SAPIENZA
UNIVERSITÀ DI ROMA

Sapienza University of Rome

Department of Basic and Applied Sciences for Engineering
PhD in Mathematical Models for Engineering, Electromagnetics and Nanosciences

Thesis For The Degree Of Doctor Of Philosophy

An AI-Based Machine Vision System for Monitoring In Vitro Diagnostic Devices: System Design and Algorithm Evaluation

Thesis Advisor

Prof. Francesca Pitolli

Candidate

Giulia Tufo

1699325

Academic Year MMXXIV-MMXXV (XXXVII cycle)

Granted by Regione Lazio and MACS s.r.l. in the framework of the call POR Lazio ESF 2014/2020 "Innovative Doctorate Intervention for the strengthening of research in Lazio - incentives for innovation doctorates for businesses" - project "IVD-AutoCert" - CUP B83C22002310009, Sigem Code 21027NP000000037




To Giorgio and Emma

Abstract

This thesis focuses on the development of Artificial Intelligence techniques for a Machine Vision System designed to improve the efficiency and reliability of In Vitro Diagnostic medical devices. The project originates from a real-world industrial need: the company holding the system's patent sought to reduce operational errors and prevent damage to the medical devices that automate the analysis of biological fluids. The work was carried out as part of an industrial PhD program, and the doctoral project was developed in collaboration with MACS s.r.l., the company that owns the patent. The proposed vision system is integrated into a laboratory diagnostic device and concentrates on monitoring the component responsible for fluid handling, the pipette tip, or simply tip.
The system provides direct and comprehensive visual feedback at two critical moments during device operation: the gripping of the tip by the IVD instrument, and the aspiration or dispensing of fluids by the tip itself. To achieve this, three distinct monitoring tasks were developed. The first checks whether the tip has been correctly picked up by the device. The second ensures that the tip being used has the correct size. The third estimates the amount of liquid aspirated or dispensed by the tip.
To implement these functions, we adopted a hybrid approach that combines classical image processing techniques with Machine Learning and Deep Learning models. While machine vision systems powered by artificial intelligence have been applied in various fields such as manufacturing, agriculture, and healthcare, their use for this specific type of diagnostic device remains unexplored in the current literature.
Different solutions were designed depending on the task. For the tip presence and size verification, we evaluated and compared several Support Vector Machines with different kernel functions and carefully selected features. For the size classification task, in addition to support vector machines, we also developed a Convolutional Neural Network to directly compare the performance of deep learning and traditional machine learning models. For the estimation of liquid volume, we implemented and compared both Convolutional Neural Networks and Vision Transformers. The results were promising across all tasks, showing high accuracy and strong generalization performance.
Given the growing importance of interpretability in deep learning, we further analysed the models using various explainability techniques to gain insights into their decision-making processes. Finally, since all models are intended for real-time application within the diagnostic workflow, we verified that they met the required processing times and were successfully integrated into the system's control software.
This work demonstrates that vision-based solutions driven by artificial intelligence can significantly enhance the reliability and automation of laboratory diagnostic tools, offering a foundation for the next generation of intelligent medical devices.

Keywords: Artificial Intelligence, Machine Vision System, In Vitro Diagnostic Device, Convolutional Neural Network, Support Vector Machine, Vision Transformer

Contents

List of Figures

List of Tables

Acronyms

C-SVC C-Support Vector Classification

AI Artificial Intelligence

ANN Artificial Neural Network

AUC Area Under Curve

Categorical CE Categorical Cross-Entropy

CNN Convolutional Neural Network

DL Deep Learning

FDs Fourier Descriptors

Grad-CAM Gradient-weighted Class Activation Mapping

HI Histogram Intersection

IG Integrated Gradients

IR Imbalance Ratio

IVD In Vitro Diagnostic

ML Machine Learning

MSSIM *Mean Structural Similarity Index Measure*

MVS Machine Vision System

NRMSE *Normalized Root Mean Square Error*

RBF Radial Basis Function

Residual *Number of residual pixels*

RMSE Root Mean Square Error

ROC Receiver Operating Characteristic

ROI Region Of Interest

SSIM Structural Similarity Index Measure

SVM Support Vector Machine

ViT Vision Transformer

XAI Explainable Artificial Intelligence

Chapter 1

Introduction

In recent years, Machine Vision Systems (MVSs) have emerged as an effective solution for the automated monitoring of processes in several sectors, such as manufacturing, healthcare, and agriculture [2–4]. Their use aims to optimize operations, reduce human errors, and automate quality control procedures. The tasks performed by an MVS range from object identification to defect detection and the measurement of geometric specifications [5, 6].

Traditionally, these operations are performed manually or through technological devices such as proximity sensors, pressure systems, or actuators [7, 8]. However, these tools present several limitations: they are often slow, expensive, and prone to errors. Furthermore, when using sensors, the measurements obtained are often indirect, which can reduce the reliability of the process [7]. In contrast, MVSs overcome these limitations by offering more objective, faster, and fully automated monitoring. Their use also reduces the subjectivity and fatigue associated with manual visual inspection [9].

MVSs integrate a combination of hardware and software components to capture and analyse visual information from the surrounding environment. The primary goal is to enable machines to make informed decisions based on the visual data they acquire. Generally, MVSs consist of several key components, such as a camera, a lighting system, and a processor. The camera is the primary input of the vision system and is responsible for acquiring images of the objects or the scene to be inspected. Cameras can be of different types depending on the task to be performed. Among the most used are the line scan cameras, which capture images sequentially, the area scan cameras, which capture images in a single frame, and the 3D scan cameras, which are capable of capturing images across three dimensions [2]. The lighting system is equally crucial, as it highlights the relevant features of the object, improving its visibility to the camera. Effective lighting is essential, as the camera cannot inspect objects it cannot see [10]. Finally, the processing unit performs preprocessing, analyses the acquired images, and executes the decision-making function. It is essential that these processors are well integrated at the software level with the system being monitored [11]

Currently, two main approaches can be distinguished in vision systems: traditional MVS and MVS based on Artificial Intelligence (AI) [5, 9]. Traditional machine vision systems are based on predefined rules, geometric models, and manual parameter tuning. Historically, MVS relied heavily on computer vision techniques, which include algorithms like edge detection (Sobel and Canny), thresholding, template matching, and various transformations such as Hough and Fourier [12]. These systems require manually defined rules, and the operating environment must remain constant: any

change in lighting, position, or shape of the object may compromise the system's functionality. In these cases, the system does not learn but merely executes predefined operations. This proves to be a limitation in production lines or in the regular operation of instruments where environmental conditions are highly variable [5].
Recent advances in artificial intelligence have led to the development of more sophisticated MVS that can learn from data and adapt to varying environmental conditions [13, 14].
The term AI refers to technology capable of performing tasks typically carried out by humans, such as image recognition, decision-making, and natural language processing. Among the main fields of AI are Machine Learning (ML) [15–17], which allows systems to learn autonomously from data, and Deep Learning (DL) [16, 18, 19], a subfield of ML that uses multi-layered neural networks to process high-dimensional and unstructured data. These two fields, or more precisely the algorithms underlying them, form the foundation of modern advanced vision systems.
The AI-based approach is based on automatic learning, allowing the system to learn from data rather than being explicitly programmed. Consequently, MVSs acquire the ability to generalize. Unlike traditional systems, AI-based MVS are capable of handling variations in lighting, object position, shape, and even background noise, making them robust and adaptable in a wide range of environments. In support of this, Manzoor et al. [20] compared classical and modern approaches, demonstrating that deep learning-based models offer better accuracy, adaptability, and generalization capabilities, especially in unstructured environments, compared to the traditional solutions. Similarly, O'Mahony et al. [21] highlights that while traditional computer vision techniques rely heavily on manual feature extraction and domain-specific knowledge, deep learning models learn to identify features directly from the data through multiple layers of processing, allowing for hierarchical representations of the input images. This capability enables deep learning models to outperform traditional methods in tasks such as image classification, segmentation, and detection.

1.1 Research motivation and scope

In this context fits the present doctoral work, which aims to develop algorithms for an intelligent artificial vision system designed for the automatic monitoring of the functioning of an In Vitro Diagnostic (IVD) medical device that automates the analysis of biological fluids. The project was carried out as part of an industrial PhD program, developed in collaboration with MACS s.r.l., the company that holds the patent for the system.
IVD devices are instruments used to automatically perform tests on biological samples in order to determine a patient's health status [22]. Like any complex instrument, IVD devices are not immune to errors. However, in the healthcare system, even a single diagnostic error, such as a false negative or a false positive, can have serious clinical consequences or lead to unnecessary costs for the healthcare system [23].
For this reason, it was considered necessary to design a system that could monitor in real time some of the key operations performed by the device. The developed system acquires and processes images of the pipette tip (or tip), the component used to aspirate and dispense fluids during testing. In particular, we focused our attention on two fundamental functions of the device: the tip grip and the aspiration and/or dispensing of fluids through the tip itself. Within these two phases, three control tasks have been defined: the *Presence Check*, the *Type Check*, and the *Volume Check*. Specifically,

these checks were designed to verify the tip grip by the medical device's retrieval system, to ensure that the picked-up tip has the correct size by distinguishing between 200 μL and 1000 μL tips, and to verify the volume of liquid aspirated and dispensed by each tip, respectively. The implementation of these checks is motivated not only by the need to reduce operational errors but also by the necessity to safeguard the device itself. Observations of the IVD system in real-world use revealed that failure to monitor these steps can result in operational faults or even mechanical damage, as will be discussed in this thesis. To develop these monitoring functions, we employed hybrid approaches that combine computer vision algorithms, machine learning models, and deep neural architectures. Several comparative analyses were performed: among Support Vector Machines (SVMs) [24, 25] with different parameter configurations for the presence check; between various SVM architectures and a deep learning model, namely, the Convolutional Neural Network (CNN) [26–28] for the type check; and between the CNN and a more recent DL approach, the Vision Transformer (ViT) [29, 30], for the volume check. The choice of these algorithms was based both on the nature of the specific tasks and on current trends in the state of the art.
Although the IVD device is already equipped with sensors to limit operational errors, these tools do not provide a direct measurement of the process, nor do they cover the entire operational cycle. The proposed vision-based monitoring solution is designed to be embedded within the actual working pipeline of the instrument, offering real-time feedback and alerting the system in the event of anomalies. However, implementing this monitoring system in real time raised both hardware and software level challenges. On the hardware side, it was necessary to define precise placements for the MVS components in order to ensure high-quality image capture while avoiding interference with the device's normal operation. On the software side, we had to take into account both the strict timing constraints of the IVD workflow and the seamless integration of AI-based algorithms into the device's native control software.

1.2 State of the art

To develop the algorithms for the three checks, presence check, type check, and volume check, we reviewed a selection of studies employing MVSs for tasks analogous to ours. Specifically, we focused on research related to object detection, shape discrimination, and liquid level estimation. These studies, primarily from non-medical or IVD fields such as manufacturing and agriculture, offer valuable insights into methodologies that support the approaches adopted in this thesis. Notably, the literature within IVD and medical device domains remains limited in this context.
The presence check, which involves the verification of the correct tip grip, is conceptually aligned with binary image classification and object presence detection. For instance, Joshi et al. [31] developed an MVS for identifying and classifying small industrial components. Their hybrid approach combines machine learning and deep learning algorithms, integrating supervised and semi-supervised layers. The study compares four different combinations of SVMs and Artificial Neural Networks (ANNs) [32], aiming to achieve zero defect tolerance in manufacturing by evaluating three distinct applications. The results are promising, with the system reliably detecting unknown objects across various scenarios, demonstrating notable flexibility. Their flexible design, validated across multiple defect types and varying operating conditions, demonstrates an important capability to generalize across unpredictable scenarios, an essential requirement for real-time monitoring in complex medical

devices. In the agricultural sector, Ireri et al. [33] proposed a system for detecting defects in tomatoes. This study involves image classification using SVMs, ANNs, and Random Forest models [34], applying targeted pre-processing and extracting features based on color, texture, and shape. The classification algorithms proved effective in identifying defects in two tomato varieties. Although the study was focused on agricultural inspection, it offers valuable guidance in feature selection and model comparison that can be applied to scenarios like ours, where the background conditions vary significantly even if image acquisition remains stable.
The type check, designed to differentiate between tips of different sizes, required an approach based on shape discrimination. This task shares similarities with the work of Bakhshipour and Jafari [35], who explored the use of SVMs and ANNs to discriminate between sugar beet plants and weeds using RGB images. Given the morphological similarities between the two plant types, the study focused on shape characteristics, such as moment invariant features and Fourier Descriptors (FDs), without considering color or texture. The results are encouraging, with SVMs outperforming ANNs in terms of accuracy, highlighting the efficacy of shape-based classification. Furthermore, these results helped us the selection of descriptors for the initial implementation.
Building on that, we further explored deep learning solutions for shape-based classification. For instance, in the agri-food industry, Lin et al. [36] introduced an MVS based on deep learning to enhance the accuracy of classifying three distinct rice kernel species, facilitating automated rice grain classification. The study compares the performance of CNNs against traditional methods like SVMs and K-nearest neighbors. The findings indicate that CNNs consistently outperform other methods in similar machine vision tasks, suggesting their potential as alternatives for rice image classification in the industry. This work motivated our use of CNNs in the type check phase, especially when the shape differences between tip types were subtle and traditional descriptors showed performance limitations.
For the volume check, focused on estimating the amount of liquid aspirated or dispensed by the tip, we reviewed several studies on liquid level detection. Pithadiya et al. [37] conducted a study on automated liquid level control in transparent bottles, employing traditional methods by comparing various optimal edge detection techniques such as, Canny, Laplacian of Gaussian, and Shen-Castan [38], to classify bottles into over-fill, under-fill, and correct fill categories. The study demonstrates that these methods significantly outperform traditional template-based methods like Sobel and Kirsch operators used in other works. Their work confirmed the advantages of optimal edge detectors over basic template-matching techniques, but also exposed the limitations of handcrafted methods in complex or noisy environments. Similarly, Felipe et al. [39] introduced a computer vision system based on OpenCV (acronym of Open Source Computer Vision Library) [40], an open-source library for computer vision algorithms, for the bottle-filling industry to monitor liquid levels in amber glass bottles, commonly used in the pharmaceutical sector. Edge detection algorithms were utilized alongside classification into over-fill, under-fill, and correct fill categories. While this approach achieves high accuracy, the system requires further development before deployment in fast-paced manufacturing lines. Finally, Cobo et al. [41] presented an AI-based study employing a CNN model to estimate red wine volume in various glass liquid containers using a single RGB image. Notably, this study does not involve classification; instead, it relies on a regression architecture to predict a continuous value (volume in mL). The model has good performance and holds potential for real-life applications in diet monitoring and wine consumption studies, offering relevant insights

for our research. Although the study uses a regression output rather than classification, it provides useful insight into the ability of deep learning for liquid volume estimation. This confirmed the value of adopting a deep learning strategy in our pipeline, even though our approach targets discrete class prediction rather than continuous estimation.
The studies reviewed in this section were selected not only to illustrate the current state of the art, but also to guide the methodological development of this work. Although they cover diverse application domains, such as industrial quality control and agriculture, they offered valuable insights into suitable techniques for object detection, shape discrimination, and liquid level estimation. These insights informed the design choices adopted in this thesis, especially in the absence of comparable prior work in the specific context of IVD systems.

1.3 Theoretical background

To provide a clearer understanding of the foundations underpinning the machine vision systems developed in this thesis, the following section presents a concise overview of the main theoretical concepts and techniques used throughout the work. In particular, it introduces core methods image processing, machine learning, and deep learning, offering a broader perspective on the tools and algorithms employed in the proposed approach.

1.3.1 Image Processing

Image processing refers to the manipulation of digital images through mathematical operations aimed at enhancing image quality or extracting relevant information [42, 43]. Typical operations include noise reduction, gray level correction, and image segmentation. In many applications, image processing constitutes the foundation for subsequent feature extraction and classification tasks [44]. Depending on the specific goal, features can be extracted either manually (handcrafted) or automatically through data-driven approaches. Traditional techniques enable direct manipulation and interpretation of image data prior to the application of machine learning algorithms [45, 46]. A common and effective technique is image thresholding, which simplifies image data by converting grayscale images into binary representations based on a specific threshold [47, 48]. One widely used approach is Otsu's method, a global thresholding technique, which automatically determines the optimal threshold by maximizing the variance between foreground and background pixel intensities [48]. This method is particularly useful for images with bimodal histograms and provides robust segmentation with minimal parameter tuning.
In the image pre-processing, morphological operations are employed to analyse and process the shape, structure, and boundaries of objects within an image [49, 50]. Among these, closing, which consists of a dilation followed by erosion. This technique helps eliminate small holes and connect adjacent components, resulting in more coherent object boundaries and improved visual consistency [49].
Filtering techniques are widely employed to smooth images, or to enhance and detect edges, improving the clarity of relevant structures. One of the most common filters is the Gaussian filter, which is particularly useful as a preparation step for edge detection, helping to reduce high-frequency noise without significantly altering object boundaries [51, 52].
Edge detection itself plays a key role in identifying the structural outlines of objects. Among the

available edge detectors, the Canny algorithm, developed by John F. Canny in 1986, is particularly popular due to its robustness and precision [53]. It combines noise reduction, gradient calculation, and threshold-based edge selection in a sequential pipeline that highlights the most significant edges in the image [54, 55].
Finally, for tasks involving object shape analysis, Fourier Descriptors offer a powerful method for representing contours in the frequency domain [56]. These descriptors represent an object's contour in the frequency domain and are useful for shape analysis and classification. FDs provide invariance to translation, rotation, and scaling, making them suitable for scenarios where robust shape recognition is required under varying conditions [33, 57].
Together, these traditional image processing techniques form the basis for effective preprocessing and feature extraction pipelines, enabling reliable performance of downstream machine learning and deep learning models

1.3.2 Supervised Machine Learning

Machine Learning is a subset of AI that allows computers to automatically learn from data and improve their performance over time without being explicitly programmed [15]. ML encompasses different learning paradigms, including supervised learning, where models are trained on labeled data; unsupervised learning, which aims to discover patterns or structures in unlabeled data; semi-supervised learning, which leverages both labeled and unlabeled data; and reinforcement learning, where agents learn optimal actions through interactions with an environment and feedback in the form of rewards or penalties [17]. In supervised learning, which is the focus of this work, an ML algorithm learns from a set of data known as the training set, from which it extracts characteristics (or features) and patterns. During the inference phase, the model uses what it learned during training to make predictions on new, unseen data [17]. In the context of this work, machine learning played a central role in in developing classification models for tasks such as object presence verification and shape-based discrimination between similar components [16].
Among the various available supervised learning techniques, SVMs were chosen for their strong theoretical foundations and practical effectiveness, especially in binary classification problems and scenarios with limited training data [24]. The fundamental principle underlying SVM is the identification of an optimal separating hyperplane, known as the decision boundary, which maximizes the margin between classes in an N-dimensional feature space. This maximum-margin criterion provides robustness to overfitting and helps ensure generalization beyond the training dataset [25]. In its linear form, the SVM attempts to find a straight decision boundary, which divides the feature space into two regions. However, real-world applications, the data is often not linearly separable. To address this issue, kernel functions were employed to map data points into a higher-dimensional space, making them separable without explicitly computing their coordinates in this new space [58]. Commonly used kernels include the polynomial, Radial Basis Function (RBF), and Histogram Intersection (HI), each providing different mechanisms for capturing data structure and non-linearity [59, 60].
The performance of an SVM is highly sensitive to the tuning of its hyperparameters, particularly the regularization parameter C and kernel-specific parameters such as γ in the RBF kernel. The parameter C controls the trade-off between the training error and the size of the margin. Typically, a high C tends to penalize misclassifications more heavily, leading to narrower margins and a greater

risk of overfitting, while a lower C allows for a wider margin and potentially more misclassified points [61]. Similarly, the γ parameter defines the influence of individual data points on the decision boundary: low values result in smoother boundaries, while higher values increase sensitivity to local variations [59].
Hyperparameter selection is commonly performed using grid search, a method that evaluates combinations of predefined parameter values and uses cross-validation. Although computationally intensive, this approach ensures comprehensive exploration of the parameter space and helps to identify optimal configurations [61].
To ensure an unbiased evaluation of model performance, k-fold cross-validation can be used throughout the experimentation. This method partitions the dataset into k equal folds, or groups. The model is trained on $k-1$ folds and evaluated on the remaining fold. This process is repeated k times, with each fold serving as the validation set once. The average performance across all k iterations is then used to evaluate the model. Typically, k is set to 5 or 10 depending on dataset size, balancing computational cost and statistical reliability [62, 63].
SVMs are frequently adopted as a baseline in classification tasks, offering a valuable point of comparison for more complex models such as deep neural networks. Their interpretability, relatively low computational demands, and good performance on structured feature sets make them particularly suitable for scenarios where handcrafted features are informative [25].

1.3.3 Deep Learning

Deep Learning is a subset of machine learning based on artificial neural networks with multiple layers, known as deep architectures [18]. While traditional ML models require multiple steps such as pre-processing, feature extraction, learning, and classification, DL algorithms are capable of operating directly on raw data, thus reducing or eliminating the need for manual pre-processing and handcrafted feature design [19]. By using huge amounts of data, DL models map the given input to output without human intervention [16].
A prominent architecture in deep learning for image analysis is the convolutional neural network. CNNs automatically extract features from input images through successive convolutional layers that learn to detect increasingly complex patterns [27]. This hierarchical approach enables CNNs to capture both low-level information, such as edges and textures, and high-level concepts like shapes and objects [28]. As a result, CNNs have proven particularly effective in image classification, segmentation, and detection tasks [26]. The fundamental element of CNNs is the convolutional layer, which applies a convolutional operation between the input image and a set of kernels to generate feature maps. These operations serve to detect pattern such as edges, corners, textures, or shapes during training. As the network deepens, it learns increasingly abstract and complex features, allowing for the accurate classification or interpretation of high-dimensional visual data [26, 28].
A typical CNN architecture includes not only convolutional layers, but also pooling layers, and one or more fully connected layers. Pooling layers perform subsampling, reducing the spatial dimensions of feature maps, decreasing computational complexity and mitigating overfitting. Fully connected layers, also referred to as dense layers connect every neuron from the previous layer of the network to every neuron in the current layer and are typically used at the end of the network to produce the final classification output [26, 28].

More recently, an alternative to CNNs has emerged in the form of ViTs. Vision transformers were proposed by Dosovitskiy et al. [29] and designed to process visual data and to execute many tasks, such as image classification, image segmentation and object detection. They build upon the transformer architecture introduced by Vaswani et al. [64] in 2017 for Natural Language Processing (NLP) tasks. ViTs split the input image into fixed-size patches, flatten them, and project them into a latent space before passing them through multiple layers of self-attention and feed-forward networks [30, 65]. One of the main advantages of ViTs is their ability to capture global characteristics of the images using the self-attention mechanism, which is a significant improvement over CNNs in certain applications. However, ViTs require significantly larger datasets and higher computational resources to reach optimal performance [66, 67].
The core innovation of ViTs lies in the self-attention mechanism, which allows models to dynamically determine the relative importance of different patches in a sequence, improving the interaction between the patches themselves [29, 64]. This mechanism enables to model long-range dependencies and relationships between different image parts.
Both CNNs and ViTs are trained using backpropagation and Stochastic Gradient Descent (SGD) or its variants [68]. During training, the model minimizes a loss function that quantifies the difference between the model's predictions and the actual outcomes. The goal of a loss function is to guide optimization algorithms on how to adjust the model parameters to reduce the discrepancy over time. To ensure generalization and avoid overfitting, various regularization techniques such as dropout, weight decay, and data augmentation are commonly applied [19].

1.3.4 Explainable Artificial Intelligence

Unlike traditional ML models, such as the SVMs, DL models are often considered black-box models due to the opacity of their internal operations and decision-making processes, which can result in mistrust, biases, and errors, especially in safety-critical domains. Explainable Artificial Intelligence (XAI) has emerged as a pivotal area of research, seeking to bridge the gap between the complexity of advanced AI models and the need for interpretability [69, 70]. In other words, XAI helps to reveal the decision-making mechanisms and provides insight into model decision in order to build trust that the AI is making correct and non-biased decisions based on facts. Moreover, explainability benefits both end-users and developers, as it can reveal weaknesses or biases in model behaviour and help identify and correct errors [71].
In the context of image classification using CNNs, several XAI methods have been proposed to enhance interpretability. Among these, attribution-based methods are particularly popular. These techniques highlight the areas of an image that have the greatest influence on a model's decision. A prominent example is Gradient-weighted Class Activation Mapping (Grad-CAM), which uses the gradient information flowing into the last convolutional layer to produce a localization map. Grad-CAM generates class-specific heatmaps that indicate which regions of the image contributed most to a given prediction [72].
Another widely adopted technique is the Integrated Gradients (IG) method, which assigns importance scores to each input feature by integrating gradients along the path from a baseline input to the actual input. It satisfies two important theoretical axioms: sensitivity, which ensures that attributions reflect meaningful differences in output, and implementation invariance, which guarantees consistency across functionally equivalent models. Although these methods differ in their

mechanisms, they all aim to offer meaningful insights into model behaviour, enhancing transparency and accountability in AI systems [73, 74].
Explainability is also critical in the context of vision transformers. Given that ViTs rely on self-attention mechanisms to process input data, the internal representations can be visualized through attention maps. An attention map is a visualization or representation that highlights the importance, or "attention", a model gives to different parts of its input data when making predictions. These maps are visual aids that highlight the importance or relevance of different parts of the input data, such as image patches. In these visualizations, areas with higher attention weights appear in warmer colors, whereas areas deemed less relevant appear in cooler shades. Such maps help users interpret how the model distributes attention across an image, contributing to the overall transparency of ViT-based systems [75].
In this work, explainability methods are employed to uncover the internal mechanisms underlying model predictions, with the aim of enhancing transparency in the classification process and improving the reliability of the system in practical applications. The use of these explainability methods enhances the transparency of deep learning models by providing insights into their decision-making processes, which is essential for model validation and informed evaluation in sensitive application domains.

1.4 Thesis Structure

Before delving into the technical implementation and experimental results, it is helpful to provide a structured overview of the thesis. This serves to clarify the logical flow of the work and to contextualize the contribution of each chapter within the broader research objectives. The content is organized to reflect the methodological pipeline adopted in this study, progressing from system design to task-specific model development and evaluation. Chapter 2 describes the structure of the developed machine vision system, detailing the design of both the hardware and software components. This chapter discusses the integration of the MVS into the IVD device, including practical considerations such as component placement and real-time constraints, to which contributions were made also in the context of hardware-related decisions. Furthermore, it describes the structural organization of the implemented checks and how they are logically correlated within the operational workflow of the IVD device. Chapters 3, 4 and 5 are dedicated respectively to the three verification tasks: Presence Check, Type Check, and Volume Check. Each chapter starts with a description of the dataset and pre-processing steps, followed by the methodology used for model training and evaluation, and concludes with a discussion of the results. The focus is on comparing different models implemented for each task, discussing their performance, and justifying the final model selection based on empirical results and task-specific constraints. These chapters also include an analysis of the role of feature extraction techniques and model parameters in achieving reliable classification. Chapter 6 presents the conclusions of the study, summarizing the main findings and contributions. It also discusses the limitations encountered and proposes future research directions to further enhance the reliability and generalizability of vision-based monitoring systems in medical applications.

Chapter 2

The real-time monitoring device

In this Chapter, we introduce a machine vision system for real-time monitoring of an IVD device, with the objective of ensuring the accuracy and reliability of its analytical operations. IVD devices are automated instruments designed to perform medical tests on biological samples, such as blood or urine, outside the human body. They typically integrate multiple subsystems for sample handling, reagent management, precise liquid transfer, reaction control, and result analysis, all coordinated to deliver rapid and reliable diagnostic results. Integrating this system directly into the diagnostic workflow reduces potential errors caused by hardware malfunctions and enables early detection of anomalies that may arise during operation.
Our aim is to minimize errors occurring in the pipetting system, the component responsible for handling pipette tips, used for aspirating and dispensing liquids. Pipette tips, or simply tips, are disposable plastic consumables commonly employed in liquid handling processes. They are typically transparent, but can also be found in other colors depending on their specific application. In addition, tips are available in different sizes to accommodate varying liquid volumes, ranging from a few microlitres to several microlitres. Figure 2.1 shows different types of tips highlighting variations in size and color (left) and inside rack (centre and right). The pipetting system includes

Figure 2.1: Example of tips different sizes and colors (left) and inside the rack (centre and right)

automated pipettors, which are tasked with selecting, gripping, and utilizing the tips during liquid handling procedures. During operation, the pipetting system follows a repetitive cycle in which the tips are first gripped, then used to aspirate the target liquid (either reagents or biological samples), and finally to dispense it into the appropriate wells of a microplate rack.
In this context, we focus on two critical operations performed by the pipetting system: the gripping

of the tip and the fluid aspiration and dispensing.

2.1 Pipetting system i +- ssues

Pipetting accuracy is critical to ensuring reliable analytical results. Any malfunction in the tip handling or in the fluid aspiration and dispensing stages can compromise reagent volumes, reaction conditions, and ultimately the validity of the analysis. It is of utmost importance to ensure that the instrument's pipetting system has correctly picked up the tip. If the tip is not taken, the device will continue to work without taking into account the anomaly, compromising the final result. This error can alter chemical reaction, as certain reagents may not be dispensed, resulting to false positives or false negatives. In the medical field such mismatches could have serious implications for the diagnosis and treatment of patients.
It is also essential to check the size of the tip taken. If the tip is taken, but not of the correct size, the amount of liquid aspirated may be incorrect, affecting reagent concentrations and distorting chemical reactions, again resulting in false positives or false negatives. Additionally, a wrong sized tip introduces mechanical risks. If the tip is smaller than expected, the instrument may draw in more liquid than the tip can contain, causing a dispersion of liquid on the mechanical parts of the instrument. Conversely, if the tip has larger dimensions than expected, the pipetting system, calibrated to operate and move over a specific dimensional range, could be subject to mechanical shocks, resulting in structural damage. Finally, errors in fluid aspiration or dispensing can compromise the accuracy of analytical processes. As shown in the problem of tip size, aspirating or dispensing incorrect amounts of liquid could alter chemical reactions and lead to wrong results. Therefore, it is essential to constantly monitor the quantities of liquid drawn in and dispensed during the analysis process.

2.2 Current monitoring systems in the IVD device

Within the IVD device, there are already monitoring systems for the tip grip and fluid aspiration and dispensing. These systems, primarily based on pressure sensors and flowmeters, provide indirect control over these two operations. Specifically, a *pressure sensor* is used to verify the presence of the tip. This sensor is positioned on a component of the pipetting system, the *mandrel*, a cylindrical mechanical shaft that engages with the pipette tip and holds it firmly in place during aspiration and dispensing operations. It measures the pressure exerted after the tip has been gripped through a predefined threshold mechanism. To check that the correct quantity of liquid is aspirated, the system relies on a *combination of a pressure sensor and a flowmeter*. These two elements work together during the aspiration phase: the pressure sensor measures the point along the z-axis (vertical) on which the free surface of the liquid is located. The amount of descent along the z-axis of the tip is measured by specific calculations for the instrument in order to aspirate the correct volume, while the flowmeter verifies that the tip is not plugged during the aspiration phase. During the dispensing phase, only the *flowmeter* is used. Its role here is again to ensure that the tip is not plugged.
These monitoring systems allow indirect control of the two operations. They do not directly measure the actual grip of the tip or the aspirated quantity of liquid, but a parameter related to them. They also do not include a check on the size of the tip, nor a measure of the amount of liquid dispensed.

2.3 The machine vision system

In contrast, the proposed system introduces an innovative approach to monitoring these two operations. Integrated into the IVD device, it captures images of the tip during operation and leverages machine learning, deep learning, and image processing algorithms to perform real-time verification. This ensures a more accurate and comprehensive monitoring process compared to traditional methods, offering a non-invasive and fully automated solution. Before delving into the adopted solutions, it is essential to outline the hardware architecture of the monitoring system and the technical considerations that guided its development.
Our monitoring system is integrated into an IVD fluid-handler device and consists of: a camera, a lighting system, a mirror, a power supply card and, an electronic board (CPU). The first three components are housed inside the structure (or case) that contains the pipetting system, while the power supply card and the CPU are located outside the case, protected by a special security structure. The electronic board controls the camera and performs all processing and algorithmic tasks. The camera is positioned in the middle of the two pipettors, 5° tilted from the vertical of the case and 3 cm away from the pipettors. The mirror is placed in front of the pipettors and the camera, at a distance of 10 cm from the pipettors and 13 cm from the camera. The presence of the mirror makes the image captured mirror-like. The lighting system is positioned on the sides of the pipettors to ensure uniform illumination. Figure 2.2 shows a schematic representation of the hardware architecture, showing a side (left) and front (right) view of the system. Figure 2.3 shows

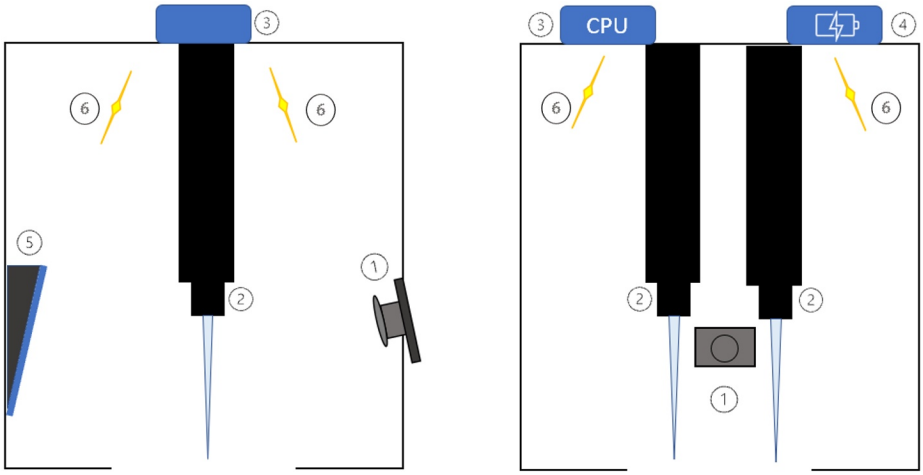


Figure 2.2: Schematic representation of the structure. Side view schematic representation of the structure (left). Front view schematic representation of the structure (right). The image shows the camera (1), the pipetting system (2), the CPU (3), the power board (4), the mirror (5) and the lighting system (6) [1].

the hardware architecture within the pipetting system: on the left there is the cross section and on the right the longitudinal section. Figure 2.4 illustrates the IVD device, the SkyLAB 752, in which the monitoring system has been integrated. Figure 2.5 shows the same device in an open state, highlighting the internal components. The red box marks the pipetting system, where the monitoring system is housed. Finally, Figure 2.6 provides a view of the interior of the IVD device, depicting the pipetting system as it picks up two tips from a rack.

Figure 2.3: Hardware architecture of the structure: cross section (left) and longitudinal section (right).

2.3.1 Hardware Design Choices

The selection of hardware components for the monitoring system was driven by functional and operational requirements. Due to the complexity of the processes involved and the limited space within the pipetting system case, it was crucial to install only the essential components for image acquisition. At the same time, careful consideration was given to ensuring that these components did not interfere with the normal operation of the IVD device. The monitoring system is a patent officially registered by MACS s.r.l. and has been integrated into the SkyLAB 752 used in MACS laboratories. The camera chosen for the system is an ELP camera with USB connection. The selection of the camera was determined by several key requirements to ensure optimal performance within the system. First, it needed to provide sufficient resolution to capture fine details of the tip and the liquid inside. Additionally, it was essential that critical parameters including focus, brightness and exposure could be adjusted to suit the operating conditions. Finally, a compact design was necessary to allow camera integration into the system without without obstructing pipettor movements. Moving to the image acquisition layout, the use of the mirror was necessary after a series of tests: by positioning the camera directly in the current position of the mirror, it was observed that it interfered with the normal operation of the pipetting system. The insertion of the mirror thus allowed to maintain a correct image acquisition without interfering with the operations of the instrument. Once the camera and mirror arrangement had been defined, attention was given to the illumination system. Given that the pipetting system case remains completely dark during operation, since the IVD instrument lights are switched off to avoid interference with chemical reactions, a dedicated lighting system was implemented. Positioned on both sides of the pipettors, it ensures uniform illumination, allowing clear and consistent image acquisition. Finally, the processing hardware was selected to meet strict performance requirements. Indeed, the monitoring system is designed to maximize accuracy and reliability, providing real-time results without slowing down the

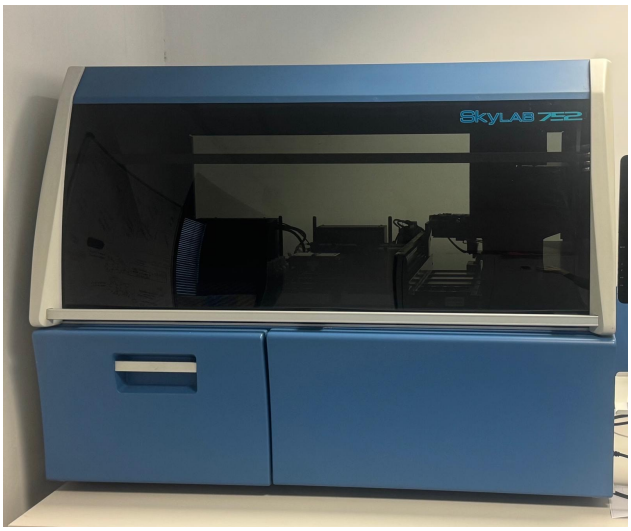


Figure 2.4: External view of the IVD device in which the monitoring system has been integrated.

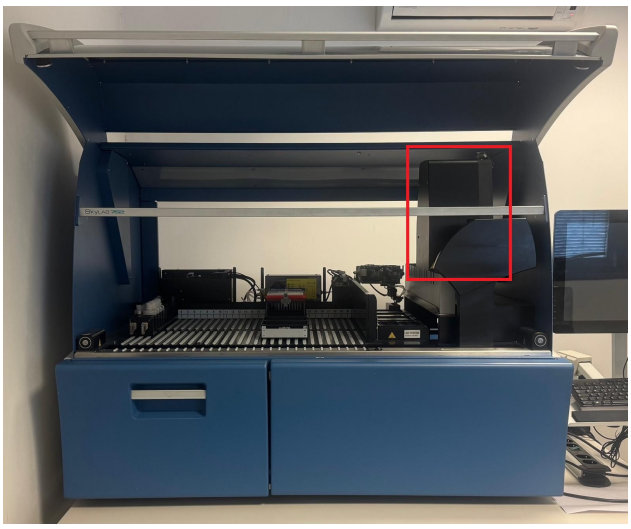


Figure 2.5: The IVD device in an open state. The red box highlights the pipetting system, which houses the monitoring system.

Figure 2.6: Internal view of the IVD device. The pipetting system (red box) is shown picking up two tips from a rack.

instrument's normal operation. For this reason, the selected CPU was chosen to ensure that the acquisition and processing time of each image did not exceed 2 seconds.

2.4 Test and checks

This section examines the tests and validations carried out prior to dataset acquisition and AI algorithm implementation. This phase is particularly critical, as it defines the system parameters and image characteristics that constitute the foundation for subsequent analyses.

2.4.1 Verification of the acquisition quality

As a first step, we verified that the hardware configuration of the system allowed to capture adequate images. In particular, we checked that the images captured contained the pipetting system clearly, especially the tips. Moreover, we ensured that the tips were present in the images in their entirety without any out-of-frame elements. Figure 2.7 shows an example of a system image before tip grip (left), after two 1000 μl tips grip (centre) and after two 200 μl tips grip (right).

Figure 2.7: Example of image acquired by the machine vision system: before tip grip (left), after two 1000 μl tips grip (centre) and after two 200 μl tips grip (right).

2.4.2 Lighting system settings

We manually conducted a series of tests to determine the best light intensity, relying on visual inspection of test images. Too low light intensity can produce a dark and poorly readable image, while too high intensity could cause overexposed areas, reducing the visibility of the liquid inside the tip, especially if the liquid is transparent or light-colored, as in the case of liquids that we will analyse in this thesis. The optimal intensity was selected qualitatively, without recording absolute luminance values, by iteratively adjusting the settings until clear and uniform visibility was achieved under all operating conditions. The tested range covered low, medium, and high settings available in the lighting controller, as illustrated in Figure 2.8. Specifically, the figure compares low intensity (left), optimal intensity (centre), and too-high intensity (right).

Figure 2.8: Example of lighting system optimization test: low intensity (left), optimal intensity (centre) and high intensity (right).

2.4.3 Camera parameter settings

After the lighting system settings were defined, we manually set the camera parameters to obtain high-quality images. The configuration was selected through visual inspection of test images. Among the various adjustable parameters, we set:

- *Resolution.* A resolution of 1280×1024 was selected to ensure detailed images while balancing quality and computational load. This choice represents an optimal compromise between the amount of visible detail and processing efficiency;
- *Focus.* It was optimized to ensure that the object of interest, the tip, in the image is clear, avoiding focusing on irrelevant background elements;
- *Brightness.* It was adjusted to stabilize the illumination of images, avoiding excessive exposure imbalances;
- *Exposure.* It was adjusted to optimize the amount of light captured, ensuring proper visibility of the object of interest without overexposures or subexposures. The exposure, brightness and intensity settings of the lighting system allow a triple control on the brightness of the images;
- *Contrast.* It was set to improve the readability of images and facilitate subsequent analysis, especially for the volume analysis phase.

This manual configuration allows to obtain images with stable and controlled characteristics, which is essential for the algorithms used.
Furthermore, we decided to acquire all BGR scale images. This choice was made because this format preserves the color quality without loss of information and allows fast conversions to other color spaces for image analysis.

2.4.4 Image pre-processing: Region Of Interest extraction

After verifying that the captured images fully contained the tip and optimizing the lighting and camera configuration, we implemented a Region Of Interest (ROI) extraction process as the first

step in image processing. This decision was driven by several factors. First, extracting only the portion of the image containing the tip removes unnecessary elements, such as parts of the working environment or the case, that could interfere with subsequent processing steps. Second, working with smaller images reduces computational cost and processing time. Third, consistently extracting the same region ensures that the algorithms receive uniform images, improving training efficiency and overall performance.
The extraction itself is performed automatically using fixed parameters, identical for all images acquired. However, the definition of these parameters, namely the ROI dimensions and coordinates, was carried out manually in a preliminary phase. Several trials were conducted to determine the optimal ROI size and position, ensuring that the entire tip was always included while minimizing the presence of irrelevant elements. The ROI height was defined according to the length of the longest tip used by the system, so that tips of all sizes would be fully contained within the selected region without risk of truncation. This choice ensures that no relevant information is lost and that all images are comparable in subsequent processing steps. This process led to the selection of a ROI of 135×635 pixels. Since the pipetting system features two sampling channels, two ROIs are extracted from each acquired image, one for each channel. The ROI will be the subject of our investigations. Figure 2.9 shows an example of a captured image and the two ROIs extracted in red rectangles. While Figure 2.10 shows the two ROI extracted from Figure 2.9.

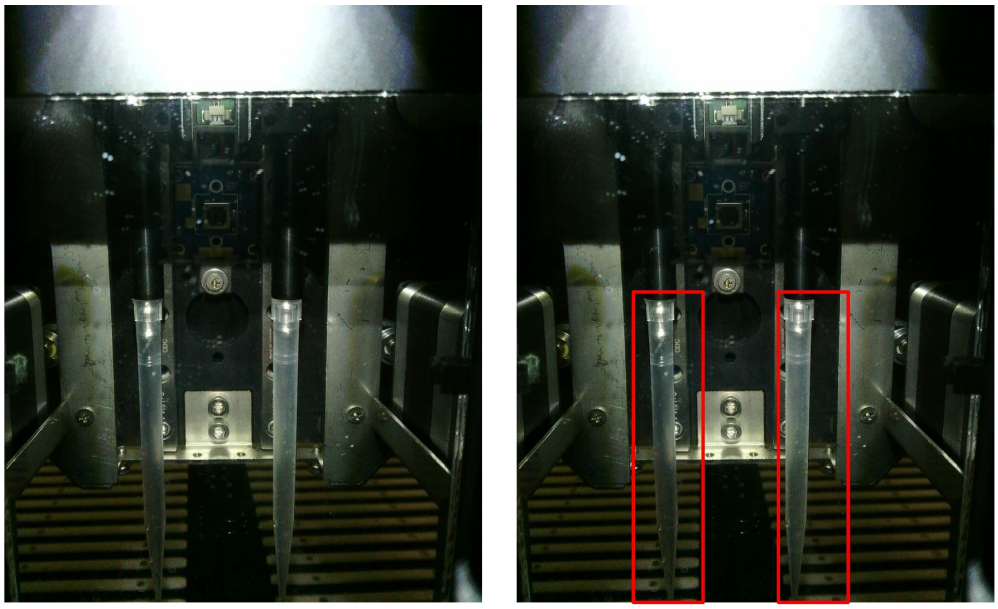


Figure 2.9: An example of ROI extraction. Image captured (left) and image captured with two ROIs selected (red rectangles) (right).

2.4.5 System performance

After optimizing the monitoring system configuration, we ran a series of tests to assess CPU performance, paying particular attention to image acquisition and processing times. Tests have shown that the system is able to capture an image and extract the ROI in less than 1 second. Furthermore, the MVS maintains a processing time compatible with the operating constraints of the IVD device. Based on these results, to ensure that the system does not interfere with the normal operation of

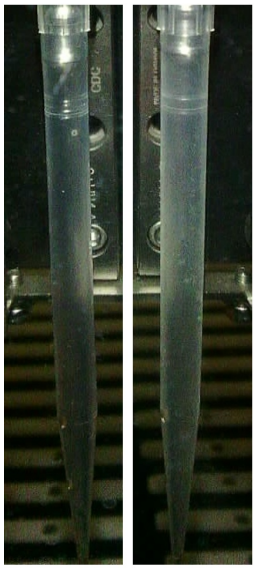


Figure 2.10: An example of two extracted ROIs

.

the instrument, it was decided that subsequent processing should not exceed a maximum time of 1 second per image.

2.5 Strategy

This section outlines the adopted strategy for monitoring the pipetting process, identifying the implemented controls and the criteria used to assess the correct functioning of the system. The IVD device operates through a repetitive cycle consisting of three main functions. First, two tips are simultaneously picked up from the same rack. These tips are then used to aspirate the target fluid, which can be either reagents or biological samples. Finally, the aspirated liquid is dispensed into the dedicated wells of a microplate rack. The monitoring system is integrated into this cycle and performs checks at specific moments to detect and prevent anomalies before they can affect the analysis.

As previously discussed, this study focuses on two critical operations of the instrument: tip gripping and fluid aspiration/dispensing. Based on these operations, the main issues identified include tip grip failure, gripping of an incorrectly sized tip, and incorrect volumes of liquid being aspirated or dispensed. To mitigate these issues, the monitoring system is designed to perform three sequential checks. The first is the *Presence Check*, which verifies the correct tip grip. This is followed by the *Type Check*, which verifies that the picked up tip corresponds to the expected size. Finally, the *Volume Check* assesses whether the correct amount of liquid has been aspirated and subsequently dispensed. These three checks are interrelated and executed in the specified order to ensure consistent and reliable operation of the device. A detailed description of each check is provided in the subsequent sections.

2.5.1 Presence Check

The first check ensures that the tip has been correctly picked up. For this task, two possible conditions are considered: tip presence, when the pipettor has correctly picked up a tip, and tip absence, when no tip is detected. If the system detects that the tip has not been successfully taken on a channel, it returns an error for that channel and terminates its verification sequence; otherwise, the process continues to the next check. The two channels are evaluated independently: an error in one channel does not affect the other. If a tip is missing on a channel, the device issues an error, discards the tip (if any) from the affected channel, and repeats the gripping operation only for that channel, while the other proceeds normally. When both channels fail to pick up a tip in the same cycle, each is handled according to the same independent-channel logic, meaning both will repeat the gripping operation individually in parallel.

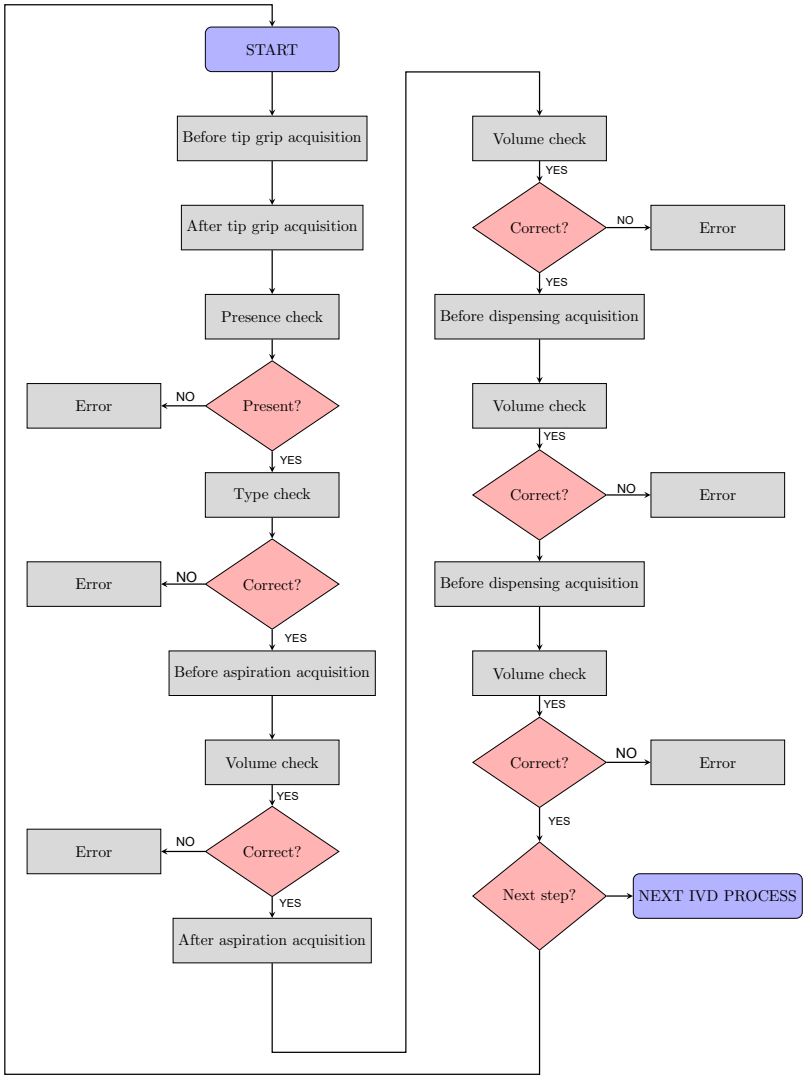
2.5.2 Type Check

The second check verifies the size of the tip taken. The IVD device equipped with the monitoring system operates with two different tip sizes: tip of 200 μL, defined as *short tip*, and tip of 1000 μL, defined as *long tip*. The monitoring system discriminates between these two types, ensuring that the tip picked up matches the expected size for the current task. As with the Presence Check, the two channels are evaluated independently. If a mismatch in tip size is detected on a channel, the system issues an error for that channel, discards the incorrect tip, and repeats the gripping operation only for that channel. If both channels have a size mismatch in the same cycle, each is corrected individually according to the same procedure, without affecting the other's workflow.

2.5.3 Volume Check

The purpose of the third check is to evaluate the amount of liquid contained in the tip at four critical moments during the pipetting cycle.
The first assessment takes place *before aspiration*, with the goal of confirming that the picked-up tip is completely empty and free from any residual liquid that could compromise the subsequent measurement. The second verification occurs immediately *after aspiration*, to ensure that the volume of liquid aspirated by the device corresponds to the expected target. A third check is performed just *before dispensing*, and serves to detect any possible leakage that might have occurred while the tip was being transferred from the aspiration station to the dispensing area. Finally, the system executes a fourth evaluation *after dispensing*, in order to confirm that the dispensed liquid volume corresponds to the expected value. If at any of these steps the detected liquid volume deviates from the expected value, the system returns an error. Otherwise, the process continues normally. The two channels are evaluated independently: an error in one channel does not interrupt or modify the operation of the other. Specifically, if, during aspiration, an incorrect volume is detected on a channel, the system issues an error, discards the liquid from the affected tip, and repeats the aspiration only for that channel, while the other proceeds without interruption. If, during dispensing, the dispensed volume does not match the expected value, the system issues an error and repeats the aspiration–dispensing operation only for the affected channel, targeting the next available well.
To implement the aforementioned checks, we established specific time points for image acquisition

and analysis algorithm application. Specifically, images are captured at the following key moments: before and after the tip grip, before and after the aspiration step, and before and after the dispensing step. This strategy ensures continuous monitoring of the process, providing reliable data for anomaly detection. A schematic workflow of the monitoring system is illustrated in Figure 2.11. Upon completing the volume check, the system can either restart from the beginning, performing checks on the next tip, or terminate the operation if the pipetting cycle is completed.



1

Figure 2.11: Schematic workflow of the monitoring system

.

Chapter 3

Presence check

The presence check is the first control made by our machine vision system. As mentioned in Section 2.2, the IVD device is already equipped with a pressure sensor that indirectly monitors the current tip grip. In contrast, our real-time monitoring system performs the same control in a direct manner focusing on the object of interest, the tip. For this task, two possible conditions are considered: tip presence, meaning that the pipettor has successfully gripped a tip, and tip absence, meaning that no tip has been gripped. This step is crucial, since—as discussed in Section 2.1—a failed tip gripping would lead the device to continue operating without detecting the anomaly, thereby compromising the overall reliability of the analysis. For the presence check, we implemented multiple SVM models, all trained using the same extracted features, but with different model configurations [1]. The variations involved different kernel functions, allowing us to evaluate which setup provided the best classification performance for detecting tip presence. The choice of SVM was driven by the need to integrate AI algorithms directly into the monitoring system's programming language, C++, ensuring compatibility and computational efficiency. To this end, all the algorithms of this, including the SVM models, have been implemented with OpenCV [1]. The SVM models were trained offline, and the best-performing configuration was subsequently deployed within the machine vision system for real-time inference.
In this Chapter, we describe all the steps developed for presence check. In detail, we will explain the dataset and the image pre-processing phases before classification, the feature extraction process and its role in the task. Furthermore, we will present the different SVM configurations tested, together with a comparative assessment of their performance. Finally, we evaluate the implemented models considering the real-time constraints to be met.

3.1 Dataset

To train and evaluate the presence check, we built a dataset of images representing both possible conditions: tip presence, where the pipettor has correctly picked up a tip, and tip absence, where no tip is detected. To this end, at each grip of the system we decided to acquire two images at the same position, but at two different times: one before the tip is gripped and one immediately after. The image captured before gripping shows the analysis scene without the tip and serves as the *background image*. Conversely, the image captured after gripping, referred to as the *scene image*,

[1] `https://opencv.org/`

depicts the analysis scene with or without the tip, depending on whether the gripping occurred or not. We acquired a dataset of 600 image pairs, each containing scene image and the corresponding background image. Since for each grip, the system picks up two tips, for each scene image and the corresponding background image were extracted 1200 ROIs, generating 1200 pairs of images which will be the input for the subsequent algorithms.
The dataset is balanced to help the model generalise effectively on new data. While class imbalance can be addressed through various techniques (e.g., re-weighting, oversampling), maintaining an even distribution between classes in our case simplified the training process and reduced the risk of bias towards the more frequent class [76]. To this end we acquired the 50% of the images that represents the presence of tip, while the other 50% represents the absence of tip.
Since the image acquisition is performed in BGR format, the extracted ROIs also retain this color representation. Before being analysed by the image processing and machine learning algorithms implemented, they are converted to grayscale to optimize subsequent analyses and enhance feature extraction. Although grayscale conversion is typically associated with a loss of color information, this step does not introduce significant distortions, as the original images contain limited color content.
Figure 3.1 shows an example of a ROI extracted from the scene image alongside its corresponding ROI from the background image under three conditions: absence of a tip (left), presence of a long tip (center), and presence of a short tip (right). Figure 3.2 displays the same ROIs after grayscale conversion. This comparison illustrates how the conversion preserves the relevant structural information while removing chromatic components, ensuring that the features used in the subsequent analysis depend solely on shape and intensity variations.

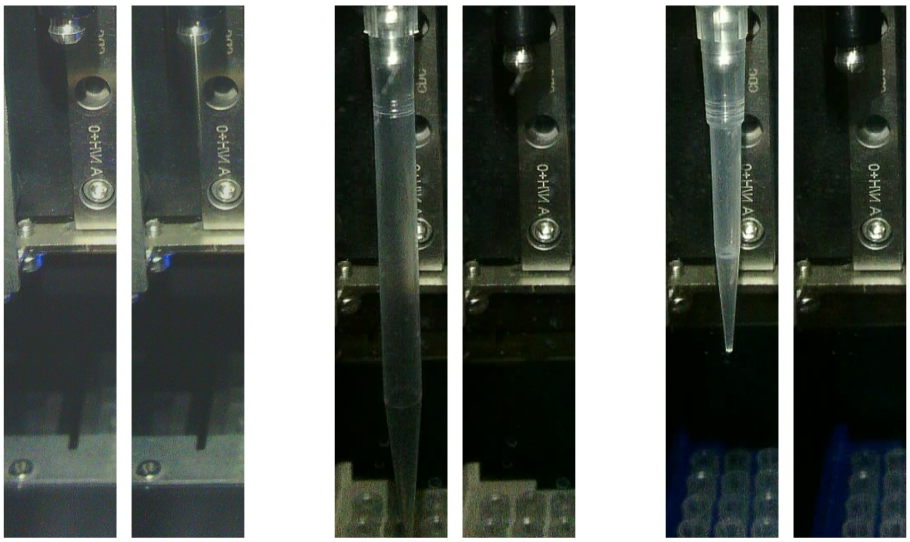


Figure 3.1: Example of a ROI extracted from the scene image alongside its corresponding ROI from the background image in case of absence of a tip (left), presence of a long tip (center), and presence of a short tip (right).

For clarity, in the remainder of this chapter, the term *scene image* will refer to the ROI extracted from the scene image, and *background image* will refer to the ROI extracted from the background image.

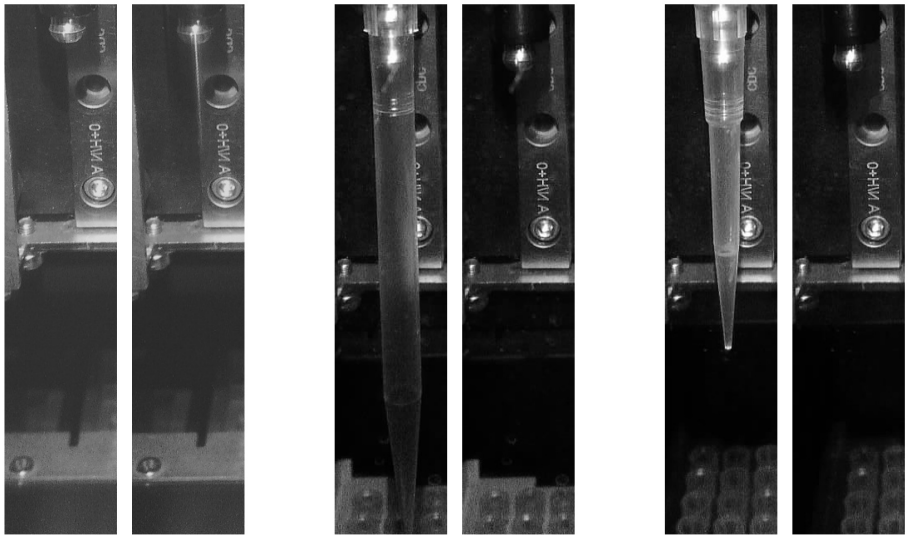


Figure 3.2: Example of a ROI extracted from the scene image alongside its corresponding ROI from the background image in case of absence of a tip (left), presence of a long tip (center), and presence of a short tip (right) after grayscale conversion.

3.2 Feature extraction

Feature extraction is a fundamental technique in machine learning and data analysis, involving the identification and extraction of relevant features from raw data to simplify complexity without losing essential informations [77]. In our application, it is critical for accurately detecting the presence or absence of the tip in the acquired images. We extract four distinct features for each pair of images (scene image and its relative background). In detail, we extracted: the *Normalized Root Mean Square Error* (NRMSE), the *Mean Structural Similarity Index Measure* (MSSIM), the *Number of residual pixels* (Residual), and the *Perimeter*.
These features were selected because they are easy to compute, which helps in reducing computational cost. Additionally, they exhibit clear differences between the classes making them effective for classification. The following section provides a detailed explanation of each extracted feature and its mathematical formulation.

3.2.1 Normalized Root Mean Square Error

The NRMSE is a metric that quantifies the error between two images, a reference image and a test image, by computing the Root Mean Square Error (RMSE) and normalizing it with respect to the range of pixel intensity values in the images [78]. This normalization makes NRMSE scale-invariant, allowing it to be used for comparisons across images with different intensity distributions or dynamic ranges. NRMSE makes a measurement by making pixel differences, treating each pixel independently. NRMSE value ranges between 0 and 1. If the value is close to 0, the error is low and the two images are similar or almost identical (0 if the images are identical). While if the value is tending to 1, the error is high and the images are different from each other.
In our case, the test image corresponds to the scene image, while the reference image is the background image.

Formally, defining the RMSE as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{M}\sum_{j=1}^{N}[X(i,j) - Y(i,j)]}{M \times N}}, \tag{3.1}$$

where $X(i,j)$ and $Y(i,j)$ are respectively the scene image and the background image of $M \times N$ dimensions, the NRMSE is calculated as follows:

$$NRMSE = \frac{RMSE}{max - min}, \tag{3.2}$$

where max and min are the maximum and the minimum values of pixels of the two images, respectively.

3.2.2 Mean Structural Similarity Index Measure

The MSSIM is derived from the Structural Similarity Index Measure (SSIM), a metric that quantifies the similarity between two images, a reference image and a test image, based on their structural, contrast, and luminance properties [78]. Unlike traditional pixel-wise error metrics, SSIM is designed to better approximate human visual perception, making it a more effective measure for perceptual image similarity.
Since MSSIM is computed from SSIM, it is essential to first understand the SSIM formulation. Let x and y be the reference image and test image, respectively. The SSIM index evaluates similarity using three main components: luminance comparison, contrast comparison, and structural comparison. In detail, the luminance comparison is defined as:

$$l(x,y) = \frac{2\mu_{\mathrm{x}}\mu_{\mathrm{y}} + C_1}{{\mu_{\mathrm{x}}}^2 + {\mu_{\mathrm{y}}}^2 + C_1}, \tag{3.3}$$

where μ_{x} and μ_{y} represent the mean pixel intensity of images x and y, respectively, and C_1 is a small constant to avoid division by zero.
The contrast comparison is defined as:

$$c(x,y) = \frac{2\sigma_{\mathrm{x}}\sigma_{\mathrm{y}} + C_2}{{\sigma_{\mathrm{x}}}^2 + {\sigma_{\mathrm{y}}}^2 + C_2}, \tag{3.4}$$

where σ_{x} and σ_{y} are the standard deviations of x and y, respectively, and C_2 is another stabilizing constant.
The structural comparison is defined as:

$$s(x,y) = \frac{\sigma_{\mathrm{xy}} + C_3}{\sigma_{\mathrm{x}}\sigma_{\mathrm{y}} + C_3}, \tag{3.5}$$

where σ_{xy} represents the covariance between x and y, and C_3 another stabilizing constant.
Finally, the three components are combined to yield the SSIM index:

$$SSIM(x,y) = l(x,y)^{\alpha} \cdot c(x,y)^{\beta} \cdot s(x,y)^{\gamma}, \tag{3.6}$$

where $\alpha > 0$, $\beta > 0$ and $\gamma > 0$ are parameters used to adjust the relative importance of the three components.
Typically, default values are set to $\alpha = \beta = \gamma = 1$ and $C_3 = C_2/2$, so the SSIM is defined as:

$$SSIM(x, y) = \frac{(2\mu_{\mathrm{x}}\mu_{\mathrm{y}} + C_1)(2\sigma_{\mathrm{xy}} + C_2)}{({\mu_{\mathrm{x}}}^2 + {\mu_{\mathrm{y}}}^2 + C_1)({\sigma_{\mathrm{x}}}^2 + {\sigma_{\mathrm{y}}}^2 + C_2)} \tag{3.7}$$

The MSSIM is calculated by averaging the SSIM values over multiple local image windows as follows

$$MSSIM(X, Y) = \frac{\sum_{j=1}^{P} SSIM(x_{\mathrm{j}}, y_{\mathrm{j}})}{P}, \tag{3.8}$$

where x_{j} and y_{j} are the image contents within the j-th local window of the scene and background image, respectively, and P is the total number of local windows. In our case, X corresponds to the scene image and Y to the background image, while x_j and y_j denote the local windows extracted from X and Y, respectively. The MSSIM value ranges between -1 and 1. A value of 1 indicates that the two images are identical, while a value close to 1 suggests that the images are highly similar. When MSSIM approaches 0, the images are considered significantly different, whereas a negative value implies that the images have no structural similarity and are completely distinct.
Figure 3.3 shows an example of a SSIM map. In detail, from left to right, you can see the SSIM maps for long tip presence, short tip presence, and tip absence, respectively.

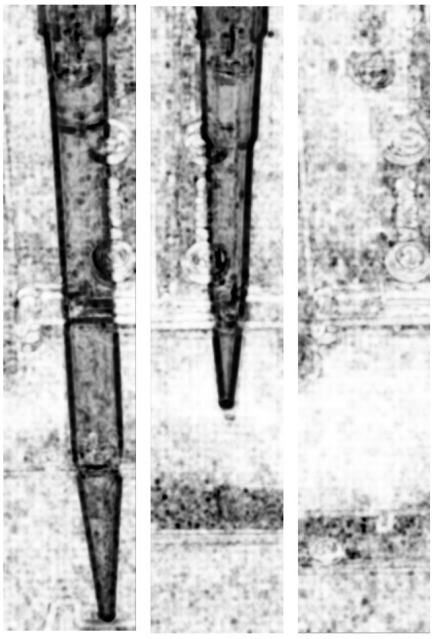


Figure 3.3: Example of the SSIM map in the case of long tip presence (left), a short tip presence (center), and tip absence (right) [1].

3.2.3 The Number of residual pixels

The decision to use the number of residual pixels, or simply Residual, was driven by the visual analysis of image pairs in both conditions (tip present and tip absent). In particular, this choice emerged from the observation of images after applying a subtraction operation between the scene image and the background image.

Residual is defined as the number of white pixels present in the image after a process of binarization and normalized to image size. The Residual computation involves a pre-processing step applied to each image pair. Specifically, the process starts with a pixel-wise subtraction of the background image from the scene image, generating what we define as the *subtracted image*. This image is then binarized using a fixed threshold value. For each pixel if the pixel intensity is less than or equal to the threshold, it is set to 0 (i.e., black pixel). While if the intensity is greater than the threshold, it is set to the maximum value (i.e., white pixel).
Denoting the resulting image of $M \times N$ dimensions as $Z(i, j)$, the Residual is calculated as follows:

$$Residual = \frac{\sum_{i=1}^{M} \sum_{j=1}^{N} Z(i,j)}{M \times N}, \tag{3.9}$$

with

$$\mathrm{Z(i,j)} = \begin{cases} 1 & \textit{if pixel is white} \\ 0 & \textit{Otherwise.} \end{cases} \tag{3.10}$$

An example of binarized subtracted image is shown in Figure 3.4.

Figure 3.4: Example of the binarized image in the case of long tip presence (left), a short tip presence (center), and tip absence (right) [1].

3.2.4 Perimeter

As in the case of the Residual, the decision to use the Perimeter was driven by the visual analysis of image pairs after applying a subtraction operation. The Perimeter is defined as the total number of pixels forming the contour of the tip normalized to image size.
To compute the Perimeter, a pre-processing step is applied to each image pair. Specifically, the process starts with a pixel-wise subtraction between the scene image and the background image, generating the subtracted image. This image is then processed as follows: a Gaussian filter with

3×3 kernel is applied to reduce noise, then the Canny edge detector is used to extract the contour of the object present in the image.
Denoting this image of $M \times N$ dimensions as $Z(i, j)$, the Perimeter is calculated as follows:

$$Perimeter = \frac{\sum_{i=1}^{M} \sum_{j=1}^{N} Z(i,j)}{M \times N}, \tag{3.11}$$

with

$$\mathrm{Z(i,j)} = \begin{cases} 1 & \textit{if pixel is white} \\ 0 & \textit{Otherwise.} \end{cases} \tag{3.12}$$

An example of image after pre-processing application is shown in Figure 3.5.

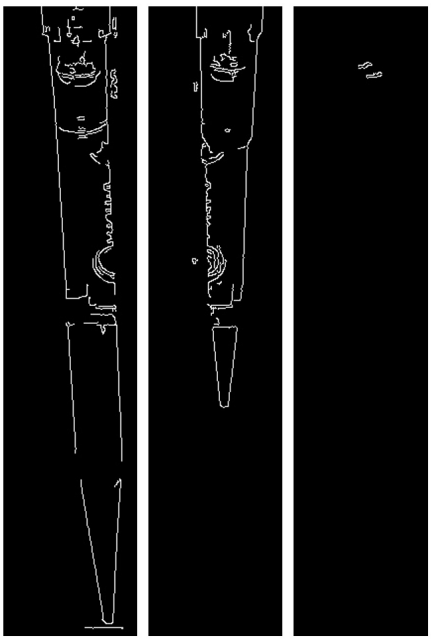


Figure 3.5: Image after pre-processing application for perimeter extraction in the case of long tip presence (left), a short tip presence (center), and tip absence (right) [1].

3.2.5 Feature extraction visualization

To better understand the relationships between the extracted features and their ability to distinguish between the two classes, presence vs absence, we generated a pairwise scatter plot. Figure 3.6 displays the distribution of the four features under consideration, based on the pairwise combination of values. Each point represents a sample, where yellow dots correspond to absence cases and purple dots to presence cases. This visualization makes it possible to visually assess whether the classes can be separated with simple linear boundaries or whether a more complex decision function is required. Some feature pairs, such as NRMSE–Residual, show a clear separation between the two classes, whereas others (e.g., NRMSE–MSSIM) exhibit partial overlap. This indicates that while certain features may allow for accurate separation using simple thresholds, in other cases a more flexible, non-linear boundary may be required to correctly classify borderline samples. This supports the choice of exploring non-linear classifiers that can exploit the joint information contained in multiple

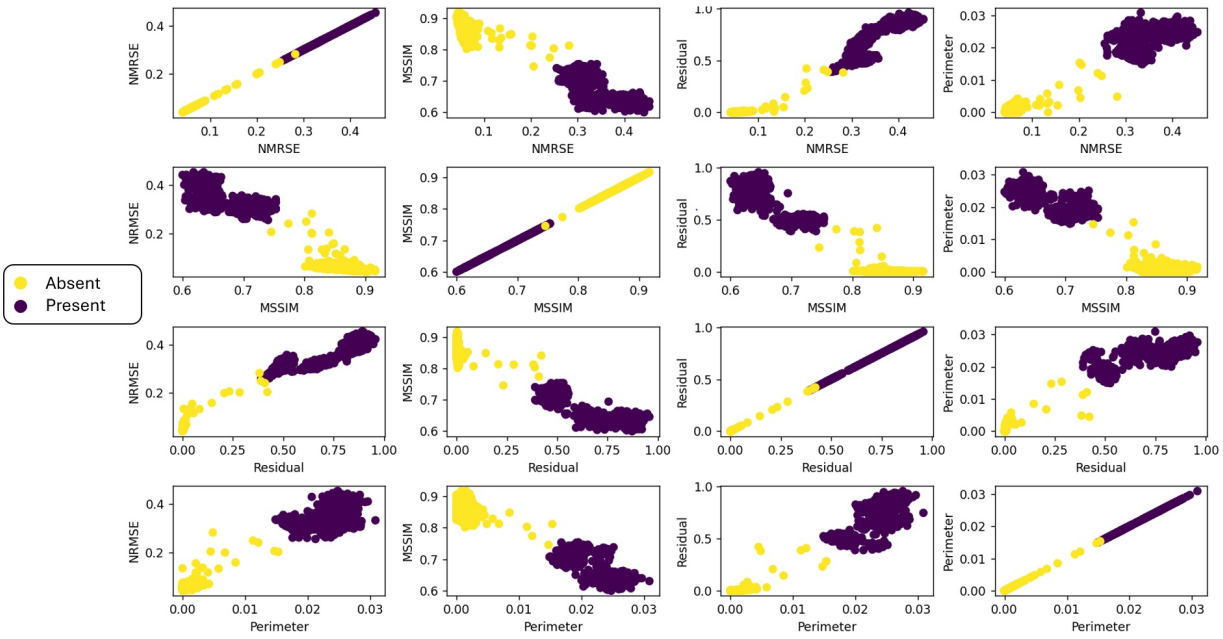


Figure 3.6: Distribution of the four features considered based on the pairwise combination of values [1].

features. To complement the visual analysis, Table 3.1 reports the mean and the relative standard deviation values for each feature, computed separately for the two classes. These statistics further highlight the numerical differences between presence and absence cases, confirming the potential discriminative power of the selected features.

Output	NRMSE	MSSIM	Residual	Perimeter
Tip absence	0.0602 ± 0.0221	0.8641 ± 0.0229	0.0090 ± 0.0371	0.0013 ± 0.0013
Tip presence	0.3423 ± 0.0442	0.6763 ± 0.0420	0.6442 ± 0.1641	0.0218 ± 0.0036

Table 3.1: Mean and corresponding standard deviation of each presence check feature, reported separately for the two classes.

3.3 Models

As already explained in this Chapter, the decision to use a support vector machine for this task was driven by the need to integrate, as much as possible, AI models into the monitoring system's programming language, C++. To achieve this, we implemented multiple SVM models using the OpenCV library, which includes a dedicated machine learning module with built-in SVM support. Consequently, both training and cross-validation were performed entirely in C++.

The same feature set extracted from the dataset was used for all SVM models to ensure consistency in the evaluation. Additionally, we employed a C-Support Vector Classification (C-SVC) model, which allows for imperfect class separation by introducing a penalty multiplier (C) to handle misclassified data points.

As previously shown in Figure 3.6, the scatter plot of extracted features shows partial class overlap in some pairwise projections; this does not rule out linear separability in the full feature space,

but it suggests that a strictly linear decision boundary could be suboptimal for borderline samples. Therefore, in addition to testing a Linear SVM, we also explored non-linear kernels, such as RBF and HI, which are better suited to capture potential local interactions among features. The selection of the C parameter was performed through an empirical analysis using a grid search approach, combined with k-fold cross-validation to ensure robust generalization. The C parameter controls the trade-off between achieving a low training error and maintaining good generalization. Indeed, a high C value encourages the model to fit the training data more strictly, reducing misclassification but increasing the risk of overfitting. While, a low C value results in a simpler decision boundary, which can help prevent overfitting but may lead to higher misclassification rates.
To determine the optimal C for each kernel, we performed a grid search over multiple values, selecting the one that achieved the best performance in terms of cross-validation accuracy.
For models using the RBF kernel, we also tuned the γ parameter, which controls the influence of individual training samples. Higher values of γ lead to more complex decision boundaries, which can improve classification but also risk overfitting. All C and γ parameters were directly configured in the C++ training implementation. In this process, the explored values of C and γ spanned the interval $[10^{-5}, 10^{7}]$, with logarithmic scaling. The final C and γ values used for each model, obtained from the grid search, are summarized in Table 3.2.

Kernel	C value	γ value
Linear	$3.125 \cdot 10^{2}$	N/A
RBF	$3.125 \cdot 10^{2}$	$5.0625 \cdot 10^{-1}$
HI	$6.25 \cdot 10^{1}$	N/A

Table 3.2: C and γ parameters for each model after grid search optimization.

To ensure robustness and prevent overfitting, we adopted a simple k-fold cross-validation strategy. We set $k = 10$, a commonly used configuration in the literature, as it provides a good balance between bias and variance, yielding a stable performance estimate [63]. After performing 10-fold cross-validation, the best-performing model was selected for further evaluation. Since cross-validation was performed with simple k-fold (not stratified), individual folds may exhibit slight shifts in class proportions [79]. Accordingly, following model selection—i.e., choosing the model that achieved the highest accuracy—we treat accuracy and F1-score as co-primary metrics throughout; precision and recall are also reported to make the trade-off between false positives and false negatives explicit.

3.4 Results

Evaluation on training and validation set

We trained the three SVM models using the same 4 features (NRMSE, MSSIM, Residual and Perimeter) extracted from the dataset, and employed 10-fold cross-validation. This means each model was trained for 10 times and validated on a different fold in each iteration. Table 3.3 presents the accuracies achieved on the validation fold for all implemented models, expressed as percentages. Table 3.4 reports the mean accuracies along with the corresponding standard deviations.

As shown, all three models achieved excellent results in distinguishing images with tip from those

Kernel	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10
Linear	100%	100%	100%	99.17%	100%	100%	100%	100%	100%	100%
RBF	100%	100%	100%	99.17%	100%	100%	100%	100%	100%	100%
HI	100%	100%	100%	100%	100%	100%	100%	100%	100%	99.17%

Table 3.3: Accuracies on the validation fold for each model type.

Kernel	Mean Accuracy
Linear	99.92% ± 0.25%
RBF	99.92% ± 0.25%
HI	99.92% ± 0.25%

Table 3.4: Mean accuracies along with the corresponding standard deviations for all models.

without tip. This indicates that the models are well-suited for the classification task for which they were designed. By selecting the best model for each kernel type, Table 3.5 presents the performance metrics for each model. All metrics are expressed as percentages, except the Area Under Curve (AUC) and the number of support vectors.

Kernel	Accuracy	Precision	Recall	F1-score	AUC	Support Vectors
Linear	100%	100%	100%	100%	1	2
RBF	100%	100%	100%	100%	1	6
HI	100%	100%	100%	100%	1	10

Table 3.5: Performance metrics for each model kernel type.

Although all three models exhibit identical performance—as evidenced by their 100% accuracy, precision, recall, F1-score, and AUC values—the primary distinguishing factor is the number of support vectors. Indeed, model selection was partially guided by this parameter. In SVM, a model with a large number of support vectors (approaching the training set size) may be a signal of higher boundary complexity and potential overfitting. On the other hand, a smaller number of support vectors can indicate better generalization, as the model depends on a narrow set of training data; however, the support-vector count alone does not determine generalization. A model that is too simple might not adequately capture the complexity of more intricate or non-linearly separable data [80, 81]. In our experiments, the Linear SVM relies on two support vectors, indicating that a single hyperplane cleanly separates the data with a wide margin. While this simplicity is attractive, a strictly linear boundary could be suboptimal for borderline samples or mild acquisition shifts that induce local non-linear interactions among features. In contrast, the RBF SVM and the HI SVM present a more reasonable number of support vectors, suggesting a good balance between complexity and generalization. Having 6 or 10 support vectors out of 1080 training samples does not indicate excessive model complexity, but rather allows them to capture the overlap highlighted in the scatter plot. Finally, although a single-feature threshold (e.g., on NRMSE) may suffice under stable conditions, it is inherently more sensitive to noise and small acquisition artifacts and cannot exploit joint information across features; the SVM approach addresses these limitations while keeping inference cost negligible. For these reasons, we retain the RBF SVM as the default model—matching the top-line metrics, keeping inference cost negligible, and offering a tunable length-scale to control boundary smoothness—while reporting the Linear model as a strong baseline

and excluding it from downstream analyses. Figures 3.7 and 3.8 show the confusion matrix and the Receiver Operating Characteristic (ROC) curve of RBF SVM and HI SVM for validation fold, respectively.

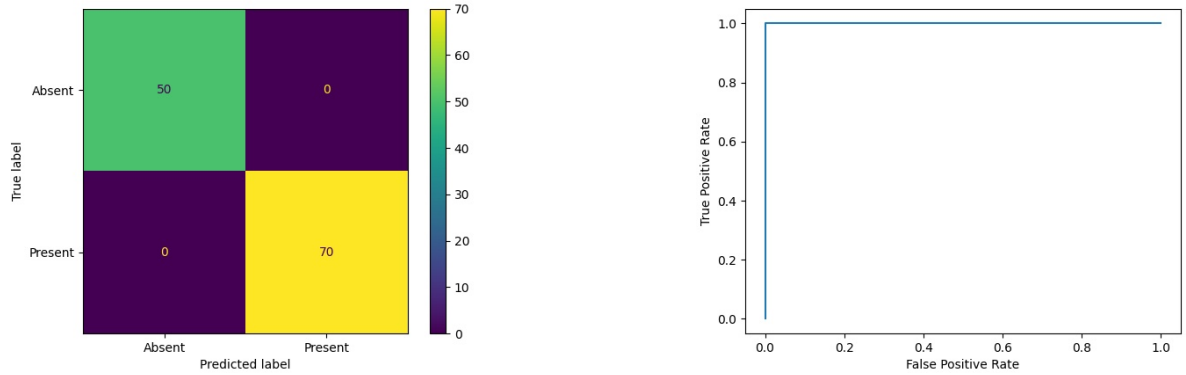


Figure 3.7: Confusion matrix (left) and the ROC curve (right) of the best RBF SVM on validation fold [1].

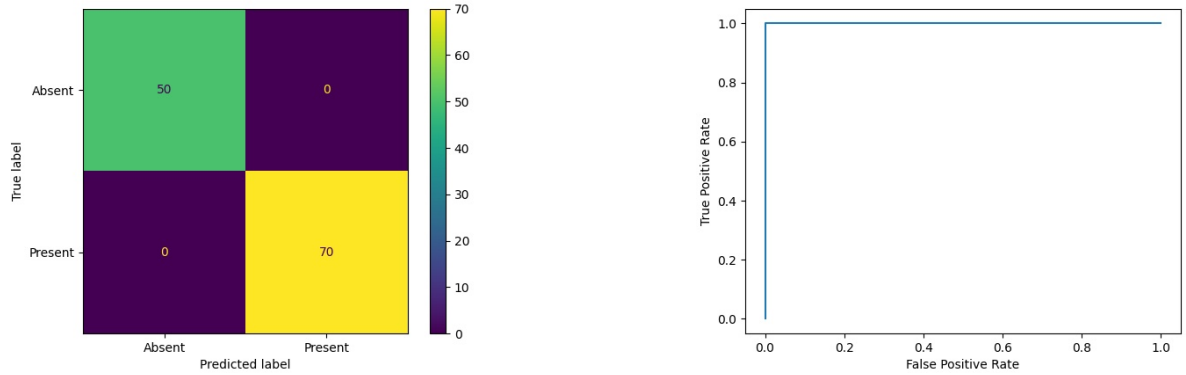


Figure 3.8: Confusion matrix (left) and the ROC curve (right) of the best HI SVM on validation fold [1].

Evaluation on an external test set

To assess the robustness of the selected models beyond cross-validation, we evaluated them on an external test set, consisting of images that were never used during training or validation. This dataset was acquired by the vision system during a standard working session and subsequently stored for testing. It comprised 432 scene images with corresponding background images, balanced between 216 images representing the presence of the tip and 216 images depicting its absence. The experiments were conducted on hardware identical to that integrated in the IVD device, ensuring that the assessment reflects the actual deployment conditions. This external test set provides an unbiased evaluation of the models' generalization performance and their reliability in real-world scenarios.

Figure 3.9 presents the confusion matrices obtained by the RBF model (left) and the HI model (right). The RBF SVM and HI SVM achieved an accuracy of 99.31% and 99.07%, respectively, demonstrating excellent performance for the task it was trained for. As k-fold cross-validation

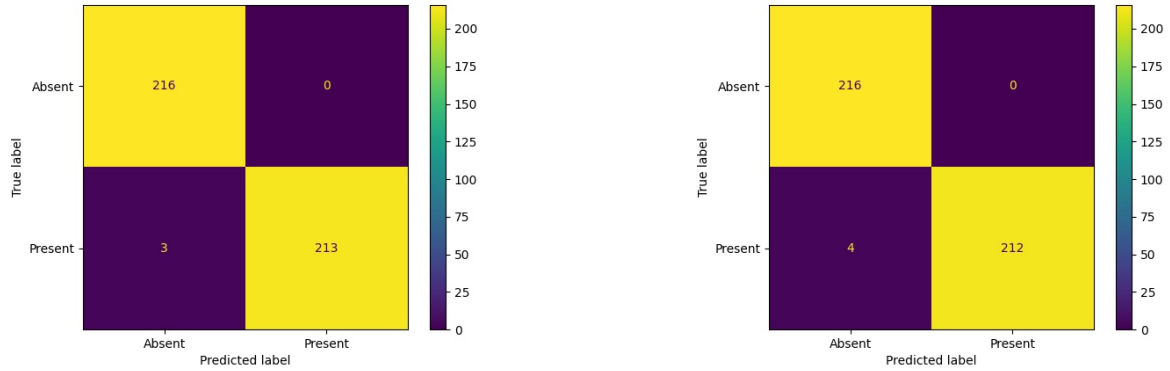


Figure 3.9: Confusion matrices obtained by the RBF model (left) and the HI model (right) on test set [1].

was not stratified, we also report the F1-score: 99.30% for the RBF model and 99.07% for the HI model. The confusion matrices also show a few false negatives, meaning that a tip correctly present was classified as absent. In the workflow of the device, this situation is handled as an error: the channel discards the tip and repeats the gripping operation, while the other channel proceeds normally. Therefore, false negatives do not affect the reliability of the process, but only cause a slight slowdown due to the repetition step.
Although the differences in accuracy, F1-score, and support-vector count are minimal, we selected the RBF model for presence check to ensure a better generalization without compromising too much performance.

Time performance

As explained in Section 2.4.5, our MVS must perform the checks within a timing compatible with the operative constraints imposed by the IVD device. Specifically, each check must be completed within 1 second per image. For this reason, we performed tests to measure the time required for feature extraction and classification by the monitoring system to perform the presence check. Table 3.6 shows the average extraction, prediction and total times, along with the corresponding standard deviations, for the selected model. The average was calculated over the 432 images of the test set, and the measurements were conducted on a system with hardware identical to the one used in the IVD device.

Function	Time
Feature extraction	$1.73 \cdot 10^{-1} \pm 1.90 \cdot 10^{-2}$ s
Prediction	$3.602 \cdot 10^{-5} \pm 1.03 \cdot 10^{-4}$ s
Total	$1.73 \cdot 10^{-1} \pm 1.90 \cdot 10^{-2}$ s

Table 3.6: Average extraction, prediction, and total times with standard deviations for the selected model.

The results demonstrate the total time employed to execute presence check is consistent with the time required for this step, and it remains well within the 1-second limit.

Chapter 4

Type check

The second check verifies the size of the tip taken. This check is only performed if the presence check has been positive, i.e. tip is detected. Its importance is crucial, because as discussed in Section 2.2, the pipetting system currently does not include an integrated mechanism to verify the type of tip selected. Such a limitation may cause errors in liquid dosing, which can compromise the accuracy of chemical reactions and diagnostic results, and may also lead to mechanical damage to IVD device components.
To mitigate these issues, we implemented a control that discriminates between two different tip sizes: tip of 200 μL, defined as *short tip*, and tip of 1000 μL, defined as *long tip*.
As for the presence check, we implemented multiple SVM models, all trained with the same features, but with different configurations, i.e. different kernels functions. Once again, the choice of SVM was driven by the need to integrate the algorithms directly into the monitoring system's programming language, C++. The models were implemented using the OpenCV library [1].
In addition to SVM, we also implemented a deep learning model, specifically a CNN [1]. This choice is motivated by two main consideration: the performances of SVM models, as will be detailed in the following sections, and the nature of the task, which concerns the discrimination of an object based on its size and, therefore, its shape. CNNs are well suited for such tasks, as they automatically extract hierarchical features from images, focusing on geometric and structural aspects. Furthermore, CNNs are able to perform more complex tasks, potentially improving performance compared to traditional machine learning models.
In contrast to the SVM models, CNN was implemented, trained and tested in Python using the TensorFlow and Keras libraries [2], two open-source deep learning frameworks [82]. Although our MVS is designed in C++, the use of a model developed in another programming language did not result in any compatibility issues. Preliminary tests confirmed the successful integration of Python models into the monitoring system. In practice, the deep learning models were trained offline in Python and subsequently integrated into the vision system for real-time inference. This strategy allows the computationally demanding training phase to be carried out with external resources, while inference remains lightweight and fully compatible with the industrial workflow of the device.
In this Chapter, we describe the dataset acquisition and the pre-processing techniques applied. Next, we explain the feature extraction process and its role in SVM models, followed by a compara-

[1] `https://opencv.org`
[2] `https://www.tensorflow.org`

tive evaluation of different SVM configurations tested. Furthermore, we introduce the CNN model, discussing its architecture, implementation, and data augmentation strategies. Finally, we evaluate the performance of models considering the real-time constraints to be respected.

4.1 Dataset

To train and evaluate the AI algorithms for the type check, we built a dataset of images considering the two classification cases: short tips and long tips. As in the presence check, we used images acquired before and after tip gripping. A total of 644 image pairs were collected, each containing a scene image and its corresponding background image, following the same definition introduced in Section 3.1. The dataset is balanced, ensuring equal representation of both classes, with 50% of the images containing short tips and 50% containing long tips.
An example of scene image and its corresponding background in the case of long tip (left) and short tip (right) is shown in Figure 4.1.

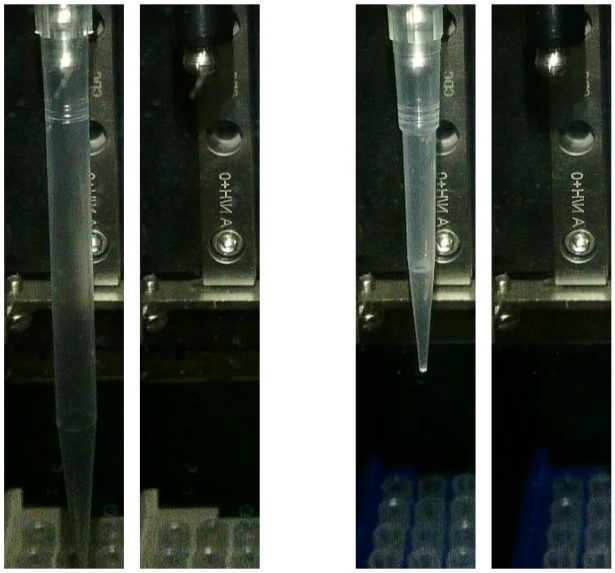


Figure 4.1: Example of scene image and its corresponding background in the case of long tip (left) and short tip (right).

4.2 General pre-processing steps

Before put images as input to the models, we applied a pre-processing step. The pre-processing consists of a common procedure, applied to both SVM and CNN models, and a model-specific pre-processing, which will be described in their respective sections.
This section focuses on the general pre-processing pipeline shared by both approaches.
Since the acquired images are in BGR format, they were first converted to grayscale. Next, we applied image subtraction to each pair of image: the background image was subtracted from the scene image. The resulting subtracted image was resized from 135×635 pixels to 256×256 pixels. This resizing step was primarily required to improve computational efficiency for the CNN model. However, to maintain consistency and ensure a fair comparison between the SVM and CNN models,

we applied the same resizing to the SVM pipeline. Although resizing can introduce minor distortions, the transformation was not excessive, and the key visual characteristics necessary for classification were preserved. Figure 4.2 illustrates an example of pre-processing up to the subtraction step for a long tip, while Figure 4.3 depicts the same process for a short tip.

Figure 4.2: Example of pre-processing up to the subtraction step for a long tip. The figure displays the scene image and the corresponding background before grayscale conversion (left), after grayscale conversion (center), and the resulting subtracted image (right).

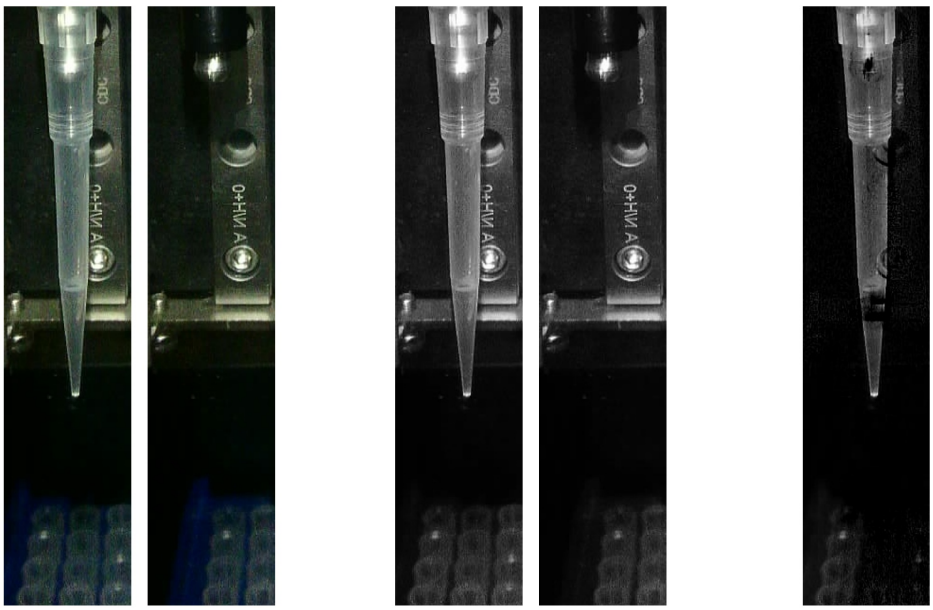


Figure 4.3: Example of pre-processing up to the subtraction step for a short tip. The figure displays the scene image and the corresponding background before grayscale conversion (left), after grayscale conversion (center), and the resulting subtracted image (right).

Figure 4.4 shows the subtracted image obtained after resizing to 256×256 for the long tip (left) and the short tip (right).

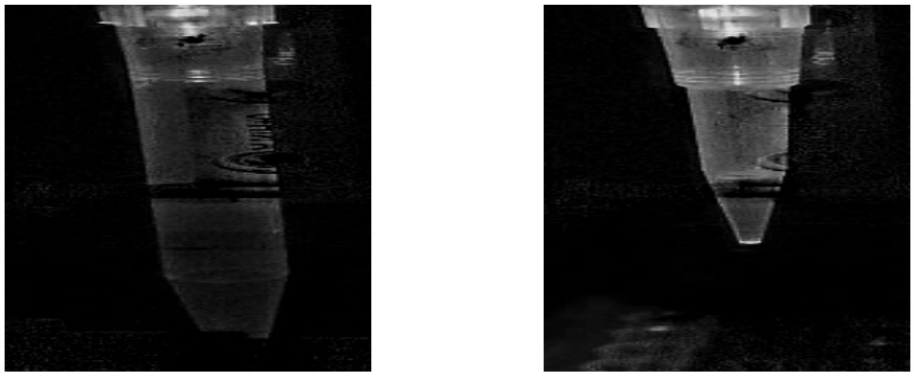


Figure 4.4: Example of subtracted image after resizing to 256×256 pixels for the long tip (left) and the short tip (right).

4.3 SVM approach

4.3.1 Feature extraction and pre-processing

As previously discussed in this Chapter, the classification of the two tip types was approached by implementing multiple SVM models, all trained on a common set of features. Since the task involves distinguishing objects based on shape, we focused exclusively on shape-based features. Specifically, we employed Fourier descriptors, which require the extraction of the object's contour [56]. For this reason, an additional pre-processing pipeline was implemented, complementing the general steps described earlier. Fourier descriptors were selected because they provide a compact and robust representation of the contour, while being invariant to translation and rotation [57]. Unlike single-dimensional measurements, they capture the overall geometry of the tip, ensuring reliable classification even when the objects appear with slight variations in position or orientation. This choice improves the robustness of the model in real operating conditions.
The pre-processing procedure for contour extraction consists of the following steps. First, binarization is performed using Otsu's method, with a minimum threshold set to 128 and a maximum to 255 [48]. This is followed by a morphological closing operation, employing a structural element with dimensions of $(5, 5)$. Finally, we applied the *findContours* [3] algorithm of OpenCV to detect all closed contours in the image. This function rank all detected contours in descending order of area. Since the image may contain irrelevant contours (i.e., noise), we selected only the four largest contours, as they were deemed sufficient to accurately represent the object's shape. An example of the pre-processing pipeline applied to both long tip and short tip, is shown in Figures 4.5 and 4.6, respectively. After pre-processing, the resulting image, referred to as the *contour image*, is used to compute the Fourier descriptors. To this end, we selected the first 8 Fourier descriptors as features. This choice was motivated by the fact that when reconstructing the contour image using only these 8 descriptors, the resulting image was visually very similar to the original one. Figure 4.7 shows an example of a contour image obtained after pre-processing (left) and its corresponding reconstruction using the first 8 Fourier descriptors (right) for a long tip. As observed, the two images appear very similar, and the tip shape is well preserved.
While Figure 4.8 presents the same comparison for a short tip, displaying the contour image (left)

[3] `https://docs.opencv.org/4.x/d3/dc0/group__imgproc__shape.html`

Figure 4.5: Example of the pre-processing pipeline applied to a long tip, showing the resized image after binarization (left), closing operation (center), and contour extraction (right).

Figure 4.6: Example of the pre-processing pipeline applied to a short tip, showing the resized image after binarization (left), closing operation (center), and contour extraction (right).

Figure 4.7: Example of a contour image obtained after pre-processing (left) and its corresponding reconstruction using the first 8 Fourier descriptors (right) for a long tip.

and the corresponding reconstruction (right). In this case as well, the two images are visually similar, and although the reconstruction of the short tip retains slightly less detail than that of the long tip, its overall shape remains clearly distinguishable.

Figure 4.8: Example of a contour image obtained after pre-processing (left) and its corresponding reconstruction using the first 8 Fourier descriptors (right) for a short tip.

4.3.2 Feature extraction visualization

As in the presence check, to better understand the relationships between the extracted features and their ability to distinguish between long tips and short tips, we generated a pairwise scatter plot, shown in Figure 4.9. This figure visualizes the distribution of the 8 extracted features, displaying their pairwise combinations. Each point represents a sample, where yellow dots correspond to long tips and purple dots to short tips. From this scatter plot, we can observe that the data distribution suggests a non-linear separability between the two classes. This indicates that a SVM model with linear kernel would not be sufficient for effective classification. Instead, the use of non-linear kernels, such as RBF and HI, is required to capture the complex relationships between features and improve classification performance. To complement this visualization, Table 4.1 reports, for each feature, the

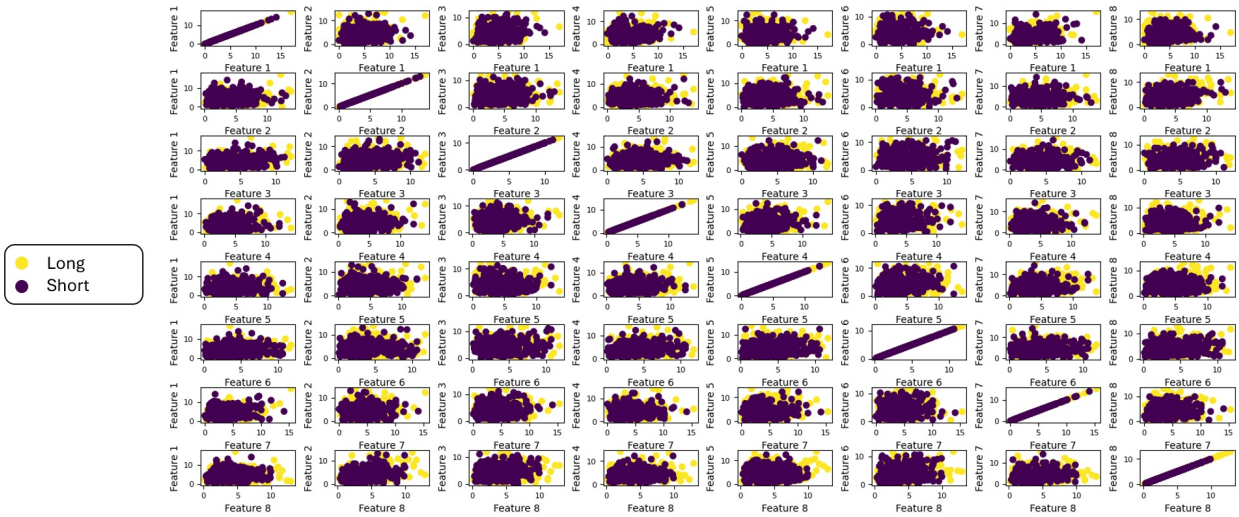


Figure 4.9: Distribution of the 8 features considered based on the pairwise combination of values.

mean and standard deviation computed across the dataset for the two classes (long vs short tip). These summary statistics provide a stable, dataset-level comparison and replace the illustrative single-image examples.

Output	FD 1	FD 2	FD 3	FD 4	FD 5	FD 6	FD 7	FD 8
Long tip	4.848 ± 2.659	4.727 ± 2.497	4.820 ± 2.489	4.590 ± 2.507	4.678 ± 2.381	4.959 ± 2.668	4.862 ± 2.603	4.793 ± 2.737
Short tip	4.804 ± 2.668	4.752 ± 2.261	4.556 ± 2.474	4.753 ± 2.436	4.798 ± 2.509	4.505 ± 2.594	4.755 ± 2.244	4.835 ± 2.464

Table 4.1: Mean and corresponding standard deviation of each type check feature, reported separately for the two classes.

4.3.3 Models

As discussed earlier in this Chapter, we implemented a set of SVM models to perform the classification task of tip type discrimination. The same set of features extracted from the dataset was used for all SVM models to ensure consistency in evaluation. Additionally, we employed a C-SVC model. Based on the observed feature distribution shown in Figure 4.9, we trained SVM models with non-linear kernels, specifically RBF and HI, to effectively model the complex relationships between features. Unlike in the presence check, we did not test a linear SVM kernel, as the feature distribution clearly suggested non-linearity, making a linear classifier unsuitable [58].
To determine the optimal C and γ values, we followed the same grid search approach used in the presence check, optimizing model performance through cross-validation. These values were chosen based on commonly used settings in the literature and were directly configured in the C++ training implementation. In this process, the explored values of C and γ spanned the interval $[10^{-5}, 10^{7}]$. The selected C and γ parameters for both models are reported in Table 4.2.

Kernel	C value	γ value
RBF	$5 \cdot 10^{-1}$	$3.375 \cdot 10^{-2}$
HI	$1 \cdot 10^{-1}$	N/A

Table 4.2: C and γ parameters for each model after grid search optimization.

To ensure model robustness and generalization, we trained all models using a simple k-fold cross-validation, setting $k = 5$. The choice of $k = 5$, rather than $k = 10$ as in the presence check SVM models, was made to ensure comparability with CNNs, where $k = 5$ is commonly used for cross-validation. This is because CNN models typically require larger validation sets to better assess generalization, making a lower k value more appropriate. After performing 5-fold cross-validation, the best-performing model was selected for further evaluation. In the subsequent analyses, accuracy and F1-score are treated as co-primary metrics, as non-stratified k-fold may induce fold-level variations in class proportions; precision and recall are also reported to characterise the error profile. A detailed discussion of the mean accuracy and performance metrics for each kernel will be provided in Section 4.5.

4.4 CNN approach

Given the performance obtained with the SVM models, which will be presented in later sections, we decided to implement a convolutional neural network as an alternative approach. The entire implementation, including model architecture design, training, and validation, was performed using the TensorFlow and Keras libraries in Python.
For this task, we designed a custom CNN architecture rather than adopting a pre-trained model. The number of convolutional layers, fully connected layers, and the training parameters were chosen

on the basis of a thorough review of the literature and best practices reported in state-of-the-art studies. Additionally, we conducted empirical experiments to optimize the architecture and hyperparameters, adapting them to our specific binary classification task. For brevity, we present only the final model configuration, which achieved the best performance.
In the following sections, we will describe the data augmentation techniques applied. Additionally, we will introduce the model architecture, outlining the rationale behind our design choices. Finally, we will present the selected training parameters and explain the reasoning behind their selection.

4.4.1 Data augmentation

The dataset used for the CNN approach is the same as that used for the SVM approach (see Section 4.1). The images were first subjected to the pre-processing steps described in Section 4.2, resulting in a dataset of 644 subtracted images, each resized to $256 \times 256 \times 1$.
Given the limited size of our dataset, we utilized the data augmentation technique, which is essential for preventing overfitting when data availability is restricted. Deep learning models, such as CNNs, typically require large datasets (exceeding thousands of samples) to avoid overfitting. Without sufficient data, the model may memorize training examples instead of learning meaningful general patterns, leading to poor generalization on unseen data [83].
To address this issue, we applied two data augmentation techniques [4] separately:

- *Flip left-right.* This transformation reflects the image across an imaginary vertical axis passing through the image center, simulating possible variations of the tip during acquisition;
- *RandomRotation.* This transformation applies small rotations to the images, with angles randomly selected from a range of -3° to 3°.

Figures of the long and the short tip, as well as the images resulting after *Flip left-right* and *RandomRotation* transformations is shown in Figure 4.10.

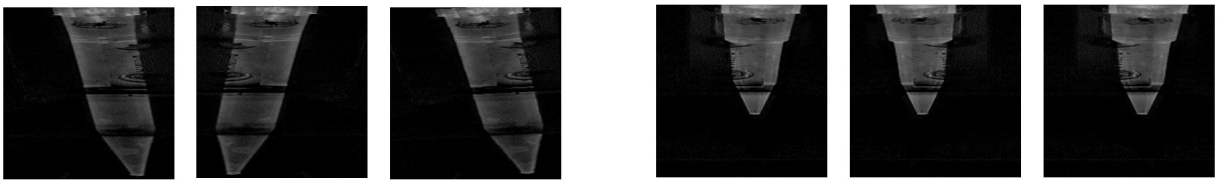


Figure 4.10: Subtracted images resized to 256×256 pixels. **Left:** 256×256 subtracted image depicting the long tip (left), after Flip left-right transformation (center), and after Rotation between -3° and 3° (right). **Right:** 256×256 subtracted image depicting the short tip (left), after Flip left-right transformation (center), and after Rotation between -3° and 3° (right) [1].

The choice of these two transformations was guided by empirical observations. Since the tips can be slightly misaligned due to the way the pipetting system picks them up, applying small rotations helps the model generalize better to real-world scenarios.
Moreover, we deliberately avoided transformations related to brightness, contrast, or other color-based alterations. As explained in the Section 2.4.2 all acquisition conditions are carefully controlled and standardized. Applying augmentations that introduce unrealistic variations could lead the model to learn irrelevant patterns that do not reflect real working conditions.

[4] `https://keras.io/api/layers/preprocessing_layers/image_augmentation/`

Notably, data augmentation was applied only during the training and not during validation. The reason is that we want to increase our model's generalization performance by adding more data and diversifying the training dataset. If augmentation were applied to validation images, it could artificially overestimate accuracy, leading to misleading performance evaluations [84].

4.4.2 CNN architecture

The architecture of the network is illustrated in Figure 4.11 [1]. The network can be divided into

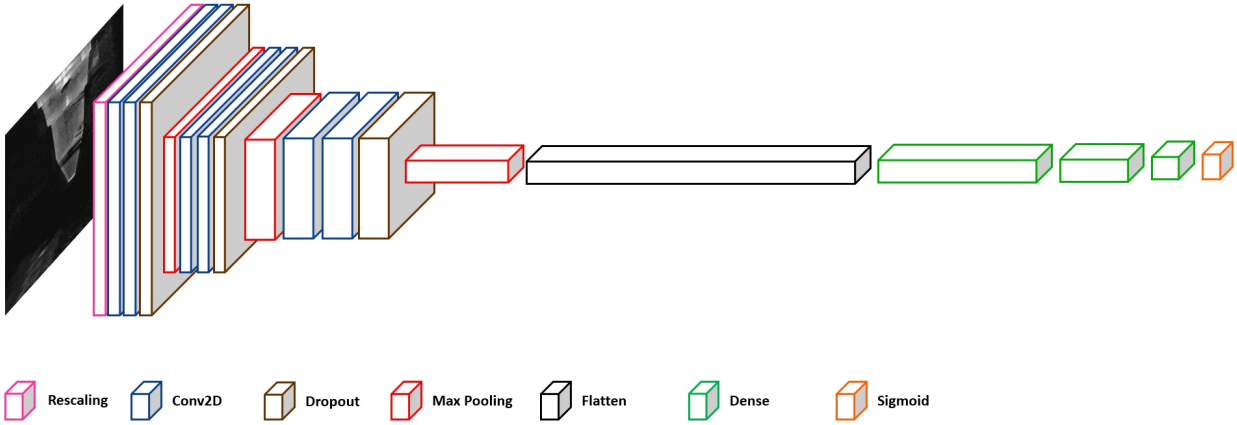


Figure 4.11: Architecture of the convolutional neural network.

two main blocks: the *feature extraction block*, responsible for extracting relevant patterns from input images, and the *classification block*, which processes the extracted features and assigns a class label [28].

Feature extraction block

The feature extraction block consists of the following type of layers:

- *Rescaling layer.* Normalizing images from the range $[0, 255]$ to $[0, 1]$ ensures consistency across inputs. Variability in pixel intensity ranges could cause the model to assign different importance to features based on their scale rather than their relevance. Furthermore, normalization helps maintain the weights within a controlled range, improving model stability and preventing saturation during training;

- *Convolutional 2D layer.* We set a kernel of 3×3. This choice is based on findings from DL literature, where 3×3 filters have become a standard due to their efficiency in extracting fine details while maintaining computational feasibility;

- *Dropout layer.* We decided to include this layer to prevent overfitting by randomly deactivating a fraction of neurons during training. We set dropout rate of 0.1. This value was determined empirically to strike a balance between regularization and model capacity;

- *Max pooling.* This layer was included to reduce the spatial dimensions of the feature maps, thereby decreasing the number of parameters and computational cost.

Classification block

The classification block combines all extracted features to make a final decision. It consists of the following type of layers:

- *Flatten layer*. This layer converts the multi-dimensional feature map from the last convolutional layer into a one-dimensional vector, ensuring compatibility with fully connected layers;
- *Fully connected layers*. These layers perform the actual classification based on the extracted features. The final dense layer, responsible for classification, utilizes the sigmoid activation function, which outputs a probability score for each class. Since this is a binary classification task, the sigmoid function is the most appropriate choice, as it maps the output to a range between 0 and 1. The output of the sigmoid function can be interpreted as the probability of the input image belonging to a specific class.

4.4.3 Architecture parameters

The Table 4.3 summarizes the key parameters of each layer in the CNN, including input and output dimensions. The input to the network consists of $256 \times 256 \times 1$ grayscale images obtained from the

Layer	# of input channels	# of output channels	Kernel size	Stride	Input size	Output size
Conv2D	1	8	3	1	256×256×1	256×256×8
Dropout (rate = 0.1)	-	-	-	-	256×256×8	256×256×8
MaxPool2D	8	8	-	2	256×256×8	128×128×8
Conv2D	8	16	3	1	128×128×8	128×128×16
Dropout (rate = 0.1)	-	-	-	-	256×256×8	256×256×8
MaxPool2D	16	16	-	2	128×128×16	64×64×16
Conv2D	16	32	3	1	64×64×16	64×64×32
Dropout (rate = 0.1)	-	-	-	-	256×256×8	256×256×8
MaxPool2D	32	32	-	2	64×64×32	32×32×32
Flatten	-	-	-	-	32×32×32	32768
FC	-	-	-	-	32768	50
FC	-	-	-	-	50	20
FC	-	-	-	-	20	10
FC	-	-	-	-	10	1

Table 4.3: Parameters of the type check CNN model for each layer.

pre-processing stage. Before being fed into the model, these images are converted into tensors, as required by the TensorFlow framework.
Observing the convolutional layers, we note that as we move deeper into the network, the number of channels increases, while the spatial dimensions of the feature maps decrease. Since we used `padding="same"` in all convolutional layers, the spatial dimensions of the feature maps remain unchanged after each convolution operation. This ensures that the extracted features retain fine details throughout the network, preventing excessive loss of information at early stages of processing. This behavior follows a fundamental principle in convolutional architectures whose formulas are set here:

$$H_{\text{out}} = \frac{H_{\text{in}} + 2P - K}{S} + 1, \quad W_{\text{out}} = \frac{W_{\text{in}} + 2P - K}{S} + 1 \tag{4.1}$$

where H_{in}, W_{in} are the height and width of the input feature map, K is the kernel size, P is the padding applied to both sides of the input (for `padding="same"` in Keras, $P = \frac{K-1}{2}$, and S is the stride of the convolution) [85].
For Max pooling layers, no padding was applied, meaning that pooling operations reduce the spatial dimensions without adding extra pixels at the borders. This progressive reduction in spatial dimensions is primarily due to the Max pooling layers, which halve the height and width of the feature maps, reducing computational complexity and focusing on the most prominent patterns. Simultaneously, the number of channels increases as deeper layers extract more complex features: while initial layers capture basic edges and textures, deeper layers learn more abstract representations, such as object parts and high-level structures. To reduce the risk of overfitting, dropout layers were added after each convolutional layer. A dropout rate of 0.1 was set, meaning that 10% of the neurons were randomly deactivated during training. This choice, made empirically, helps prevent the network from relying too much on specific features, promoting better generalization [27].
In contrast, within the fully connected layers, we observe the opposite trend: the number of neurons progressively decreases. This compression is designed to condense the extracted features into the most relevant information for classification. The final layer consists of a single neuron, as the task is a binary classification problem [28]. A sigmoid activation function is applied to this output neuron, ensuring that the output is a probability value in the range [0,1], interpreted as follows:

$$\hat{y} = \begin{cases} \text{Short tip}, & \text{if } \hat{y} \geq 0.5 \\ \text{Long tip}, & \text{otherwise} \end{cases} \tag{4.2}$$

This decision threshold enables the network to classify each input image as either a short or long tip.

4.4.4 Training parameters

The custom CNN model was trained 5 times, each time with a different initialization seed. Employing multiple replications helps mitigate the risk of overestimating the model's performance due to fortuitous circumstances.
All training and validation parameters were empirically selected and are summarized in Table 4.4.

Parameter	Value
# of replications	5
# of epochs	25
Batch size	16
Learning rate	10^{-3}
Weight decay	10^{-4}
Pooling size	2

Table 4.4: Parameters of the type check CNN model for training.

We trained the model for 25 epochs, applying early stopping to prevent overfitting. Specifically, we monitored the validation loss starting from the 18th epoch, and the training was stopped if the loss did not improve within a predefined tolerance range, empirically determined. The pooling size was set equal 2, value often used in the literature because excellent compromise between decrease

of the size of feature maps and data loss [28]. The batch size was chosen to balance computational efficiency, model performance, and convergence speed [86]. For optimization, we used the Adam optimizer with weight decay, which mitigates the effects of noisy gradients, particularly in scenarios with small batch sizes such as in our case. As with the SVM models, we applied a simple k-fold cross-validation to further reduce overfitting and improve generalization without enforcing stratification. We set $k = 5$. This choice was driven by the need for larger validation sets in CNNs to better assess generalization. After performing the 5-fold cross-validation, the model's performance was evaluated using Binary Cross-Entropy loss, accuracy, and F1-score. Although the dataset as a whole was balanced, the absence of stratification may introduce slight class imbalances within individual folds; therefore, the F1-score was included to provide a more robust evaluation by jointly accounting for precision and recall [87].

4.5 Results

4.5.1 SVM approach

Evaluation on training and validation set

We trained the two SVM models using the same set of features extracted from the dataset, and employed k-fold cross-validation with $k = 5$. This means that each model was trained on 480 samples and evaluated on 120 across five folds, ensuring that each fold served once as the validation set. Table 4.5 shows the accuracies achieved on the validation fold for the two models, and Table 4.6 presents the mean accuracies along with the corresponding standard deviations.

Kernel	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
RBF	51.16%	60.47%	50.39%	55.81%	50.78%
HI	57.36%	52.71%	48.06%	54.26%	47.66%

Table 4.5: Accuracies on the validation fold for each model type.

Kernel	Mean Accuracy
RBF	53.72% ± 3.9%
HI	52.01% ± 3.7%

Table 4.6: Mean accuracies along with the corresponding standard deviations for all models.

As shown, both models perform poorly in distinguishing between long tips and short tips. Indeed, the average accuracies are close to 50%, which is comparable to random behaviour. By selecting the best model for each kernel type, Table 4.7 reports the performance for each model. All metrics in the Table are expressed as a percentages, expect for the AUC and the number of support vectors. Figures 4.12 and 4.13 show the confusion matrix (right) and the ROC curve (left) of RBF SVM and

Kernel	Accuracy	Precision	Recall	F1-score	AUC	Support Vectors
RBF	60.47%	62.90%	58.21%	60.44%	0.606	393
HI	57.36%	67.92%	48.65%	56.70%	0.589	454

Table 4.7: Performance metrics for each model kernel type.

HI SVM for validation fold, respectively.

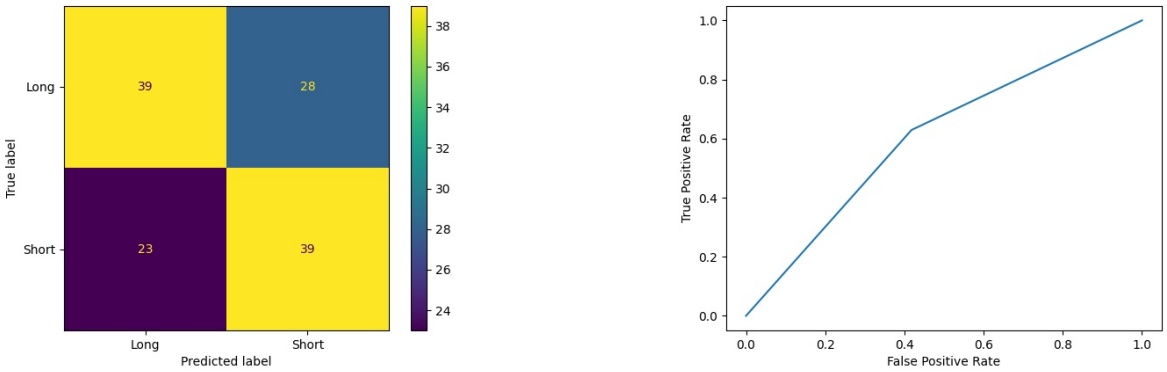


Figure 4.12: Confusion matrix (left) and the ROC curve (right) of the best RBF SVM on validation fold.

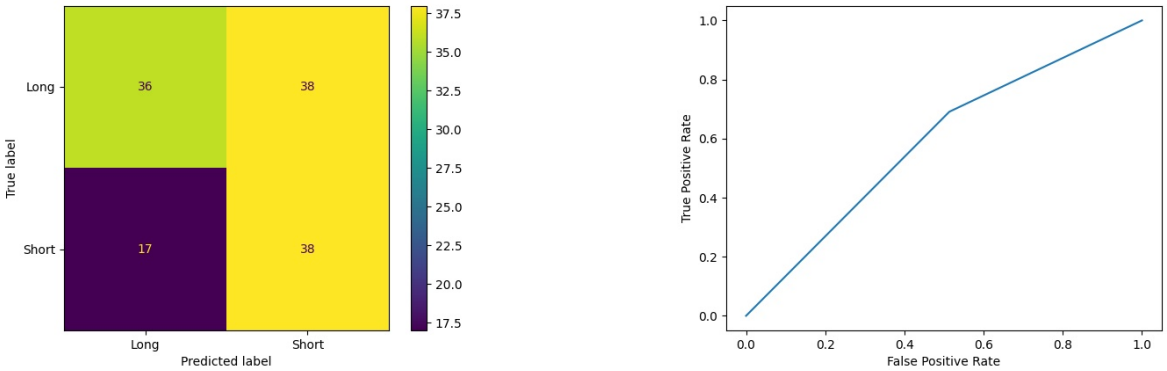


Figure 4.13: Confusion matrix (left) and the ROC curve (right) of the best HI SVM on validation fold.

Analysing the metrics, we can observe that the RBF SVM has better performance in terms of accuracy, recall, F1-score and AUC than the HI SVM model. Specifically, the higher accuracy and AUC indicates a greater ability of the model to discriminate between the two classes. While, the higher recall suggests a greater ability of the RBF SVM to identify long tips. In contrast, the HI SVM achieves a higher precision, meaning it is more accurate when predicting long tips. However, its lower recall indicates that the model has a tendency to omit a greater number of actual long tips. In addition, the RBF model shows better trade-off between precision and recall, as confirmed by the higher F1-score.

Regarding model complexity, for both models use very high number of support vectors, 393 for RBF and 454 for HI, which are close to the training size. This suggests a high degree of boundary complexity and a high risk of overfitting; however, the number of support vectors is not, by itself, decisive for generalization capability, which is therefore evaluated on a set of external data not used for training or validation. Finally, from an operational perspective, classification errors such as predicting a long tip as short, or vice versa, do not compromise the integrity of the diagnostic process. In both cases, the monitoring system flags the mismatch, discards the misclassified tip, and repeats the gripping operation for the affected channel. The direct impact of these errors is therefore limited to an increase in execution time and a higher consumption of tips, while the accuracy and

reliability of the diagnostic outcome remain unaffected.

Evaluation on an external test set

To assess the generalizability and robustness of both models, we decided to test them on an external test set. This dataset consists of 216 images, 108 depicting long tips and 108 short tips. The images were acquired in the same way as the external test set described for the presence check (Section 3.4). Figure 4.14 shows the confusion matrices obtained on the dataset of the RBF model (left) and the HI model (right). The two models achieve accuracies of 50.93% and 58.8%, respectively. These results confirm that both models do not generalize and are not effective for the classification task, with the RBF model showing performance close to random. When considering the F1-score, which

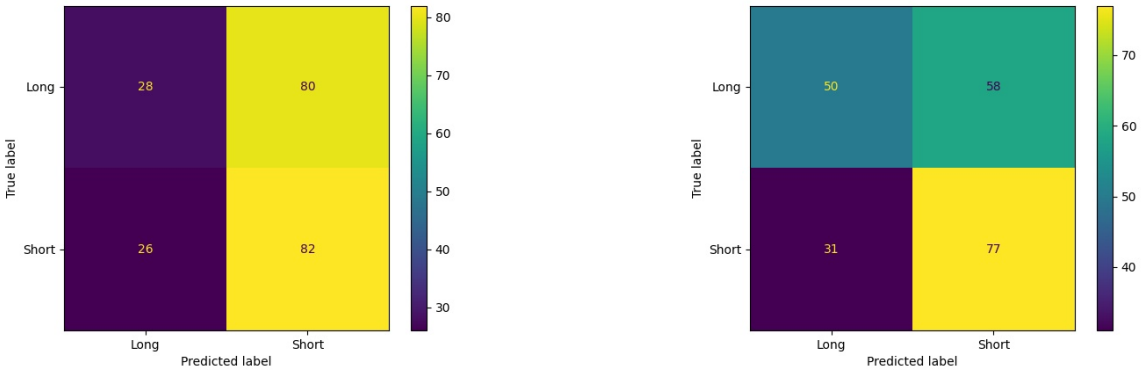


Figure 4.14: Confusion matrices obtained by the RBF model (left) and the HI model (right) on external test set.

better balances class-wise performance, the results are essentially comparable to accuracy, with values of 50.9% for the RBF model and 58.6% for the HI model. These results confirm that both SVM models fail to generalize and are not effective for the type classification task. While Fourier descriptors offer a more advanced representation than simple one-dimensional features, their limited discriminative power in this context indicates that they are not sufficient to capture the subtle geometric variations of the tips. In particular, although they provide invariance to translation and rotation, the resulting feature space does not fully represent the structural complexity needed for reliable classification. From an operational standpoint, the misclassifications produced by these models would not compromise the diagnostic process, as the monitoring system automatically detects and corrects size mismatches, although at the expense of execution time and tip consumption. This limitation highlights the need for more expressive approaches, such as convolutional neural networks.

4.5.2 CNN approach

Evaluation on training and validation set

The CNN model for type check was trained on 1545 images and evaluated on the remaining 129 images. The model was trained using 5-fold cross-validation, and loss function binary cross-entropy was used as the performance measure. Figures 4.15 and 4.16 show the average loss and accuracy curves during the training and validation phases, respectively. These curves represent the average

trend across all replications (using different seeds) for each fold. Specifically, Figure 4.15 shows the average loss during training (left) and validation (right) across all replications and, Figure 4.16 displays the average accuracy during training (left) and validation (right) across all replications for each fold.

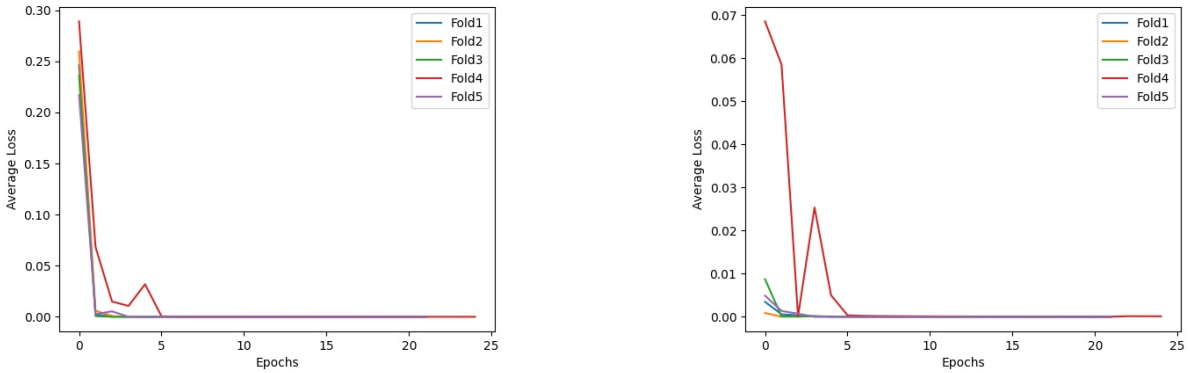


Figure 4.15: Average loss curves during training (left) and validation (right) across all replications for each fold. Each curve is obtained by averaging loss curves of each replication across epochs [1].

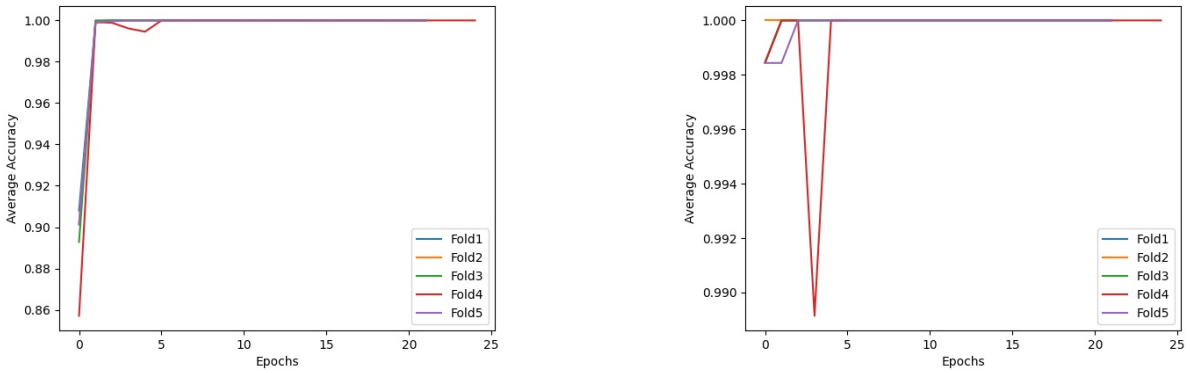


Figure 4.16: Average accuracies during training (left) and validation (right) across all replications for each fold. Each curve is obtained by averaging loss curves of each replication across epochs [1].

Both figures show an excellent convergence of the model. In detail, in Figure 4.15, the loss curves during training decreases until they tend to zero for all folds, expect for fold 4, where a peak is observed around the third/fourth epoch. This peak is due a sudden increase of the loss in one of the replications, likely related to a particular batch of images. A similar trend can be seen during the validation phase. Importantly, this behaviour was detected in only one replication out of the five and did not reappear in the others, suggesting that it is not related to systematic dataset artifacts but more likely to the specific composition of a single batch that temporarily altered the learning dynamics. Moreover, in Figure 4.16, accuracy curves for both training and validation phase increase and stabilize at 100% around the second/third epoch for all folds, except for fold 4, which exhibits a peak around the third/fourth epoch. As with the loss curves, this peak can be related to a particular batch of images during one of the replications.
To better understand the model's overall performance, we calculated the average accuracy achieved at the end of validation for each of the five replications. Table 4.8 summarizes these results. The

last column represents the average accuracy obtained by averaging across all folds.

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean Accuracy
100%	100%	100%	100%	100%	100% ± 0%

Table 4.8: Average accuracies for each fold and overall mean accuracy.

By selecting the best model only, Figure 4.17 shows the confusion matrix (left) and the ROC curve (right) obtained on the validation set. The CNN achieved 100% accuracy, an AUC equal to 1, and an F1-score of 100%, meaning it classified all 129 images correctly. This indicates that our model is a perfect classifier and well-suited for the type classification task.

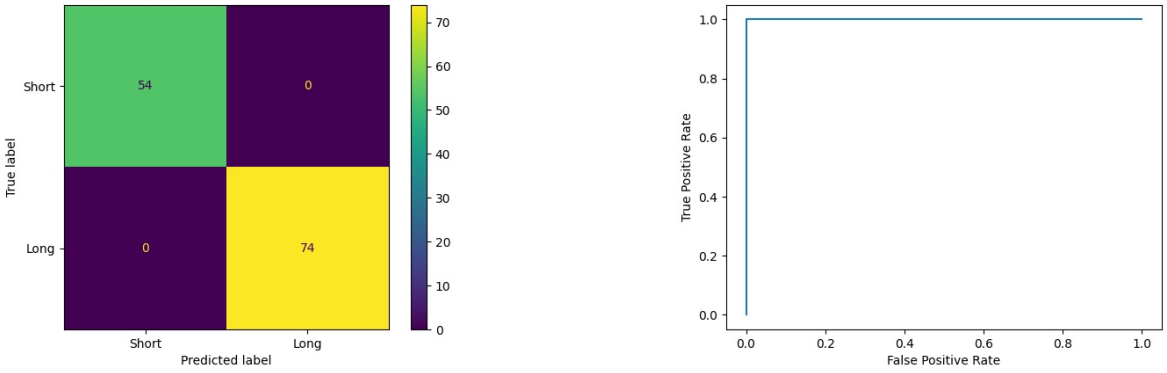


Figure 4.17: Confusion matrix (left) and ROC curve (right) of the best CNN model on the validation set [1].

Explainability

In the domain of DL, it is essential not only develop a model capable of achieving excellent performance but also to understand the mechanisms behind its predictions. To address this, we performed a feature analysis using two different methods, the IGs[5] and Grad-CAM[6]. These methods help understand which parts of the image have contributed most to the model's predictions. Figures 4.18, 4.19 and 4.20 show for both a long tip (left) and short tip (right), the reference images from the dataset, the feature map after calculating IG, and the heatmap after calculating Grad-CAM, respectively.

Integrated gradients values can be either positive or negative. Positive values indicate that a region contributes positively to the model's prediction, while negative values suggest a negative contribution. Values close to zero indicate little or no influence on the prediction. In contrast, Grad-CAM values range from 0 to 1, where a value of 0 indicates no attribution by the model, meaning that the corresponding part of the image does not influence the prediction. Values closer to 1 denote a strong influence from that area.

Upon observing the IG feature map of the long tip, we note that our model assigns negative attribution to the tip point and to the edges, and positive attribution to the tip base and center part of the tip. In addition the model assigns no attribution to the background. Similarly, for the short tip,

[5] `https://www.tensorflow.org/tutorials/interpretability/integrated_gradients`
[6] `https://keras.io/examples/vision/grad_cam/`

Figure 4.18: Dataset reference images for XAI computation of a long tip (left) and short tip (right) [1].

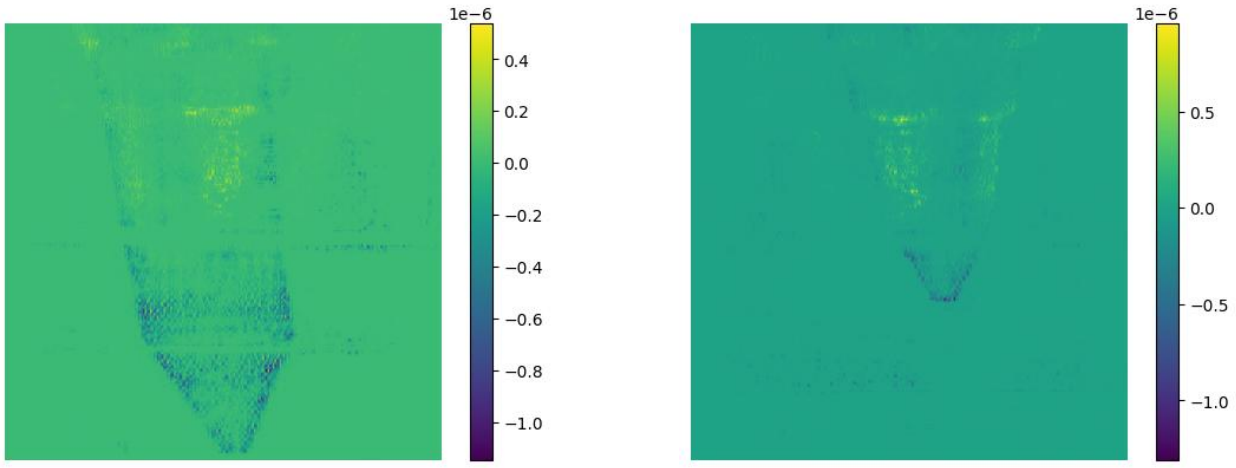


Figure 4.19: Feature map after integrated gradients computation of long tip (left) and short tip (right) [1].

the model gives negative attribution to the tip point, positive attribution to the base and center, and no attribution to the background.
Analysing the Grad-CAM heatmap of the long tip, it is evident that the model focuses primarily on the tip, with the background receiving a value close to zero. The color gradient, from blue (low) to yellow (high), indicates that the model's focus is predominantly on the tip. A similar pattern is observed for the short tip, confirming that the model is mainly influenced by the tip's features, particularly the base and edges.
This behavior is consistent across the entire training and validation datasets, suggesting that the mechanism behind the model's predictions is reliable and focused on the relevant areas of the image.

Evaluation on an external test set

As with the SVM approach, we assessed robustness and generalization by evaluating the CNN on the same external test set: 216 images (108 long, 108 short), acquired following the same procedure described for the external test set of the presence check (Section 3.4). Figure 4.21 shows the confusion matrix obtained on this dataset. The CNN model achieved 100% accuracy and an F1-score of 100%, confirming that it correctly classified all images. These results demonstrate that, unlike the

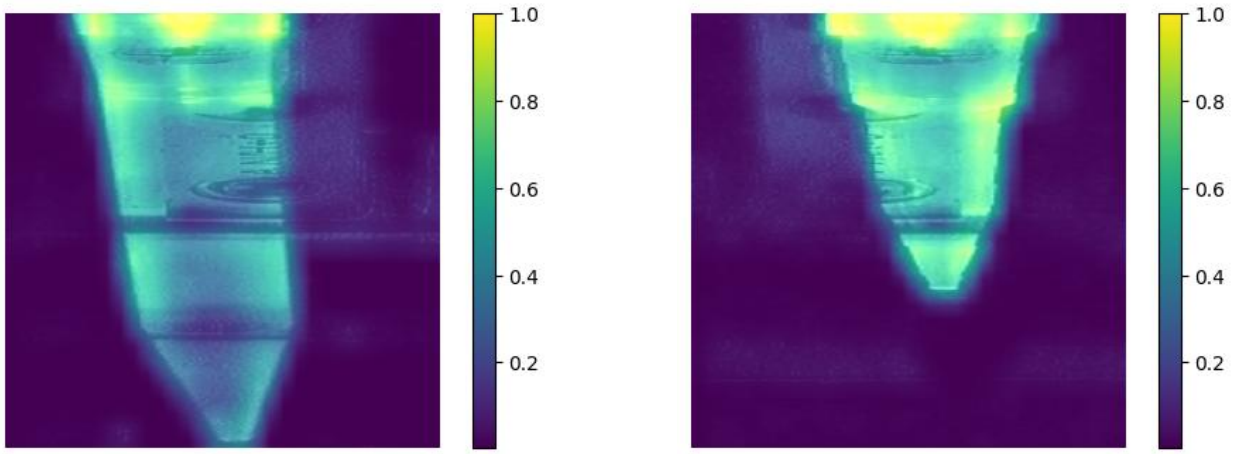


Figure 4.20: Heatmap after Grad-CAM computation of long tip (left) and short tip (right) [1].

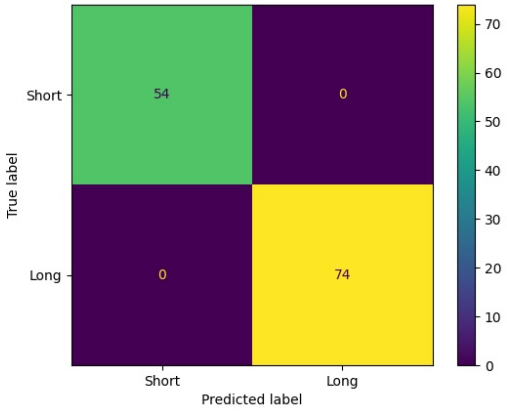


Figure 4.21: Confusion matrix on test set of CNN model.

SVM models, the CNN retained perfect discriminative power even on external data, further supporting its suitability for the type classification task. In addition, we performed an explainability analysis to assess how well the results obtained on the test set align with those from the training set. Figures 4.22, 4.23 and 4.24 show the images from the test set (left: long tip, right: short tip), the feature map after IG computation, and the Grad-CAM heatmap, respectively.
Upon analysing the images, we find a consistency between the results obtained on the training dataset and those obtained on the test dataset. The IG maps indicate that the model is again not influenced by the background, and the attribution values are similar in magnitude and localization. The same pattern is observed in the Grad-CAM heatmaps, confirming that the model's interpretability is consistent across both datasets.

Time performance

Finally, we evaluated our CNN model measuring the time taken to execute pre-processing and classification. Table 4.9 shows the average pre-processing, prediction and total times, along with the corresponding standard deviations, for the selected model. The average was calculated over the 216 images in the test set, and the measurements were conducted on a system with hardware

Figure 4.22: Test set reference images for XAI computation of a long tip (left) and short tip (right) [1].

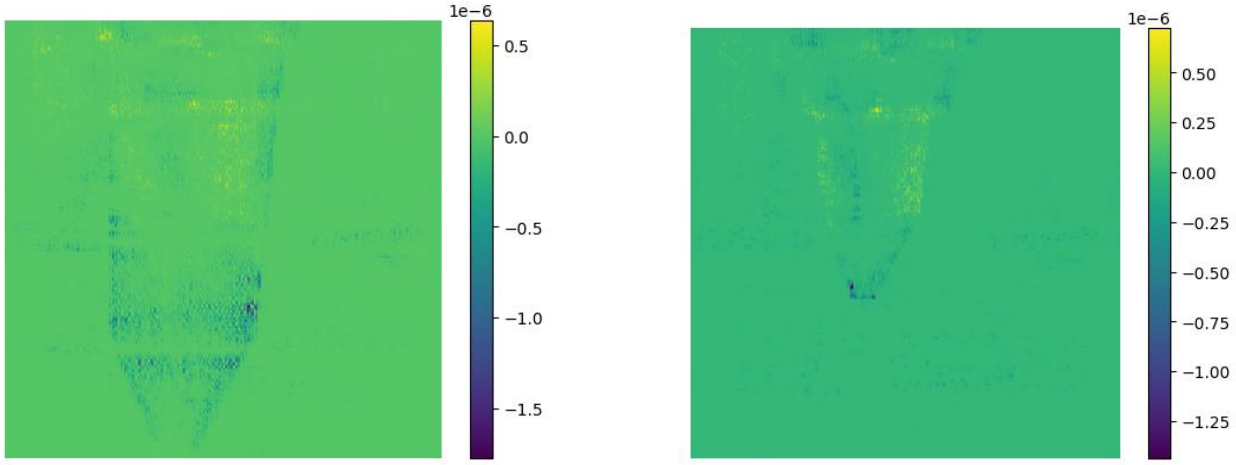


Figure 4.23: Feature map after integrated gradients computation of long tip (left) and short tip (right) [1].

identical to the one integrated in the IVD device.

Function	Time
Pre-processing	$3.179 \cdot 10^{-3} \pm 4.407 \cdot 10^{-3}$ s
Prediction	$8.411 \cdot 10^{-2} \pm 4.473 \cdot 10^{-2}$ s
Total	$8.73 \cdot 10^{-2} \pm 4.559 \cdot 10^{-2}$ s

Table 4.9: Average extraction, prediction, and total times with standard deviations for the selected model.

These results demonstrate the total time employed to execute the type check is consistent with time required for this step, as it remains below the limit of 1 second.

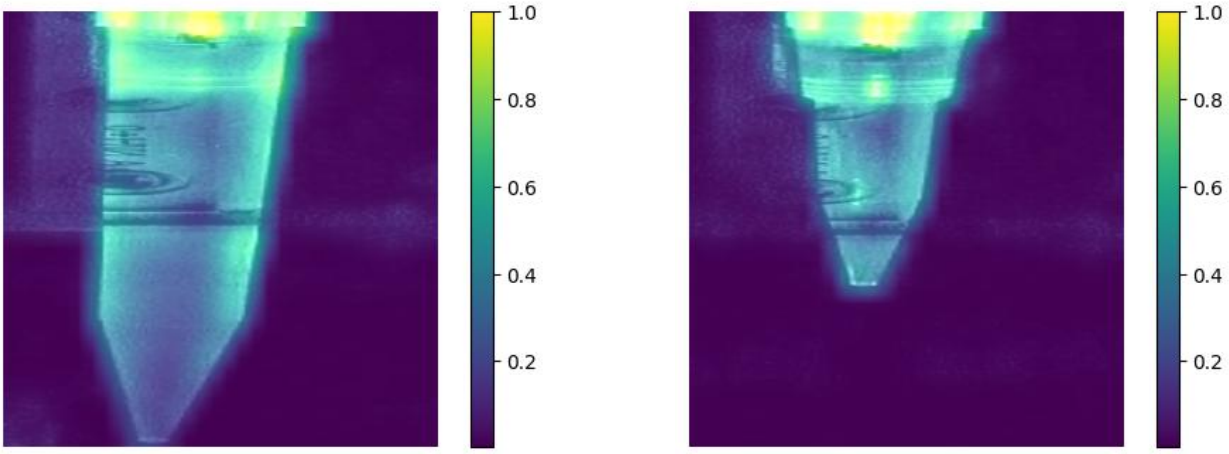


Figure 4.24: Heatmap after Grad-CAM computation of long tip (left) and short tip (right) [1].

Chapter 5

Volume check

The volume check is the third and final control performed by our real-time monitoring system. As discussed in Section 2.2, the IVD device in which the MVS is integrated already includes mechanisms to mitigate this issue, such as a pressure sensor and a flowmeter. However, these systems are insufficient, as they only verify the liquid volume during aspiration. In the dispensing phase, the system only checks whether the tip is clogged, without providing a quantitative evaluation of the dispensed volume. To address these limitations, the volume check performs a continuous monitoring of the liquid quantity inside the tip. Specifically, our MVS analyses the liquid volume at four key time points: before and after aspiration, and before and after dispensing.
Since short tips and long tips handle different liquid volumes, we decided to train separate models for each tip type. Indeed, we decide to acquire two distinct datasets, one for short tips and one for long tips. Each dataset was processed independently, and each model was trained and tested separately to ensure optimal performance in both cases.
Given the complexity of this task, we decided to implement a CNN. Additionally, to compare the CNN with an alternative deep learning approach, we implemented a ViT, a more recent model that has gained significant attention in deep learning field. Both models were trained in Python, but using different frameworks. The CNN was implemented, trained and evaluated with Tensorflow/Keras[1], while the ViT was implemented, trained and evaluated with Pytorch[2], a framework for building and deploying AI algorithms [88]. As in the case of the CNN used for the type check, also here both deep learning models were trained offline in Python and then the chosen one deployed within the monitoring system exclusively for inference. In this way, the demanding training phase is carried out externally, while the deployed models operate efficiently in real time without interfering with the workflow of the device.
The following sections describe in detail the methodologies adopted for both short tip and long tip scenarios. In detail we show the two datasets used for training and evaluation and the pre-processing steps applied to the data. We then describe the CNN and ViT models, including their architecture, training strategy, and data augmentation techniques. Finally, we evaluate the performance of the models in relation to the real-time constraints and provide a comparative analysis of the different approaches adopted.

[1] `https://www.tensorflow.org`
[2] `https://pytorch.org`

5.1 Dataset

To perform the volume check at the four key time points, we built a dataset of images representing short and long tips containing different liquid volumes. Given the different capacities of short and long tips, we created two distinct datasets, one for each tip type. However, both datasets were acquired under the same conditions, ensuring consistency in lighting, camera parameters, and acquisition setup.
For each tip type, we selected the standard volumes handled by the IVD device during normal operations. Specifically, for short tips, which have 200 μL max capacity, the selected volumes are summarized in Table 5.1.

Labels				
0 μL	50 μL	100 μL	150 μL	200 μL

Table 5.1: Selected volumes for short volume check.

For long tips, which have 1000 μL max capacity, the selected volumes are summarized in Table 5.2.

Labels					
0 μL	200 μL	400 μL	600 μL	800 μL	1000 μL

Table 5.2: Selected volumes for long volume check.

The two datasets were collected using the monitoring system integrated into the IVD device. A total of 1189 images for short tips and 826 images for long tips were acquired. The difference in dataset sizes is explained by the fact that, for long tips, an additional volume class was initially included. This class was later excluded after consultation with the MACS's engineers, since it did not correspond to a standard operational volume. As a result, the number of images for long tips is lower than that for short tips, while still ensuring consistency across both datasets. Since the pipetting system picks up two tips simultaneously, we extracted 2378 ROIs for short tips and 1652 for long tips. For clarity, each ROI will be hereafter referred to as an *input image*. The input image dimensions were defined to ensure that the entire tip was fully visible while minimizing unnecessary background elements, which could interfere with the training process. In particular, input images with short tips were resized to a resolution of 135×420 pixels, whereas those with long tips were set to 135×635 pixels.
Since the IVD device typically handles transparent liquids, such as blood plasma and chemical reagents, and red liquids, such as blood, we ensured that our datasets reflected this characteristic. To this end, images were acquired using water to simulate colorless reagents and blood plasma, and a red ginger drink to emulate the visual properties of blood. The use of these substitute liquids was necessary because laboratory safety regulations at MACS prohibited handling actual biological fluids or chemical reagents. The selection of water and red ginger extract was based on their visual similarity to the actual fluids used in diagnostic procedures. Figure 5.1 shows an example of images captured by the monitoring system, displaying short tips (left) and long tips (right) filled with transparent liquid. The corresponding extracted input images, used for model training, are shown in Figure 5.2.

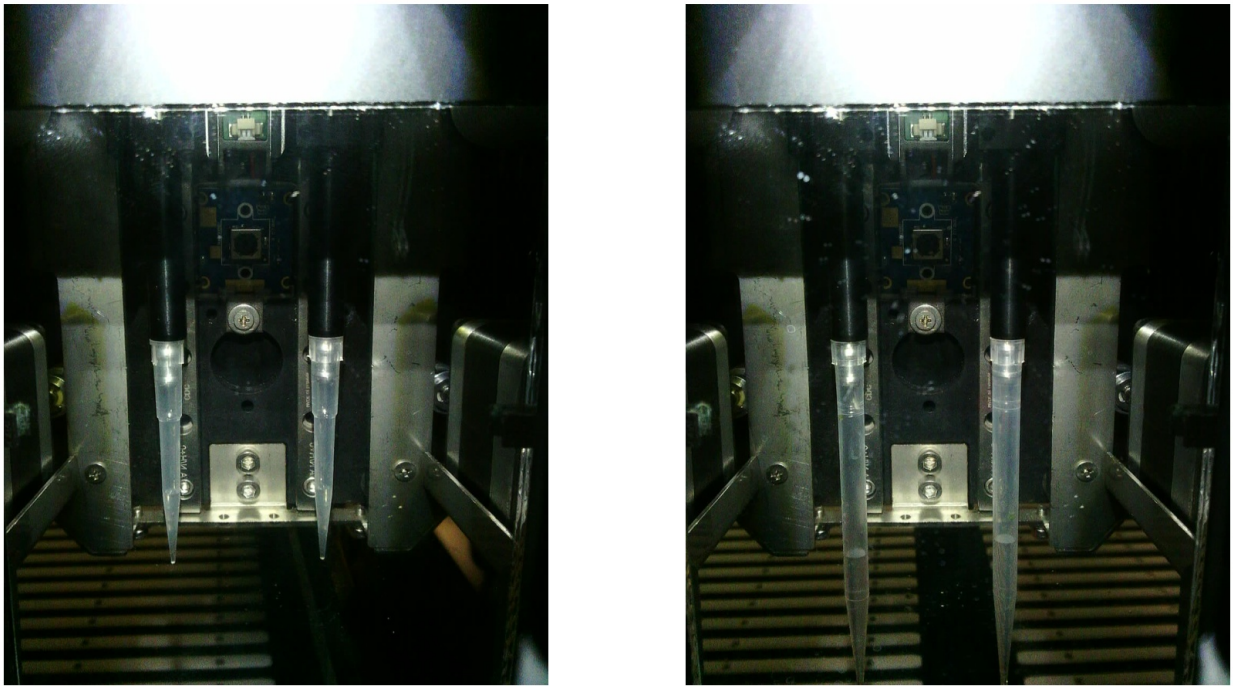


Figure 5.1: Example of short (left) and long (right) tips filled with transparent liquid, as captured by the monitoring system.

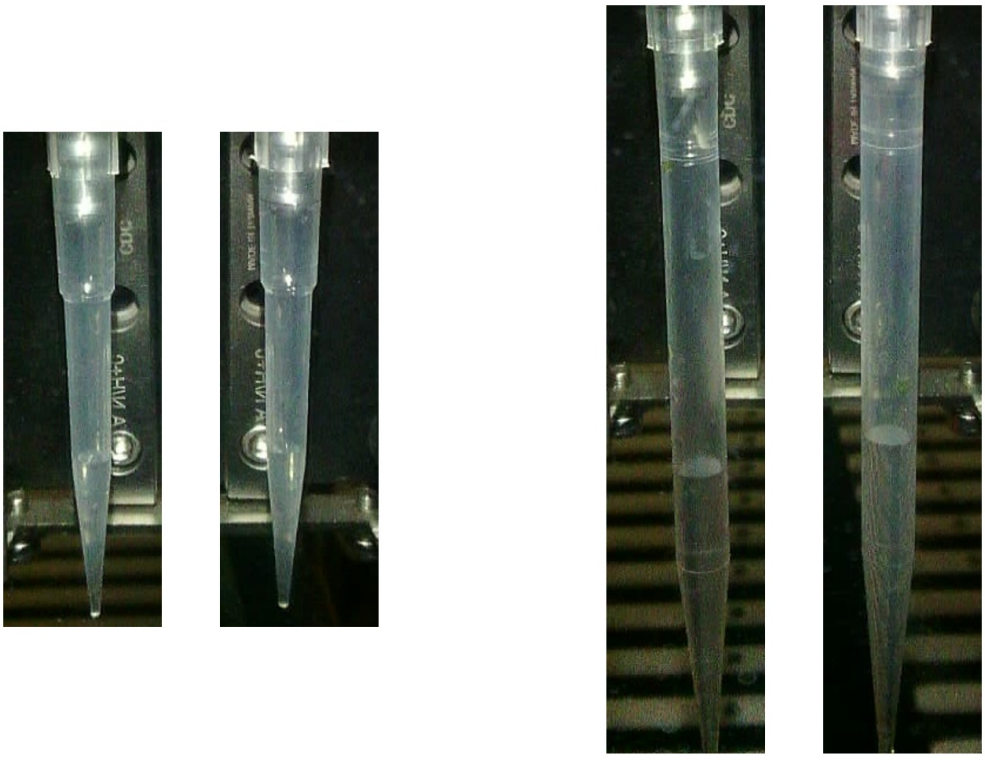


Figure 5.2: Extracted input images corresponding to those shown in Figure 5.1

Figure 5.3 and Figure 5.4 present, in the form of bar plots, the distribution of images across liquid volumes and colors for short and long cases, respectively. For the 0 μL class (empty tip), we maintained an equal 50%-50% distribution between transparent and red liquids. This choice was made because an empty tip contains no liquid, rendering the color distinction irrelevant. Ensuring

this balanced distribution allowed for consistent dataset labeling while avoiding potential biases.

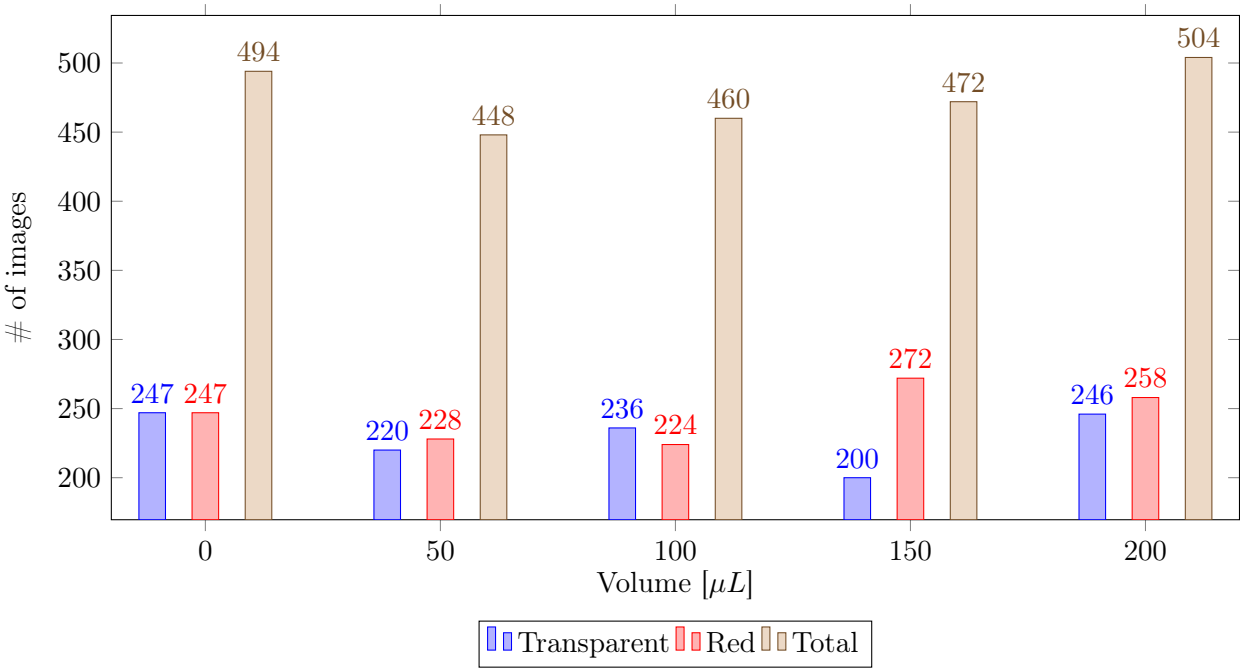


Figure 5.3: Distribution of short dataset images across liquid volume and colors.

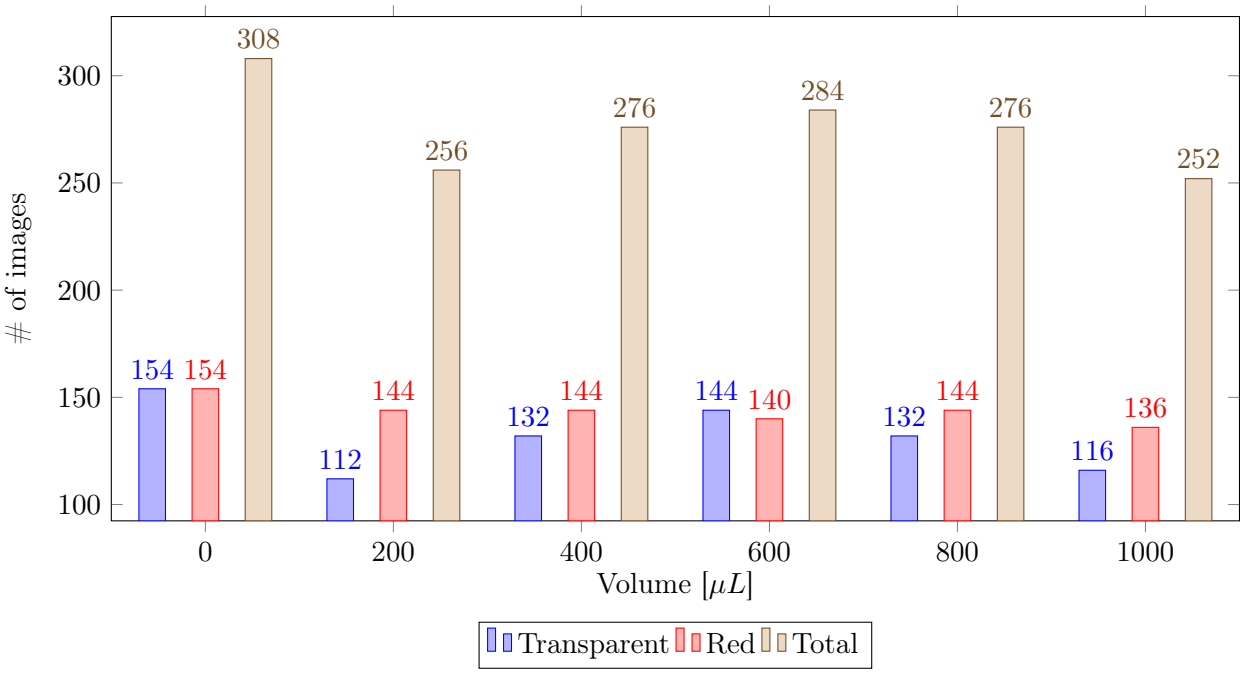


Figure 5.4: Distribution of long dataset images across liquid volume and colors.

Although the two datasets are not perfectly balanced (i.e., each class does not contain exactly 1/5 of the total images for short dataset and 1/6 for long dataset), the distribution is relatively uniform, minimizing the risk of model bias. To quantify this, we computed the Imbalance Ratio (IR), defined as:

$$IR = \frac{\max(N_i) - \min(N_i)}{\sum N_i} \times 100\% \tag{5.1}$$

where $\max(N_i)$ and $\min(N_i)$ represent the maximum and minimum number of samples across output classes, with class here referring to the discrete volume categories (five for short tips and six for long tips), and N_i denotes the total number of samples regardless of liquid color [89]. The calculated values are summarized in Table 5.3.

Tip	IR
Short	2.35%
Long	3.39%

Table 5.3: IR values across all volume classes, regardless of liquid color.

Similarly, if we calculate $\max(N_i)$ and $\min(N_i)$ based on liquid type (transparent vs. red), with N_i representing the total number of samples for each liquid type regardless of the volume class, we obtain the distribution summarized in Table 5.4.

Tip	IR
Short	3.36%
Long	4.35%

Table 5.4: IR values across all liquid colors, regardless of volume classes.

These results confirm that the dataset maintains a balanced distribution between liquid types as well. Considering that all values remain below the 5% threshold, the distribution can be regarded as satisfactory.

5.2 Pre-processing

Before being fed into the models for training and classification, the input images undergo a pre-processing phase to ensure compatibility across different deep learning frameworks. Since the dataset consists of two distinct sets of images, one for short tips and one for long tips, the pre-processing steps were applied separately to each dataset.

The input images were originally stored in BGR format, as acquired by the monitoring system. However, since both TensorFlow/Keras and Pytorch conventionally operate with RGB images, all images were converted to RGB format before training. This conversion preserved all visual characteristics while ensuring consistency across deep learning frameworks. Moreover, unlike previous tasks, we decided not to convert images to grayscale. This choice was made to retain critical color information, such as subtle variations in liquid appearance, which may be relevant for classification.

The input images were resized to 256×256 pixels for CNN model, and to 224×224 pixels ViT model. The reason for using different resizing dimensions lies in the distinct architectures of the two models. The CNN maintains the same 256×256 dimensions used in the type check task (see Section 4.2), ensuring consistency in computational efficiency and performance. On the other hand, the ViT approach leverages a pre-trained model, which requires an input size of $224 \times 224 \times 3$ (details provided in later sections).

Despite the resizing process potentially introducing minor distortions, the transformation remains minimal, ensuring that key visual elements are largely preserved. Figures 5.5 and 5.6 illustrate the pre-processing steps applied to short tip images containing 100 μL of liquid, while Figures 5.7 and

5.8 show the corresponding steps for long tip images containing 600 μL of liquid.

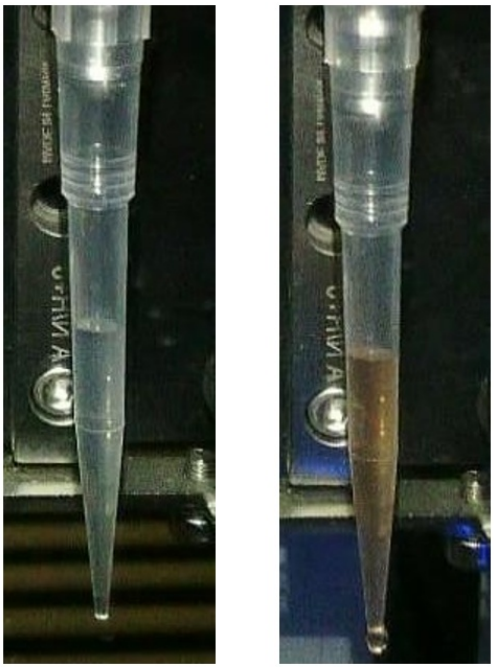


Figure 5.5: Example of two short tip images containing 100 μL of liquid, one transparent (left) and one red (right), before resizing.

Figure 5.6: Example of the resizing process applied to the images shown in Figure 5.5. The images were resized to 256×256 pixels (left) and 224×224 pixels (right).

After pre-processing, the input images have final dimensions of $256 \times 256 \times 3$ for the CNN and $224 \times 224 \times 3$ for the ViT, both in RGB format.

5.3 Data augmentation

As in the CNN-based type check approach, we applied data augmentation to mitigate the risk of overfitting, given the limited size of the dataset. This step is particularly important for ViTs which typically require significantly larger datasets to achieve robust generalization [90]. Data augmentation was applied separately to the short tip and long tip datasets to ensure consistency in training while maintaining dataset-specific variations.
To address this issue, we adopted the same data augmentation transformations used for the CNN-based type check approach, namely horizontal flipping and rotation, but with some modifications. Specifically, instead of applying a random rotation within a range of -3° to +3°, we opted for

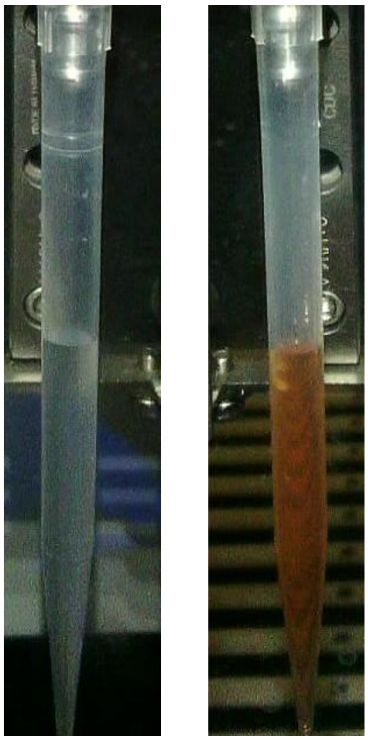


Figure 5.7: Example of two long tip images containing 600 μL of liquid, one transparent (left) and one red (right), before resizing.

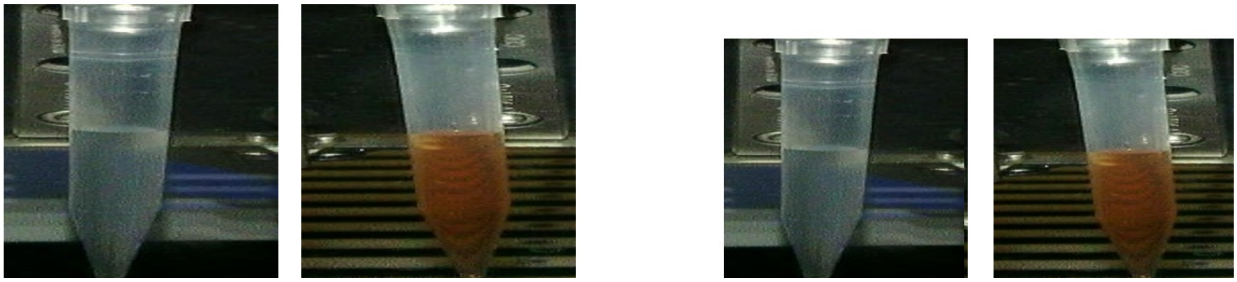


Figure 5.8: Example of the resizing process applied to the images shown in Figure 5.7. The images were resized to 256×256 pixels (left) and 224×224 pixels (right).

fixed rotations of -3° and +3°. This choice is due to the fact that when using different deep learning frameworks, as in our case, the implementations of transformations can vary slightly. In particular, randomization functions may generate slightly different distributions between Pytorch and TensorFlow/Keras, with possible effects on model convergence. By setting fixed angles, we ensure that the input data is transformed identically between the two frameworks, making the results more easily comparable.
For the CNN approach, we applied the following transformations:

- *Flip left right*[3], as previously described in Section 4.4.1;
- *Fixed Rotation of -3°*, implemented using the scientific computation library Scipy[4]. Specifically, we utilized its *rotate* function, which ensures consistent, non-randomized rotations;
- *Fixed Rotation of 3°*, applied using the same method as the previous point.

[3] `https://keras.io/api/layers/preprocessing_layers/image_augmentation/`
[4] `https://scipy.org/`

For the ViT approach, we used transformations provided directly by Pytorch[5]:

- *RandomHorizontalFlip*, defined in the same way as *Flip left right* provided by TensorFlow/Keras. In our implementation, we set the probability $p = 1$, ensuring that all images undergo this transformation, making it effectively deterministic;
- *RandomRotation of -3°*. Although the function name suggests randomness, we explicitly fixed the rotation angle at -3°, ensuring a consistent transformation across all training images;
- *RandomRotation of 3°*. Similarly, we fixed the rotation angle at 3° to maintain symmetry in the augmentation process and avoid unnecessary variability during training.

These transformations were selected based on empirical observations of the monitoring system's behavior. We deliberately avoided additional augmentations, as the pipetting system operates within a standardized and controlled environment. Introducing further variations could introduce unrealistic distortions that do not align with real-world conditions.
Figure 5.9 illustrates the effects of these transformations on the 256×256 images presented earlier in Figures 5.6 (top) and 5.8 (bottom). Figure 5.10 illustrates the effects of these transformations on the 224×224 images presented earlier in Figures 5.6 (top) and 5.8 (bottom).
Finally, we applied online data augmentation, dynamically generating new variations of the training samples at each iteration. This approach prevents overfitting by continuously exposing the model to slightly modified versions of the original data while preserving the integrity of the validation set.

5.4 CNN approach

For both cases, short and long, we adopted the same CNN architecture as the one implemented in the type check approach (see Section 4.4.2). The architecture consists of the same two main blocks: the *feature extraction block* and the *classification block*, as shown in Figure 4.11.
The main differences between between the type check and volume check lie in the hyperparameters, which have been optimized experimentally for this task, and the output layer, as this is a multi-class classification problem rather than a binary one. Specifically, in the final dense layer, we used five output neurons for short volume check case and six output neurons for long volume check, each corresponding to a liquid volume class. Instead of a sigmoid activation, we applied a softmax activation function, which is more suitable for multi-class classification.
The entire implementation, including training and validation, was carried out using Tensorflow and Keras libraries in Python.
For brevity, we present for both cases only the final model configuration, which achieved the best performance. The decision to continue using a custom CNN, rather than a pre-trained model, was driven by extensive testing. Despite its relatively lightweight architecture, the model has proven effective even for more complex tasks such as this one.

5.4.1 Architecture parameters

Tables 5.5 and 5.6 summarize the key parameters for each layer of the CNNs, including input and output dimensions. The input to the network consists of a $256 \times 256 \times 3$ image, obtained after pre-

[5] `https://docs.pytorch.org/vision/0.22/transforms.html`

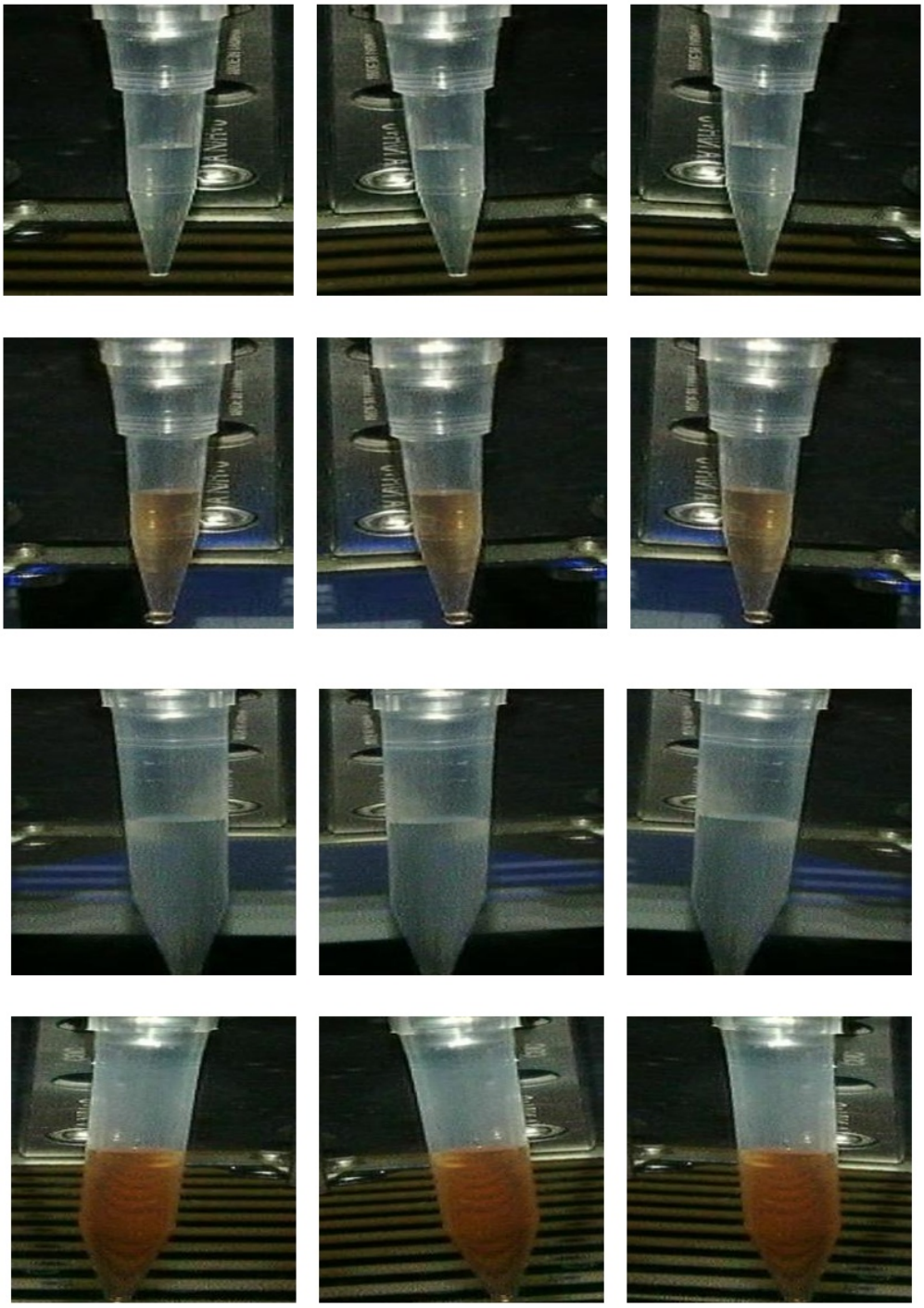


Figure 5.9: Example of data augmentation applied to the 256×256 images shown in Figures 5.6 (top) and 5.8 (bottom). In both cases, the images after applying horizontal flip (left), $+3^\circ$ rotation (center), and -3° rotation (right) are displayed.

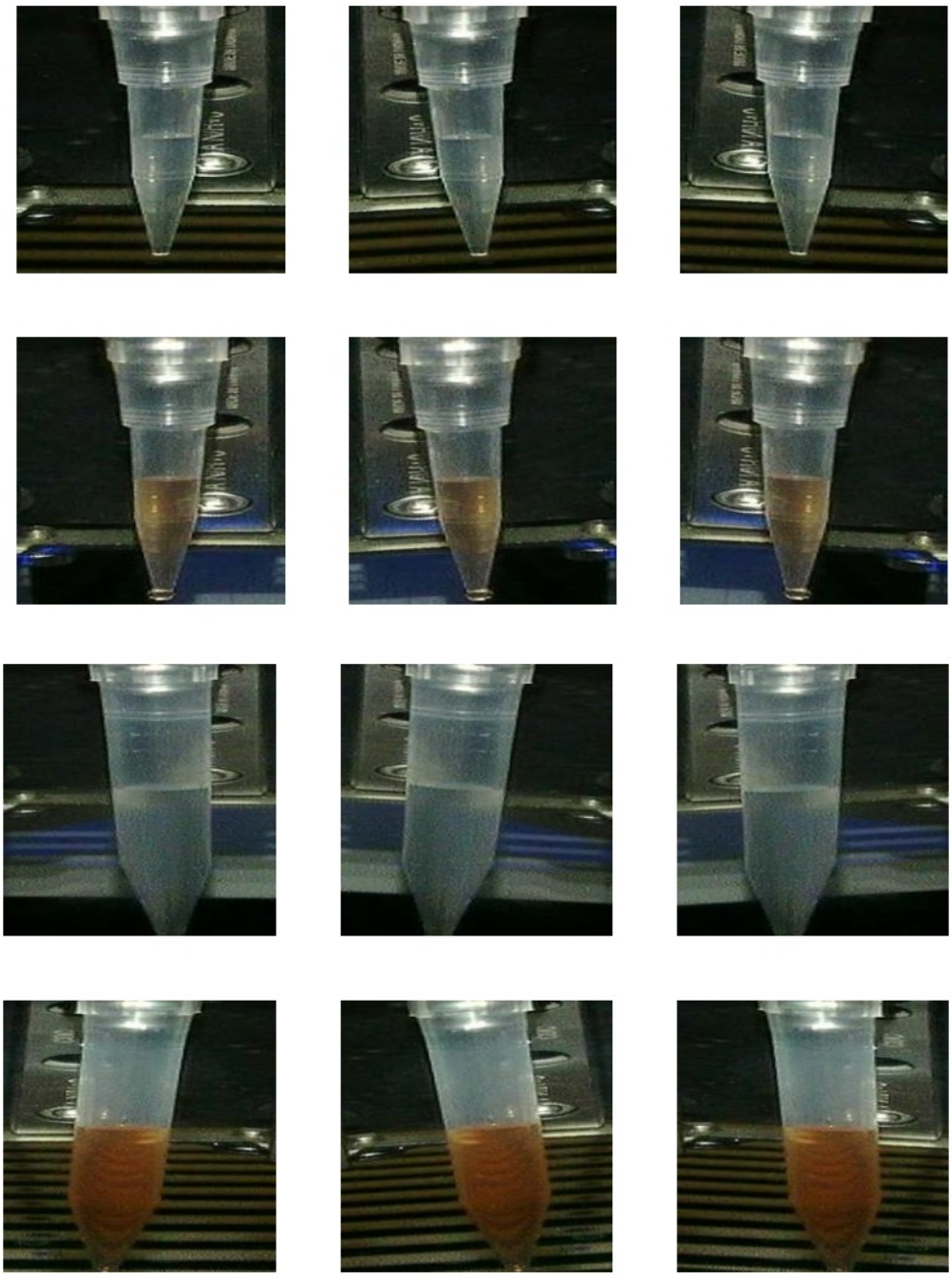


Figure 5.10: Example of data augmentation applied to the 224×224 images shown in Figures 5.6 (top) and 5.8 (bottom). In both cases, the images after applying horizontal flip (left), $+3^{\circ}$ rotation (center), and -3° rotation (right) are displayed.

processing. Before being fed into the model, these images are converted into tensors, as required by the TensorFlow framework.

Layer	# of input channels	# of output channels	Kernel size	Stride	Input size	Output size
Conv2D	3	8	3	1	256×256×3	256×256×8
Dropout (rate = 0.1)	-	-	-	-	256×256×8	256×256×8
MaxPool2D	8	8	-	2	256×256×8	128×128×8
Conv2D	8	16	3	1	128×128×8	128×128×16
Dropout (rate = 0.1)	-	-	-	-	128×128×16	128×128×16
MaxPool2D	16	16	-	2	128×128×16	64×64×16
Conv2D	16	32	3	1	64×64×16	64×64×32
Dropout (rate = 0.1)	-	-	-	-	64×64×32	64×64×32
MaxPool2D	32	32	-	2	64×64×32	32×32×32
Flatten	-	-	-	-	32×32×32	32768
FC	-	-	-	-	32768	50
FC	-	-	-	-	50	20
FC	-	-	-	-	20	10
FC	-	-	-	-	10	5

Table 5.5: Parameters of the short volume check CNN model for each layer.

Layer	# of input channels	# of output channels	Kernel size	Stride	Input size	Output size
Conv2D	3	8	3	1	256×256×3	256×256×8
Dropout (rate = 0.1)	-	-	-	-	256×256×8	256×256×8
MaxPool2D	8	8	-	2	256×256×8	128×128×8
Conv2D	8	16	3	1	128×128×8	128×128×16
Dropout (rate = 0.1)	-	-	-	-	128×128×16	128×128×16
MaxPool2D	16	16	-	2	128×128×16	64×64×16
Conv2D	16	32	3	1	64×64×16	64×64×32
Dropout (rate = 0.1)	-	-	-	-	64×64×32	64×64×32
MaxPool2D	32	32	-	2	64×64×32	32×32×32
Flatten	-	-	-	-	32×32×32	32768
FC	-	-	-	-	32768	50
FC	-	-	-	-	50	20
FC	-	-	-	-	20	10
FC	-	-	-	-	10	6

Table 5.6: Parameters of the long volume check CNN model for each layer.

As observed in both Tables 5.5 and 5.6, moving through the convolutional layers, the number of channels increases, allowing the model to capture more complex patterns, while the spatial dimensions decrease, due to the Max pooling layers, which help reduce computational complexity while preserving the most important features. In addition, we applied `"same"` padding to all convolutional layers to maintain spatial consistency. To prevent overfitting, a dropout layer with a rate of 0.1 was applied after each convolutional layer. Conversely, in the fully connected layers, the number of neurons gradually decreases, condensing the extracted information before classification. The final layer consists of five neurons for the short tip volume check, each representing a class (0, 50, 100,

150, 200 μL), and six neurons for the long tip volume check, corresponding to the classes (0, 200, 400, 600, 800, 1000 μL). The softmax activation function ensures that the output is a probability distribution over these five and six classes, with the predicted class corresponding to the neuron with the highest probability. The sum of all output values is always equal to 1.

5.4.2 Training parameters

The two custom networks were trained only once due to the high computational cost associated with the training process. Given the complexity of the classification task and the need for extensive hyperparameter tuning, multiple training repetitions would have significantly increased processing time without a guaranteed improvement in performance. Instead, we prioritized careful hyperparameter selection and a k-fold cross-validation strategy to enhance model robustness. All training and validation parameters, identical for both cases, were empirically chosen and are summarized in Table 5.7.

Parameter	Value
# of replications	1
# of epochs	25
Batch size	8
Learning rate	10^{-3}
Weight decay	10^{-4}
Pooling size	2

Table 5.7: Training parameters for both short and long tip volume check CNN models.

We trained both models for 25 epochs, a value determined experimentally to ensure convergence without excessive training time. Unlike in the type check CNN, early stopping was not applied, as it led to suboptimal performance by halting training too early. Instead, dropout was used as the primary regularization technique to prevent overfitting. The pooling size was set to 2, a widely used value in the literature that balances computational efficiency and feature retention. The batch size was fixed at 8, chosen as a trade-off between training stability, memory constraints, and processing speed. A smaller batch size helps avoid memory saturation while still allowing for efficient gradient updates. For optimization, we employed the Adam optimizer with weight decay, which helps mitigate noisy gradients, particularly important given the small batch size. As in the CNN-based type check approach, we applied k-fold cross-validation ($k = 5$) to improve generalization and reduce overfitting. The choice of $k = 5$, instead of higher values, aligns with common practice in deep learning, where larger validation sets help better assess the model's ability to generalize. After performing 5-fold cross-validation, the model's performance was evaluated using Categorical Cross-Entropy (Categorical CE) loss, average accuracy, and the macro F1-score. Unlike binary classification, where micro (classic) and macro F1-scores coincide, in multi-class settings the two metrics may diverge. In particular, the macro F1-score was preferred because it computes the F1 independently for each class and then averages the results, thereby ensuring that all classes contribute equally to the final evaluation [87]. This choice is especially relevant in our case, where the use of a simple non-stratified k-fold cross-validation may temporarily alter the balance among classes, despite the overall dataset being well balanced according to the IR.

5.5 ViT approach

As previously introduced in this Chapter, for this task, we implemented a vision transformer, which, unlike the CNN, processes images in their entirety and captures long-range dependencies through a self-attention mechanism. For both classification tasks, we adopted the same base ViT architecture. The main difference between the two implementations lies in the final classification layer configuration. Specifically, for the short volume check, the output layer consists of five neurons, while for the long volume check, it contains six neurons. For brevity, we present only the final configuration that achieved the best performance for both tasks.
The model used in both cases is a pre-trained ViT, provided by the HuggingFace[6] library [91]. We selected the model *google/vit-base-patch-224*. The decision to use a pre-trained model, instead of building and training one from scratch, was motivated by the high computational cost and the large amount of data required to effectively train a ViT model (typically tens of thousands of images) [92]. Furthermore, this specific model offered an excellent balance between computational efficiency and performance. The entire implementation, including training and validation, was carried out using Pytorch library in Python. Preliminary tests were performed to ensure the compatibility and seamless integration of the deep learning libraries implemented in Python with the machine vision system's operational environment.
In the following sections, we will describe the architecture of the model, explaining all its parameters in detail. We will also present the training parameters and explain the rationale behind their selection.

5.5.1 ViT Architecture

The architecture of the model selected for both volume check tasks is based on the google/vit-base-patch-224 model, which has the same architectural characteristics as the ViT-Base variant described by Dosovitskiy et al. [29] The model includes all the standard components expected in a vision transformer. The only modification compared to the original model is the addition of a dropout layer before the final classification layer, which helps reduce overfitting.
Our vision transformer architecture is shown in Figure 5.11. The key parameters of the ViT used for both volume check tasks are summarized in Table 5.8. The input image consists of a $224 \times 224 \times 3$

Parameter	Value
Patch size	16 ×16
Image size	224 × 224 × 3
Hidden size	768
MLP hidden dimension	3072
Number of transformer layers	12
Dropout rate	0.2

Table 5.8: Training parameters for both short and long tip volume check ViT models.

image, obtained after pre-processing. Before being fed into the vision transformer, the image is converted into a tensor and normalized according to the statistics of the dataset on which the model was pre-trained, specifically the *ImageNet-21k* statistics [93], in order to match the format

[6] `https://huggingface.co`

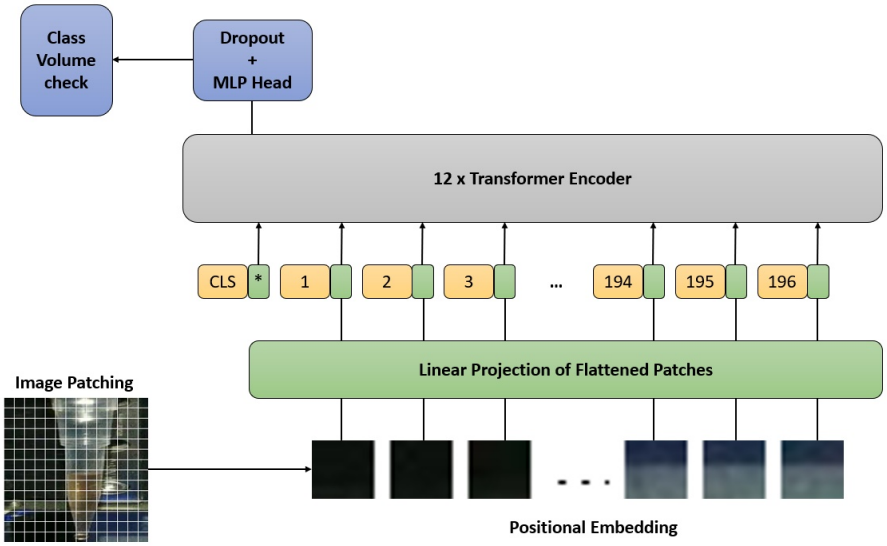


Figure 5.11: Visualization of the vision transformer architecture.

and distribution expected by both the model and the pytorch framework.
The pre-trained model divides the input image into patches of size $16 \times 16 \times 3$ pixels, resulting in 196 patches, with the addition of the CLS token, which is necessary to capture the encoded features of the entire image. The 12 transformer layers indicate that the model consists of 12 parallel heads performing the task. The dropout rate was set to 0.2, meaning that 20% of the neurons were randomly deactivated during training. This value was empirically determined to balance between regularization and the capacity of the model, and is consistent with standard best practices. The final classification layer consists of 5 neurons for the short volume check, each representing a class (0, 50, 100, 150, 200 μL), and 6 neurons for the long volume check, corresponding to the classes (0, 200, 400, 600, 800, 1000 μL). As per standard practice, a softmax activation function is applied at the final layer to ensures that the output represents the probability distribution over classes. The predicted class corresponds to the neuron with the highest probability.

5.5.2 Training parameters

As with the CNNs, the two custom ViT models were trained a single time, due to the significant computational effort required by the training process. Considering the complexity of the classification task and the necessity of extensive hyperparameter optimization, repeating the training would have greatly increased processing time without necessarily improving performance. Instead, we prioritized careful hyperparameter selection and the adoption of a k-fold cross-validation strategy to enhance model robustness and prevent overfitting. All training and validation parameters, identical for both cases, were empirically chosen and are summarized in Table 5.9.
We trained both models for 25 epochs, a value determined empirically to ensure convergence. As for CNN approach, we decided not to apply early stopping, to avoid prematurely underestimating the model's learning potential. The batch size was fixed at 8, chosen as a trade-off between training stability and memory constraints. Larger batch sizes would have caused memory saturation on the available computational resources. For optimization, we employed the Adam optimizer with weight decay, a technique useful for mitigating noisy gradients, particularly important given the

Parameter	Value
# of replications	1
# of epochs	25
batch size	8
learning rate	10^{-4}
weight decay	10^{-2}

Table 5.9: Training parameters for both short and long tip volume check ViT models.

small batch size. We applied a simple k-fold cross-validation with $k = 5$, to prevent overfitting and improve the models' generalization ability. This choice was maintained to ensure a fair comparison between the ViT and CNN models. As with the CNN model, after performing the 5-fold cross-validation, the ViT performance was evaluated using Categorical CE loss, accuracy, and macro F1-score. The latter is particularly relevant in the multi-class setting, as it provides a balanced evaluation across all classes, complementing accuracy and offering a more reliable assessment of the model's performance.

5.6 Short volume check results

5.6.1 CNN approach

Evaluation on training and validation set

The CNN model for the short volume check was trained on 7624 images, obtained through data augmentation applied to the 1,906 original training images, and evaluated on 472 images. The model was trained using 5-fold cross-validation, meaning it was trained five times, each time using a different fold as the validation set and the remaining folds as training data. Figures 5.12 and 5.13 show the loss curve and accuracy curves, during training and validation phases for each fold.

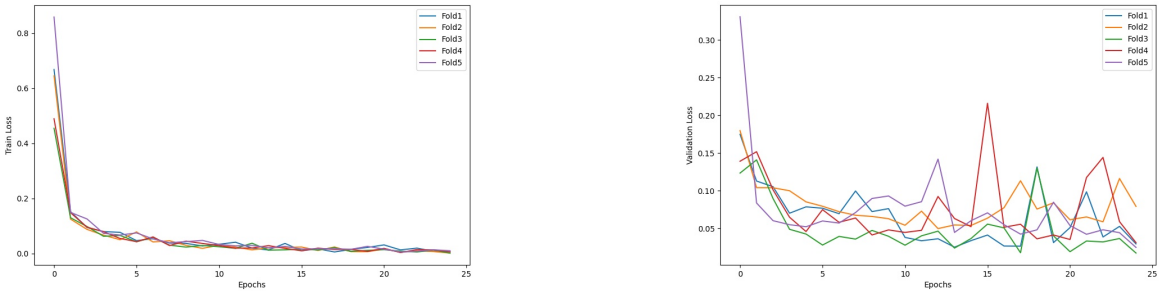


Figure 5.12: Loss curves during training (left) and validation (right) for each fold for short volume check.

As shown in figures, the curve behaviours during the training and validation phases are different. The training curves show a stable and consistent trend across all folds. The training loss rapidly decreases rapidly during the first few epochs and then stabilizes at very low values, approaching zero. In parallel, the training accuracy increases very quickly and stabilises around 98%. This behaviour indicates an effective and stable performance. On the other hand, regarding the validation phase, the trend is different. The validation loss shows a tendency to decrease, as the accuracy tends to increase until it stabilizes around 97%, with peaks occasionally dropping to approximately 93%. However, in both cases, the curves for all the folds have pronounced oscillations, which suggest a

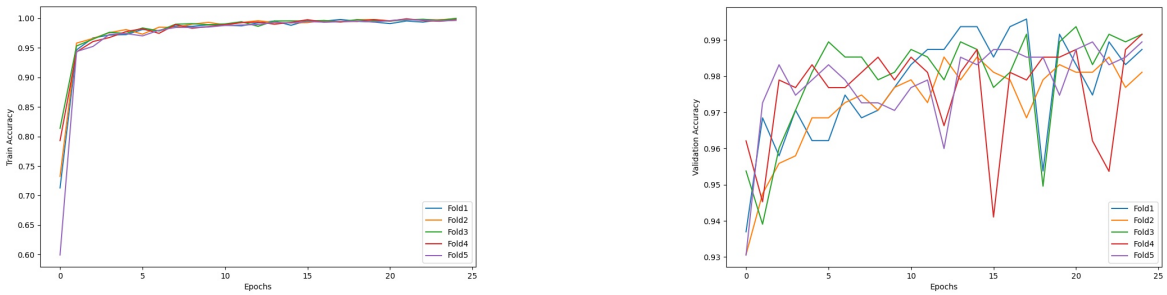


Figure 5.13: Accuracies during training (left) and validation (right) for each fold for short volume check.

higher sensitivity to fold-specific variations and reflect the natural variability in model generalization.

Despite these fluctuations, the validation accuracy remains consistently high, confirming a the model's good generalization ability. It is worth noting that the use of dropout and weight decay likely contributed to controlling overfitting, maintaining a small performance gap between training and validation phases.

Table 5.10 shows the accuracies achieved at the end of validation for each fold. In addition, the last column represents the average accuracy obtained by averaging across all folds.

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean Accuracy
98.74%	98.11%	99.16%	99.16%	98.95%	98.82% ± 0.39%

Table 5.10: Average accuracies for each fold and overall mean accuracy.

By selecting the best model, Figure 5.14 presents the confusion matrix obtained on the validation set. The model achieved an accuracy of 99.16% and a macro F1-score of 99.14%, demonstrating excellent performance in the classification of different liquid quantities. Since our model was designed for a multi-class classification problem, the traditional ROC curve, which is a binary classification metric, is not directly applicable. In multi-class scenarios, the ROC curve can be computed using the one-vs-rest approach, where a ROC curve is generated for each class, treating that class as positive and the others as negative. However, this approach presents different computational challenges, when dealing with a large number of classes, as in our case. Specifically, the generation of a separate ROC curve for each class would be computationally expensive and could lead to difficulties in interpretation. Given these challenges, we opted to focus on accuracy, F1-score, and the confusion matrix, which together offer a more direct and reliable interpretation of the model performance. From an operational perspective, classification errors in the volume check do not affect the correctness of the diagnostic result. The monitoring system automatically detects mismatches, discards the affected liquid, and repeats the aspiration–dispensing cycle for the specific channel involved. The impact of these errors is therefore limited to a modest increase in execution time and a slightly higher consumption of consumables, without compromising the accuracy or reliability of the analytical outcome.

Explainability

As for type check, we performed a feature analysis using the IG method and Grad-CAM method. Figures 5.15, 5.16, 5.17, 5.18, 5.19, 5.20, 5.21, 5.22 and 5.23 illustrate the original image (left), the IG

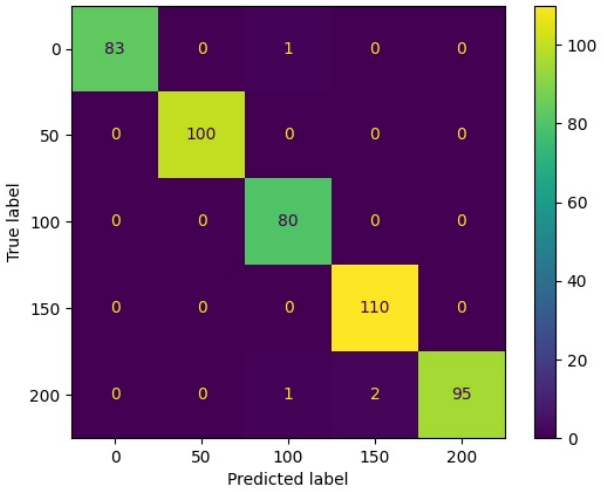


Figure 5.14: Confusion matrix of the best CNN model on the validation set.

map (center) and the Grad-CAM heatmap (right) for the following cases: 0 μL, 50 μL transparent liquid, 50 μL red liquid, 100 μL transparent liquid, 100 μL red liquid, 150 μL transparent liquid, 150 μL red liquid, 200 μL transparent liquid, and 200 μL red liquid, respectively.

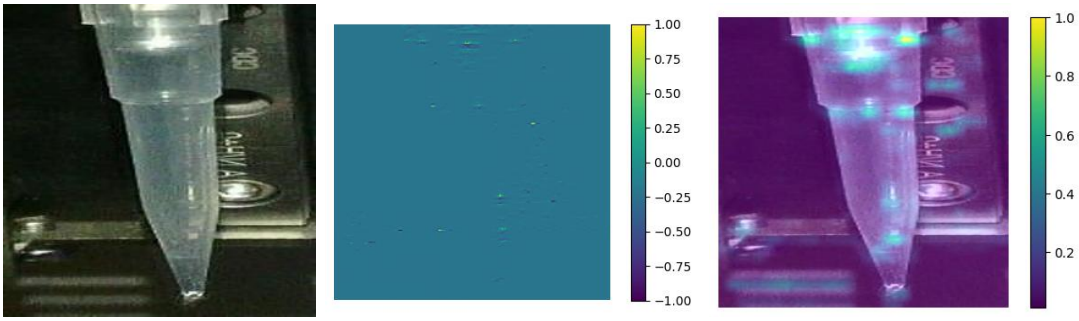


Figure 5.15: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 0 μL

In contrast to the integrated gradients of the type check, where the values oscillate between negative and positive values, in the volume check we rescaled the IG scores to the range $[-1, 1]$ because the raw attributions were too small to be visible; this rescaling does not affect any quantitative analysis. In our setting, we observed IG maps are comparatively faint and sparse. Upon analysing the IG feature maps, we observe that the model focuses mainly on the level of liquid in all images, except for the empty case, where attributions are distributed randomly and focused on reflections. In the case of 50 μL of transparent liquid, the attribution is strong on the liquid level, with negative values in other areas. For the red liquid, the attribution is not as marked, with a lower concentration on the level. For the larger volumes (100, 150 and 200 μL), the model continues to focus on the liquid level, but with variations in the attributes related to the contrast and color of the liquid. In particular, for the case of 1000 μL of red liquid, the attribution extends over the entire area of the liquid, showing a diffused positive and negative influence. Overall, the model appears to be able to accurately detect the level of the liquid, especially for larger volume.
With respect to the Grad-CAM heatmaps, the regions of higher relevance are clearly localized at

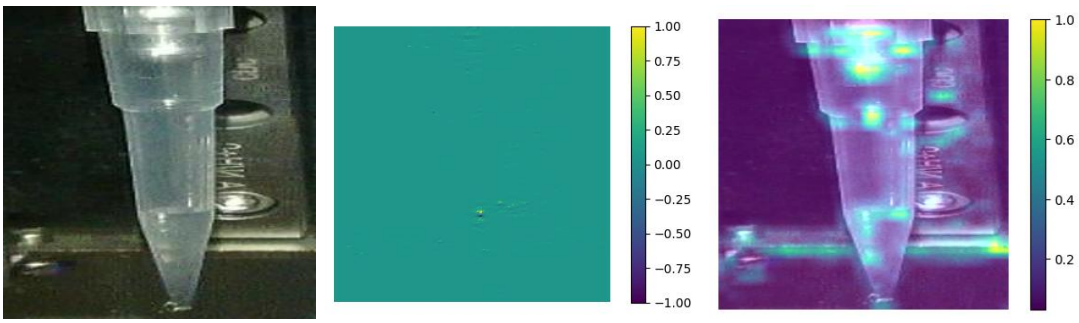


Figure 5.16: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 50 μL transparent liquid.

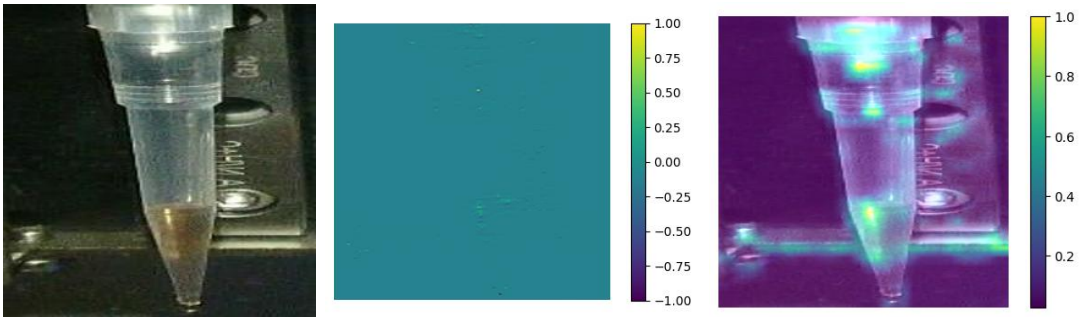


Figure 5.17: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 50 μL red liquid.

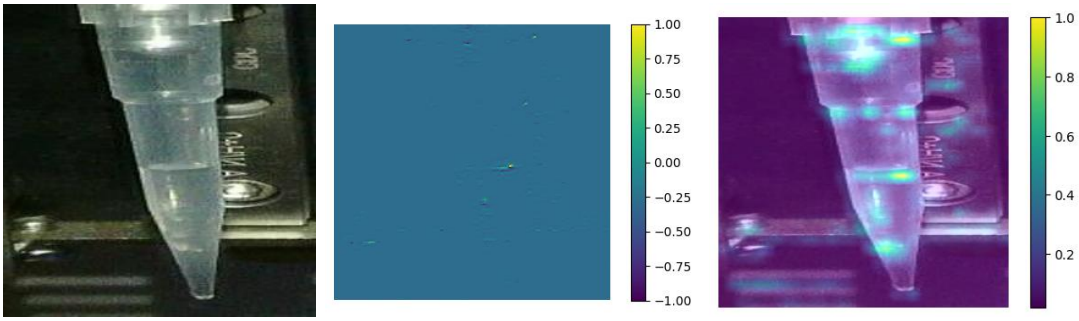


Figure 5.18: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 100 μL transparent liquid.

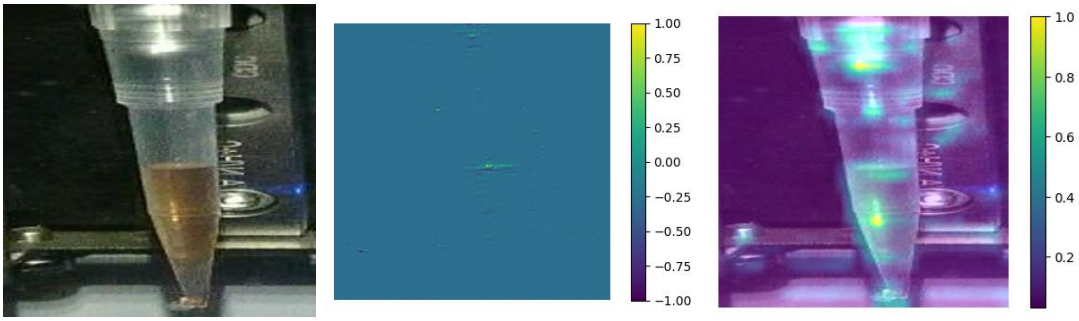


Figure 5.19: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 100 μL red liquid.

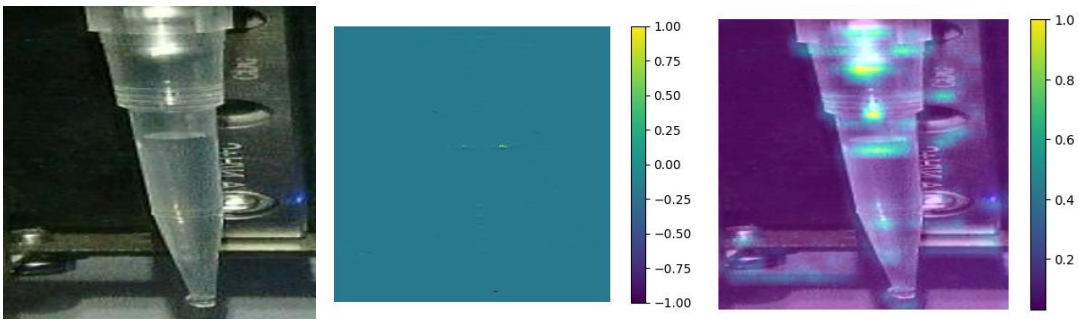


Figure 5.20: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 150 μL transparent liquid.

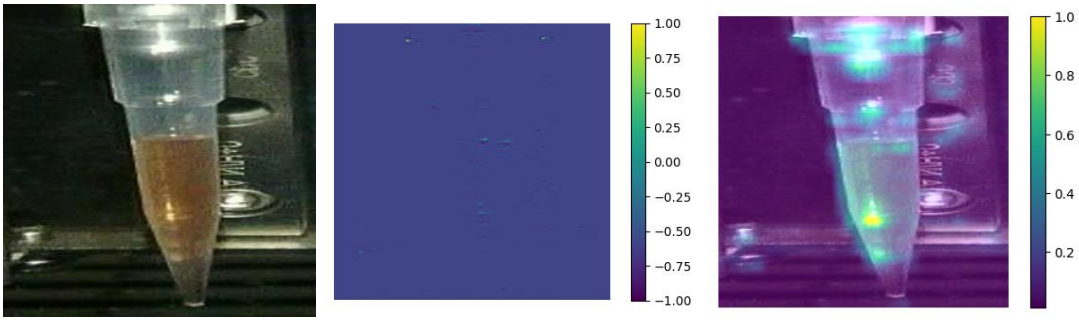


Figure 5.21: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 150 μL red liquid.

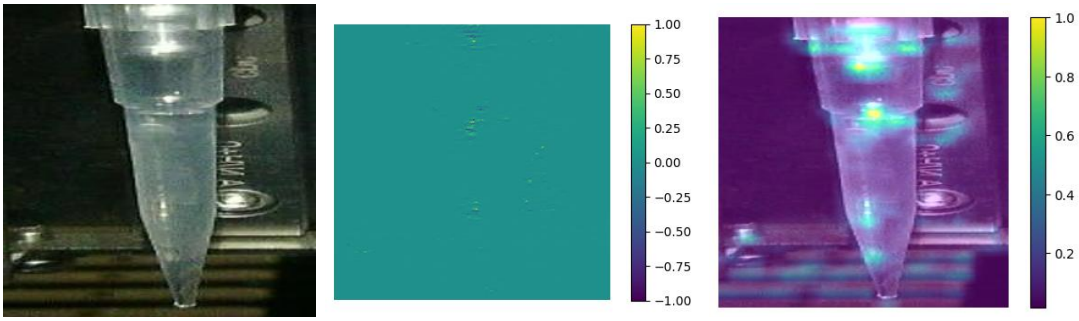


Figure 5.22: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 200 μL transparent liquid.

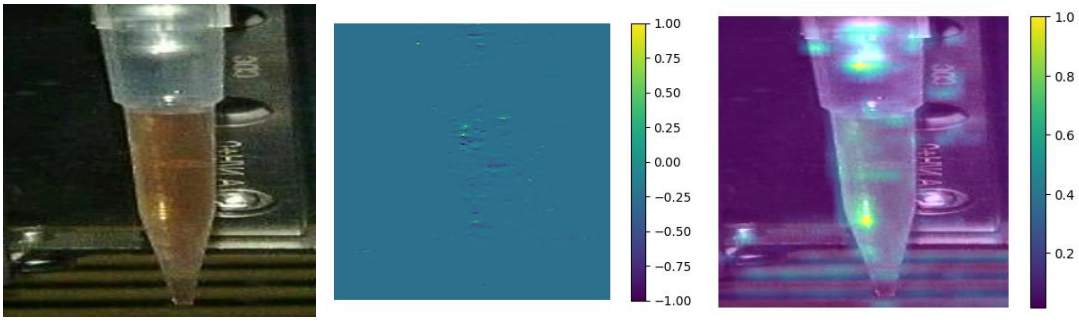


Figure 5.23: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 200 μL red liquid.

the liquid level for the transparent liquid, and they are more widely distributed across the entire liquid area for the red liquid. This difference is likely due to the sharper contrast between the red liquid and the background, making it more distinct for the model. In the case of transparent liquid, the highest contrast is observed at the liquid level, with noticeable blue-green colors in the heatmap. Additionally, across all images, the Grad-CAM heatmaps indicate that the model also attends to background structures—such as specular reflections near the tip base and the white bars—likely driven by local contrast rather than task-relevant evidence. As a practical aid for qualitative inspection, future work could apply a tip mask when visualising attributions, in order to exclude background regions while leaving training and evaluation unchanged.
These findings are consistent across the entire dataset. Although the results obtained in terms of explainability confirm the model's ability to focus on key image features, such as the liquid level, the way this information is represented by the two explainability techniques reveals some differences. Specifically, IGs provides a more fine-grained view by assigning relevance at the pixel level, while Grad-CAM highlights broader regions where the model's attention is concentrated. As a result, Grad-CAM offers a more general overview of the model's behavior. These differences in visualization suggest that, unfortunately, combining the two techniques may lead to inconsistencies in the precise localization of the model's areas of interest.

Evaluation on an external test set

To assess the models' ability to generalize beyond the training data, we evaluated the CNN model on an external test set of 384 images, covering all liquid volumes for short tips and both liquid types (transparent and red). The distribution of the test set is reported in Figure 5.24. This dataset was acquired independently from the training/validation data but under the same experimental conditions, ensuring consistency while avoiding data leakage. Figure 5.25 present the confusion

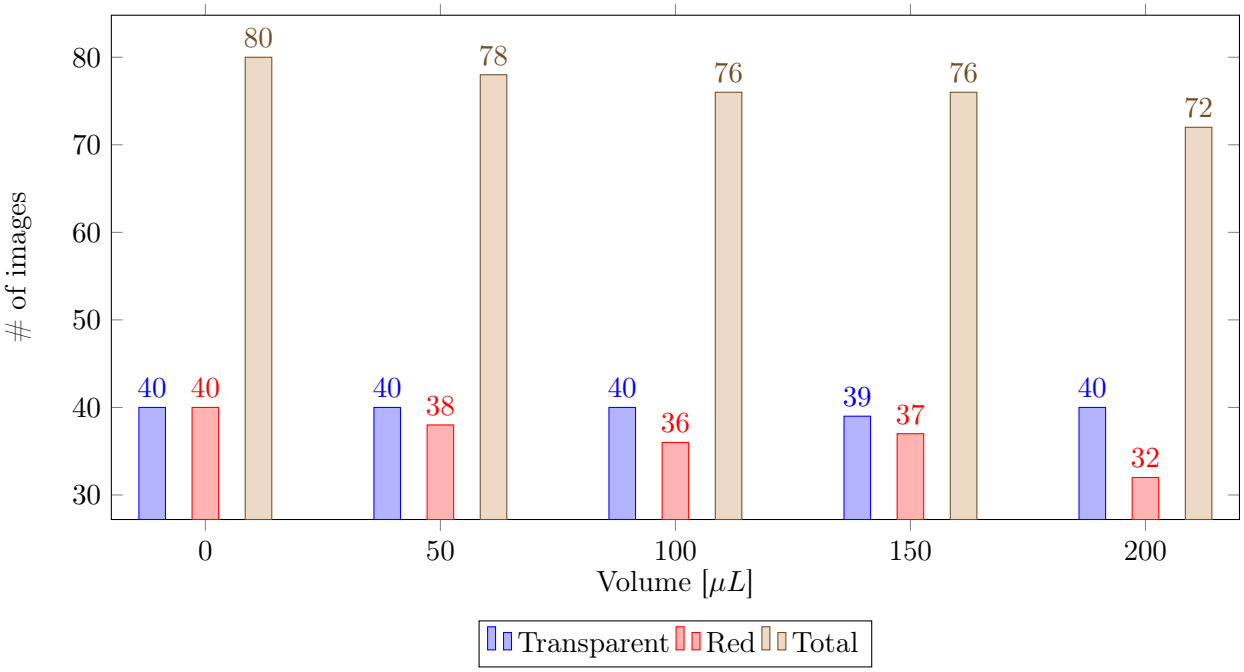


Figure 5.24: Distribution of the test set for the short volume across different liquid volumes and colors.

matrix obtained on the test set. Our model achieved an accuracy of 95.03% and a macro F1-score

of 94.9%, indicating that the model is well-suited for this multi-class classification task and has an excellent ability to generalize. Although the model achieves high accuracy and a macro F1-score, the observed errors only lead to discarding the affected tip and repeating the cycle for that channel. This slightly increases execution time and tip consumption, without affecting the reliability of the diagnostic outcome. Furthermore, we performed an feature map analysis using IG method and the

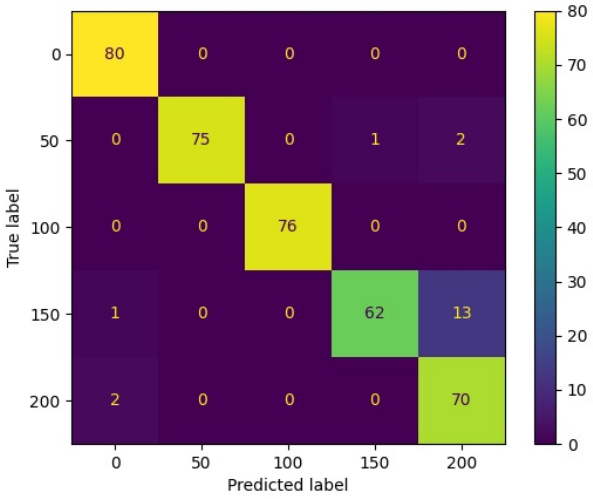


Figure 5.25: Confusion matrix of CNN model on the test set.

Grad-CAM technique to understand the mechanism behind the prediction. Figures 5.26, 5.27, 5.28, 5.29, 5.30, 5.31, 5.32, 5.33, and 5.34 illustrate the original image (left), the IG map (center) and the Grad-CAM heatmap (right) for the following cases: 0 μL, 50 μL transparent liquid, 50 μL red liquid, 100 μL transparent liquid, 100 μL red liquidi, 150 μL transparent liquid, 150 μL red liquid, 200 μL transparent liquid, and 200 μL red liquid, respectively.

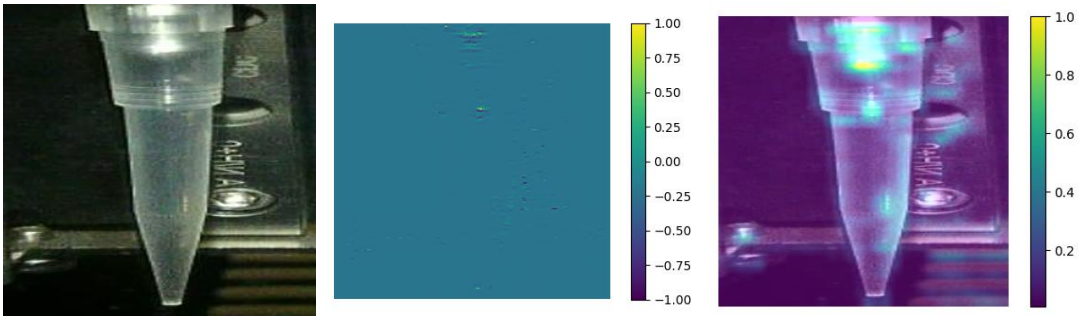


Figure 5.26: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 0 μL

Upon analysing the IG maps from the test set, we observe that the model consistently focuses on the liquid level across different volumes, aligning with the patterns seen during training. In the case of the empty tip, the attributions are sparse and focused on the edges, as observed previously. As the liquid volume increases, the model's attention is concentrated primarily on the liquid level, though the strength of the attribution varies depending on the volume. In some cases, the attributions are more localized to the liquid area, while in others, the attribution extends across the tip, with both positive and negative influences. Furthermore, the Grad-CAM heatmaps also show an alignment

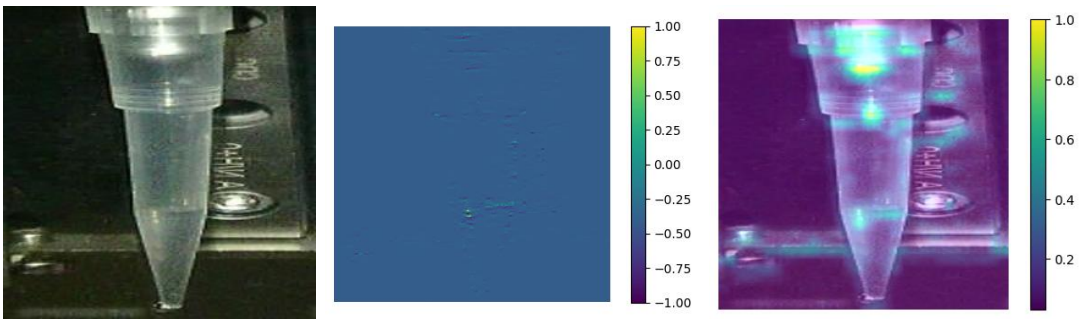


Figure 5.27: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 50 μL transparent liquid.

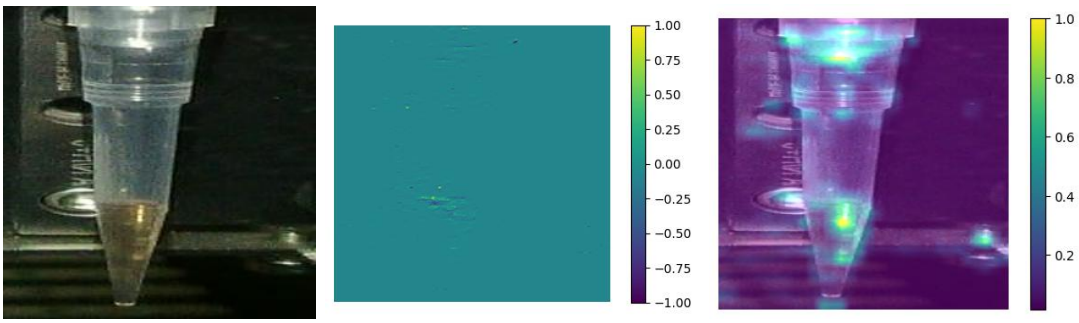


Figure 5.28: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 50 μL red liquid.

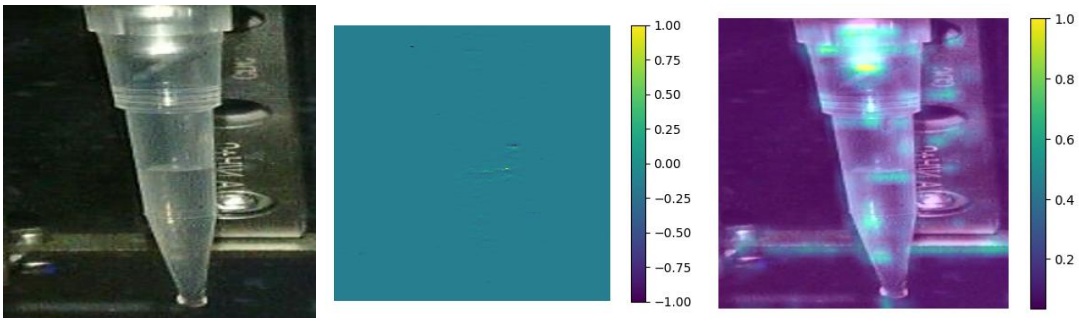


Figure 5.29: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 100 μL transparent liquid.

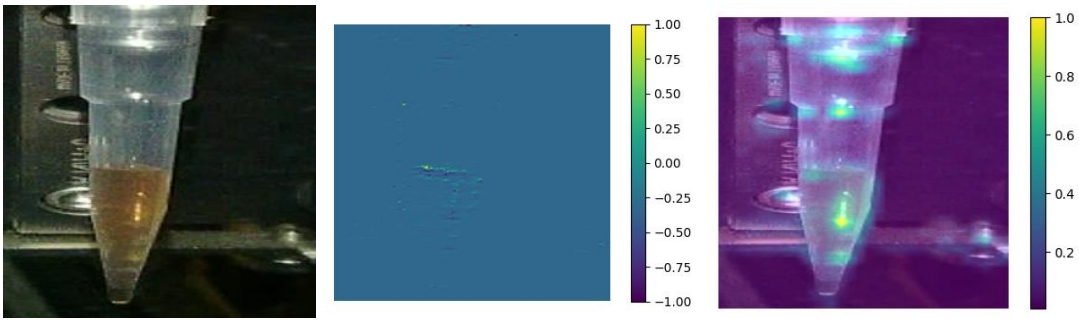


Figure 5.30: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 100 μL red liquid.

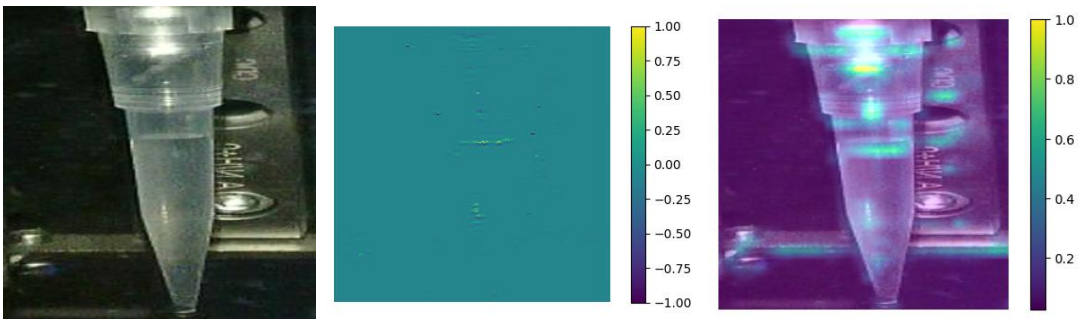


Figure 5.31: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 150 μL transparent liquid.

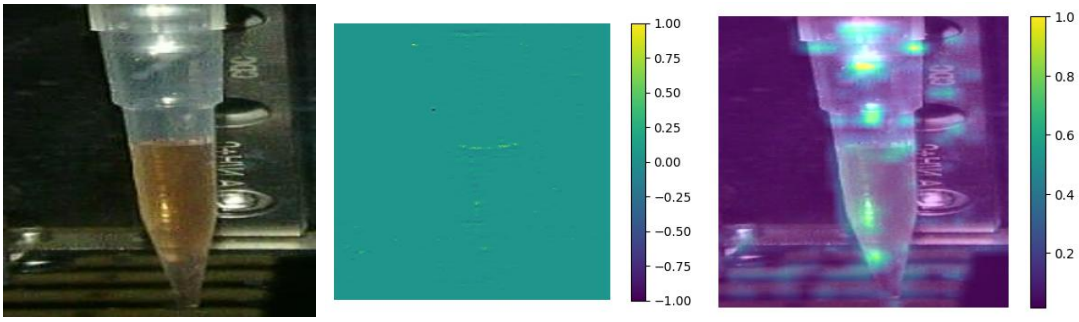


Figure 5.32: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 150 μL red liquid.

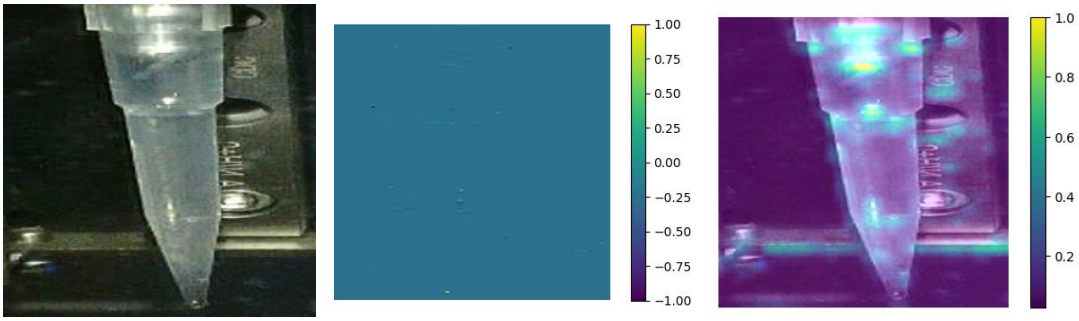


Figure 5.33: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 200 μL transparent liquid.

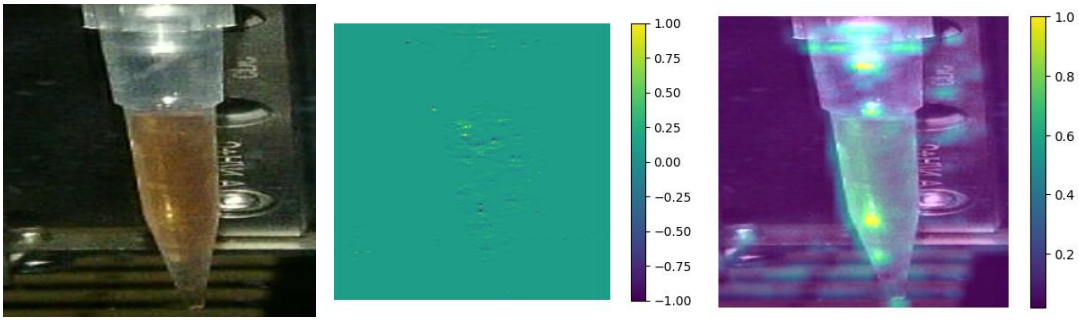


Figure 5.34: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 200 μL red liquid.

with the results observed during training. The heatmaps confirm that the model primarily focuses on the liquid inside the tip. In the case of transparent liquid, the model highlights the liquid level and in the case of red liquid, the focus is on the entire liquid volume within the tip. Moreover, as observed during the training phase, the model focuses attention to reflections on the tip base and background areas with higher contrast. This suggests that our model is sensitive to features that create significant contrast in the image. Nonetheless, the same caveats observed during training and validation persist: IG attributions remain comparatively faint and sparse, and Grad-CAM occasionally highlights background structures (e.g., specular reflections and white bars), indicating sensitivity to contrast.
These results demonstrate consistent behavior across both the training and test datasets, confirming that the model's decision-making process is reliable and generalizable to unseen data.

Time performance

Finally, we assessed the computational efficiency of our model by measuring the time required to perform the pre-processing and classification procedures. Table 5.11 presents the average pre-processing, prediction and total times, along with the corresponding standards deviations for our model. The average times were calculated on the 384 images of the test set, and performed on the system with the hardware identical to the one mounted in the medical device.

Function	Time
Pre-processing	$3.715 \cdot 10^{-2} \pm 2.879 \cdot 10^{-2}$ s
Prediction	$1.916 \cdot 10^{-1} \pm 1.159 \cdot 10^{-1}$ s
Total	$2.288 \cdot 10^{-1} \pm 1.312 \cdot 10^{-1}$ s

Table 5.11: Average extraction, prediction, and total times with standard deviations for the selected model.

These time results demonstrate that our model requires less than one second to perform the short volume check and complies with the predefined time constraint for task execution.

5.6.2 ViT approach

Evaluation on training and validation set

The ViT model for the short volume check was trained and validated using the same dataset employed for the CNN model. This means that our ViT model was trained on 7624 images, validated on 472 images, and 5-fold cross-validation was applied. Figures 5.35 and 5.36 show the loss curves and accuracy curves during training and validation phases for each fold.
As shown in figures, the training and validation curves of our ViT model exhibit distinct behaviours. During the training phase, the loss decreases rapidly, stabilizing at very low values, while accuracy increases sharply and then plateaus around 97-99% for all folds. This indicates strong learning and effective performance during the training step. However, the validation loss and accuracy curves show more fluctuation. In detail, the validation loss tends to decrease, as expected, but we can observe many oscillations. The validation accuracy increases and stabilizes around 96-98%, with occasional drops to approximately 94%. These fluctuations likely reflect natural variations in model generalization, where each fold is trained on a slightly different subset of data. Despite these oscillations, the validation accuracy remains consistently high and the loss curves tend to zero. The use

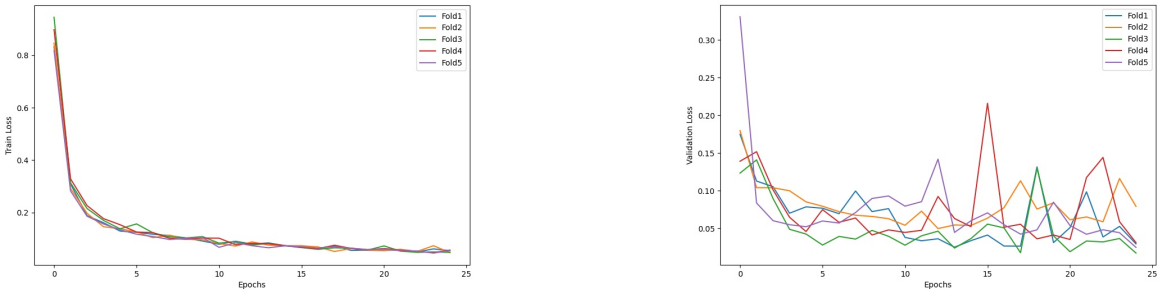


Figure 5.35: Loss curves during training (left) and validation (right) for each fold for short volume check.

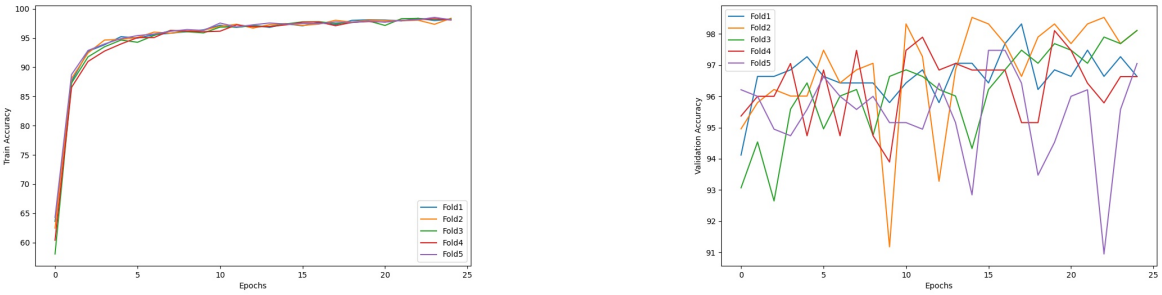


Figure 5.36: Accuracies during training (left) and validation (right) for each fold for short volume check.

of techniques such as dropout and weight decay likely helped mitigate overfitting, ensuring that the performance gap between the training and validation phases remains minimal.
Table 5.12 presents the accuracies achieved during validation for each fold. The last column represents the average accuracy computed by averaging across all folds.

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean Accuracy
96.64%	98.11%	98.11%	96.64%	97.05%	97.31% ± 0.67%

Table 5.12: Average accuracies for each fold and overall mean accuracy.

By selecting the best model, Figure 5.37 shows the confusion matrix obtained on the validation set. Our model achieved an accuracy of 98.11% and a macro F1-score of 98.1%, demonstrating that is capable on distinguishing between different liquid volumes. Given that our model addresses a multi-class classification task and considering the computational complexity involved in generating the ROC curve, we chose not to compute it, focusing instead on accuracy, macro F1-score and the confusion matrix. Finally, the few observed errors do not compromise the diagnostic outcome but only lead to a slight slowdown of the workflow due to the need of repeating the pipetting operation on the affected channel.

Explainability

As with CNN, an analysis of the internal mechanisms was also performed for our ViT model. In this case, however, due to the different architecture of the ViT, the analysis focused on attention maps, as the model relies on attention mechanisms rather than convolutional feature maps. Figures 5.38, 5.39, 5.40, 5.41, 5.42, 5.43, 5.44, 5.45 and 5.46 illustrate the original image (left) and the

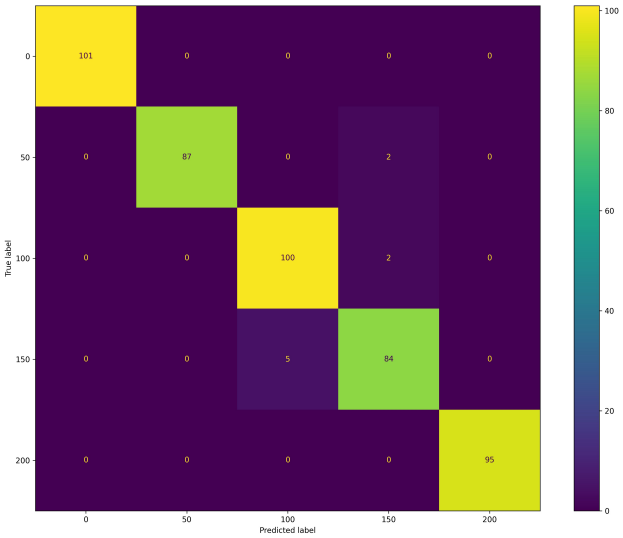


Figure 5.37: Confusion matrix of the best ViT model on the validation set.

attention map (right) for the following cases: 0 μL, 50 μL transparent liquid, 50 μL red liquid, 100 μL transparent liquid, 100 μL red liquid, 150 μL transparent liquid, 150 μL red liquid, 200 μL transparent liquid, and 200 μL red liquid, respectively.

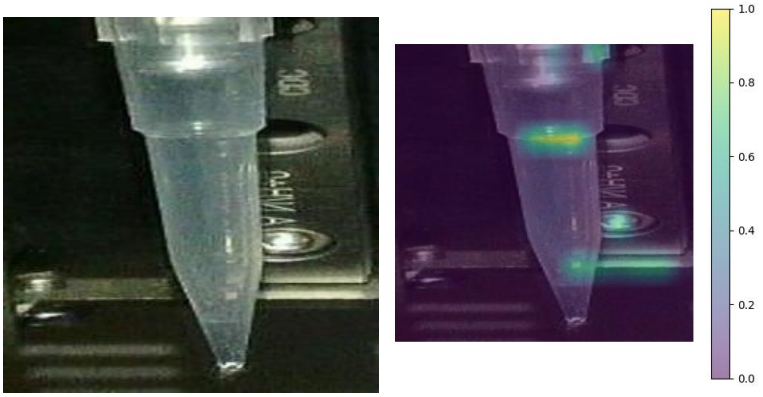


Figure 5.38: Original image (left) and attention map (right) for the case with 0 μL

Upon examining the attention maps, we can observe that, in all cases, the model predominantly focuses on a specific point at the base of the tip. This high attention to this point may be due to the significant contrast or brightness variation at the tip base, which the model considers important. For tips filled with liquid, both transparent and red, the model's focus shifts slightly to the liquid itself. However, this attention is not uniform, as reflected by the blue-purple colors, indicating less certainty in those areas. The only exception occurs in the case 100 μL volume, where the model highlights the the region between the base of the tip and the liquid level. This suggests that the model might encounter some difficulty interpreting the liquid in this specific region. Finally, the model appears to focus on reflections and areas of high contrast in the background. Reflections on the tip or nearby surfaces may provide the model with additional cues for accurate classification of the scene.

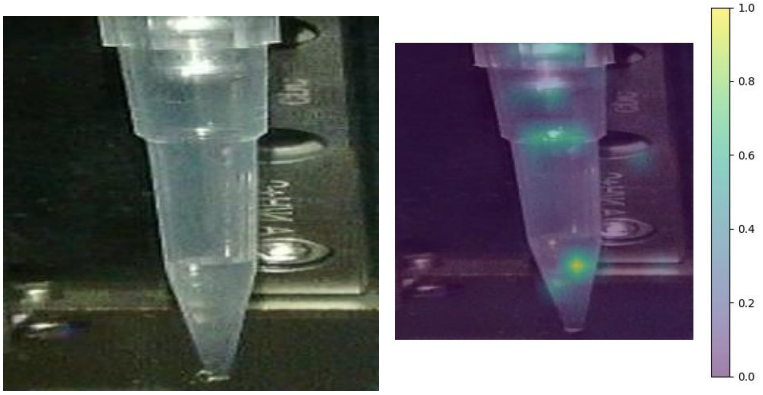


Figure 5.39: Original image (left) and attention map (right) for the case with 50 μL transparent liquid.

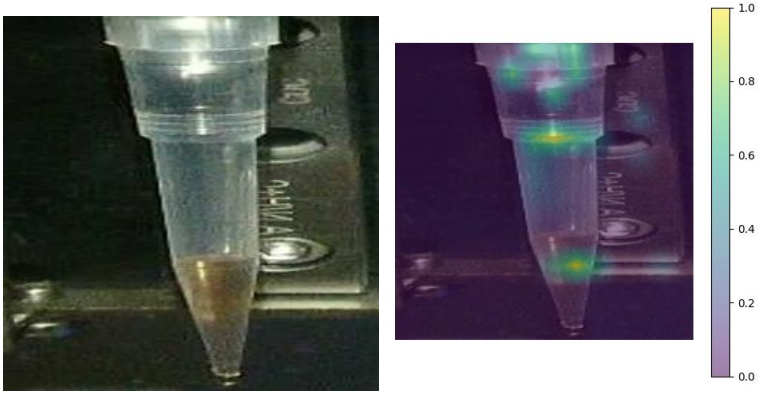


Figure 5.40: Original image (left) and attention map (right) for the case with 50 μL red liquid.

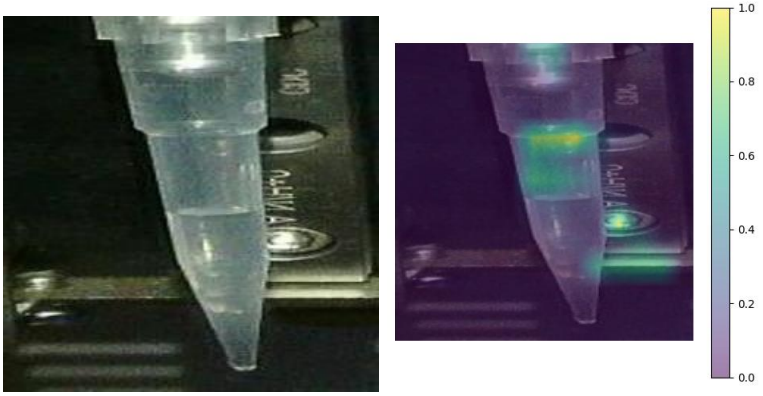


Figure 5.41: Original image (left) and attention map (right) for the case with 100 μL transparent liquid.

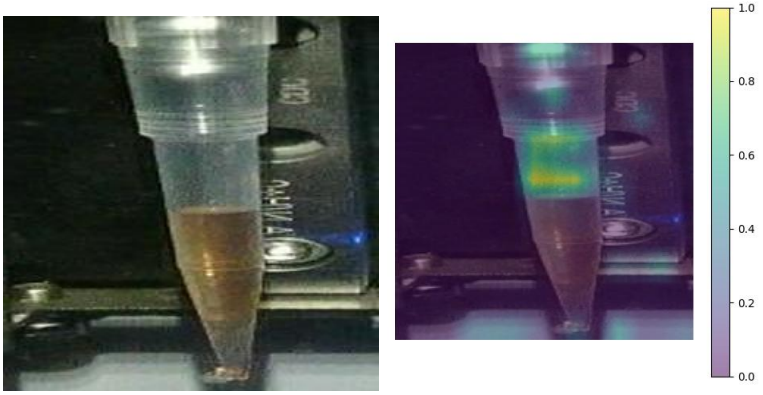


Figure 5.42: Original image (left) and attention map (right) for the case with 100 μL red liquid.

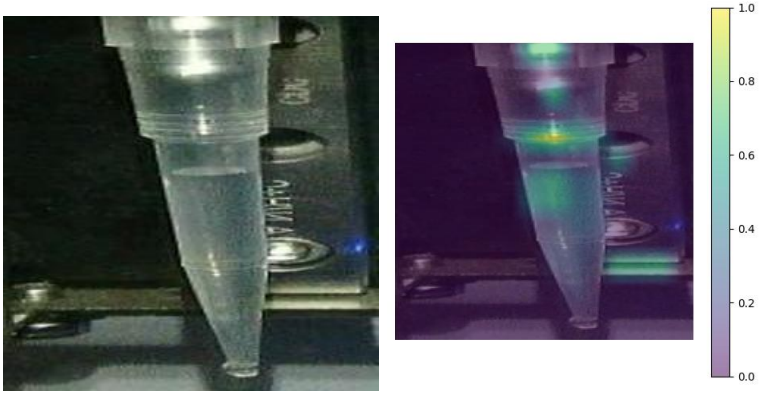


Figure 5.43: Original image (left) and attention map (right) for the case with 150 μL transparent liquid.

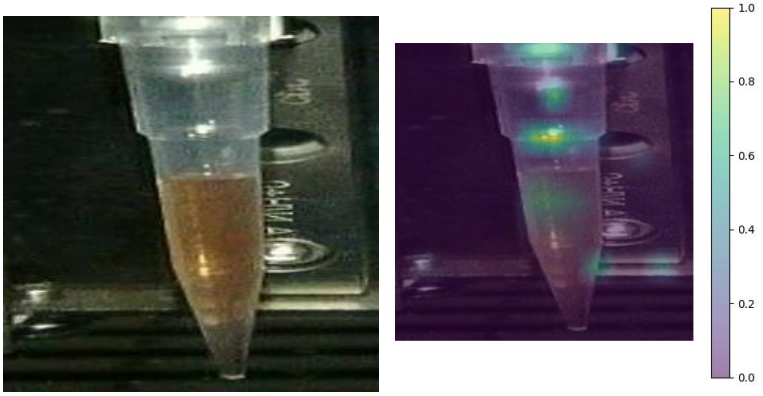


Figure 5.44: Original image (left) and attention map (right) for the case with 150 μL red liquid.

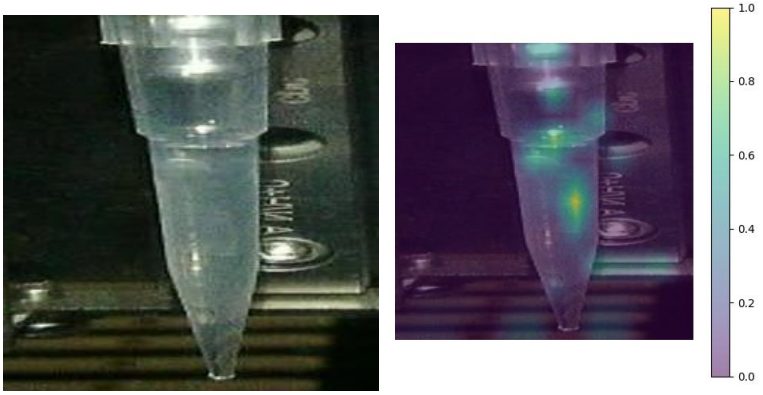


Figure 5.45: Original image (left) and attention map (right) for the case with 200 μL transparent liquid.

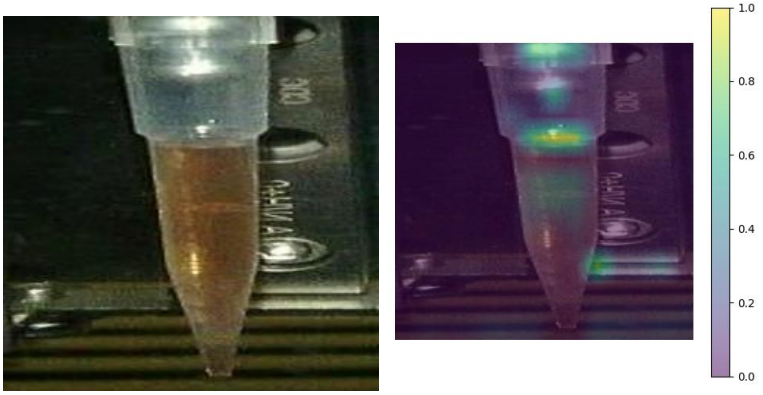


Figure 5.46: Original image (left) and attention map (right) for the case with 200 μL red liquid.

Evaluation on an external test set

To investigate potential training-related bias and assess the model's ability to generalize, we evaluated the ViT model on a test set of images. This test set is the same used for the CNN model and introduces in Section 5.6.1. Figure 5.47 shows the confusion matrix obtained on the test set. Our ViT model achieved an accuracy of 98.95% and a macro F1-score of 98.93%, meaning that it

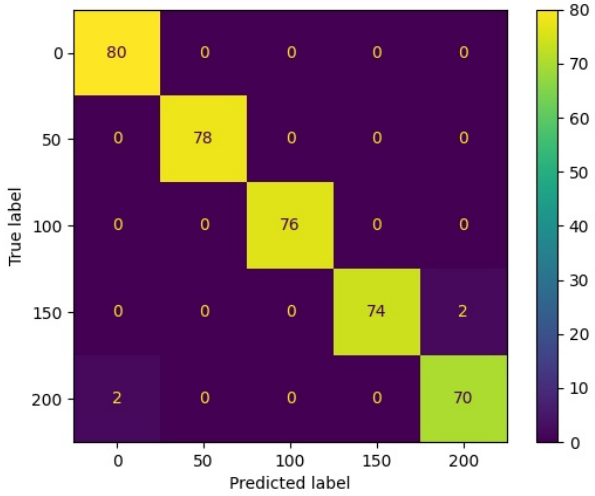


Figure 5.47: Confusion matrix of the ViT model on the test set.

is well-suited for the short volume check. Importantly, The few misclassifications observed do not affect the diagnostic outcome, but only result in a slight slowdown of the workflow due to the need to repeat the pipetting operation for the affected channel. Furthermore we computed the attention map on the test set. Figures 5.48, 5.49, 5.50, 5.51, 5.52, 5.53, 5.54, 5.55 and 5.56 illustrate the original image (left) and the attention map (right) for the following cases: 0 μL, 50 μL transparent liquid, 50 μL red liquid, 100 μL transparent liquid, 100 μL red liquid, 150 μL transparent liquid, 150 μL red liquid, 200 μL transparent liquid, and 200 μL red liquid, respectively.

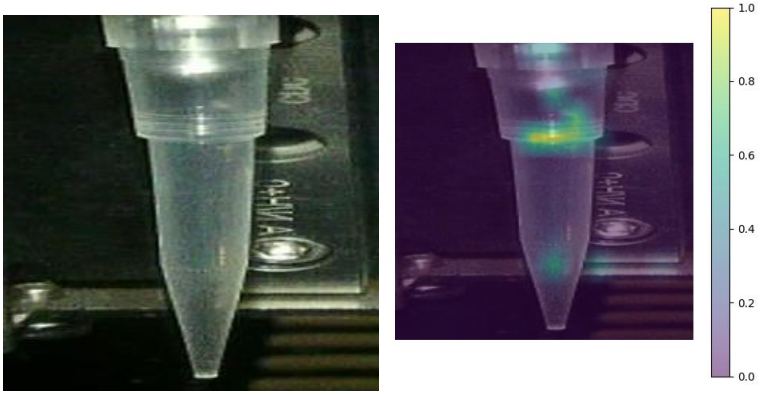


Figure 5.48: Original image (left) and attention map (right) for the case with 0 μL

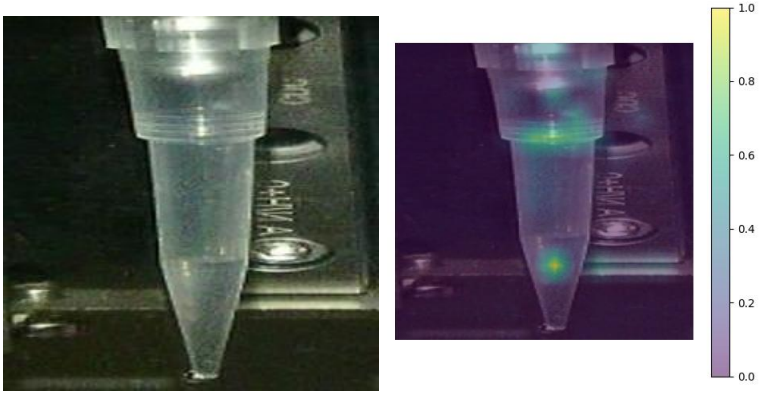


Figure 5.49: Original image (left) and attention map (right) for the case with 50 μL transparent liquid.

The results obtained during the testing phase confirm the observations made during training. Indeed, we also see a concentration of the model's attention on the tip base across all volumes. Furthermore, the model continues to focus on the liquid inside the tip, for both transparent and red liquids. In the case of the 100 μL volume, we again observe that the model concentrates on the area between the base of the tip and the liquid level. Finally, as seen during training, the model seems to pay particular attention to reflections and areas with high contrast. This suggests that the model continues to use information from reflective surfaces to improve its prediction.

Time performance

Finally, we measured the time required to perform pre-processing and classification steps. Table 5.13 presents the average pre-processing, prediction and total times, along with the corresponding standards deviations for our model. The average times were calculated on the 384 images of the

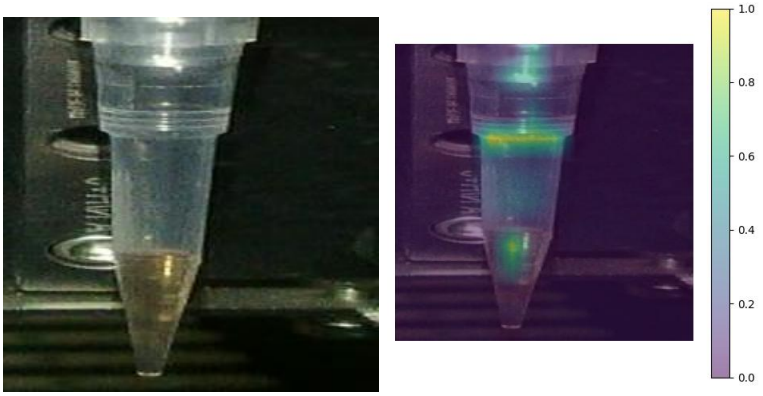


Figure 5.50: Original image (left) and attention map (right) for the case with 50 μL red liquid.

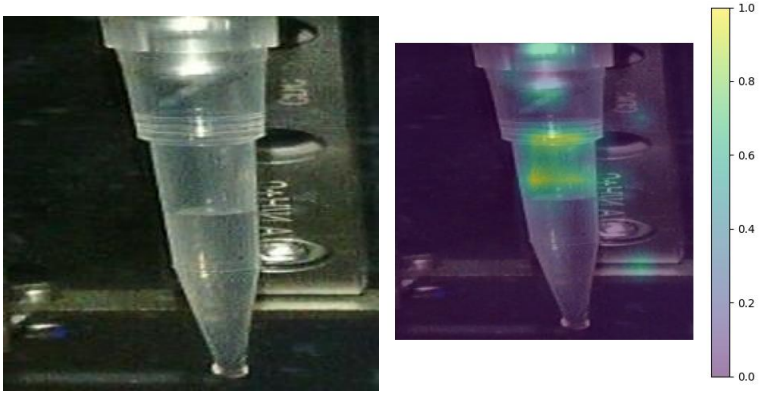


Figure 5.51: Original image (left) and attention map (right) for the case with 100 μL transparent liquid.

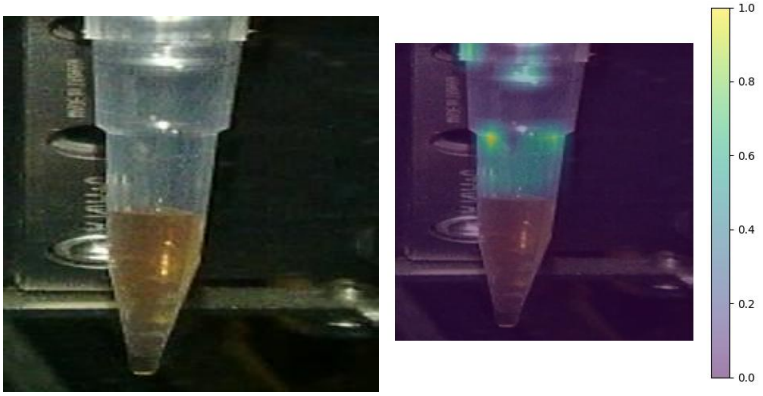


Figure 5.52: Original image (left) and attention map (right) for the case with 100 μL red liquid.

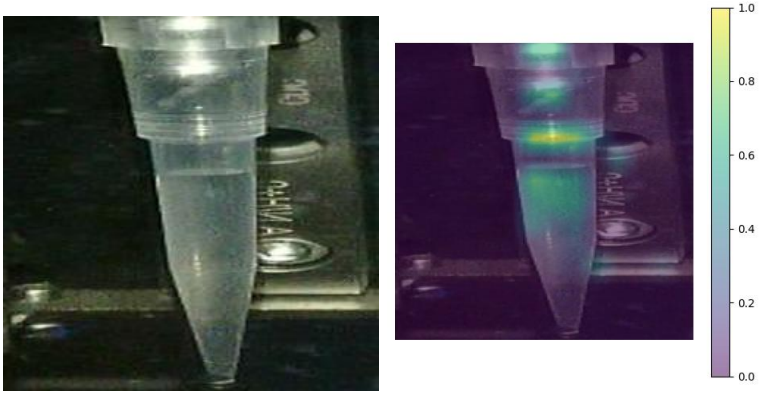


Figure 5.53: Original image (left) and attention map (right) for the case with 150 μL transparent liquid.

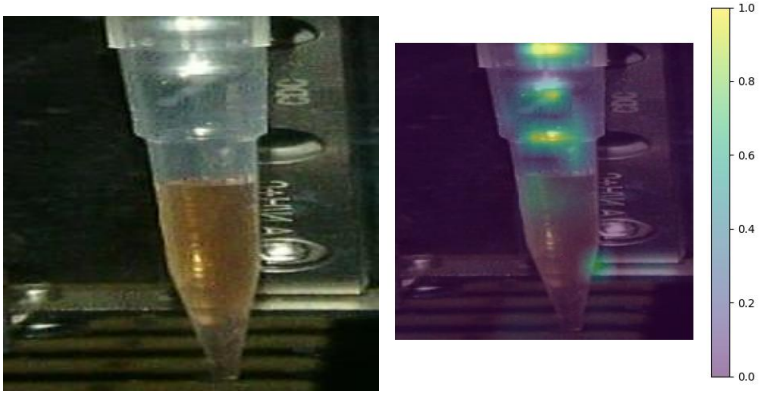


Figure 5.54: Original image (left) and attention map (right) for the case with 150 μL red liquid.

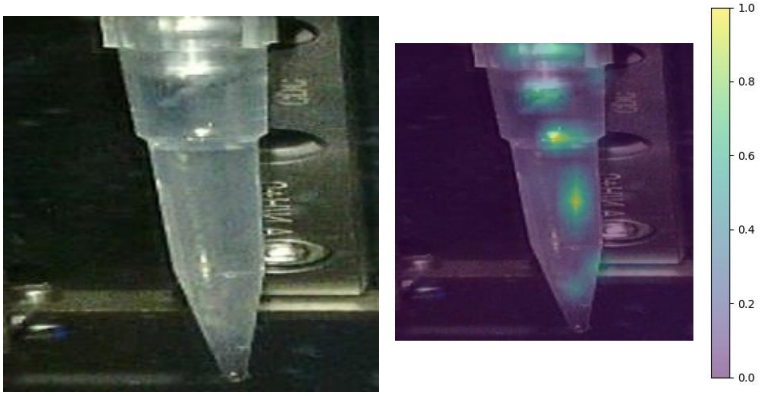


Figure 5.55: Original image (left) and attention map (right) for the case with 200 μL transparent liquid.

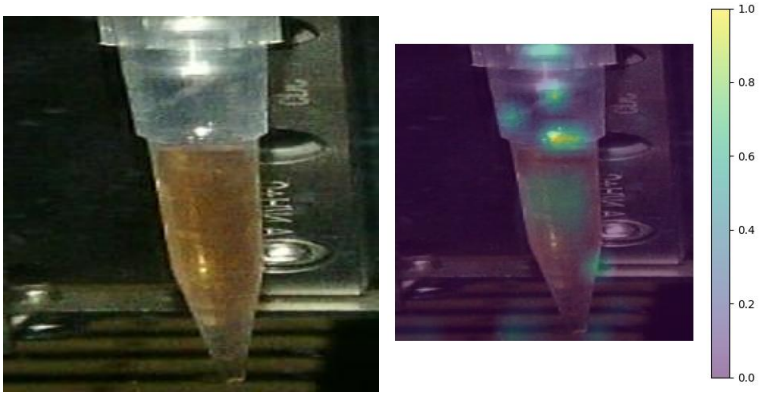


Figure 5.56: Original image (left) and attention map (right) for the case with 200 μL red liquid.

test set, and performed on the system with the hardware identical to the one integrated used in the IVD device. The total execution time, which is $3.411 \cdot 10^{-1}$ seconds, falls well within the required

Function	Time
Pre-processing	$1.058 \cdot 10^{-2} \pm 7.779 \cdot 10^{-3}$ s
Prediction	$3.305 \cdot 10^{-1} \pm 4.357 \cdot 10^{-2}$ s
Total	$3.411 \cdot 10^{-1} \pm 4.314 \cdot 10^{-2}$ s

Table 5.13: Average extraction, prediction, and total times with standard deviations for the selected model.

standards for the task, being well below the 1-second threshold.

5.6.3 Models comparison

After presenting the results obtained with both approaches, we compared the two models to determine the most suitable one for the short volume check application. The comparison was based on three main criteria: accuracy achieved during validation and testing phases, model interpretability, and execution time.
Regarding validation accuracy, the CNN model outperformed the ViT model, achieving 99.16% compared to 98.11%. However, during the testing phase, the ViT model demonstrated a higher accuracy (98.95%) compared to the CNN model (95.03%). Consistently, macro F1-scores confirmed the overall robustness of both models, mitigating the possible effect of class imbalance introduced by the k-fold procedure.
In terms of interpretability, the results obtained with XAI methods revealed some key differences between the models. Specifically, the ViT model did not appear to focus on the relevant parts of the image, particularly the liquid inside the tip. While the IG maps for the CNN were sometimes weak and less informative, the Grad-CAM visualizations consistently highlighted the relevant liquid regions with greater clarity, providing a more reliable interpretation of the model's decision process. Indeed, in all liquid-filled cases, the CNN model consistently focuses either on the entire liquid region (in the case of red liquid) or specifically on the liquid level (in the case of transparent liquid).
Lastly, considering the execution time, the CNN model was faster in completing the task compared to the ViT model, which is an important factor for real-time applications. Based on these findings, we decided to select the CNN model for the execution of the short volume check, given its better balance of performance, interpretability, and execution time.

5.7 Long volume check results

5.7.1 CNN approach

The CNN model for the long volume check was trained on 5296 images, obtained after data augmentation application, and evaluated on 328 images. The model was trained using 5-fold-cross-validation. Figures 5.57 and 5.58 show the loss curves and accuracy curves during the training and the validation phases for each fold.

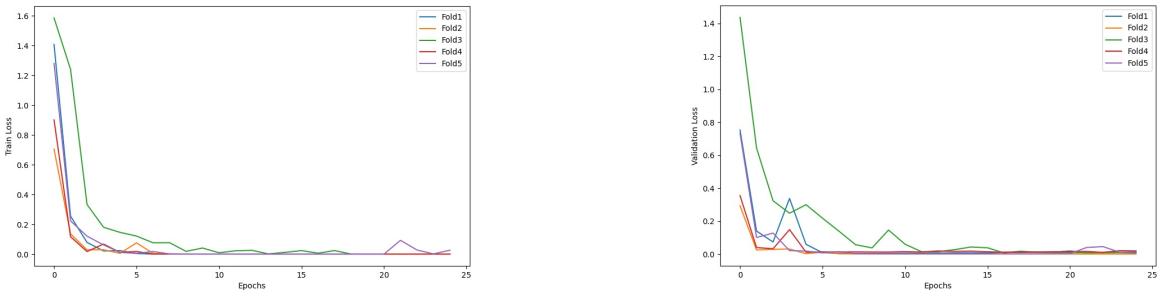


Figure 5.57: Loss curves during training (left) and validation (right) for each fold for long volume check.

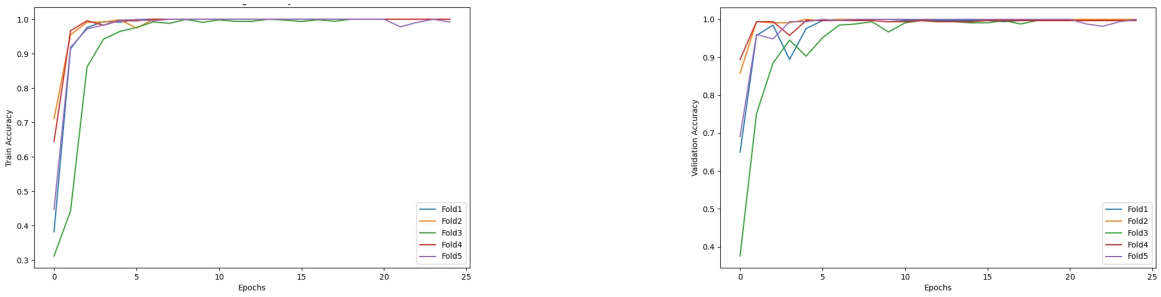


Figure 5.58: Accuracies during training (left) and validation (right) for each fold for long volume check.

By analysing the loss curves during the training phase, we observe that all curves show a rapid decrease in the early epochs, stabilizing at very low values close to zero. There are oscillations at the beginning of the training, which could be attributed to the model's sensitivity to variations between folds. A similar pattern is seen during the validation phase, where fluctuations continue until the tenth epoch, with the exception of Fold 5, which shows a peak towards the end of the training. This could be due to a particular batch of images encountered during training, introducing some variability in the model's performance.
Regarding the accuracy curves during training, we observe that curve increase rapidly and stabilize around 98-99%. The slight fluctuations at the beginning of the training could be an indication of overfitting. The peak towards the end of the training for Fold 5, which does not exceed 90%, may indicate a difficulty of the model in adapting to a specific batch of images. For the validation accuracy curves, we observe a very similar pattern to that of the training phase. The curves rise quickly and stabilize around the tenth epoch, reaching an accuracy of 98-99%. As with the training phase, slight fluctuations can be observed during the early epochs, and a small peak appears towards the end of training. This behavior may be attributed to small differences between folds. Since the accuracies stabilize at high values, it indicated that the model generalizes well and has effectively learned the features of the data.

To better evaluate the model's overall performance, we computed the average accuracy achieved at the end of validation phase. Table 5.14 shows these results. The last column represents the average accuracy obtained by averaging across all folds. By selecting the best model, Figure 5.59 shows the

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean Accuracy
99.70%	100%	99.70%	99.70%	100%	99.82% ± 0.15%

Table 5.14: Average accuracies for each fold and overall mean accuracy.

confusion matrix obtained on the validation set. The model achieved 100% accuracy and a macro F1-score of 100%, confirming its behaviour as a perfect classifier. As for short volume check CNN model, since our model was designed for a multi-class classification problem and computational challenges to build the ROC curve, we decided to not computed the ROC curve, but we opted to focus only on accuracy and confusion matrix.

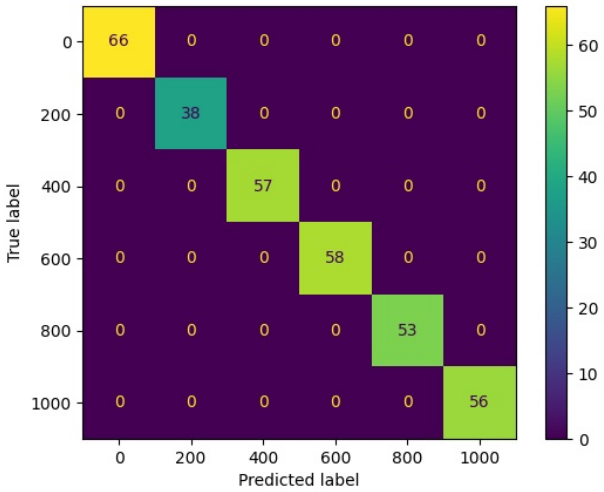


Figure 5.59: Confusion matrix of the best CNN model on the validation set.

Explainability

To better understand the model performance, we performed a feature map analysis, using the IG and Grad-CAM methods. Figures 5.60, 5.61, 5.62, 5.63, 5.64, 5.65, 5.66, 5.67, 5.68, 5.69, and 5.70 illustrate the original image (left), the IG map (center) and the Grad-CAM heatmap (right) for the following cases: 0 μL, 200 μL transparent liquid, 200 μL red liquid, 400 μL transparent liquid, 400 μL red liquid, 600 μL transparent liquid, 600 μL red liquid, 800 μL transparent liquid, 800 μL red liquid, 1000 μL transparent liquid and, 1000 μL red liquid, respectively.
Analysing the IG maps, we observe that the model correctly focuses its attention on the most relevant regions of the image, specifically, on areas corresponding to the liquid or its level. An exception is observed in the case of empty tip, where the model appears to attribute importance to the outline of the tip and, in some instances, to certain background regions. For tips containing liquid, the behavior of the model varies based on the color of liquid. In particular, in the presence of transparent liquid, the model assigns positive attribution mainly to the liquid level. In contrast, for

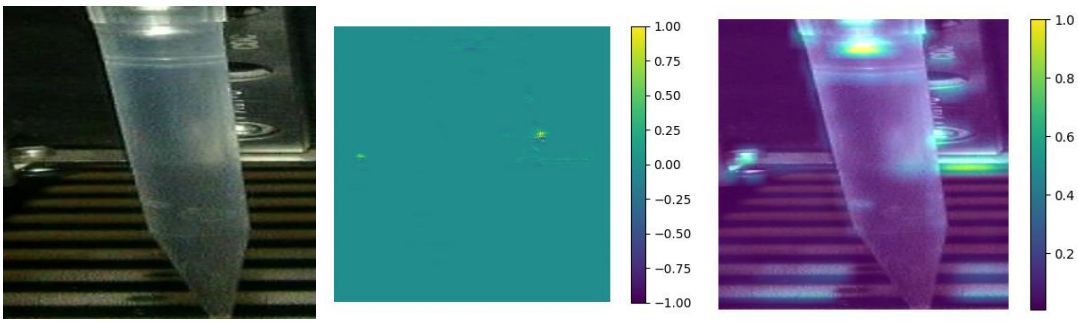


Figure 5.60: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 0 μL.

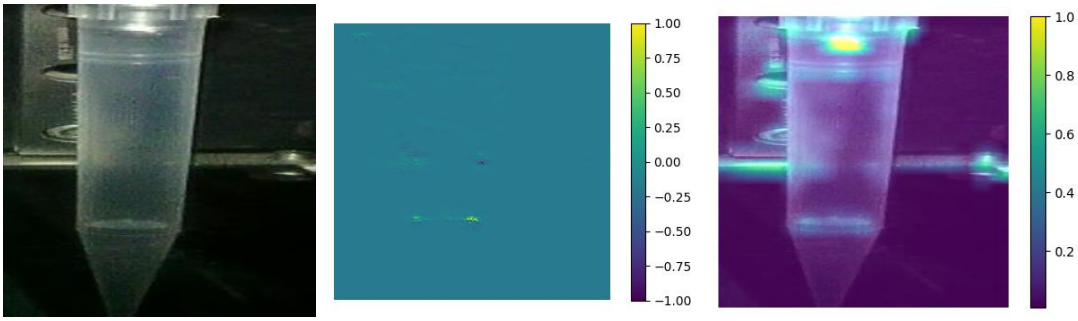


Figure 5.61: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 200 μL transparent liquid.

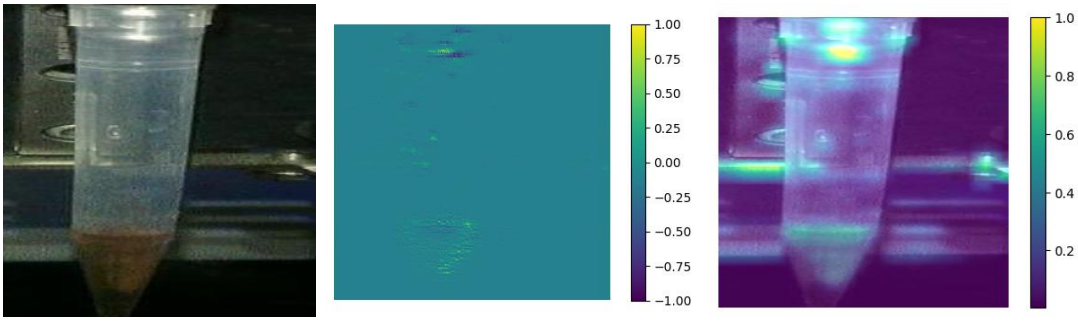


Figure 5.62: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 200 μL red liquid.

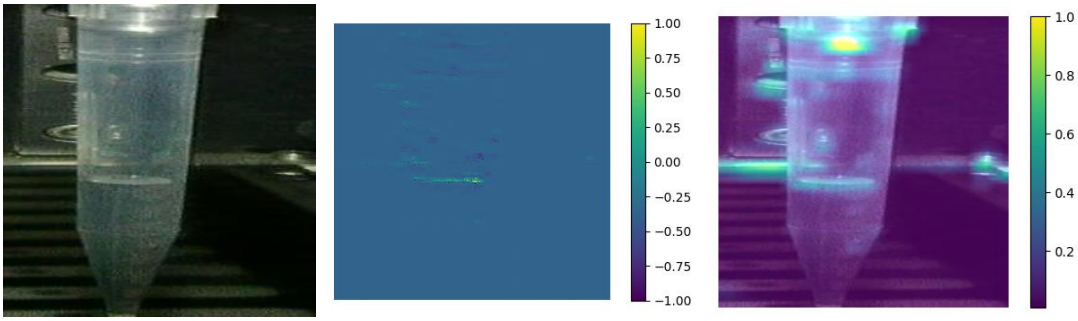


Figure 5.63: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 400 μL transparent liquid.

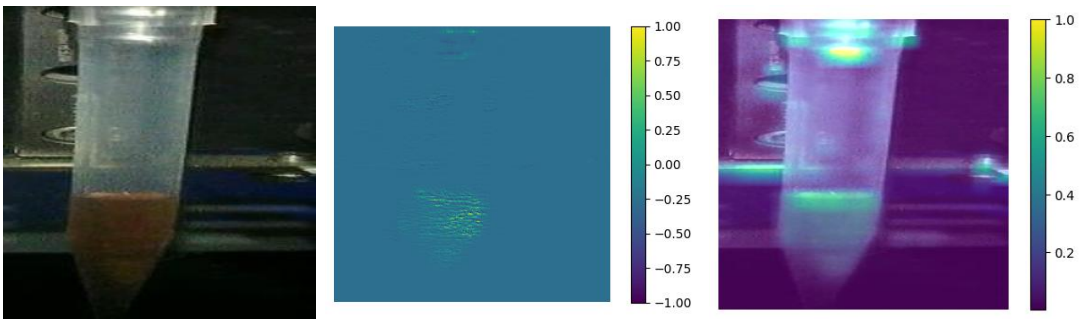


Figure 5.64: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 400 μL red liquid.

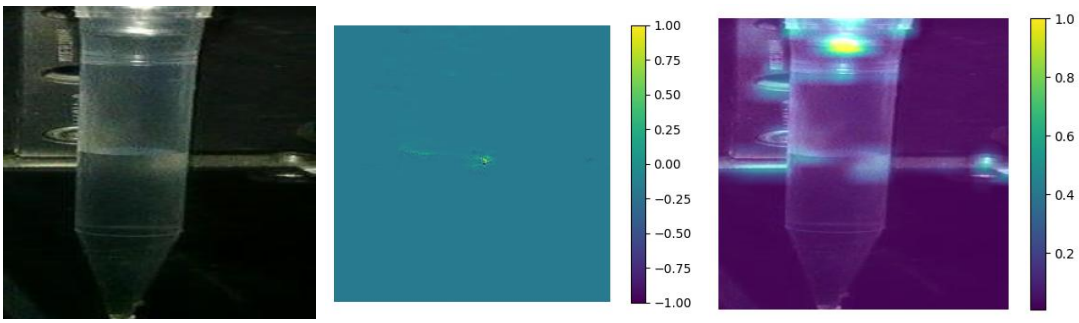


Figure 5.65: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 600 μL transparent liquid.

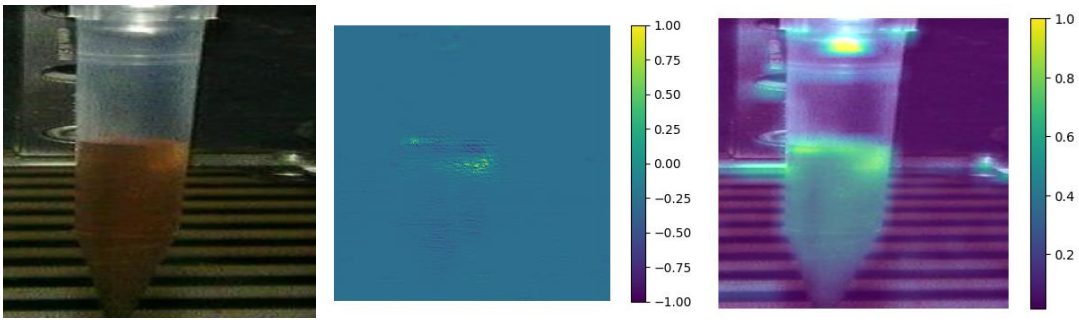


Figure 5.66: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 600 μL red liquid.

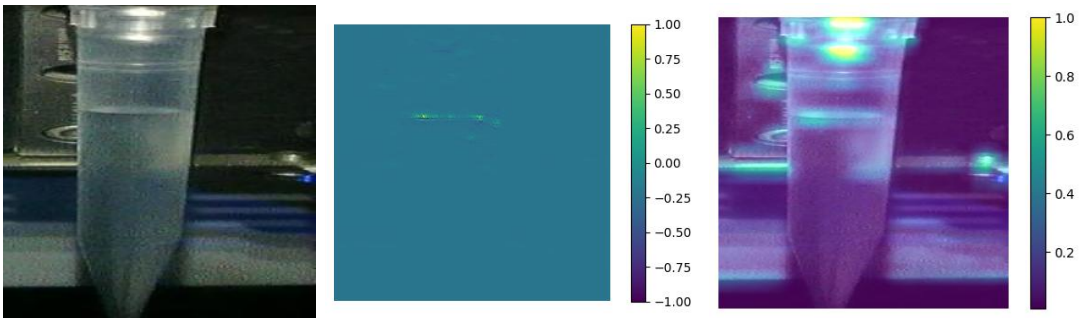


Figure 5.67: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 800 μL transparent liquid.

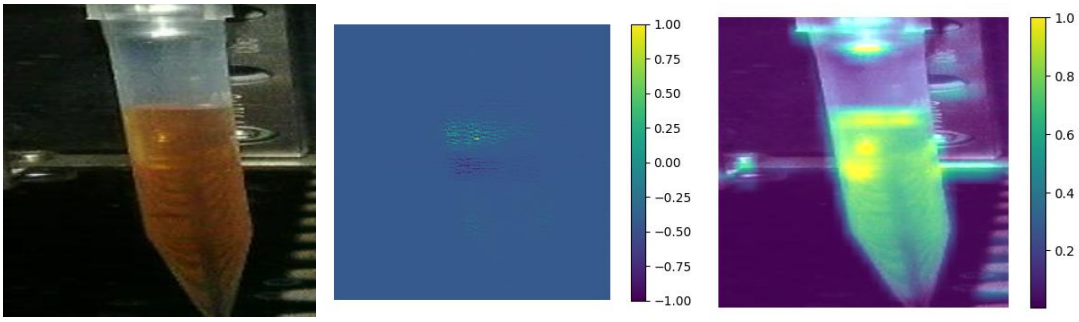


Figure 5.68: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 800 μL red liquid.

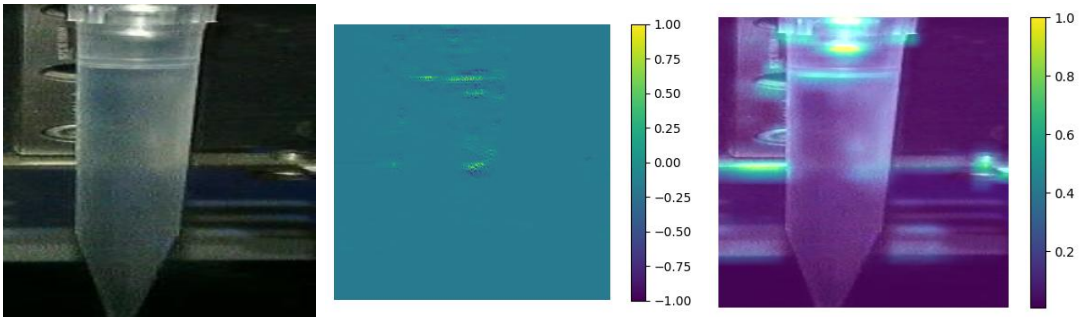


Figure 5.69: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 1000 μL transparent liquid.

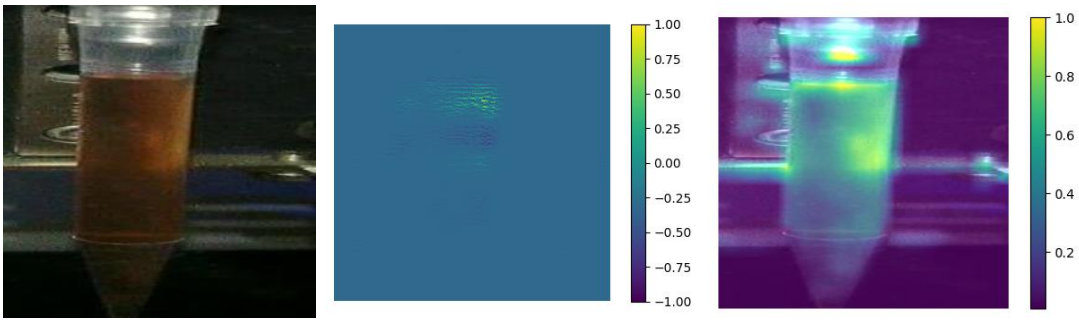


Figure 5.70: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 1000 μL red liquid.

red liquid, we observe positive attribution across the entire liquid region inside the tip. However, for larger volumes, such as 600, 800 and 1000 μL, the attention appears to shift back to the liquid level. the model assigns positive attribution mainly to the liquid level. In contrast, for red liquid, we observe positive attribution across the entire liquid region inside the tip.
Regarding the Grad-CAM heatmaps, results confirm these observations. Indeed, for empty tip, the model's focus is distributed across various regions, including parts of the background. For liquid-filled tip, the model consistently highlights the liquid. Specifically, for transparent liquid, the prediction is most influenced by the liquid level, while for red liquid the CNN focuses on the entire volume of liquid. In samples with higher volume, such as 800 and 1000 μL, the attention becomes even more pronounced. These differences between transparent and red liquids may be explained by the higher contrast offered by red liquid, which makes the entire liquid volume more visually salient compared to the subtler boundary of transparent liquid, where the level is more distinguishable.
Moreover, as observed in the short volume classification task, the model also assigns importance to reflections within the image, particularly those occurring near the tip base and background elements with high contrast.
In this specific configuration, the two attribution methods appear to converge, both in terms of spatial relevance and interpretability. This consistency could be favored by the stronger contrast between the red liquid and the tip background, as well as by the larger horizontal cross-section of the long tips, which may help highlight key visual features more clearly. It is worth noting that these patterns are consistently observed across the entire dataset, indicating a stable and interpretable decision-making process.

Evaluation on an external test set

In order to evaluate the model's applicability in real-world conditions, its performance was tested on a dataset comprising images not included in either the training or validation stages. This test set consists of 438 images, distributed as shown in Figure 5.71, and was acquired with the same acquisition conditions of the short volume external test set. Figure 5.72 shows the confusion matrix obtained on the test set. The CNN achieved an accuracy of 95.89% and a macro F1-score of 95.90%, demonstrating that the model has excellent performance for the long volume check task. The few misclassifications observed at this stage do not compromise the diagnostic outcome, but only slow down the device workflow, as each error requires the pipetting operation to be repeated.
To assess alignment with the results obtained during the testing phase, we performed a feature map analysis using the two attribution methods previously employed, IG and GradCAM. Figures 5.73, 5.74, 5.75, 5.76, 5.77, 5.78, 5.79, 5.80, 5.81, 5.82, and 5.83 illustrate the original image (left), the IG map (center) and the Grad-CAM heatmap (right) for the following cases: 0 μL, 200 μL transparent liquid, 200 μL red liquid, 400 μL transparent liquid, 400 μL red liquid, 600 μL transparent liquid, 600 μL red liquid, 800 μL transparent liquid, 800 μL red liquid, 1000 μL transparent liquid and, 1000 μL red liquid, respectively.
By analysing all the IG maps and Grad-CAM heatmaps, we observe consistency between the results obtained during the training and testing phases. In detail, the IG maps reinforce the results previously obtained. Specifically, in the empty case, a generally sparse focus is observed, with greater attribution around the reflections. For tips filled with liquid, a positive attribution remains concentrated at the liquid level; as the liquid volume increases, particularly for the red liquid, the

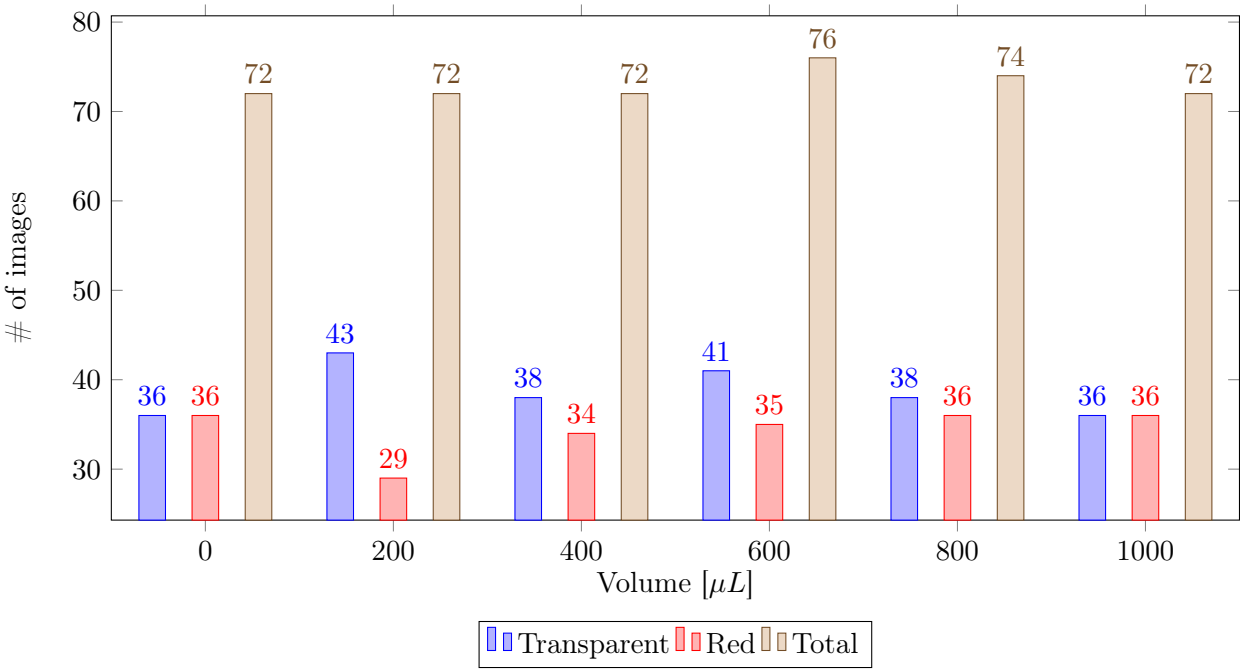


Figure 5.71: Distribution of the test set for the long volume across different liquid volumes and colors.

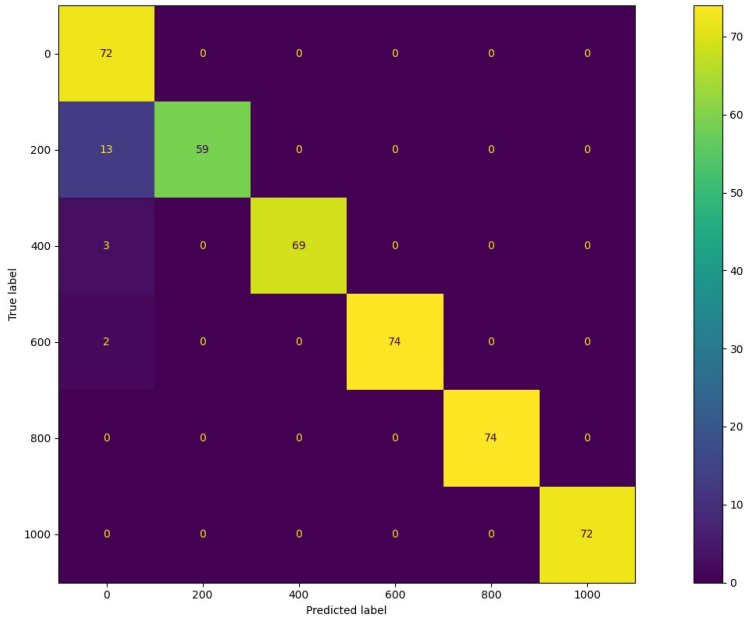


Figure 5.72: Confusion matrix of CNN model on the test set.

attribution becomes distributed across the entire liquid area, showing both positive and negative contributions.

Grad-CAM heatmaps show that as the volume of red liquid increases, the model's attention becomes increasingly concentrated on the liquid, as reflected by more intense green-yellow regions in the heatmap. These differences in the model's focus between transparent and red liquids may be due to the higher contrast of red liquid compared to transparent liquid, making the red liquid more visually distinguishable. Additionally, as with the previous analysis for training and validation images, the model also seems to focus on reflections present on the tip base and in background areas where contrast is particularly high.

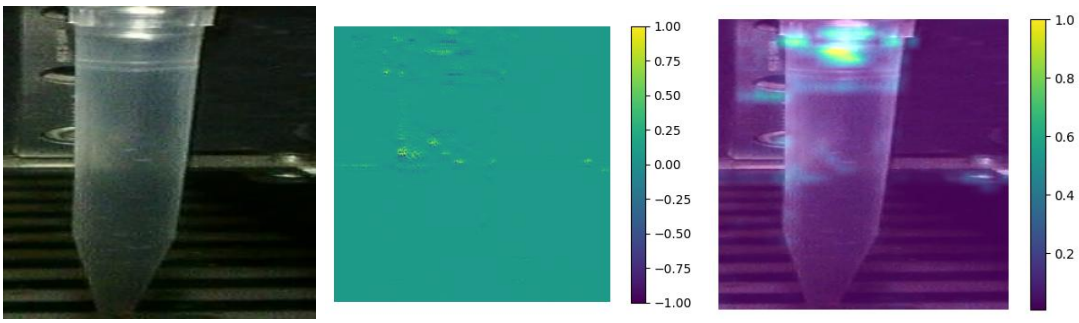


Figure 5.73: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 0 μL.

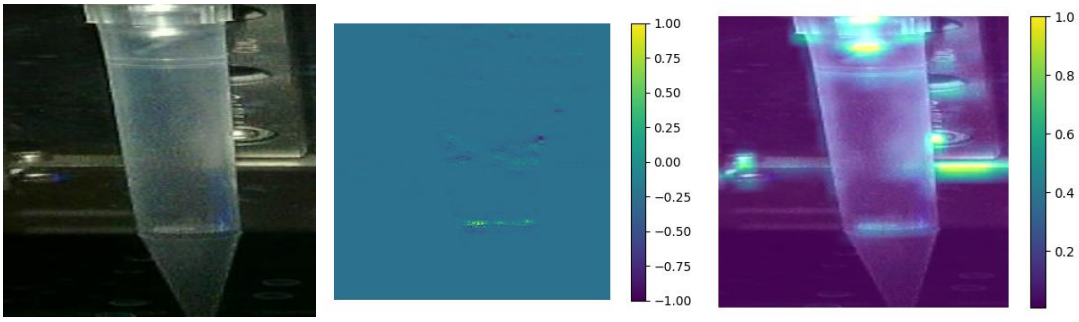


Figure 5.74: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 200 μL transparent liquid.

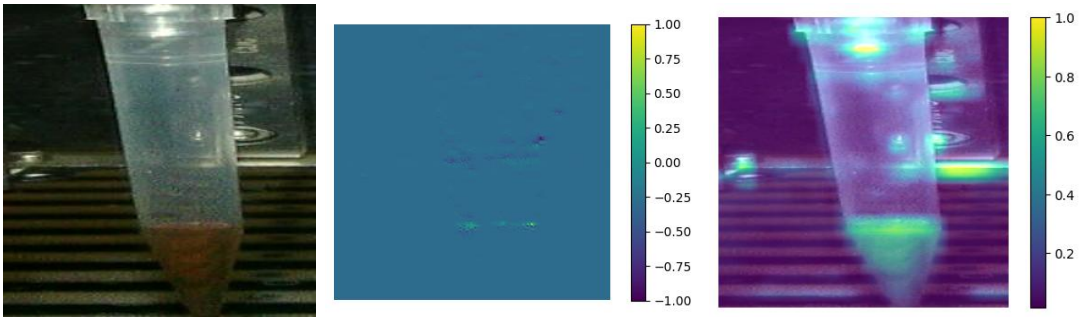


Figure 5.75: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 200 μL red liquid.

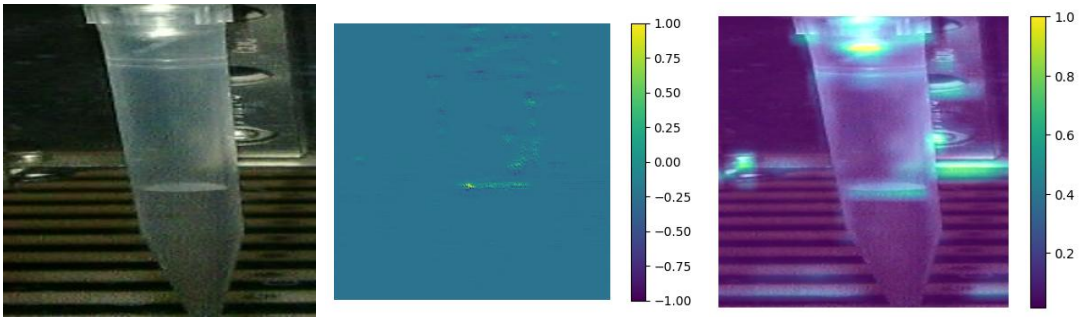


Figure 5.76: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 400 μL transparent liquid.

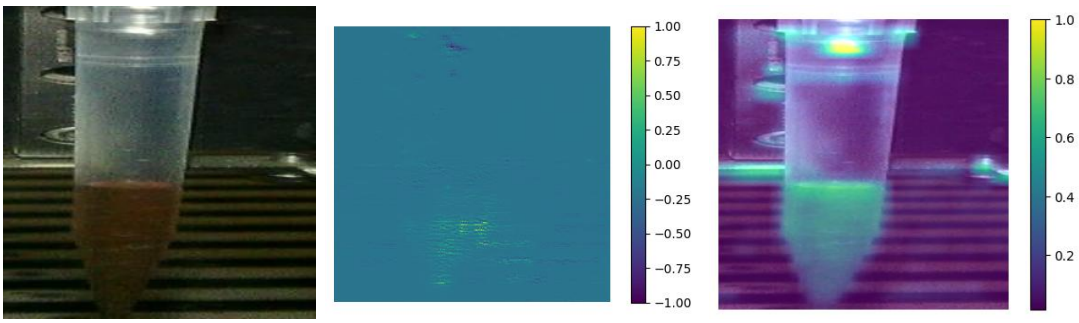


Figure 5.77: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 400 μL red liquid.

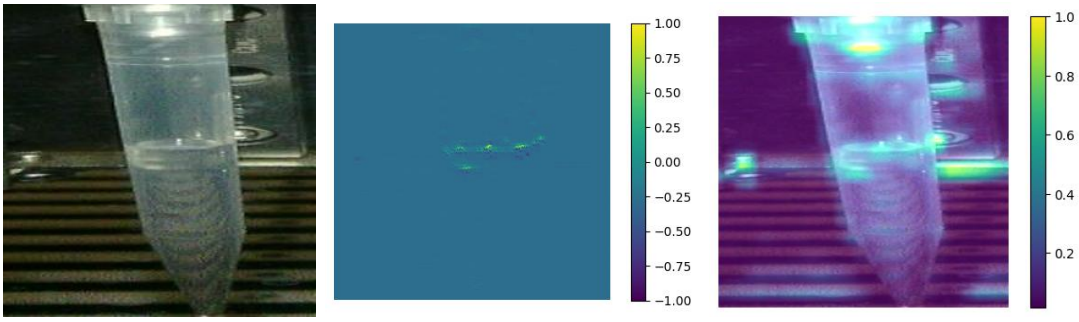


Figure 5.78: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 600 μL transparent liquid.

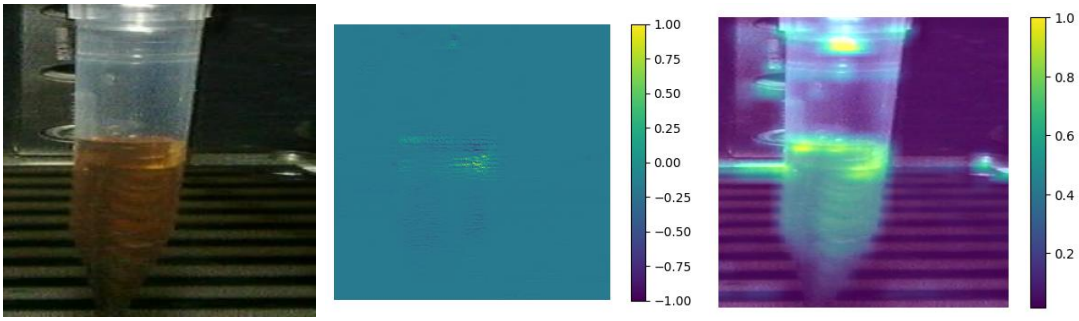


Figure 5.79: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 600 μL red liquid.

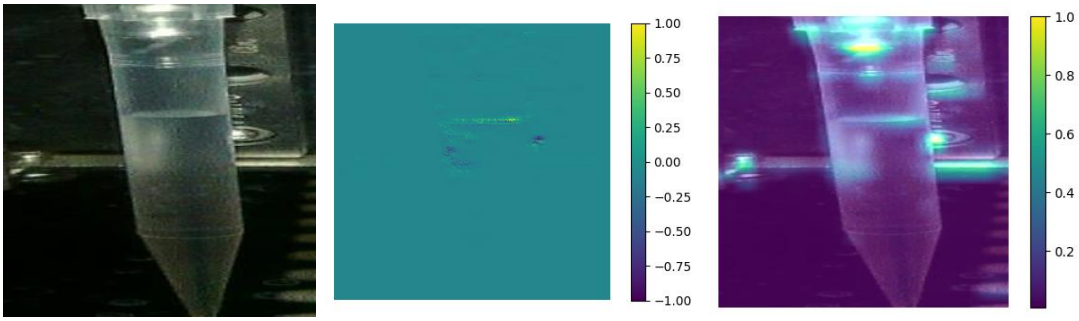


Figure 5.80: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 800 μL transparent liquid.

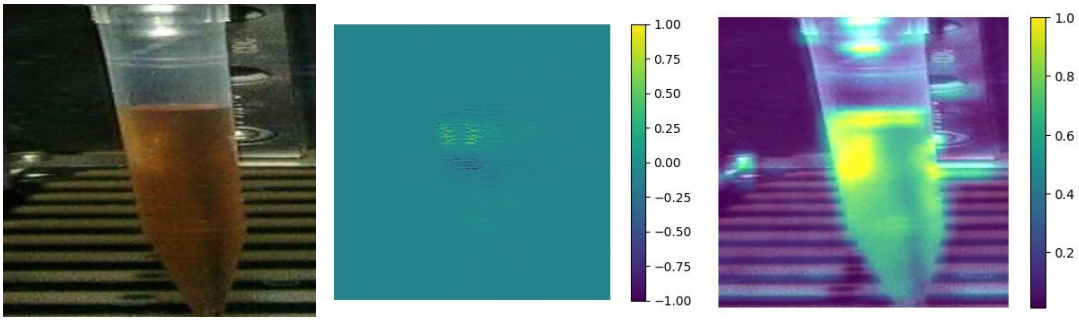


Figure 5.81: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 800 μL red liquid.

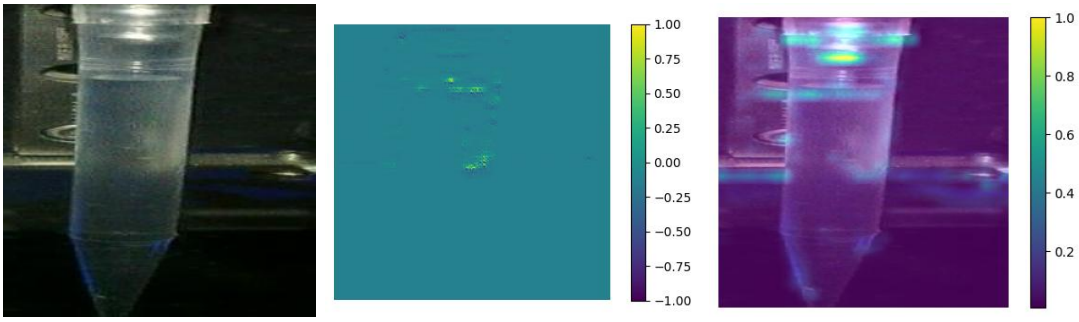


Figure 5.82: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 1000 μL transparent liquid.

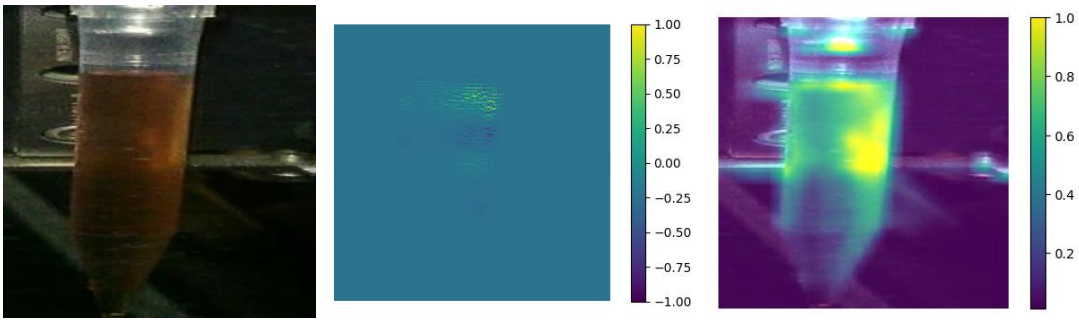


Figure 5.83: Original image (left), IG map (center), and Grad-CAM heatmap (right) for the case with 1000 μL red liquid.

These results indicate a high degree of consistency across the entire test set, confirming that the model maintains its ability to generalize well to unseen data.

Time performance

Finally, we evaluated the computational time of the CNN model during the pre-processing and prediction steps. Table 5.15 shows the average pre-processing, classification and total times, along the corresponding deviations. Times were calculated on the 438 images of the test set, and performed on system hardware identical to the each one integrated in the IVD device.

Function	Time
Pre-processing	$5.824 \cdot 10^{-3} \pm 7.649 \cdot 10^{-3}$ s
Prediction	$9.082 \cdot 10^{-2} \pm 4.502 \cdot 10^{-2}$ s
Total	$9.664 \cdot 10^{-2} \pm 4.926 \cdot 10^{-2}$ s

Table 5.15: Average extraction, prediction, and total times with standard deviations for the selected model.

Since the maximum allowed time for performing the check is 1 second, and the total time taken by the model to complete the check is $9.664 \cdot 10^{-2}$ seconds, it follows that we are within the required time limits, even when accounting for the standard deviation.

5.7.2 ViT approach

Evaluation on training and validation set

The ViT model for the long volume check was trained and evaluated using the same dataset employed for the CNN model, meaning that the model was trained on 7624 images, and evaluated on the remaining 472 images. Figures 5.84 and 5.85 present the loss and accuracy curves during the training and the validation for each fold.

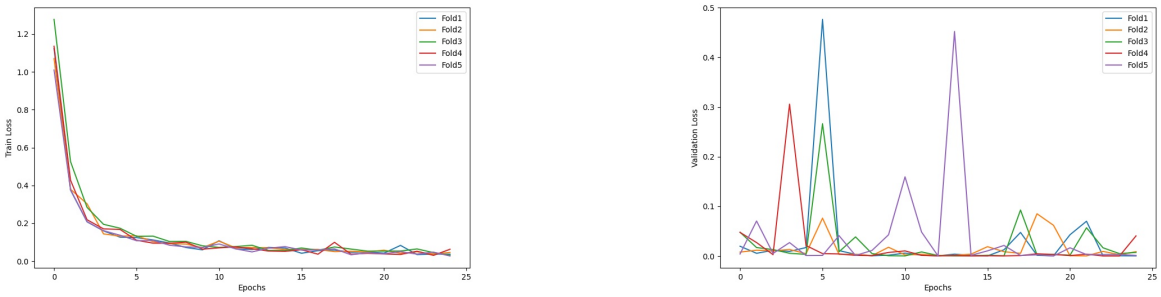


Figure 5.84: Loss curves during training (left) and validation (right) for each fold for long volume check.

Figures show two distinct trends between the training and validation phases. The training loss curves show a rapid decrease in the first few epochs, reaching very low values, and stabilizes around a small positive value. This behavior is typical of a well-functioning model. Despite some small oscillations in the loss curve after the initial drop, the general trend is towards convergence, which is a positive sign that the model is learning effectively. The training loss curves increase rapidly, stabilizing around high values such as 98-99%.
On the other hand, during validation we observe a different trend. The validation losses exhibit

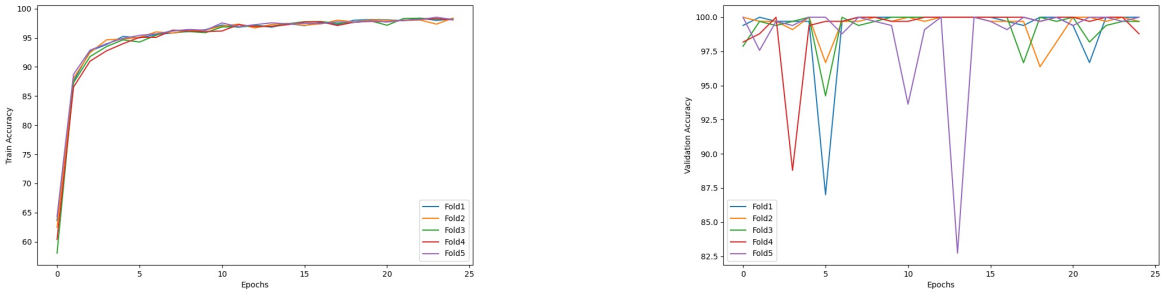


Figure 5.85: Accuracies during training (left) and validation (right) for each fold for long volume check.

significant fluctuations throughout the epochs, with occasional sharp spikes. The presence of these peaks suggests that the model is struggling with certain data points, potentially due to overfitting or higher variability in the data for specific folds. Likewise, the validation accuracy shows substantial oscillations. The accuracies initially rise but experiences several dips, dropping to as low as 82.5% for some folds before rising again and stabilizing around 94-96%. These fluctuations are typical in cross-validation tasks and suggest that the model is sensitive to the particularities of each fold of the data. However, they should be carefully monitored to ensure that the model maintains a good generalization capability when applied to real-world data.
Table 5.16 presents the accuracies achieved during validation for each fold. The last column represents the average accuracy computed by averaging across all folds.

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean Accuracy
100%	99.70%	99.70%	98.78%	100.0%	99.64% $\pm$ 0.67%

Table 5.16: Average accuracies for each fold and overall mean accuracy.

By selecting the best model, Figure 5.86 shows the confusion matrix obtained on the validation set.

The model achieved 100% of accuracy and macro F1-score, demonstrating that is perfeclty suited for long volume check. Due to computational constraints, we opted not to present the ROC curves for the multi-class classification task. Instead, we focused on alternative performance metrics, specifically accuracy and confusion matrix, which provide a clear and efficient evaluation of model performance in this context.

Explainability

To better understand the mechanism behind our vision transformer, we performed analysed its attention map for training dateset. Figures 5.87, 5.88, 5.89, 5.90, 5.91, 5.92, 5.93, 5.94, 5.95, 5.96, and 5.97 illustrate the original image (left) and the attention map (right) for the following cases: 0 μL, 200 μL transparent liquid, 200 μL red liquid, 400 μL transparent liquid, 400 μL red liquid, 600 μL transparent liquid, 600 μL red liquid, 800 μL transparent liquid, 800 μL red liquid, 1000 μL transparent liquid and, 1000 μL red liquid, respectively.
By analysing the attention maps, we observe distinct behaviours depending on the volume and type of liquid present in the tip. For volumes ranging from 0 to 400 μL, the model predominantly focuses on regions in the background rather than directly on the tip or the liquid level. Interestingly, these

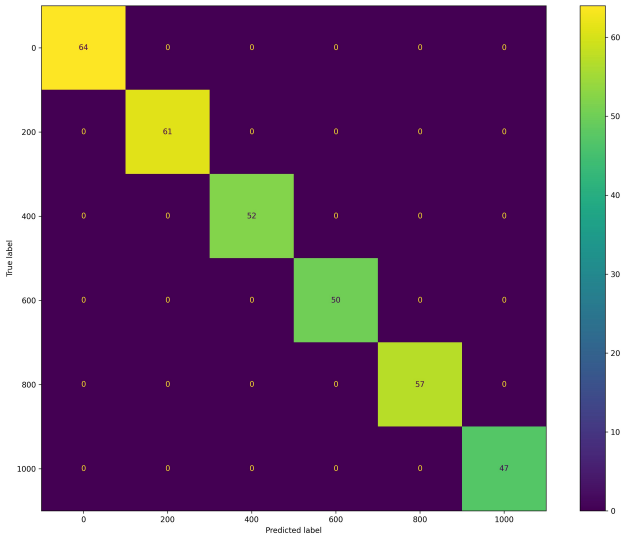


Figure 5.86: Confusion matrix of the best ViT model on the validation set.

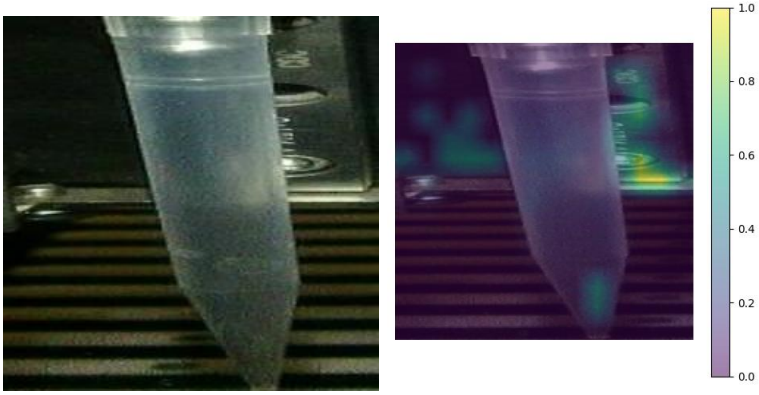


Figure 5.87: Original image (left) and attention map (right) for the case with 0 μL

background regions do not correspond to areas of high contrast or strong reflections, suggesting that the model may be influenced by features that are not directly related to the main object of interest. Starting from 600 μL and higher, the model's attention shifts more clearly toward the liquid level inside the tip. This behaviour is particularly evident for the red liquid, where the attention intensity, indicated by the yellow regions, is stronger compared to the transparent liquid. This suggests that the model finds the red liquid easier to detect, likely due to the higher visual contrast between the red liquid and the background.
Overall, these findings indicate that the ViT model, while capable of identifying the liquid level for higher volumes, may still be partially distracted by irrelevant background features, especially at lower volumes.

Evaluation on an external test set

We performed the model's capability to classify images that were not used in training and validation steps. The test set used was the same used for the CNN approach and introduced in Section 5.7.1.

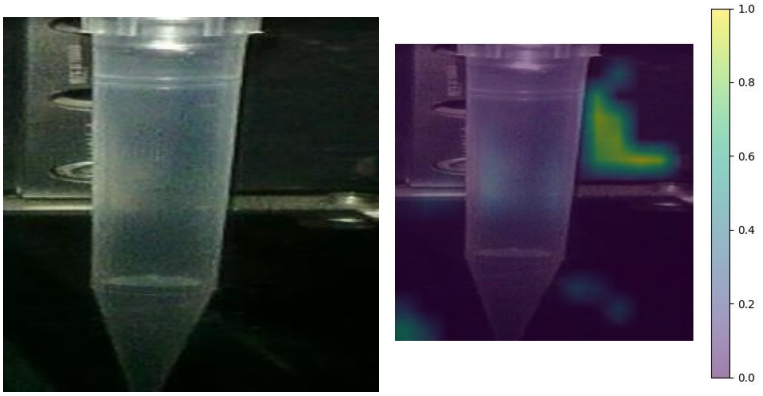


Figure 5.88: Original image (left) and attention map (right) for the case with 200 μL transparent liquid.

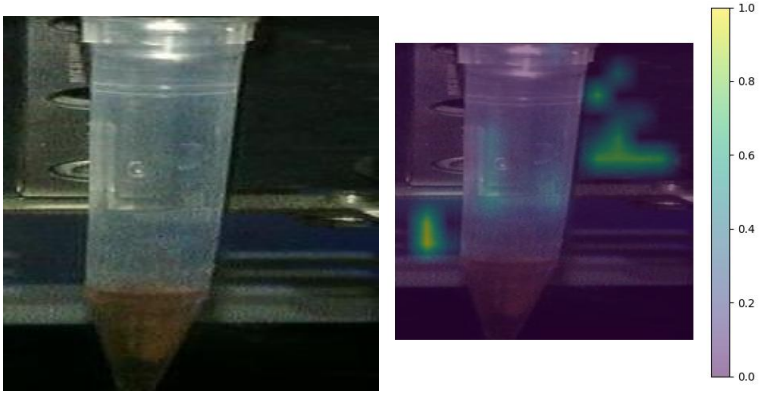


Figure 5.89: Original image (left) and attention map (right) for the case with 200 μL red liquid.

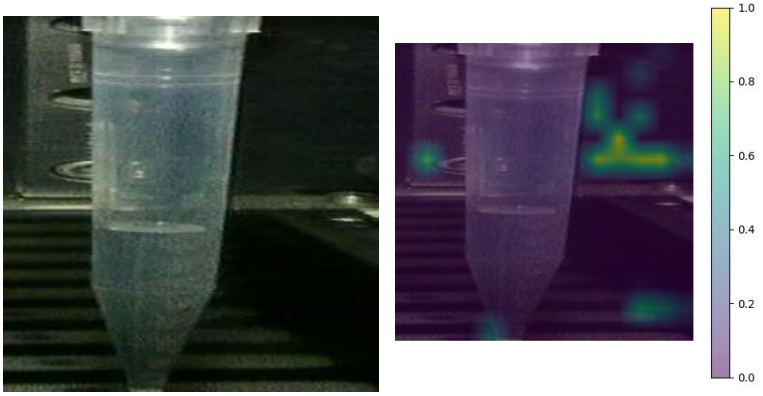


Figure 5.90: Original image (left) and attention map (right) for the case with 400 μL transparent liquid.

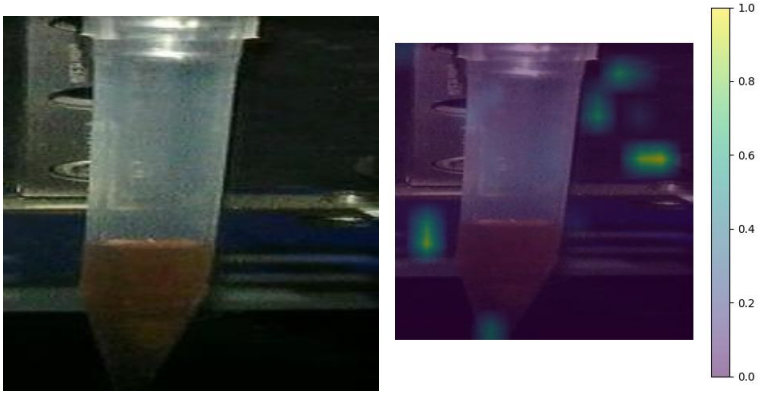


Figure 5.91: Original image (left) and attention map (right) for the case with 400 μL red liquid.

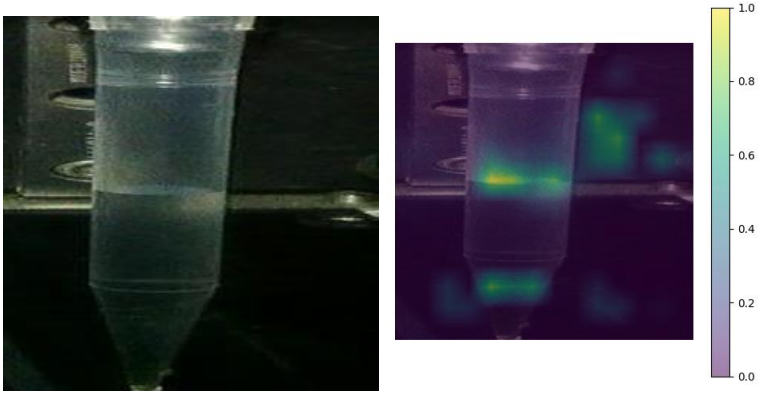


Figure 5.92: Original image (left) and attention map (right) for the case with 600 μL transparent liquid.

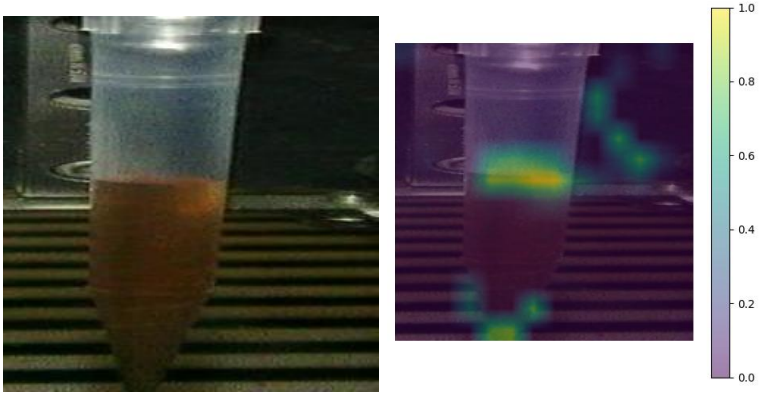


Figure 5.93: Original image (left) and attention map (right) for the case with 600 μL red liquid.

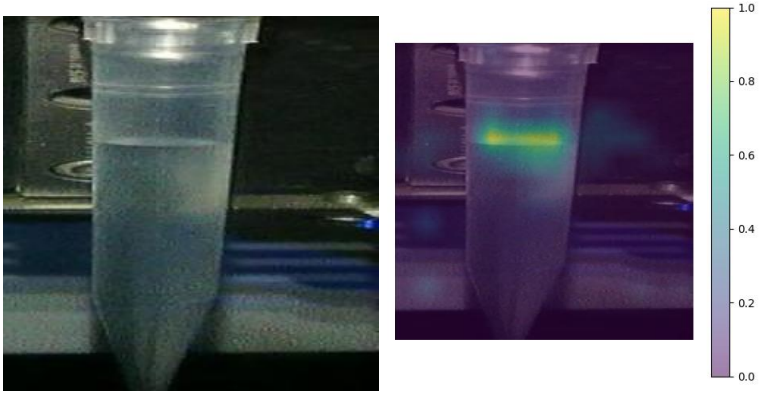


Figure 5.94: Original image (left) and attention map (right) for the case with 800 μL transparent liquid.

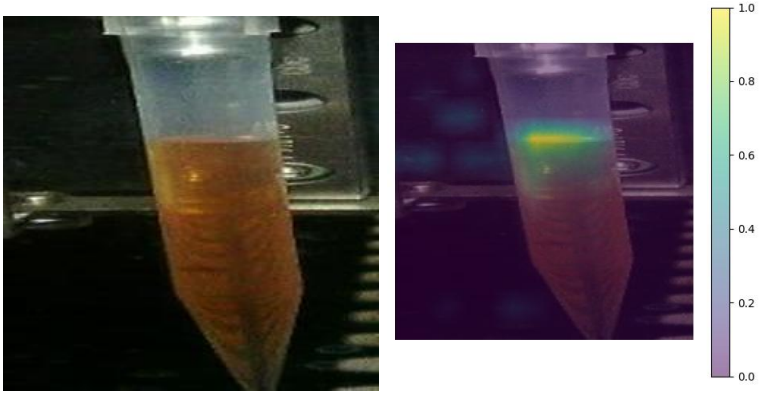


Figure 5.95: Original image (left) and attention map (right) for the case with 800 μL red liquid.

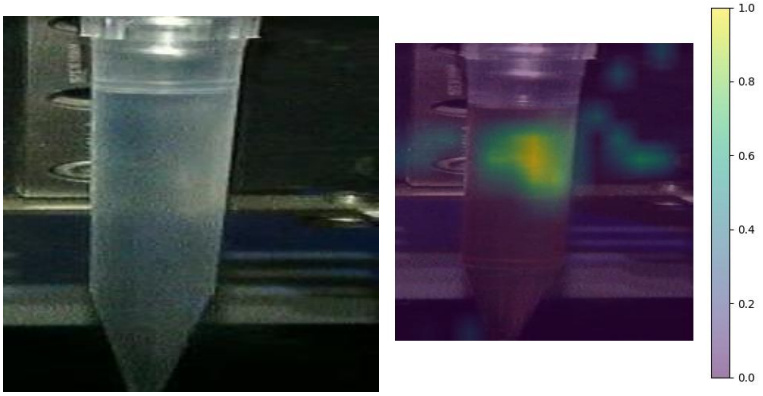


Figure 5.96: Original image (left) and attention map (right) for the case with 1000 μL transparent liquid.

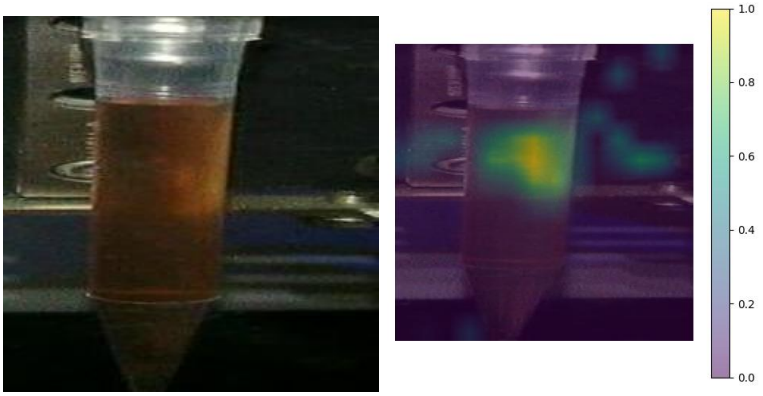


Figure 5.97: Original image (left) and attention map (right) for the case with 1000 μL red liquid.

Figure 5.98 presents the confusion matrix obtained on the test set. Our model achieved an accuracy and a macro F1-score of 98.86%, confirming its robustness under external test conditions. Moreover,

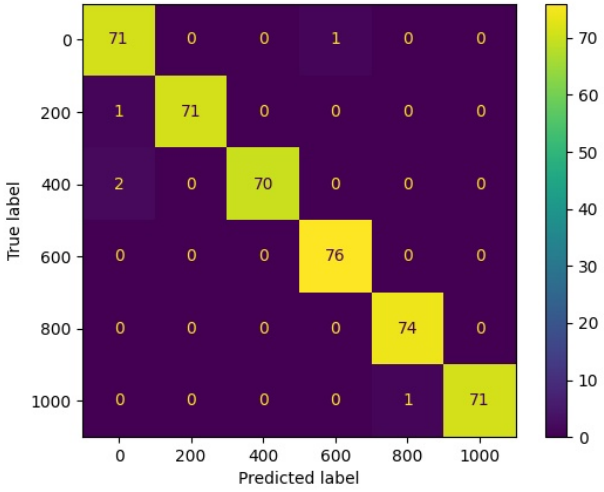


Figure 5.98: Confusion matrix of the ViT model on the test set.

we evaluated the alignment between the previous results and that one obtained on the test set in terms of explainability. Figures 5.99, 5.100, 5.101, 5.102, 5.103, 5.104, 5.105, 5.106, 5.107, 5.108, and 5.109 illustrate the original image (left) and the attention map (right) for the following cases: 0 μL, 200 μL transparent liquid, 200 μL red liquid, 400 μL transparent liquid, 400 μL red liquid, 600 μL transparent liquid, 600 μL red liquid, 800 μL transparent liquid, 800 μL red liquid, 1000 μL transparent liquid and, 1000 μL red liquid, respectively.

Upon analysing the attention maps for the test set, we observe a similar trend to the one seen during training. In all cases, the model primarily focuses on the base of the tip, particularly up to 400 μL, where the attention appears more diffuse, possibly due to lower contrast or other background factors. As the liquid volume increases, from 600 μL onward, the model's attention becomes more focused on the liquid itself, especially for the red liquid. This trend supports the idea that the model is able to concentrate on the relevant areas as the volume increases, providing more stable predictions. This observation aligns well with the model's behaviour in training, further confirming its ability to focus on the relevant features as the liquid volume increases. The consistency across

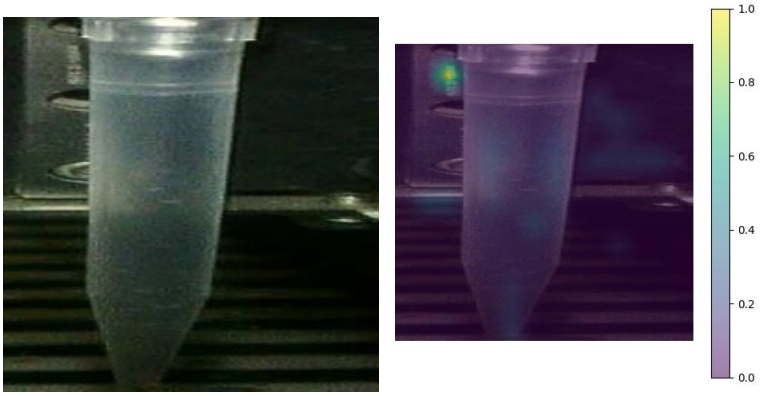


Figure 5.99: Original image (left) and attention map (right) for the case with 0 μL

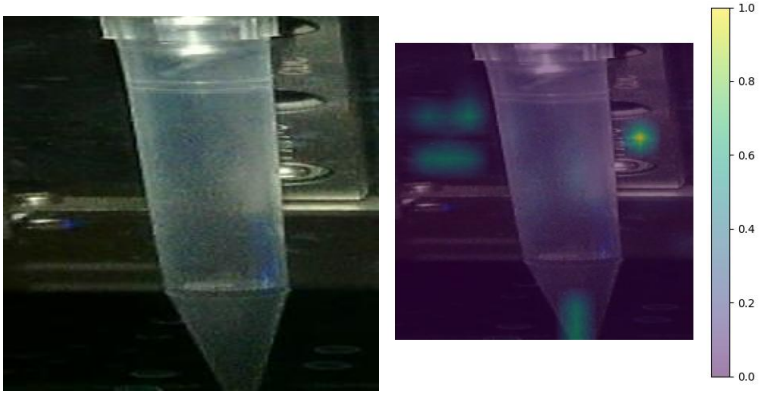


Figure 5.100: Original image (left) and attention map (right) for the case with 200 μL transparent liquid.

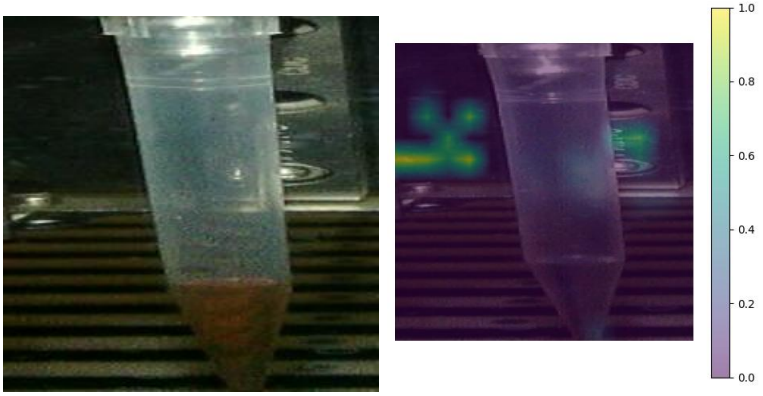


Figure 5.101: Original image (left) and attention map (right) for the case with 200 μL red liquid.

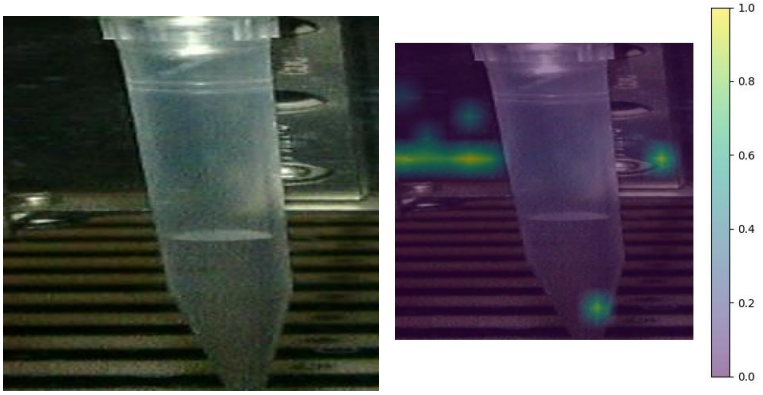


Figure 5.102: Original image (left) and attention map (right) for the case with 400 μL transparent liquid.

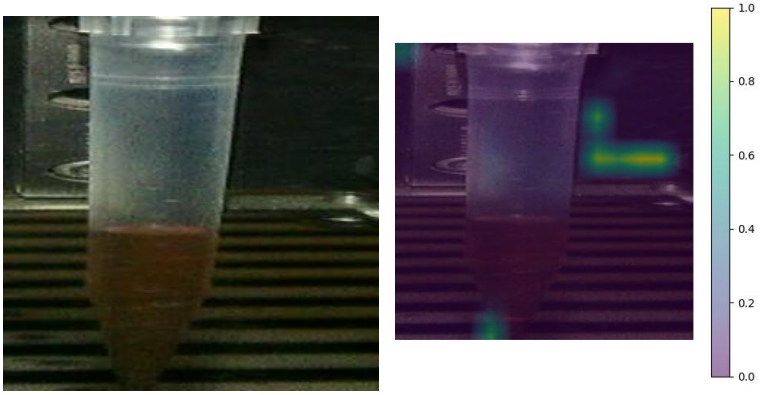


Figure 5.103: Original image (left) and attention map (right) for the case with 400 μL red liquid.

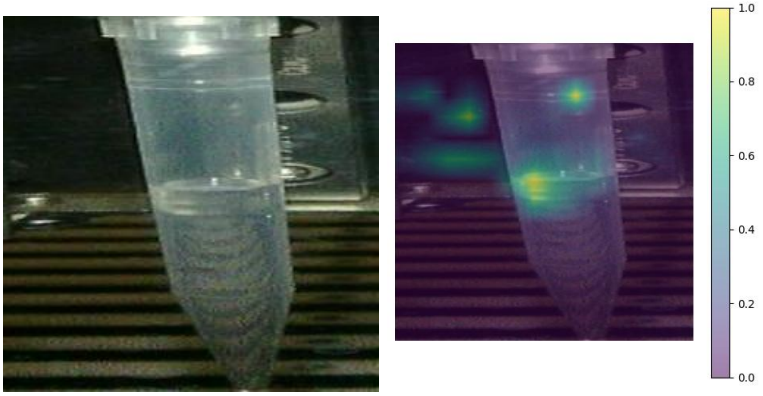


Figure 5.104: Original image (left) and attention map (right) for the case with 600 μL transparent liquid.

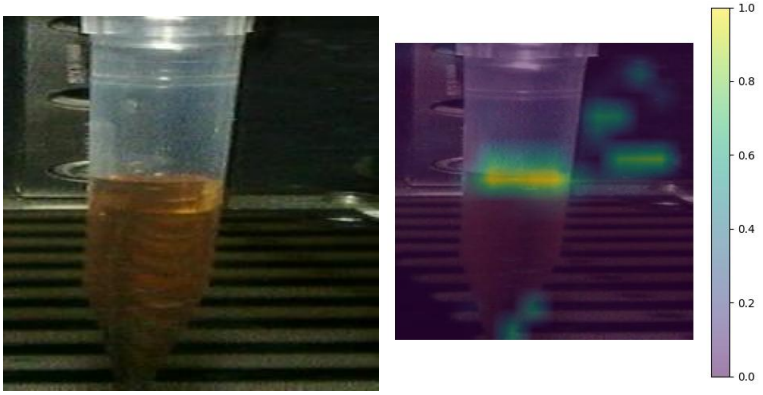


Figure 5.105: Original image (left) and attention map (right) for the case with 600 μL red liquid.

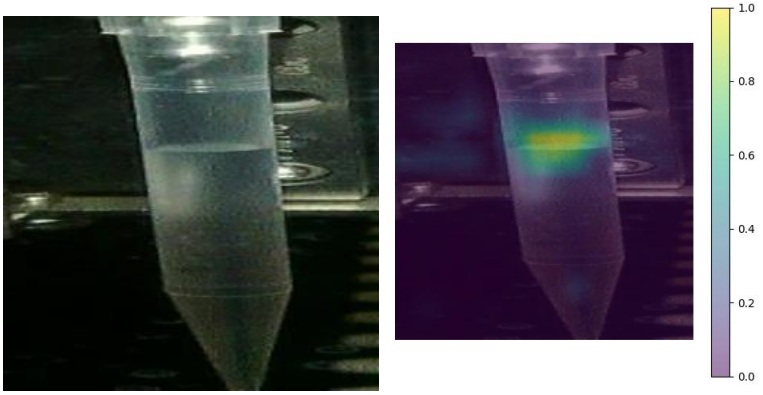


Figure 5.106: Original image (left) and attention map (right) for the case with 800 μL transparent liquid.

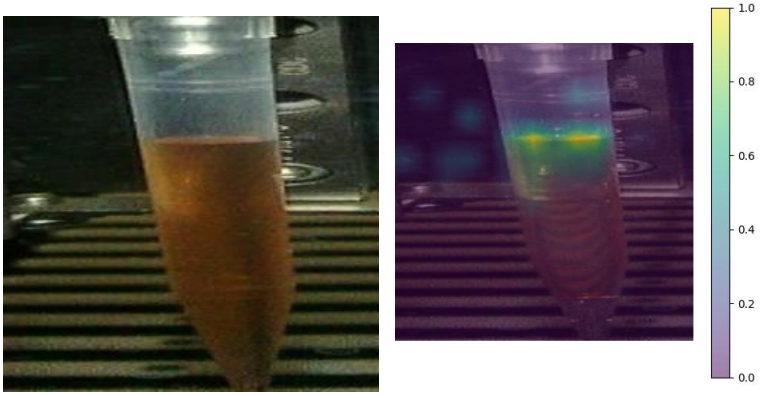


Figure 5.107: Original image (left) and attention map (right) for the case with 800 μL red liquid.

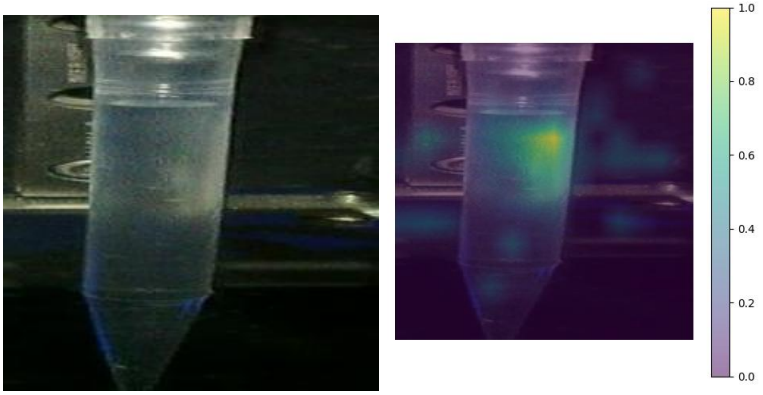


Figure 5.108: Original image (left) and attention map (right) for the case with 1000 μL transparent liquid.

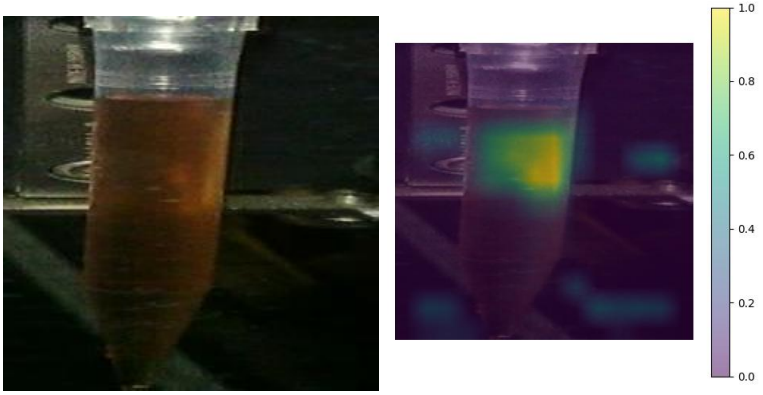


Figure 5.109: Original image (left) and attention map (right) for the case with 1000 μL red liquid.

both training and test sets suggests that the model is generalizing well to new data.

Time performance

Finally, we evaluated the time required by our vision transformer to perform both the pre-processing and classification steps, in order to verify their compatibility with the time constraints imposed by the application context. Table 5.17 reports the average pre-processing, classification, and total execution times, together with their corresponding standard deviations. These measurements were obtained on the 438 images of the test set and executed on hardware systems identical to those integrated within the medical devices available in MACS laboratories.

Function	Time
Pre-processing	$1.188 \cdot 10^{-2} \pm 8.744 \cdot 10^{-3}$ s
Prediction	$3.706 \cdot 10^{-1} \pm 6.556 \cdot 10^{-2}$ s
Total	$3.825 \cdot 10^{-1} \pm 6.541 \cdot 10^{-2}$ s

Table 5.17: Average extraction, prediction, and total times with standard deviations for the selected model.

Since the total time required by the model to process and classify a single image is approximately $3.825 \cdot 10^{-1}$ seconds, and the maximum allowed execution time for this check is 1 second, we can conclude that the model complies fully with the timing requirements for deployment within the target system.

Models comparison

In order to identify the most suitable model for integration into the machine vision system for the long volume check, we compared the performance of the two models, CNN and ViT. The comparison was carried out based on several criteria: accuracy achieved during both validation and testing phases, explainability of predictions, and execution time.
When analysing the accuracy, we observe that both models achieved a maximum accuracy of 100% during the validation phase. However, in the testing phase, the ViT model outperformed the CNN, achieving an accuracy of 98.86% compared to the CNN's 95.89%. Importantly, the macro F1-score values confirmed these results, showing a consistent alignment with the accuracies reported in both validation and testing.
Despite these differences in testing performance, further analysis of the feature maps and attention maps reveals significant disparities in the models' behaviour. The CNN consistently attributes relevance to the portions of the image corresponding to the liquid, whether transparent or red, across all cases examined. On the other hand, the ViT model appears to focus on regions considered less relevant for volumes up to 400 μL, only starting to concentrate on the liquid level from 600 μL onward, regardless of its color. This difference in focus indicates that the CNN is more consistent in identifying the relevant features across all volumes.
Lastly, when considering the processing and prediction times, the CNN demonstrates a significant advantage, requiring less time than the ViT model (9.964×10^{-2} seconds versus 3.825×10^{-1} seconds).
Despite the superior accuracy of the ViT model in testing, we opted for the CNN due to its stronger alignment between predictions and feature maps, which contributes to a higher level of explainability. This consistency between the model's attention and the expected behavior is a crucial factor for the robustness of the machine vision system in real-world applications.

Chapter 6

Conclusions

This thesis investigated the application of Artificial Intelligence for the automated monitoring of the functioning of an In Vitro Diagnostic device. The project originated from the need to limit operational errors that may compromise patient safety and damage the instrument. The monitoring system developed focuses on the key functional element of the device: the pipette tip, which is responsible for fluid handling.
The current version of the device is equipped with internal sensors intended to detect operational anomalies; however, two major limitations were identified: incomplete coverage of all functional steps and the lack of direct measurement capabilities. To address these critical issues, a novel monitoring system based on machine vision and AI algorithms was designed and fully developed by MACS s.r.l., the company holding the patent for the device.
The monitoring system specifically targets two core functions of the device: tip gripping and liquid handling. Three sequential checks were implemented: verification of correct tip gripping (presence check), identification of the correct tip size (type check), and estimation of the liquid volume inside the tip (volume check). Following a preliminary feasibility assessment, data collection was carried out to support the development of the AI algorithms, leveraging libraries such as OpenCV, TensorFlow/Keras, and Pytorch. In addition, several pre-processing and data augmentation techniques were applied to enhance model robustness and performance. The implemented models span from traditional image processing and machine learning to deep learning, including Support Vector Machines, Convolutional Neural Networks, and Vision Transformers. For the presence check, multiple SVM models with different kernel functions were trained for binary classification. Custom-designed feature sets were extracted and specifically tailored for this task. All trained models demonstrated excellent performance, achieving perfect accuracy, precision, recall, and F1-score. The selected model, an SVM with RBF kernel, was chosen not only for its perfect validation and external testing performance, but also because it required fewer support vectors compared to alternatives, indicative of a compact decision function. The model was integrated into the machine vision system in C++ for real-time inference, confirming both robustness and computational efficiency.
For the type check task, several SVM models with different kernel functions were implemented using Fourier descriptors as input features. In parallel, a CNN was developed to perform image classification. The results highlighted a clear difference: while the support vector machines relying on handcrafted descriptors, performed close to chance both in validation and testing phases, the CNN demonstrated significantly superior performance, achieving perfect classification of 200 μL and

1000 μL tips in both the validation and testing phases. Explainability analyses using Integrated Gradients and Grad-CAM confirmed that the CNN accurately localized the relevant features corresponding to the tip. Moreover, in some cases, Grad-CAM showed that the model also attended to background reflections, which could serve as misleading cues.

For the volume check task, separate multi-class classification models were developed for 200 μL and 1000 μL tips. For both cases, CNNs and ViTs were trained and compared. The CNNs consistently demonstrated superior performance in validation and provided interpretable Grad-CAM heatmaps, which accurately highlighted the liquid regions (red liquid) or the liquid level (transparent liquid). By contrast, the ViTs achieved higher accuracy on external test sets, but their attention maps often focused on irrelevant background regions, particularly for lower volumes, and only at higher volumes did they consistently attend to the liquid. IGs for CNNs provided fine-grained but weak attributions, while Grad-CAM heatmaps proved more informative, reinforcing the importance of complementing different explainability techniques. Taken together, these analyses confirmed that CNNs aligned more closely with expected features, while ViTs, although accurate, raised concerns about the consistency and reliability of their internal reasoning.

Finally, a computational performance evaluation was carried out for each selected model. All chosen models met the predefined timing constraints for pre-processing and inference, demonstrating suitability for integration into the real-time workflow of the device. Importantly, all deep learning models were trained offline in Python and then deployed in the C++-based machine vision system for inference only, ensuring computational efficiency during operation without interfering with the device's workflow.

This thesis demonstrated that integrating artificial intelligence into vision-based monitoring systems can significantly enhance the safety, reliability, and automation of laboratory diagnostic devices. The experimental results confirmed the robustness and efficiency of the implemented methods in addressing the specific monitoring tasks. These outcomes show the practical feasibility of integrating such solutions into real-world diagnostic environments. Explainability analyses further revealed that the models occasionally rely on spurious cues, such as reflections or background elements, particularly in the case of transparent liquids. These behaviors highlight the need for additional refinement, possibly through pre-processing strategies (e.g., reflection suppression, masking) or targeted data augmentation. Nevertheless, although accuracy exceeded 95% in most cases, misclassifications still translate into repeated operations within the workflow, potentially slowing down execution and increasing consumable use.

Despite these promising results, several limitations were identified. The proposed models were developed and validated on a specific IVD device and dataset, which may restrict their generalizability to other platforms. The monitoring system currently covers only selected functional phases, leaving other critical steps unaddressed. Future work will aim to address these limitations by extending the monitoring system to additional checks, such as the detection of air bubbles or foam during liquid handling, by improving pre-processing and acquisition protocols to minimize background reflections and irrelevant cues, by enhancing model robustness on transparent liquids where attention occasionally shifts away from the liquid level, and by scaling and validating the proposed solutions on different diagnostic devices to assess generalizability and industrial applicability.

Bibliography

[1] Giulia Tufo, Meriam Zribi, and Francesca Pitolli. A novel approach to monitoring ivd devices via machine learning and computer vision. *Computers & Industrial Engineering*, page 111443, 2025.

[2] Hossein Golnabi and A Asadpour. Design and application of industrial machine vision systems. *Robotics and Computer-Integrated Manufacturing*, 23(6):630–637, 2007.

[3] Heidi Lindroth, Keivan Nalaie, Roshini Raghu, Ivan N Ayala, Charles Busch, Anirban Bhattacharyya, Pablo Moreno Franco, Daniel A Diedrich, Brian W Pickering, and Vitaly Herasevich. Applied artificial intelligence in healthcare: a review of computer vision technology application in hospital settings. *Journal of Imaging*, 10(4):81, 2024.

[4] Sumaira Ghazal, Arslan Munir, and Waqar S Qureshi. Computer vision in smart agriculture and precision farming: Techniques and applications. *Artificial Intelligence in Agriculture*, 2024.

[5] Zhonghe Ren, Fengzhou Fang, Ning Yan, and You Wu. State of the art in defect detection based on machine vision. *International Journal of Precision Engineering and Manufacturing-Green Technology*, 9(2):661–691, 2022.

[6] Bin Li. Application of machine vision technology in geometric dimension measurement of small parts. *EURASIP Journal on Image and Video Processing*, 2018(1):127, 2018.

[7] Paola Pierleoni, Alberto Belli, Lorenzo Palma, Michela Palmucci, and Luisiana Sabbatini. A machine vision system for manual assembly line monitoring. In *2020 International Conference on Intelligent Engineering and Management (ICIEM)*, pages 33–38. IEEE, 2020.

[8] Abdelrahman Alzarooni, Ehtesham Iqbal, Samee Ullah Khan, Sajid Javed, Brain Moyo, and Yusra Abdulrahman. Anomaly detection for industrial applications, its challenges, solutions, and future directions: A review. *arXiv preprint arXiv:2501.11310*, 2025.

[9] Chengsi Lin, Muhamad Arfauz A Rahman, and Paul G Maropoulos. Enhancing the quality inspection process in the food manufacturing industry through automation. In *2022 8th International Conference on Control, Decision and Information Technologies (CoDIT)*, volume 1, pages 379–384. IEEE, 2022.

[10] QM Yuan and H Zhang. Research on the characteristics of light sources in machine vision. *Acad J Sci Technol*, 3(1):1–6, 2022.

[11] Lili Zhu, Petros Spachos, Erica Pensini, and Konstantinos N Plataniotis. Deep learning and machine vision for food processing: A survey. *Current Research in Food Science*, 4:233–249, 2021.

[12] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Nature, 2022.

[13] Yuchen Jiang, Xiang Li, Hao Luo, Shen Yin, and Okyay Kaynak. Quo vadis artificial intelligence? *Discover Artificial Intelligence*, 2(1):4, 2022.

[14] Earl B Hunt. *Artificial intelligence*. Academic Press, 2014.

[15] Issam El Naqa and Martin J Murphy. *What is machine learning?* Springer, 2015.

[16] Christian Janiesch, Patrick Zschech, and Kai Heinrich. Machine learning and deep learning. *Electronic Markets*, 31(3):685–695, 2021.

[17] Zhi-Hua Zhou. *Machine learning*. Springer nature, 2021.

[18] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[19] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[20] Sumaira Manzoor, Sung-Hyeon Joo, and Tae-Yong Kuc. Comparison of object recognition approaches using traditional machine vision and modern deep learning techniques for mobile robot. In *2019 19th International Conference on Control, Automation and Systems (ICCAS)*, pages 1316–1321. IEEE, 2019.

[21] Niall O'Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. Deep learning vs. traditional computer vision. In *Advances in computer vision: proceedings of the 2019 computer vision conference (CVC), volume 1 1*, pages 128–144. Springer, 2020.

[22] Lei Wang, Wenchang Xu, Biao Wang, Xiaonan Si, and Shengyu Li. Methods and advances in the design, testing and development of in vitro diagnostic instruments. *Processes*, 11(2):403, 2023.

[23] Mark Graber. Diagnostic errors in medicine: a case of neglect. *The joint commission journal on quality and patient safety*, 31(2):106–113, 2005.

[24] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20:273–297, 1995.

[25] Jair Cervantes, Farid Garcia-Lamont, Lisbeth Rodríguez-Mazahua, and Asdrubal Lopez. A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, 408:189–215, 2020.

[26] Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan.

Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, 8:1–74, 2021.

[27] Dulari Bhatt, Chirag Patel, Hardik Talsania, Jigar Patel, Rasmika Vaghela, Sharnil Pandya, Kirit Modi, and Hemant Ghayvat. Cnn variants for computer vision: History, architecture, application, challenges and future scope. *Electronics*, 10(20):2470, 2021.

[28] Asifullah Khan, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi. A survey of the recent architectures of deep convolutional neural networks. *Artificial intelligence review*, 53: 5455–5516, 2020.

[29] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[30] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM computing surveys (CSUR)*, 54 (10s):1–41, 2022.

[31] Keyur D Joshi, Vedang Chauhan, and Brian Surgenor. A flexible machine vision system for small part inspection based on a hybrid svm/ann approach. *Journal of Intelligent Manufacturing*, 31(1):103–125, 2020.

[32] Maher GM Abdolrasol, SM Suhail Hussain, Taha Selim Ustun, Mahidur R Sarker, Mahammad A Hannan, Ramizi Mohamed, Jamal Abd Ali, Saad Mekhilef, and Abdalrhman Milad. Artificial neural networks based optimization techniques: A review. *Electronics*, 10(21):2689, 2021.

[33] David Ireri, Eisa Belal, Cedric Okinda, Nelson Makange, and Changying Ji. A computer vision system for defect discrimination and grading in tomatoes using machine learning and image processing. *Artificial Intelligence in Agriculture*, 2:28–37, 2019.

[34] Gérard Biau and Erwan Scornet. A random forest guided tour. *Test*, 25(2):197–227, 2016.

[35] Adel Bakhshipour and Abdolabbas Jafari. Evaluation of support vector machine and artificial neural networks in weed detection using shape features. *Computers and Electronics in Agriculture*, 145:153–160, 2018.

[36] P Lin, XL Li, YM Chen, and Y He. A deep convolutional neural network architecture for boosting image discrimination accuracy of rice species. *Food and Bioprocess Technology*, 11: 765–773, 2018.

[37] Kunal J Pithadiya, Chintan K Modi, and Jayesh D Chauhan. Comparison of optimal edge detection algorithms for liquid level inspection in bottles. In *2009 Second international conference on emerging trends in engineering & technology*, pages 447–452. IEEE, 2009.

[38] GT Shrivakshan and Chandramouli Chandrasekar. A comparison of various edge detection techniques used in image processing. *International Journal of Computer Science Issues (IJCSI)*, 9(5):269, 2012.

[39] Mikhael Anthony A Felipe, Tanya V Olegario, Nilo T Bugtai, and Renann G Baldovino. Vision-based liquid level detection in amber glass bottles using opencv. In *2019 7th International Conference on Robot Intelligence Technology and Applications (RiTA)*, pages 148–152. IEEE, 2019.

[40] OpenCV. Open source computer vision library, 2015.

[41] Miriam Cobo, Ignacio Heredia, Fernando Aguilar, Lara Lloret Iglesias, Daniel García, Begoña Bartolomé, M Victoria Moreno-Arribas, Silvia Yuste, Patricia Pérez-Matute, and Maria-Jose Motilva. Artificial intelligence to estimate wine volume from single-view images. *Heliyon*, 8(9), 2022.

[42] Maria MP Petrou and Costas Petrou. *Image processing: the fundamentals*. John Wiley & Sons, 2010.

[43] Jiss Kuruvilla, Dhanya Sukumaran, Anjali Sankar, and Siji P Joy. A review on image processing and image segmentation. In *2016 international conference on data mining and advanced computing (SAPIENCE)*, pages 198–203. IEEE, 2016.

[44] Mark Nixon and Alberto Aguado. *Feature extraction and image processing for computer vision*. Academic press, 2019.

[45] Allah Bux Sargano, Plamen Angelov, and Zulfiqar Habib. A comprehensive review on handcrafted and learning-based action representation approaches for human activity recognition. *applied sciences*, 7(1):110, 2017.

[46] Jiayi Ma, Xingyu Jiang, Aoxiang Fan, Junjun Jiang, and Junchi Yan. Image matching from handcrafted to deep features: A survey. *International Journal of Computer Vision*, 129(1): 23–79, 2021.

[47] Mehmet Sezgin and Bu¨ lent Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic imaging*, 13(1):146–168, 2004.

[48] Ta Yang Goh, Shafriza Nisha Basah, Haniza Yazid, Muhammad Juhairi Aziz Safar, and Fathinul Syahir Ahmad Saad. Performance analysis of image thresholding: Otsu technique. *Measurement*, 114:298–307, 2018.

[49] Jean Serra. *Image analysis and mathematical morphology*. Academic Press, Inc., 1983.

[50] Martha Rebeca Canales-Fiscal and José Gerardo Tamez-Peña. Hybrid morphological-convolutional neural networks for computer-aided diagnosis. *Frontiers in Artificial Intelligence*, 6:1253183, 2023.

[51] Guang Deng and LW Cahill. An adaptive gaussian filter for noise reduction and edge detection. In *1993 IEEE conference record nuclear science symposium and medical imaging conference*, pages 1615–1619. IEEE, 1993.

[52] Mehdi Mafi, Harold Martin, Mercedes Cabrerizo, Jean Andrian, Armando Barreto, and Malek Adjouadi. A comprehensive survey on impulse and gaussian denoising filters for digital images. *Signal Processing*, 157:236–260, 2019.

[53] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[54] Lijun Ding and Ardeshir Goshtasby. On the canny edge detector. *Pattern recognition*, 34(3): 721–725, 2001.

[55] Paul Bao, Lei Zhang, and Xiaolin Wu. Canny edge detection enhancement by scale multiplication. *IEEE transactions on pattern analysis and machine intelligence*, 27(9):1485–1490, 2005.

[56] Eric Persoon and King-Sun Fu. Shape discrimination using fourier descriptors. *IEEE Transactions on systems, man, and cybernetics*, 7(3):170–179, 1977.

[57] Dengsheng Zhang and Guojun Lu. Shape-based image retrieval using generic fourier descriptor. *Signal Processing: Image Communication*, 17(10):825–848, 2002.

[58] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[59] Arti Patle and Deepak Singh Chouhan. Svm kernel functions for classification. In *2013 International conference on advances in technology and engineering (ICATE)*, pages 1–9. IEEE, 2013.

[60] Subhransu Maji, Alexander C. Berg, and Jitendra Malik. Classification using intersection kernel support vector machines is efficient. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008. doi: 10.1109/CVPR.2008.4587630.

[61] Lai Jiang and Runming Yao. Modelling personal thermal sensations using c-support vector classification (c-svc) algorithm. *Building and Environment*, 99:98–106, 2016.

[62] Sanjay Yadav and Sanyam Shukla. Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification. In *2016 IEEE 6th International Conference on Advanced Computing (IACC)*, pages 78–83, 2016. doi: 10.1109/IACC.2016.25.

[63] Davide Anguita, Luca Ghelardoni, Alessandro Ghio, Luca Oneto, Sandro Ridella, et al. The'k'in k-fold cross validation. In *ESANN*, pages 441–446, 2012.

[64] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[65] Arshi Parvaiz, Muhammad Anwaar Khalid, Rukhsana Zafar, Huma Ameer, Muhammad Ali, and Muhammad Moazam Fraz. Vision transformers in medical computer vision—a contemplative retrospection. *Engineering Applications of Artificial Intelligence*, 122:106126, 2023.

[66] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *Advances in neural information processing systems*, 34:12116–12128, 2021.

[67] José Maurício, Inês Domingues, and Jorge Bernardino. Comparing vision transformers and convolutional neural networks for image classification: A literature review. *Applied Sciences*, 13(9):5521, 2023.

[68] Nikhil Ketkar. Stochastic gradient descent. In *Deep learning with Python: A hands-on introduction*, pages 113–132. Springer, 2017.

[69] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, 6:52138–52160, 2018.

[70] David Gunning, Mark Stefik, Jaesik Choi, Timothy Miller, Simone Stumpf, and Guang-Zhong Yang. Xai—explainable artificial intelligence. *Science robotics*, 4(37):eaay7120, 2019.

[71] Arun Das and Paul Rad. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*, 2020.

[72] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[73] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[74] Ivan Čík, Andrindrasana David Rasamoelina, Marián Mach, and Peter Sinčák. Explaining deep neural network using layer-wise relevance propagation and integrated gradients. In *2021 IEEE 19th world symposium on applied machine intelligence and informatics (SAMI)*, pages 000381–000386. IEEE, 2021.

[75] Minjae Chung, Jong Bum Won, Ganghyun Kim, Yujin Kim, and Utku Ozbulak. Evaluating visual explanations of attention maps for transformer-based medical imaging. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 110–120. Springer, 2024.

[76] Debashree Devi, Saroj K. Biswas, and Biswajit Purkayastha. A review on solution to class imbalance problem: Undersampling approaches. In *2020 International Conference on Computational Performance Evaluation (ComPE)*, pages 626–631, 2020. doi: 10.1109/ComPE49325.2020.9200087.

[77] Mohanad Sarhan, Siamak Layeghy, Nour Moustafa, Marcus Gallagher, and Marius Portmann. Feature extraction for machine learning-based intrusion detection in iot networks. *Digital Communications and Networks*, 10(1):205–216, 2024.

[78] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[79] Szilvia Szeghalmy and Attila Fazekas. A comparative study of the use of stratified cross-validation and distribution-balanced stratified cross-validation in imbalanced learning. *Sensors*, 23(4):2333, 2023.

[80] Dries Geebelen, Johan AK Suykens, and Joos Vandewalle. Reducing the number of support vectors of svm classifiers using the smoothed separable case approximation. *IEEE Transactions on Neural Networks and Learning Systems*, 23(4):682–688, 2012.

[81] Daniel Hsu, Vidya Muthukumar, and Ji Xu. On the proliferation of support vectors in high dimensions. In *International Conference on Artificial Intelligence and Statistics*, pages 91–99. PMLR, 2021.

[82] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[83] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[84] Michael A Lones. How to avoid machine learning pitfalls: a guide for academic researchers. *arXiv preprint arXiv:2108.02497*, 2021.

[85] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[86] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.

[87] Juri Opitz and Sebastian Burst. Macro f1 and macro f1, 2021. URL `https://arxiv.org/abs/1911.03347`.

[88] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[89] Rui Zhu, Yiwen Guo, and Jing-Hao Xue. Adjusting the imbalance ratio by the dimensionality of imbalanced data. *Pattern Recognition Letters*, 133:217–223, 2020.

[90] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *arXiv preprint arXiv:2106.10270*, 2021.

[91] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180, 2023.

[92] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12104–12113, 2022.

[93] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses. *arXiv preprint arXiv:2104.10972*, 2021.