of the images, located at the center of the frequency domain representation. Higher frequencies are located away from the center, near the edges. The discrete Fourier transform can be represented in polar coordinates as

$$\mathfrak{F}\{f(x,y)\} = |F(\omega,v)|e^{j\phi(\omega,v)} \tag{6a}$$

where $|F(\omega,v)|$ is the Fourier spectrum and $\phi(\omega,v)$ is the phase frequency defined by

$$|F(\omega,v)| = \sqrt{Re^2(\omega,v) + Im^2(\omega,v)} \tag{6b}$$

$$\phi(\omega,v) = tan^{-1}\left[\frac{Im(\omega,v)}{Re(\omega,v)}\right] \tag{6c}$$

The use of Fourier Transforms for practical purposes is often infeasible due to computational costs of $O(HW)^2$. To address this issue, the Fast Fourier Transform (FFT) algorithm was discovered, reducing the operations to $HW\log_2(HW)$, which led to the development of image processing as it is known today. One fundamental property of FT that made it possible to implement is the separability of the 2D FTs. For more details about the algorithm, please consult Gonzalez and Woods [11].

2.2. Machine Learning Techniques

Artificial Intelligence (AI) is the capacity of a computer system to display cognition. AI research has focused on achieving a certain goal by a computer with the best possible outcome, similar to what is carried out in optimization theory, which minimizes a cost function. The AI subfield dedicated to designing systems that learn based on experience—without being explicitly programmed—is called Machine Learning (ML) [59]. The strategy lies in discovering underlying relationships based on data acquisition.

Artificial neural networks (ANNs) are part of the ML field. A recurrent definition of ANNs found in the literature is that the networks mimic the functioning of the biological brain, in which the connections between neurons—axons and dendrites—are arranged in a massively parallel way and stimulated by synaptic weights, which constitute the learning-driving force [60]. In a multilayered ANN, the information flows through connections between inputs and outputs, which are the nodes (neurons) of the ANN organized in layers. There are other traditional machine learning approaches, such as support vector machines. They are generally more suitable for simpler problems with limited amounts of data. In contrast, the application presented here involves real, noisy data, and the problem of leakage itself can be considered complex. Therefore, deep neural networks are expected to deliver superior performance.

2.2.1. Convolutional Neural Networks

A type of neural network is the Convolutional Neural Network (CNN). CNN became one of the most popular networks for deep learning. Its popularity is due to its ability to process the data in the form of tensors (multidimensional arrays), such as signals and language (1-D), images and audio (2D), and videos (3D) [25].

Figure 1 shows an overview of the architecture. The classical Convolutional Neural Networks [24] comprise convolutional layers, pooling layers, and a fully connected layer. It is a type of feed-forward neural network. The convolution layer extracts important characteristics from input into a feature map by using a kernel (or filter). In a convolutional layer l, the input I from layer $(l-1)$ is processed by several kernels K of size $k \times k$. A single convolution operation at the point $(u+i, v+j)$ involves multiplication of $I_{u+i,v+j}$ by $K(i,j)$ as stated in Equation (7) [61,62]:

$$O_{u,v}^{(l)} = \theta\left(\sum_{i=0}^{k-1}\sum_{j=0}^{k-1} I_{u+i,v+j}^{(l-1)} \cdot K(i,j) + b^{(l)}\right) \; 0 < u \leq H, 0 < v \leq W \tag{7}$$

where H and W are the input height and width, respectively; b is the bias; θ is an activation function, and $O_{u,v}$ is the output. The feature map size is reduced by using a smaller kernel than matrix [61], leading to sparse connectivity, which helps save considerable computational resources. In turn, traditional multilayer perceptrons go through a full matrix multiplication operation between patterns and weights, which could slow down the training process. The number of kernels and their shape are hyperparameters, and each kernel element is a neuron activated by a function defined by the user.

The subsequent layer of a CNN evaluates nearby feature maps by using a summary statistic function of them (average or maximum); this is referred to as pooling. As an effect, dimensionality data are reduced, lessening possible overfitting and memory requirements. It performs a type of downsampling operation, reducing the size and aggregating the information forwarded in the subsequent layers. This property can reduce computational costs. They are commonly followed by a dropout layer that randomly deactivates neurons to prevent overfitting. Several convolutional and pooling layers can be arranged together to extract features.

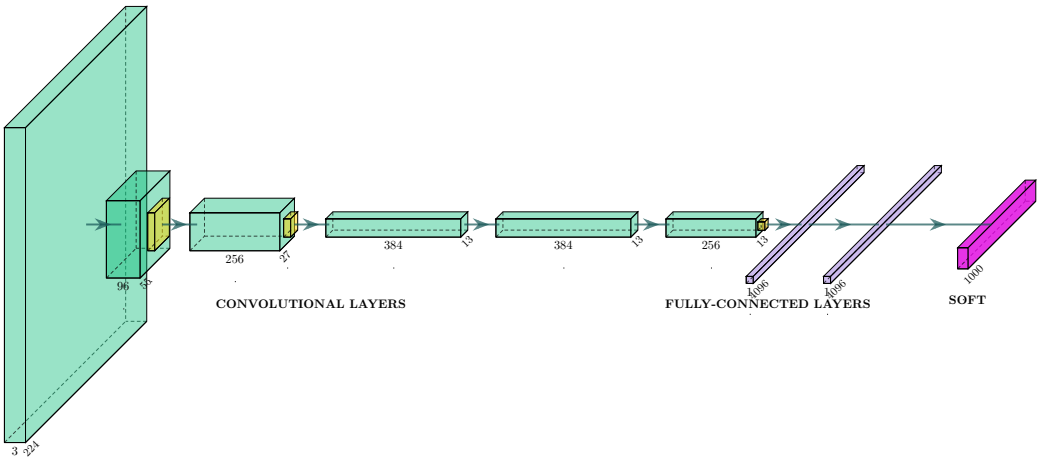


Figure 1. An example of a Convolutional Neural Network: the Alexnet. The channels and dimension of the input data are described in the image.

These first CNNs were developed in a sense to be invariant to location, i.e., if a motif appears in a particular part of an image, it could appear anywhere [25]. Since the focus was on pattern recognition, this approach can save computational costs and speed up the classification process. However, in image segmentation, more specifically in semantic

segmentation, in which the classification is performed pixel-wise, the concern is spatial localization. In order to capture both context and spatial information, Ronneberger et al. [52] developed the U-Net architecture, composed of two parts: a contracting and an expanding path. The contracting path is responsible for the image downsampling; i.e., the feature map sizes are successfully reduced in the subsequent blocks at the cost of expanding the number of filters. The contracting path is organized in blocks of convolutional and pooling layers, which captures the context (for instance, "do we have a bubble present here?"). After that, it is then followed by blocks of deconvolutional layers that perform the inverse convolutional operation and more convolutional layers to maintain the context obtained. These blocks and the skipped connections form the expanding path that reconstructs the feature map to its original size and regains space information. Thus, the expanding part carries out an upsampling process. The skipped connections are copy layers transmitting data from the downsampling section to a twin block in the upsampling one [52,63,64]. At the end of the network, there is another convolutional layer whose output gives a probability tensor of the same image dimension. The value given by a sigmoid activation function (or a softmax if multilabel segmentation) predicts the likelihood of belonging to some class. The U-Net structure is shown in Figure 2.

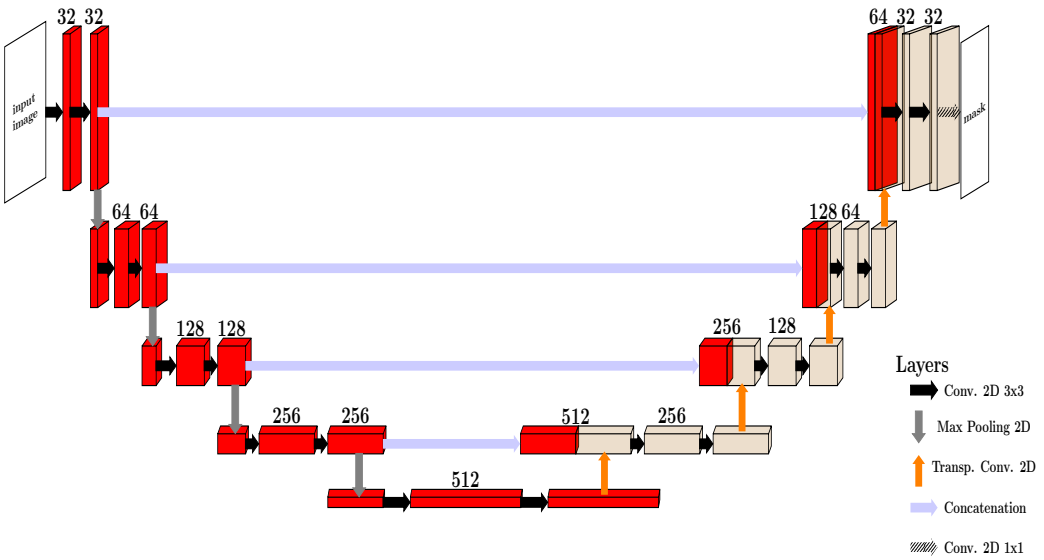


Figure 2. An example of a U-Net architecture with the number of channels described.

Batch normalization is a generalized technique applied in the Machine Learning community to improve training. According to the original paper [65], batch normalization aims to stabilize the input changes introduced by updating the preceding layers' parameters, also known as internal covariant shift (ICS). In each layer input, the ICS is minimized by setting the batch distributions' two moments (mean and standard deviation) to one and zero, respectively. In turn, Santurkar et al. [66] argue that batch normalization smooths the optimization landscape by making the gradients more predictable and well behaved.

Another common strategy employed in ML is dropout. It aims to reduce overfitting by randomly deactivating neurons and their connections, temporarily removing them from the network and making the neural network thinner. According to Srivastava et al. [67], this feature enables a neural network to be more robust by preventing co-dependence on nearby connections and to be more exposed to a diversified type of input.

2.2.2. Hyperparameter Optimization

A topic of discussion in ANNs is network hyperparameter tuning. Hyperparameters are the parameters defined a priori from training (in contrast with the weights, which are

optimized during training), and they directly affect the performance and the accuracy of the training results. Therefore, the hyperparameter optimization (HPO) problem is composed of several hyperparameters, such as neural network's architecture, learning rate, batch size, and momentum [68,69]. Yet, it can be a complex problem to solve due to the costly computation (for large/complex datasets), complex configuration space (comprised of continuous, categorical, and conditional hyperparameters), lack of smoothness, and a loss gradient function [70].

Several methods were developed to tackle this issue. One of them is grid search. Grid search is the most basic HPO approach, in which the complete combination of hyperparameters is trained, and the best one is elected based on criteria. However, it can be computationally unfeasible due to the number of combinations to be evaluated. It suffers from the curse of dimensionality; i.e., the number of objective functions to be computed increases exponentially with the search space, which leads to an unfeasible amount of data available for a reliable result. A second approach is to use random search, i.e., choosing randomly hyperparameters based on their importance according to a budget B [69,70]. According to Bergstra and Bengio [71], random search has a low effective dimensionality because the HPO objective function is more sensitive to changes in some dimensions than others.

Bayesian optimization (BO) is an HPO framework that evaluates a black-box function (model-based) to find an optimum set of hyperparameters [69,70]. The optimization goal can be described as searching for the global maxima at the sampling point of an unknown objective function ψ:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\chi} \psi(\mathbf{x}) \tag{8}$$

where χ represents the search space, and $\mathbf{x}^*$ the vector space of the hyperparameter inputs part of χ. Bayesian optimization is based on Bayes' theorem, which states that the posterior probability $P(M|E)$ of a model M given an event E is proportional to the prior probability $P(M)$ and the likelihood $P(E|M)$ of E given M:

$$P(M|E) \propto P(E|M)P(M) \tag{9}$$

which is also applicable to BO: it employs some prior information of the function ψ and sample points to evaluate the posterior of ψ. This posterior information guides where $\psi(x)$ is maximized, driving the next optimization step based on a criterion, an acquisition function. Thus, Bayesian optimization consists of two components: a surrogate model fitting all observations to the unknown function, and an acquisition function that decides the region to evaluate next, determining the utility of each candidate, taking into account the exploration and exploitation trade-off [69,70,72].

The most common surrogate model employs Gaussian kernels as prior distribution because of their simplicity and their capabilities to deal with uncertainty. For the acquisition function, the expected improvement function is usually employed [46,69,70,72]. Alternatively to BO, Tree–Parzen Estimator (TPE) models $P(M)$ and $P(E|M)$ are direct [73,74].

A major disadvantage of BO is that it does not stop unpromising trials from occurring, which, in turn, can be costly from a computational point of view. In this regard, hyperband is an extension of the successive halving method (and a variation of random search), which is focused on allocating a budget B among the N hyperparameter configurations. First, it evaluates all configurations with a B/N budget for each, then discards the worst half and allocates more resources to promising candidates until one single configuration is left with the maximum budget. Since Bayesian optimization considers prior experiments, and hyperband considers early stopping to save computational time, recently, Falkner et al. [75] integrated both Bayesian optimization and bandit-based methods into a single one: BOHB.

2.2.3. Transfer Learning

Transfer learning (TL) is a valuable technique for transferring knowledge across domains, particularly useful when dealing with small datasets, complex hyperparameter tuning, and time-consuming training processes. TL involves using a pre-trained model, one that has already been trained on a large dataset, and adapting it to a new, but related, task. This approach is made by transferring the knowledge captured in the model's weights and biases to the new dataset, which helps in generalizing predictions effectively [43].

In chemical engineering, TL can be beneficial for tasks such as scaling up processes or modeling similar types of equipment. For example, a model trained on data from one type of reactor can be adapted to predict performance in a different but similar reactor type, leveraging previously learned patterns to improve accuracy and reduce training time [76].

3. Materials and Methods

In this work, a transfer learning approach is employed, in which the weights trained in Caldas et al. [40] were used as an initial guess for training real images. The training conducted with synthetic images from CFD can be found in Caldas et al. [39,40]. Section 3.1 describes the experimental procedure, Section 3.2 details the techniques for generating data to train the neural networks, and Section 3.3 explains the CNN training method for real images. Figure 3 provides an overview of the methodology employed in this work.

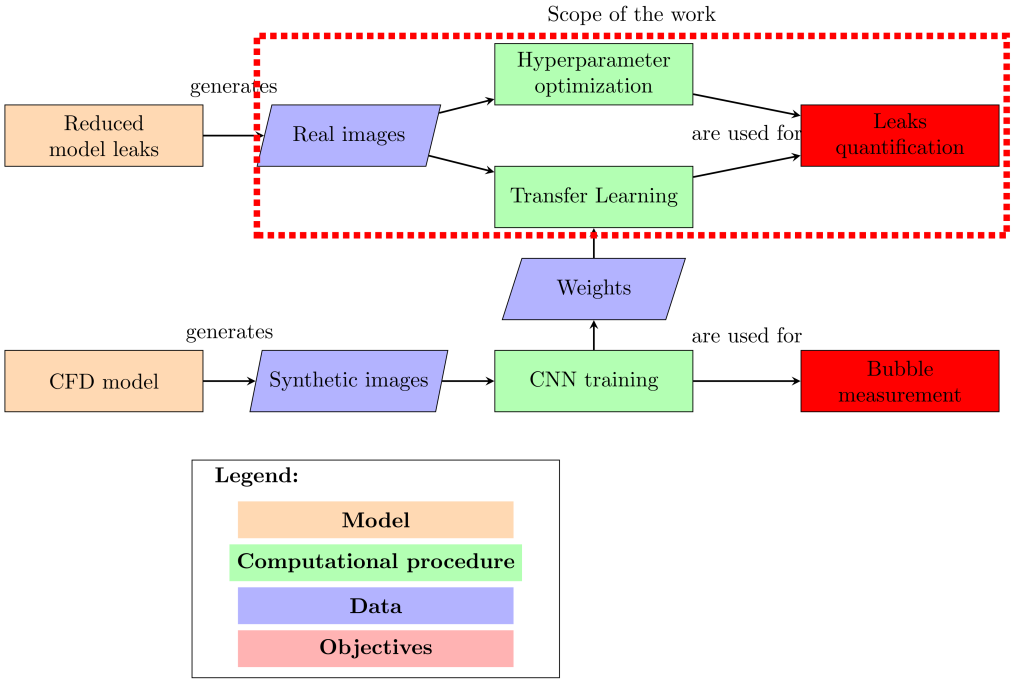


Figure 3. Summary of the methodology.

3.1. Experimental Procedure

A laboratory or a reduced model is a prototype that reproduces the system or phenomena on a minor scale in order to test a hypothesis or a scientific proposal. Nascimento et al. [77] designed an experimental apparatus to generate a reduced model of the subsea leakage, as shown in Figure 4. In this case, the gas employed is air for safety purposes. The air is blown through an orifice in a tap water glass reservoir. The blower consists of an air compressor connected to a voltage regulator and stabilizer. The output voltage sign is responsible for varying the air flow rate. The air flows from the compressor through a flow rate stabilizer and a silicone tube, and then it is injected into the water tank through an orifice. The orifice of injection has different hole diameters d_i: 0.5 mm, 1.0 mm, 2.0 mm,

and 5.0 mm. The temperature and pressure are at room conditions (20 °C and 1 atm). In the Supplementary Materials, Table S1 details the different conditions: the orifice diameters d_i, the different types of orifice from where the bubble is released, the flow rate Q set, and the two-phase fluid system (tap water/air and saltwater/air). The apparatus is described below:

- Voltage regulator 115 V manufactured by JNG, maximum 11 electric current 11 A, maximum power consumption 1.5 kVA, input voltage 115 V, electric tension variation 0–130 V;
- Voltage stabilizer manufactured by SMS, electric power 300 VA, input current 2.5 A, input and output voltage 115 V;
- Air compressor (positive displacement by diaphragm type) manufactured by Big Air, model A320, flow rate 3.5 L/min;
- Flow rate stabilizer manufactured by Cole Palmer, model GV-32908-73, flow rate range 0.5 to 50 liters per minute;
- Glass tank with 96 liters in volume. Dimensions: 40 cm × 60 cm × 40 cm.

The wall tank was coated with ethylene–vinyl acetate. According to Nascimento et al. [77], the intent was to reduce reflectivity by using a black matte color. A light source was provided inside the tank next to the orifice. The images were captured using a camera from a Xiaomi MI 9T cellphone at a rate of 240 frames per second (fps).

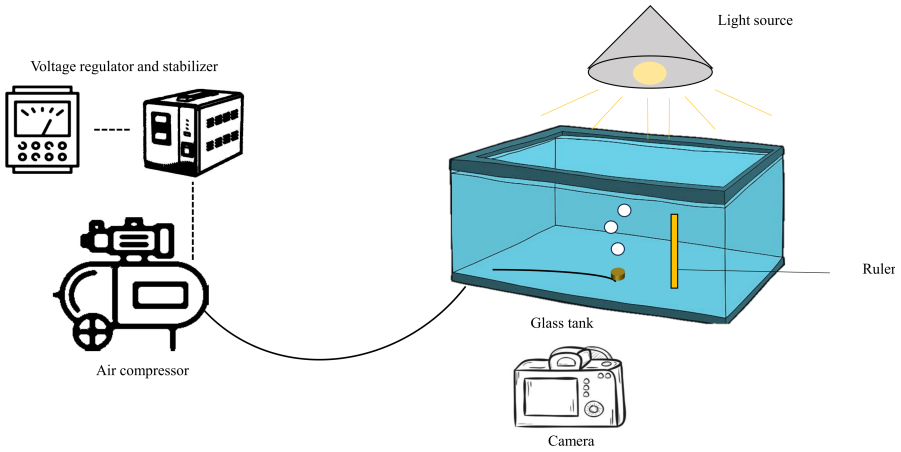


Figure 4. Schematic representation of the experimental apparatus.

The volumetric flow rate Q is obtained by counting the bubbles collected in a beaker during a specific time t_c recorded by a stopwatch, Equation (10).

$$Q = \frac{Vol_{beaker}}{\Delta t_c} \tag{10}$$

According to [77], a possible approach to estimate the bubble diameter d_b is based on the number of bubbles n_b collected within the time interval t_c in the beaker. In light of this approach, the volume of bubbles is calculated considering the bubbles ideally occupy the whole volume of the beaker Vol_{beaker} without empty spaces and overlaps, Equation (11a). The estimation also treats bubbles as perfect spheres in all dimensions, Equation (11b).

$$Vol_b = \frac{Vol_{beaker}}{n_b} \tag{11a}$$

$$d_b = \sqrt[3]{\frac{6Vol_b}{\pi}} \tag{11b}$$

3.2. Ground Truth Dataset Generation

Figure 5 shows examples of frames (for an extract of a frame for all 44 video IDs, please consult Figure S1 in the Supplementary Materials), each one representing a video ID. In the laboratory, the experimental images of the reduced model cannot be uniform in illumination, as seen in Figure 5, with either bright or dark spots present in the background. Furthermore, the light intensity does not precisely mark the liquid and gas interface. The background can sometimes be brighter than the bubble, as seen in video #8 in Figure 5a. On the contrary, from videos #10 to #33, the bubble region is clearly brighter than the background (see #14 and #19 in Figure 5b,c). In a third scenario, the background and foreground share similar pixel intensities (#34 to #44, see #37 in Figure 5d). Also, another challenge is the presence of other objects that make the segmentation less straightforward, as the back bright "line" seen in #1–#9 moves. Since the lighting differs not only across the images but especially between the different leak states (represented by a video ID), this suggests that a general method that comprises all these conditions must be developed to carry out segmentation properly. Convolutional Neural Networks are good candidates for this task.

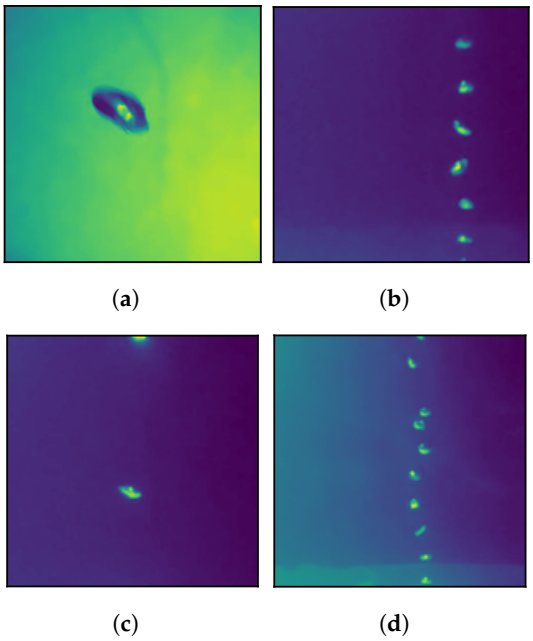


Figure 5. Examples of frames extracted by video ID: (**a**) 08, (**b**) 14, (**c**) 19, (**d**) 37.

High-quality data have to be fed for proper training of ANNs, the so-called supervised learning process. These data used as standard are otherwise known as "target" or "ground truth data". When using synthetic images, the pixel intensity is well defined as described by Caldas et al. [40]. Thus, a simple Otsu's thresholding method (please refer to Section 2.1.1) satisfies as an image segmentation technique to generate the masks for training. Nonetheless, different procedures may be employed to tackle mask generation due to the challenges presented in real images. At first, we proceed with the Global Otsu methodology as described in Caldas et al. [40]. It did not yield good results (an improper value of threshold suggested) as shown in Figure 6 due to the poor lighting conditions in the area analyzed. For instance, Figure 6a shows that the method could capture part of the information of the bubble contour. However, the penumbra on the left side of the image made it impossible to establish a value that could properly delimit the background and the bubble. It should be noticed that the thresholding was inverted since yellow points represent pixels with the value number "one". On the other hand, Figure 6b shows better thresholding results, but still, the contours were not adequately defined. Figure 6c depicts

a poor segmentation due to a key lighting difference in the background, which led the algorithm to compute these differences in pixels as the threshold. Figure 6d presents another case of failed segmentation, in this case, because the left side of the image was brighter than the right one; therefore, the algorithm determined this bright side as belonging to the bubble. Evidently, an alternative procedure must be undertaken to overcome the issues mentioned above. A certain degree of proper segmentation when using Otsu's thresholding suggests that a mixture of techniques may be the solution, which needs to be developed.

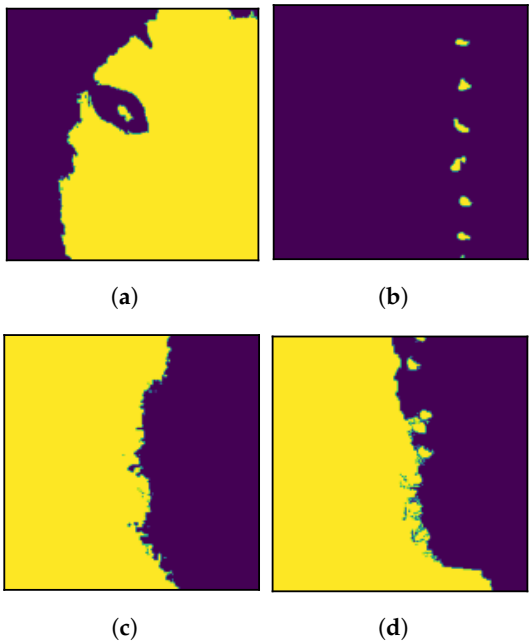
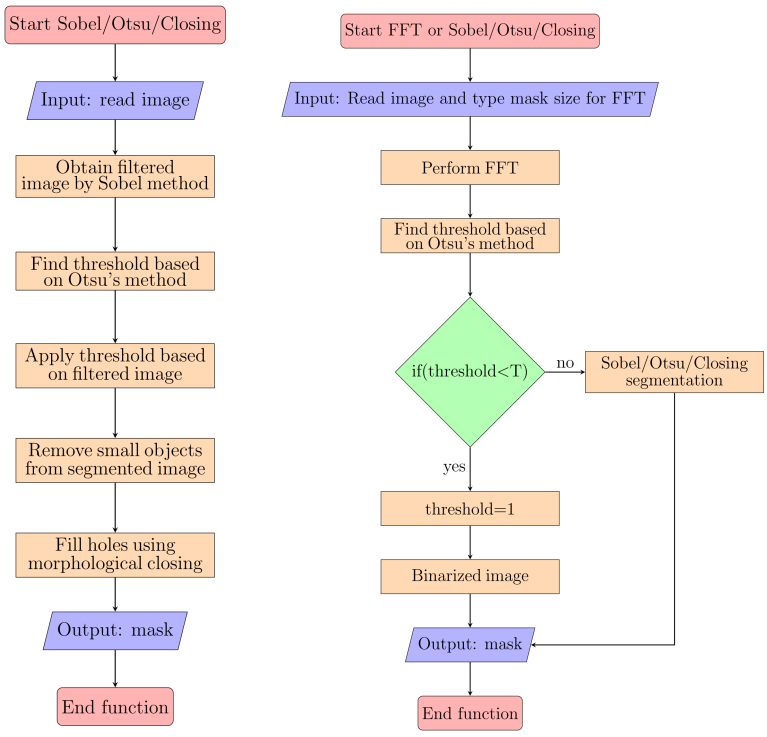


Figure 6. Examples of frames pre-processed with Global Otsu thresholding by video ID: (**a**) 08, (**b**) 14, (**c**) 19, (**d**) 37. Bubbles are in yellow; background in purple.

Combining the image processing methods described in Section 2.1 was the solution to overcome the pixel intensity heterogeneity. As the first step, the Sobel filter was used to delimit the contours, in which clear contours were found.

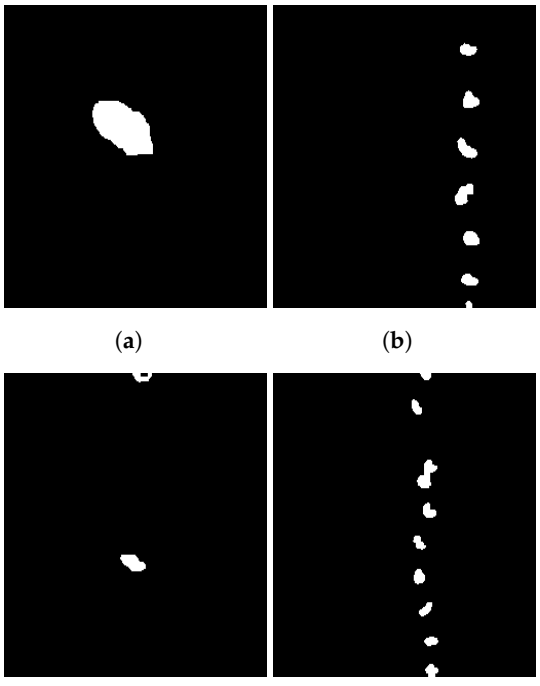
Based on these contours, Otsu's thresholding was used to partition the regions. Then, the morphological operation of closing was used to fill holes and improve contours. Figures 7a and 8 depict the methodology employed and the results in this case, respectively.

Nevertheless, the "Sobel/Otsu/Closing" algorithm returned poor results for videos #17, #25, and #33. The reason can be found after analyzing the flow conditions: leaks that flow under intermittent regimes, i.e., bubbles that do not flow continuously, led to poor identification of contours by the Sobel filter since some frames captured were empty, without any bubbles. Thus, another high-pass filter should be employed. In the videos above, the Fast Fourier Transform was used to highlight the boundaries. When creating the "FFT or Sobel/Otsu/Closing" algorithm, the consideration used for choosing either the FFT or Sobel filter on each frame was based on whether the image was blank. A decision criterion was developed based on a value T, manually tested for each of the videos. Figure 7b illustrates the flowchart, and Figure 9 shows the final results from the "FFT or Sobel/Otsu/Closing" method. It should be noticed that the method was capable not only of segmenting the bubble and the background but also of returning an empty image when no flow was present, as depicted in Figure 9c.



(**a**) (**b**)

Figure 7. Algorithm flow chart of the method employed for generating ground truth data for CNN training: (**a**) Sobel/Otsu/Closing, (**b**) FFT, or Sobel/Otsu/Closing.



(**a**) (**b**)

(**c**) (**d**)

Figure 8. Examples of frames processed with the "Sobel/Otsu/Closing" algorithm: (**a**) 08, (**b**) 14, (**c**) 19, (**d**) 37.

Table S2 in the Supplementary Materials summarizes the methods employed for each video. We decided not to use videos #40–#44 because the camera was not static, which would compromise the measurement of the equivalent bubble diameter. Other videos that could not produce good true masks were discarded (01, 04, 07, 38, 39). For

converting the area computed as a bubble to a physical scale, the height of the image was measured by a ruler and compared to the height of the frame in pixels, which resulted in a conversion factor fc.

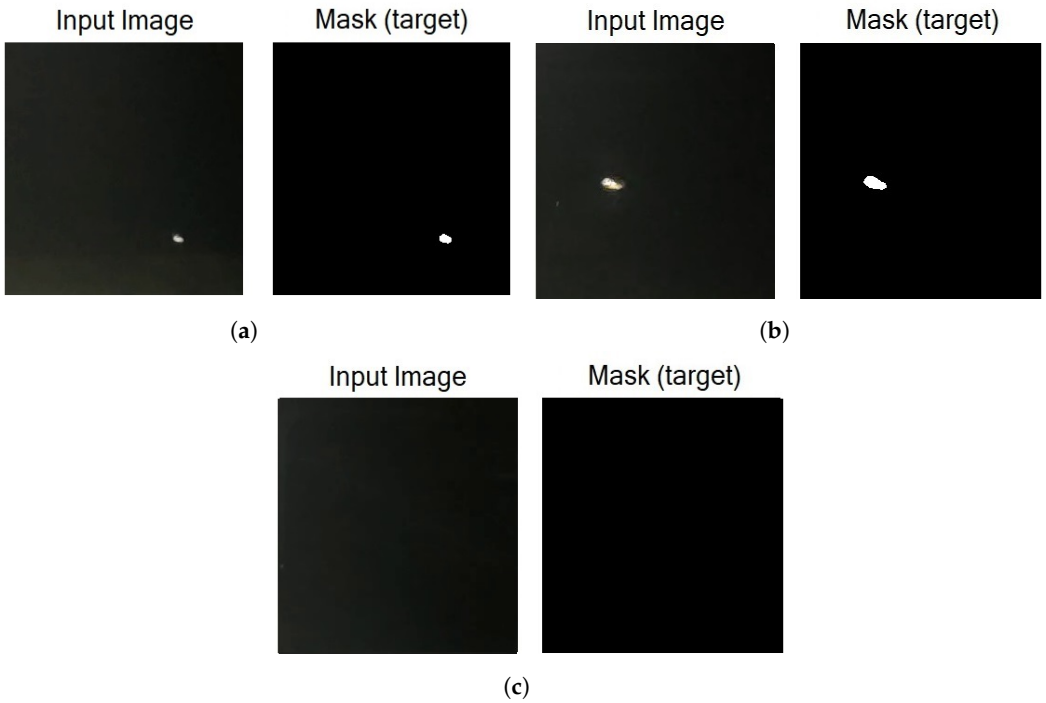


Figure 9. Examples of frames processed with the "FFT or Sobel/Otsu/Closing" algorithm: (**a**) 17, (**b**) 25, (**c**) 33.

3.3. Training CNNs with Real Images

In this part of the work, two methodologies are compared employing U-Net. The steps followed for each one are:

1. Importing the U-Net architecture and weights from a synthetic image model performed by Caldas et al. [40]. No hyperparameter optimization is carried out. This step is referred to as "transfer learning".
2. Building new U-Net models based on the hyperparameter optimization of the architecture and training, i.e., number of filters on a block, activation functions, whether to apply batch normalization, dropout value, optimization method, learning rate, and weights decay. The weights are trained from scratch; i.e., no prior information is given to the network. This step is referred to as "hyperparameter optimization".

In method 1, the network structure is maintained. A total of 7,759,521 parameters are trained. The first block of the contracting path has 32 filters of size 3×3, and the number of filters is doubled on each block, reaching 512 filters in the most profound block. Then, the number of filters is halved in the expanding path. On each block of the contracting path, a convolutional layer employing ReLU as an activation function and a dropout of 20% is succeeded by a max pooling layer and by another convolutional layer with ReLU. The contracting path is composed of blocks of transposed convolutional layers and convolutional layers. The last convolutional layer has a sigmoid as an activation function. The batch size is 32.

In method 2, the hyperparameter optimization employed a combination of Tree–Parzen Estimator (TPE) to provide sampling and a Hyperband algorithm as the watchdog for early stopping. The Optuna framework was the library used [78]. The search space is described in Table 1. A total of 432 configurations are possible. However, the idea is not to

perform a grid search but to obtain a fast response with fewer experiments. One-fifth of the dataset was employed for time-saving purposes during the hyperparameter optimization. Then, the best configuration found was trained again on the whole dataset, which was selected based on the maximum global Dice–Sørensen coefficient.

In both cases, the dataset split was 60%, 24%, and 16% for training, validation, and testing, respectively.

Table 1. Hyperparameter optimization search space.

Hyperparameter	Range
Initial number of filters	[16; 32]
Activation functions	[ReLU; ELU; SeLU]
Dropout	[0.1; 0.2; 0.3]
Batch Normalization	[True; False]
Dropout on Upsampling	[True; False]
Optimization method	[RMSprop; Adamax; Adam]
Learning rate	$[1\times10^{-4}; 1\times10^{-3}]$

As a metric to train the neural network, we employed the Dice–Sørensen coefficient, Equation (12a), a measure that computes the spatial overlap between two segmentation sets. Representing y_n as the ground-truth sample n, $\hat{y_n}$ as the corresponding predicted sample, and i, j indexes relative to points in an image, the following expression can be written:

$$DSC(y_n, \hat{y_n}) = 2\frac{|y_n \cap \hat{y_n}|}{|y_n| + |\hat{y_n}|} = \frac{2\sum_{j=0}^{256}\sum_{i=0}^{256} \hat{y}_{ij,n} \cdot y_{ij,n}}{\sum_{j=0}^{256}\sum_{i=0}^{256} \hat{y}_{ij,n} + \sum_{j=0}^{256}\sum_{i=0}^{256} y_{ij,n}} \tag{12a}$$

For a dataset with N samples, the global Dice–Sørensen coefficient is written as in Equation (12b).

$$GDSC(y, \hat{y}) = \frac{2|y \cap \hat{y}|}{|y| + |\hat{y}|} = \frac{2\sum_{n=1}^{N}\sum_{j=0}^{256}\sum_{i=0}^{256} \hat{y}_{ij,n} \cdot y_{ij,n}}{\sum_{n=1}^{N}\sum_{j=0}^{256}\sum_{i=0}^{256} \hat{y}_{ij,n} + \sum_{n=1}^{N}\sum_{j=0}^{256}\sum_{i=0}^{256} y_{ij,n}} \tag{12b}$$

It should be highlighted that the loss function considered for training was $1 - DSC(y_n, \hat{y_n})$, in which each sample and mask are computed individually. Then, the average for all samples and masks is calculated [79]. This strategy, rather than using the global metric as loss—Equation (12b)—was used to force the network to learn each image individually, not the whole batch. Since the background composes most of the data, it has a significant impact on the loss calculation. Thus, missing one bubble in the whole batch may have a negligible effect on the global Dice, whereas in an individual image, missing a bubble may be noticed by the individual Dice [79]. It is a workaround to avoid the well-known problem of class imbalance [80].

It should be stressed that the U-Net is incapable of quantifying by itself. The neural network only supplies a predicted mask with defined contours. The bubble diameter $\hat{d_b}$ is computed in a post-processing step. Yet it is straightforward since regions that belong to the bubble have a pixel with code "one", and the ones to the background have a pixel number "zero". For this purpose, the Regionprops function was employed in the Scikit-image library [81]. This widespread function is also present in other imaging libraries, like OpenCV and MATLAB. This function calculates the area based on the pixels, as explained before, and the equivalent diameter is based on the area:

$$\hat{d_b} = \sqrt[2]{\frac{4A_b}{\pi}} \tag{13}$$

For measuring the flow rate, first, the major axis a and minor axis b of the bubble are measured in a determined horizontal position. Since the area of the bubble can vary during upwelling, we chose to use two positions ($y = 120$ px and $y = 220$ px from the top). The flow meter only calculates the geometric properties if a bubble passes through. Since the bubble is an ellipsoid, the volume is calculated as follows [20]:

$$\hat{r} = \sqrt[3]{\frac{a^2 b}{8}} \tag{14a}$$

$$\hat{V}_{eq} = \frac{4}{3}\pi \hat{r}^3 \tag{14b}$$

When the bubble passes through the flow meter, the average of the volumes is considered to avoid multiple counting.

3.4. Computational Resources

All code was written in Python 3. To generate the masks described in Section 3.2, the library Scikit-image was employed [81]. The neural network models were built through the high-level deep learning API Keras [82] in the Tensorflow v2 platform [83]. For the hyperparameter optimization, as mentioned above, we used the Optuna framework [78], including their TPE and Hyperband algorithm implementations. All experiments were monitored with Keras callbacks and, when applicable, Optuna callbacks.

We carried out the training and hyperparameter optimization in a Dell XPS 8940 built with an Intel® Core™ i7-10700 CPU, 16 GB RAM, 8 cores and coupled with a GPU NVIDIA® GeForce® RTX™ 3060 with 12 GB of memory.

4. Results and Discussion

In this section, the numerical flow rates and bubble diameter values obtained by transfer learning and hyperparameter optimization are presented. For the validation of the model, the equivalent bubble diameter $\hat{d}_b$ is compared with the experimental ones, d_b, performed by Nascimento et al. [77], as shown in Equation (11b). In this section, the analysis is separated into groups, which are organized according to the type of flow, orifice, and fluid. The first has a continuous flow, the capillary tube was used as the orifice type, and tap water was used as the fluid (Table 2, where d_i, Q, v_o are the orifice diameter, flow rate, and release velocity, respectively). The second one has the same conditions but has an intermittent flow behavior. Group 3 has other types of orifice but has a continuous flow, and the system is composed of tap water (Table 3). The last one, group 4, has salt water, continuous flow, and a capillary tube as an orifice type (Table 4). Only in groups 1 and 3 were the experimental bubble diameters d_b available.

Furthermore, the bubble diameter values predicted by the literature in the same conditions are presented, i.e., with the same leak flow rate and orifice diameter. It should be highlighted that these equations are only valid for the first bubble released in a water column and not for the average of bubbles in a water column. They were described here just for qualitative purposes, not for validation. Next, the probability distribution function is computed based on a specific statistic value for the bubble diameter. Finally, some computational considerations are also discussed in this work. It should be noted that during the discussion of the results, the frames were shuffled during the split of training, validation, and test datasets (the training of the network does not consider the temporal relationship). This step is necessary to guarantee a good fit for the CNN training. Therefore, the results are not presented in the same order as the original videos. Furthermore, the dataset split was performed to avoid neural network overfitting, as explained in Section 3.3.

Since the training dataset has 60% of the original dataset, the results will be discussed using this set, except where noted.

Table 2. Group 1: continuous flow, capillary tube as orifice type, and tap water as fluid.

ID	d_i (mm)	Q (mL/min)	v_o (m/s)	d_b (mm)
26	1.0	21.1	0.45	4.1
27		31.0	0.66	4.1
28		54.1	1.15	4.4
29		69.2	1.47	4.1
30		82.9	1.76	4.6
31		93.9	1.99	4.4
32		101.2	2.15	4.5
10	2.0	25.2	0.13	4.8
11		30.9	0.16	4.9
12		46.4	0.25	4.5
13		72.3	0.38	5.0
14		95.6	0.51	5.3
15		110.8	0.59	5.5
16		153.2	0.81	5.9
18	5.0	30.0	0.03	6.4
19		42.1	0.04	6.6
20		69.0	0.06	7.0
21		86.6	0.07	7.0
22		131.9	0.11	7.5
23		189.5	0.16	7.9
24		234.4	0.2	8.2

Table 3. Group 3: continuous flow, Bunsen burner (01–03, 07–09)/Brass T pipe (04–06) as orifice type, and tap water as fluid.

ID	d_i (mm)	Q (mL/min)	v_o (m/s)	d_b (mm)
01	0.5	3.0	0.25	4.9
02		51.0	4.33	5.5
03		96.0	8.155	5.9
04	1.0	11.1	0.24	4.1
05		23.5	0.5	4.1
06		32.0	0.68	4.2
07	5.0	19.0	0.02	6.7
08		109.0	0.09	7.3
09		240.0	0.2	7.8

Table 4. Group 4: continuous flow, capillary tube as orifice type, and salt water as fluid.

ID	d_i (mm)	Q (mL/min)	v_o (m/s)
34	1.0	15.2	0.32
35		20.1	0.43
36		32.5	0.69
37		32.5	1.27
38		32.5	1.90
39		32.5	2.37

4.1. Computational Performance

The training and validation curves represent the model's performance over epochs for the training and validation datasets, respectively. These curves are useful for evaluating

how well a machine learning model is learning and generalizing to unseen data. A small gap between the curves indicates good generalization, while a large gap suggests overfitting (high variance) or underfitting (high bias). Figure 10 illustrates the history of the transfer learning training. Regarding computational performances, a quick convergence was found in terms of loss. The algorithm stopped in less than 25 epochs by employing the early-stop approach. Furthermore, the training loss ($1 - DSC(y_n, \hat{y_n})$) achieved a value of less than 0.10 in the second epoch, which indicates that the training is efficient due to the large amount o data available. The average of the Dice–Sørensen coefficient computed for the image and targets individually reported were 0.91 and 0.68 for training and validation, respectively, while the global Dice–Sørensen coefficient was 0.97 for training and validation. The difference in global and local metrics can be explained due to the major presence of the background (class imbalance in semantic segmentation), as explained in Section 3.3.

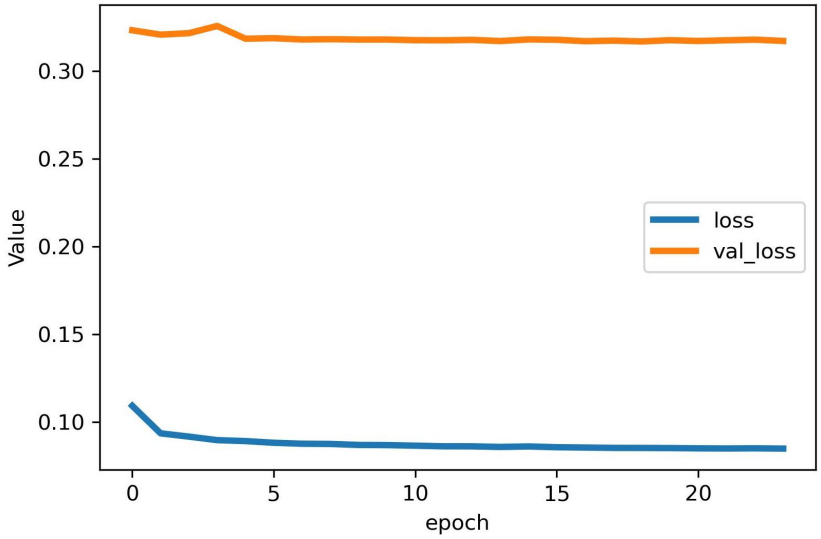


Figure 10. Transfer learning training evolution.

Figure 11 illustrates the hyperparameter optimization and the values achieved in each trial. The pruning algorithm allows a quick choice of promising candidates. We ran 64 trials that lasted for 110.2 h. Out of 64 trials, 41 were pruned, and 23 were complete. The early stopping works in a two-fold way: by using the pruning algorithm itself and by using the callback criteria of stopping if no improvement 1×10^{-4} is achieved during three epochs. Therefore, only seven trials (#2, #9, #21, #47, #54, #56, and #58) lasted more than 10 epochs of training. Trial #58 lasted 20 epochs, which also traduced in more computational time (6.62 h) and had the best validation loss.

Figure 12 illustrates the parallel coordinate plot. The training returned, in general, elevated values of the target metric (>0.7). It can also be seen that the combination of Adamax as an optimizer, a dropout of 0.3, and "False" on dropout on upsampling is a bad choice. All three candidates that contained this combination figured in the six worst ones in terms of metric performance. To investigate further, Table 5 shows the 10 trials that demonstrated the best performance for the global Dice–Sørensen coefficient. It can be inferred that using batch normalization is vital for high performance. A good balance between computational cost and precision is needed to find an optimal model. In this sense, the number of filters substantially influences the time necessary for computing as the number of weights to be trained increases. Therefore, the best architecture was selected to be trained through the complete dataset (#58) based on the similarity coefficient of Dice–Sørensen.

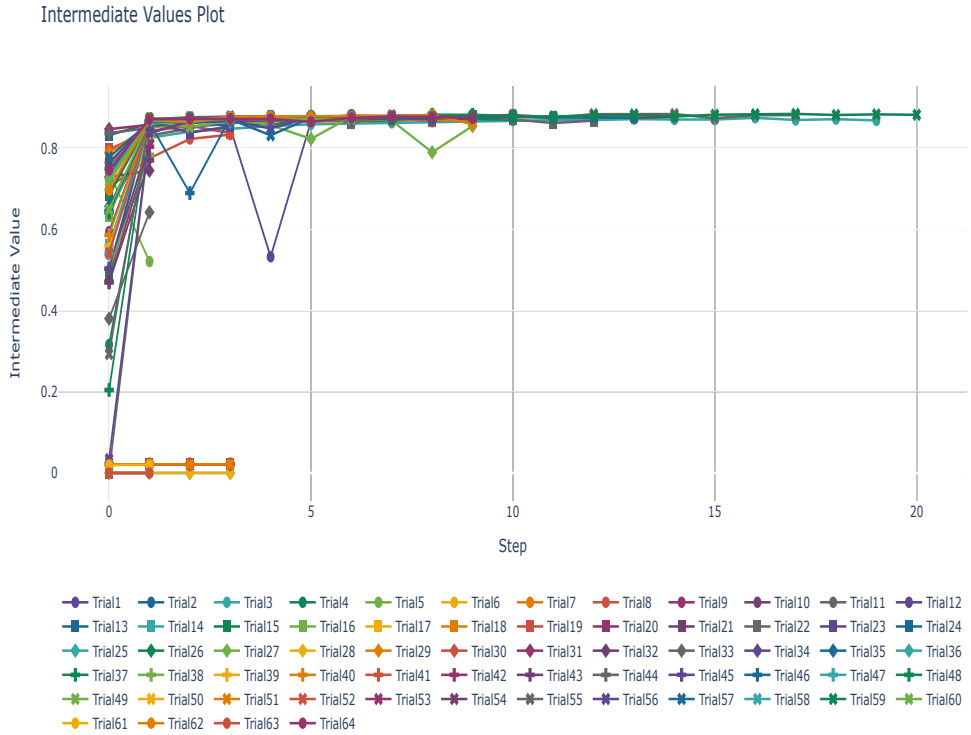


Figure 11. Evolution of hyperparameter optimization with intermediate values plot.

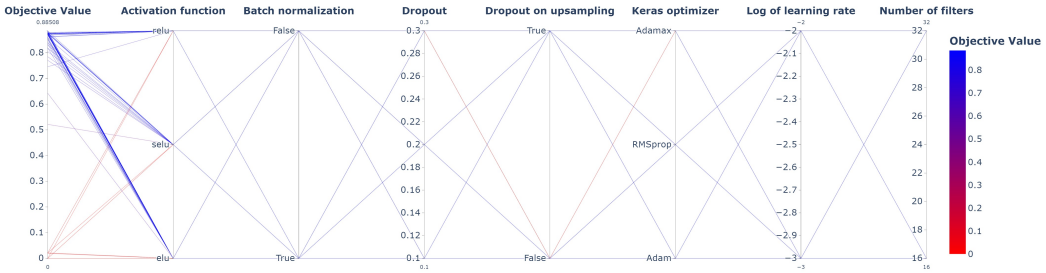


Figure 12. Parallel coordinates plot of the hyperparameter optimization.

Table 5. Best 10 trials in terms of global Dice–Sørensen coefficient.

Trial	Global Dice–Sørensen Coefficient	Duration (h)	Activation Func.	Batch Normalization	Dropout	Dropout Upsampling	Optimizer	Learning Rate	Filter
58	0.8851	6.62	selu	Yes	0.2	Yes	Adamax	0.01	32
54	0.8846	4.82	elu	Yes	0.2	No	Adam	0.01	16
47	0.8831	3.30	elu	Yes	0.2	No	Adam	0.01	16
51	0.8810	2.78	elu	Yes	0.2	No	Adam	0.01	16
21	0.8791	3.81	relu	No	0.3	Yes	RMSprop	0.001	32
56	0.8785	4.43	selu	Yes	0.2	Yes	Adamax	0.01	32
9	0.8780	3.96	elu	Yes	0.1	No	Adam	0.001	16
55	0.8770	2.92	selu	Yes	0.2	Yes	Adamax	0.01	32
45	0.8767	2.27	elu	Yes	0.1	Yes	Adam	0.01	16
14	0.8763	2.98	relu	Yes	0.2	Yes	RMSprop	0.001	32

To better understand the influence of each parameter, Figure 13 shows the parameter importance (in terms of the fraction of variance explained) using a functional Analysis of Variance (ANOVA) evaluation [84]. As expected, batch normalization exerts the most substantial influence among all parameters, followed by the activation function, though the former has a much greater effect. On the other hand, the number of filters has a minor influence, as evidenced by the contour plot of the hyperparameter combinations

shown in Figure 14. In this figure, the number of filter charts is, in most cases, symmetric, whereas when batch normalization is used, better results are seen. The optimizer method selected still has some effect: studies that employed "Adam" showed poorer outcomes depending on the combination. For dropout, architectures using 0.1 and 0.2 obtained, in general, superior performance. With respect to the learning rate, both 0.01 and 0.001 values shared similar results. Using dropout on upsampling seems to be more promising than not using it. Regarding the activation function, the "selu" function may be the best choice. Therefore, based on the discussion above, an apparent successful combination matches the architecture of trials #58, #56, and #55, which are the first, sixth, and eighth best hyperparameter performances, respectively.

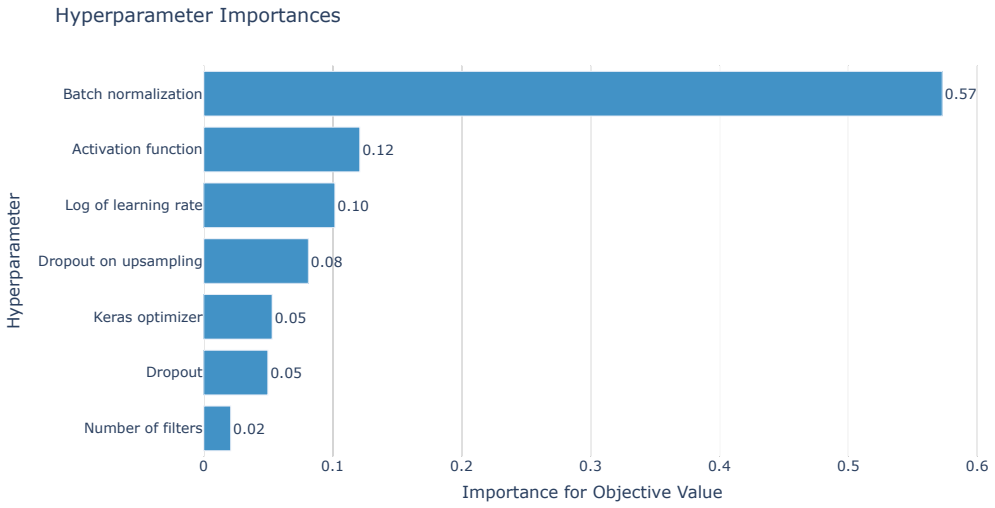


Figure 13. Hyperparameter importances.

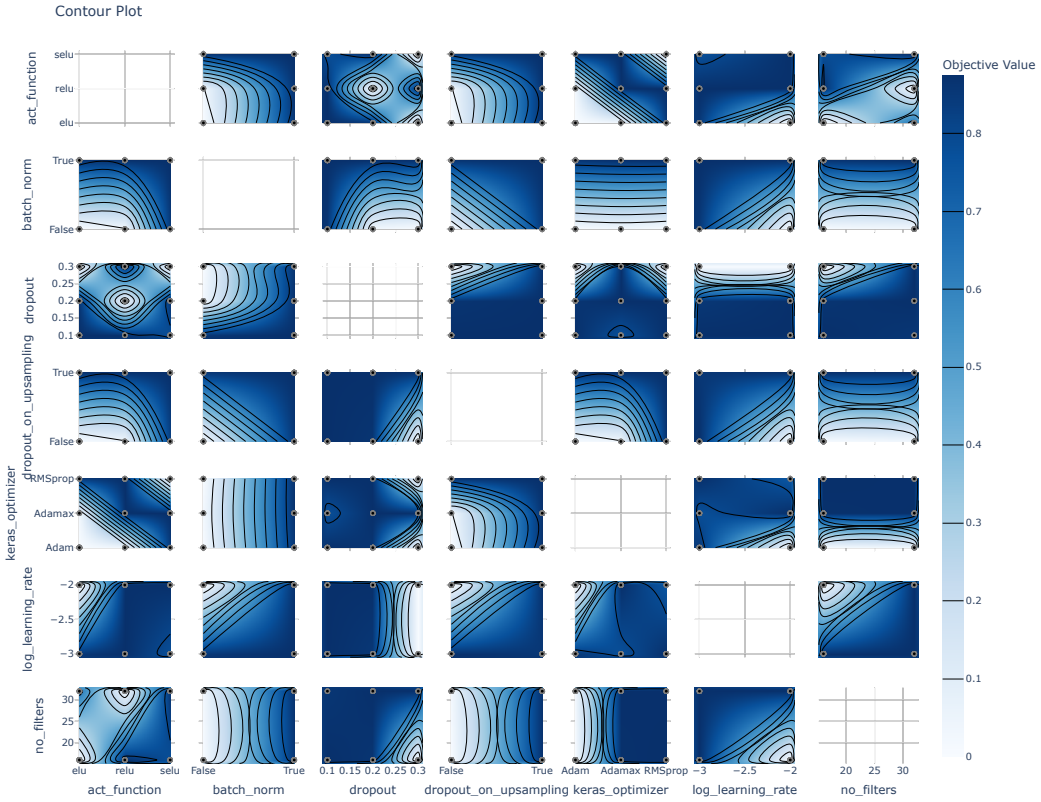


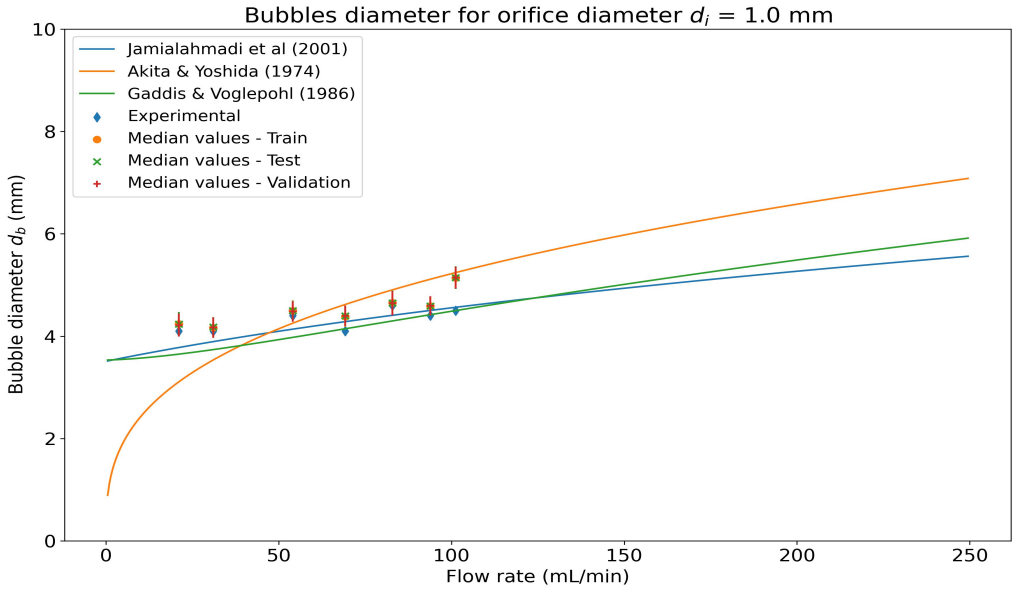
Figure 14. Contours plot based on hyperparameters studied during optimizations.

In deep learning training, the learning rate directly influences the generalization of the model, which trades off the overfitting and underfitting [68]. As a rule, a minimal learning rate can lead to overfitting and slow performance. In our model, the learning rate range was stable and had little influence over the metric. For instance, for the configuration found in trial # 9, changing the learning rate from 0.001 to 0.01 impacted the metric from 0.8780 to 0.8758. However, the computational cost is significant; the first lasted for 3.96 h, while the latter lasted for 0.84 h. This result is according to the findings in the work of Smith [68], who studied the influence of several hyperparameters in benchmarking datasets. The author suggests that an optimal learning rate can speed up the training without becoming unstable. The results of the complete training from hyperparameter optimization are discussed in the following sections. In general, they were very similar to the results of the transfer learning step.

For the whole dataset training in hyperparameter optimization, the Dice–Sørensen coefficients for training and validation were identical to those obtained through transfer learning, both for individual images and the overall group (individual images: 0.91 for training, 0.68 for validation; overall group: 0.97 in both cases).

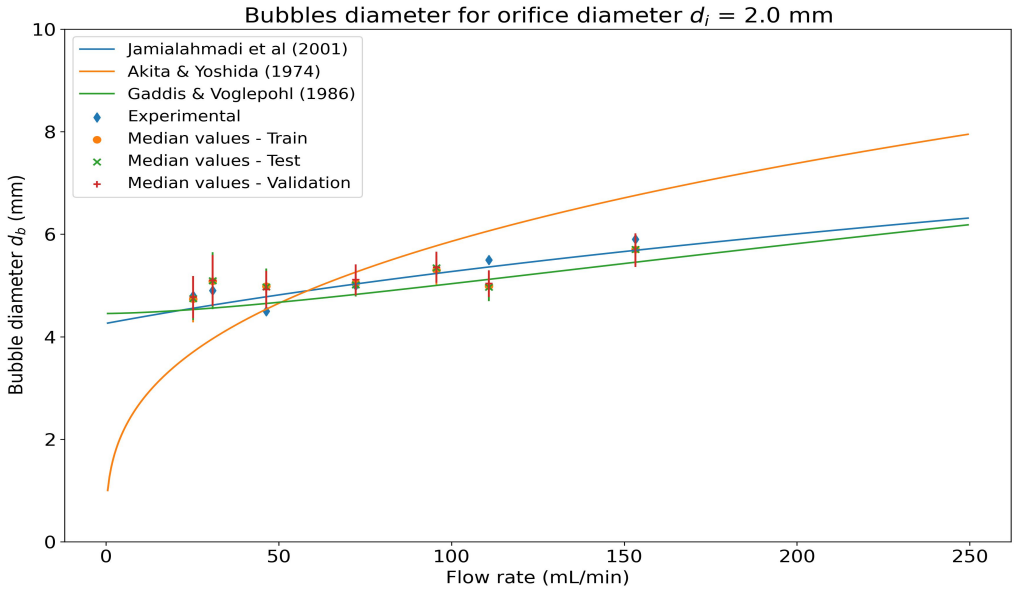
4.2. Group 1

The numerical values of bubble diameter $\hat{d}_b$ obtained by hyperparameter optimization are shown in Figure 15. As mentioned, trial #58 was selected for training through the whole dataset. A good agreement is found between the median values and the experimental ones, found by counting the bubbles in a beaker [77]. Using the maximum bubble could provide some overestimation for the bubble diameter. The exception is on the largest orifice diameter ($d_i = 5.0$ mm) and shown in Figure 15c. However, it should be noted that the experimental values for this case also lie within the median diameter error bars for three out of seven. The root mean square error (RMSE) of the bubble diameter for orifice diameters of 1.0 mm, 2.0 mm, and 5.0 mm was 0.28 mm, 0.28 mm, and 1.31 mm, respectively, with an overall RMSE of 0.80 mm for the entire group.

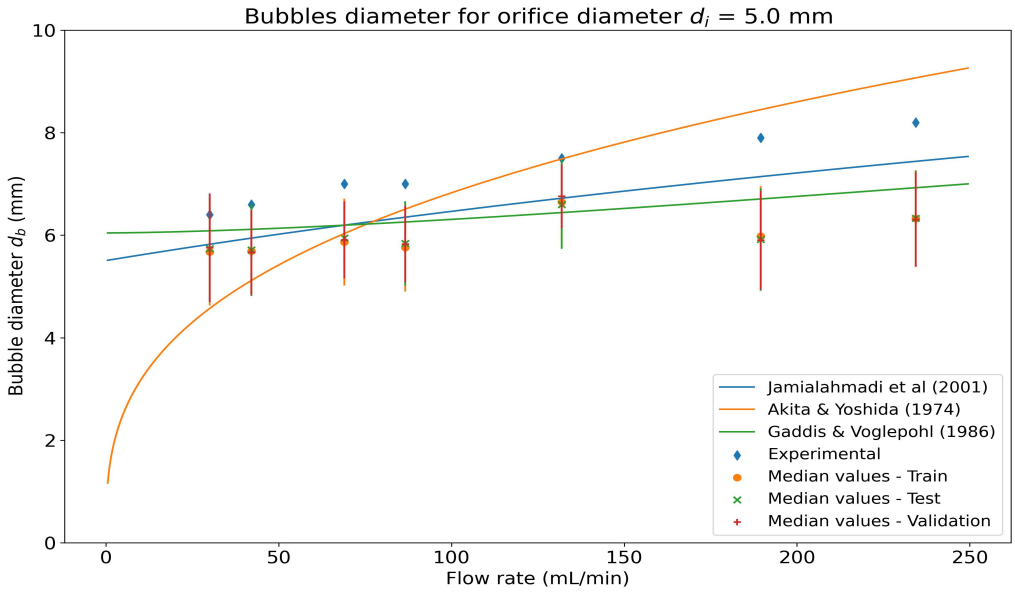


(**a**)

Figure 15. *Cont.*



(**b**)



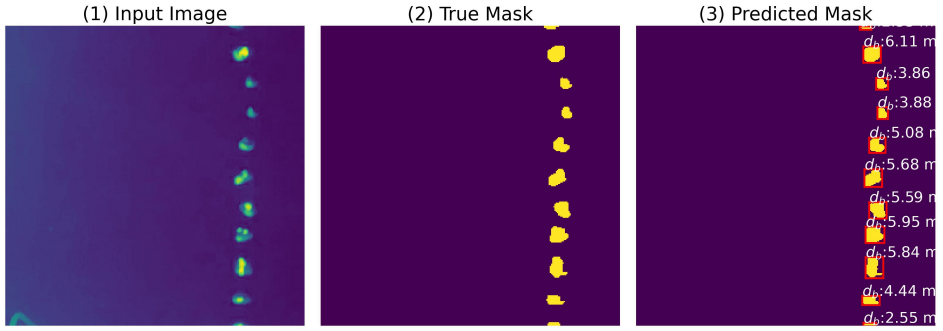
(**c**)

Figure 15. Comparison of bubble diameter values found by hyperparameter optimization with the experimental and literature values [85–87] for different orifice diameters in Group 1: (**a**) $d_i = 1.0$ mm, (**b**) $d_i = 2.0$ mm, (**c**) $d_i = 5.0$ mm.

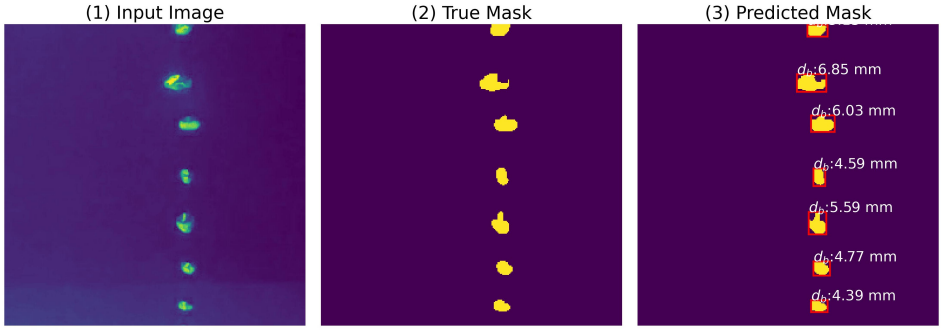
Furthermore, after plotting the values predicted by the correlations from the literature [85–87], the median values were found above them (but close to the experimental values, Figure 15a,b), or their error bars lay in the region of the literature curve (Figure 15c). Moreover, the values are consistent for each CNN dataset—training, validation, and test—which shows the algorithm's robustness. Therefore, the methodology presented herein is considered valid for computing the bubble diameter.

The bubble diameter curves from the literature [85–87] reported here are for the detachment from the nozzle. The bubble diameter in a column can change due to the force dynamics. An increase in bubble diameter along the column can occur due to the decrease in static pressure (considering no breakup/coalescence) [88]. This is a hypothesis for overestimating the values from those expected in the literature.

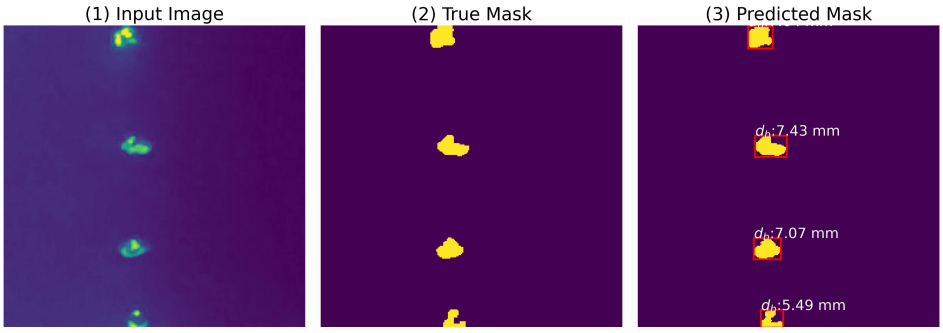
Figure 16 shows examples of prediction masks and the corresponding bubbles printed. By analyzing this figure, it can be inferred that the CNN trained by transfer learning could reproduce the ground truth data (or the true mask) well. In addition, in Figure 16a, the U-Net did not consider other objects (see the left bottom corner) as bubbles, which is a positive performance. The algorithm is capable of processing multiple bubbles successfully, even for high-speed leakages, like in Figure 16a. However, it should be noted that dense bubbly flows with several overlapping bubbles were not considered in this work.



(**a**)



(**b**)



(**c**)

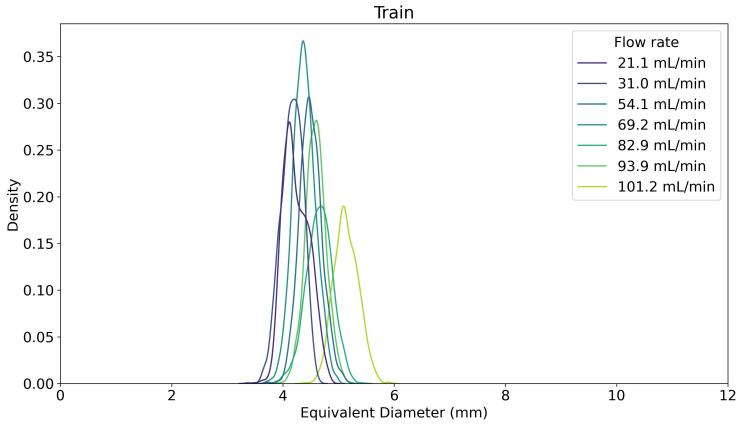
Figure 16. Input image, true mask, and predicted mask by hyperparameter optimization training in Group 1 in different conditions: (**a**) $d_i = 1.0$ mm and $Q = 101.2$ mL/min, (**b**) $d_i = 2.0$ mm and $Q = 110.8$ mL/min, (**c**) $d_i = 5.0$ mm and $Q = 131.9$ mL/min. The predicted mask is printed with the bubble diameter d_b (in mm).

The bubble probability density function (PDF) was estimated using kernel distribution [89]:

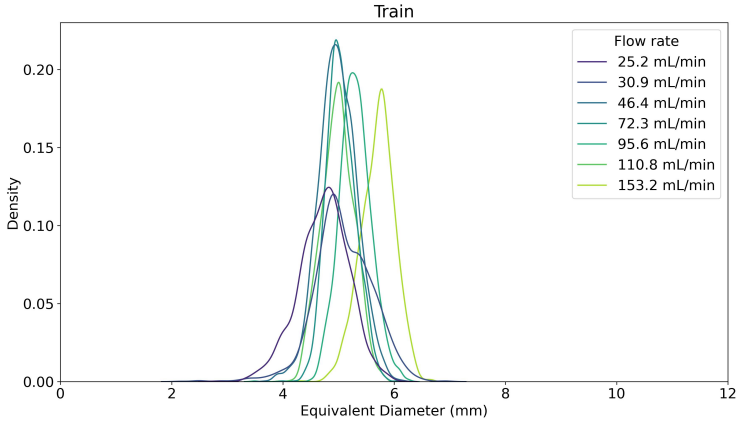
$$\widehat{f}_h(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right) \tag{15}$$

where n is the number of samples, h is a smoothing parameter known as bandwidth, and K is the kernel used—for Gaussian, $K(x;h) \propto \exp(-\frac{x^2}{2h^2})$.

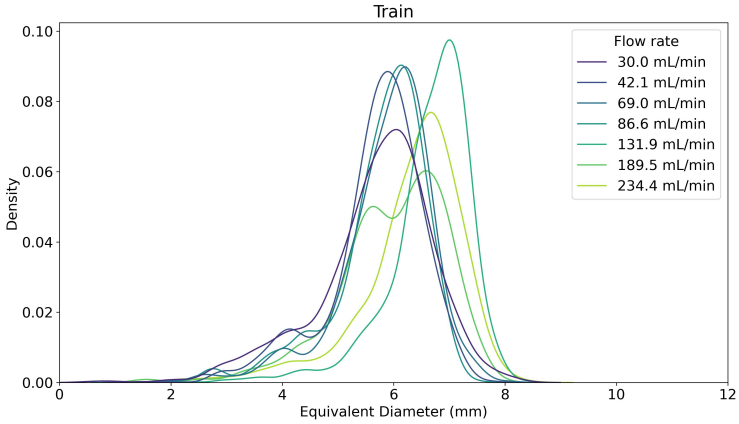
The PDFs for the training dataset for median values of the equivalent bubble diameter are shown in Figure 17. The median values were chosen here as they are the most representative statistic based on the preceding discussion. After analyzing Figure 17a,b, they present a high probability density (all with >0.10) with a well-defined bell shape, which suggests a low level of randomness. However, for the largest orifice, as shown in Figure 17c, the bubble diameter median values have a low density, which indicates an elevated level of randomness, as already pointed out by the greater error bars.



(**a**)



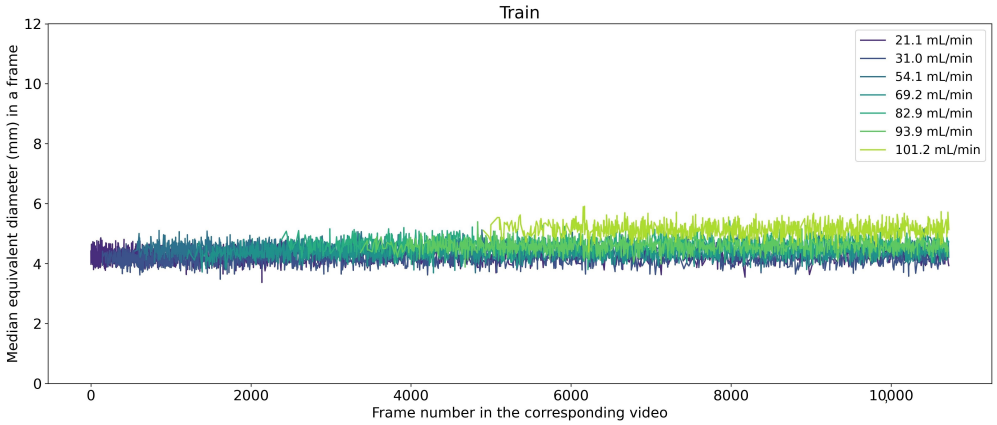
(**b**)



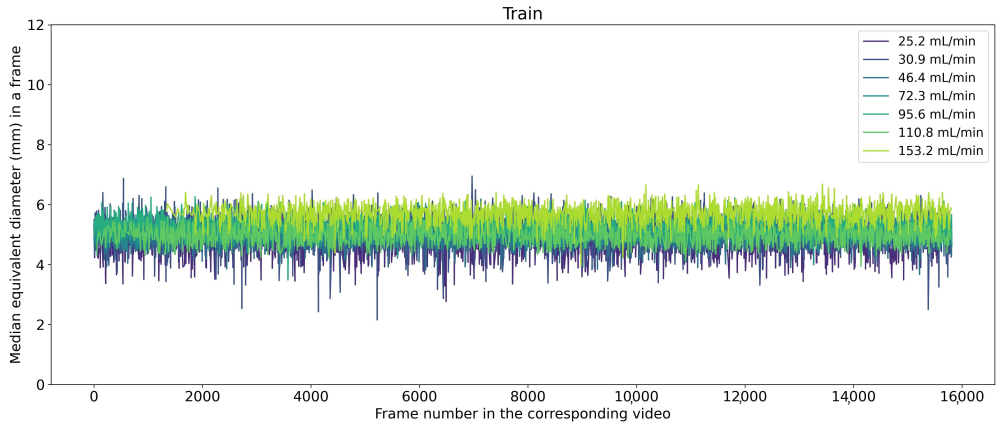
(**c**)

Figure 17. Probability density functions of equivalent bubble diameter based on the median value of hyperparameter optimization for the training dataset in different orifice diameter: (**a**) $d_i = 1.0$ mm, (**b**) $d_i = 2.0$ mm, (**c**) $d_i = 5.0$ mm.

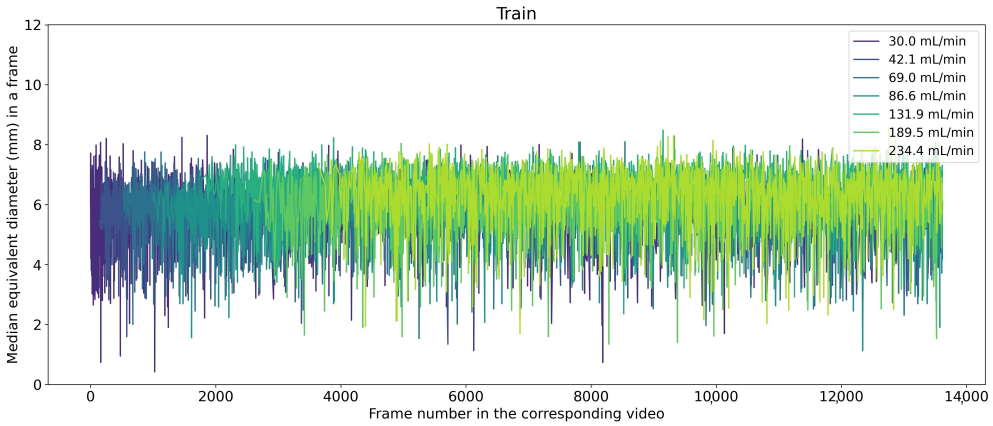
Figure 18 describes the median equivalent bubble diameter values on each frame for the different orifice diameters. There is a clear trend towards more variability in larger orifice diameters, as noted in Lu et al. [90], who studied the influence of orifice size diameter on an external loop airlift reactor. The authors also observed a decrease in the probability density by increasing the orifice diameter. The smaller peaks in a PDF, as seen in Figure 17, are equivalent to a less uniform bubble diameter.



(**a**)



(**b**)



(**c**)

Figure 18. Median values of equivalent bubble diameter computed by hyperparameter optimization for the training dataset in different orifice diameters: (**a**) $d_i = 1.0$ mm, (**b**) $d_i = 2.0$ mm, (**c**) $d_i = 5.0$ mm.

Transfer learning yielded results similar to those of hyperparameter optimization. For instance, the bubble diameters predicted only diverge in the second decimal number, and the figures generated are almost identical to Figures 15–18. Please refer to Figures S2–S5, which show the comparison of bubble diameter values with the experimental ones, the average of the median values of bubble diameter, the diameter of the bubble predicted, and the PDFs for transfer learning.

4.3. Group 2

Figure 19 depicts the bubble diameters' median values for the group with intermittent flow for hyperparameter optimization. The CNN could correctly assign images without bubbles, which resulted in several images with diameters' values equal to zero. Furthermore, it correctly captured the frames when the bubble was present. It should be highlighted that these videos contained few bubble releases. Thus, the algorithm recognized a sequence of no-flow frames, which resulted in zero bubble diameter. Since the dataset was split randomly, Figure 19 also had random values. The average values of the bubble diameter, not considering the frames without bubbles, were 3.93 mm for video #33 (d_i = 1.0 mm), 3.36 mm for video #17 (d_i = 2.0 mm), and 6.73 mm for video #25 (d_i = 5.0 mm). It should be noted that only one bubble is released in this intermittent flow regime. Therefore, a comparison with the literature on bubble detachment is valid. The model of Jamialahmadi et al. [85] predicted 3.5 mm, 4.3 mm, and 5.5 mm, respectively, while equations from Gaddis and Vogelpohl [87] indicated 3.5 mm, 4.5 mm, and 6.0 mm, respectively. The underestimation in the video from d_i = 2.0 mm can be attributed to the slower release from the nozzle, in which the U-Net calculates the diameter even before the bubble detaches from the nozzle.

Transfer learning presented some differences from hyperparameter optimization. The average for the videos, not considering the frames without bubbles, were 3.97 mm for video #33 (d_i = 1.0 mm), 3.37 mm for video #17 (d_i = 2.0 mm), and 1.64 mm for video #25 (d_i = 5.0 mm). Again, the underestimation in video #17 is because the bubble is formed and stays on the nozzle for some time before total release. In video #25, small objects (<2.0 mm) contaminated the water column and were confounded with bubbles by transfer learning. Results from video #25 for hyperparameter optimization are much closer to the ones expected by the literature because the small objects were correctly not considered as bubbles. The median values for group 2 using transfer learning are shown in Figure S6. A potential limitation under intermittent flow is that circular objects may be misclassified as bubbles, which is a common challenge in semantic segmentation.

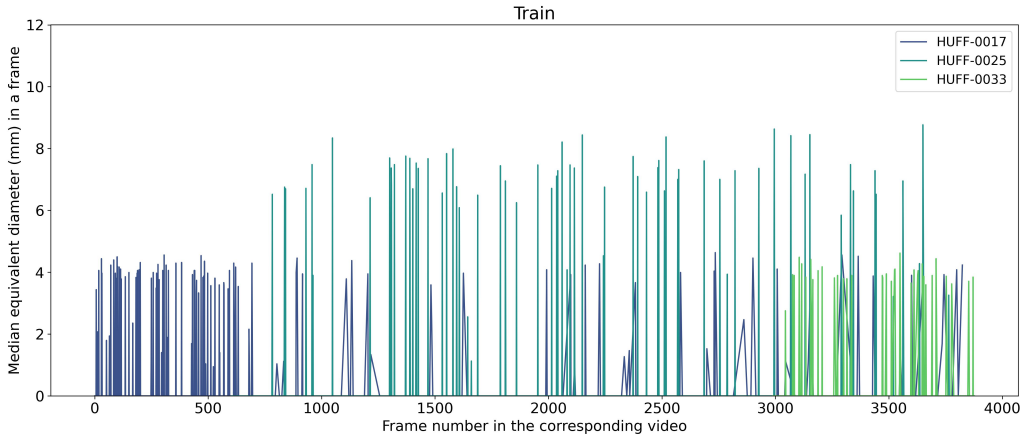


Figure 19. Median values of equivalent bubble diameter computed by hyperparameter optimization for the training dataset in Group 2. For animations, please check Video S2 for Group 2.

4.4. Group 3

The bubble diameter results from Group 3 after trial #58 are displayed in Table 6. The RMSE for Group 3 was 0.76 mm. The results showed that $d_i = 1.0$ mm was again the one that found the closest match to the experimental values, with less than 2% of deviation from the average of median predicted bubble diameter values. For $d_i = 5.0$, the algorithm underestimated $\hat{d}_b$, but all values lie within the standard deviation, which was larger in both cases. It should be highlighted that the values expected from the literature were divergent for $d_i = 1.0$ and $d_i = 5.0$. The correlations of Jamialahmadi et al. [85], Akita and Yoshida [86], Gaddis and Vogelpohl [87] expected 3.5–3.7 mm for video 02, 3.9–4.6 for video 03, 6.3–7.0 for video 08, and 7.0–9.5 for video 09. Hence, it could be inferred that the measurement of bubble diameter for larger orifice diameters is an object of divergence. In particular, Akita and Yoshida [86] does not agree with other authors for higher flow rates.

Table 6. Results of Group 3 against values obtained experimentally by Nascimento et al. [77] for the hyperparameter optimization (HPO) and transfer learning (TL).

ID	d_i (mm)	Q (mL/min)	v_o (m/s)	d_b (mm)	Median $\hat{d}_b$ HPO (mm)	Max. $\hat{d}_b$ HPO (mm)	Median $\hat{d}_b$ TL (mm)	Max. $\hat{d}_b$ TL (mm)
02	0.5	51.0	4.33	5.5	5.01 ± 1.11	5.60 ± 0.88	4.75 ± 1.4	5.63 ± 0.86
03		96.0	8.15	5.9	5.22 ± 0.90	6.01 ± 0.54	5.24 ± 0.91	6.04 ± 0.53
05	1.0	23.5	0.5	4.1	4.03 ± 0.40	4.26 ± 0.28	4.05 ± 0.38	4.27 ± 0.30
06		32.0	0.68	4.2	4.12 ± 0.38	4.40 ± 0.31	4.13 ± 0.43	4.42 ± 0.3
08	5.0	109.0	0.09	7.3	6.04 ± 1.25	6.47 ± 0.85	5.90 ± 1.38	6.47 ± 0.85
09		240.0	0.2	7.8	6.70 ± 1.51	7.75 ± 0.68	6.74 ± 1.43	7.75 ± 0.69

Figure 20 illustrates the predicted masks for Group 3. This group is different from the other group in terms of the proximity scale with the bubble. The camera is much closer to the bubble flows. The U-Net was capable of reproducing the true mask well. For instance, in Figure 20a, the line behind the bubble was correctly considered as background in the image. This consistency of accurately measuring bubbles from different scales and perspectives demonstrates the potential and flexibility of the algorithm.

Comparing both approaches in Table 6, it can be seen that the results are similar for $d_i = 1.0$ and videos 03 and 09 but closer to the experimental values for videos 02 and 08 for the hyperparameter optimization. The standard deviations stay in the same numerical range for all cases analyzed. Therefore, the CNN could differentiate again very well the foreground (bubble) from the background even when the true mask incorrectly assigned some pixels to the foreground (please refer to Figure S7, which shows predictions for transfer learning in Group 3 for other conditions than in Figure 20). The reason for this is the large amount of information the CNN was exposed to, which led to a precise recognition of objects.

4.5. Group 4

Regarding equivalent bubble diameter, Figure 21 shows the probability distribution function and median values for Group 4 for hyperparameter optimization (for transfer learning, please refer to Figure S8). Table 7 compares the predicted values of bubble diameter with values from literature [85]. The results display very low deviations from the equivalent diameter as indicated by the high density in the PDF, as seen in Figure 21b. Of all the groups studied, Group 4 had the lowest deviation. Figure 22 shows the different output masks predicted by transfer learning (for a similar figure by HPO, please refer to

Figure S9). The contours were again well delimited. However, a small object on the left in Figure 22b is noticed. Despite being capable of differentiating some objects as background, circular objects' pixels could be misclassified as bubbles, which is a limitation of semantic segmentation. An object-detection algorithm must be coupled with the U-Net to avoid this.

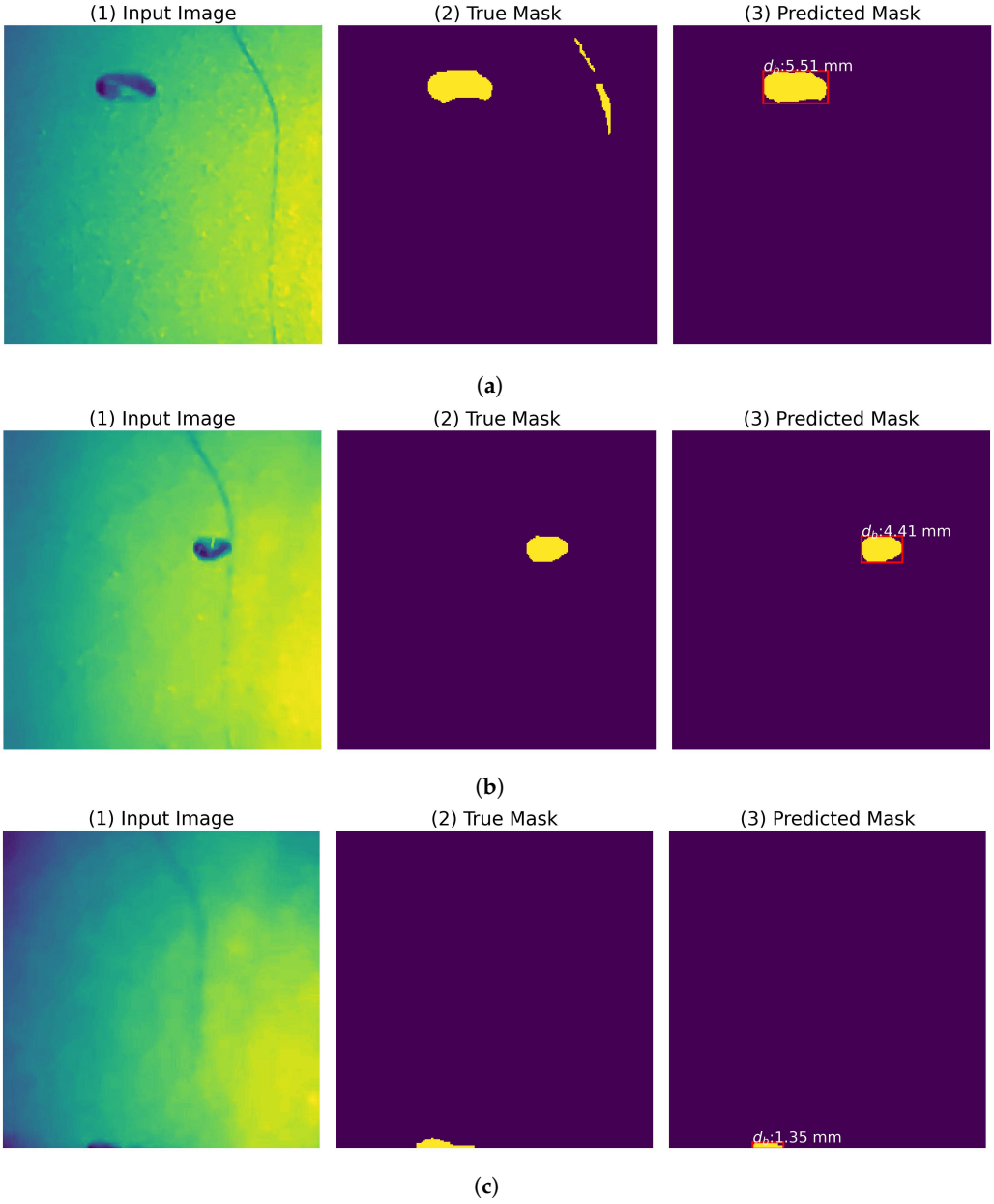
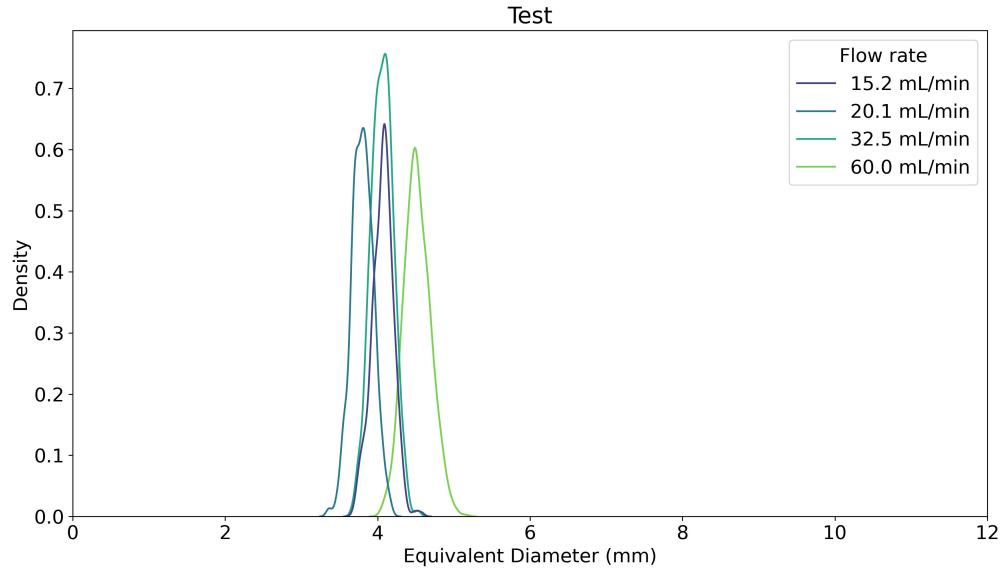
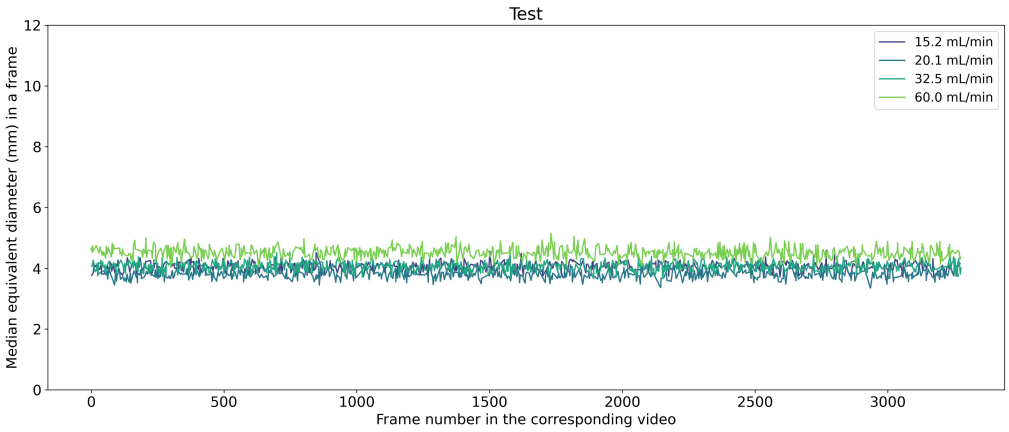


Figure 20. Input image, true mask, and predicted mask by hyperparameter optimization training in Group 3 in different conditions: (**a**) $d_i = 0.5$ mm and $Q = 51.0$ mL/min, (**b**) $d_i = 1.0$ mm and $Q = 23.5$ mL/min, (**c**) $d_i = 5.0$ mm, and $Q = 109.0$ mL/min. For animations, please check the Video S3 for Group 3.

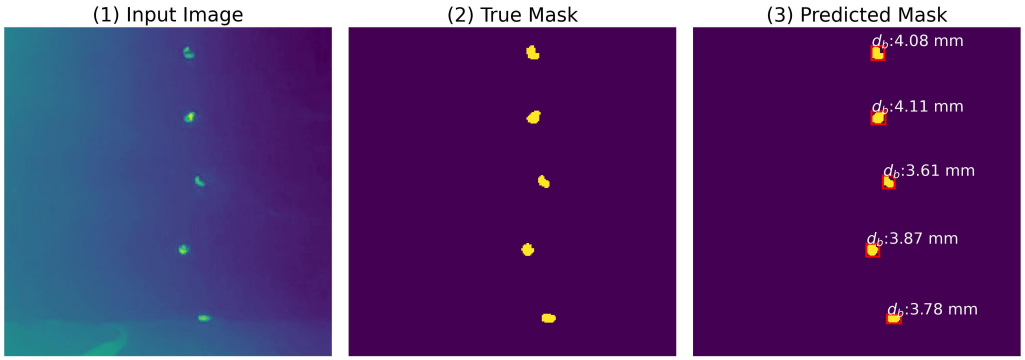


(**a**)



(**b**)

Figure 21. (**a**) Probability density functions and (**b**) median values of equivalent bubble diameter based on the median value of hyperparameter optimization for the test dataset.



(**a**)

Figure 22. *Cont.*

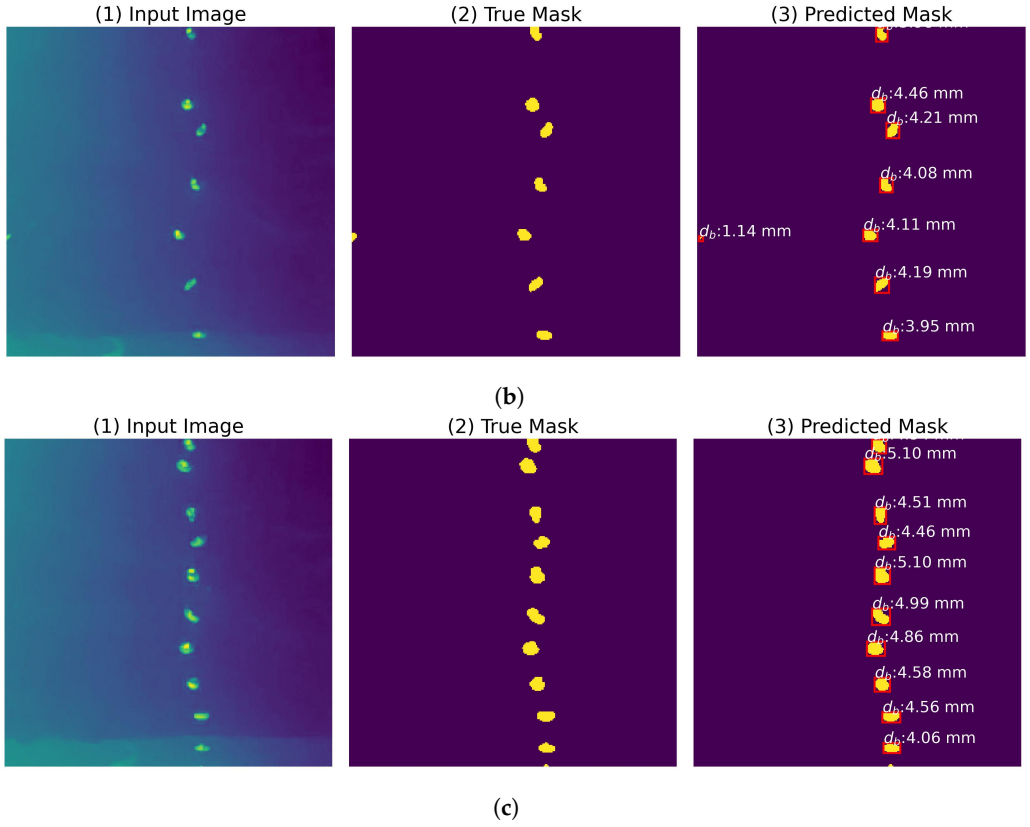


Figure 22. Input image, true mask, and predicted mask by transfer learning training in Group 4 in different conditions (d_i = 1.0 mm): (**a**) Q = 20.1 mL/min, (**b**) Q = 32.5 mL/min, (**c**) Q = 60.0 mL/min. The predicted mask is printed with the bubble diameter d_b (in mm). For animations, please check the Video S4 for group 4.

Table 7. Results of Group 4 against values for the hyperparameter optimization (HPO) and transfer learning (TL) against comparison with literature value by Jamialahmadi et al. [85].

ID	d_i (mm)	Q (mL/min)	v_o (m/s)	d_b (mm)	Median $\hat{d}_b$ HPO (mm)	Max. $\hat{d}_b$ HPO (mm)	Median $\hat{d}_b$ TL (mm)	Max. $\hat{d}_b$ TL (mm)
34	1.0	15.2	0.32	3.7	4.06 ± 0.14	4.27 ± 0.20	4.09 ± 0.14	4.30 ± 0.20
35		20.1	0.43	3.7	3.79 ± 0.15	4.11 ± 0.19	3.82 ± 0.15	4.14 ± 0.19
36		32.5	0.69	3.9	4.05 ± 0.14	4.55 ± 0.22	4.07 ± 0.14	4.55 ± 0.22
37		60.0	1.27	4.2	4.52 ± 0.17	5.37 ± 0.52	4.52 ± 0.17	5.35 ± 0.51

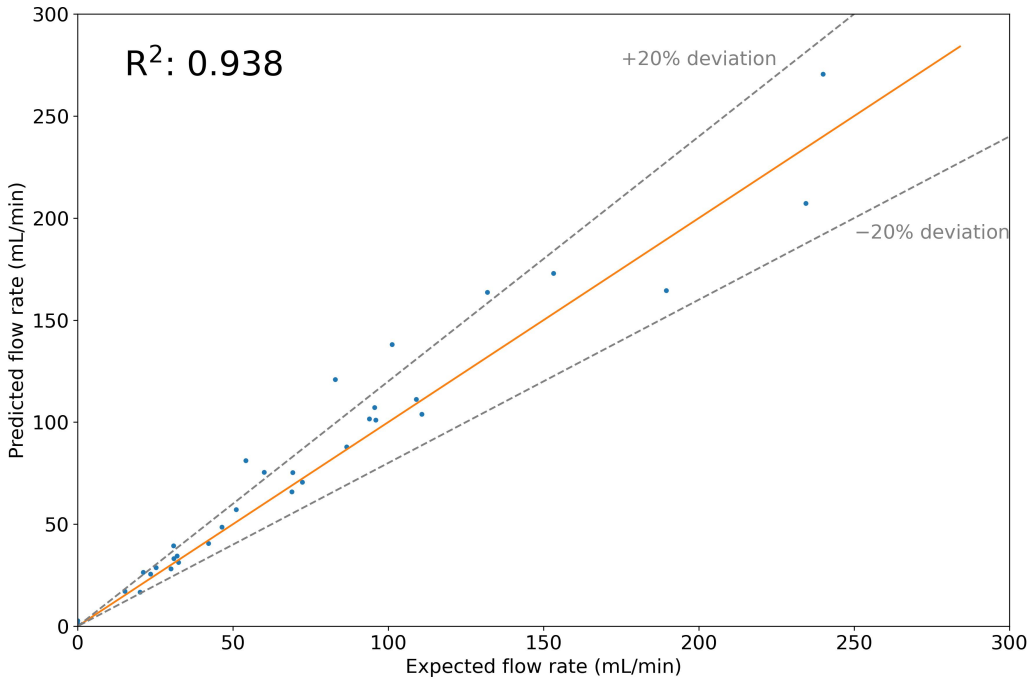
4.6. Flow Rate Prediction

Considering the fact that, in a frame, some bubbles can be in the time step of entering that particular frame, the algorithm can underestimate the bubble diameter for that specific bubble. Therefore, the maximum and median bubble diameter values of a frame were chosen as the representative statistics of this frame. Then, the average of the bubble diameter of each statistic value was computed in the different conditions of flow rate Q and orifice diameter d_i.

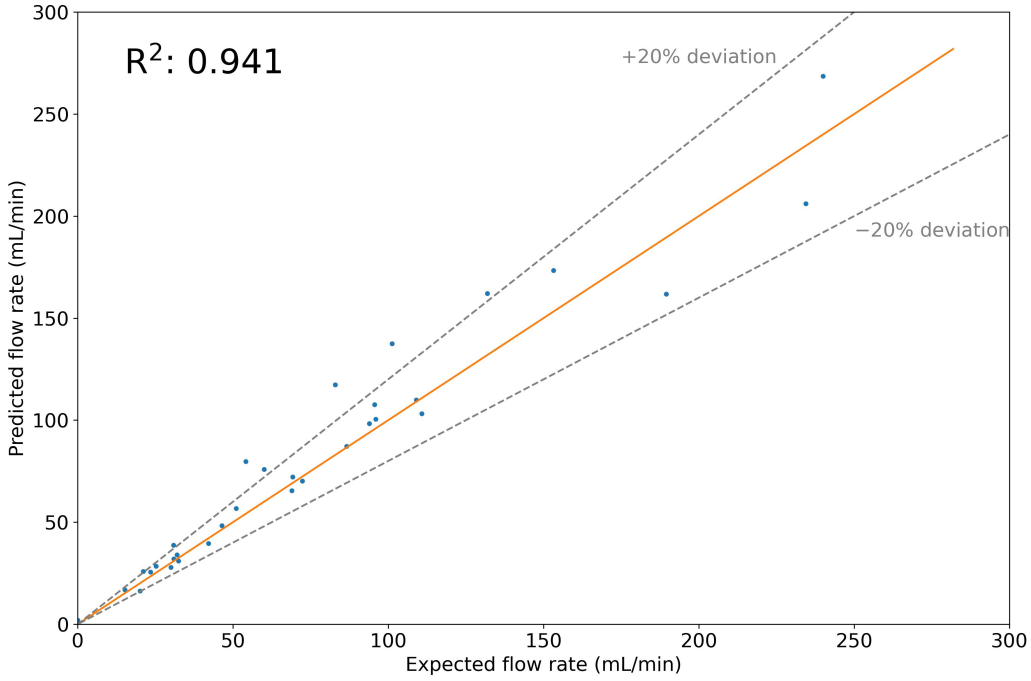
The predicted flow rate obtained by training CNN by transfer learning against the experimental ones is illustrated in Figure 23a. The values obtained by the algorithm correspond closely to the observed ones. Out of 35 points, only 8 are above the 20% deviation line.

As mentioned, trial # 58 was selected for training through the whole dataset for the hyperparameter optimization. The results of the predicted flow rate against the expected one are shown in Figure 23b. A good accuracy is found between them, with a coefficient

of determination of 0.941. In practical terms, 94% of the variance of the flow rate is explained by the choice of model parameters. This suggests a strong predictive capability OF THE model.



(**a**)



(**b**)

Figure 23. Flow rate: predicted vs. expected for the transfer learning and hyperparameter optimization.

The main limitation of the proposed study for practical applications is that it relies on having a measuring device on the seabed, such as a ruler, to convert pixels to physical quantities. Furthermore, the camera may need to be recalibrated if the distance to the leak point changes. Another issue is the presence of noise or low visibility, which could compromise the model's performance. Therefore, possible directions for future research

include testing under these conditions, coupling with other deep learning strategies, such as modifying the current U-Net structure [64] or by adding a denoising autoencoder before passing the input to the CNN. As mentioned, circular objects can mislead the U-Net's indication of bubbles.

5. Conclusions

In this work, we presented a methodology for measuring bubble diameter for subsea leaks. Besides that, we added several novelties to the literature, such as the automatic generation of ground truth data and hyperparameter optimization for U-Net. The results obtained are in line with the data obtained in the literature both by transfer learning and hyperparameter optimization. The algorithm uses the binarized image not only to calculate the area but also to generate outlines of where the bubble is present, along with a screen print of the diameter. For instance, in Group 1, the RMSE of the bubble diameter obtained was 0.90 mm, while in Group 3, the RMSE presented was 0.76 mm. The good prediction capacity is observed by the coefficient of determination for the flow rates of 0.94. Thus, measuring gas flow rates in the early stages can significantly enhance process safety, advancing the state of the art beyond merely detecting leaks. According to Figueredo et al. [3], most subsea accidents involve the release of gas fluid, which leads to loss of lives, explosions, fire, and environmental impact. In the case of Brazil, the authors pointed out the lack of integrity assessment by the operating companies after analyzing National Agency of Petroleum, Natural Gas, and Biofuels (ANP) reports. The present study helps to tackle this issue by a direct evaluation of gas leakages from images in the initial stage (minor leaks), which is a more proactive approach than remediation after propagation. Thus, the methodology is applicable to leaks in deep waters, and the application of the algorithm in the oil and gas Industry is immediate, supporting the decision-making for intervention. The algorithm can be incorporated into an alarm system for automatic monitoring, in which operators are only alerted when the leaked flow rate exceeds a certain threshold, reducing the need for continuous human monitoring oversight. The methodology could also be extended to other applications involving bubbly flows, such as bioreactors, or to other equipment with multiphase flows where the droplet size of the dispersed phase plays a critical role.

Furthermore, considering the transfer learning and the hyperparameter optimization, both approaches yielded similar results with a slightly superior performance for the latter. Hyperparameter optimization could be more computationally intensive, but it seeks the best possible outcome in terms of topology design. On the other hand, transfer learning is a potential route when a numerical simulation is available.

6. Patents

The authors declare that a patent is pending based on this work, whose assignees are Universidade Federal do Rio de Janeiro and Universidade Federal Fluminense.

Supplementary Materials: The following supporting information can be downloaded at https://www.mdpi.com/article/10.3390/pr13010118/s1. Figure S1: Snapshots extracted from video IDs with different flow rates; Figure S2: Comparison of bubble diameter values found by transfer learning with the experimental and literature values for different orifice diameter in Group 1: (a) $d_i = 1.0$ mm, (b) $d_i = 2.0$ mm, (c) $d_i = 5.0$ mm; Figure S3: Input image, true mask, and predicted mask by transfer learning training in Group 1 in different conditions: (a) $d_i = 1.0$ mm and $Q = 101.2$ mL/min, (b) $d_i = 2.0$ mm and $Q = 30.9$ mL/min, (c) $d_i = 5.0$ mm and $Q = 131.9$ mL/min; Figure S4: Median values of bubble equivalent diameter computed by transfer learning for the training dataset in different orifice diameter: (a) $d_i = 1.0$ mm, (b) $d_i = 2.0$ mm, (c) $d_i = 5.0$ mm; Figure S5: Probability density functions of bubble equivalent diameter based on median value of

transfer learning training for the training dataset in different orifice diameter: (a) $d_i = 1.0$ mm, (b) $d_i = 2.0$ mm, (c) $d_i = 5.0$ mm; Figure S6: Median values of bubble equivalent diameter computed by transfer learning for the training dataset in Group 2; Figure S7: Input image, true mask, and predicted mask by transfer learning training in Group 3 in different conditions: (a) $d_i = 0.5$ mm and $Q = 96.0$ mL/min, (b) $d_i = 1.0$ mm and $Q = 23.5$ mL/min, (c) $d_i = 5.0$ mm and $Q = 109.0$ mL/min; Figure S8: Probability density functions (a) and median values (b) of bubble equivalent diameter based on the median value of transfer learning training for the validation dataset for Group 4; Figure S9: Input image, true mask, and predicted mask by hyperparameter optimization in Group 4 in different conditions ($d_i = 1.0$ mm): (a) $Q = 15.2$ mL/min, (b) $Q = 32.5$ mL/min, (c) $Q = 60.0$ mL/min. Table S1: Reduced model; Table S2: Image processing methods used for generating data for CNNs and respective conversion factors fc, organized by video ID; Video S1: application of flow rate prediction algorithm to video #15 with d_i = 2.0 mm and Q = 110.8 mL/min by hyperparameter optimization. The average predicted flow rate was 103.3 mL/min; Video S2: application of flow rate prediction algorithm to video #33 with d_i = 1.0 mm by transfer learning. The average predicted flow rate was 0.44 mL/min; Video S3: application of flow rate prediction algorithm to video #03 with d_i = 5.0 mm and Q = 109.0 mL/min by hyperparameter optimization. The average predicted flow rate was 109.9 mL/min; Video S4: application of flow rate prediction algorithm to video #36 with d_i = 1.0 mm and Q = 32.5 mL/min by transfer learning. The average predicted flow rate was 31.3 mL/min.

Author Contributions: Conceptualization, G.L.R.C., R.M.M. and M.B.d.S.J.; methodology, G.L.R.C.; software, G.L.R.C.; validation, G.L.R.C.; formal analysis, G.L.R.C.; investigation, G.L.R.C.; resources, R.M.M. and M.B.d.S.J.; data curation, G.L.R.C.; writing—original draft preparation, G.L.R.C.; writing—review and editing, G.L.R.C., R.M.M. and M.B.d.S.J.; visualization, G.L.R.C.; supervision, R.M.M. and M.B.d.S.J.; project administration, R.M.M. and M.B.d.S.J.; funding acquisition, R.M.M. and M.B.d.S.J. All authors have read and agreed to the published version of the manuscript.

Funding: Gustavo L.R. Caldas acknowledges the financial support from the Human Resources Program of the Brazilian National Agency of Petroleum, Natural Gas, and Biofuels—PRH-ANP, funded by oil companies qualified in the R&D and Innovation clause from ANP Resolution 50/2015 [process no. 041319] and from the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior–Brasil (CAPES)–Finance Code 001 [Grant No. 88887.911081/2023-00]. Maurício B. de Souza Jr. is grateful to the financial support from Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) [Grant No. 311153/2021-6] and Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ) [Grant No. E-26/201.148/2022].

Data Availability Statement: Dataset available on request from the authors.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ANN	Artificial neural networks
BO	Bayesian optimization
CFD	Computational Fluid Dynamics
CNN	Convolutional Neural Network
HPO	Hyperparameter optimization
ML	Machine learning
PDF	Probability density function
TL	Transfer learning

References

1. Olsen, J.E.; Skjetne, P. Current Understanding of Subsea Gas Release: A Review. *Can. J. Chem. Eng.* **2016**, *94*, 209–219. [CrossRef]
2. Ho, M.; El-Borgi, S.; Patil, D.; Song, G. Inspection and Monitoring Systems Subsea Pipelines: A Review Paper. *Struct. Health Monit.* **2020**, *19*, 606–645. [CrossRef]

3. Figueredo, A.K.M.; Coelho, D.G.; Miranda, P.P.; de Souza Junior, M.B.; Frutuoso e Melo, P.F.F.; Vaz Junior, C.A. Subsea Pipelines Incidents Prevention: A Case Study in Brazil. *J. Loss Prev. Process Ind.* **2023**, *83* , 105007. [CrossRef]
4. Adegboye, M.A.; Fung, W.K.; Karnik, A. Recent Advances in Pipeline Monitoring and Oil Leakage Detection Technologies: Principles and Approaches. *Sensors* **2019**, *19*, 2548. [CrossRef]
5. Zhang, W.; Zhou, T.; Li, J.; Xu, C. An Efficient Method for Detection and Quantitation of Underwater Gas Leakage Based on a 300-kHz Multibeam Sonar. *Remote Sens.* **2022**, *14*, 4301. [CrossRef]
6. Zhang, Y.; Feng, Z.; Rui, X.; Wang, B.; Feng, H.; Huang, X. Underwater Gas Flow Measurement Based on Adaptive Passive Acoustic Characteristic Frequency Extraction. *Chem. Eng. Sci.* **2021**, *240*, 116663. [CrossRef]
7. Wang, X.; Jiao, J.; Yin, J.; Zhao, W.; Han, X.; Sun, B. Underwater Sonar Image Classification Using Adaptive Weights Convolutional Neural Network. *Appl. Acoust.* **2019**, *146*, 145–154. [CrossRef]
8. Murvay, P.S.; Silea, I. A Survey on Gas Leak Detection and Localization Techniques. *J. Loss Prev. Process Ind.* **2012**, *25*, 966–973. [CrossRef]
9. Idachaba, F.; Tomomewo, O. Surface Pipeline Leak Detection Using Realtime Sensor Data Analysis. *J. Pipeline Sci. Eng.* **2022**, *3*, 100108. [CrossRef]
10. Vrålstad, T.; Melbye, A.G.; Carlsen, I.M.; Llewelyn, D. Comparison of Leak-Detection Technologies for Continuous Monitoring of Subsea-Production Templates. *SPE Proj. Facil. Constr.* **2011**, *6*, 96–103. [CrossRef]
11. Gonzalez, R.C.; Woods, R.E. *Digital Image Processing*, 4th ed.; Pearson: New York, NY, USA, 2018.
12. Zhang, S.; Guan, Q. Development of an Elevated Flare Monitor Using Video Image Processing Technique. *IOP Conf. Ser. Earth Environ. Sci.* **2018**, *199*, 032058. [CrossRef]
13. Miguel, R.B.; Talebi-Moghaddam, S.; Zamani, M.; Turcotte, C.; Daun, K.J. Assessing Flare Combustion Efficiency Using Imaging Fourier Transform Spectroscopy. *J. Quant. Spectrosc. Radiat. Transfer* **2021**, *273*, 107835. [CrossRef]
14. Grossi, C.D.; Barbosa, V.P.; Gedraite, R.; De Souza, M.B.; Scheid, C.M.; Calçada, L.A.; Da Cruz Meleiro, L.A. Monitoring of the Drilling Region in Oil Wells Using a Convolutional Neural Network. In *Computer Aided Chemical Engineering*; Elsevier: Amsterdam, The Netherlands, 2023; Volume 52, pp. 1353–1358. [CrossRef]
15. Zhou, Y.; Doan, X.T.; Srinivasan, R. Real-Time Imaging and Product Quality Characterization for Control of Particulate Processes. In *Computer Aided Chemical Engineering;* Marquardt, W., Pantelides, C., Eds.; Elsevier: Amsterdam, The Netherlands, 2006; Volume 21, pp. 775–780. [CrossRef]
16. Chen, W.; Huang, G.; Hu, Y.; Yin, J.; Wang, D. Experimental Study on Continuous Spectrum Bubble Generator with a New Overlapping Bubbles Image Processing Technique. *Chem. Eng. Sci.* **2022**, *254*, 117613. [CrossRef]
17. Wang, B.; Socolofsky, S.A. A Deep-Sea, High-Speed, Stereoscopic Imaging System for in Situ Measurement of Natural Seep Bubble and Droplet Characteristics. *Deep Sea Res. Part I* **2015**, *104*, 134–148. [CrossRef]
18. Wang, B.; Socolofsky, S.A.; Breier, J.A.; Seewald, J.S. Observations of Bubbles in Natural Seep Flares at MC 118 and GC 600 Using in Situ Quantitative Imaging. *J. Geophys. Res.* **2016**, *121*, 2203–2230. [CrossRef]
19. Razaz, M.; Di Iorio, D.; Wang, B.; Daneshgar Asl, S.; Thurnherr, A.M. Variability of a Natural Hydrocarbon Seep and Its Connection to the Ocean Surface. *Sci. Rep.* **2020**, *10*, 12654. [CrossRef] [PubMed]
20. Di, P.; Feng, D.; Tao, J.; Chen, D. Using Time-Series Videos to Quantify Methane Bubbles Flux from Natural Cold Seeps in the South China Sea. *Minerals* **2020**, *10*, 216. [CrossRef]
21. She, M.; Weiß, T.; Song, Y.; Urban, P.; Greinert, J.; Köser, K. Marine Bubble Flow Quantification Using Wide-Baseline Stereo Photogrammetry. *ISPRS J. Photogramm. Remote Sens.* **2022**, *190*, 322–341. [CrossRef]
22. Liu, J.; Kuang, W.; Liu, J.; Gao, Z.; Rohani, S.; Gong, J. In-Situ Multi-Phase Flow Imaging for Particle Dynamic Tracking and Characterization: Advances and Applications. *Chem. Eng. J.* **2022**, *438*, 135554. [CrossRef]
23. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105. [CrossRef]
24. LeCun, Y.; Boser, B.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.; Jackel, L.D. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Comput.* **1989**, *1*, 541–551. [CrossRef]
25. LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
26. Shi, J.; Chang, Y.; Xu, C.; Khan, F.; Chen, G.; Li, C. Real-Time Leak Detection Using an Infrared Camera and Faster R-CNN Technique. *Comput. Chem. Eng.* **2020**, *135*, 106780. [CrossRef]
27. Li, L.; Jia, X.; Tian, W.; Sun, S.; Cao, W. Convolution Neural Network Based Chemical Leakage Identification. In *Computer Aided Chemical Engineering*; Elsevier: Amsterdam, The Netherlands, 2018; Volume 44, pp. 2329–2334. [CrossRef]
28. Sun, X.; Shi, J.; Liu, L.; Dong, J.; Plant, C.; Wang, X.; Zhou, H. Transferring Deep Knowledge for Object Recognition in Low-quality Underwater Videos. *Neurocomputing* **2018**, *275*, 897–908. [CrossRef]
29. Stamoulakatos, A.; Cardona, J.; McCaig, C.; Murray, D.; Filius, H.; Atkinson, R.; Bellekens, X.; Michie, C.; Andonovic, I.; Lazaridis, P.; et al. Automatic Annotation of Subsea Pipelines Using Deep Learning. *Sensors* **2020**, *20*, 674. [CrossRef]

30. Souza, A.C.O.; De Souza, M.B., Jr.; Da Silva, F.V. Development of a CNN-based Fault Detection System for a Real Water Injection Centrifugal Pump. *Expert Syst. Appl.* **2024**, *244*, 122947. [CrossRef]
31. Guo, P.; Zheng, S.; Yan, J.; Xu, Y.; Li, J.; Ma, J.; Sun, S. Leak detection in water supply pipeline with small-size leakage using deep learning networks. *Process Saf. Environ. Prot.* **2024**, *191*, 2712–2724. [CrossRef]
32. Wu, H.; Zhao, J. Deep Convolutional Neural Network Model Based Chemical Process Fault Diagnosis. *Comput. Chem. Eng.* **2018**, *115*, 185–197. [CrossRef]
33. Lei, W.; Luo, J.; Hou, F.; Xu, L.; Wang, R.; Jiang, X. Underground Cylindrical Objects Detection and Diameter Identification in GPR B-Scans via the CNN-LSTM Framework. *Electronics* **2020**, *9*, 1804. [CrossRef]
34. Zhou, M.; Yang, Y.; Xu, Y.; Hu, Y.; Cai, Y.; Lin, J.; Pan, H. A Pipeline Leak Detection and Localization Approach Based on Ensemble TL1DCNN. *IEEE Access* **2021**, *9*, 47565–47578. [CrossRef]
35. Poletaev, I.; Tokarev, M.P.; Pervunin, K.S. Bubble Patterns Recognition Using Neural Networks: Application to the Analysis of a Two-Phase Bubbly Jet. *Int. J. Multiphase Flow* **2020**, *126*, 103194. [CrossRef]
36. Ilonen, J.; Juránek, R.; Eerola, T.; Lensu, L.; Dubská, M.; Zemčík, P.; Kälviäinen, H. Comparison of Bubble Detectors and Size Distribution Estimators. *Pattern Recognit. Lett.* **2018**, *101*, 60–66. [CrossRef]
37. Bazai, H.; Kargar, E.; Mehrabi, M. Using an Encoder-Decoder Convolutional Neural Network to Predict the Solid Holdup Patterns in a Pseudo-2d Fluidized Bed. *Chem. Eng. Sci.* **2021**, *246*, 116886. [CrossRef]
38. Thuerey, N.; Weissenow, K.; Prantl, L.; Hu, X. Deep Learning Methods for Reynolds-Averaged Navier-Stokes Simulations of Airfoil Flows. *AIAA J.* **2020**, *58*, 25–36. [CrossRef]
39. Caldas, G.L.R.; Bento, T.; Moreira, R.M.; Bezerra de Souza Júnior, M. Detection of Subsea Gas Leakages via Computational Fluid Dynamics and Convolutional Neural Networks. In Proceedings of the 26th International Congress of Mechanical Engineering, Online, 22–26 November 2021. [CrossRef]
40. Caldas, G.L.; Bento, T.F.; Moreira, R.M.; de Souza, M.B. Quantifying Subsea Gas Leakages Using Machine Learning: A CFD-based Study. In *Computer Aided Chemical Engineering*; Elsevier: Amsterdam, The Netherlands, 2022; Volume 49, pp. 1345–1350. [CrossRef]
41. Kopbayev, A.; Khan, F.; Yang, M.; Halim, S.Z. Gas leakage detection using spatial and temporal neural network model. *Process Saf. Environ. Prot.* **2022**, *160*, 968–975. [CrossRef]
42. Han, X.; Zhu, J.; Li, H.; Xu, W.; Feng, J.; Hao, L.; Wei, H. Deep learning-based dispersion prediction model for hazardous chemical leaks using transfer learning. *Process Saf. Environ. Prot.* **2024**, *188*, 363–373. [CrossRef]
43. E Souza, A.C.O.; De Souza, M.B., Jr.; Da Silva, F.V. Enhancing Fault Detection and Diagnosis Systems for a Chemical Process: A Study on Convolutional Neural Networks and Transfer Learning. *Evol. Syst.* **2024**, *15*, 611–633. [CrossRef]
44. Vermeire, F.H.; Green, W.H. Transfer Learning for Solvation Free Energies: From Quantum Chemistry to Experiments. *Chem. Eng. J.* **2021**, *418*, 129307. [CrossRef]
45. Ureel, Y.; Vermeire, F.H.; Sabbe, M.K.; Van Geem, K.M. Beyond Group Additivity: Transfer Learning for Molecular Thermochemistry Prediction. *Chem. Eng. J.* **2023**, *472*, 144874. [CrossRef]
46. Sadoune, H.; Rihani, R.; Marra, F.S. DNN Model Development of Biogas Production from an Anaerobic Wastewater Treatment Plant Using Bayesian Hyperparameter Optimization. *Chem. Eng. J.* **2023**, *471*, 144671. [CrossRef]
47. Khaydarov, V.; Heinze, S.; Graube, M.; Knupfer, A.; Knespel, M.; Merkelbach, S.; Urbas, L. From Stirring to Mixing: Artificial Intelligence in the Process Industry. In Proceedings of the 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vienna, Austria, 8–11 September 2020; pp. 967–974. [CrossRef]
48. Xiang, Z.; Xie, B.; Fu, R.; Qian, M. Advanced Deep Learning-Based Bubbly Flow Image Generator under Different Superficial Gas Velocities. *Ind. Eng. Chem. Res.* **2022**, *61*, 1531–1543. [CrossRef]
49. Rutkowski, G.P.; Azizov, I.; Unmann, E.; Dudek, M.; Grimes, B.A. Microfluidic Droplet Detection via Region-Based and Single-Pass Convolutional Neural Networks with Comparison to Conventional Image Analysis Methodologies. *Mach. Learn. Appl.* **2022**, *7*, 100222. [CrossRef]
50. Zhu, H.; Xie, W.; Li, J.; Shi, J.; Fu, M.; Qian, X.; Zhang, H.; Wang, K.; Chen, G. Advanced Computer Vision-Based Subsea Gas Leaks Monitoring: A Comparison of Two Approaches. *Sensors* **2023**, *23*, 2566. [CrossRef] [PubMed]
51. Hu, S.; Feng, A.; Shi, J.; Li, J.; Khan, F.; Zhu, H.; Chen, J.; Chen, G. Underwater Gas Leak Detection Using an Autonomous Underwater Vehicle (Robotic Fish). *Process Saf. Environ. Prot.* **2022**, *167*, 89–96. [CrossRef]
52. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Proceedings of the Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015, Munich, Germany, 5–9 October 2015; Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F., Eds.; Springer: Cham, Switzerland, 2015; pp. 234–241.
53. Schäfer, J.; Schmitt, P.; Hlawitschka, M.W.; Bart, H.J. Measuring Particle Size Distributions in Multiphase Flows Using a Convolutional Neural Network. *Chem. Ing. Tech.* **2019**, *91*, 1688–1695. [CrossRef]

54. Cerqueira, R.F.L.; Perissinotto, R.M.; Verde, W.M.; Biazussi, J.L.; de Castro, M.S.; Bannwart, A.C. Development and Assessment of a Particle Tracking Velocimetry (PTV) Measurement Technique for the Experimental Investigation of Oil Drops Behaviour in Dispersed Oil–Water Two-Phase Flow within a Centrifugal Pump Impeller. *Int. J. Multiphase Flow* **2023**, *159*, 104302. [CrossRef]
55. Bergau, M.; Strahl, T.; Ludlum, K.; Scherer, B.; Wöllenstein, J. Flow rate quantification of small methane leaks using laser spectroscopy and deep learning. *Process Saf. Environ. Prot.* **2024**, *182*, 752–759. [CrossRef]
56. Bankman, I.N. *Handbook of Medical Imaging: Processing and Analysis*; Academic Press Series in Biomedical Engineering; Academic Press: San Diego, CA, USA, 2000.
57. Otsu, N. A Threshold Selection Method from Gray-Level Histograms. *IEEE Trans. Syst. Man, Cybern.* **1979**, *9*, 62–66. [CrossRef]
58. Szeliski, R. *Computer Vision: Algorithms and Applications*, 2nd ed.; Springer Nature: Cham, Switzerland, 2022.
59. Russell, S.J.; Norvig, P. *Artificial Intelligence: A Modern Approach*, 4th ed.; Pearson Series in Artificial Intelligence; Pearson: Hoboken, NJ, USA, 2021.
60. Krogh, A. What Are Artificial Neural Networks? *Nat. Biotechnol.* **2008**, *26*, 195–197. [CrossRef] [PubMed]
61. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; Adaptive Computation and Machine Learning; The MIT Press: Cambridge, MA, USA, 2016.
62. Liu, B.; Zou, D.; Feng, L.; Feng, S.; Fu, P.; Li, J. An FPGA-Based CNN Accelerator Integrating Depthwise Separable Convolution. *Electronics* **2019**, *8*, 281. [CrossRef]
63. Horwath, J.P.; Zakharov, D.N.; Mégret, R.; Stach, E.A. Understanding Important Features of Deep Learning Models for Segmentation of High-Resolution Transmission Electron Microscopy Images. *npj Comput. Mater.* **2020**, *6*, 108. [CrossRef]
64. Komatsu, R.; Gonsalves, T. Comparing U-Net Based Models for Denoising Color Images. *AI* **2020**, *1*, 465–487. [CrossRef]
65. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceedings of the ICML'15: 32nd International Conference on International Conference on Machine Learning, Lille, France, 6–11 July 2015; Volume 37, pp. 448–456.
66. Santurkar, S.; Tsipras, D.; Ilyas, A.; Madry, A. How Does Batch Normalization Help Optimization? *arXiv* **2019**, arXiv:1805.11604. [CrossRef]
67. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
68. Smith, L.N. A Disciplined Approach to Neural Network Hyper-Parameters: Part 1—Learning Rate, Batch Size, Momentum, and Weight Decay. *arXiv* **2018**, arXiv:1803.09820. [CrossRef]
69. Wu, J.; Chen, X.Y.; Zhang, H.; Xiong, L.D.; Lei, H.; Deng, S.H. Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimizationb. *J. Electron. Sci. Technol.* **2019**, *17*, 26–40. [CrossRef]
70. Feurer, M.; Hutter, F. Hyperparameter Optimization. In *Automated Machine Learning: Methods, Systems, Challenges*; Hutter, F., Kotthoff, L., Vanschoren, J., Eds.; The Springer Series on Challenges in Machine Learning; Springer International Publishing: Cham, Switzerland, 2019; pp. 3–33. [CrossRef]
71. Bergstra, J.; Bengio, Y. Random Search for Hyper-Parameter Optimization. *J. Mach. Learn. Res.* **2012**, *13*, 281–305.
72. Sicard, D.; Briois, P.; Billard, A.; Thevenot, J.; Boichut, E.; Chapellier, J.; Bernard, F. Deep Learning and Bayesian Hyperparameter Optimization: A Data-Driven Approach for Diamond Grit Segmentation toward Grinding Wheel Characterization. *Appl. Sci.* **2022**, *12*, 12606. [CrossRef]
73. Bergstra, J.; Bardenet, R.; Bengio, Y.; Kégl, B. Algorithms for Hyper-Parameter Optimization. In Proceedings of the NIPS'11: 24th International Conference on Neural Information Processing Systems, Granada, Spain, 12–15 December 2011; Curran Associates Inc.: Red Hook, NY, USA, 2011; pp. 2546–2554.
74. Bergstra, J.; Yamins, D.; Cox, D.D. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. In Proceedings of the ICML'13: 30th International Conference on International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013; Volume 28, pp. I–115–I–123.
75. Falkner, S.; Klein, A.; Hutter, F. BOHB: Robust and Efficient Hyperparameter Optimization at Scale. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; Dy, J., Krause, A., Eds.; PMLR Proceedings of Machine Learning Research: Cambridge, MA, USA, 2018; Volume 80, pp. 1437–1446.
76. Sansana, J.; Rendall, R.; Castillo, I.; Chiang, L.; Reis, M.S. Hybrid Modeling for Improved Extrapolation and Transfer Learning in the Chemical Processing Industry. *Chem. Eng. Sci.* **2024**, *300*, 120568. [CrossRef]
77. Nascimento, G.D.C.; Moreira, R.M.; Moura, F.P.D.; Tavares, W.A.; Bento, T.F.B.; Calazan, L.B.; Andrade, M.S.; Rezende, B.F. Analysis and Prediction of Equivalent Diameter of Air Bubbles Rising in Water. *CFD Lett.* **2024**, *16*, 33–47. [CrossRef]
78. Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; Koyama, M. Optuna: A Next-generation Hyperparameter Optimization Framework. In Proceedings of the KDD'19: 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 2623–2631. [CrossRef]
79. Popescu, D.; Stanciulescu, A.; Pomohaci, M.D.; Ichim, L. Decision Support System for Liver Lesion Segmentation Based on Advanced Convolutional Neural Network Architectures. *Bioengineering* **2022**, *9*, 467. [CrossRef]

80. Genc, A.; Kovarik, L.; Fraser, H.L. A Deep Learning Approach for Semantic Segmentation of Unbalanced Data in Electron Tomography of Catalytic Materials. *Sci. Rep.* **2022**, *12*, 16267. [CrossRef]
81. van der Walt, S.; Schönberger, J.L.; Nunez-Iglesias, J.; Boulogne, F.; Warner, J.D.; Yager, N.; Gouillart, E.; Yu, T.; the Scikit-Image Contributors. Scikit-Image: Image Processing in Python. *PeerJ* **2014**, *2*, e453. [CrossRef] [PubMed]
82. Chollet, F. Keras. 2015. Available online: https://keras.io (accessed on 1 September 2022).
83. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-scale Machine Learning on Heterogeneous Systems, 2016. *arXiv* **2016**, arXiv:1603.04467. [CrossRef]
84. Hutter, F.; Hoos, H.; Leyton-Brown, K. An Efficient Approach for Assessing Hyperparameter Importance. In Proceedings of the 31st International Conference on Machine Learning, Bejing, China, 21–26 June 2014; Xing, E.P., Jebara, T., Eds.; Proceedings of Machine Learning Research: Cambridge, MA, USA, 2014; Volume 32, pp. 754–762.
85. Jamialahmadi, M.; Zehtaban, M.R.; Müller-Steinhagen, H.; Sarrafi, A.; Smith, J.M. Study of Bubble Formation Under Constant Flow Conditions. *Chem. Eng. Res. Des.* **2001**, *79*, 523–532. [CrossRef]
86. Akita, K.; Yoshida, F. Bubble Size, Interfacial Area, and Liquid-Phase Mass Transfer Coefficient in Bubble Columns. *Ind. Eng. Chem. Process Des. Dev.* **1974**, *13*, 84–91. [CrossRef]
87. Gaddis, E.; Vogelpohl, A. Bubble Formation in Quiescent Liquids under Constant Flow Conditions. *Chem. Eng. Sci.* **1986**, *41*, 97–105. [CrossRef]
88. Tomiyama, A.; Celata, G.P.; Hosokawa, S.; Yoshida, S. Terminal Velocity of Single Bubbles in Surface Tension Force Dominant Regime. *Int. J. Multiphase Flow* **2002**, *28*, 1497–1519. [CrossRef]
89. Silverman, B.W. *Density Estimation for Statistics and Data Analysis*; Number 26 in Monographs on Statistics and Applied Probability; Chapman & Hall/CRC: Boca Raton, FL, USA, 1998.
90. Lu, X.; Zheng, X.; Ding, Y.; Lin, W.; Wang, W.; Yu, J. Experimental Study on the Influence of the Orifice Size on Hydrodynamic Characteristics and Bubble Size Distribution of an External Loop Airlift Reactor. *Can. J. Chem. Eng.* **2020**, *98*, 1593–1606. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.