Article

Real-Time Droplet Detection for Agricultural Spraying Systems: A Deep Learning Approach

Nhut Huynh [†] and Kim-Doang Nguyen [*,†]

Department of Mechanical and Civil Engineering, Florida Institute of Technology, Melbourne, FL 32901, USA; nhuynh2023@my.fit.edu

* Correspondence: knguyen@fit.edu

† These authors contributed equally to this work.

Abstract: Nozzles are ubiquitous in agriculture: they are used to spray and apply nutrients and pesticides to crops. The properties of droplets sprayed from nozzles are vital factors that determine the effectiveness of the spray. Droplet size and other characteristics affect spray retention and drift, which indicates how much of the spray adheres to the crop and how much becomes chemical runoff that pollutes the environment. There is a critical need to measure these droplet properties to improve the performance of crop spraying systems. This paper establishes a deep learning methodology to detect droplets moving across a camera frame to measure their size. This framework is compatible with embedded systems that have limited onboard resources and can operate in real time. The method leverages a combination of techniques including resizing, normalization, pruning, detection head, unified feature map extraction via a feature pyramid network, non-maximum suppression, and optimization-based training. The approach is designed with the capability of detecting droplets of various sizes, shapes, and orientations. The experimental results demonstrate that the model designed in this study, coupled with the right combination of dataset and augmentation, achieved a 97% precision and 96.8% recall in droplet detection. The proposed methodology outperformed previous models, marking a significant advancement in droplet detection for precision agriculture applications.

Keywords: agricultural nozzles; droplet properties; deep learning; real-time detection; mobile platform; YOLOv8; precision agriculture

check for updates

Citation: Huynh, N.; Nguyen, K.-D. Real-Time Droplet Detection for Agricultural Spraying Systems: A Deep Learning Approach. *Mach. Learn. Knowl. Extr.* **2024**, *6*, 259–282. https://doi.org/10.3390/make6010014

Academic Editor: Ramón Alberto Mollineda Cárdenas

Received: 1 December 2023
Revised: 21 January 2024
Accepted: 23 January 2024
Published: 26 January 2024

1. Introduction

Spray nozzles play a crucial role in agricultural practices as they are widely used for distributing pesticides, fertilizers, and other treatments to crops. To ensure the effectiveness of agricultural crop spraying systems, it is essential to determine the properties of the droplets sprayed from the nozzles, such as droplet size, velocity, and spray patterns [1]. These factors directly influence spray retention and drift, which, in turn, have significant impacts on both crop health and the environment [2]. The effectiveness of crop spraying systems and their treatment performance are intrinsically linked to the dimensions and speed of the sprayed droplets. Hence, precise and accurate measurements of droplet size and velocity are essential prerequisites to optimize nozzle design and enhance the overall efficiency of the spraying process. However, state-of-the-art methodologies for assessing these droplet properties suffer from several problems, including manual and complicated implementation, suboptimal accuracy, and excessive cost.

For instance, one simplistic approach to estimating droplet sizes involves dispersing colored liquid onto a white sheet, which is followed by a subsequent analysis of the resultant patterns to obtain approximate droplet size information. On the other end of the spectrum, a more sophisticated technique, known as the immersion sampling method, employs a silicone oil-coated glass plate to yield more precise measurements of droplet dimensions [3]. Nonetheless, the traditional methodologies employed for evaluating droplet properties generally involve intricate experimentation, thus incurring high expenses, as well as requiring specialized equipment and skilled personnel to operate [4,5].

To address these challenges, this research project proposes an innovative alternative approach grounded in recent advancements in the fields of computer vision and machine learning. Our proposed technique not only enables accurate measurements of droplet size, but also facilitates the real-time detection of droplets in video recordings. By integrating these technological innovations, our method offers promising advantages in terms of accessibility, cost-effectiveness, and reliability, which can significantly contribute to advancing agricultural spray nozzle design and optimizing treatment efficiency.

The methodology employed in this study for droplet detection is rooted in the principles of deep learning, and it is a powerful approach capable of real-time detection of droplet sizes. This droplet detection methodology aims to significantly improve the precision of droplet assessments, even under challenging conditions. A benchmark dataset for droplet detection is introduced in this paper, which facilitates the evaluation of the system's performance in real-world spraying scenarios. The obtained results substantiate the accuracy and efficiency of the proposed method, underscoring its potential to advance sustainable agricultural practices. Prior to delving into the proposed methodology, the following section provides a review of the existing state-of-the-art research in this domain and identifies their inherent limitations.

2. Literature Review

Particle image velocimetry (PIV) is the current advanced state-of-the-art methodology for measuring the flow of droplets; it is used to visualize and measure fluid velocities in a flow field [6]. This optical flow measurement technique involves introducing tiny particles into the flow, illuminating them with a laser, capturing consecutive images with a camera, and using image processing software to calculate velocity vectors based on the particle displacements. PIV provides non-intrusive, quantitative data that helps researchers and engineers gain valuable insights into fluid dynamics for various applications. Conventional computer vision techniques coupled with a complicated sensor system were also used for droplet detection on captured images [7–9]. However, these existing methods do not offer a solution for measuring the characteristics of individual droplets to estimate their properties. Our hypothesis is that computer vision integrated with recent advances in deep learning may fill this critical gap.

Object detection enabled by deep learning has emerged as a dynamic and intensely investigated domain, garnering notable attention in recent years owing to its remarkable practicality in diverse fields, such as computer vision, robotics, and autonomous vehicles. A plethora of studies have been undertaken in this realm, in which the efficacy of deep learning methodologies for object detection has been delved into. Specifically, the seminal work of Ren et al. (2015) [10] introduced a region-based convolutional neural network (R-CNN) that incorporates a region proposal network (RPN) to generate potential candidate object regions called Faster R-CNN, which they subsequently classified by a deep neural network. This pioneering approach attained state-of-the-art outcomes across various object detection benchmarks. Other notable approaches for object detection include the YOLO framework proposed in Redmon et al. (2016) [11], which uses a single neural network to predict bounding boxes and class probabilities directly from full images in real time. In addition to these approaches, several other deep learning-based object detection methods have been proposed, including the Single Shot MultiBox Detector (SSD) [12], RetinaNet [13], and Mask R-CNN [14]. These techniques have demonstrated superior performance in various benchmarks and have been applied in various applications.

Recent research in this field has focused on improving object detection accuracy and efficiency using deep learning. For example, researchers have proposed techniques to optimize deep learning models [15,16], explored alternative architectures for object detection [17], and studied the problem of object detection under challenging conditions, such as in occlusion and changing lighting conditions [18]. The framework developed by

Acharya et al. (2022) [19] was the first deep learning approach that aimed to address the problem of droplet detection to improve crop spraying systems. This pipeline incorporates four essential components, namely feature extraction, region proposal, pooling, and inference. These components are devised using deep neural networks, convolutional layers, and filters, which are proven to be highly effective in various computer vision applications.

In the context of droplet detection, the work of Wang et al. (2021) [9] presented a noteworthy development. Their research introduced a highly integrated and intelligent droplet detection sensor system utilizing the MobileNetSSD model. This model, designed for portability, is particularly adept in drone applications for detecting spray droplet depositions. The sensor system developed by Wang et al. incorporates a droplet deposition image loop acquisition device and a supporting host computer interactive platform. Notably, their approach uses an adaptive light model to counter the impact of environmental light on image recognition, thus enhancing field detection capabilities. The MobileNetSSD network, as revealed through their quantitative analysis, successfully balances accuracy and lightweight requirements, making it suitable for embedded devices. Subsequently, the work of Gardner et al. (2022) [20] stands out, where they employed YOLOv3 and YOLOv5 models for the high-precision monitoring of cell encapsulation in microfluidic droplets. Their approach, capable of processing over 1080 droplets per second, demonstrated the application of deep learning in high-throughput environments and addressed the challenge of class imbalance in training datasets. Their study underscored the versatility of deep learning models in analyzing microfluidic droplets, a context that is significantly different yet conceptually related to agricultural applications.

A notable advancement in the field is the work by Hasti and Shin (2022) [21]. They proposed a deep learning-based method for denoising and detecting fuel spray droplets from light-scattered images utilizing a modified UNet architecture. Their method represents a significant leap in addressing the challenges of noise and clarity in droplet detection. Their approach, which effectively differentiates droplets from noise, shows potential for the real-time processing of data, a critical aspect in dynamic fluid systems.

Recent advancements in medical imaging segmentation offer valuable insights that can be applied to other domains such as agricultural droplet detection. For instance, Wang et al. (2022) [22] developed AWSnet, a novel segmentation method for myocardial pathology in CMR data. This method combines reinforcement learning with a coarse-to-fine approach, demonstrating success in accurately delineating complex pathological regions. Such computational techniques may hold potential for enhancing segmentation tasks in agricultural settings. Yu et al. (2022) [23] addressed the challenges of varying object scales and blurred edges in medical imaging by developing a weakly supervised semantic segmentation method for thyroid ultrasound images. Their approach underscores the complexity and necessity of precision in segmentation tasks. Additionally, Zhou et al. (2023) [24] introduced DSANet, a dual-branch shape-aware network tailored for echocardiography segmentation in apical views. Their network employs shape-aware modules, including an anisotropic strip attention mechanism and cross-branch skip connections, to enhance feature representation and segmentation accuracy (particularly for cardiac structures such as the left ventricle and left atrium). The integration of a boundary-aware rectification module by Zhou et al. emphasizes the criticality of accurate boundary identification, a principle that could be advantageous in droplet detection. Furthermore, data augmentation techniques have gained significant attention in improving deep learning model performance in medical imaging. Guan et al. (2022) [25] and Guan et al. (2022) [26] demonstrated the effectiveness of a novel multichannel, progressive generative adversarial network with texture constraints for pancreas dataset augmentation. Their work highlighted the role of continuous texture and accurate lesion representation in enhancing detection performance, an approach that might be beneficial in the context of agricultural droplet detection.

Our research, while drawing inspiration from these studies, focuses on the agricultural domain, specifically targeting the detection of water droplets in crop spraying scenarios. We employed the YOLOv8 framework to create a robust, accurate, and efficient method for droplet detection. Our approach contrasts with Gardner et al.'s [20] focus on microfluidic droplets and Wang et al.'s emphasis on lightweight model deployment. We address the challenges in existing methodologies, such as occlusions and variations in droplet size and shape, with advanced deep learning techniques. Furthermore, we have developed an autonomous annotation method for large-scale droplet data and optimized our model for embedded systems, thereby extending the feasibility for real-time implementations in mobile crop spraying systems. In comparison to Acharya et al. (2022) [19], the key improvements of this work include the following:

- The droplet detector in [19] was designed to strike a balance between processing speed and achieving higher accuracy. In the current work, we show that it is possible to achieve even higher accuracy while reducing the processing speed.
- The data annotation in [19] was carried out manually. Manual labeling of droplet data is not practical in real-world settings, especially when dealing with large volumes of data. In this work, we developed a method for autonomous annotation of droplet data.
- The pipeline discussed in [19] can only operate with a high-performance computing cluster. In this work, we propose a model pruning technique to substantially reduce the number of parameters, thereby shortening the processing time and making it deployable in embedded systems. This opens up the feasibility for real-time implementations in mobile crop spraying systems.

Therefore, the main goal of this work is to create a more robust, accurate, compact, and faster inference pipeline for droplet detection.

Statement of Contributions

We leverage deep learning and computer vision to detect the motion of droplets from sprayed nozzles moving across camera frames in real time. Our algorithm is grounded in the recent progress in state-of-the-art detection methods, such as YOLOv8 and model pruning, to improve droplet detection precision and enable real-time operation. We address several challenges in the previous work of [19], including occlusion and variations in droplet size and shape. To overcome the occlusion problem, we implemented a robust algorithm that utilizes deep learning models capable of capturing and interpreting complex visual patterns, even in the presence of occlusions. By leveraging these methods, we are able to accurately detect droplets, even when they are partially obscured or hidden behind other droplets. Furthermore, we addressed the challenge of variations in droplet size and shape by developing a flexible and adaptable framework. Our approach incorporated adaptive learning mechanisms that enable the model to dynamically adjust its detection parameters based on the characteristics of the droplets observed in the input data. This allowed our system to effectively handle variations in droplet size and shape, ensuring accurate and reliable detection results across different scenarios and conditions.

The experimental data utilized in this study were collected through a dedicated setup featuring crop spraying nozzles, as illustrated in Figure 1. The data collection process was conducted via a user interface controller, which regulated the flow rate of the water pumped from the water storage tank to the nozzles. The setup comprises an array of nozzles capable of producing different spray patterns. To record droplet motion, an Olympus i-SPEED 3 camera (Tokyo, Japan) with a frame rate of 2000 fps and a lighting system were positioned beneath the nozzles. However, annotating the data posed a tremendous challenge because there were thousands of droplets to be labeled in order to construct a sufficiently large ground truth dataset for training. To address this challenge, this work proposes a method for automatic annotation, which results in an accurately labeled and substantial dataset.

The key contributions of this work are summarized as follows:

- Design and implementation of a real-time droplet detection approach that achieves a higher level of accuracy.
- An AI-enabled automatic labeling and augmentation tool for establishing big droplet datasets.

- The innovative modification of state-of-the-art models for droplet detection to optimize their performance with respect to speed and precision, thereby facilitating the practical implementation of our method.
- Overcoming the limitations associated with previous droplet detection frameworks, such as in [19], as discussed earlier.
- Implementing a framework on an onboard computer platform, Jetson Orin, as well as enabling advanced mobile robotics and edge AI capabilities for droplet analysis.

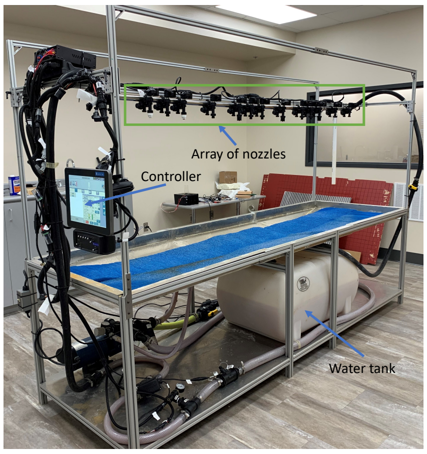

Figure 1. Experimental setup with crop spray nozzles for droplet data collection.

3. Real-Time Deep Learning-Based Droplet Detection

3.1. Overview of the Methodology

The methodology in this paper builds upon the recent architecture provided by YOLOv8 [27] to deliver a higher precision and faster processing rate compared to the state-of-the-art methods in droplet detection. The selection of YOLOv8 is underpinned by several key theoretical considerations:

1. Enhanced Speed and Real-time Processing: YOLOv8, as a single-stage detector, inherently offers faster processing speeds, which is crucial for real-time applications in agricultural spraying where immediate feedback is essential.
2. Robust Accuracy in Diverse Conditions: Our dataset includes complex backgrounds and varying lighting conditions. YOLOv8's architecture maintains high detection accuracy in these diverse conditions, which is an important consideration mirrored in our methodology.
3. Effective Detection of Small Objects: The precise detection of small droplets is a critical aspect of our study. YOLOv8 improves feature extraction and bounding box prediction mechanisms, thereby providing enhanced capabilities in detecting small droplets.

YOLOv8's design is particularly beneficial for detecting small, fast-moving objects like droplets as it effectively addresses challenges such as varying sizes and overlapping objects. The use of YOLOv8 allows for the integration of advanced features like multi-scale prediction and a more refined anchor box mechanism, which are vital for accurately identifying droplets in diverse conditions. In particular, the process is shown in Figure 2 and encompasses the following key steps:

1. Preprocessing: The input image is resized and normalized to values between 0 and 1.
2. Feature Extraction: A total of 53 convolutional layers were created to extract a series of feature maps, which were then unified via a feature pyramid network (FPN) for detecting droplets across various sizes.
3. Detection Head: The unified feature map was passed to the detection head, which is composed of convolutional and fully connected layers. It predicts bounding boxes, objectness scores, and class probabilities for droplets, which are respectively classified

into large, medium, and small sizes. Anchor boxes were also utilized here, which helped in identifying droplets with diverse dimensions.

4. Anchor Boxes: As seen in Figure 3, the anchor boxes were predefined to cover different droplet sizes and shapes, thus aiding in predicting precise locations and sizes.
5. Non-Maximum Suppression (NMS): Finally, NMS was applied to reduce overlapping bounding boxes, and only those with the highest objectness score within each overlapping region were retained.

The rest of the section will be dedicated to elaborating on the technical details of these steps.

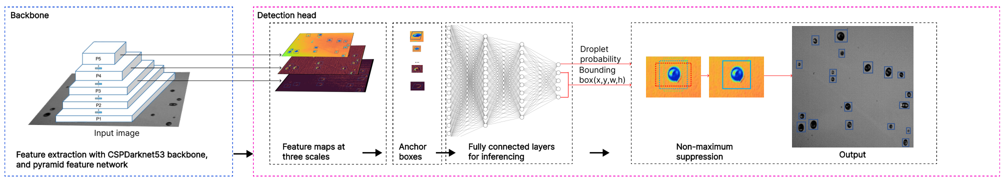
Backbone
P5
P4
P3
P2
P1
Input image
Feature extraction with CSPDarknet53 backbone, and pyramid feature network
Detection head
Feature maps at three scales
Anchor boxes
Fully connected layers for inferencing
Droplet probability
Bounding box(x,y,w,h)
Non-maximum suppression
Output

Figure 2. The framework of the droplet detector.

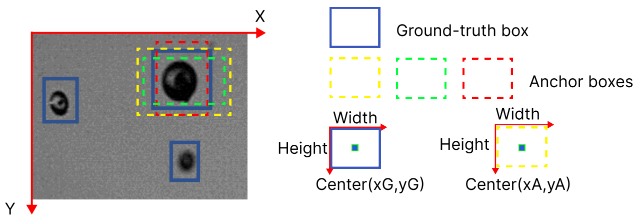

Figure 3. Anchor boxes of an input image.

The rest of the methodology section will delve deeper into the key aspects of the droplet detection framework.

3.2. Feature Extraction

The backbone network used in this work is composed of a series of convolutional layers that extract features from input images. These convolutional layers use filters or kernels to perform a series of convolutions on the input image. The output of each convolutional layer is a set of feature maps that represent different levels of abstraction in the input image.

- Early Layer Feature Maps: The initial layers predominantly focus on low-level features. These include the following:
    - Edges: Refers to the sharp changes in intensity or color in the image and marks the boundaries of objects within the field of view.
    - Textures: Patterns or repetitive arrangements found on the surface of objects in the image, often manifesting as fine details.
- Later Layer Feature Maps: Advanced layers capture more complex, high-level features. These encompass the following:
    - Shapes: Refers to the geometric or outline properties of objects, as well as captures the overall form and structure of the droplets.
    - Boundaries: A more holistic view of the contours of objects, and it also helps to distinguish the droplets from the background by recognizing their complete periphery.

To improve the efficiency of the feature extraction process, we used the CSPNet architecture [28] to split the input feature maps into two parallel paths, which were processed separately and then merged back together. This approach allowed us to reduce the number of computations required while still maintaining the accuracy of the predictions. The algorithm was designed in such a way that the feature extraction process was followed by a series of upsampling and concatenation operations to increase the resolution and combine the feature maps from different layers. The resulting feature maps were then used to predict bounding boxes and class probabilities for droplets in the image.

For example, when a droplet image of a size of 608 × 608 pixels is input, it goes through a series of convolutional layers. These layers gradually down sample the image to sizes of 304 × 304, 152 × 152, and finally 76 × 76 pixels. Furthermore, it then extracts features at different abstraction levels like edges, textures, shapes, and boundaries. The feature maps are subsequently processed through additional convolutional layers, restoring them to the original input dimensions of 608 × 608 pixels. These final feature maps are used to predict the bounding boxes and class probabilities for the droplets in an image. As demonstrated in Figure 4, this process of feature extraction produces clearly identifiable patterns. Figure 4 displays the feature extraction from a droplet image, in which the distinct patterns on the feature maps are highlighted. The left panel shows the early layer's feature maps, and it focuses on low-level, specific details like edges and corners with low spatial resolution. The right panel represents the last layer's feature maps, whereby more abstract, high-level features like shapes and boundaries with higher spatial resolution are captured. These varying feature maps contribute to tasks like droplet detection and classification, with clear distinctions between the early and final layers in terms of resolution, receptive field, and abstraction level being made.

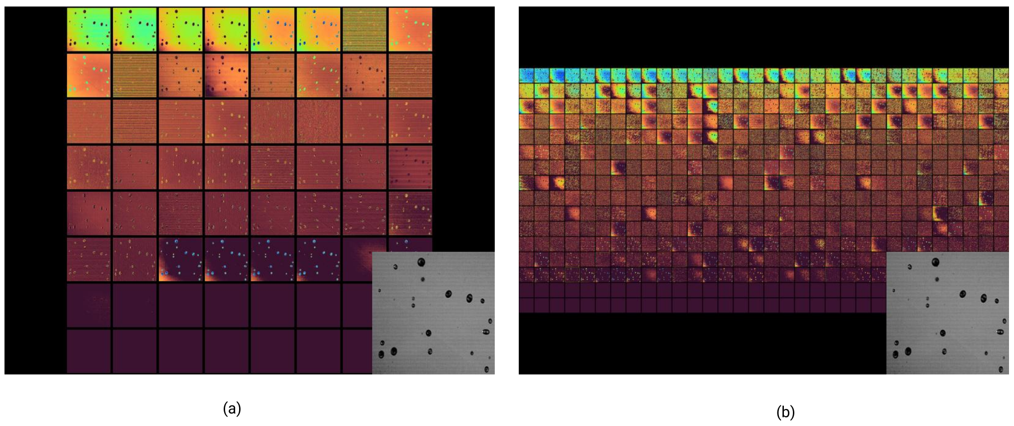

Figure 4. Examples of the feature maps obtained from the feature extraction module. (**a**) Feature maps of the second layer and (**b**) feature maps of the last layer.

3.3. Detection Head

The detection head plays a critical role in the droplet detection framework in this paper. It is a set of layers added on top of the backbone network that takes the high-level features extracted by the backbone and performs droplet detection by predicting the bounding boxes' offsets and associated class probabilities. The droplets are classified into three size-based categories: large, medium, and small.This classification is derived from their physical dimensions, as follows:

- Small Droplets: Droplets with sizes up to the 33rd percentile of the size range in our dataset. In our study, this threshold is defined as 299.0 units.
- Medium Droplets: Droplets whose sizes fall between the 33rd and 67th percentiles, representing the mid-range sizes.
- Large Droplets: Droplets exceeding the 67th percentile, which, in our dataset, corresponds to a size of 525.0 units and above.

These thresholds were calculated based on the percentile distribution of droplet sizes within our dataset, thus helping to ensure an empirical and data-driven approach to categorization.

The class probabilities represent the probability that the droplet in the anchor box belongs to a particular class, while the offsets represent the distance between the anchor box and the ground truth bounding box. These parameters are predicted using a series of convolutional layers that transform the input feature maps into a set of feature vectors. The class probabilities are predicted using a sigmoid activation function and are normalized such that they sum to 1 across all classes:

$$Pr(c|\text{droplet}) = \sigma(\mathbf{w}_c^T \cdot f(\text{droplet})), \tag{1}$$

where $Pr(c|\text{droplet})$ is the probability that the droplet in the anchor box belongs to class c, σ is the sigmoid activation function, $\mathbf{w}_c$ is the weight vector for class c, and $f(\text{droplet})$ is the feature vector for the anchor box.

Each anchor box is associated with a set of offsets, which are represented by a vector of 4 elements encoding the distance between the anchor box and the ground truth bounding box. The offsets are predicted using a linear activation function applied to the coordinates of the anchor box to obtain the coordinates of the predicted bounding box via the following equation:

$$b_x = \sigma(\mathbf{w}_x^T \cdot f(\text{droplet})) + x_a, \tag{2}$$

$$b_y = \sigma(\mathbf{w}_y^T \cdot f(\text{droplet})) + y_a, \tag{3}$$

$$b_w = p_w e^{\mathbf{w}_w^T \cdot f(\text{droplet})}, \tag{4}$$

$$b_h = p_h e^{\mathbf{w}_h^T \cdot f(\text{droplet})}, \tag{5}$$

where b_x, b_y, b_w, and b_h are the predicted coordinates and dimensions of the bounding box; x_a and y_a are the coordinates of the center of the anchor box; p_w and p_h are the dimensions of the anchor box; $\mathbf{w}_x, \mathbf{w}_y, \mathbf{w}_w$, and $\mathbf{w}_h$ are the weight vectors for the offsets; and $f(\text{droplet})$ is the feature vector for the anchor box.

After the bounding boxes are predicted, the post-processing module applies non-maximum suppression (NMS) to remove the redundant bounding boxes that overlap with each other. The NMS algorithm applies an intersection over union (IoU) threshold to the class probabilities (Equation (1)) and removes all bounding boxes that have a lower probability (IoU) than the threshold. It then selects the bounding box with the highest probability for each droplet class and removes all other bounding boxes that have a high overlap with the selected bounding box. IOU measures the overlap between the predicted bounding box and the ground truth bounding box of a droplet. It is calculated as the ratio of the area of intersection between the two bounding boxes, i and j, to the area of their union as follows:

$$IoU(\text{box}_i, \text{box}_j) = \frac{area(intersection(\text{box}_i, \text{box}_j))}{area(union(\text{box}_i, \text{box}_j))} \tag{6}$$

where box_i and box_j are two bounding boxes, $intersection(\text{box}_i, \text{box}_j)$ is the intersection of the two boxes, $union(\text{box}_i, \text{box}_j)$ is the union of the two boxes, and $IoU(\text{box}_i, \text{box}_j)$ is the intersection over union between the two boxes.

3.4. Optimization-Driven Model Training

The process of training a model is crucial in deep learning to tune the hyperparameters that enable the model to produce predictions as close to the ground truth as possible. During this phase, the model learns to identify patterns and make predictions based on the ground truth data. In the context of our droplet detection system, the model needs to be trained to recognize droplets of different sizes and shapes and to track their movement across frames. The training process involves feeding the model with labeled data, which contain images of droplets with their corresponding bounding boxes and class labels. The model then automatically adjusts its parameters to minimize the difference between the predicted droplet detections and the ground truth droplet labels. This process is iterative and continues until the model's performance on the training data reaches a satisfactory level or until a specified number of iterations (epochs) have been completed.

In this work, the training process for the deep learning model was formulated as an optimization problem, as described in Figure 5, where the objective is to find an optimal set of model parameters that minimize a loss function. The loss function quantifies the difference between the predicted and actual labels for the training data. In the case of our droplet detection system, we designed the loss function as a combination of the following:

- Localization Loss (L_{loc}): This loss is calculated using the intersection over union (IoU) between the predicted bounding box and the ground truth bounding box. The IoU is a measure of the overlap between two bounding boxes and is calculated using Equation (6). The localization loss is then calculated as

$$L_{loc} = 1 - IoU. \tag{7}$$

- Classification Loss (L_{cls}): This loss is calculated using the cross-entropy loss between the predicted droplet class score (P_{pred}) and the ground truth droplet class labels (P_{gt}). The equation for weighted cross-entropy loss is

$$L_{\text{cls}} = -\sum_{i}\sum_{j} \alpha \cdot y_{ij} \cdot \Big(\sigma(S_{\text{pred,i,j}})\Big)^{\gamma} \cdot \log\Big(\sigma(S_{\text{pred,i,j}})\Big). \tag{8}$$

  Here,
  - The outer summation $\sum_i$ iterates over the detected droplets.
  - The inner summation $\sum_j$ iterates over the classes.
  - y_{ij} is 0 or 1, and it corresponds to whether the ground truth class of a detected droplet i is class j or not. Specifically, $y_{ij} = 1$ if the true class for the detected droplet i is class j, otherwise $y_{ij} = 0$.
  - $S_{\text{pred,i,j}}$ represents the predicted score that the detected droplet i belongs to class j. This score is used as an input to the sigmoid function to convert it into a predicted probability for the corresponding class.
  - α and γ are hyperparameters that exist within the range [0,1].
  - σ represents the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$, where $x = S_{\text{pred,i,j}}$.

- Confidence Loss (L_{conf}): This loss is calculated using the binary cross-entropy loss between the predicted objectness score (S_{pred}) and the ground truth objectness score (S_{gt}). The equation for the binary cross-entropy loss is

$$L_{conf} = -S_{gt}\log(S_{pred}) - (1 - S_{gt})\log(1 - S_{pred}). \tag{9}$$

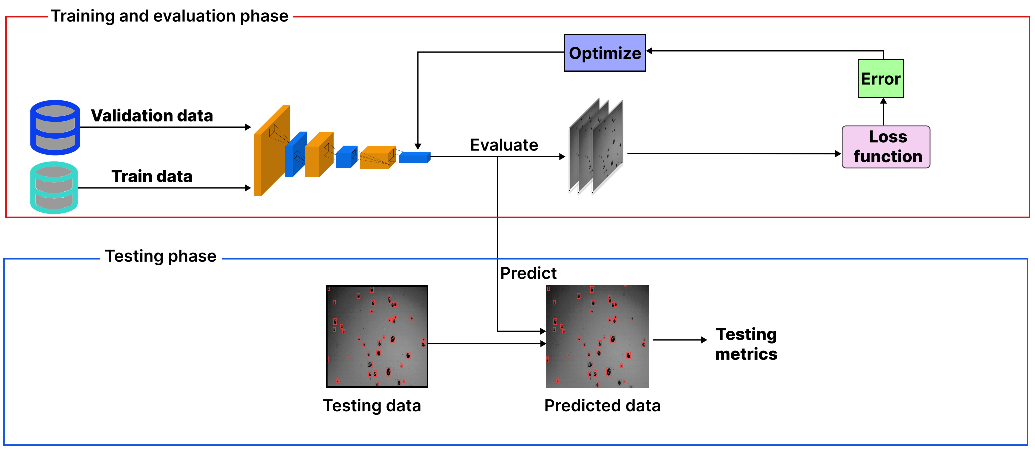

Figure 5. Optimization-driven model training process for droplet detection.

The total loss (L) is then a weighted sum of the following three losses:

$$L = \lambda_{loc}L_{loc} + \lambda_{cls}L_{cls} + \lambda_{conf}L_{conf} \tag{10}$$

where λ_{loc}, λ_{cls}, and λ_{conf} are the weights for the localization loss, classification loss, and confidence loss, respectively.

The model parameters are then tuned automatically via the stochastic gradient descent scheme to minimize the following loss function:

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t; x_{i_t}, y_{i_t}), \tag{11}$$

where η is the learning rate, $\nabla L(\theta_t; x_{i_t}, y_{i_t})$ is the gradient of the loss with respect to the parameters, and i_t is a randomly selected index from $1, \ldots, N$ at iteration t.

The training process also involves a validation dataset, which is a separate set of data that is not used in the initial training phase. We split our data into three sets: training, validation, and testing. The training set was used to adjust the model parameters, while the validation set was used to tune the hyperparameters and monitor the model's performance during the training process. The performance of the model on the validation set was monitored during the training process to prevent overfitting (i.e., the model performs well on the training data but poorly on unseen data). The training process was stopped when the performance on the validation set stopped improving. After the training and validation phases, the final model was evaluated on the testing set. This set was completely unseen during the training and validation phases, and it also provided a measure of how well the model will perform on real-world data.

3.5. Model Pruning

The initial deep neural networks designed in this work have a large number of connections and weights, which lead to tremendous computational and memory demands during training and deployment. To alleviate the demands, we implemented a model pruning technique to reduce the complexity of the neural networks by removing redundant connections and weights, thus making the codes easier to load, store, and deploy. Since the resulting model is substantially more compact and efficient, it can be trained and deployed even with resource-constrained devices such as embedded systems and onboard computers. Additionally, the smaller model also leads to energy and memory efficiency gains, which are crucial for battery-powered devices such as smartphones or drones [29]. In this section, we will describe the pruning procedures detailed in Algorithm 1.

Algorithm 1 Pseudo-code for model pruning with a threshold.

```
    Input: Trained neural network model, pruning threshold $\tau$
 1: Initialization:
 2: Let $W$ be the set of weights in the model
 3: Let $S$ be an empty dictionary to store importance scores
 4: Compute Importance Score:
 5: for each weight $w \in W$ do
 6:     $S[w] \leftarrow$ ComputeImportanceScore($w$)        ▷ Use a criterion like L1 and L2
 7: end for
 8: Prune Weights:
 9: for each weight $w \in W$ do
10:     if $S[w] < \tau$ then
11:         $w \leftarrow 0$                                  ▷ Prune the weight
12:     end if
13: end for
14: Fine-tune Model:
15: Train the model with the remaining weights using a reduced learning rate
16: Repeat:
17: If the desired level of compression is not achieved, return to step 4.
    Output: The pruned and fine-tuned neural network model
```

Compute the importance score: This step involves computing an importance score for each layer, neuron, or weight in the model based on a specific pruning criterion. The pruning criterion can be based on various factors, such as weight magnitude, sensitivity analysis, or information entropy. The importance score represents the contribution of each layer/neuron/weight to the overall performance of the model, and it is used to determine which connections/weights to prune. In this work, we employed the weight magnitude parameter to compute the importance scores of weights in the model pruning process. The process of how weight magnitude can be used to compute importance scores is as follows:

- Compute the Magnitude of the Weights: The algorithm computes the magnitude of each weight in the neural network by taking the absolute value of the weight and disregarding its sign.
- Normalize the Magnitudes: The algorithm normalizes the magnitudes of the weights by dividing each weight's magnitude by the maximum magnitude observed across all weights in the model. e.g., $m_w = \frac{|w|}{M}$. This step ensures that the magnitudes are scaled between 0 and 1, thus allowing for a fair comparison among the weights.
- Importance Score Calculation: The importance score of a weight can be directly derived from its normalized magnitude by assigning the normalized magnitude m_w as the importance score for weight w.

Prune the least important connections/weights using the pruning threshold: After computing the importance scores, the least important connections/weights are pruned based on a pruning threshold. The pruning threshold represents the threshold value used to determine which connections/weights to prune based on their importance scores. Connections/weights with importance scores below the pruning threshold are considered unimportant and are pruned by setting the weight value to zero. The pruning process aims to remove redundant connections/weights and reduce the size of the model while maintaining its accuracy.

Fine tune the pruned model to recover the performance loss caused by pruning: After pruning, the model is fine tuned to improve the accuracy of the pruned model and mitigate the performance loss caused by pruning. The fine tuning process involves training the model on the same dataset as the original model using a smaller learning rate before pruning.

Algorithm 1 describes the entire model pruning process, including the computation of importance scores, pruning, and fine tuning. Meanwhile, Algorithm 2 focuses specifically on the computation of the importance scores of each weight in the model, which is an essential step in the model pruning process. The importance score represents the contribution of each weight to the overall performance of the model, and it is used to determine which weights to prune during the model pruning process.

Algorithm 2 Pseudo-code explaining the calculation of importance scores.

```
   Input: Trained neural network model
1: Initialize an empty dictionary to store the importance scores of each weight
2: for each layer in the model do
3:     $L_1$-norm of weights: $L_1(w) = \sum_{i=1}^{n} |w_i|$
4:     $L_2$-norm of weights: $L_2(w) = \sqrt{\sum_{i=1}^{n} w_i^2}$
5:     Store the norm value in the dictionary for each weight in the layer
6: end for
7: Normalize the importance scores by dividing by the total number of weights in the
   model
   Output: The importance scores of each weight in the model
```

The first step of the algorithm initializes an empty dictionary to store the importance scores of each weight in the model. Next, the algorithm computes the L_1-norm or L_2-norm of the weights in each layer of the neural network. The L_1-norm and L_2-norm are mathematical formulas that measure the magnitude of a vector (in this case, the weights in a layer) in different ways. The L_1-norm represents the sum of the absolute values of the weights, while the L_2-norm represents the square root of the sum of the squares of the weights. The importance score of each weight is then computed using the L_1-norm or L_2-norm value computed in the previous step. The importance scores represent the relative importance of each weight compared to other weights in the same layer, and they are then stored in the dictionary initialized in the first step. Finally, the importance scores are normalized by dividing each score by the total number of weights in the model. Normalizing the scores ensures that the importance scores are between 0 and 1, which facilitates the pruning process. The output of the algorithm is the importance score of each weight in the model.

4. Annotation and Augmentation of Experimental Data

In order to create the optimal deep learning-based droplet detection model, we established a dataset that satisfies the following specific requirements:

- Diversity of Droplet Characteristics: Ensuring variations in the droplet sizes, shapes, and orientations.
- Image Quality and Variation: Images are collected under various conditions.
- Multiple Droplet Instances: Including images with more than one droplet.
- Sufficient Training Data: An adequate number of images for training, validation, and testing.

4.1. Autonomous Labeling

To meet these requirements, we conducted experiments and collected images/videos of droplets sprayed from an array of nozzles (see Figure 1 for the experiment setup), as well as created a database with all the images for training, validation, and testing. However, constructing a large-scale dataset of droplets is a time-consuming and resource-intensive process, mainly because it involves the manual labeling of thousands of droplets. The manual annotation also poses challenges in terms of efficiency, accuracy, and scalability. Thus, there is a critical need for an automatic labeling tool to be integrated with the deep learning model. This tool could then alleviate the burden of manual annotation by automating the process and improving the precision and efficiency of labeling.

In order to develop this automatic labeling tool, first, we manually labeled a small set of droplet data to train the droplet detector designed in the first part of this paper. Then, we used this pre-trained model to localize and classify the droplets, thereby assigning appropriate bounding boxes and class labels automatically. Since the droplet detector was trained with a small dataset, there were errors in the droplet detection. Hence, the last step in the process was to manually correct these errors. Although manual annotation was still involved, it was reduced to a manageable task, and the automatic labeling tool bore the main burden of annotation. The tool significantly reduced the time and effort required for dataset annotation, thus allowing us to create large-scale datasets more efficiently. This efficiency is crucial when dealing with thousands or even millions of droplet instances. Since the droplet detector worked so well at that point, there was no manual annotation involved in the data annotation for the model training at all.

4.2. Data Augmentation

We also implemented data augmentation techniques, such as Cutout, Noise, and Mosaic, to increase the dataset's size and diversity. These techniques were tested for their impact on the mean average precision (mAP), with the Mosaic augmentation technique being the most effective. Figure 6 illustrates these augmentations.

In the pursuit of enhancing the dataset's size and diversity, these data augmentation techniques were employed and analyzed for their impact on the mean average precision (mAP). As can be seen in Table 1, the baseline results, obtained without applying any augmentation, served as a starting point for comparison, with mAP50 values of 0.792 for validation and 0.884 for the test. Applying a 5% noise yielded significant improvements, whereby mAP50 values of 0.904 and 0.903 for the validation and test, respectively, were achieved. The Cutout technique, involving the removal of 10% of the random sections from the images, also demonstrated improvement, with mAP50 values of 0.834 and 0.922 for the validation and test, respectively. Most strikingly, the Mosaic augmentation technique was particularly effective as it achieved superior metrics when compared to the other augmentation techniques with mAP50 values of 0.977 and 0.967 for the validation and test, respectively. However, a combined approach using Noise, Cutout, and Mosaic did not outperform the individual techniques, with mAP50 values of 0.893 for validation and 0.858 for the test. The previous model without augmentation was also evaluated, which achieved a test precision of 0.908. The results clearly indicated that the Mosaic augmentation technique is the most effective method among those tested as it consistently achieved the highest mAP50 and mAP50-95 values. Therefore, the Mosaic augmentation technique was selected as the optimal choice for improving the classification and detection outcomes in this project.

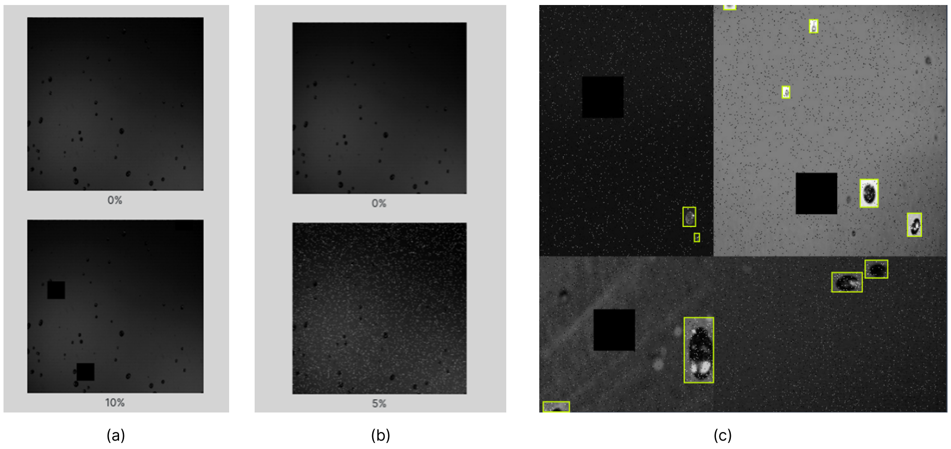

Figure 6. Illustration of the data augmentation techniques: (**a**) Cutout; (**b**) Noise; and (**c**) Mosaic augmentations. Green boxes indicates the labels of droplets.

Table 1. Statistics on the mean average precision with respect to data augmentation techniques.

Train	Valid	Test	Augmentation	Model	Class	Validation Precision	Recall	mAP50	mAP50-95	Test Precision	Recall	mAP50	mAP50-95
468	41	44	None	YOLOv8n	All	0.934	0.744	0.792	0.048	0.957	0.829	0.884	0.526
1404	41	44	Noise (5%)	YOLOv8n	All	0.952	0.862	0.904	0.557	0.949	0.865	0.903	0.53
1404	41	44	Cutout (10%)	YOLOv8n	All	0.954	0.796	0.834	0.507	0.937	0.893	0.922	0.552
1404	41	44	Mosaic	YOLOv8s	All	**0.935**	**0.935**	**0.977**	**0.65**	**0.95**	**0.945**	**0.967**	**0.663**
1404	41	44	Noise, Cutout, Mosaic	YOLOv8n	All	0.921	0.852	0.893	0.532	0.914	0.814	0.858	0.496
1404	41	44	Mosaic	YOLOv8m	All	0.95	0.944	0.981	0.664	0.955	0.955	0.971	0.67
1404	41	44	Mosaic	YOLOv8l	All	0.937	0.937	0.976	0.662	0.948	0.943	0.973	0.624
1404	41	44	Mosaic	YOLOv8x	All	0.928	0.935	0.972	0.674	0.959	0.945	0.977	0.658
1404	41	44	Mosaic	YOLOv8x6	All	0.944	0.963	0.981	0.677	0.954	0.96	0.976	0.653
468	41	44	None	Previous model	All	-	-	-	-	0.908	-	-	-

4.3. Data Organization

Figures 7 and 8 provide insights into the dataset characteristics. This new collection was twelve times larger than that which was previously used, and it encompassed various complexities like varying lighting conditions, multiple droplets, and high droplet density. In Figure 7a, the aspect ratio distribution of the images within our dataset is portrayed, which provides insights into the diversity of the image dimensions before any standardization process.

Figure 7b presents an annotation heatmap that visualizes the density and distribution of droplets across the standardized image frames. The heatmap employed a color gradient to signify the frequency and location of the droplets. Cooler colors (blues and greens) indicate the areas with a lower likelihood of droplet presence, thus representing fewer annotations and a sparser occurrence of droplets. Warmer colors (yellows and reds) delineate the regions with a higher likelihood of droplet detection, and they correspond to a higher density of annotations. This heatmap effectively illustrates the spatial distribution of droplets within the image frames, and it provides a visual summary of the most and least common locations for droplet occurrence in our dataset.

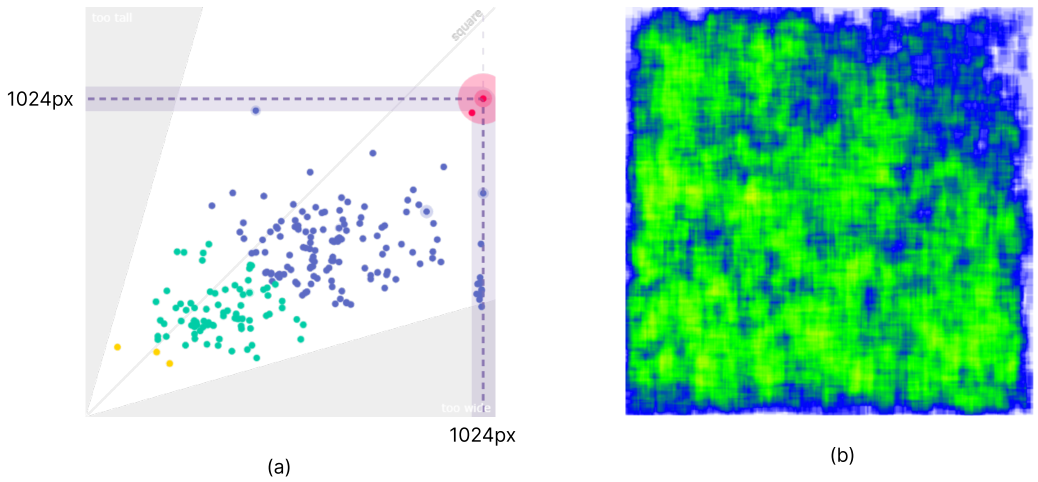

Figure 7. Visual representations of the dataset characteristics. (**a**) Aspect ratio distribution of the image sizes: yellow dots (18%) represent the 'tall' images, sky-blue dots (38%) denote the 'square' images, and deep-sea blue dots (44%) symbolize the 'wide' images—all of which reflect the proportions of images before standardization. (**b**) Annotation heatmap: the color gradient represents the likelihood of finding droplets in each size category across standardized frames, with cooler colors indicating a lower frequency (i.e., sparse droplet occurrence) and warmer colors depicting a higher frequency (i.e., dense droplet occurrence).

In the construction of our dataset, we have considered a range of complexities to ensure a robust evaluation of the droplet detection algorithm. As illustrated in Figure 8, our dataset encompasses the following various challenges:

(a) Different lighting conditions that can significantly affect the appearance of droplets and potentially the performance of detection algorithms. These conditions range from uniform illumination to more complex scenarios with shadows and highlights.

(b) Scenes with multiple droplets, which present the challenge of detecting each droplet accurately while also distinguishing between overlapping droplets.

(c) High droplet density, where the number of droplets in a single frame can be substantial, thus requiring the algorithm to maintain high precision and recall despite the increased difficulty.

These complexities are representative of real-world conditions that any effective droplet detection system must handle. Regarding the construction of a benchmark dataset, the images used in this study are part of a dataset funded and managed by the USDA National Institute of Food and Agriculture. Due to data sharing agreements and privacy constraints, we are unable to make the dataset publicly available. However, we provide representative images from the dataset in Figure 8, which demonstrate the variety and complexity of the imaging conditions under which our droplet detection system was tested and validated.

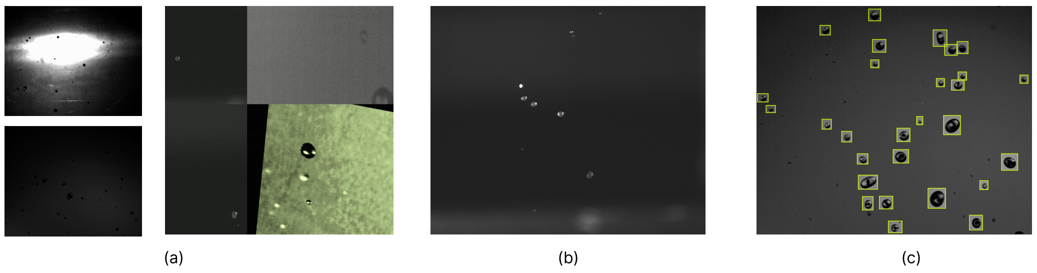
Figure 8. (**a**) Different lighting conditions; (**b**) multiple droplets; and (**c**) the max number of droplets labeled in a given frame.

5. Evaluation Metrics

To assess the overall performance of our framework, we focused on evaluating the detection performance of the droplets.

Droplet Detection Performance Metrics

For the evaluation of droplet detection performance, a set of evaluation images with manually labeled ground truth bounding boxes for each droplet was utilized. The ground truth represents the ideal performance achieved through meticulous human inspection. Each image was fed into the trained model, thus allowing the algorithm to place bounding boxes around the droplets it detected. These droplet detections were subsequently compared to the corresponding ground truth annotations. During this comparison process, we retained the following information for each image as follows:

- Intersection over Union (IoU): IoU measures the overlap between the predicted bounding boxes and the ground truth bounding boxes. It quantifies how well the model's detections align with the actual droplet locations. A higher IoU indicates a better match. The general formula used for calculating IoU is presented in (6).
- The Precision–Recall Curve: For the droplet class, we calculated the precision and recall at different IoU thresholds. Precision represents the ratio of the correctly detected droplets to the total number of detections, while recall measures the proportion of the true droplets that are correctly detected. By varying the IoU threshold, we generated a precision–recall curve. Let us consider the following definitions:
  - True Positives (TP): The number of correctly detected droplets when compared to the ground truth labels.
  - False Positives (FP): The number of incorrectly detected droplets when compared to the ground truth labels.
  - False Negatives (FN): The number of droplets that were missed by the droplet detection algorithm when compared to the ground truth labels.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}, \tag{12}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}. \tag{13}$$

- Average Precision (AP): AP is calculated by computing the area under the precision–recall curve. It captures the overall detection performance across various levels of precision and recall. A higher AP indicates better accuracy in droplet detection. To calculate the AP, we first computed the interpolated precision at each recall level by taking the maximum precision value to the right of that recall level.

$$\text{Interpolated Precision} = \max(\text{Precision}(R')) \quad \text{where} \quad R' \geq R, \tag{14}$$

where R represents the recall level and n is the total number of recall levels. Next, we calculated the area under the interpolated precision–recall curve, which represents the AP value as per the following:

$$\text{Average Precision (AP)} = \frac{1}{n}\sum_{R}\text{Interpolated Precision}(R). \tag{15}$$

- Mean Average Precision (mAP): The mAP is a commonly used evaluation metric in object detection [30–32]. To obtain the mAP, we averaged the AP values across all the droplet classes. This accounted for the variations in the detection performance for different droplet sizes, shapes, or orientations. The mAP provides an overall assessment of the algorithm's ability to detect droplets accurately across multiple classes. Let $c_1, c_2, \ldots, c_n$ be the classes of droplets, then the mAP can be calculated by the following equation:

$$\text{mAP} = \frac{1}{n}\sum_{i=1}^{n}\text{AP}(c_i), \tag{16}$$

where n represents the total number of droplet classes and AP(c_i) represents the average precision for a class c_i, which is computed using (15).

6. Experimental Results

6.1. Droplet Datasets

We have discussed various evaluation metrics for droplet detection extensively thus far. As we delve into the results obtained by our trained model in this section, we will be using some specific variants of the mean average precision (mAP) metric, which may fit better for performance evaluation depending on the scenarios. Specifically, we will use mAP50, which refers to the mAP calculated at an intersection over union (IoU) threshold of 0.50; this means that a prediction is considered correct if its bounding box has at least a 50% overlap with the ground truth bounding box. We also used mAP50-95, which is the average of mAP values calculated at IoU thresholds ranging from 0.50 to 0.95 with a step size of 0.05. This gave us a more comprehensive view of the model's performance across a range of precision levels.

Table 2 shows the experimental results that indicate the comparisons made among the four different datasets and two different models. The models in the comparison include the framework designed in this work and the model developed in [19]. The datasets include the following:

- Previous dataset: The dataset used to produce the results reported in [19] with the respective model. This dataset includes 468 images in the training set, 41 images in the validation set, and 44 images in the testing set.
- Augmented dataset: This is obtained by performing a Mosaic augmentation on the previous dataset to increase the size of the training set to 1404 images.
- New dataset: This is obtained by implementing data augmentation and automatic labeling to increase the size to 6124 images in the training set, 494 images in the validation set, and 275 images in the testing set.

The performance was evaluated using precision, recall, mAP50, and mAP50-95 on both the validation and test sets.

6.2. Results on Performance Metrics

We can see from Table 2 that our model (i.e., the model designed in this work) trained with the Previous dataset, without any augmentation, had the lowest performance as it achieved a validation mAP50 of 0.792 and a test mAP50-95 of 0.526. When trained with the Augmented dataset with Mosaic augmentation, our model achieved a better performance with a validation mAP50 of 0.976 and a test mAP50-95 of 0.624. Additionally, when our model was trained with the New dataset, also with Mosaic augmentation and automatic labeling, it had the best performance, whereby it achieved a validation mAP50 of 0.988 and a test mAP50-95 of 0.737. The precision reached 0.968, i.e., the model could successfully detect 96.8% of the droplets in the testing set that it has never seen before. Moreover, all these performances were superior when compared to the model developed in [19], which was trained with the Previous dataset.

The comparison of the results showed that the data augmentation and the design of the model are crucial for achieving good performance in the droplet detection task. The model designed in this paper when coupled with the Mosaic data augmentation outperformed the other models, and the use of the Augmented dataset improved the performance significantly. The results demonstrated that choosing the right combination of dataset, augmentation, and model can lead to a state-of-the-art performance in droplet detection.

Table 3 presents a detailed evaluation of the droplet detection model's performance across different droplet classes (i.e., small, medium, or large), as well as an aggregate of all the classes. Furthermore, the table also offers insights into various aspects of the model's effectiveness. The metrics are provided for both validation and test datasets. The following results were particularly noteworthy:

- For small droplets, the model's performance showed a consistent behavior between the validation (precision: 0.846 and recall: 0.92) and test sets (precision: 0.891 and recall: 0.898). The mAP50 values were 0.929 for the validation set and 0.934 for the test set, while the mAP50-95 values were 0.64 and 0.621, respectively. The consistency and relatively high precision and recall values suggested effectiveness in identifying small droplets, but the gap between the mAP50 and mAP50-95 values indicated a sensitivity to IoU thresholds.
- In the case of medium droplets, the precision dropped slightly from 0.845 in the validation set to 0.813 in the test set, while the recall values were 0.854 and 0.858, respectively. The mAP50 values were 0.918 for the validation set and 0.912 for the test set, with mAP50-95 values at 0.761 and 0.724, respectively. This indicates that the model maintained a balance between the precision and recall, but there was also a slight drop in the precision in the test set.
- For large droplets, the analysis showed consistency between the validation (precision: 0.927 and recall: 0.881) and the test sets (precision: 0.921 and recall: 0.844). The mAP50 values were 0.95 for the validation set and 0.939 for the test set, with mAP50-95 values of 0.792 and 0.751, respectively. The higher precision for large droplets and the closer values between mAP50 and mAP50-95 suggested a robustness to different IoU thresholds.
- The aggregate metrics for all the classes showed a precision of 0.873 and a recall of 0.885 for the validation set, as well as a precision of 0.875 and a recall of 0.867 for the test set. The mAP50 values were 0.932 for the validation set and 0.929 for the test set, with mAP50-95 values of 0.731 and 0.699, respectively. These numbers provided a well-rounded view of the model's performance across all droplet sizes, thus indicating overall stability and generalization ability.

Table 2. Statistics of the mean average precision.

Dataset	Train	Valid	Test	Augmentation	Model	Validation				Test			
						Precision	Recall	mAP50	mAP50-95	Precision	Recall	mAP50	mAP50-95
Previous [19]	468	41	44	None	This work	0.934	0.744	0.792	0.548	0.957	0.829	**0.884**	**0.526**
Augmented	1404	41	44	Mosaic	This work	0.937	0.937	0.976	0.662	0.948	0.943	0.973	0.624
New	6124	494	275	Mosaic	This work	0.954	0.966	0.988	0.765	**0.97**	**0.968**	**0.988**	**0.737**
Previous[19]	468	41	44	None	Previous model [19]	-	-	-	-	**0.908**	-	-	-

Table 3. Statistics of the mean average precision for the three classes.

Classes	Validation						Test					
	Images	**Instance**	**Precision**	**Recall**	**mAP50**	**mAP50-95**	**Images**	**Instance**	**Precision**	**Recall**	**mAP50**	**mAP50-95**
Small	494	727	0.846	0.92	0.929	0.64	275	438	0.891	0.898	0.934	0.621
Medium	494	814	0.845	0.854	0.918	0.761	275	492	0.813	0.858	0.912	0.724
Large	494	665	0.927	0.881	0.95	0.792	275	482	0.921	0.844	0.939	0.751
All	494	2206	0.873	0.885	0.932	0.731	275	1412	0.875	0.867	0.929	0.699

Figure 9 provides a visualization of the training performance of our model when trained with the New dataset, which also achieved the best performance, as shown in Table 1. Figure 9a illustrates the F1-confidence curve of our detection model across the different droplet sizes, namely small, medium, and large. The F1 score is a crucial metric in the field of machine learning and computer vision, particularly in tasks such as object detection, where the balance between precision and recall is vital. It provides a single metric that encapsulates the performance of the model in terms of both precision and recall. We can see that the high F1 scores ranging from 0.6 to 0.9 with a confidence threshold from 0 to 0.8 for all classes—including small, medium, and large droplets—are indicative of the model's strong performance. The model's performance was consistent across different droplet sizes, as evidenced by the close proximity of the F1 scores for the small, medium, and large droplets. This consistency was maintained over a wide range of confidence thresholds, thus indicating the model's robustness in handling droplets of various sizes. The F1 scores exhibited a fast increase with a confidence threshold of 0 to 0.1, after which they stabilized up to a confidence threshold of 0.8. This suggests that the model's performance remains steady over a wide range of confidence levels, thus reinforcing its reliability. However, a drastic decrease in the F1 scores was observed at a confidence threshold above 0.8. This is a common phenomenon in object detection models, as a confidence threshold close to 1 implies absolute certainty in the predictions, which is rarely achievable in practice due to inherent uncertainties and potential errors in the data or model.

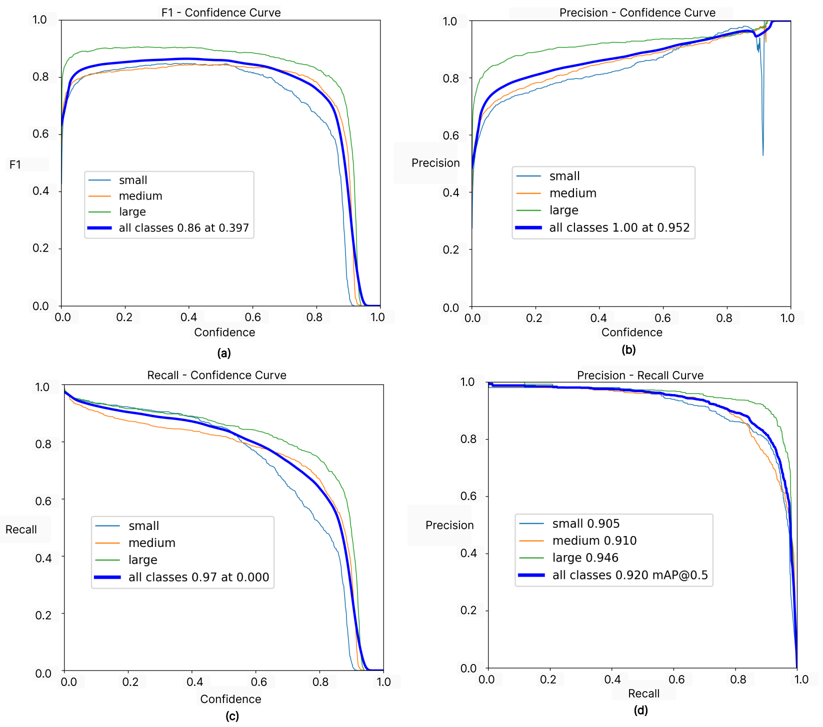

Figure 9. Performance metrics of the droplet detection model. (**a**) F1—confidence curve; (**b**) precision—confidence curve; (**c**) recall—confidence curve; and (**d**) precision—recall curve.

The precision, which indicates the proportion of true droplets (i.e., correct detections) among all detected droplets, was used as a measure of the model's performance. Regarding the precision–confidence curve, as shown in Figure 9b, for small droplets, the precision started at 0.5 and then rapidly climbed to 0.7 at a confidence level of 0.08. This suggests that the model's accuracy in predicting small droplets quickly improves as the confidence level increases. When the confidence level is between 0.1 and 0.85, the precision is rising gradually for all classes of droplets. However, a sudden drop in precision from above 0.9 down to 0.5 was observed at a confidence level of about 0.9, which was followed by a rebound to 1 at a confidence level close to 1, for the small droplets. This volatility suggests that the model's performance on small droplets may be less stable at high confidence levels, which could potentially be due to the inherent challenges associated with smaller droplets. In contrast, the precision for medium and large droplets exhibited a steady increase for the majority of the precision–confidence curve. The absence of any sudden drops in precision indicates that the model's performance on medium and large droplets is stable across all confidence levels and its performance improves consistently as the confidence level increases.

The recall–confidence curve of our model was analyzed to understand the model's performance across various droplet classes, including small, medium, and large. Recall, which is defined as the ratio of true positive predictions to the sum of true positives and false negatives, serves as an essential metric through which to gauge the model's ability to correctly identify the droplets it was trained to detect. Regarding the recall–confidence curve, as shown in Figure 9c, for small droplets, the recall commences at 1 at a confidence level of 0, which gradually diminished to 0.5 at a confidence level of 0.8. This gradual decline illustrates the model's decreasing efficacy in identifying small droplets as the confidence level rises. A precipitous decline to a 0 recall at a confidence level of 1 underscores a potential vulnerability in the model's performance with small droplets at elevated confidence levels. Simultaneously, the trend for the medium droplets mirrored that of the small droplets, which initiated at 1 at a 0 confidence and further attenuated to 0.7 at a 0.8 confidence. The less pronounced decline compared to decline in the small droplets indicates a more robust performance with medium-sized droplets, though the trend still suggests a diminishing ability to identify these droplets as the confidence increases. In addition, the recall trajectory for large droplets begins at 1 at a 0 confidence, which then tapers to 0.8 at a 0.8 confidence. This pattern, with the highest recall at a 0.8 confidence among the three sizes, signifies the model's relative strength in detecting large droplets.

The precision–recall curve is a valuable tool for understanding the trade-off between precision (the ratio of true positive predictions to the sum of true positives and false positives) and recall (the ratio of true positive predictions to the sum of true positives and false negatives). The precision–recall curve, as shown in Figure 9d, demonstrated that both the small and medium droplets exhibited a similar trend in terms of precision and recall. As the recall increased from 0.0 to 0.8, the precision gradually decreased from 1.0 to 0.9. This gradual decline illustrated that, as the model identified more droplets (increasing recall), it became slightly less accurate in its predictions (decreasing precision). The decrease in precision for the small and medium droplets implied a balanced trade-off between these two metrics for these droplet sizes. Meanwhile, the large droplets followed a similar pattern but with a milder decline in precision. Starting at 1 precision with 0 recall, the precision decreases to 0.95 at a recall of 0.8. This higher precision at the same recall level compared to small and medium droplets suggests that the model is more accurate in detecting large droplets.

6.3. Results of the Confusion Matrix and Training Loss

Figure 10 presents the confusion matrix for the three classes of droplets of our droplet detection model. The model correctly identifies 92% of the small droplets but misclassified 8% as medium. For the medium droplets, the model showed good performance with an 85% true positive rate, but it also had some confusion between the small and large categories.

Large droplets were identified with an 87% true positive rate, with some confusion with the medium category and an 18% false positive rate with the background. The model's performance with large droplets stood out as particularly strong, while there were areas of confusion between the small and medium categories.

In addition to the mAP metrics, we also evaluated the training loss of our droplet detector, as indicated by the lost functions defined in Equations (7)–(10). The training loss provided insights into how well the model is learning from the training data and is an essential aspect of understanding the model's convergence and overall performance. Figure 10 illustrates the results of training the droplet detection model over 200 epochs. It includes three types of loss: localization loss, classification loss, and confidence loss. The localization loss started at 1.68 and gradually decreased to 0.52 by the end of the training. This indicated that the model's ability to accurately locate droplets within an image improved over time. The consistent decrease in the localization loss suggested that the model was learning the spatial characteristics of the target droplets effectively. The classification loss began at 1.93 and ended at 0.29. This metric measures the model's ability to correctly classify the droplets within the image. The substantial reduction in classification loss over the epochs indicated that the model was becoming increasingly proficient at identifying the correct classes for the droplets. Simultaneously, starting at 1.06, the confidence loss dropped down to 0.8 after 200 epochs. Confidence loss typically measures the model's certainty in its predictions. The decrease in this loss shows that the model's confidence in its predictions improved but at a slower rate when compared to the other two losses. This is because the features that describe large, medium, and small droplets are very similar to each other, especially if the size differences are subtle.

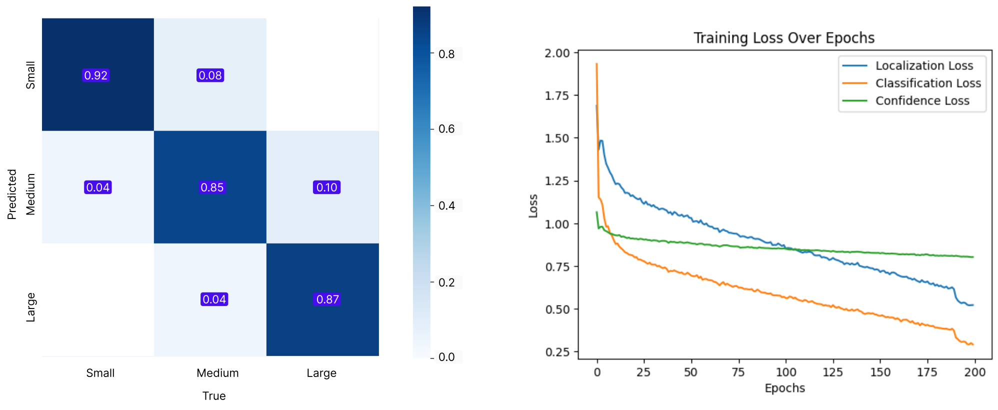

Figure 10. The confusion matrix and training loss of our droplet detection model.

Figure 11a consists of the labeled ground truth images, which contain multiple bounding boxes that denote the droplets. Figure 11b shows the detection results on the right, and it highlights the droplets detected by the model with high confidence scores (>0.9). The confidence score represents the level of certainty or confidence assigned by the model to each detected droplet. It indicates the model's confidence in the accuracy of the detection, with higher scores indicating a stronger belief that the droplet is present. Notably, the detection results, as shown in Figure 11b, encompassed all of the droplets present in the ground truth data, which is shown in Figure 11a. Interestingly, the model can even detect additional droplets that do not have corresponding bounding boxes that the ground truth data missed. It appears that the droplet detection model achieved an exceptional performance, in which it potentially indicated convergence and exhibited a high level of effectiveness.

It is also evident from Table 2 that our model, when trained with the Previous dataset without any augmentation, demonstrated the least robustness to occlusion. In contrast, the model trained with the Augmented dataset, when using Mosaic augmentation, exhibited an improved performance, which was evidenced in it maintaining high mAP50 values even with partial droplet occlusions. The model trained with the New dataset, which included

Mosaic augmentation and automatic labeling, showcased the highest resilience to occlusion with only a marginal decrease in the mAP50-95 when faced with up to 50% occlusion. The consistent precision and recall across varying levels of occlusion underscore the model's ability to detect and correctly classify partially visible droplets, which is a critical aspect for real-world applications where occlusions are commonplace. In addition, in Figure 11b, we present a series of image patches that showcase the droplet detection model's capacity to recognize droplets under varying conditions, including instances of partial occlusion. This robust detection even amidst occlusions can be attributed to the model's learned feature representations, which are capable of capturing the essential characteristics of droplets even when they are not fully visible. The ability to discern partially obscured droplets is crucial for applications in dynamic environments, where complete visibility of every droplet cannot always be guaranteed. The results depicted in Figure 11b affirm the model's proficiency in maintaining high detection accuracy—a testament to its robustness against occlusion.

The excellent performance of the model can be attributed to its advanced architecture, effective training process, and robust feature representation. The model's deep neural network structure allows for the efficient and precise detection of droplets, whereby it leverages multi-scale features to capture intricate details. Through extensive training, the model has learned to effectively discriminate between droplets and background regions, thereby resulting in accurate and comprehensive detection. The convergence of the model's training process, coupled with its ability to generalize in unseen instances and handle various droplet sizes, enables it to surpass expectations and achieve a desirable performance in droplet detection tasks.

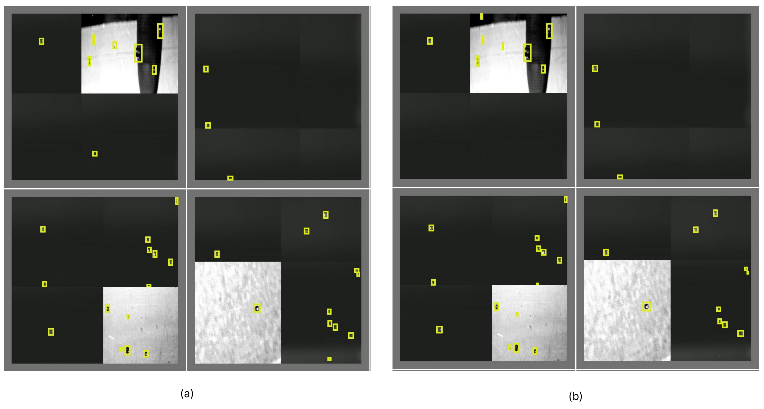

Figure 11. Droplet detection in image patches. (**a**): Several instances of an image patch, which are annotated with bounding boxes to indicate the droplets based on ground truth labels. (**b**): Corresponding image patches with bounding boxes denoting the droplets identified by the droplet detector with a confidence level higher than 80%.

6.4. Results of Model Pruning and Implementation in an Embedded System

Tables 4 and 5 demonstrate the inference time of the droplet detection and tracking framework before and after pruning, which were measured in milliseconds per frame on different devices with distinct GPU and CPU configurations. The pruning process described in Algorithm 1 resulted in a more compact model with approximately a 30% reduction in the model size. This was achieved by removing the redundant or less significant parameters, connections, and layers. The compactness makes the model compatible with mobile onboard computers.

The observed improvements in the inference time before and after pruning are as follows:

- For the Jetson AGX Orin device, the CPU inference time improved by approximately 4.9% after pruning, while the GPU inference time remained almost the same with a reduction of approximately 0.4%.

- In the case of the AI supercomputer, both the CPU and GPU inference times improved after pruning. The CPU time decreased by approximately 1.3%, while the GPU time decreased by approximately 8.4%.
- On the work station, there were slight improvements in both the CPU and GPU inference times after pruning. The CPU time reduced by approximately 0.8%, and the GPU time reduced by approximately 7.8%.

The improvements in inference time after pruning can be attributed to several factors. Pruning removes redundant parameters and reduces memory access and computational complexity. This leads to optimized computation, increased parallelism, and streamlined execution, thus resulting in faster inference times. These findings suggest that the performance of the droplet detection framework can be optimized through pruning, thereby resulting in more efficient processing of frames, especially with GPUs. However, it is worth noting that the specific GPU and CPU configurations of each device also play a role in determining the overall inference time. The percentage reductions in inference time after pruning indicate an improved efficiency and faster processing of droplet detection. These improvements contribute to better performance and near real-time capabilities. This makes the droplet detection framework more practical and efficient for deployment in various mobile applications.

Table 4. Device specifications for inferencing deep learning models (ms).

	GPU	CPU
Jetson AGX Orin	NVIDIA Ampere (Santa Clara, CA, USA)	ARM(R) Cortex(R) (Cambridge, UK)
Work Station	NVIDIA GeForce RTX 3070	Intel(R) Core™ i7-i010700F (Santa Clara, CA, USA)
AI Supercomputer	A100	AMD EPYC 7402P Rome (Santa Clara, CA, USA)

Table 5. Statistics of the inference time (ms).

	Before Pruning		After Pruning	
	CPU	GPU	CPU	GPU
Jetson AGX Orin	7693.3	45.8	7317.1	45.6
AI Supercomputer	7655.4	22.3	7555.6	20.5
Work Station	7662.4	32.7	7604.6	30.14

Upon further analysis, we recognized that the nuances of GPU inference on the Jetson AGX Orin platform required additional explanation, especially in the context of model pruning. Unlike more powerful AI supercomputers and workstations, the Jetson AGX Orin is designed with a focus on power efficiency for embedded systems. As such, the pruning process, while reducing the model's complexity, does not significantly impact the already optimized GPU inference due to the embedded system's limited parallel processing resources when compared to full-scale GPUs.

The Jetson AGX Orin's embedded GPU is tailored for high efficiency in a constrained power envelope, and this architectural focus on efficiency means that there is less redundant computation to eliminate through model pruning. Therefore, the performance gains on this platform from pruning are more noticeable on the CPU than the GPU. This contrasts with more powerful, dedicated GPU platforms, where the larger number of parallel processing cores can benefit from the reduced computational load post-pruning. Furthermore, it should be noted that the observed marginal reduction in GPU inference time on the Jetson AGX Orin could be within the range of measurement variability, thus suggesting that the performance on such embedded systems can be primarily constrained by hardware capabilities rather than model complexity.

7. Conclusions

This study represents a pioneering step in the field of droplet detection, and it exhibits an innovative methodology grounded in deep learning and an architecture laid by YOLOv8. By leveraging cutting-edge techniques—such as CSPDarknet53 for feature extraction, non-maximum suppression for clutter-free detection, and model pruning to make it compact—our approach significantly enhances the precision and processing rate in droplet detection. The use of anchor boxes and unification through a feature pyramid network further contributes to the model's ability to detect droplets across various sizes with solitary, optimally situated bounding boxes.

The experimental results, which showcase a 97% precision and 96.8% recall in droplet detection, affirm the advantages of our methodology. Moreover, its capability of real-time operation on embedded systems like Jetson Orin makes it a viable and revolutionary tool for practical applications in agriculture. Our approach thus provides a foundation for sustainable farming practices by facilitating efficient resource utilization and waste reduction.

This research not only sets a new benchmark in droplet detection, but it also opens up promising avenues for further exploration and refinement. Future work can delve into extending the applicability of this model to other domains and enabling more efficient pixel processing by integrating policy networks and reinforcement learning into the droplet detection framework. We are also working toward making this technology deployable in mobile crop spaying systems in the field.

Author Contributions: N.H. performed data labeling, coding, model training, experimentation, and writing the article. K.-D.N. supervised the work, helped in data collection, model selection, data management, article writing, and made revisions with significant constructive feedback. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by grants #2021-67022-38910 and #2022-67021-38911 from the USDA National Institute of Food and Agriculture. APC was funded by USDA National Institute of Food and Agriculture.

Data Availability Statement: Due to confidentiality agreements, supporting data cannot be made openly available. Further inquiries can be directed to the corresponding author.

Acknowledgments: This work is supported by USDA National Institute of Food and Agriculture. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the view of the U.S. Department of Agriculture. The authors would also like to extend special thanks to Praneel Acharya for his invaluable assistance and insights. His work on AI-enabled droplet detection and tracking for agricultural spraying systems [19] has been a crucial reference in our study .

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

References

1. Butts, T.R.; Butts, L.E.; Luck, J.D.; Fritz, B.K.; Hoffmann, W.C.; Kruger, G.R. Droplet size and nozzle tip pressure from a pulse-width modulation sprayer. *Biosyst. Eng.* **2019**, *178*, 52–69. [CrossRef]
2. Butts, T.R.; Samples, C.A.; Franca, L.X.; Dodds, D.M.; Reynolds, D.B.; Adams, J.W.; Zollinger, R.K.; Howatt, K.A.; Fritz, B.K.; Clint Hoffmann, W.; et al. Spray droplet size and carrier volume effect on dicamba and glufosinate efficacy. *Pest Manag. Sci.* **2018**, *74*, 2020–2029. [CrossRef] [PubMed]
3. Tate, R. Immersion sampling of spray droplets. *AIChE J.* **1961**, *7*, 574–577. [CrossRef]
4. Zalay, A.; Bouse, L.; Carlton, J.; Crookshank, H.; Eberle, W.; Howie, R.; Shrider, K. Measurement of airborne spray with a laser Doppler velocimeter. *Trans. ASAE* **1980**, *23*, 548–0552. [CrossRef]
5. Hoffmann, W.C.; Fritz, B.K.; Lan, Y. Using laser diffraction to measure agricultural sprays: Common sources of error when making measurements. *Int. J. Precis. Agric. Aviat.* **2018**, *1*, 15–18. [CrossRef]
6. Basu, A.S. Droplet morphometry and velocimetry (DMV): A video processing software for time-resolved, label-free tracking of droplet parameters. *Lab Chip* **2013**, *13*, 1892–1901. [CrossRef]
7. Massinon, M.; De Cock, N.; Forster, W.A.; Nairn, J.J.; McCue, S.W.; Zabkiewicz, J.A.; Lebeau, F. Spray droplet impaction outcomes for different plant species and spray formulations. *Crop. Prot.* **2017**, *99*, 65–75. [CrossRef]
8. De Cock, N.; Massinon, M.; Nuyttens, D.; Dekeyser, D.; Lebeau, F. Measurements of reference ISO nozzles by high-speed imaging. *Crop Prot.* **2016**, *89*, 105–115. [CrossRef]

9. Wang, L.; Song, W.; Lan, Y.; Wang, H.; Yue, X.; Yin, X.; Luo, E.; Zhang, B.; Lu, Y.; Tang, Y. A smart droplet detection approach with vision sensing technique for agricultural aviation application. *IEEE Sens. J.* **2021**, *21*, 17508–17516. [CrossRef]
10. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Proceedings of the Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, Montreal, BC, Canada, 7–12 December 2015; Volume 28.
11. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
12. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In Proceedings of the Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016; Proceedings, Part I 14; Springer: Berlin/Heidelberg, Germany, 2016; pp. 21–37.
13. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2980–2988.
14. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2961–2969.
15. Sun, R. Optimization for deep learning: Theory and algorithms. *arXiv* **2019**, arXiv:1912.08957.
16. Lee, H.; Lee, N.; Lee, S. A Method of Deep Learning Model Optimization for Image Classification on Edge Device. *Sensors* **2022**, *22*, 7344. [CrossRef]
17. Li, L.; Zhang, S.; Wu, J. Efficient object detection framework and hardware architecture for remote sensing images. *Remote Sens.* **2019**, *11*, 2376. [CrossRef]
18. Mirani, I.K.; Tianhua, C.; Khan, M.A.A.; Aamir, S.M.; Menhaj, W. Object Recognition in Different Lighting Conditions at Various Angles by Deep Learning Method. *arXiv* **2022**, arXiv:2210.09618.
19. Acharya, P.; Burgers, T.; Nguyen, K.D. AI-enabled droplet detection and tracking for agricultural spraying systems. *Comput. Electron. Agric.* **2022**, *202*, 107325. [CrossRef]
20. Gardner, K.; Uddin, M.M.; Tran, L.; Pham, T.; Vanapalli, S.; Li, W. Deep learning detector for high precision monitoring of cell encapsulation statistics in microfluidic droplets. *Lab Chip* **2022**, *22*, 4067–4080. [CrossRef]
21. Hasti, V.R.; Shin, D. Denoising and fuel spray droplet detection from light-scattered images using deep learning. *Energy AI* **2022**, *7*, 100130. [CrossRef]
22. Wang, K.N.; Yang, X.; Miao, J.; Li, L.; Yao, J.; Zhou, P.; Xue, W.; Zhou, G.Q.; Zhuang, X.; Ni, D. AWSnet: An auto-weighted supervision attention network for myocardial scar and edema segmentation in multi-sequence cardiac magnetic resonance images. *Med. Image Anal.* **2022**, *77*, 102362. [CrossRef] [PubMed]
23. Yu, M.; Han, M.; Li, X.; Wei, X.; Jiang, H.; Chen, H.; Yu, R. Adaptive soft erasure with edge self-attention for weakly supervised semantic segmentation: Thyroid ultrasound image case study. *Comput. Biol. Med.* **2022**, *144*, 105347. [CrossRef]
24. Zhou, G.Q.; Zhang, W.B.; Shi, Z.Q.; Qi, Z.R.; Wang, K.N.; Song, H.; Yao, J.; Chen, Y. DSANet: Dual-branch shape-aware network for echocardiography segmentation in apical views. *IEEE J. Biomed. Health Inform.* **2023**, *27*, 4804–4815. [CrossRef] [PubMed]
25. Guan, Q.; Chen, Y.; Wei, Z.; Heidari, A.A.; Hu, H.; Yang, X.H.; Zheng, J.; Zhou, Q.; Chen, H.; Chen, F. Medical image augmentation for lesion detection using a texture-constrained multichannel progressive GAN. *Comput. Biol. Med.* **2022**, *145*, 105444. [CrossRef] [PubMed]
26. Chen, Y.; Yang, X.H.; Wei, Z.; Heidari, A.A.; Zheng, N.; Li, Z.; Chen, H.; Hu, H.; Zhou, Q.; Guan, Q. Generative adversarial networks in medical image augmentation: A review. *Comput. Biol. Med.* **2022**, *144*, 105382. [CrossRef] [PubMed]
27. Jocher, G.; Chaurasia, A.; Qiu, J. YOLOv8 by Ultralytics. 2023. Available online: https://docs.ultralytics.com/ (accessed on 20 January 2023).
28. Wang, C.Y.; Liao, H.Y.M.; Wu, Y.H.; Chen, P.Y.; Hsieh, J.W.; Yeh, I.H. CSPNet: A new backbone that can enhance learning capability of CNN. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, 13–19 June 2020; pp. 390–391.
29. Buchelt, A.; Adrowitzer, A.; Kieseberg, P.; Gollob, C.; Nothdurft, A.; Eresheim, S.; Tschiatschek, S.; Stampfer, K.; Holzinger, A. Exploring artificial intelligence for applications of drones in forest ecology and management. *For. Ecol. Manag.* **2024**, *551*, 121530. [CrossRef]
30. Nguyen, K.; Huynh, N.T.; Nguyen, P.C.; Nguyen, K.D.; Vo, N.D.; Nguyen, T.V. Detecting objects from space: An evaluation of deep-learning modern approaches. *Electronics* **2020**, *9*, 583. [CrossRef]
31. Zhu, H.; Wei, H.; Li, B.; Yuan, X.; Kehtarnavaz, N. A review of video object detection: Datasets, metrics and methods. *Appl. Sci.* **2020**, *10*, 7834. [CrossRef]
32. Zhu, X.; Lyu, S.; Wang, X.; Zhao, Q. TPH-YOLOv5: Improved YOLOv5 based on transformer prediction head for object detection on drone-captured scenarios. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 2778–2788.