FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Exploratory Analysis of Segmentation Techniques for Multiphase Flow Experimental Images

Jorge Levi Perdigoto da Costa

Mestrado em Engenharia Informática e Computação

Supervisors: Mónica Silva, Zafeiris Kokkinogenis

February 27, 2026

Exploratory Analysis of Segmentation Techniques for Multiphase Flow Experimental Images

Jorge Levi Perdigoto da Costa

Mestrado em Engenharia Informática e Computação

February 27, 2026

Resumo

A obtenção de medições fiáveis de bolhas em experiências de escoamento multifásico depende de uma segmentação de imagem precisa; no entanto, as imagens experimentais são exigentes (sobreposição, reflexos, variações de iluminação) e os dados anotados são, em geral, escassos. Esta dissertação avalia um pipeline end-to-end de segmentação por instâncias para imagens de escoamento multifásico baseado em Mask R-CNN, explorando supervisão sintética e aprendizagem por transferência de sintético para real. Três backbones do TorchVision (ResNet-50-FPN, ResNet-50-FPN v2 e ResNet-101-FPN) são comparados em quatro regimes de treino (apenas sintético, apenas real, transferência e transferência com aumento de dados) em experiências controladas sobre uma partição de dados reais no formato COCO. O desempenho é avaliado com indicadores complementares: sobreposição entre máscaras-union para quantificar a delineação do primeiro plano e medidas derivadas de instâncias (erro de contagem e estatísticas de área) para refletir a qualidade das medições, em conjunto com o desempenho em tempo de inferência.

Ao longo dos regimes, a transferência de sintético para real apresenta os ganhos mais consistentes, com a melhor configuração a atingir Dice $= 0.963$, IoU $= 0.930$ e erro médio absoluto de contagem $|\Delta N| = 0.40$ no conjunto de teste real, enquanto o aumento de dados durante a transferência não melhora o desempenho médio neste conjunto de dados. Um estudo de eficiência de anotação mostra ainda que a anotação assistida por modelos reduz substancialmente o esforço manual, diminuindo o tempo médio de anotação de 179.18 s/imagem (manual) para 25.11 s/imagem ao combinar pré-anotações sintéticas com refinamento assistido por SAM2. Em suma, os resultados sustentam a aprendizagem por transferência e a anotação assistida por modelos como facilitadores práticos de uma segmentação por instâncias escalável e orientada a medições em fluxos de trabalho experimentais de escoamento multifásico.

Abstract

Reliable bubble measurements in multiphase-flow experiments depend on accurate image segmentation, yet experimental imagery is challenging (overlap, reflections, illumination changes) and labeled data are typically scarce. This dissertation evaluates an end-to-end instance-segmentation pipeline for multiphase-flow images based on Mask R-CNN, leveraging synthetic supervision and synthetic-to-real transfer learning. Three TorchVision backbones (ResNet-50-FPN, ResNet-50-FPN v2, and ResNet-101-FPN) are compared under four training regimes—synthetic-only, real-only, transfer, and transfer with augmentation—in controlled experiments on a COCO-formatted split of real data. Performance is assessed with complementary indicators: union-mask overlap to quantify foreground delineation and instance-derived measures (count error and area statistics) to reflect measurement fidelity, alongside inference throughput.

Across regimes, synthetic-to-real transfer provides the most consistent gains, with the best configuration reaching Dice $= 0.963$, IoU $= 0.930$, and mean absolute count error $|\Delta N| = 0.40$ on the real test set, while augmentation during transfer does not improve average performance on this dataset. An annotation-efficiency study further shows that model-assisted labeling substantially reduces manual effort, cutting mean annotation time from 179.18 s/image (manual) to 25.11 s/image when combining synthetic pre-annotations with SAM2-assisted refinement. Overall, the results support transfer learning and model-assisted annotation as practical enablers for scalable, measurement-oriented instance segmentation in experimental multiphase-flow workflows.

Acknowledgements

I would like to express my sincere gratitude to everyone who supported me throughout my years at university.

First, I thank my thesis supervisors, Mónica and Zafeiris, for their guidance and trust. Mónica showed an exceptional and unwavering availability, always ready to help and discuss ideas. Zaffeiris provided crucial feedback, suggestions, and constructive criticism that significantly improved the quality and direction of this work.

I am deeply grateful to my parents for their constant support and for giving me the conditions to complete my Master's degree.

To my girlfriend Carolina, thank you for the encouragement, and for keeping me focused and disciplined throughout this journey, your support made a real difference in my academic path.

I also want to thank Vieira and Tiago for their friendship and for the guidance and support they gave me during my first years at university, when everything was still new and challenging. I am equally grateful to Malva, my housemate, for his friendship and for the help in the later years, especially through our work together.

Finally, I thank Fernando for giving me the opportunity to start my professional career and for providing the flexibility and conditions that allowed me to finish this degree alongside work.

Jorge Costa

"Start where you are. Use what you have. Do what you can."

Arthur Ashe

Contents

List of Figures

List of Tables

List of Abbreviations and Symbols

Abbreviations

AI	Artificial Intelligence
AMP	Automatic Mixed Precision
AP	Average Precision
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CPU	Central Processing Unit
CSPDarknet53	Cross Stage Partial Darknet-53
CUDA	Compute Unified Device Architecture
CSV	Comma-Separated Values
ECDF	Empirical Cumulative Distribution Function
CEFT	Centro de Estudos de Fenómenos de Transporte
FEUP	Faculdade de Engenharia da Universidade do Porto
FPN	Feature Pyramid Network
FPS	Frames per Second
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
GT	Ground Truth
IoU	Intersection over Union
JSON	JavaScript Object Notation
LR	Learning Rate
mAP	mean Average Precision
Mask R-CNN	Mask Region-based Convolutional Neural Network
MLP	Multilayer Perceptron
MOTA	Multiple Object Tracking Accuracy
NSD	Nucleation Site Density
R-CNN	Region-based Convolutional Neural Network
RDC	Radial Distance Correction
RAM	Random Access Memory
RLE	Run-Length Encoding
RoI	Region of Interest
SAM	Segment Anything Model
SAM2	Segment Anything Model 2
SDG	Sustainable Development Goal
SGD	Stochastic Gradient Descent
SNN	Siamese Neural Network
StepLR	Step learning-rate scheduler
U-Net	Encoder–decoder CNN architecture for semantic segmentation
UN	United Nations
YOLO	You Only Look Once
YAML	YAML Ain't Markup Language

Symbols

ΔN	Signed bubble-count error (Pred – GT)
$\lvert\Delta N\rvert$	Absolute bubble-count error
γ	StepLR multiplicative decay factor

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework to achieve a better and more sustainable future for all. They include 17 goals addressing major challenges such as poverty, inequality, climate change, environmental degradation, peace, and justice.

This dissertation contributes to sustainable development by improving the efficiency and reproducibility of experimental analysis in multiphase-flow research. It develops and evaluates instance-segmentation workflows for experimental multiphase-flow imagery, including synthetic-to-real transfer learning and model-assisted annotation strategies. By reducing manual labeling effort and enabling faster extraction of bubble-level statistics (e.g., bubble counts and size-related measurements), the proposed pipeline supports more scalable and accessible experimental data processing.

The relevance of this research is most closely aligned with:

SDG 9: Industry, Innovation, and Infrastructure — fostering innovation and improved industrial and experimental processes through reliable computational tools.

SDG	Target	Contribution	Project-level indicators (evidence)
9	9.5	Enhances scientific research and supports technological capability by providing a reproducible, instance-level segmentation pipeline for multiphase-flow experimental images.	Reduction in human effort required to produce reliable labeled datasets for experimental analysis (e.g., lower annotation time per image and faster correction cycles); increased capacity to process and analyze experimental imagery at scale without a proportional increase in manual effort (e.g., higher throughput for bubble-level statistics extraction); improved reproducibility and comparability of experimental processing through standardized, configuration-driven workflows and fixed evaluation protocols.

Chapter 1

Introduction

1.1 Context and Motivation

Multiphase flow, in which two or more phases (gas, liquid, and/or solid) coexist within a moving fluid, is fundamental to many industrial and natural processes. These systems are characterized by strong phase-phase interactions across evolving interfaces, with continuous exchange of momentum, heat, and mass, and with interfacial structures that can deform, merge, fragment, and transition between flow patterns under turbulent conditions [21].

The complexity of multiphase flow lies in its dynamic nature, where the size, distribution, and morphology of the bubble directly affect critical processes such as mass transfer and chemical reaction rates [19]. In particular, the morphology of bubbles, their size, shape, and arrangement, is of the most importance to determine the efficiency of processes such as gas-liquid mass transfer and heat transfer [24]. Other important parameters are the gas fraction distribution, the velocity of the bubbles, and eventually their trajectory through the prevailing flow condition [7]. Precise measurement of bubble size and shape is essential for understanding the factors that influence bubble morphology. These measurements allow for a better understanding of how different bubble shapes form, which in turn makes it possible to control bubble morphology [24].

In industrial reactors such as stirred tanks and bubble columns, bubble size and morphology strongly influence gas-liquid mass transfer and overall process efficiency, motivating accurate characterization of bubble populations [24, 2].

Segmentation in image processing, the process of partitioning an image into meaningful regions, helps to analyze multiphase flows by extracting quantitative information and gaining a deeper understanding of flow dynamics. By segmenting individual bubbles, it is not only possible to obtain bubble morphology, but also to track their spatial and temporal evolution over time [20]. This enables the measurement of important flow characteristics like bubble trajectories, growth rates, and velocity fields, using metrics such as nucleation site density (NSD), bubble departure frequency, bubble lifetime, and waiting times between departures [12].

Traditional methods of bubble segmentation, such as the Hough transform, the breakpoint method, and the watershed method, are highly sensitive to specific sets of bubble images [20]. These methods often fail when confronted with varying lighting conditions, complex bubble shapes, reflections, shadows, and overlapping bubble clusters. Other conventional segmentation methods, such as edge detection, threshold segmentation, and the snake model, are effective under certain conditions but suffer from limited performance and accuracy when dealing with complex images [24]. Additionally, manual identification and analysis of bubble boundaries is labor-intensive, time-consuming, and prone to human error, especially in large datasets.

In contrast, deep learning methods, like convolutional neural networks (CNNs), offer more flexibility and scalability in addressing these challenges [20]. Advanced segmentation techniques, particularly those using deep learning, improve accuracy under complex conditions such as poor lighting, shadows, and noisy images [20], which reduces the likelihood of errors in recognizing bubbles, even in noisy or distorted environments, leading to more reliable data for further analysis. Deep learning methods also provide the ability to accurately segment bubbles even when they overlap [2], something traditional methods struggle with.

Improving segmentation in multiphase flow images is not only a technical necessity but also a step towards optimizing industrial processes. Recent advances in machine learning and computer vision, particularly deep learning, offer promising solutions for overcoming the limitations of manual methods. Convolutional neural networks, and more specifically Mask R-CNN models [2], have enabled pixel-level instance segmentation under challenging overlap. Foundation segmentation models such as the Segment Anything Model (SAM) [24] introduce additional flexibility by supporting prompt-based segmentation and efficient refinement, which can be used in model-assisted annotation workflows.

Figure 1.1 summarizes the link between multiphase-flow physics, the imaging artefacts that make segmentation difficult, and the deep-learning approaches that are commonly used to address these challenges. This conceptual framing motivates the problem definition and the methodological choices adopted in this dissertation.

The combination of strong measurement relevance and difficult imaging conditions leads to a practical tension: bubble segmentation must be accurate enough to support reliable physical statistics, yet scalable enough to be applied across experiments and conditions. This tension motivates the problem addressed in the next section, which focuses on limitations of data availability, generalization across setups, and the cost of producing high-quality instance masks.

1.2 Problem

While deep learning methods, such as convolutional neural networks, have demonstrated considerable promise in improving segmentation accuracy, several key challenges hinder their application to multiphase flow analysis. A major limitation lies in the reliance on large amounts of labeled data, which are difficult and costly to obtain for multiphase flows due to their complex and dynamic nature. These

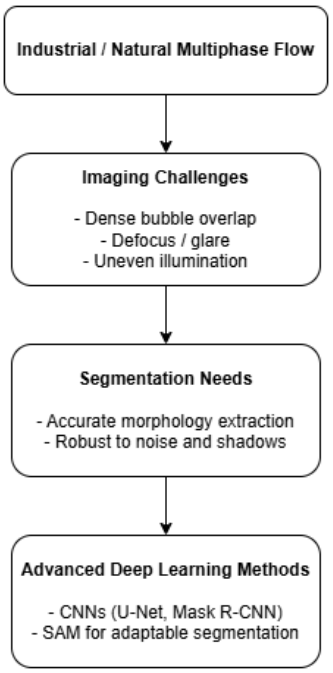


Figure 1.1: Conceptual link between multiphase-flow physics, imaging challenges, segmentation requirements, and modern deep-learning solutions.

models typically demand retraining or careful adaptation to new experimental setups, which increases computational cost and reduces scalability across conditions [24].

Deep learning for bubble segmentation is an active research area, but results can still be strongly dependent on the imaging setup, flow regime, and annotation protocol. This means that even when strong performance is achieved in a given scenario, it may not transfer reliably to new experimental conditions without additional data and adaptation [24, 6].

Recently, interactive foundation models for segmentation have been adapted to bubble imagery, with *bubSAM* proposed as a representative example [24]. Existing models, such as the *bubSAM* framework [24], encounter limitations under certain conditions. For example:

- **Defocused bubbles:** At higher aeration rates, defocused bubbles frequently appear in the images. These artifacts are difficult to remove and are often mistakenly segmented as real bubbles, reducing precision.
- **Small-bubble detection:** The model struggles to capture sufficient detail to accurately segment small bubbles, resulting in reduced recall.
- **Efficiency constraints:** The detection process can be computationally intensive, and practical deployment depends on balancing segmentation quality with processing throughput under a fixed hardware setup.

Therefore, current deep-learning approaches either achieve high accuracy under narrow experimental conditions or generalize poorly when image characteristics vary. This highlights the need for segmentation models and workflows that can adapt across domains, maintain robustness under noise and overlap, and reduce dependence on extensive manual labeling [24, 2, 6].

Another challenge concerns the subjectivity introduced during data labeling and model tuning. Manual annotation and hyperparameter selection can bias the model toward specific conditions, reducing generalization. Integrating pre-trained models and model-assisted labeling tools can mitigate this issue by reducing repetitive manual effort and improving reproducibility of dataset creation [24, 7].

In summary, the core problem lies not in the absence of capable segmentation algorithms but in their limited adaptability across experimental conditions and their dependence on large annotated datasets. In multiphase flow imaging, variations in bubble morphology, overlap, illumination, and noise can degrade performance when models are applied outside the conditions they were trained on, making generalization and data efficiency central challenges [24, 2, 6].

To address these challenges, this dissertation evaluates deep learning instance segmentation models for experimental multiphase flow images and investigates strategies to reduce reliance on manual annotation. In particular, it studies training on synthetic data followed by fine- tuning on real images, and it quantifies the benefits of model-assisted annotation workflows that combine synthetic pre-annotations with interactive refinement using foundation segmentation models such as SAM and SAM2 integrated in Label Studio[1] [9, 15].

1.3 Research questions

Given the problems identified above, the following research questions were established:

1. How do the selected deep learning instance segmentation models compare when trained and evaluated on the real multiphase flow experimental dataset?

2. What is the impact of training on synthetic bubble data and subsequent fine-tuning on real images on segmentation performance, particularly when real annotations are limited?

3. How much can model-assisted annotation, using synthetic pre-annotations and SAM2-assisted interactive refinement in Label Studio, reduce annotation effort while preserving high-quality bubble masks?

[1] https://labelstud.io/guide

Chapter 2

Literature Review

This chapter reviews the state of the art in bubble identification and segmentation for multiphase-flow imaging, with emphasis on techniques that support reliable quantitative analysis of experimental data [2, 20].

Segmentation performance in multiphase-flow imagery is primarily constrained by scarce high-quality instance annotations and by variability across experimental conditions, which limits generalization across setups [2, 8]. The strategies surveyed in this chapter (synthetic data, transfer learning, and model-assisted annotation) aim to reduce manual labeling effort while improving robustness under domain shift [8, 23, 9, 15].

The remainder of the chapter is organized as follows. Section 2.1 introduces segmentation in multiphase-flow imaging and clarifies the role of semantic and instance formulations. Section 2.2 surveys representative segmentation approaches, from classical baselines to modern deep learning and foundation-model methods. Section 2.3 discusses training strategies under limited annotation budgets, including synthetic-to-real fine-tuning and interactive annotation pipelines. Section 2.4 summarizes available datasets and evaluation practices. Finally, Section 2.5 synthesizes research gaps that motivate the methodological choices explored in this dissertation.

2.1 Segmentation in multiphase flow imaging

In multiphase-flow experiments, segmentation is not an end in itself: it is the interface between raw imagery and quantitative bubble statistics. By converting image content into explicit bubble regions (and, when possible, per-bubble instances), segmentation enables repeatable measurement of bubble populations across operating conditions and recording setups [2, 20]. Even visually minor boundary errors can bias these measurements, particularly in dense regimes [2, 6].

In this dissertation, *multiphase flow* is used as an umbrella term for flows involving more than one phase/component, with the experimental focus here being gas-liquid imagery. Within gas-liquid systems, *bubbly flow* refers more narrowly to the dispersed-bubble regime, and it is only one among several

commonly discussed flow patterns (e.g., bubbly, slug, churn, annular) [21]. Because experimental operating conditions and imaging setups can span different regimes and because the segmentation methods surveyed are not inherently restricted to a single regime, the dissertation adopts *multiphase-flow* terminology for the general context[2, 6, 8].

2.1.1 Measurement goals supported by segmentation

Segmentation masks support a range of geometric and statistical descriptors that are central to multiphase-flow analysis. At the single-frame level, masks can be used to compute bubble count and size-related quantities (e.g., equivalent diameter/area), as well as shape descriptors that summarize deformation [2]. These measurements are often aggregated into distributions (e.g., size histograms) that characterize the state of the dispersed phase and are used to compare operating regimes or to validate physical models [2].

Beyond static morphology, consistent segmentation across frames enables time-resolved analysis. When masks can be associated over time (explicitly via tracking or implicitly via consistent instance delineation), they support estimation of bubble motion and derived dynamic indicators [20]. In boiling and other transient scenarios, such time-resolved segmentation can be used to study bubble evolution patterns and relate them to heat-transfer-relevant behavior [12]. Recent work also highlights that robust bubble detection/segmentation can be a stepping stone towards more advanced tracking pipelines, including tracking under strong overlap and deformation [26, 7].

2.1.2 Imaging challenges and their implications for measurement reliability

The main difficulty in multiphase imagery is not simply that images are “hard to segment”, but that common artefacts directly translate into biased measurements if the segmentation output is not reliable. Two effects are especially consequential.

First, dense multiphase flows produce frequent apparent overlap because 2D recordings are projections of a 3D scene, increasing occlusion and ambiguity at interfaces [6]. Merge/split errors therefore directly bias bubble-resolved statistics, making overlap handling a core requirement when segmentation outputs are used for quantitative analysis [2, 6].

Second, boundary uncertainty (e.g., due to uneven illumination, reflections, blur/defocus, and contrast variability) affects whether masks can be trusted for fine-grained morphology [2, 24]. In practice, these issues concentrate errors around edges and thin interfaces, precisely where small inaccuracies can cause large changes in computed areas for small bubbles, and can propagate into derived statistics in a regime-dependent way [24, 2]. For this reason, segmentation approaches in the literature are commonly evaluated not only by visual plausibility, but also by how robustly they preserve the measurement-relevant structure under challenging conditions [6, 2].

2.1.3 Semantic vs. instance segmentation

Most approaches can be grouped by the type of output they produce, which determines which measurements they support the most directly [6].

Semantic segmentation assigns each pixel a class (e.g., bubble vs. background). This can be sufficient when the objective is phase separation or bulk foreground estimation, but it does not inherently separate touching bubbles, limiting direct per-bubble statistics in dense scenes [6]. In principle, semantic masks can be post-processed to split connected regions (e.g., marker-based separation), but such steps are sensitive to noise and interface ambiguity and can introduce additional failure modes [25, 6].

Instance segmentation outputs one mask per bubble instance, making bubble-level measurements (counting and per-bubble geometry) available by construction, even under partial overlap [2]. This is a key reason why Mask R-CNN-style methods are frequently adopted as baselines for dense multiphase flows, and why instance-level supervision is emphasized when the measurement goal is bubble-by-bubble characterization [2, 6].

Panoptic segmentation combines semantic and instance outputs into a unified representation, but multiphase-flow studies typically focus on semantic or instance formulations depending on whether bulk phase separation or per-bubble statistics is the primary goal [6].

Because the experimental analysis relies on *bubble-resolved* measurements, the key requirement is not only to delineate the overall gas region but also to separate individual bubbles even when they are in close proximity or partially overlapping. Instance-level masks also facilitate split/merge corrections during annotation. Accordingly, this chapter places particular emphasis on instance segmentation methods and on training strategies that make them effective under limited real annotations [2, 6].

2.2 Segmentation methods

Segmentation approaches for multiphase-flow images can be grouped by the type of representation they produce and by how much of the feature extraction is learned from data. In the literature, this typically spans (i) classical image-processing pipelines, (ii) deep learning models for semantic segmentation (often U-Net-based), (iii) deep learning instance-segmentation models (commonly Mask R-CNN variants) that explicitly separate bubbles under overlap, and (iv) foundation segmentation models that enable promptable and interactive mask generation [6, 2, 9, 15].

2.2.1 Classical approaches

Classical pipelines based on thresholding, edge detection, and marker-based separation can be effective when bubbles are well separated and contrast is stable, but they degrade rapidly with overlap, weak boundaries, or illumination artefacts that are frequent in experimental multiphase imagery [6, 20]. Watershed-based strategies can separate touching regions yet are sensitive to noise and marker errors, which may lead to over-segmentation in cluttered scenes [25]. Similarly, edge-based methods such as

the Canny detector can fail when boundaries are blurred or ambiguous, which is common under reflections, defocus, and strong overlap [1, 6]. These limitations motivate learning-based methods that can adapt feature extraction to the visual variability of experimental data [6].

2.2.2 Deep learning: U-Net and semantic approaches

U-Net-style architectures are widely adopted for bubble/background segmentation because their encoder–decoder structure combines contextual information with spatial detail, supporting accurate boundary prediction in many imaging scenarios [6]. In bubble-imaging studies, transfer learning with U-Net variants has been reported to achieve strong performance with relatively limited labeled data, particularly in high-speed visualization settings [17].

However, in dense regimes semantic masks often merge touching or overlapping bubbles into a single connected region, so bubble-by-bubble measurements rely on a subsequent separation stage (e.g., marker-controlled splitting or contour-based post-processing) [6]. Because overlap-boundary errors dominate in these regimes, semantic segmentation is often insufficient as a stand-alone solution when bubble-resolved statistics are required, motivating instance-level formulations [2, 6].

2.2.3 Deep learning: Mask R-CNN and instance segmentation

Mask R-CNN is a common baseline for bubble instance segmentation in dense multiphase-flow imagery [2]. It extends the two-stage detector Faster R-CNN by adding a parallel mask-prediction head for each selected region proposal, producing pixel-level instance masks in addition to class labels and bounding boxes [16, 5]. In this formulation, an RPN proposes candidate regions, RoIAlign extracts fixed-size features, and parallel heads perform classification, bounding-box refinement, and binary mask prediction [5, 16].

A practical challenge in experimental bubble images is the strong multi-scale nature of the scene: small bubbles and larger deformed bubbles may coexist in the same frame. Feature Pyramid Networks (FPNs) are therefore commonly paired with Mask R-CNN backbones because they provide semantically rich feature maps at multiple spatial resolutions, allowing the proposal and mask heads to use higher-resolution features for small instances while preserving strong context for larger ones [10]. In addition, because segmentation outputs are often used for physical measurements rather than solely for visual correctness, several works incorporate post-processing or explicit reconstruction steps to improve boundary plausibility under occlusion and to reduce measurement bias in dense regimes [2].

2.2.4 Foundation models (SAM, SAM2)

Foundation segmentation models introduce a different paradigm: instead of training a task-specific model to directly output masks for a narrow domain, they provide a promptable segmentation interface that can be guided by sparse user inputs (e.g., points or boxes) and refined interactively [9]. The Segment Anything Model (SAM) is built around three components: (i) an *image encoder* that produces

a high-level embedding of the input image, (ii) a *prompt encoder* that converts user prompts (points, boxes, or coarse masks) into an embedding in a compatible space, and (iii) a lightweight *mask decoder* that predicts one or more candidate masks together with a quality score for each proposal [9]. SAM was trained at scale using a large and diverse segmentation dataset (SA-1B), enabling strong zero-shot generalization across domains when prompted appropriately [9].

SAM2 extends this direction with an architecture designed to support segmentation across images and videos, enabling efficient interactive refinement in time-dependent data [15]. While the interaction paradigm remains prompt-based, SAM2 introduces mechanisms to propagate and update object representations across frames (i.e., maintaining temporal consistency), so that prompts provided on one frame can be leveraged to segment subsequent frames with limited additional input [15].

In the multiphase-flow context, recent work has explored combining SAM-style models with domain-specific refinements for bubble imagery, highlighting both the potential and the need for careful handling of challenging cases such as defocus and small bubbles [24]. Importantly, for practical dataset creation, these models also enable model-assisted annotation workflows: annotation platforms such as Label Studio integrate SAM/SAM2-style interaction, allowing annotators to correct boundaries rapidly using prompt-guided masks rather than drawing every instance from scratch. This capability is particularly relevant under limited annotation budgets and directly supports the labeling-efficiency strategies discussed later in Section 2.3 [23].

Quantitative comparison. Table 2.1 provides a quantitative snapshot of representative approaches discussed in this section, including semantic segmentation, instance segmentation with reconstruction, detection-plus-geometry pipelines, and promptable foundation-model workflows [17, 2, 4, 24, 22]. Because studies differ in datasets, operating regimes, and reporting practices, the metrics should be interpreted in context rather than as a direct ranking across works (see Section 2.4) [6].

Gas holdup denotes the volumetric fraction of gas in the flow; higher holdup typically corresponds to denser bubble fields and stronger overlap/occlusion [2].

Overall, the results reflect a consistent trade-off reported in the literature: semantic segmentation can perform well for phase separation but requires additional logic to separate touching bubbles, while instance-level methods better match bubble-by-bubble measurement goals but degrade as overlap and imaging artefacts increase [2, 6]. Detection-focused pipelines can achieve strong AP under overlap, yet do not provide pixel-accurate masks without additional segmentation stages [22]. Promptable foundation models can accelerate refinement and dataset creation, but reported failure cases in bubble imagery (e.g., defocus and small bubbles) motivate careful fine-tuning and human-in-the-loop workflows [24, 9, 15].

Taken together, these findings suggest that robust measurement-oriented segmentation in dense multiphase-flow imagery benefits from combining instance-level modeling with data-efficient adaptation strategies and interactive refinement, rather than relying on a single model family in isolation.

Table 2.1: Quantitative comparison of representative segmentation approaches for multiphase-flow imagery (NR: not reported explicitly in the cited source).

Study	Architecture / type	Dataset context	IoU / mAP	Noted limitations
Seong et al. [17]	U-Net style (semantic)	High-speed experimental imagery	NR (reports overall correct classifications of 98.95%)	Overlap and touching bubbles still require additional handling; the paper notes remaining segmentation errors in challenging regions
Cui et al. [2]	Mask R-CNN + reconstruction	Dense, high-overlap sub-millimeter bubbles	mAP stratified by gas holdup (GH): mAP > 0.95 for GH $< 14.67\%$; still > 0.90 for GH $< 20\%$	Performance degrades as holdup and overlap increase; difficult cases include uneven illumination, shadows, and blurred boundaries
Haas et al. [4]	Faster R-CNN detector + shape regression (BubCNN)	Experimental bubbly-flow imagery	mAP@0.5 = 0.84 (also reports F1 = 0.86)	Detection and shape quality depend on imaging conditions and generalization across setups can require adaptation
Xu et al. [24]	SAM-based bubble segmentation (BubSAM)	Challenging bubbly-flow imagery	Reports Precision/Recall/F1 (precision up to 97%; precision $>99\%$ at low aeration) and a composite score σ ranging from 94.46% (low holdup) to 84.81% (higher holdup)	Reported failure modes include defocus and small bubbles; robustness depends on image quality and scenario
Wen et al. [22]	YOLOv5 detector + tracking (detection-focused)	Overlapping bubbles (tracking pipeline)	AP = 0.90; AP50 = 0.94 (Soft-NMS), 0.92 (no Soft-NMS)	Provides boxes not masks; pixel-accurate delineation would require an extra segmentation stage and tuning

2.3 Training with limited annotations

Reported dataset scales in bubble-segmentation studies are often modest (Table 2.2), commonly on the order of 10^2–10^3 annotated images (e.g., 100 images in [2], 600 in [20], and 800 GT images in [6]), with some works complementing limited ground truth using synthetic data or large-scale pseudo-labeling (e.g., 10k pseudo-labeled images in [8]). This motivates training paradigms that increase effective supervision under limited manual labeling, including synthetic data generation, transfer learning and fine-tuning, and human-in-the-loop/model-assisted annotation workflows [8, 23].

2.3.1 The annotation bottleneck

Producing pixel-accurate instance masks for bubble imagery is time-consuming and inherently difficult in regimes with strong overlap, partial occlusion, and ambiguous boundaries [2, 6]. Dense scenes also increase the likelihood of subjective split/merge decisions, which can introduce label noise and limit the effective quality of the training signal [6].

From a learning perspective, the bottleneck is amplified by the need for diversity. Models trained on a narrowly sampled set of operating conditions or optical settings may not generalize well when illumination, camera parameters, or flow regimes change [2, 8]. As a result, research commonly targets methods that (i) expand the training distribution without proportional manual labeling effort and (ii) leverage existing pretrained representations to adapt effectively from small domain-specific datasets [8, 4].

2.3.2 Synthetic data and the domain gap

Synthetic image generation can provide large volumes of perfectly labeled data at low marginal cost [8, 23]. Synthetic datasets can be constructed to span a range of bubble sizes, shapes, and densities, and can explicitly generate difficult configurations such as overlap and occlusion, which are costly to annotate in real images [2]. In practice, synthetic data is often used either to pretrain a model prior to adaptation on real imagery or to augment a small real dataset with additional labeled examples [8].

A persistent challenge is the *domain gap*: even visually plausible synthetic images may differ from experimental recordings in subtle but important ways, such as boundary texture, illumination non-uniformity, sensor noise, optical blur, or specular reflections [8]. These mismatches can lead to strong degradation when a model trained primarily on synthetic data is evaluated on real experimental images [8]. Consequently, synthetic pipelines are typically paired with invariance-promoting variability (during generation or augmentation) and followed by real-domain fine-tuning to reduce the gap [8, 23].

2.3.3 Transfer learning and fine-tuning

Transfer learning is widely adopted in segmentation because pretrained backbones already encode generic visual features (edges, textures, and mid-level structures), so fine-tuning mainly needs to adapt the higher-level representations to the target imagery, reducing the amount of labeled data needed compared to training from scratch [4]. This is particularly relevant for instance segmentation frameworks such as Mask R-CNN, which are commonly initialized from large-scale datasets and then fine-tuned on domain-specific imagery [5, 16, 10]. In bubble segmentation studies, fine-tuning strategies are often chosen to balance stability and adaptation: for example, updating only higher layers initially (or using reduced learning rates for the backbone) can preserve general visual features while adapting the detection and mask heads to bubble-specific appearance and scale [4, 2].

For semantic segmentation settings, U-Net-style architectures are also frequently trained using transfer learning principles when available, and performance gains are reported even with limited labeled data [17].

2.3.4 Data augmentation

Data augmentation is a practical mechanism to increase effective dataset diversity when annotation budgets are limited. Standard geometric transformations (e.g., flips, rotations, scaling) help reduce sensitivity to viewpoint and spatial configuration, while appearance-based perturbations (e.g., brightness/contrast shifts, blur, and noise injection) can mimic common variability in experimental imaging [8]. For bubble imagery, augmentation is especially useful to expose the model to plausible changes in illumination and boundary clarity that are known to affect segmentation reliability [2, 8].

Augmentation must nevertheless be designed with care. Excessive or unrealistic perturbations can introduce training examples that do not correspond to physically plausible observations, potentially harming generalization [8]. As a result, augmentation policies in multiphase-flow segmentation are typically motivated by known sources of variability in acquisition conditions and by the intended downstream measurements, prioritizing changes that preserve instance geometry while varying appearance in controlled ways [2, 8].

2.3.5 Model-assisted annotation workflows

An increasingly influential direction is to reduce manual labeling cost through interactive, model-assisted annotation. Foundation models such as SAM provide promptable segmentation in which a user supplies sparse guidance (e.g., points or boxes) and iteratively refines the resulting masks [9]. SAM2 extends this paradigm toward efficient segmentation across images and videos, which is particularly relevant when annotations must be propagated through sequences or when temporal consistency is desired [15]. In practice, annotation tools have begun integrating these capabilities; for example, Label Studio supports SAM-based workflows in which annotators correct and refine predicted masks rather than drawing them from scratch.[1]

For bubble imagery, recent studies indicate that such interactive workflows can accelerate mask creation, but may still require careful refinement in difficult cases such as small bubbles, defocus, and ambiguous boundaries [24]. More generally, foundation-model data engines report multi-fold reductions in annotation time as model assistance improves: in SAM, the average annotation time decreased from 34 s to 14 s per mask over successive collection cycles, while in SAM2 the reported annotation time decreased from 37.8 s/frame (manual, per-frame) to 7.4 s/frame and 4.5 s/frame in later model-in-the-loop phases (5.1$\times$ and 8.4$\times$ speedups under their protocol) [9, 15]. A common pattern is iterative dataset growth: preliminary labels are produced with assistance from a model, corrected by a human,

[1] Example implementation: Label Studio SAM2 auto-annotation guide https://labelstud.io/guide/ml

and then used to retrain or fine-tune the segmentation model, progressively improving both model performance and annotation efficiency [23]. This human-in-the-loop perspective is especially aligned with experimental research settings, where new operating conditions may continually introduce distribution shifts and require practical mechanisms for rapid dataset expansion [8].

2.4 Datasets and evaluation

2.4.1 Available datasets

Table 2.2 summarizes representative datasets and dataset-building strategies, highlighting the typical supervision level, reported scale, evaluation metrics, and whether the data are publicly accessible [2, 6, 20, 8, 24]. Pseudo-labeling (or self-training) refers to generating automatic annotations for unlabeled or weakly labeled images using a model, and then using these predicted labels as additional supervision during training [8]. In practice, pseudo-labels are commonly filtered by confidence thresholds and may be selectively corrected before inclusion, increasing effective training data without fully manual labeling.

Table 2.2: Representative datasets and dataset-building strategies used in bubble segmentation studies (NR: not reported explicitly in the cited source).

Work	**Data**	**Output**	**# Img**	**# Inst.**	**Metrics reported**	**Public?**	**Notes (context + relevance)**
Cui et al. [2]	Exp.	Instance + recon.	100	>20k bubbles	Prec., Rec., mAP, F1, IoU	Not stated	Dense sub-millimeter bubbly flow with strong overlap; combines Mask R-CNN-style instance segmentation with shape reconstruction to improve measurement reliability in high-holdup regimes.
Hessenkemper et al. [6]	Exp. + synth.	Instance ID / segm.	800 (GT) + 1000 (synth.)	24.4k (GT bubbles)	AP (IoU-based), detection/segm. errors	Yes	Comparative benchmark including classical and learning-based methods; reports an annotated dataset and synthetic compositions; provides open repositories (dataset/-code/models) (BubMask).
Soibam et al. [20]	Exp. (high-speed)	Instance masks	600 (80/20 split)	2237 (val); total NR	Prec., Rec., AP@0.5, mAP@[0.5:0.95]	On request	Transfer learning for boiling/high-speed imagery; reports detection-style metrics and explicit splits; availability stated as "upon request".
Homan & Deen [8]	Synth. + public real	Instance masks / pseudo-labels	100 (synth.) + 10k (pseudo)	10k (synth. bubbles)	mAP@0.5, mAP@[0.5:0.95]	Partial	"Shoestring" pipeline: synthetic scenes (100 images, 100 bubbles/image) and large-scale pseudo-labeling on a publicly available real dataset; code released, while derived labels depend on the underlying dataset terms.
Xu et al. (BubSAM) [24]	Exp.	Masks + recon.	NR	∼8000 bubbles/-condition	Prec., Rec., F1; composite score σ; recon.-related errors	On request	SAM-based bubble segmentation with reconstruction; reports results across multiple holdup conditions; full dataset not publicly released (data availability: on request).

Overall, Table 2.2 illustrates two recurring patterns: (i) fully annotated experimental datasets remain relatively limited compared with general-purpose vision benchmarks, and (ii) public availability is inconsistent, which complicates reproducible comparison across studies [6, 2, 8]. These constraints motivate training strategies that maximize data efficiency and robustness, discussed in Section 2.3 [8, 23].

2.4.2 Evaluation metrics and practices

Evaluation protocols in multiphase-flow segmentation typically combine pixel-level overlap measures with instance-level detection-style measures [2, 6]. For semantic segmentation, the most common overlap metric is Intersection over Union (IoU), often complemented by Dice/F1-type overlap scores and pixel-level precision/recall [2, 24]. When instance masks are required, studies frequently adopt Average Precision (AP) and mean Average Precision (mAP) computed by matching predicted instances to ground-truth instances under IoU criteria (e.g., AP@0.5 and mAP over multiple IoU thresholds) [6, 20, 2].

Several works report measurement-oriented indicators derived from masks, such as bubble counting accuracy and errors in size-related quantities (e.g., area or equivalent diameter) and their distributions [2, 24]. Moreover, given the strong dependence on operating regime and imaging setup, evaluation is often stratified by conditions (e.g., increasing gas holdup) or performed across distinct acquisition settings to assess generalization, and this regime-aware reporting is essential for interpreting whether a model is reliable for experimental use [2, 8, 24].

Implication for this dissertation. Pixel-level overlap measures (e.g., IoU/Dice and pixel Precision/Recall) are informative about overall foreground delineation, but they can remain high even when instances are incorrectly merged or split, which directly biases measurement outputs.

2.5 Research gaps

Although modern segmentation architectures and promptable foundation models have improved performance and usability, reliable bubble segmentation in experimental multiphase imaging remains limited by a small set of practical gaps.

(1) Robustness across regimes and setups is still not guaranteed. Reported results are often tied to specific operating conditions and imaging configurations, and performance can degrade under increased overlap, blur/defocus, and illumination variability [2, 8, 24]. This makes it difficult to assume that a model trained or validated in one setting will remain reliable when experimental conditions change.

(2) Data efficiency remains a bottleneck for instance-level supervision. Bubble-by-bubble measurements favor instance segmentation (Section 2.1), yet dense scenes make pixel-accurate instance labeling costly and inconsistent, and public datasets with comparable annotation standards remain limited [2, 6, 20]. As a consequence, data-efficient training strategies are central when building models for new experimental conditions (Section **??**) [8, 23].

(3) Evaluation is not always aligned with measurement impact. IoU and AP/mAP are standard, but they do not necessarily reveal whether segmentation errors introduce unacceptable bias in downstream quantities (e.g., counts and size distributions), especially in dense regimes where split/merge errors dominate [2, 6]. More consistent reporting of regime-stratified performance and measurement-oriented error indicators would improve comparability and experimental trustworthiness [2, 24]. While a comprehensive measurement-aligned evaluation framework is beyond the scope of this dissertation, this limitation motivates complementing overlap metrics with instance-derived indicators in Section 3.6.

(4) The practical value of interactive model-assisted annotation needs clearer quantification. SAM/SAM2-style workflows can accelerate annotation, but bubble imagery includes difficult cases (small bubbles, defocus, ambiguous edges) where interactive refinement is still required, and the resulting time savings and performance gains are not yet consistently quantified across conditions [9, 15, 24].

Implications for this dissertation. These gaps motivate the research questions defined in Section 1.3. First, the limited robustness reported across regimes and imaging setups motivates a controlled comparison of selected instance-segmentation backbones on the same real experimental dataset (RQ1). Second, the persistent data-efficiency bottleneck and synthetic–real domain gap motivate an explicit study of synthetic pretraining followed by fine-tuning on a limited set of real annotations, including the role of transfer strategies in improving real-domain performance (RQ2). Third, the practical cost of producing high-quality instance masks motivates a quantitative assessment of model-assisted annotation workflows that combine synthetic pre-annotations with SAM2-assisted interactive refinement in Label Studio (RQ3). Finally, although a comprehensive measurement-aligned evaluation framework is beyond the scope of this dissertation, the evaluation gap motivates reporting instance-derived indicators alongside standard overlap metrics in the protocol adopted in the following chapters.

Chapter 3

Methodology

3.1 Overview of the research workflow

This dissertation follows an end-to-end workflow that connects dataset preparation, model training, evaluation, and model-assisted annotation into a reproducible pipeline. The goal is to study bubble *instance segmentation* in experimental multiphase-flow images under realistic constraints, namely limited labeled data and strong variability across experimental conditions.

To avoid ambiguity about how the different datasets, training stages, and evaluation activities connect, Figure 3.1 summarizes the complete methodology as a single pipeline. In particular, it highlights (i) the parallel use of a synthetic COCO-formatted dataset and an experimentally annotated COCO dataset, (ii) the four training conditions repeated for each architecture (synthetic training, real baseline training, synthetic-to-real transfer learning, and transfer learning with augmentation), and (iii) the two outcome streams of the work: quantitative model evaluation on the experimental test split and a separate annotation-efficiency study using model-assisted labeling.

The workflow is organized as follows:

1. **Data preparation and label standardization.** Two data sources are used—synthetic bubble images and experimental frames—and both are standardized to a common COCO-style instance-segmentation format for training and evaluation [11] (Sections 3.2.1 and 3.2.3). Experimental masks are produced in Label Studio and exported to COCO for consistent downstream processing.

2. **Baseline model selection and synthetic training.** A set of deep-learning instance segmentation baselines is trained on synthetic data, with Mask R-CNN variants playing a central role [5]. The approach builds on the two-stage detection paradigm introduced by Faster R-CNN [16] and employs feature pyramid backbones when applicable to support multi-scale representation [10]. Model configuration is controlled via YAML experiment files to enable systematic comparisons

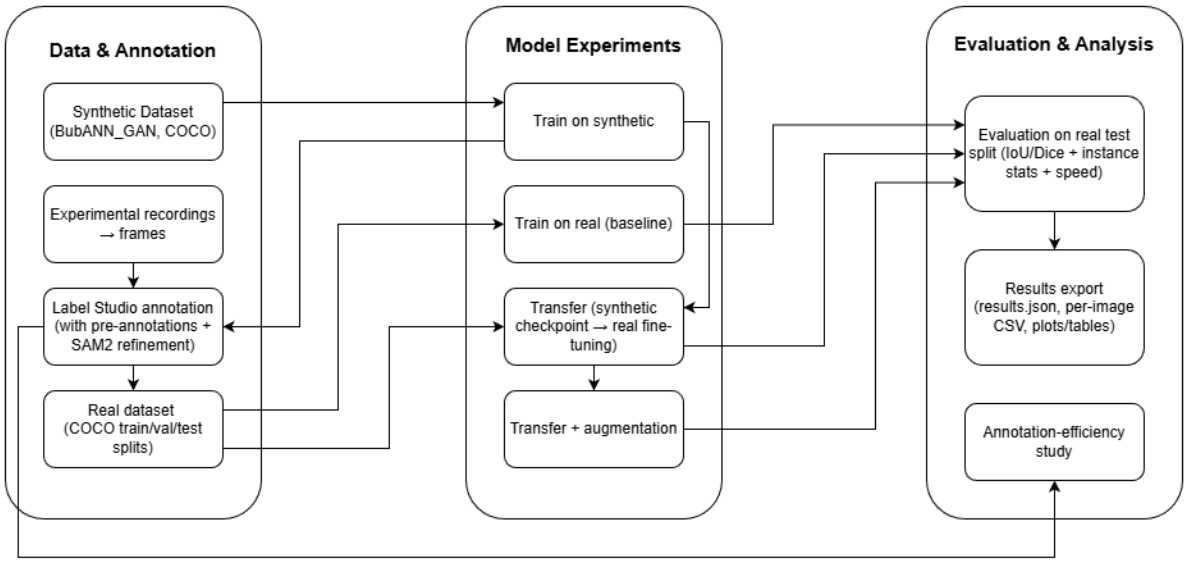


Figure 3.1: End-to-end research workflow adopted in this dissertation: dataset preparation and annotation, repeated model experiments (synthetic training, real baseline training, synthetic-to-real transfer learning, and transfer learning with augmentation), evaluation on a held-out experimental test split, and an annotation-efficiency study using model-assisted labeling.

across architectures and training settings, and reference implementations are based on PyTorch/-TorchVision [14][1]

3. **Experimental annotation and model-assisted labeling.** To reduce the cost of producing pixel-accurate instance masks in experimental images, the workflow includes model-assisted labeling in Label Studio. In this loop, pre-annotations (generated automatically) are refined by the annotator, consistent with semi-automatic annotation strategies described in the computer-vision literature [23]. Interactive refinement is incorporated using SAM2 to accelerate boundary correction through prompt-guided mask editing [15]. An annotation-efficiency study compares manual-only annotation against workflows with pre-annotations and SAM2-assisted refinement.

4. **Real-domain training and synthetic-to-real transfer learning.** For each architecture, four training regimes are evaluated—synthetic, real baseline, synthetic-to-real transfer, and transfer with augmentation—as defined in Section 3.4.3 [13, 8, 18].

5. **Model evaluation on experimental test data and results export.** After training, performance is evaluated on held-out experimental test images using a unified protocol that reports (i) pixel-level overlap metrics (e.g., IoU and Dice, with precision and recall), (ii) instance-level bubble statistics (e.g., bubble counts and area summaries, including counting error), and (iii) indicative inference

[1] TorchVision model documentation for `maskrcnn_resnet50_fpn_v2`: https://docs.pytorch.org/vision/main/models/generated/torchvision.models.detection.maskrcnn_resnet50_fpn_v2.html

throughput (average inference time and frames per second). Results are exported in a structured format to support cross-experiment analysis.

6. **Experiment tracking and reproducibility.** Experiments are managed through consistent configuration files and structured outputs. Implementation is based on PyTorch and TorchVision, providing stable reference implementations for detection and instance segmentation models [14].

3.2 Data sources and label formats

This dissertation uses two complementary data sources: a synthetic dataset with fully available instance masks and an experimental dataset obtained from multiphase-flow recordings. The synthetic data supports large-scale supervised training under controlled conditions, while the experimental data is required to evaluate generalization and to study synthetic-to-real fine-tuning under realistic artefacts such as overlap, reflections, and illumination variability.

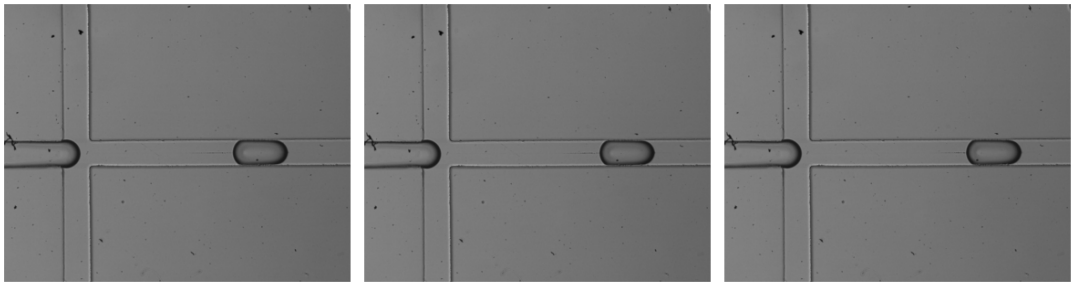
3.2.1 Synthetic dataset

The synthetic training data is based on the **BubANN_GAN** dataset, which provides bubble instances with pixel-level annotations in a COCO-style segmentation format. The version used in this dissertation contains approximately **1.5k images** and is distributed with predefined **training, validation, and test** splits.

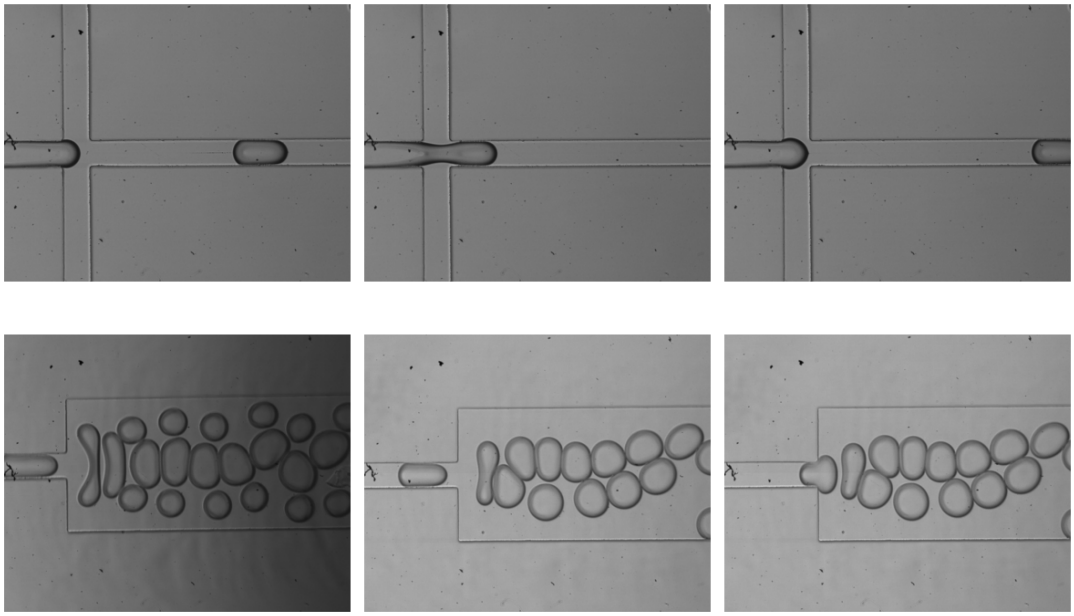
The dataset follows the common COCO-segmentation directory structure, where each split contains the images and a corresponding `_annotations.coco.json` file:

```
BubANN_GAN.v1i.coco-segmentation/
 train/
    _annotations.coco.json
    *.jpg
 val/
    _annotations.coco.json
    *.jpg
 test/
     _annotations.coco.json
     *.jpg
```

No additional transformations are permanently applied to the dataset on disk. Instead, resizing and augmentations (when enabled) are applied *on-the-fly* during training to ensure consistent preprocessing across synthetic and experimental data (Section 3.4).



(a) Examples from the extracted pool (redundancy across adjacent frames).



(b) Examples from the selected set (higher visual diversity).

Figure 3.2: Illustration of the frame-selection step used to build the annotated experimental dataset.

3.2.2 Experimental dataset and frame selection

Experimental bubble images were provided in collaboration with CEFT/FEUP as video recordings from multiphase-flow experiments. From these videos, a total of **1,397** individual frames were extracted to form an initial pool of candidate images.

Only a subset of these frames was selected for annotation. This selection was performed to maximize visual diversity (for example, different bubble sizes, overlap regimes, and lighting conditions) and to reduce redundancy caused by near-duplicate frames originating from the same short time intervals. As a result, **243 frames** were chosen as a first representative set for annotation. This curated set was treated as an initial supervised dataset that could be expanded in future work; however, due to time constraints, the final experimental dataset used in this dissertation is based on these **243 annotated images**.

3.2.3 Annotation tool and export formats

All experimental ground-truth masks in this dissertation were annotated by a single annotator (the author) using **Label Studio**[2], following a consistent annotation criterion across images and annotation modes. Label Studio was used both for fully manual annotation and for model-assisted refinement workflows. The 243-image experimental dataset was annotated primarily using the assisted workflow described in Section 3.5 (synthetic pre-annotations refined with SAM2), with fully manual tools used when pre-annotations were insufficient. For training and evaluation, annotations were exported to a COCO-compatible instance segmentation format [11], which represents each bubble instance through: (i) a per-instance segmentation (polygon-based representation), (ii) a bounding box, and (iii) an image-level identifier linking instances to the source frame.

In addition to COCO export, Label Studio task JSON was also used when operating with model-assisted annotation and interactive refinement, because it supports importing pre-annotations as prediction layers and refining them within the annotation interface.

3.2.4 Dataset splits and preprocessing

Both datasets follow the COCO-segmentation split convention (`train/val/test`); the directory structure is shown in Section 3.2.1, and the experimental annotations are exported to match it.

The synthetic dataset is already provided in this format and is already split. For the experimental dataset, the annotated export is post-processed to (i) normalize image file names, (ii) reindex image and annotation identifiers, and (iii) generate reproducible train/validation/test splits with a fixed seed. In addition, images are grouped by filename prefix before splitting to reduce leakage between visually similar frames (for example, consecutive frames extracted from the same original sequence). The split ratios used in this dissertation are 70% training, 20% validation, and 10% test. In addition, a separate subset of 14 images from the experimental dataset is reserved for the annotation-efficiency study (Section 3.5); this subset is used only for timing comparisons and is not part of the test-set performance evaluation.

Images are stored at native resolution; input resizing and the training-time augmentation policy are defined in Section 3.4.4.

[2]Label Studio documentation: `https://labelstud.io/`

Table 3.1: Datasets used in this dissertation and their role in the experimental workflow.

Source	**Quantity**	**Supervision**	**Role in this dissertation**
BubANN_GAN	∼1.5k images (pre-split)	Instance masks (COCO)	Large-scale supervised training and baseline comparison under controlled synthetic conditions
Experimental frames (CEFT/FEUP)	1,397 extracted; 243 annotated	Instance masks (Label Studio → COCO)	Real-domain evaluation, fine-tuning (synthetic-to-real transfer learning), and annotation workflow analysis

3.3 Segmentation models evaluated

Architectural background and design choices (including multi-scale considerations) are summarized in Section 2.2.3. This section defines the specific TorchVision Mask R-CNN variants used in the quantitative experiments.

3.3.1 Model family and evaluated variants

The quantitative experiments in this dissertation focus on the Mask R-CNN family, which is widely used as an instance segmentation baseline and has been successfully applied to dense multiphase-flow imagery with strong overlap [2]. Mask R-CNN combines region-based detection with per-instance mask prediction [5, 16]. All evaluated variants use an FPN backbone; the motivation for FPN-based multi-scale features is discussed in Section 2.2.3.

To study how backbone capacity and implementation choices impact accuracy and inference throughput under the same training and evaluation protocol, three TorchVision Mask R-CNN variants are evaluated:

- `maskrcnn_resnet50_fpn`: ResNet-50 backbone with FPN, representing a widely used mid-capacity baseline with comparatively lower compute and faster inference.
- `maskrcnn_resnet50_fpn_v2`: an updated TorchVision implementation of the ResNet-50-FPN variant, retaining similar capacity but incorporating implementation/training refinements in the reference model, making it a practical comparison point for "same-backbone" improvements.
- `maskrcnn_resnet101_fpn`: ResNet-101 backbone with FPN, increasing depth and representational capacity at the cost of higher compute and slower inference, which is expected to benefit difficult cases (dense overlap, blur, and low-contrast boundaries) but may reduce throughput.

This set was chosen because it provides a controlled sweep over backbone depth and practical runtime without changing the overall detection-and-mask architecture: ResNet-50 and ResNet-101 are standard reference backbones, while the ResNet-50-FPN v2 variant isolates the effect of updated TorchVision reference design choices under essentially the same nominal capacity. Using TorchVision implementations also ensures consistent heads, anchors, and training code across variants, so observed differences can be attributed primarily to backbone capacity and the reference-model versioning rather than to custom re-implementations.

3.3.2 Configuration and implementation choices

All models are trained and evaluated within a unified PyTorch-based pipeline using TorchVision model implementations and pretrained weights when applicable [14]. Experiment configuration is managed through YAML files (Section 3.4), which define the model variant, training settings, dataset splits, and evaluation thresholds used in each run.

Output definition and class convention. The task is formulated as *binary* instance segmentation with a single foreground class (`bubble`). Following the standard COCO convention [11], the background is treated implicitly, so the model is configured with `num_classes = 2` (background + one foreground class). Each predicted bubble instance is represented by a mask, a bounding box, and a confidence score.

Fixed post-processing for comparable measurements. Inference outputs are filtered using a single, fixed set of thresholds shared by all runs (Section 3.6.2) so that differences in bubble count and size statistics reflect model behavior rather than per-run tuning.

3.4 Training pipeline and experiment configuration

All training and evaluation runs are specified through YAML configuration files. Each configuration defines dataset paths, the model variant, optimization hyperparameters, optional augmentation switches, output directories, and the fixed evaluation thresholds used during testing. This section describes the configuration structure and the default values adopted across experiments.

3.4.1 Configuration-driven experiments (YAML)

Each YAML file is organized into logically separated blocks:

- `experiment`: experiment name and random seed.
- `model`: model identifier and task definition (`num_classes`).
- `dataset`: paths to `train`, `val`, and `test` splits (COCO-style).

- `train`: hyperparameters (batch size, learning rate, scheduler, workers) and early stopping (`patience`).
- `augmentation`: preprocessing (e.g., resize) and optional random augmentations.
- `logging`: TensorBoard log directory (and a `use_tensorboard` switch when supported).
- `output`: base directory where checkpoints and logs are stored.
- `eval`: post-processing thresholds used during evaluation (score threshold, minimum area, mask threshold, and optional visualization seed).
- `transfer` (transfer runs only): checkpoint initialization and adaptation controls (`init_checkpoint`, `strict_load`, `freeze_backbone_epochs`).

Table 3.2 summarizes a representative YAML configuration used for standard training on the synthetic dataset.

Table 3.2: Representative YAML experiment configuration used for a standard synthetic-data training run (flattened key notation).

Key	**Example value**	**Purpose**
`experiment.name`	maskrcnn_resnet50_fpn_synth	Experiment identifier used for output folders/logs.
`experiment.seed`	`42`	Pseudo-random seed applied to Python/NumPy/PyTorch.
`model.type`	maskrcnn_resnet50_fpn	TorchVision model variant.
`model.num_classes`	`2`	Background + one foreground class (bubble).
`model.pretrained_backbone`	`true`	Initialize backbone from pre-trained weights.
`dataset.train`	`../../..` `/BubANN_GAN.v1i.` `coco-segmentation/` `train`	Training split (COCO-style directory).
`dataset.val`	`../../..` `/BubANN_GAN.v1i.` `coco-segmentation/` `valid`	Validation split.
`dataset.test`	`../../..` `/BubANN_GAN.v1i.` `coco-segmentation/` `test`	Test split.
`train.batch_size`	`2`	Mini-batch size.

Continued on next page

Table 3.2 (continued)

Key	Example value	Purpose
train.epochs	50	Maximum epochs (early stopping may stop earlier).
train.lr	0.001	Base learning rate for heads.
train.backbone_lr_factor	0.1	Backbone LR multiplier relative to train.lr.
train.momentum	0.9	SGD momentum.
train.weight_decay	0.0005	L2 weight decay.
train.lr_step_size	7	StepLR decay period (epochs).
train.gamma	0.1	StepLR multiplicative decay factor.
train.num_workers	4	DataLoader workers.
train.print_freq	20	Training logging frequency (iterations).
train.patience	7	Early-stopping patience (epochs).
augmentation.enabled	false	Enable augmentation during training.
augmentation.resize	[512, 512]	Input resize applied consistently across splits.
augmentation.hflip_p	0.0	Horizontal flip probability.
augmentation.strong_color_jitter	false	Enable stronger brightness/contrast jitter.
augmentation.blur	false	Enable blur augmentation.
augmentation.intensity_shift	false	Enable sharpness/intensity adjustment.
augmentation.rotation	false	Rotation augmentation switch.
logging.use_tensorboard	true	Enable TensorBoard logging.
logging.log_dir	runs/	TensorBoard run directory.
output.dir	outputs/	Base directory for checkpoints and logs.
eval.score_thresh	0.5	Confidence threshold for keeping predictions.
eval.min_area	50	Minimum predicted mask area (pixels).
eval.mask_thresh	0.5	Mask binarization threshold.

3.4.2 Default configuration values and rationale

Mask R-CNN runs use the same default training and evaluation configuration (Table 3.3). Inputs are resized to 512×512 and optimized with SGD (momentum and weight decay) following TorchVision reference practice for detection and instance segmentation training[3]. A conservative base learning rate (10^{-3}) with StepLR decay ($\gamma = 0.1$, `lr_step_size`=7) is used given the batch size 2 constraint; transfer-learning-specific settings are described in Section 3.4.3.

The batch size is fixed to 2 across experiments to remain within the available GPU memory budget while maintaining stable training dynamics. Training is capped at 50 epochs and uses early stopping (`patience`=7); the validation criterion and checkpoint selection procedure are described in Section 3.4.5.

Transfer-learning-specific controls (checkpoint initialization and optional backbone freezing) are listed in Table 3.3; the transfer-learning procedure is described in Section 3.4.3.

Table 3.3: Default training and evaluation parameters used across Mask R-CNN experiments.

Parameter	**Default value**
Input resize	512×512
Batch size	2
Epochs (max)	50
Optimizer	SGD (momentum 0.9, weight decay 5×10^{-4})
Learning rate	10^{-3}
Backbone LR factor	0.1
Scheduler	StepLR (step=7, γ=0.1)
Early stopping patience	7
Transfer backbone freeze	3 epochs (transfer runs)
Eval score threshold	0.5
Eval mask threshold	0.5
Eval min area	50 pixels

3.4.3 Training regimes and execution

The experimental protocol repeats the same training logic across four regimes for each evaluated model: (i) **synthetic training**, (ii) **real baseline training**, (iii) **synthetic-to-real transfer learning**, and (iv) **transfer learning with augmentation**. Only the dataset domain, checkpoint initialization, and the augmentation flag differ; all other defaults follow Table 3.3.

Regimes (i) and (ii) are standard training runs configured for different dataset domains (synthetic vs. real) through the YAML dataset paths, while keeping preprocessing, optimization, and evaluation thresholds unchanged. Regimes (iii) and (iv) additionally include a `transfer` block specifying a synthetic-trained checkpoint used to initialize fine-tuning on the real dataset; regime (iv) further enables training-time augmentation.

[3] https://docs.pytorch.org/tutorials/intermediate/torchvision_tutorial

Transfer learning is implemented by loading `transfer.init_checkpoint` (optionally with strict state-dict matching), and optionally freezing the backbone for a small number of epochs using `transfer.freeze_backbone_epochs`. After the freeze period, the backbone is unfrozen and training continues with a reduced backbone learning rate via the `backbone_lr_factor`. This follows common transfer-learning practice for adapting pretrained features to a new domain under limited labeled data [13].

3.4.4 Dataset interface, preprocessing, and augmentation

All experiments load annotations from the COCO-compatible export described in Section 3.2.3 [11]. During loading, polygon and/or RLE segmentations are decoded into binary instance masks and paired with bounding boxes in `XYXY` format. Category indices are mapped so that the foreground class `bubble` is label 1, with background treated implicitly as label 0.

Images are stored at their original resolution in the dataset folders and resized to 512×512 before tensor conversion, following the default configuration (Table 3.3). This resolution balances detail retention with the GPU memory constraint (batch size 2).

Random augmentation is controlled by `augmentation.enabled`. When enabled for training, the pipeline can apply horizontal flipping, optional color jitter, optional blur, and a sharpness adjustment (controlled by `intensity_shift`). Rotation is intentionally not applied for Mask R-CNN runs in this pipeline to avoid inconsistencies with region-based instance targets; it is reserved for semantic models only. This controlled augmentation design follows standard recommendations for improving generalization while keeping target consistency under geometric transforms [18].

3.4.5 Optimization, checkpointing, and reproducibility

Optimization uses SGD with momentum and weight decay. To stabilize fine-tuning of pretrained features, the optimizer is built with two parameter groups: detection/mask heads use the base learning rate, while backbone parameters use `lr`×`backbone_lr_factor`. Learning-rate scheduling follows a StepLR schedule configured by `lr_step_size` and `gamma`. When training on GPU, automatic mixed precision is enabled through gradient scaling to reduce memory use and improve throughput.

Validation is performed at every epoch. For Mask R-CNN runs, model selection follows COCO-style instance segmentation evaluation, and the best checkpoint is selected by the segmentation AP averaged over IoU thresholds (AP@[0.50:0.95]) [11]. Early stopping is applied when this validation criterion does not improve for `patience` epochs, and the best model parameters are stored as `model_best.pth`. The 50-epoch budget is therefore an upper bound rather than a target. In this setup, potential overfitting would manifest as continually decreasing training loss while validation mAP stagnates or degrades, whereas underfitting would appear as persistently high losses and low validation mAP; training dynamics are monitored through TensorBoard loss curves and the validation mAP trajectory.

Each experiment records the full YAML configuration in the run logs (TensorBoard text summary), ensuring traceability of hyperparameters, thresholds, and dataset paths across runs [14]. Random-seed handling and determinism considerations are described in Section 3.7.

3.4.5.1 Training objective: Mask R-CNN multi-task loss

Mask R-CNN is trained with a multi-task objective that combines proposal generation (RPN), detection, and instance-mask prediction. Following the standard formulation, the total loss is the sum of five terms:

$$\mathscr{L} = \mathscr{L}_{\text{rpn\_cls}} + \mathscr{L}_{\text{rpn\_box}} + \mathscr{L}_{\text{cls}} + \mathscr{L}_{\text{box}} + \mathscr{L}_{\text{mask}}. \tag{3.1}$$

Here, $\mathscr{L}_{\text{rpn\_cls}}$ and $\mathscr{L}_{\text{rpn\_box}}$ supervise the Region Proposal Network (objectness classification and anchor-box regression). The detection head is supervised by $\mathscr{L}_{\text{cls}}$ (classification of RoIs) and $\mathscr{L}_{\text{box}}$ (bounding-box regression), while $\mathscr{L}_{\text{mask}}$ supervises pixel-wise mask prediction for the foreground class within each positive RoI. In the TorchVision implementation used in this dissertation, these components are returned as a dictionary (e.g., `loss_classifier`, `loss_box_reg`, `loss_mask`, `loss_objectness`, `loss_rpn_box_reg`) and summed to obtain the total loss used for optimization.

Final evaluation metrics and exported artifacts are described in Section 3.6.

3.5 Annotation-efficiency study: synthetic pre-annotations and SAM2-assisted refinement

A Mask R-CNN model (ResNet-50-FPN) trained on the BubANN_GAN synthetic dataset was used to generate instance-mask pre-annotations for a subset of real experimental images. This section defines a controlled protocol to quantify how (i) synthetic pre-annotations and (ii) SAM2-assisted interactive refinement affect annotation effort in Label Studio, using the platform's built-in time tracking (`lead_time`). This timing study is separate from the quantitative segmentation evaluation reported in Chapter 4 (25-image held-out real test split). SAM2 was used through the Label Studio ML backend integration, using the default configuration `sam2_hiera_l.yaml` and checkpoint `sam2_hiera_large.pt`[4].

3.5.1 Study design

From the full pool of 1,397 experimental frames (Subsection 3.2.2), a subset of 14 images was selected to span a range of bubble densities, magnifications, contrast levels, and interface complexity. The same 14 images are used in all annotation modes to enable paired, within-image comparisons of annotation time. Annotation time is reported using `lead_time` recorded by Label Studio for each annotated task.

[4] https://github.com/HumanSignal/label-studio-ml-backend/tree/master/label_studio_ml/examples/segment_anything_2_image

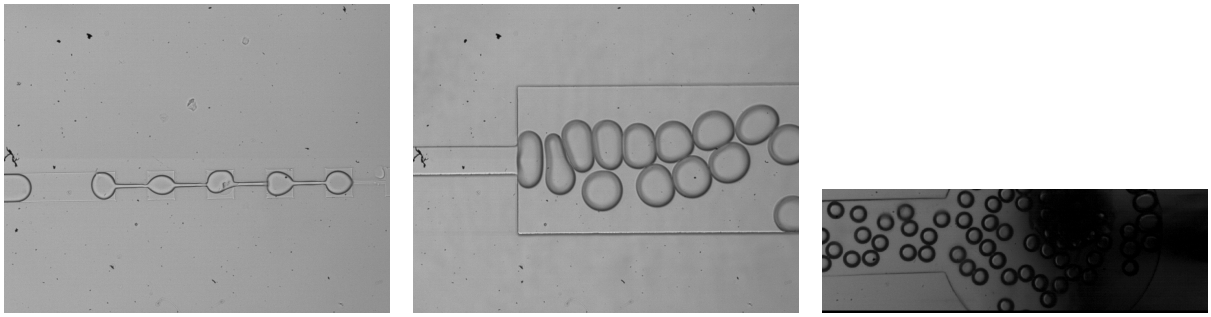
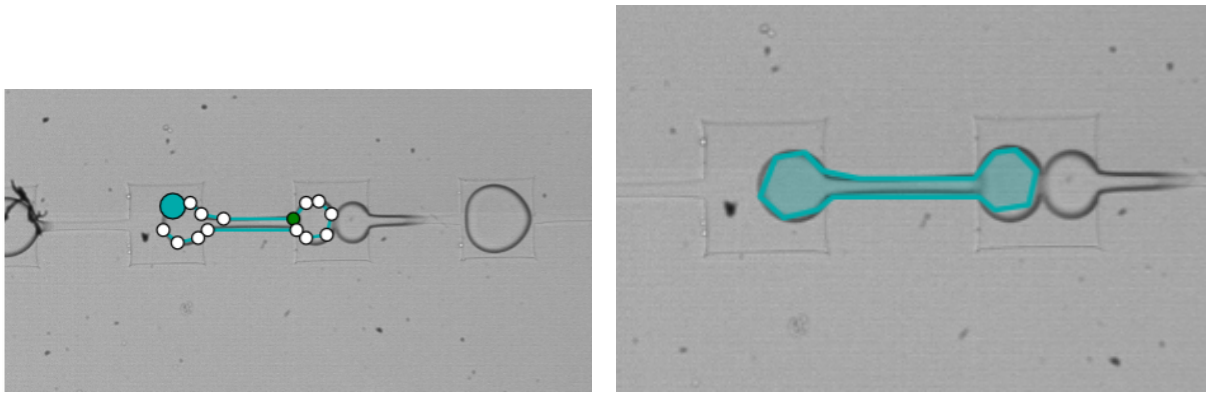


Figure 3.3: Representative examples from the 14 selected frames, illustrating variability in bubble density,magnification, and contrast targeted in the annotation-efficiency study.



(a) Editing view. (b) Final mask.

Figure 3.4: Manual polygon-based annotation in Label Studio (Mode A), used when bubble contours are irregular or require careful contour.

Figure 3.3 illustrates the targeted variability. All annotation modes were performed by a single annotator (the author) to control for inter-annotator variability. As a result, absolute times may differ for annotators with different levels of experience.

Three annotation modes were defined to isolate the effect of automation:

Mode A (manual baseline). All bubble masks were created from scratch in Label Studio using manual tools (brush and polygon). This reflects conventional manual annotation practice reported in multiphase-flow segmentation work [25, 17]. The brush tool is efficient for near-elliptical bubbles, while the polygon tool provides vertex-level control for irregular boundaries and strong overlap. Figures 3.4 and 3.5 illustrate typical baseline cases.

Mode B (synthetic pre-annotations). A Mask R-CNN ResNet-50-FPN model trained on BubANN_GAN was used to generate instance mask predictions for the same 14 experimental images. Predictions were filtered using fixed score and minimum-area thresholds, converted into polygon annotations (via contour extraction / polygonization of binary masks), and imported into Label Studio as a prediction layer. The annotator then corrected missed instances, removed false positives, and refined inaccurate boundaries instead of drawing all masks from scratch. This is consistent with semi-automatic annotation strategies

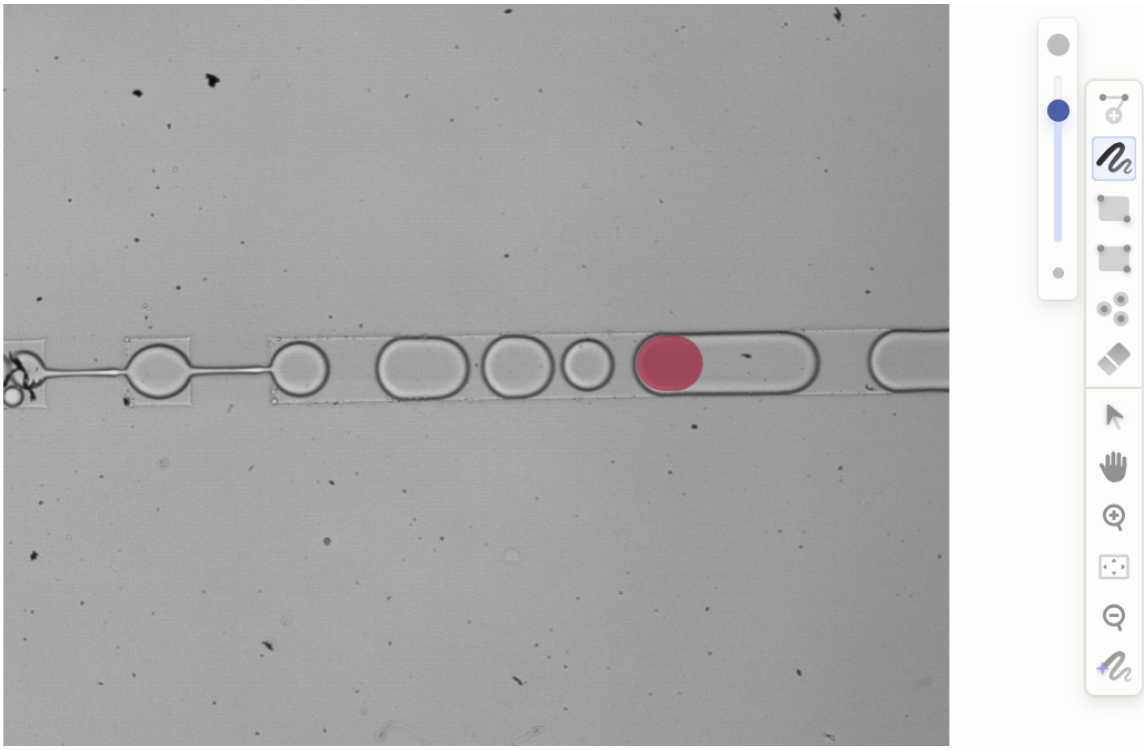


Figure 3.5: Manual brush-based annotation in Label Studio (Mode A), used for regular and approximately circular/elliptical bubbles.

commonly used for detection/segmentation datasets [23].

Figure 3.6 shows a case requiring minimal intervention, while Figures 3.7–3.8 illustrate failure patterns (missed bubbles, over-segmentation, or boundary drift) that require correction.

Mode C (pre-annotations + SAM2 refinement). Mode C starts from the same Mode B prediction layer but enables SAM2 within Label Studio to accelerate interactive refinement. SAM2 refinement is triggered using point prompts (click inside/outside a bubble) and/or box prompts around regions of interest, and its predicted masks are used to replace coarse or incorrect regions before final manual adjustments. This leverages prompt-based segmentation to speed up boundary corrections in difficult regions [15]. Figure 3.9 illustrate typical interactions and a challenging example requiring refinement.

An annotation task was considered complete when **all visible bubble instances** in the frame were delineated with a closed instance mask. Bubbles were annotated whenever a visually discernible interface was present (including partially visible bubbles at the image border). Ambiguous or low-contrast regions were annotated according to the most consistent visible boundary, keeping the same interpretation across modes (A/B/C) to ensure that timing comparisons reflect tool support rather than changes in labeling strictness.

The same completion criterion was applied during construction of the full 243-image experimental dataset (for the later experiments); in practice, the assisted workflow (Mode C) was used for the majority of images, with manual editing used only when pre-annotations were insufficient.

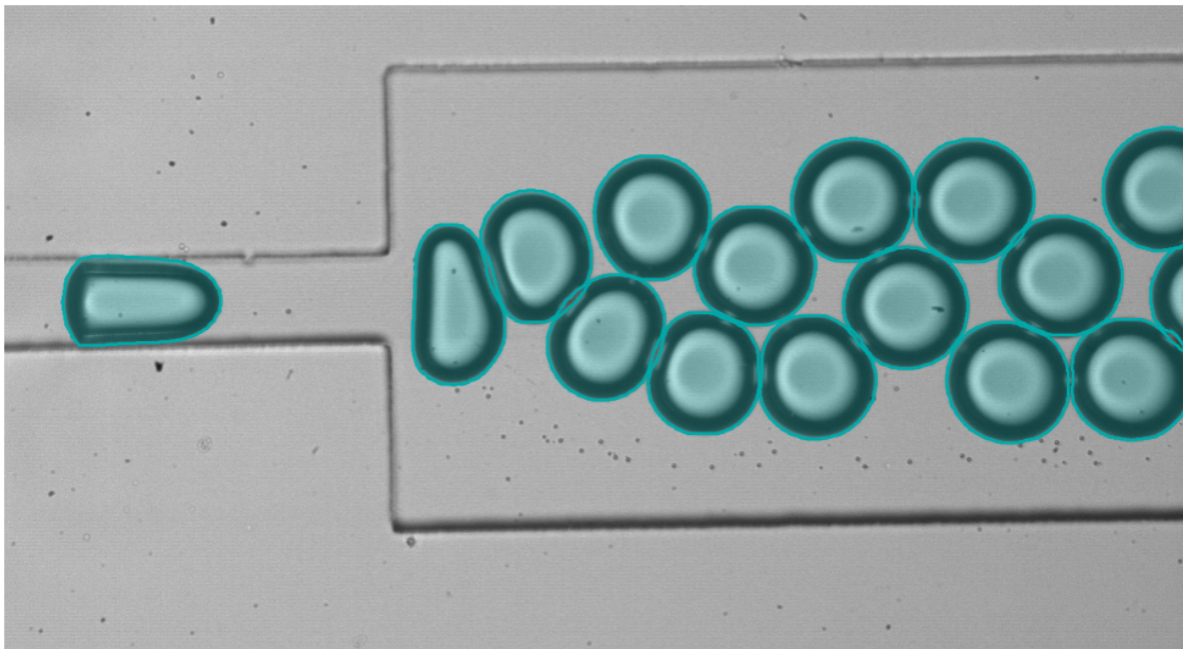
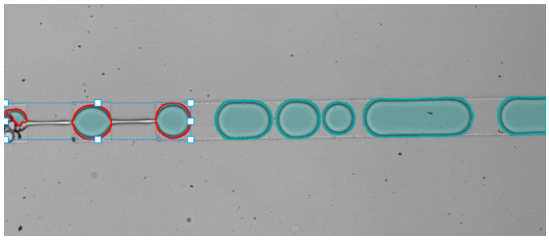
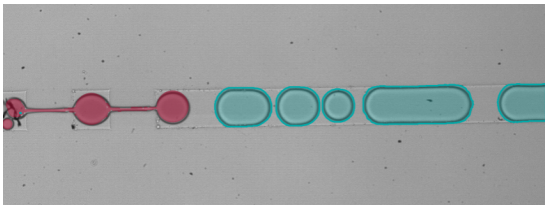


Figure 3.6: Mode B example where synthetic pre-annotations closely match the target masks and require only minor edits.



(a) Removed.



(b) Corrected.

Figure 3.7: Mode B correction examples: removal of incorrect instances, and final corrected annotation.

3.5.2 Timing signal and derived efficiency metrics

Label Studio records annotation effort per task using the `lead_time` field (seconds between opening and submitting a task)[5]. For each mode (A/B/C), `lead_time` was exported for the 14 images. Because `lead_time` can include short idle periods (e.g., pauses during annotation), results are interpreted as practical end-to-end effort under the same tool and interface.

From each exported task, the final number of annotated bubble instances was computed, and the following per-image indicators were derived for each mode: (i) total seconds per image (`lead_time`), (ii) number of bubbles, (iii) seconds per bubble, and (iv) bubbles per minute. Outputs were stored as long-format (image $\times$ mode) and wide-format (one row per image, columns per mode) CSV tables, and summary statistics (mean/median/std) were computed per mode. Finally, per-image percentage

[5]Label Studio task timing field `lead_time`: https://labelstud.io/

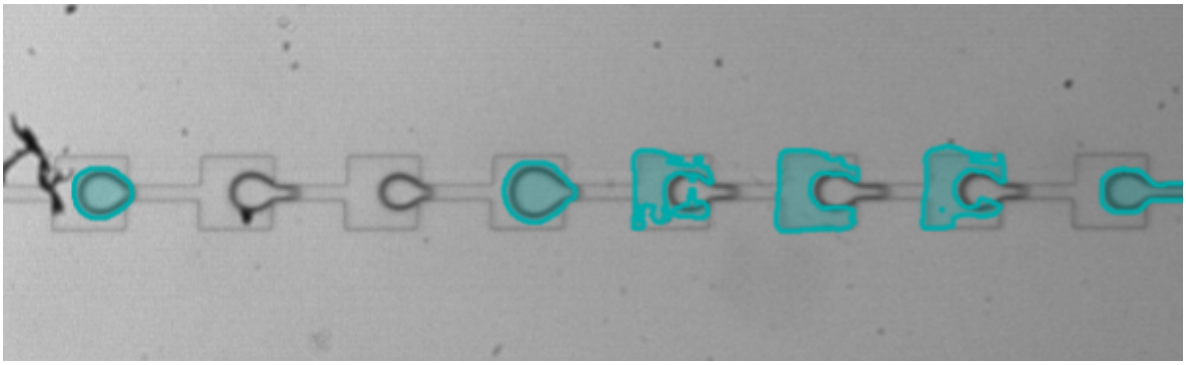
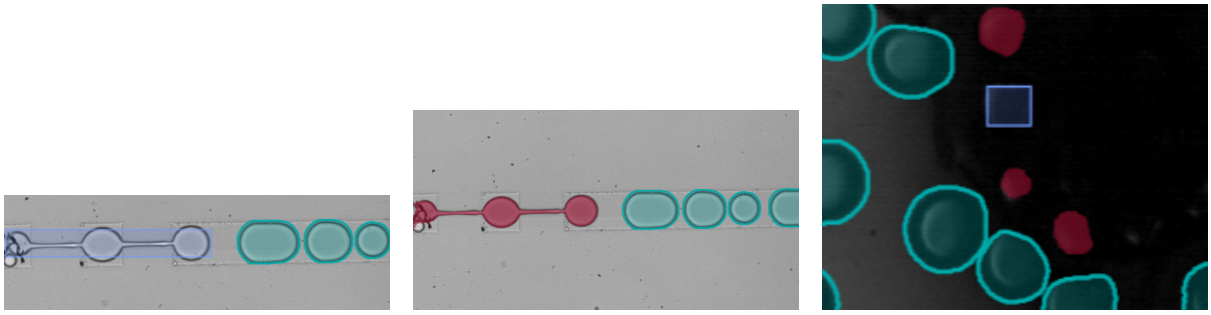


Figure 3.8: Mode B failure example where pre-annotations are significantly incorrect and require substantial manual correction.



(a) Prompt. (b) Mask. (c) Hard case.

Figure 3.9: Mode C (SAM2-assisted refinement) in Label Studio: example interaction, resulting mask, and a challenging case requiring additional refinement.

reductions were computed between modes (manual → pre-annotations, and pre-annotations → pre-annotations+SAM2) to quantify relative gains.

3.6 Evaluation protocol

This section defines the evaluation procedure used to compare trained models on held-out test splits. The protocol is designed to (i) quantify foreground delineation quality at the pixel level, (ii) derive bubble-by-bubble statistics relevant to multiphase-flow measurements (count and area), and (iii) report indicative inference throughput under a consistent runtime setup. All evaluation runs operate on COCO-style instance segmentation datasets [11] and use the same deterministic preprocessing as training (resize and tensor conversion), with random augmentation disabled.

3.6.1 Evaluation pipeline and metrics

Two complementary evaluation perspectives are used at different stages. During training/validation of Mask R-CNN models, checkpoint selection follows standard COCO instance segmentation evaluation,

using the segmentation average precision AP@[0.50:0.95] [11]. In contrast, the final held-out test evaluation reported in this dissertation uses a dedicated evaluator that computes (a) semantic overlap metrics from the union of predicted instance masks and (b) instance-derived bubble statistics (count and area) from filtered predictions. This test-time protocol is intentionally aligned with downstream measurement objectives, where stable occupancy/delineation and bubble count/size statistics are central.

This design follows the evaluation practices summarized in Section 2.4, combining union-mask overlap scores with instance-derived indicators to reflect measurement impact.

3.6.2 Prediction filtering and derived representations

For Mask R-CNN models, each predicted instance is filtered using the fixed thresholds specified in the YAML `eval` block (Table 3.3): (i) a confidence-score threshold `score_thresh`, (ii) a mask-binarization threshold `mask_thresh` applied to predicted mask probabilities, and (iii) a minimum pixel area `min_area` to remove small spurious detections. These values are set once and kept identical across experiments to avoid per-run tuning and ensure fair comparisons.

After filtering, the evaluator constructs two complementary representations:

- an **instance set** (kept masks and boxes), used for bubble count and area statistics; and
- a **semantic foreground mask**, formed as the pixelwise union of all kept instance masks.

For semantic evaluation, all predicted instance masks are merged (pixelwise union) into a single binary foreground mask; ground-truth instance masks are merged in the same way. Semantic overlap metrics computed on these merged masks quantify overall foreground delineation/occupancy, but they are not sensitive to whether the foreground is correctly partitioned into individual bubbles. Therefore, semantic overlap is reported together with instance-derived indicators (bubble count and area statistics, including ΔN and $|\Delta N|$) that capture split/merge behavior relevant to downstream measurements.

The following subsections describe each metric group in detail.

Pixel-level (semantic) overlap metrics. From the merged prediction mask and merged ground-truth mask, the evaluator computes Precision, Recall, and F1-score on flattened binary masks, as well as Dice coefficient and Intersection over Union (IoU). If both the ground truth and prediction are empty for a frame (no bubbles present and none predicted), all semantic metrics are set to 1.0 to reflect perfect agreement on the absence of bubbles.

Instance-derived bubble statistics. From the filtered instance set, the evaluator reports per-image bubble count (number of kept instances), per-image mean and median bubble area, and corresponding ground-truth values computed from GT masks. Bubble area is defined as the pixel count of the binarized instance mask, and is therefore reported in pixel units. In addition, the evaluator reports counting error (`pred - gt`) and absolute counting error per image, and flags frames where the ground truth is empty and/or the prediction is empty.

Inference-time indicators. Inference timing is measured per image using wall-clock time around the forward pass. When running on GPU, CUDA synchronization is performed immediately before and

after inference to ensure the measured time corresponds to completed GPU execution. The evaluator reports mean inference time per image and frames per second (FPS); these timings are interpreted as indicative and are used primarily for relative comparisons under the same runtime environment.

3.6.3 Outputs and reporting

The evaluator loads ground-truth masks from the test split `_annotations.coco.json` and decodes instance masks using standard COCO utilities [11]. A test dataloader is constructed with `batch_size=1` and `shuffle=False` to enable per-image reporting. The preprocessing pipeline is identical to training preprocessing (resize to the configured resolution, conversion to `float32` with scaling, and tensor conversion), while random augmentation is disabled to ensure deterministic evaluation.

To make qualitative examples reproducible, the evaluator uses a fixed `example_seed` (when provided) to select a consistent subset of test indices for visualization. The evaluation configuration used (thresholds and seed) is stored in the experiment summary output.

For each experiment, the evaluator produces: (i) a per-image CSV file (`test_semantic_metrics.csv`) containing semantic overlap metrics, instance-derived statistics, and empty-frame flags; (ii) a structured JSON summary (`results.json`) containing averaged metrics, timing summaries, and the evaluation configuration used; and (iii) an `examples/` directory with qualitative visualizations (image, GT overlay, predicted mask overlay, and predicted boxes).

After evaluation, summary plots and comparisons are generated from `results.json` and `test_semantic_metr` to support consistent reporting across experiments.

3.7 Implementation and reproducibility

Reproducibility in this work is supported by configuration-driven runs and logging; the YAML experiment structure and traceability mechanisms are described in Sections 3.4.1 and 3.4.5.

3.7.1 Experiment management and artifacts

Runs are defined by the YAML configurations described in Section 3.4 and executed by passing the configuration path to the corresponding pipeline script (e.g., `-config <file.yaml>`). Each run produces a standardized set of artifacts: periodic checkpoints and the best-validation checkpoint (`model_best.pth`), TensorBoard logs for training losses and validation metrics, and exported evaluation summaries and qualitative overlay visualizations.

Two training pipelines are used:

- `synthetic_train_pipeline.py` implements supervised training on the dataset splits defined in the YAML configuration.

- `transfer_train_pipeline.py` implements synthetic-to-real fine-tuning by initializing from a checkpoint (`transfer.init_checkpoint`) and training on real dataset splits specified in the YAML configuration.

Each run creates a time-stamped output directory under the base output folder defined in the YAML configuration (e.g., `outputs/<experiment>_<timestamp>/`). Per-epoch progress is recorded to a CSV metrics log, and the best-performing checkpoint is saved as `model_best.pth`. For Mask R-CNN runs, "best" is defined by the validation criterion used during training, which follows COCO-style instance segmentation evaluation (segmentation AP@[0.50:0.95]) [11]. This validation-time criterion is used for checkpoint selection, while final test reporting follows the separate evaluation protocol described in Section 3.6.

Training curves and scalar metrics are logged with TensorBoard when enabled in the YAML configuration. In addition, configuration text (including the full YAML content) is recorded in the run logs to preserve a complete trace of hyperparameters, thresholds, and dataset paths used for each experiment.

Evaluation outputs (CSV/JSON summaries and qualitative overlays) are produced by the evaluator described in Section 3.6.

3.7.2 Reproducibility controls

A fixed pseudo-random seed is specified per experiment via the YAML file and is applied to Python, NumPy, and PyTorch random number generators at the start of training (including CUDA seeds when available). This reduces run-to-run variance and supports consistent comparisons. However, strict determinism is not enforced by default (e.g., some CUDA kernels may be non-deterministic), so exact bitwise reproducibility is not guaranteed across different hardware or execution settings. In this dissertation, comparisons are made under a consistent software stack and execution environment to mitigate this effect.

3.7.3 Execution environment and software

All experiments were executed on a personal workstation equipped with a GPU NVIDIA GeForce RTX 2060 Max-Q GPU and 16 GB of system RAM. This hardware context motivates several practical configuration choices (batch size, image resolution, and the use of mixed precision) and provides a reference for interpreting training time and inference throughput reported later.

All training and evaluation code is implemented in PyTorch and TorchVision, using TorchVision reference implementations of Mask R-CNN model variants and standard PyTorch components such as optimizers, learning-rate scheduling, and DataLoader abstractions [14].

Chapter 4

Experimental Results and Discussion

4.1 Chapter overview and evaluation scope

This chapter reports quantitative results from the protocol described in Chapter 3. Three TorchVision Mask R-CNN variants are evaluated under four training regimes (12 runs total) on the held-out real experimental test split, using the fixed post-processing thresholds and evaluation protocol of Section 3.6. TensorBoard logs are used only as convergence diagnostics and for checkpoint selection; all comparisons and conclusions reported here are based on the standardized test-set metrics.

The remainder of the chapter is organized as follows. Section 4.2 presents the main comparison table across all 12 experiments. Sections 4.3–4.4 isolate the effect of synthetic-to-real transfer learning and the additional effect of enabling augmentation during real fine-tuning, respectively. Section 4.5 discusses the speed-accuracy trade-off associated with backbone capacity, and Section 4.6 analyzes stratified performance by scene difficulty (e.g., bubble density and scale). Section 4.7 provides qualitative error analysis using representative overlay examples, linking typical failure modes to the metric design and measurement objectives. Finally, Section 4.8 reports the annotation efficiency results for the model-assisted labeling workflows, and Section 4.9 summarizes the key takeaways of the chapter.

4.2 Summary of results and main comparison table

Table 4.1 summarizes the principal comparison across the 12 experiments (3 backbones $\times$ 4 training regimes) on the held-out real experimental test split (25 images). Finally, the test split contains no empty frames (num_empty_gt=num_empty_pred=0 for all runs), so overlap metrics are not affected by the evaluator's empty-frame convention. Run names follow the pattern `maskrcnn_<backbone>_<regime>_<timestam` (e.g., `..._real`, `..._synth`, `..._transfer_real`, `..._transfer_real_aug`).

Several consistent trends emerge from the main comparison. First, training directly on real data (Real baseline) yields high union-mask overlap across all backbones (Dice $\approx$ 0.92–0.96), confirming that Mask R-CNN can delineate bubbly foreground reliably under the controlled split and thresholds.

Table 4.1: Main comparison of the 12 experiments on the held-out real experimental test split (25 images). Dice/IoU and pixel-level Precision/Recall are computed on the union of predicted instance masks versus the union of ground-truth masks. $|\Delta N|$ denotes the mean absolute bubble-count error and ΔN the signed mean count error (Pred–GT). FPS denotes inference throughput. Arrows indicate whether higher (↑) or lower (↓) values are better.

Backbone	Regime	Dice ↑	IoU ↑	Prec. ↑	Rec. ↑	$\lvert\Delta N\rvert$ ↓	ΔN	FPS ↑
maskrcnn_resnet50_fpn	Synthetic	0.892	0.824	0.891	0.902	2.48	2.24	5.23
maskrcnn_resnet50_fpn	Real	0.953	0.914	0.929	0.981	0.60	0.60	6.52
maskrcnn_resnet50_fpn	Transfer	0.954	0.916	0.927	0.986	0.44	0.44	**7.01**
maskrcnn_resnet50_fpn	Transfer+Aug	0.951	0.912	0.928	0.979	0.88	0.80	6.88
maskrcnn_resnet50_fpn_v2	Synthetic	0.740	0.657	0.702	0.906	2.48	1.28	4.27
maskrcnn_resnet50_fpn_v2	Real	0.960	0.925	0.950	0.971	0.48	0.32	5.45
maskrcnn_resnet50_fpn_v2	Transfer	**0.963**	**0.930**	0.950	0.977	**0.40**	0.16	6.05
maskrcnn_resnet50_fpn_v2	Transfer+Aug	0.962	0.928	0.950	0.974	0.48	0.24	5.19
maskrcnn_resnet101_fpn	Synthetic	0.629	0.509	0.887	0.535	9.84	-7.12	4.41
maskrcnn_resnet101_fpn	Real	0.921	0.866	0.911	0.944	3.52	3.52	3.42
maskrcnn_resnet101_fpn	Transfer	0.955	0.915	0.935	0.977	0.96	0.80	5.58
maskrcnn_resnet101_fpn	Transfer+Aug	0.941	0.893	0.945	0.940	0.96	0.96	5.69

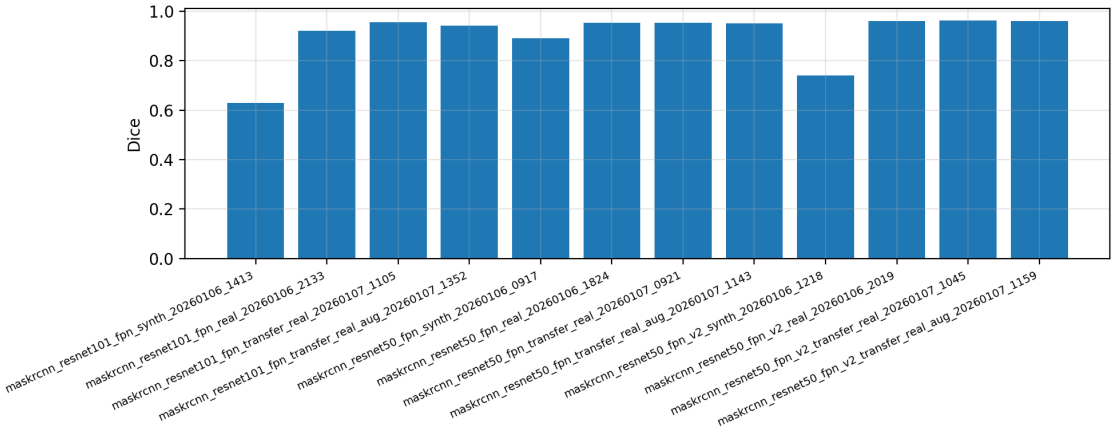
Second, synthetic-only models underperform on real test images due to the domain gap between synthetic renderings and real experimental appearance; this effect depends on the backbone: while ResNet-50-FPN achieves relatively strong union-mask overlap (Dice=0.892), it still shows larger counting errors ($|\Delta N| = 2.48$) than real-trained models, and the ResNet-101-FPN synthetic run performs substantially worse both in overlap (Dice=0.629) and counting stability ($|\Delta N| = 9.84$), indicating limited transferability from purely synthetic appearance in the most challenging setting.

Third, synthetic-to-real transfer learning improves measurement stability most strongly. For example, ResNet-101-FPN improves from Dice=0.921 and $|\Delta N| = 3.52$ (Real) to Dice=0.955 and $|\Delta N| = 0.96$ (Transfer), reducing systematic count bias while also increasing overlap. The best overall result in this grid is obtained with the updated ResNet-50-FPNv2 backbone under transfer learning, which achieves the highest mean Dice (0.963) and IoU (0.930) together with the lowest mean absolute count error ($|\Delta N| = 0.40$). Inference-speed implications are analyzed separately in Section 4.5 [14].

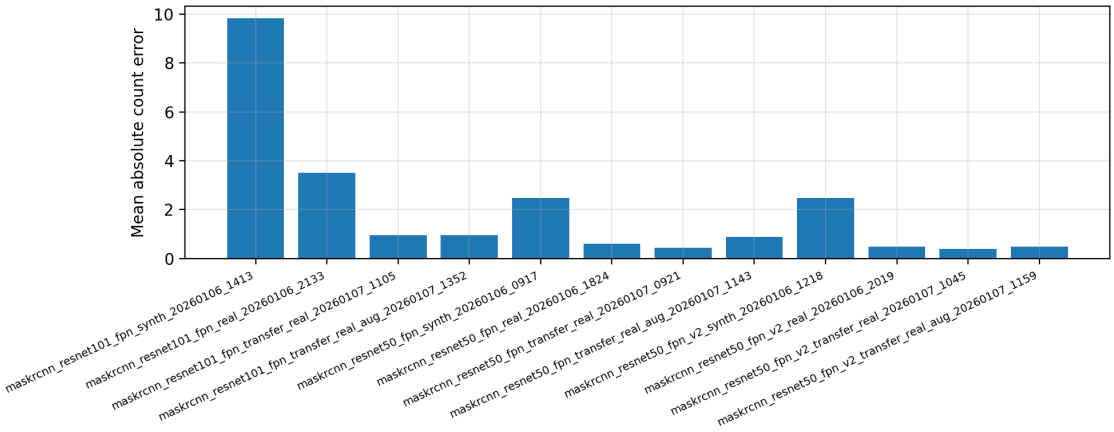
For readability, the full 12-experiment bar-plot comparisons (Dice, $|\Delta N|$, FPS) are used in this section as an overview of the grid. Subsequent sections present focused paired comparisons to isolate the effect of transfer learning (Real vs Transfer), augmentation (Transfer vs Transfer+Aug), and backbone capacity (R50 vs R50v2 vs R101) under the best-performing regime. Figure 4.1 provides a compact overview of accuracy, counting stability, and throughput across the full grid.

4.3 Effect of synthetic-to-real transfer learning

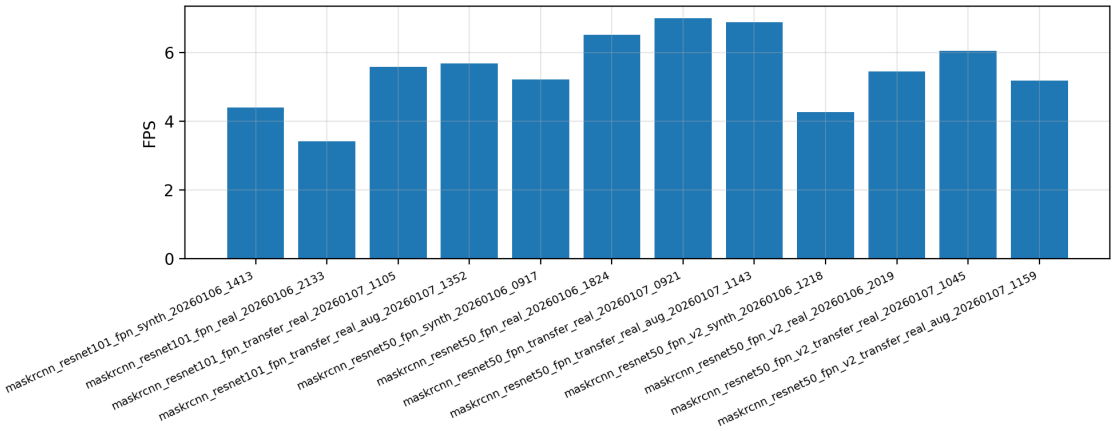
This section isolates the effect of synthetic-to-real transfer learning by comparing, for each backbone, a *Real* baseline against the corresponding *Transfer* configuration initialized from the synthetic-trained



(a) Mean Dice (union-mask).



(b) Mean absolute count error $|\Delta N|$.



(c) Inference throughput (FPS).

Figure 4.1: Overview comparison across the 12 experiments (3 backbones $\times$ 4 regimes). All metrics are computed on the same real test split under fixed evaluation thresholds.

checkpoint and fine-tuned on the real training set (Chapter 3) [13].

Test-set impact: overlap and measurement stability. Figures 4.2, 4.3, and 4.4 compare, for each backbone, the *Real* and *Transfer* regimes on the held-out real test split, reporting union-mask overlap (Dice) and measurement stability via counting error ($|\Delta N|$ and ΔN). Table 4.2 reports the corresponding numerical deltas.

For the ResNet-50 backbones, transfer learning yields small but consistent gains in union-mask overlap and reduces counting errors: `resnet50_fpn` improves Dice from 0.9530 to 0.9544 while reducing $|\Delta N|$ from 0.60 to 0.44 bubbles/image; `resnet50_fpn_v2` improves Dice from 0.9602 to 0.9628 and reduces $|\Delta N|$ from 0.48 to 0.40. The modest Dice/IoU changes are consistent with a ceiling effect: when foreground overlap is already high, improvements from transfer are expressed more clearly in instance-level indicators (reduced count error) than in union-mask overlap.

The strongest transfer effect is observed for `resnet101_fpn`, where Dice increases from 0.9207 to 0.9552 and $|\Delta N|$ decreases from 3.52 to 0.96. This suggests that, for the higher-capacity backbone, training from scratch on the available real labels can lead to poor calibration and unstable instance predictions (here, systematic over-detection), whereas synthetic pretraining provides a substantially better initialization that fine-tunes to the real domain more reliably.

To interpret systematic over/under-counting, the signed mean count error is defined as

$$\Delta N = E[N_{\text{pred}} - N_{\text{gt}}], \tag{4.1}$$

where $\Delta N > 0$ indicates systematic over-counting and $\Delta N < 0$ indicates under-counting. Transfer learning moves ΔN closer to zero for all backbones, most notably for `resnet101_fpn`, where the real-only baseline shows an average overcount of $+3.52$ and transfer reduces this bias to $+0.80$. Because union-mask overlap does not fully penalize instance split/merge errors, reductions in both $|\Delta N|$ and ΔN are particularly relevant for downstream measurement objectives such as bubble number density and size statistics [2, 6, 24].

Table 4.2: Real→Transfer deltas on the real test split (Transfer − Real). Positive deltas in Dice/IoU indicate improved overlap; negative deltas in $|\Delta N|$ indicate improved counting stability.

Backbone	ΔDice	ΔIoU	$\Delta\lvert\Delta N\rvert$	$\Delta(\Delta N)$
resnet50_fpn	+0.0014	+0.0024	−0.16	−0.16
resnet50_fpn_v2	+0.0026	+0.0046	−0.08	−0.16
resnet101_fpn	+0.0344	+0.0491	−2.56	−2.72

Training dynamics: validation mAP evidence (TensorBoard). The training pipeline logs COCO-style validation segmentation performance each epoch using mAP (AP@[0.50:0.95]) [11]. Figures 4.5–4.7 show the TensorBoard validation curves for Real vs. Transfer runs. For `resnet50_fpn` and `resnet50_fpn_v2`, transfer learning reaches a slightly higher validation plateau; for `resnet101_fpn`,

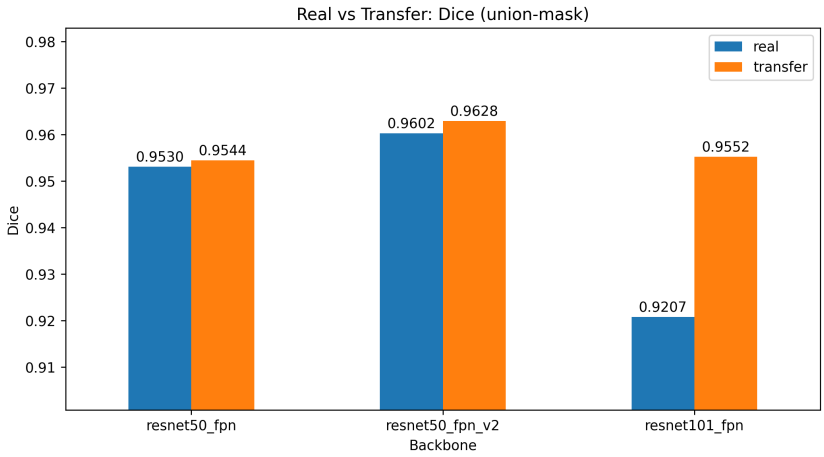


Figure 4.2: Effect of synthetic-to-real transfer learning on the real test split (Real vs. Transfer): mean Dice (union-mask) across backbones.

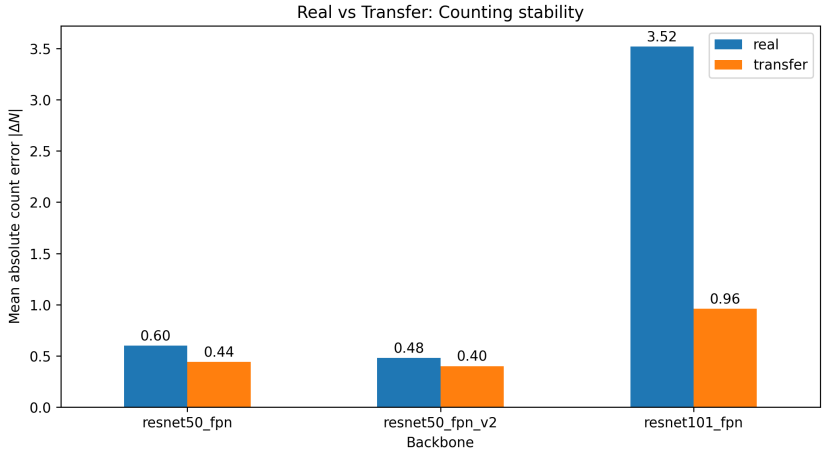


Figure 4.3: Effect of synthetic-to-real transfer learning on the real test split (Real vs. Transfer): mean absolute count error $|\Delta N|$ across backbones.

transfer learning produces a markedly higher validation trajectory and final plateau. The best validation mAP values extracted from the logged training CSVs (Table 4.3) corroborate this observation: transfer increases best validation mAP from 0.8871 to 0.8979 (`resnet50_fpn`), from 0.8867 to 0.8955 (`resnet50_fpn_v2`), and from 0.8298 to 0.8657 (`resnet101_fpn`). Overall, the TensorBoard results support the test-set findings: synthetic initialization improves optimization and generalization during fine-tuning, with the largest benefit for the higher-capacity backbone [13].

TensorBoard scalar plots show both the raw logged values (faint curve) and the TensorBoard-smoothed trend (prominent curve); smoothing is used only for visualization and does not affect training. The run summary table below each plot reports the raw scalar value at the selected epoch (Step) together with the corresponding smoothed value and relative wall-clock time.

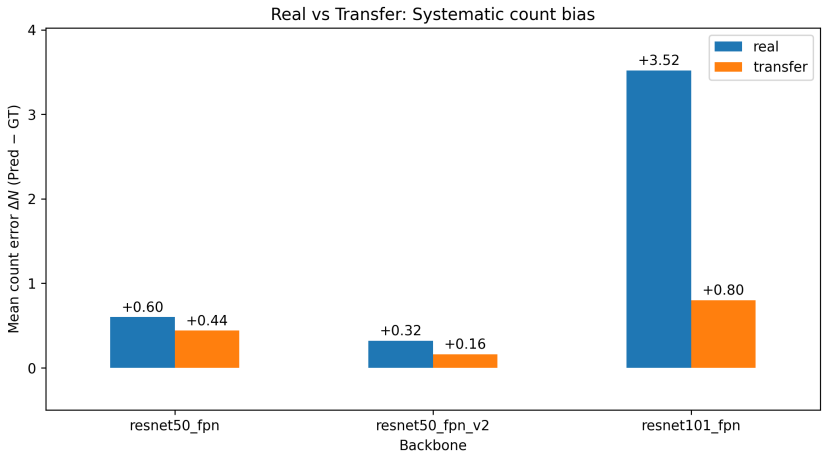


Figure 4.4: Effect of synthetic-to-real transfer learning on the real test split (Real vs. Transfer): mean signed count error ΔN (Pred $-$ GT) across backbones.

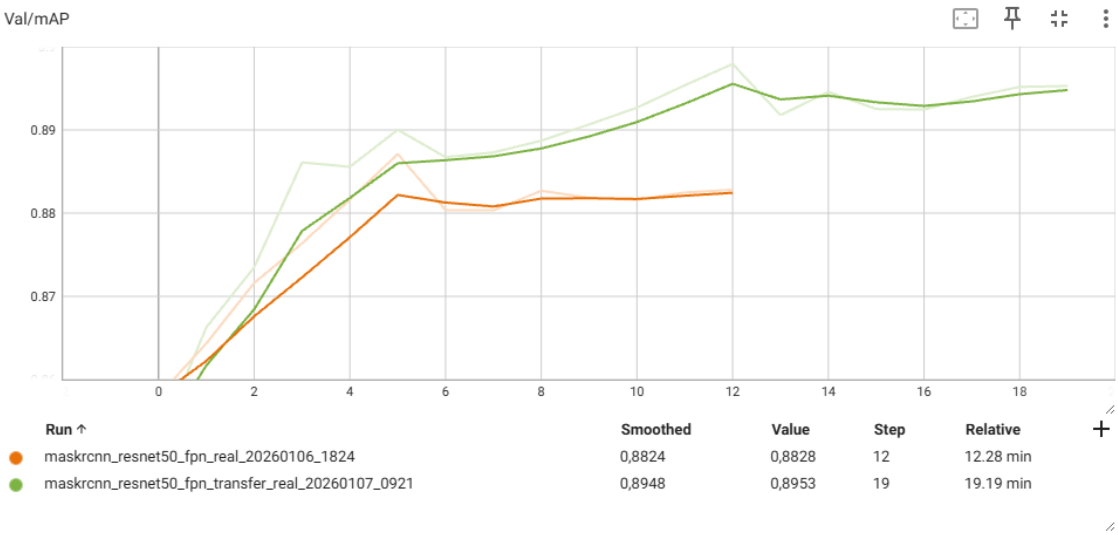


Figure 4.5: TensorBoard validation mAP (AP@[0.50:0.95]) versus epoch for `resnet50_fpn`: Real vs. Transfer training.

Training dynamics: Mask R-CNN loss decomposition (TensorBoard). In addition to validation mAP, the training pipeline logs the Mask R-CNN loss terms described in Section 3.4.5.1. These curves provide diagnostic insight into optimization behavior. The total loss is the sum of the RPN losses (objectness and proposal box regression), the detection losses (classification and box regression), and the instance mask loss. Total training loss is reported together with two representative components: $\mathscr{L}_{\text{mask}}$ (mask quality within positive RoIs) and $\mathscr{L}_{\text{rpn\_cls}}$ / `loss_objectness` (proposal objectness), which is often sensitive to dense overlap.

Figures 4.8–4.11 show the loss trajectories across the four regimes. Note that the *synthetic-only*

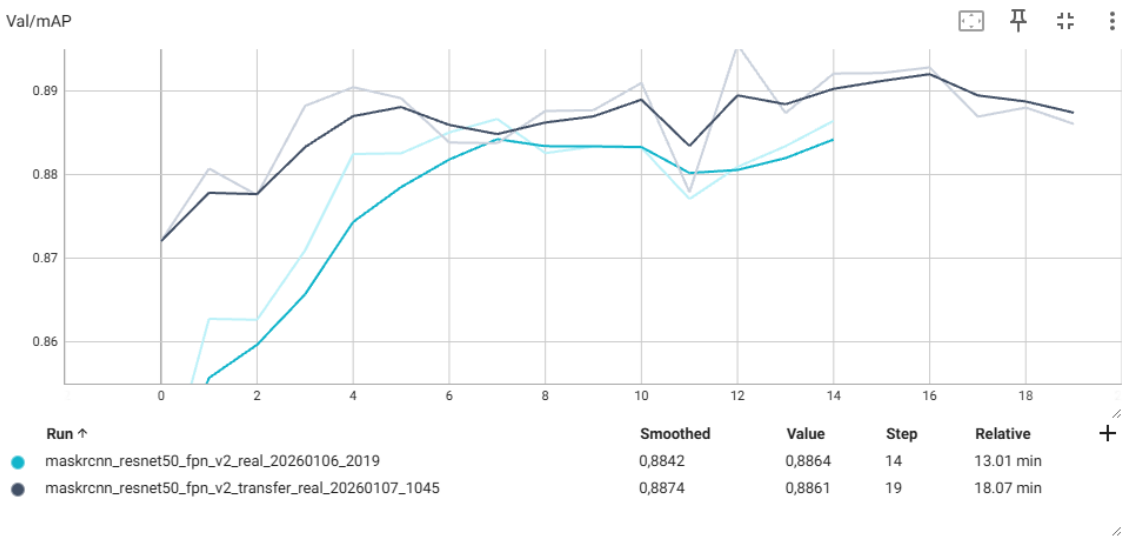


Figure 4.6: TensorBoard validation mAP (AP@[0.50:0.95]) versus epoch for `resnet50_fpn_v2`: Real vs. Transfer training.

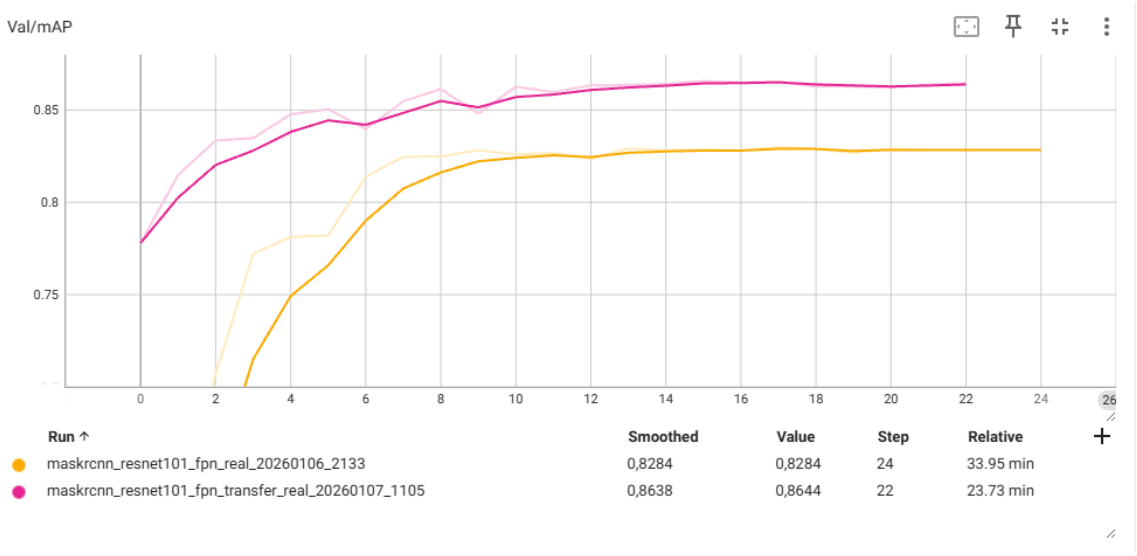


Figure 4.7: TensorBoard validation mAP (AP@[0.50:0.95]) versus epoch for `resnet101_fpn`: Real vs. Transfer training.

regime is trained on synthetic data, whereas the remaining regimes are trained on real data; therefore, absolute loss magnitudes are not directly comparable across domains and are used here primarily to assess convergence and optimization stability. Final comparisons rely on the standardized test-set evaluation metrics reported in Section 4.2 (Table 4.1). Within the real-data regimes, transfer learning shows systematically lower and faster-stabilizing losses than training from scratch, particularly in `loss_objectness`, consistent with improved proposal quality early in training. In contrast, trans-

Table 4.3: Best validation segmentation mAP (AP@[0.50:0.95]) recorded during training (from the per-epoch metrics CSV exported by the training pipeline).

Backbone	Real best mAP (epoch)	Transfer best mAP (epoch)	Δ best mAP
resnet50_fpn	0.8871 (5)	0.8979 (12)	+0.0108
resnet50_fpn_v2	0.8867 (7)	0.8955 (12)	+0.0089
resnet101_fpn	0.8298 (17)	0.8657 (15)	+0.0358

fer with augmentation converges more slowly and plateaus at higher loss in these runs, matching the absence of average gains observed in the quantitative metrics.

4.4 Effect of augmentation during transfer learning

This section tests whether enabling data augmentation during the real fine-tuning stage changes performance in the synthetic-to-real transfer regime. For each backbone, *Transfer* is compared against *Transfer+Aug* under the same split and evaluation settings [18].

Test-set impact: mean overlap and counting indicators. Figures 4.12–4.14 summarize the effect of augmentation on the held-out real test split. In contrast to transfer learning alone (Section 4.3), augmentation does not improve the mean semantic overlap metrics in this dataset. For `resnet50_fpn`, Dice decreases slightly (0.9544→0.9514) and counting stability worsens, with mean absolute count error increasing (0.44→0.88 bubbles/image) and the signed bias increasing (ΔN: 0.44→0.80). The same trend is observed for `resnet50_fpn_v2`, where Dice decreases marginally (0.9628→0.9616) and $|\Delta N|$ increases (0.40→0.48). For `resnet101_fpn`, Dice decreases more noticeably (0.9552→0.9413), while the mean absolute count error remains unchanged (0.96→0.96) and the signed count bias increases slightly (0.80→0.96). Overall, on this test split, augmentation does not yield a mean-performance gain and may introduce a small degradation in overlap and/or instance counting.

Table 4.4: Transfer→Transfer+Aug changes on the real test split (Transfer+Aug − Transfer). Negative deltas in Dice/IoU indicate reduced overlap; positive deltas in $|\Delta N|$ indicate worse counting stability.

Backbone	ΔDice	ΔIoU	$\Delta\lvert\Delta N\rvert$	$\Delta(\Delta N)$
resnet50_fpn	−0.0030	−0.0047	+0.44	+0.36
resnet50_fpn_v2	−0.0012	−0.0022	+0.08	+0.08
resnet101_fpn	−0.0139	−0.0216	+0.00	+0.16

Robustness and error tails. Mean values can hide whether augmentation helps in the most difficult frames. To complement the mean metrics, the empirical cumulative distribution function (ECDF) of the per-image absolute count error $|\Delta N|$ is plotted. This curve reports, for each threshold t, the fraction of test images with $|\Delta N| \leq t$ and highlights tail behavior (rare large errors) [1]. To assess robustness, the distribu-

[1] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ecdf.html

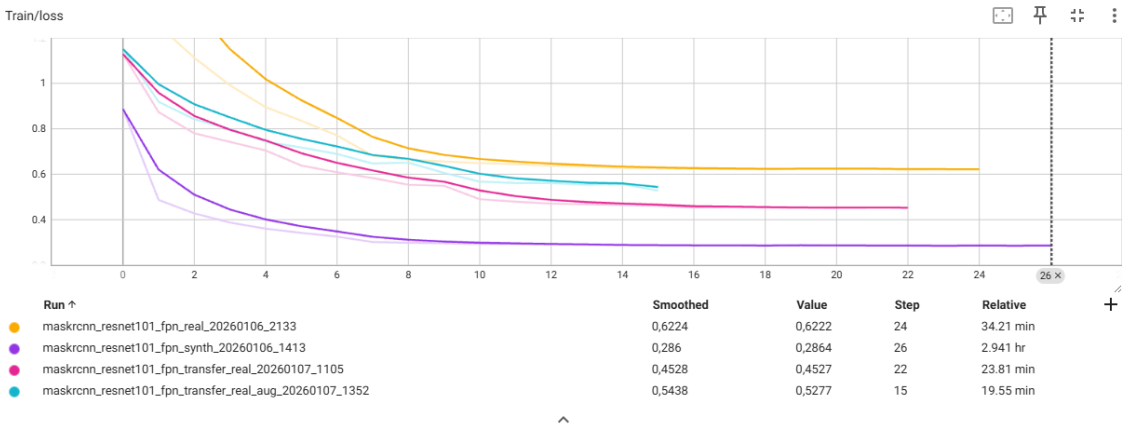


Figure 4.8: TensorBoard total training loss for Mask R-CNN with ResNet-101-FPN under the four training regimes. The synthetic-only regime is trained on synthetic data; the remaining regimes are trained on real data, so loss values are used primarily as convergence diagnostics rather than as direct cross-domain comparisons.

tions of $|\Delta N|$ for Transfer and Transfer+Aug are compared using the ECDF curves in Figures 4.15–4.17. For `resnet101_fpn`, augmentation reduces extreme errors even though the mean $|\Delta N|$ is unchanged: the upper-tail of the distribution contracts (e.g., the near-maximum observed error decreases from approximately 4.8 to 3.0 bubbles/image), and the fraction of frames with $|\Delta N| \geq 2$ decreases (0.28→0.24). In contrast, for the ResNet-50 backbones the tail behavior worsens: the fraction of frames with $|\Delta N| \geq 2$ increases from 0.08 to 0.20 (`resnet50_fpn`) and from 0.08 to 0.12 (`resnet50_fpn_v2`). Given the relatively small test set (25 images), these tail statistics should be interpreted as indicative rather than definitive, but they suggest that the current augmentation setup is not uniformly beneficial and may be more helpful for the higher-capacity model in the hardest cases [18].

Interpretation and limitations. The mixed or negative effect of augmentation in this study should be interpreted in the context of dataset scale and distribution. The real dataset available for fine-tuning is limited, and the held-out test split contains only 25 images. With such a small test set, performance estimates (both means and tail statistics) have high variance, and a small number of challenging frames can disproportionately influence the measured impact of augmentation. Moreover, augmentation improves robustness primarily when the applied transformations approximate plausible variability of the target acquisition conditions. If the augmentation policy is even partially mismatched to the real imaging process (e.g., overly strong photometric perturbations or geometric transformations that do not occur in the experimental setup), it can act as a form of distribution shift during training and degrade performance under the fixed test protocol. These observations suggest that (i) augmentation policy selection should be guided by knowledge of the imaging process and validated on a sufficiently large validation/test set, and (ii) the potential robustness benefits of augmentation are more reliably assessed when the evaluation split covers a broader range of operating conditions [18].

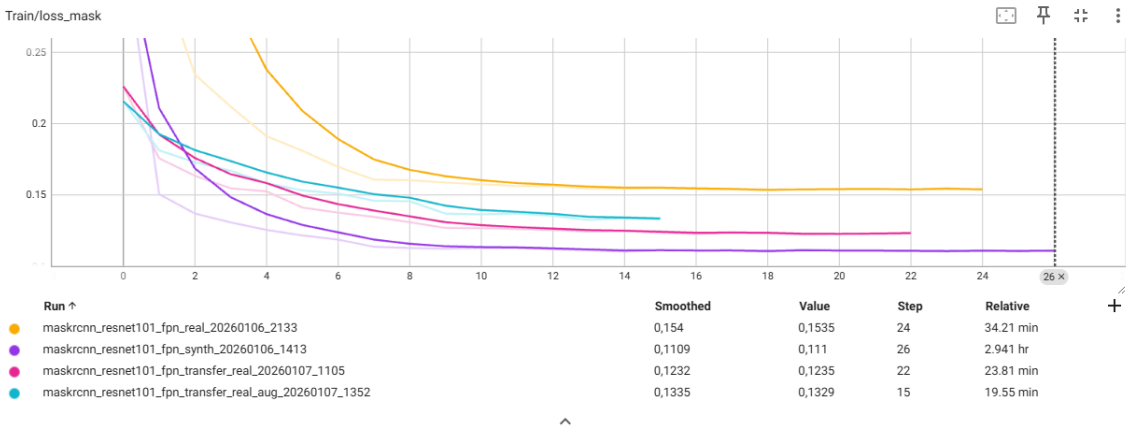


Figure 4.9: TensorBoard `loss_mask` for Mask R-CNN with ResNet-101-FPN under the four training regimes. The synthetic-only regime is trained on synthetic data; the remaining regimes are trained on real data, so loss values are used primarily as convergence diagnostics rather than as direct cross-domain comparisons.

4.5 Backbone capacity vs inference speed

This section analyzes the trade-off between backbone capacity and inference throughput for the Mask R-CNN architectures evaluated in this work. While higher-capacity backbones can improve representation power, they typically increase computational cost (deeper feature extraction and higher memory bandwidth demands), which can reduce inference speed. This trade-off is particularly relevant for practical use cases where bubble segmentation is a preprocessing step for measurement and the processing rate may be constrained by hardware [5, 16, 10].

The analysis focuses on the best-performing training regime identified above, namely synthetic-to-real transfer learning without additional augmentation. Inference timing is measured under the same runtime environment (NVIDIA RTX 2060 Max-Q 6GB, Intel Core i5-10300H @ 2.50 GHz), and the reported FPS should be interpreted as indicative throughput for the implemented pipeline (including model forward pass and post-processing) under these conditions.

Table 4.5: Backbone capacity versus inference throughput in the transfer regime (real test).

Backbone	**Dice**	**IoU**	$\lvert\Delta N\rvert$	ΔN	**FPS**
resnet50_fpn	0.9544	0.9164	0.44	+0.44	7.01
resnet50_fpn_v2	0.9628	0.9298	0.40	+0.16	6.05
resnet101_fpn	0.9552	0.9151	0.96	+0.80	5.58

Table 4.5 shows that `resnet50_fpn` is the fastest configuration, achieving 7.01 FPS, while `resnet101_fpn` is the slowest at 5.58 FPS. The updated `resnet50_fpn_v2` backbone provides an intermediate throughput of 6.05 FPS, corresponding to an approximately 14% slowdown relative to `resnet50_fpn`. In

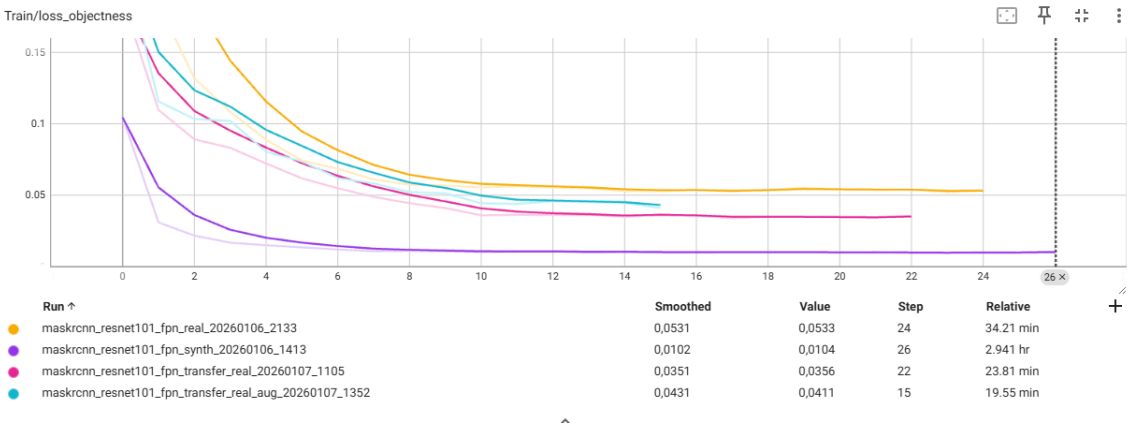


Figure 4.10: TensorBoard `loss_objectness` for Mask R-CNN with ResNet-101-FPN under the four training regimes. The synthetic-only regime is trained on synthetic data; the remaining regimes are trained on real data, so loss values are used primarily as convergence diagnostics rather than as direct cross-domain comparisons.

terms of accuracy and measurement stability, `resnet50_fpn_v2` provides the best overall test-set performance among the evaluated backbones in this regime (Dice=0.9628, $|\Delta N|$=0.40), while `resnet101_fpn` does not yield a corresponding improvement despite its higher capacity (Dice=0.9552, $|\Delta N|$=0.96). These results suggest that, for the available training set size and imaging conditions, increasing backbone depth from ResNet-50 to ResNet-101 increases computational cost without translating into better test-set performance.

Figures 4.18–4.20 visualize this trade-off. Figure 4.18 summarizes the throughput differences across backbones, while Figure 4.19 shows that `resnet50_fpn_v2` achieves the highest overlap metric with only a modest speed penalty. The speed–accuracy scatter in Figure 4.20 highlights `resnet50_fpn_v2` as the best compromise point for this dataset, whereas `resnet50_fpn` remains preferable if maximum throughput is the dominant constraint.

4.6 Stratified performance by scene difficulty

Although strongly overlapping bubbles are limited in the available real dataset, scene difficulty in multiphase-flow imagery is also driven by bubble density, bubble scale, and crowding effects that impact instance separation and counting stability. To analyze where errors concentrate, performance is stratified on the real test split using two ground-truth difficulty proxies computed per image: (i) bubble density, measured by GT_NumBubbles, and (ii) characteristic bubble scale, measured by GT_MeanArea. These quantities are available in the per-image evaluation logs and are independent of the prediction model.

Three density bins are defined from the test-set distribution (25 images): Low (1–8 bubbles), Mid (9–16), and High (17–48). Similarly, size bins are defined as Small, Medium, and Large based on the

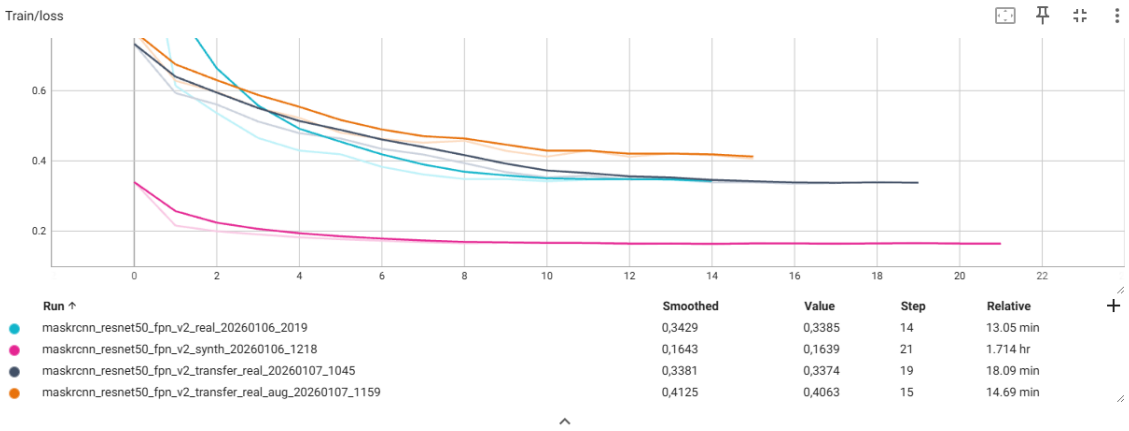


Figure 4.11: TensorBoard total training loss for Mask R-CNN with ResNet-50-FPN v2 under the four training regimes (used as a convergence/stability diagnostic).

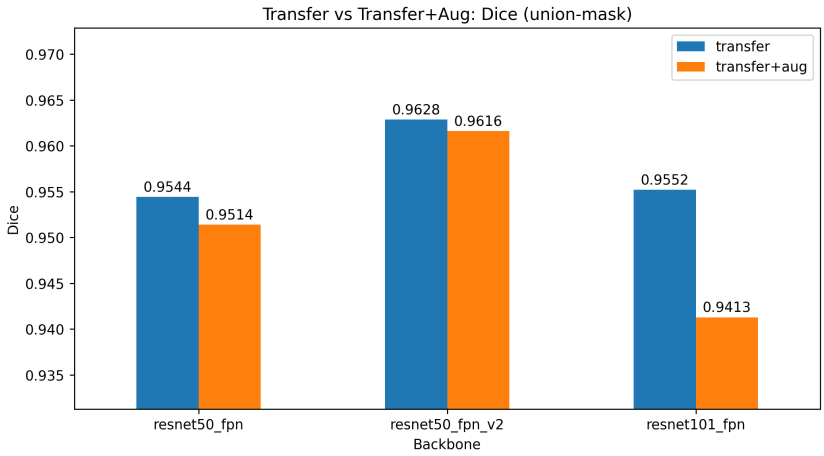


Figure 4.12: Effect of augmentation during transfer learning (Transfer vs. Transfer+Aug): mean Dice (union-mask) across backbones.

GT_MeanArea quartiles. All stratified results in this section correspond to the synthetic-to-real transfer regime, as it provided the most consistent overall accuracy in previous comparisons.

Stratification by bubble density. Table 4.6 and Figures 4.21–4.23 show performance as a function of GT bubble density. For overlap quality, Dice remains high across bins for all backbones, with the highest Dice values generally observed in the high-density bin. Counting behavior, however, varies more noticeably by backbone: the ResNet-101 backbone shows substantially larger $|\Delta N|$ and positive ΔN in the high-density bin, indicating persistent over-counting in crowded scenes even when union-mask overlap is strong. In contrast, the ResNet-50 backbones exhibit lower mean absolute count error in the mid/high bins, suggesting more stable instance separation under the fixed post-processing thresholds.

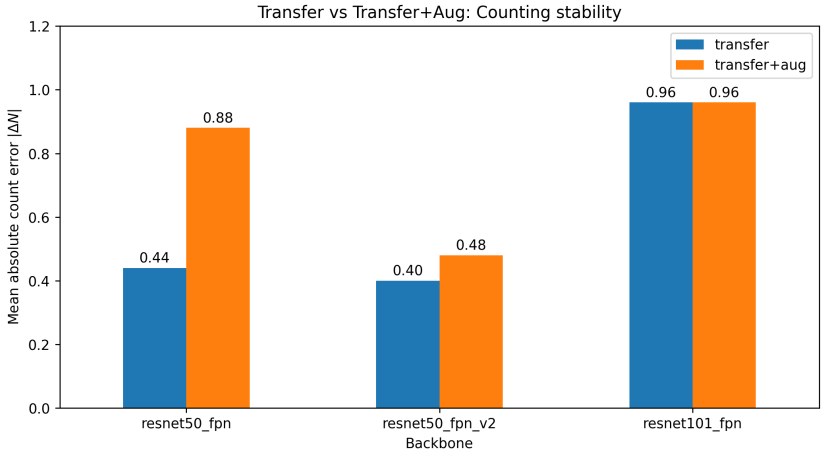


Figure 4.13: Effect of augmentation during transfer learning (Transfer vs. Transfer+Aug): mean absolute count error $|\Delta N|$ across backbones.

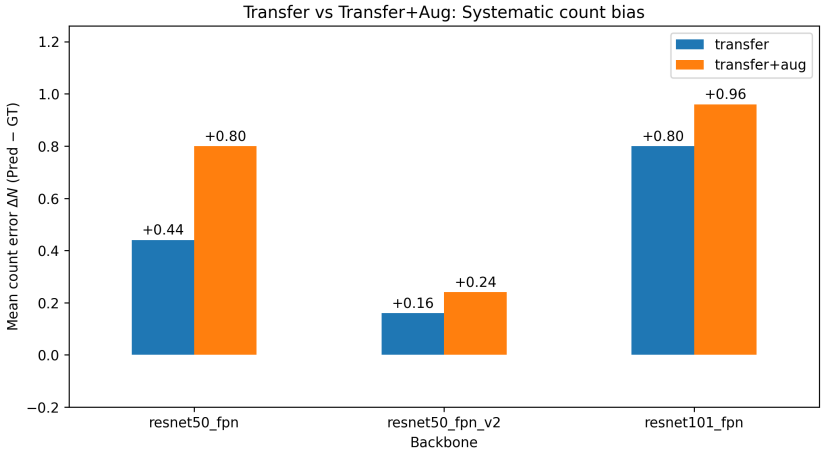


Figure 4.14: Effect of augmentation during transfer learning (Transfer vs. Transfer+Aug): mean signed count error ΔN (Pred $-$ GT) across backbones.

Stratification by bubble scale. Table 4.7 and Figures 4.24–4.25 show performance as a function of GT mean bubble area. A consistent pattern is that small-bubble frames tend to have lower Dice than medium/large frames for some backbones, which is expected since small instances occupy fewer pixels and are more sensitive to thresholding and boundary errors. Counting stability also degrades for large-bubble frames in the ResNet-101 backbone, reflected by an increase in $|\Delta N|$ and ΔN, suggesting that the model tends to generate extra instances in these scenes (over-segmentation) under the fixed filtering parameters. Overall, the stratified analysis supports the earlier conclusion that union-mask overlap alone can appear strong even when instance-level measurement indicators degrade, reinforcing the need to report both overlap and instance-derived statistics.

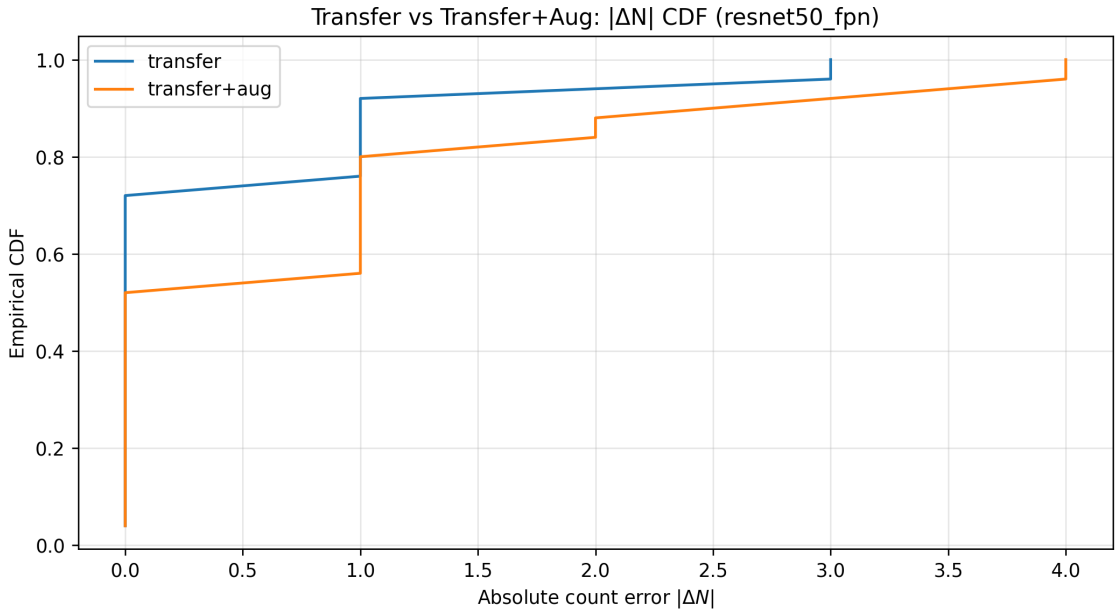


Figure 4.15: Empirical CDF of absolute count error $|\Delta N|$ for `resnet50_fpn` (Transfer vs. Transfer+Aug).

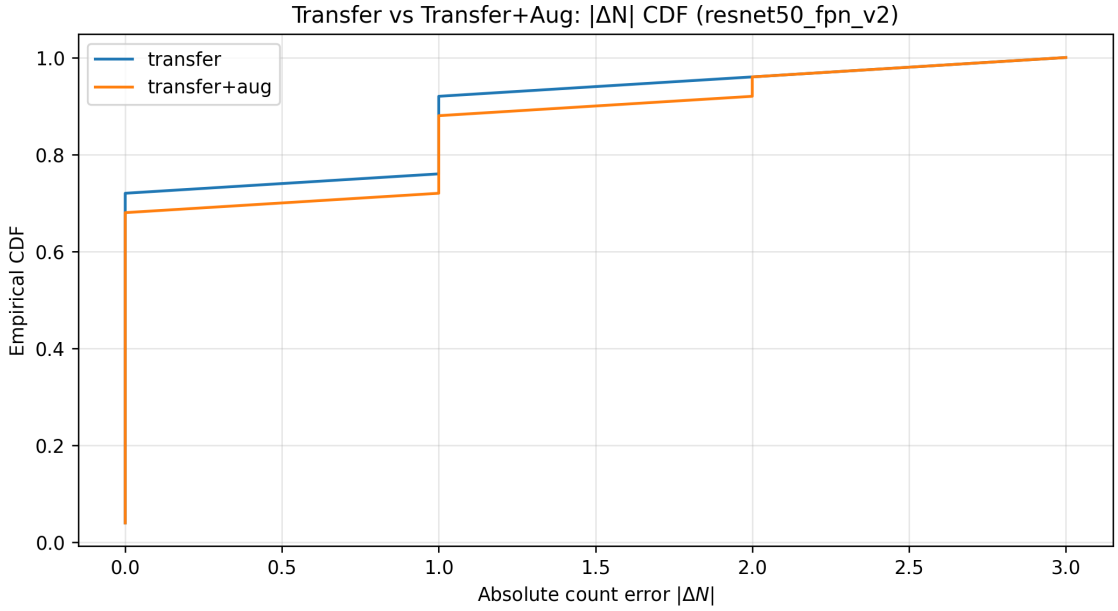


Figure 4.16: Empirical CDF of absolute count error $|\Delta N|$ for `resnet50_fpn_v2` (Transfer vs. Transfer+Aug).

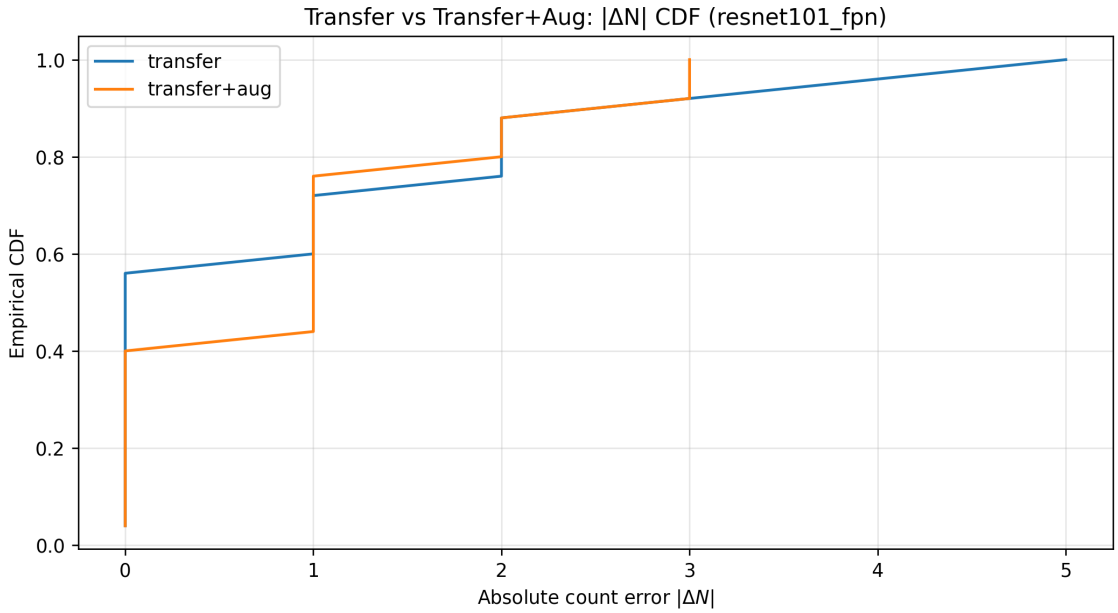


Figure 4.17: Empirical CDF of absolute count error $|\Delta N|$ for `resnet101_fpn` (Transfer vs. Transfer+Aug).

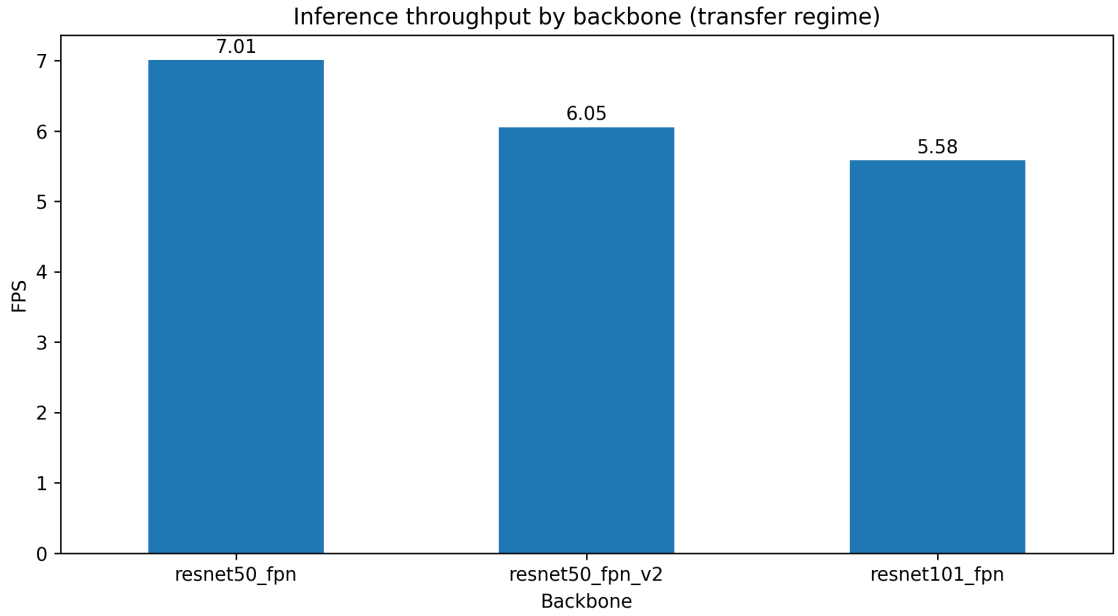


Figure 4.18: Inference throughput (FPS) by backbone in the transfer regime on the real test split (same hardware and evaluation pipeline).

4.7 Qualitative error analysis

Quantitative metrics summarize average behavior but do not always reveal the visual causes of failure. This section complements the numerical comparisons by inspecting representative frames from the real

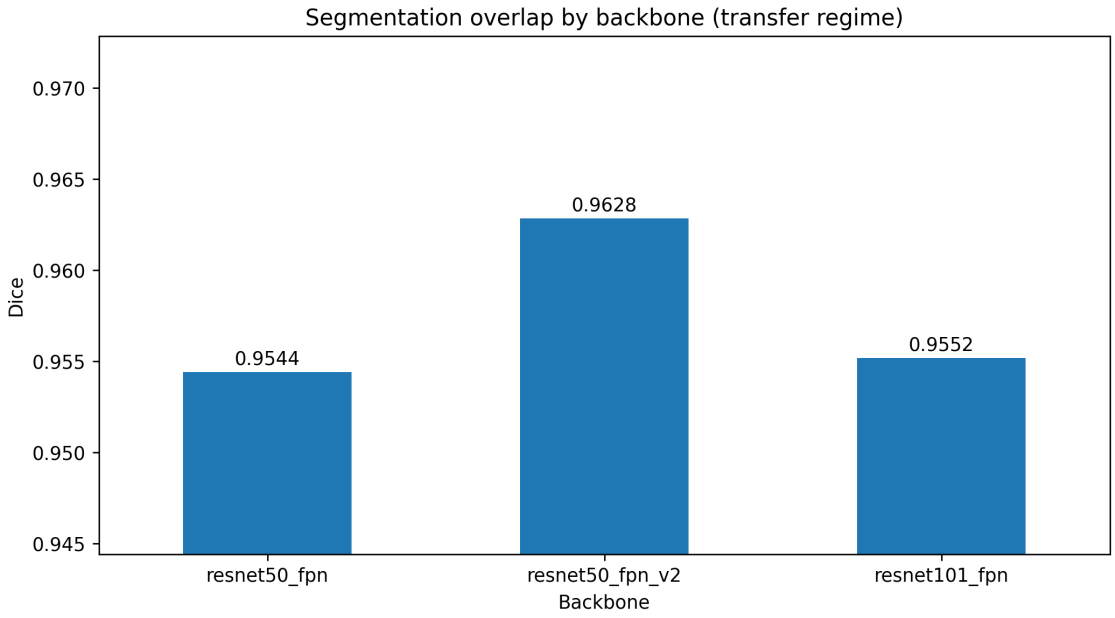


Figure 4.19: Mean Dice (union-mask) by backbone in the transfer regime on the real test split.

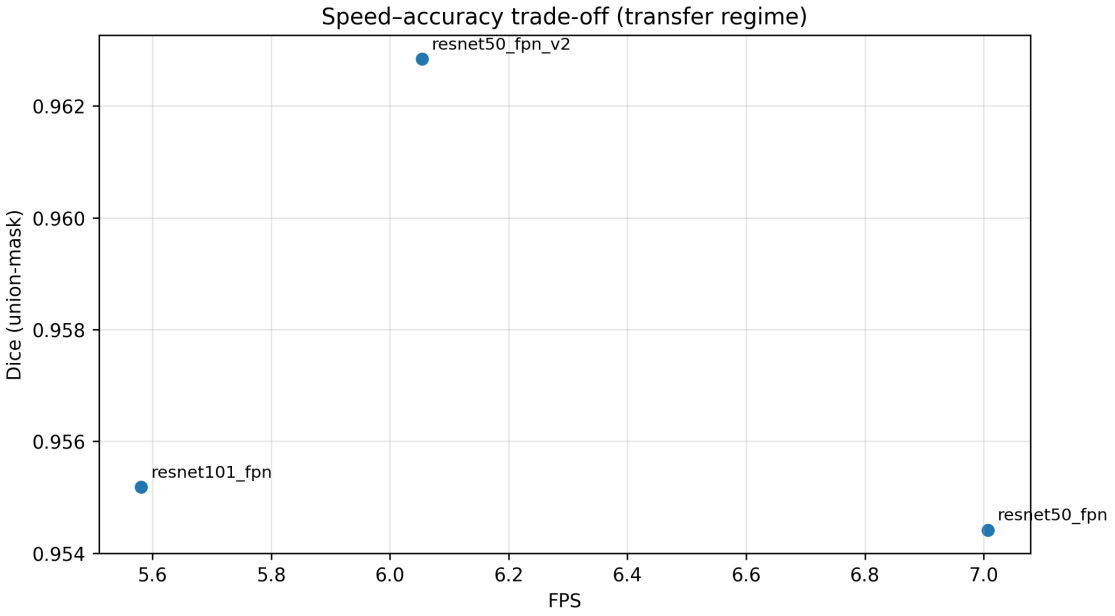


Figure 4.20: Speed–accuracy trade-off in the transfer regime (Dice versus FPS).

test split that expose recurring error modes relevant to measurement. In particular, union-mask overlap (Dice/IoU) can remain high even when instance-level behavior degrades (missed bubbles, spurious detections, or split/merge effects), so qualitative inspection is used to contextualize the count-based indicators $|\Delta N|$ and ΔN reported in the previous sections.

Table 4.6: Stratified performance in the transfer regime by bubble density (bins defined by GT_NumBubbles quartiles on the 25-image real test split).

Bin	n	**R50 Dice**	**R50** $\lvert\Delta N\rvert$	**R50v2 Dice**	**R50v2** $\lvert\Delta N\rvert$	**R101 Dice**	**R101** $\lvert\Delta N\rvert$
Low (1–8)	7	0.953	1.00	0.950	0.43	0.936	1.00
Mid (9–16)	6	0.922	0.17	0.963	0.33	0.958	0.50
High (17–48)	12	0.971	0.25	0.970	0.42	0.965	1.17

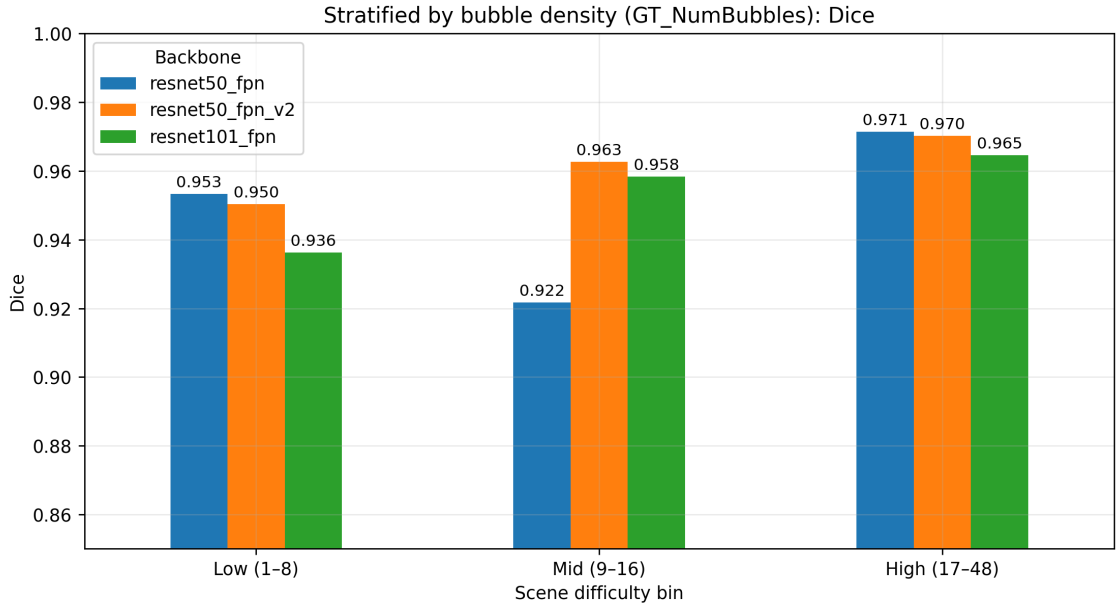


Figure 4.21: Stratified Dice (union-mask) in the transfer regime by bubble density (GT_NumBubbles bins).

Each example figure shows (left-to-right) the raw image, the ground-truth union mask overlay, and the predicted instance masks with bounding boxes. The selected cases are indexed by the test-set image order (ImageIdx) and were chosen to illustrate: (i) a typical success case with a backbone-dependent counting failure, (ii) false positives induced by bubble-like artifacts, (iii) dark and dense conditions that tend to increase overprediction, and (iv) a small/irregular-bubble scene that is handled well by the best-performing regime.

Case A: strong segmentation with backbone-dependent overcounting (ImageIdx=12). ImageIdx=12 is a representative frame where `resnet50_fpn_v2` produces visually consistent results in both the real-only and transfer regimes. However, `resnet101_fpn` in the transfer regime exhibits overcounting, consistent with the larger positive ΔN observed for the higher-capacity backbone in the quantitative results. This example illustrates an important point for downstream measurements: strong overlap can coexist with unstable instance counts, and increasing backbone capacity does not necessarily improve counting reliability for this dataset.

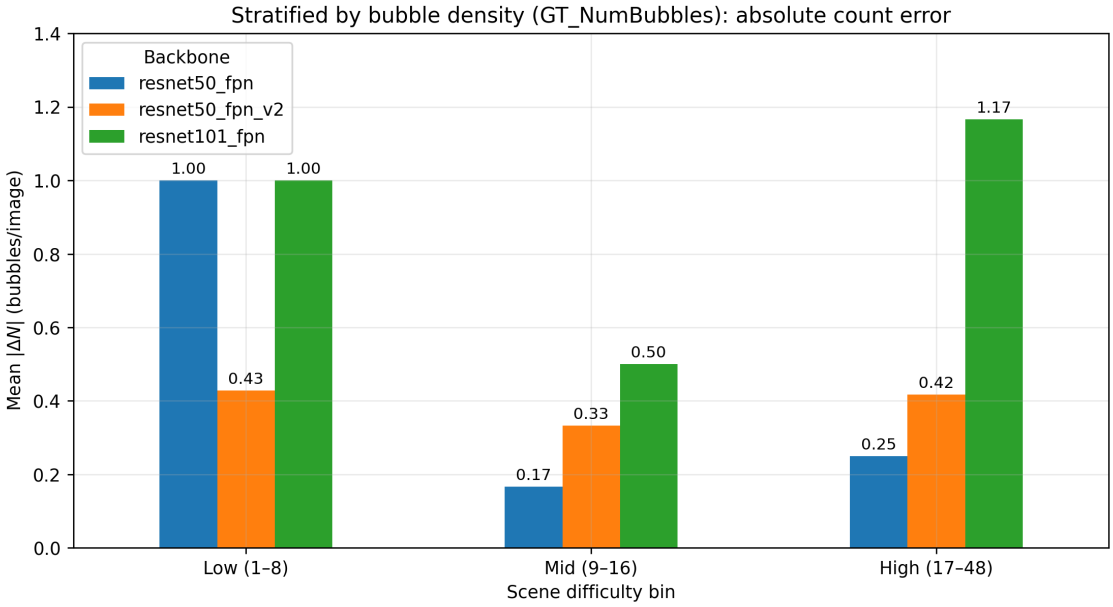


Figure 4.22: Stratified mean absolute count error $|\Delta N|$ in the transfer regime by bubble density (GT_NumBubbles bins).

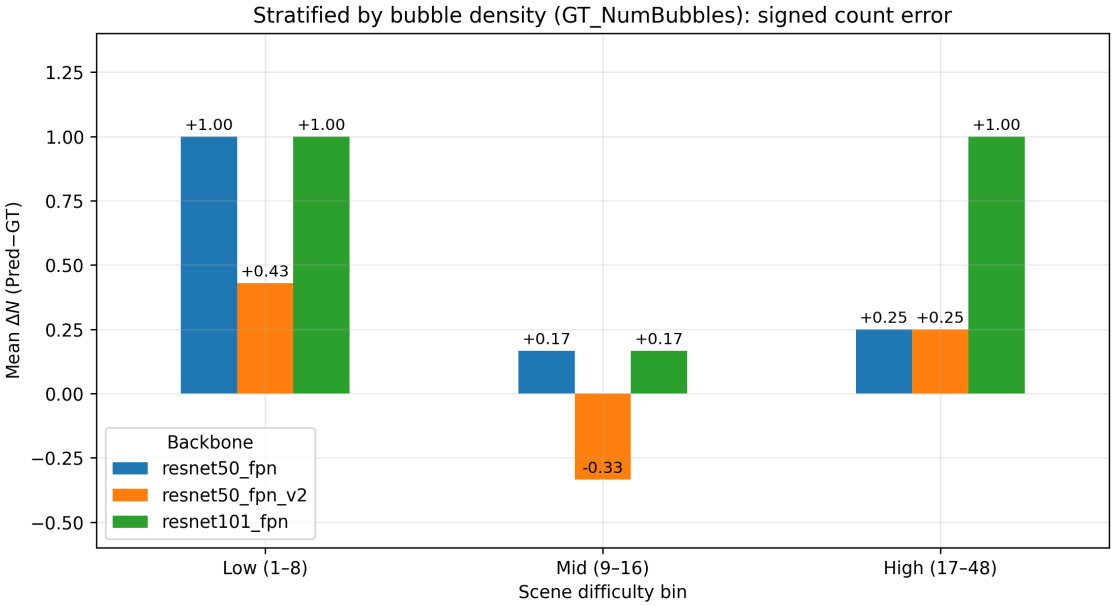


Figure 4.23: Stratified mean signed count error ΔN (Pred−GT) in the transfer regime by bubble density (GT_NumBubbles bins).

Case B: false positives induced by bubble-like structures (ImageIdx=5). ImageIdx=5 contains structures that resemble bubbles but are not annotated as such, leading to spurious detections. This is a challenging failure mode because false positives can appear visually reasonable and may not be

Table 4.7: Stratified performance in the transfer regime by mean bubble size (bins defined by GT_MeanArea quartiles on the 25-image real test split).

Bin	n	**R50 Dice**	**R50** $\lvert\Delta N\rvert$	**R50v2 Dice**	**R50v2** $\lvert\Delta N\rvert$	**R101 Dice**	**R101** $\lvert\Delta N\rvert$
Small	7	0.918	0.43	0.957	0.29	0.951	0.57
Medium	6	0.971	0.17	0.962	0.67	0.963	0.83
Large	12	0.967	0.58	0.967	0.33	0.954	1.25

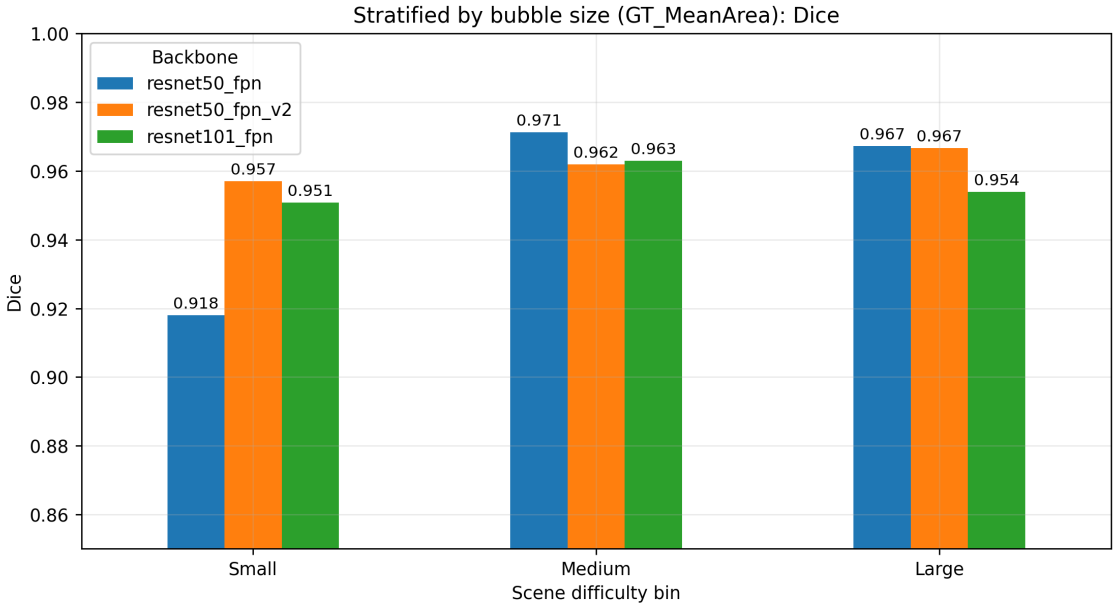


Figure 4.24: Stratified Dice (union-mask) in the transfer regime by bubble scale (GT_MeanArea bins).

strongly penalized by union-mask overlap when they occur outside the main bubble regions. The qualitative comparison shows that even the best-performing configuration can be affected by this ambiguity, while the higher-capacity backbone may exacerbate over-detection under the same thresholds.

Case C: dark and dense conditions with overprediction tendency (ImageIdx=20). ImageIdx=20 represents a visually difficult condition with darker appearance and high bubble density. Such frames increase sensitivity to contrast, boundary visibility, and post-processing thresholds, and can lead to over-prediction (extra instances) or fragmented masks. The qualitative comparison highlights how counting stability can degrade in dense scenes even without strong overlap, reinforcing the need to consider $|\Delta N|$ and ΔN alongside overlap-based metrics.

Case D: small and irregular bubble shapes successfully handled (ImageIdx=3). While error cases are essential, it is also informative to show that the best-performing regime generalizes beyond near-circular bubble shapes. ImageIdx=3 contains small and irregular shapes, yet the transfer configuration

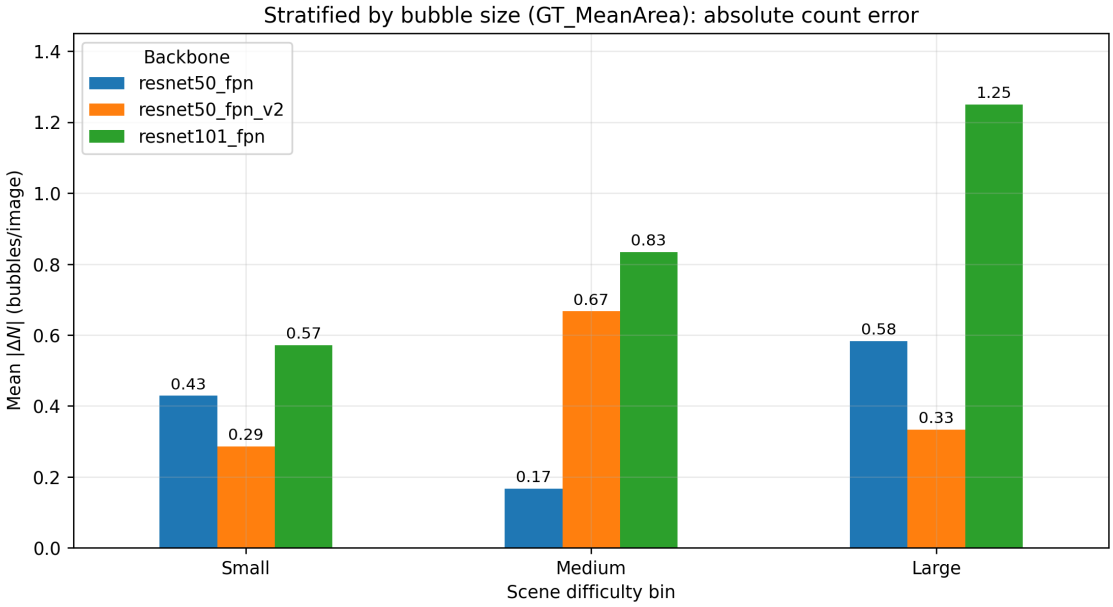


Figure 4.25: Stratified mean absolute count error $|\Delta N|$ in the transfer regime by bubble scale (GT_MeanArea bins).

remains stable and visually consistent. This case supports the quantitative observation that transfer learning improves generalization in the presence of limited labeled real data, even when bubble morphology departs from idealized shapes.

Summary of qualitative findings. Across the inspected frames, the dominant qualitative error modes are: (i) overcounting driven by spurious instances, (ii) ambiguity introduced by bubble-like artifacts in the background, and (iii) increased sensitivity to darker and denser conditions that can promote overprediction. These observations align with the quantitative finding that overlap metrics should be complemented with instance-derived indicators when segmentation is used for downstream measurements.

4.8 Annotation-efficiency results summary

This section reports results from the annotation-efficiency study described in Section 3.5. Manual annotation is compared against assisted workflows based on synthetic pre-annotations, with and without SAM2-assisted refinement, to quantify achievable time savings when expanding a real dataset suitable for fine-tuning and evaluation.

4.8.1 Quantitative results

Table 4.8 reports the mean, median, and standard deviation of the annotation time per image for each mode. The results show a strong reduction in annotation time as model assistance increases: the mean

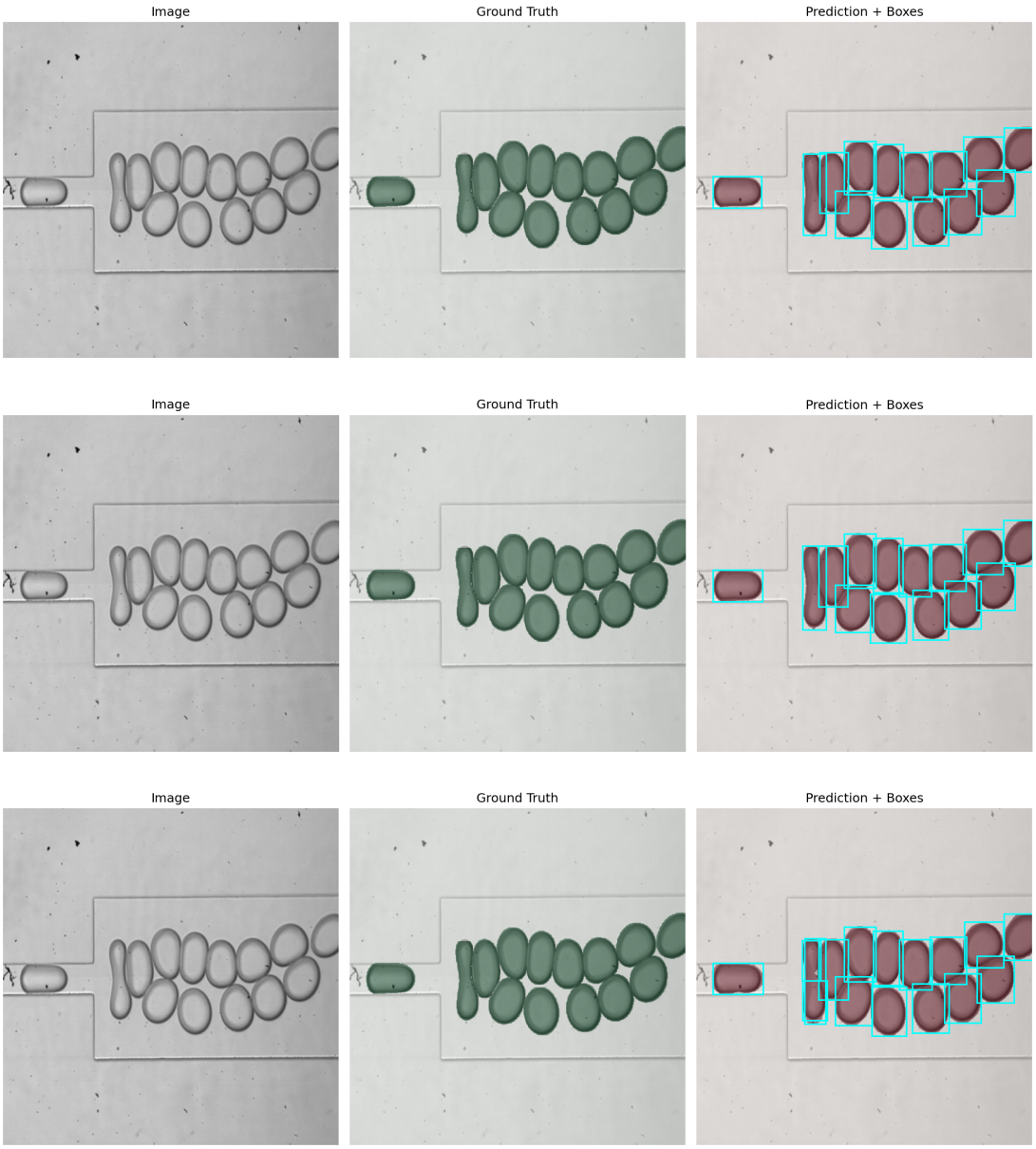


Figure 4.26: Qualitative comparison for ImageIdx=12. Top: `resnet50_fpn_v2` real-only. Middle: `resnet50_fpn_v2` transfer. Bottom: `resnet101_fpn` transfer, showing overcounting despite visually plausible overlap.

time per image decreases from 179.18 s in fully manual annotation (Mode A) to 57.07 s when starting from synthetic pre-annotations (Mode B), and to 25.11 s when combining pre-annotations with SAM2 refinement (Mode C). Relative to the manual baseline, Mode B reduces mean annotation time per image by approximately 68%, and Mode C achieves a total reduction of about 86%.

Per-image annotation times across the three modes are visualised in Figure 4.30. Each line corresponds to one of the 14 selected images and illustrates both the magnitude of the speedup and the

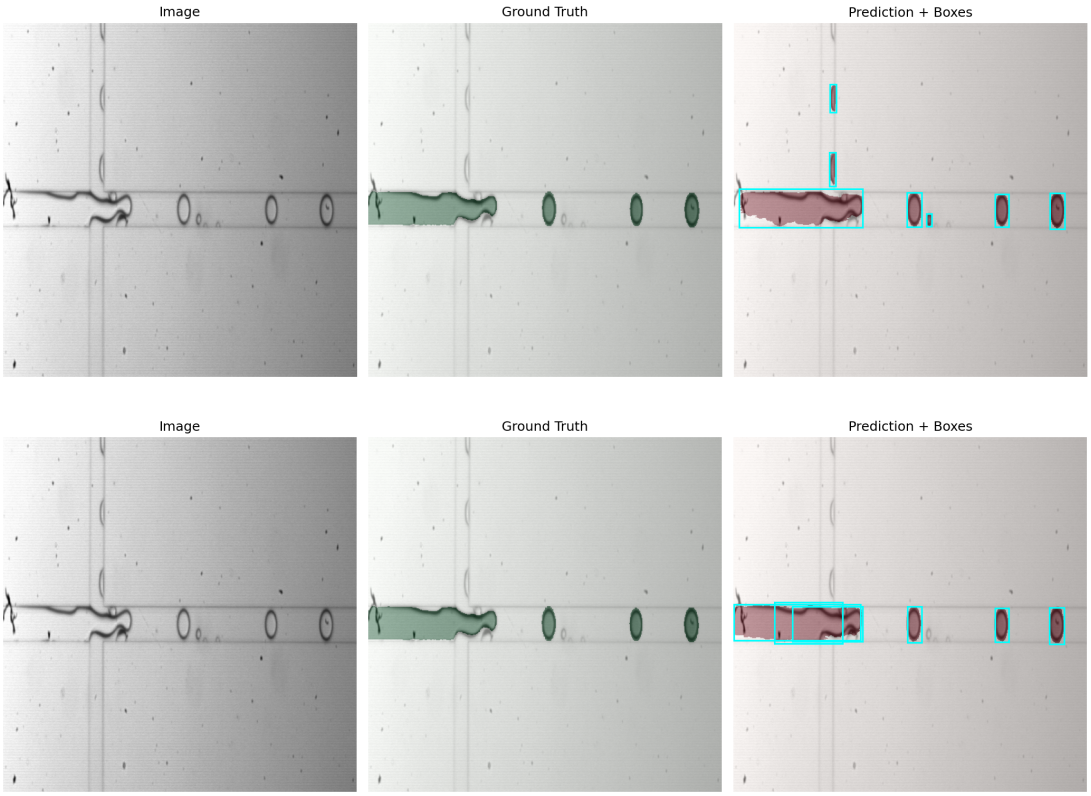


Figure 4.27: False-positive example for ImageIdx=5. Both models are challenged by bubble-like structures that are not true bubbles, leading to spurious instances and increased counting error.

remaining variability across frames.

Because the number of bubbles varies across images, annotation effort is also reported as seconds per bubble. This normalisation is important because instance segmentation time tends to scale with the number of instances that must be created, adjusted, or verified. Table 4.9 summarises the seconds-per-bubble statistics for each mode. Fully manual annotation required 94.49 s per bubble on average, whereas synthetic pre-annotations reduced this to 19.43 s per bubble. With SAM2 refinement on top of the pre-annotations, the mean annotation time per bubble decreased to 3.20 s, corresponding to an overall improvement of nearly 30× relative to the manual baseline.

Figure 4.31 visualises the distribution of seconds per bubble per mode. The plot highlights both the large reduction in central tendency and the reduction in variability for the SAM2-assisted workflow.

4.8.2 Discussion

The comparison between Modes A and B isolates the contribution of synthetic pre-annotations. Even without SAM2, starting from synthetic model predictions reduced the mean time per image by more than 68% and lowered the mean seconds per bubble by approximately a factor of five. This behaviour is

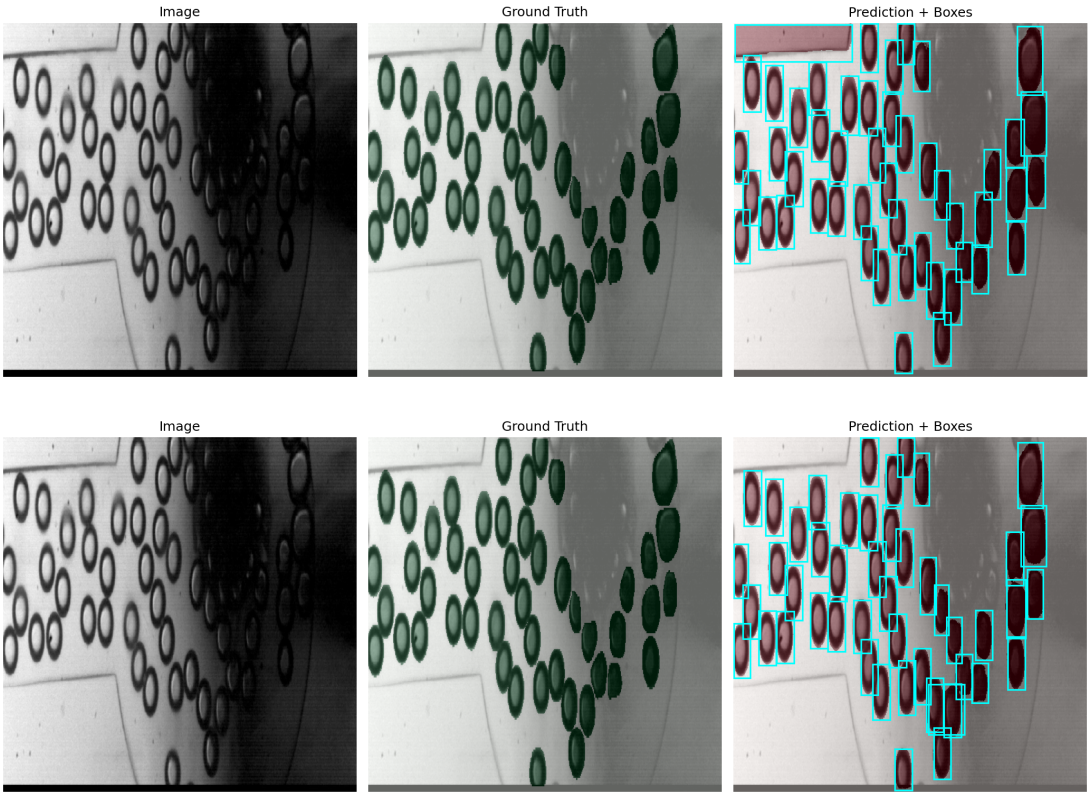


Figure 4.28: Dense, low-visibility case for ImageIdx=20. Comparison between `resnet50_fpn_v2` transfer and `resnet101_fpn` transfer highlights differences in overprediction tendency under darker and denser conditions.

consistent with strategies reported in the literature where synthetic images are used to train segmentation models that are then applied to experimental data with limited additional annotation [8, 6].

The difference between Modes B and C quantifies the incremental benefit of SAM2-based refinement. SAM2 reduces the mean per-bubble annotation time from 19.43 s to 3.20 s and the mean time per image from 57.07 s to 25.11 s. These gains are consistent with the reported capabilities of SAM and SAM2 as interactive segmentation models that can produce accurate masks from a small number of prompts [9, 15].

An example of SAM2 producing clean refinements over synthetic pre-annotations is shown in Figure 4.32.

For several images, synthetic pre-annotations were already of near ground-truth quality, so no corrections were required in Modes B and C. Label Studio therefore recorded a `lead_time` of zero seconds. These zero-effort cases indicate that, for certain operating conditions, the synthetic-trained model generalised extremely well to real data. At the same time, variability across images was observed: frames containing many small, low-contrast or strongly deformed bubbles tended to require more corrections. In such cases, SAM2 was particularly valuable, as a small number of prompts could rapidly refine

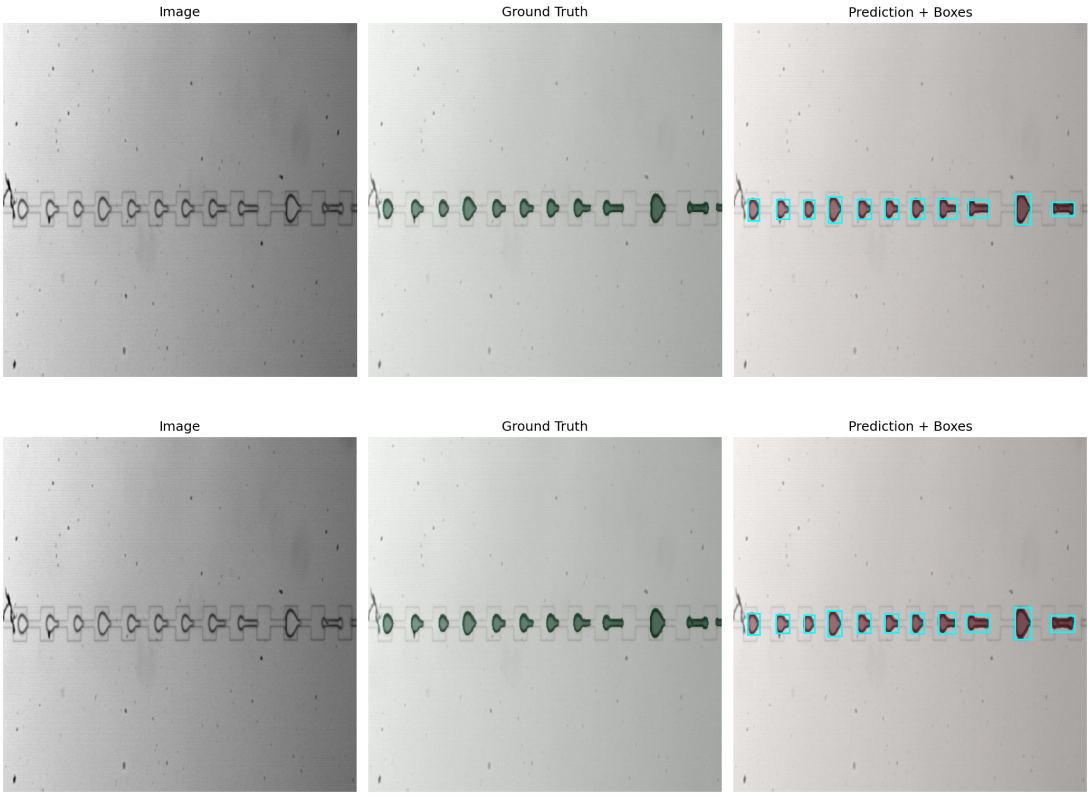


Figure 4.29: Robust success example for ImageIdx=3 with small/irregular bubbles. The transfer regime remains stable, indicating that improved generalization is not limited to near-circular instances.

boundaries that would otherwise be tedious to correct manually.

4.8.3 Implications and limitations

The achieved annotation speeds suggest that larger real datasets can be produced efficiently using this hybrid workflow. Reducing annotation time per bubble from approximately 95 s (manual) to about 3 s (synthetic pre-annotations + SAM2) corresponds to nearly an order-of-magnitude increase in effective throughput. Even if these ratios are optimistic when extrapolated beyond the 14 representative images, the gains strongly indicate that several hundred high-quality annotations can be produced within a practical timeframe, enabling robust transfer learning from synthetic to real data [8, 3]. Moreover, the presence of zero-effort cases suggests that, under some conditions, the marginal annotation cost can approach zero. This opens the possibility of future active-learning strategies where annotators focus on the most difficult images, while easy images are accepted directly or only lightly verified.

Several limitations should be considered when interpreting these results:

- **Single annotator.** All annotations were carried out by a single annotator, which ensures consistency but does not capture inter-annotator variability; time savings may differ for annotators with

Table 4.8: Mean, median, and standard deviation of annotation time per image for each mode.

Mode	Mean [s]	Median [s]	Std [s]
Manual annotation (A)	179.18	170.69	88.01
Pre-annotations only (B)	57.07	45.40	35.77
Pre-annotations + SAM2 (C)	25.11	15.34	22.78

different levels of expertise.

- **Small sample size.** Only 14 images were used for the controlled comparison. Although the frames were selected to be representative, larger-scale studies would further validate the observed trends.
- **Learning effects.** Because the same 14 images were annotated three times (once per mode), a familiarity effect may be present. This effect likely benefits the later, more automated modes; therefore, the measured speedups are, if anything, conservative with respect to the true benefit of model assistance.
- **Task definition.** The study focused on instance segmentation of bubbles only. Additional tasks such as discarding artefacts, assigning physical attributes, or labeling other flow structures were not timed and could increase annotation complexity in practice.

4.8.4 Summary

The annotation-efficiency study demonstrates that:

- synthetic Mask R-CNN pre-annotations reduced annotation time by approximately 68% per image and by about a factor of five in seconds per bubble compared with fully manual annotation;
- adding SAM2-based interactive refinement on top of these pre-annotations further reduced time by roughly a factor of six, resulting in an overall improvement of nearly 30$\times$ in seconds per bubble relative to the manual baseline;
- for several images, synthetic pre-annotations were already of near ground-truth quality, leading to essentially zero-effort annotation in the assisted modes;
- the combined synthetic pre-annotation and SAM2-assisted workflow aligns with modern semi-automatic labeling strategies and foundation-model integration in annotation tools.

These results support the practical feasibility of scaling real annotations to strengthen the synthetic-to-real transfer learning experiments presented in this dissertation.

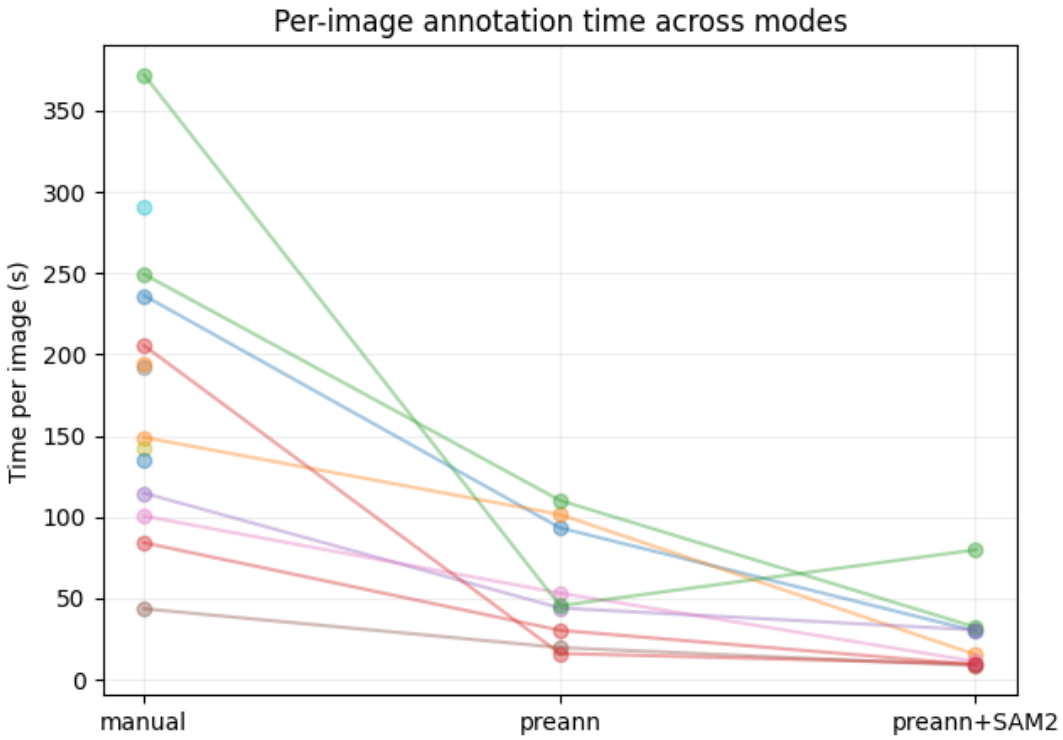


Figure 4.30: Annotation time per image across the three annotation modes. Each coloured line corresponds to one of the 14 selected images.

4.9 Key takeaways

The experimental results lead to the following main conclusions:

- **Transfer learning is the most consistently effective regime.** Initializing from synthetic-trained checkpoints and fine-tuning on real data improves both segmentation overlap and measurement-oriented indicators, with the largest gains observed for the higher-capacity backbone (`resnet101_fpn`). In addition, TensorBoard validation mAP curves show higher validation plateaus under transfer learning, supporting the test-set findings.

- **Overlap metrics and measurement indicators are complementary.** Dice/IoU quantify union-mask agreement but do not fully capture instance split/merge behavior. Reporting $|\Delta N|$ and ΔN is necessary to assess counting stability and systematic over/under-counting, which directly affect downstream bubble statistics.

- **Augmentation during transfer does not improve mean performance in this dataset.** On the 25-image real test split, the Transfer+Aug configuration does not increase mean Dice and generally worsens or leaves unchanged counting stability. ECDF analysis of per-image $|\Delta N|$ indicates

Table 4.9: Seconds per bubble across different annotation modes.

Mode	Mean [s/bubble]	Median [s/bubble]	Std [s/bubble]
Manual annotation (A)	94.49	96.23	52.90
Pre-annotations only (B)	19.43	10.09	31.51
Pre-annotations + SAM2 (C)	3.20	3.06	2.18

that augmentation effects are not uniformly beneficial and may primarily affect the error tail rather than the mean.

- **Backbone choice involves a clear speed–accuracy trade-off.** In the transfer regime, `resnet50_fpn` is fastest, while `resnet101_fpn` is slowest. `resnet50_fpn_v2` provides the best overall compromise, achieving the strongest test-set accuracy with only a moderate reduction in throughput.
- **Difficulty stratification shows where errors concentrate.** Stratifying results by GT bubble density (GT_NumBubbles) and bubble scale (GT_MeanArea) confirms that instance-level errors increase in denser and visually harder frames, even when overlap remains high. These trends reinforce the need for combined evaluation protocols in multiphase-flow measurement contexts.
- **Qualitative inspection confirms the dominant failure modes.** Representative cases show that false positives from bubble-like artefacts and overprediction in dark/dense conditions are key contributors to positive ΔN, while the best-performing configuration remains robust in typical and irregular-shape cases. Backbone capacity does not guarantee improved counting behavior under fixed post-processing thresholds.
- **Annotation efficiency gains make scaling real datasets feasible.** Assisted labeling (synthetic pre-annotations with optional SAM2 refinement) substantially reduces annotation time (Section 4.8), supporting the practicality of expanding real training data for future transfer-learning and robustness studies.

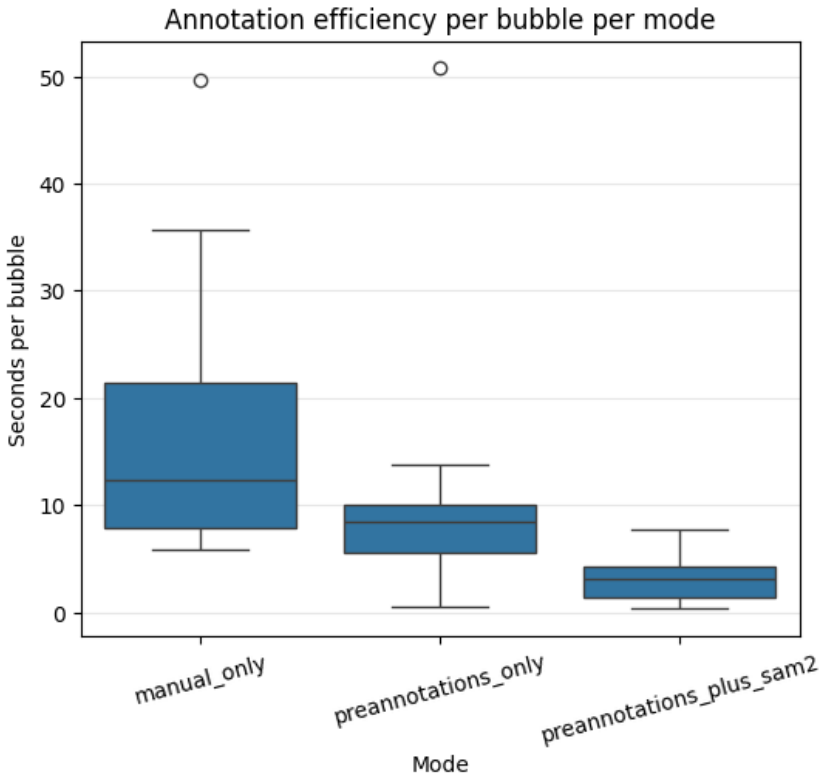


Figure 4.31: Boxplot of seconds per bubble for each annotation mode. The combined synthetic pre-annotation + SAM2 workflow (Mode C) reduces annotation cost by nearly 30× compared with manual annotation.

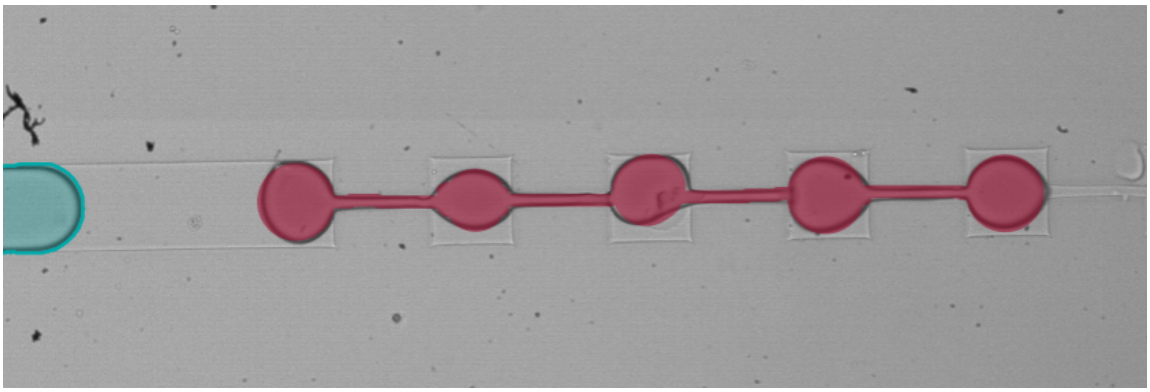


Figure 4.32: Example of SAM2-assisted refinement applied to synthetic pre-annotations, producing accurate bubble masks along the channel.

Chapter 5

Conclusion

5.1 Summary of the work

This dissertation investigated deep-learning segmentation techniques for experimental multiphase-flow imagery, with emphasis on *instance-level* bubble delineation under realistic constraints: limited real annotations, strong appearance variability, and frequent occlusion. The work was organized as an end-to-end pipeline that connects (i) dataset preparation in a COCO-style format, (ii) systematic training across multiple regimes (synthetic, real, synthetic-to-real transfer learning, and transfer learning with augmentation), (iii) a unified evaluation protocol aligned with measurement-relevant outputs (foreground overlap, bubble count/area statistics, and throughput), and (iv) a quantitative study of model-assisted annotation in Label Studio using synthetic pre-annotations and SAM2-assisted refinement.

Experimentally, three TorchVision Mask R-CNN variants were used as consistent baselines across the same protocol: `maskrcnn_resnet50_fpn`, `maskrcnn_resnet50_fpn_v2`, and `maskrcnn_resnet101_fpn`. For each architecture, four training conditions were executed (12 experiments total): training on the synthetic dataset, training on the annotated real dataset, synthetic-to-real transfer learning initialized from the corresponding synthetic checkpoint, and transfer learning with augmentation enabled.

5.2 Answers to the research questions

RQ1: How do the selected instance-segmentation models compare on the real experimental dataset?

Across the experimental grid, Mask R-CNN variants provided a practical and consistent foundation for bubble instance segmentation, matching the measurement-driven need to separate bubbles (i.e., producing one mask per bubble instance rather than a single foreground region). Under the same evaluation thresholds and preprocessing pipeline, the three backbones exhibited the expected trade-off between capacity and practicality: deeper backbones can capture richer representations but may increase training cost and inference time, while lighter variants can offer stronger speed/efficiency.

RQ2: What is the impact of synthetic pretraining and subsequent fine-tuning on real images?

A central outcome of this work is the validation of a *synthetic-to-real* strategy for bubble imagery under limited real supervision. The project leveraged a synthetic dataset with dense instance-level labels (BubANN_GAN) to establish strong initial segmentation priors, and then adapted those priors to experimental images via fine-tuning on a smaller, curated set of annotated real frames (243 images).

The transfer-learning stage was implemented as a practical protocol: initialize from the synthetic-trained checkpoint of the *same architecture*, optionally freeze the backbone for a small number of epochs to stabilize early adaptation, then unfreeze and continue fine-tuning with a reduced backbone learning rate relative to the heads. This design aimed to mitigate the domain gap while preserving reusable features learned from synthetic data.

RQ3: How much can model-assisted annotation reduce effort while preserving mask quality?

This dissertation also treated annotation as a first-class experimental variable. A controlled annotation-efficiency study compared three Label Studio workflows on the same set of 14 images: (A) manual-only annotation, (B) correction of synthetic-model pre-annotations, and (C) pre-annotations plus SAM2-assisted interactive refinement. The study used Label Studio's `lead_time` logging to quantify end-to-end effort and derived per-image and per-bubble efficiency indicators (e.g., seconds per bubble and bubbles per minute), as well as paired percentage reductions between modes.

Qualitatively and operationally, the results support the core premise that model-assisted labeling can substantially reduce the time required to produce pixel-accurate instance masks in complex multiphase-flow frames, particularly when pre-annotations are already reasonably close and SAM2 interaction is used to accelerate boundary refinement. At the same time, difficult cases (e.g., low contrast, defocus, and small bubbles) still require careful human correction, reinforcing that automation improves throughput but does not remove the need for domain-aware supervision.

5.3 Main contributions

The work delivers four concrete contributions:

1. **A reproducible, configuration-driven training framework** for instance segmentation in multiphase-flow imagery, with consistent preprocessing, logging, checkpointing, and early stopping across regimes and architectures.

2. **A systematic experimental protocol** spanning synthetic training, real training, synthetic-to-real transfer learning, and transfer learning with augmentation, repeated across three Mask R-CNN variants (12 experiments in total).

3. **A measurement-oriented evaluation pipeline** that exports per-image metrics (CSV), aggregate summaries (JSON), qualitative overlays, and plots, enabling consistent comparison across experiments beyond a single scalar metric.
4. **A quantified annotation-efficiency study** demonstrating the practical value of combining synthetic pre-annotations with SAM2-assisted refinement inside Label Studio for reducing labeling effort under dense, ambiguous multiphase imagery.

5.4 Limitations

Several limitations should be considered when interpreting the conclusions:

- **Dataset size and diversity (real domain).** The real dataset used for supervised training and evaluation is intentionally limited (243 annotated images selected from a larger extracted pool), which constrains the extent to which findings can be generalized across different experimental setups, optics, and flow regimes.
- **Single-annotator ground truth.** All experimental masks were produced by a single annotator (the author). This improves internal consistency of labeling decisions, but does not capture inter-annotator variability in ambiguous split/merge cases and boundary placement. Consequently, both the learned models and the reported evaluation metrics may reflect a specific (but consistent) annotation style. A multi-annotator protocol would be required to quantify annotation uncertainty and to assess robustness to labeling variance.
- **Domain gap and regime coverage.** Synthetic images provide scalable supervision but cannot fully reproduce experimental artefacts (illumination non-uniformity, defocus, reflections, and sensor characteristics). Transfer learning mitigates this gap, but robustness under *new* regimes remains an open practical concern.
- **Union-mask semantic metrics do not fully capture split/merge errors.** While union-mask overlap (IoU/Dice) is useful for foreground delineation, it can mask instance-level mistakes that matter for measurement (e.g., merged bubbles vs. correct separation). This is why the evaluation also reports counting error and area statistics, but the limitation remains inherent to any single-view metric summary.
- **Compute and throughput constraints.** All experiments were executed under limited hardware resources, which shaped choices such as batch size, resolution, and training schedules. Reported throughput is indicative and should be revalidated under the target deployment environment.

5.5 Future work

The results point to several high-impact directions for extending this dissertation:

1. **Expand and stratify the real dataset.** Increasing the number of annotated experimental frames and explicitly stratifying evaluation by operating condition (e.g., holdup/overlap levels, magnification, lighting) would improve the strength of conclusions about robustness and generalization.

2. **Active learning and iterative dataset growth.** The demonstrated model-assisted labeling loop can be turned into an iterative training strategy: use the current best model to propose labels, correct them efficiently, retrain, and repeat, prioritizing frames where uncertainty is highest or errors most affect measurements.

3. **Domain-gap reduction beyond standard augmentation.** In addition to the implemented training-time augmentations, future work can explore more domain-targeted perturbations (physically motivated blur/defocus models, illumination gradients, and noise models) and systematic ablations to identify which variations most improve real-domain robustness.

4. **Measurement-aware objectives and evaluation.** Since the end-goal is quantitative bubble statistics, loss functions and reporting can be further aligned with measurement error (e.g., explicit penalties for split/merge mistakes, calibration of bubble counts, and distribution-level comparisons of bubble size statistics).

5. **Temporal consistency and tracking.** Extending segmentation to video sequences and coupling it to tracking would enable richer multiphase analysis (trajectories, velocities, and evolution of bubble populations), and would also create opportunities to exploit temporal redundancy for both inference and annotation.

5.6 Final remarks

This dissertation showed that accurate and scalable bubble instance segmentation in experimental multiphase-flow imagery is feasible using modern deep-learning pipelines, but that performance and practicality depend as much on *data strategy* as on architecture choice. Synthetic supervision and synthetic-to-real adaptation provide a viable route when real labels are scarce, and interactive model-assisted annotation (pre-annotations plus SAM2 refinement) directly addresses the annotation bottleneck that often limits progress in experimental settings. Together, these components form a coherent workflow that supports both improved segmentation performance and faster dataset expansion, enabling more reliable downstream measurements from multiphase-flow experiments.

References

[1] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[2] Yizhou Cui, Chengxiang Li, Wanli Zhang, Xiaoqi Ning, Xiaogang Shi, Jinsen Gao, and Xingying Lan. A deep learning-based image processing method for bubble detection, segmentation, and shape reconstruction in high gas holdup sub-millimeter bubbly flows. *Chemical Engineering Journal*, 449, 12 2022.

[3] Behzad Karkari Gharehchobogh, Ziaddin Daie Kuzekanani, Jafar sobhi, and Abdolhamid Moallemi Khiavi. Flotation froth image segmentation using mask r-cnn. *Minerals Engineering*, 192:107959, 2 2023.

[4] Tim Haas, Christian Schubert, Moritz Eickhoff, and Herbert Pfeifer. Bubcnn: Bubble detection using faster rcnn and shape regression network. *Chemical Engineering Science*, 216:115467, 4 2020.

[5] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[6] H. Hessenkemper, S. Starke, Y. Atassi, T. Ziegenhein, and D. Lucas. Bubble identification from images with machine learning methods. *International Journal of Multiphase Flow*, 155, 2022.

[7] Hendrik Hessenkemper, Lantian Wang, Dirk Lucas, Shiyong Tan, Rui Ni, and Tian Ma. 3d detection and tracking of deformable bubbles in swarms with the aid of deep learning models. *International Journal of Multiphase Flow*, 179:104932, 9 2024.

[8] Tess A.M. Homan and Niels G. Deen. Deep learning bubble segmentation on a shoestring. *Industrial and Engineering Chemistry Research*, 63(17):7800–7806, 5 2024.

[9] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[10] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[11] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision – ECCV 2014*, pages 740–755, 2014.

[12] Ivan Malakhov, Aleksandr Seredkin, Andrey Chernyavskiy, Vladimir Serdyukov, Rustam Mullyadzanov, and Anton Surtaev. Deep learning segmentation to analyze bubble dynamics and heat transfer during boiling at various pressures. *International Journal of Multiphase Flow*, 162:104402, 5 2023.

[13] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 2010.

[14] Adam Paszke, Sam Gross, Francisco Massa, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[15] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv*, abs/2408.00714, 2024.

[16] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[17] Jee Hyun Seong, Madhumitha Ravichandran, Guanyu Su, Bren Phillips, and Matteo Bucci. Automated bubble analysis of high-speed subcooled flow boiling images using u-net transfer learning and global optical flow. *International Journal of Multiphase Flow*, 159:104336, 2 2023.

[18] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6:60, 2019.

[19] Mónica C. F. Silva, José D. P. Araújo, and João B. L. M. Campos. Cfd studies coupling hydrodynamics and solid-liquid mass transfer in slug flow for matter removal from tube walls. *AIChE Journal*, 63(6):2420–2439, 2017.

[20] Jerol Soibam, Valentin Scheiff, Ioanna Aslanidou, Konstantinos Kyprianidis, and Rebei Bel Fdhila. Application of deep learning for segmentation of bubble dynamics in subcooled boiling. *International Journal of Multiphase Flow*, 169:104589, 12 2023.

[21] Yehuda Taitel, Dvora Barnea, and A. E. Dukler. Modelling flow pattern transitions for steady upward gas-liquid flow in vertical tubes. *AIChE Journal*, 26(3):345–354, 1980.

[22] Daizhou Wen, Wuguang Chen, Junlian Yin, Yuchen Song, Mingjun Ren, and Dezhong Wang. Overlapping bubble detection and tracking method based on convolutional neural network and kalman filter. *Chemical Engineering Science*, 263:118059, 2022.

[23] B. Xin, H. Zhang, and Y. Chen. Semi-automatic annotation strategies in computer vision. *Pattern Recognition*, 113:107804, 2021.

[24] Haohan Xu, Xin Feng, Yuqi Pu, Xiaoyue Wang, Dingwang Huang, Weipeng Zhang, Xiaoxia Duan, Jie Chen, and Chao Yang. Bubsam: Bubble segmentation and shape reconstruction based on segment anything model of bubbly flow. *AIChE Journal*, 2024.

[25] Hu Zhang, Zhaohui Tang, Yongfang Xie, Xiaoliang Gao, and Qing Chen. A watershed segmentation algorithm based on an optimal marker for bubble size measurement. *Measurement: Journal of the International Measurement Confederation*, 138:182–193, 5 2019.

[26] Wen Zhou, Shuichiro Miwa, Ryoma Tsujimura, Thanh Binh Nguyen, Tomio Okawa, and Koji Okamoto. Bubble feature extraction in subcooled flow boiling using ai-based object detection and tracking techniques. *International Journal of Heat and Mass Transfer*, 222:125188, 5 2024.