



**JIMMA UNIVERSITY
JIMMA INSTITUTE OF TECHNOLOGY
FACULTY OF ELECTRICAL AND
COMPUTER ENGINEERING**

**A Two-Stage Deep Learning Approach with Image-
Tiling for Malaria Diagnosis from Microscopic
Blood**

A Thesis Submitted to Jimma Institute of Technology, School of Graduate
Studies, Jimma University in Partial Fulfillment of the Requirement for the
Degree of Master of Science in Computer Engineering

Eyosiyas Tibebu

**Date Submitted
November/2024
Jimma, Ethiopia**

DECLARATION

The researcher certifies that this thesis, “**A Two-Stage Deep Learning Approach with Image-Tiling for Malaria Diagnosis from Microscopic Blood Smears**” is wholly mine and has not been submitted in whole or in part in any previous application for a degree. Unless otherwise referenced or acknowledged, the work presented is all mine.

Advisor: Dr. Kinde Anlay

Signature: _____

Date: _____

Co-Advisor: Mr. Adane Tadesse

Signature: _____

Date: _____

This Thesis has been Approved by the Board of Examiners

External Examiner: Dr. Abebe Tesfahun

Signature: _____


Date: 11/11/2024

Internal Examiner: Mr. Kris Calpotura

Signature: _____

Date: _____

Chair Person: Mr. Chara Assefa

Signature: _____

Date: _____

ACKNOWLEDGMENTS

In this respect, I perpetually thank the Almighty God for the strength and wisdom granted to him while conducting this research. I also thank the family, as true fans, who always cheered me up whenever I lost energy and hope during this challenging yet exciting process.

I also would like to thank Mr. Boaz Berhanu for providing me access to High-Performance Computing (HPC) and for the valuable advice he has given me during debugging processes to improve the technical part of my work. Thanks to his efforts, this study's computational models have improved.

I would also like to extend my deepest thanks to my advisors, Dr. Kinde Anlay and Mr. Adane Tadesse, for their advice, bearing with me, and professional suggestions regarding the direction and nature of this research. Their guidance was instrumental, especially in helping me better understand my studies and manage my research.

Also, I would like to thank everyone who contributed this or that way to my research. Their confidence was an incentive to work as hard as possible. This research would not have been successfully implemented without everyone's efforts.

Many thanks to all of you for your efforts and support in completing this work.

ABSTRACT

Malaria diagnosis is a severe difficulty in preserving human health, particularly in developing nations. It is essential if correct treatment and control measures are to be instituted and implemented. Thus, this research focuses on A Two-Stage Deep Learning Approach with Image-Tiling for Malaria Diagnosis from Microscopic Blood Smears. Due to the necessary image downscaling, traditional deep-learning models for malaria screening in high-resolution microscopic images often lose crucial spatial details. This research introduces an image-tiling technique with overlaps to preserve spatial resolution and enhance detection accuracy. Comparing YOLOv9 and RT-DETR, the study aims to identify which models are the most suitable for accurate malaria detection in blood smears, focusing on eliminating false positives and enhancing the model's reliability. This method represents a significant development since it preserves the integrity of high-resolution data throughout the detection procedure. The researchers propose a thorough two-stage detection methodology employing sophisticated tiling and refined object identification algorithms to improve malaria diagnosis. Images are divided into 640x640 tiles, and overlapping between tiles is 30% so as not to lose spatial information. The first stage utilizes the YOLOv9 algorithm to identify malaria parasites with an accuracy of 0.896. The MobileViT model subsequently enhances these detections by minimizing false positives, attaining a validation accuracy of 0.9833. After establishing malaria's existence, the procedure advances to the second stage, where thin blood smears are examined to identify kinds of malaria parasites using identical models, achieving detection accuracies of 0.931 and classification accuracies of 0.929188. This two-part approach significantly improves diagnostic accuracy and decreases the time needed to arrive at a diagnosis. The culmination of this method is integrated into a fully autonomous desktop application designed to seamlessly mesh with existing healthcare systems, dramatically reducing diagnostic response times. This novel application enhances patient quality by providing quick and accurate malaria diagnosis. It enhances the possibilities of deploying deep learning models in medical imaging, which promises to transform patient outcomes in several medical fields.

Keywords: Malaria Detection, Deep learning, Blood Smear Analysis, YOLOv9, MobileViT

TABLE OF CONTENTS

DECLARATION.....	i
ACKNOWLEDGMENTS.....	ii
ABSTRACT.....	iii
TABLE OF CONTENTS	iv
LIST OF FIGURES	vii
LIST OF TABLES	x
ACRONYMS	xi
CHAPTER ONE	1
1. INTRODUCTION.....	1
1.1 Background of the Study	1
1.2 Statement of the Problem.....	2
1.3 Motivations	3
1.4 Research Questions.....	4
1.5 Research Objectives.....	4
1.5.1 General Objectives.....	4
1.5.2 Specific Objectives	5
1.6 Significance of the Research.....	5
1.7 Scope and Limitation	5
1.7.1 Scope.....	5
1.7.2. Limitation.....	6
CHAPTER TWO	7
2. REVIEW OF RELATED LITERATURE	7
2.1 Challenges and Solutions in Malaria Diagnosis	7
2.2.1 Theoretical Background and Literature Review on Malaria Detection in Ethiopia .	7
2.2.2. Current Methods of Malaria Diagnosis.....	8
2.2.3 Systemic Challenges and Technological Opportunities	8
2.2 Developments in Advanced Neural Networks for MALARIA Detection.....	9
2.3 Accuracy and Reliability of Automated Malaria Diagnosis	12
2.4 Ethical and Practical Considerations in Implementing Artificial Intelligence in the Healthcare Sector.....	15
CHAPTER THREE.....	18
3. METHODOLOGY	18

3.1 Research Design	18
3.1.1 Research Paradigm.....	18
3.1.2 Research Flow.....	19
3.2 Sources of Data	20
3.2.1 Primary Source.....	20
3.2.2 Secondary Source.....	20
3.3 Data Collection	21
3.4 Data Annotation	22
3.5 Data Processing.....	22
3.6 Dataset Splitting.....	25
3.7 Feature Extracting.....	25
3.7.1 YOLOv9 Feature Extraction and DEtection.....	26
3.7.2 Feature Extracting and Detection for RT-DETR	36
3.7.3 Feature Extracting for Classification	42
3.8 Fine Tuning.....	52
3.9 Model Training	54
3.9.1 Hyperparameter Tuning	54
3.9.2 Training Environment	55
3.10 Model Evaluation.....	56
3.10.1 Validation and Testing	56
3.11 Ethical Considerations	1
3.12 System Architecture.....	50
3.13 Software Methodology	53
3.14 Software Requirements Specification.....	55
3.14.1 Functional Requirements of the System	55
3.14.2 Non-Functional Requirements	57
3.14.3 Software Requirements.....	58
3.14.4 Hardware Requirements.....	58
3.15 Use case Diagram	59
3.15.1 Use Case Scenario.....	60
Chapter Four	63
4. Results and Discussions	63
4.1 Experimental Setup.....	63
4.2 Hyperparameter Selection and Optimization.....	63

4.2.1 Detection Phase.....	64
4.2.2 Classification Phase	65
4.3 Model Evaluation.....	66
4.3.1 Detection of Malaria Parasite.....	66
4.3.2 Classifying the Malaria Parasite	70
4.3.3 Detection of Malaria Types	76
4.3.4 Classifying the Malaria Type.....	80
4.4 The Developed System	86
4. 4. 1 User Interface and Accessibility	86
4. 4. 2 Diagnostic Workflow	87
4.4.3 Integration and Storage	89
4.5 Testing Result	89
4.5.1 Thick Blood Smear Testing	89
4.5.2 Thin Blood Smear Testing	91
4.6 Discussion.....	92
CHAPTER FIVE	93
5. Conclusion and Recommendations	93
5.1 Conclusion	93
5.2 Recommendations.....	94
REFERENCES.....	96
Appendix.....	106

LIST OF FIGURES

Figure 3.1: Research Paradigm	18
Figure 3.2: Research Flow	19
Figure 3.3 Data Preprocessing Method.....	25
Figure 3.4: YOLOv9 Architecture [43]	27
Figure 3.5 Max.Pooling vs Average Pooling [47]	29
Figure 3.6: Residual Connections	30
Figure 3.7: Spatial Pyramid Pooling [51]	30
Figure 3.8: Feature Pyramid Network [51].....	31
Figure 3.9: Head component of YOLOv9	34
Figure 3.10: Detection Transformer Architecture	37
Figure 3.11 Mobile Vit Architecture [60].....	43
Figure 3.12: Xception Architecture [62].....	44
Figure 3.13: Inceptions Architecture [65].....	45
Figure 3.14: EfficientNetB0 Architecture [67].....	46
Figure 3.15: DenseNet121 Architecture [70].....	47
Figure 3.16: NASNet Mobile Architecture [74]	48
Figure 3.17: MobileNetV2 Architecture [77]	49
Figure 3.18: VGG16 Architecture [79].....	49
Figure 3.19: ResNet50 Architecture [81].....	50
Figure 3.20: Vision Transform Architecture [83].....	51

Figure 3.21 Left: The original VGG-16 Network Architecture. Middle: Removing the fully connected Layers. Right: Removing the Original Fully Connected Layers and Replacing Them with a Brand New Fully Connected [86]	53
Figure 3.22: Sample Understanding of Fine-tuning Using VGG-16 [86]	53
Figure 3.23: K-Fold cross-validation	57
Figure 3.24 System Architecture YOLOv9 + classification	51
Figure 3.25 Rapid Application Development Model Life Cycle	54
Figure 3.26 Usecase Diagram	59
Figure 4.1: The Training Loss, Validation Loss, mPA_50 Result of Malaria Parasite Detection	68
Figure 4.2: Confusion Matrix of YOLOv9 for Malaria Detection	68
Figure 4.3: The Accuracy and Loss of Mobile Vit	73
Figure 4.4: Confusion Matrix of Mobile ViT	73
Figure 4.5: Model Performance Comparison for MobileViT	76
Figure 4.6 The Training Loss, Validation Loss, mPA_50 Result of Malaria Type Detection .	78
Figure 4.7: Confusion Matrix of YOLOv9s for Malaria Type Detection.....	79
Figure 4.8: The Accuracy and Loss of Mobile Vit for Malaria Type Classification.....	83
Figure 4.9: Confusion Matrix of Mobile ViT for Malaria Type Detection	83
Figure 4.10: Performance metrics across models for malaria type.....	86
Figure 4.11: Login and Signup Pages	87
Figure 4.12: Malaria Diagnosis Application for Thick Blood Smear.....	88
Figure 4.13: Malaria Diagnosis Application for Thin Blood Smear	89
Figure 4.14: Thick Image Result Which Shows Malaria Detection	90

Figure 4.15: The Above One Shows the <i>P. falciparum</i> and the Below Image Shows the <i>P. Vivax</i> . Result.....	91
---	----

LIST OF TABLES

Table 2.1:Challenges and Solutions in Malaria Diagnosis	8
Table 2.2: Developments in neural networks for malaria detection	10
Table 2.3: Accuracy and reliability of automated malaria diagnosis.....	14
Table 2.4: Ethical considerations in AI for healthcare	16
Table 3.1: Data Collection	22
Table 3.2 Use case scenario	60
Table 4.1: Detection of malaria parasites	67
Table 4. 2: Classification of the malaria parasites	71
Table 4. 3: KFold result for malaria parasite classification for MobileViT model	75
Table 4.4: Detection of Malaria Types	77
Table 4.5: Classifying malaria type	81
Table 4. 6: KFold result for malaria type classification for MobileViT	85

ACRONYMS

AI - Artificial Intelligence

ARB: Auxiliary Reversible Branch

CNN - Convolutional Neural Network

CONV: Convolution

ConvNets: Convolutional Networks

CPU: Central Processing Unit

DSR - design science research

EbD - Ethics by Design

ELSI - Ethical, Legal, and Social Implications

FPN: Feature Pyramid Network

GB: Gigabyte

GPU: Graphics Processing Unit

HPC: High-Performance Computing

IoU: Intersection over Union

mAP: Mean Average Precision

NADAM: Nesterov-accelerated Adaptive Moment Estimation

NAS - Neural Architecture Search

NLP: Natural Language Processing

NMS: Non-Maximum Suppression

P. Falciparum - Plasmodium Falciparum

P. Vivax - Plasmodium Vivax

p: Precision

PAN: Path Aggregation Network

QA: Quality Assurance

R: Recall

R-CNN - Region-based Convolutional Neural Network

RDTs - Rapid Diagnostic Tests

ReLU: Rectified Linear Unit

ROI - Region of Interest

RPN: Region Proposal Network

SGD: Stochastic Gradient Descent

SiLU: Sigmoid-Weighted Linear Unit

SPP: Spatial Pyramid Pooling

SSD - Single Shot Multi-Box Detector

TB: Terabyte

TPU: Tensor Processing Unit

U-Net: (Does not have a complete form, named after its U-shaped architecture)

ViT: Vision Transformer

YOLO - You Only Look Once

CHAPTER ONE

1. INTRODUCTION

1.1 BACKGROUND OF THE STUDY

Among the most dangerous public health complications around the world is malaria, particularly prevalent in tropical and subtropical regions.[1] This disease is caused by Plasmodium parasites, transmitted through the bites of female Anopheles mosquitoes. The symptoms can range from mild to life-threatening.[2] In 2022, there were 249 million reported instances of malaria globally, which was five million more than anticipated before the COVID-19 pandemic.[3] This disease continues to be a significant cause of mortality annually, despite various campaigns aimed at its eradication, especially in Sub-Saharan Africa.[4] Consequently, effective therapy necessitates early and accurate diagnosis, underlining the importance of reliable diagnostic tools.

Traditionally, malaria is diagnosed by viewing blood smears under a microscope. This method requires high proficiency as it involves specific staining and microscope examination to detect malaria parasites.[4] The accuracy of this technique heavily relies on the microscopist's expertise, making it labor-intensive and low throughput. Furthermore, in resource-limited settings, the lack of proper laboratory facilities or trained personnel exacerbates the risks associated with accurate diagnosis.

Over the years, the approach to malaria diagnosis has evolved, particularly with the advent of artificial intelligence (AI) and digital imaging. These technologies have introduced new methods for detecting the disease.[5] Machine learning and deep learning, significant advancements in the diagnostic field, rely on algorithms that learn from vast amounts of data to identify patterns and make informed decisions. [6] These techniques are revolutionizing various sectors, including healthcare, by providing new tools for diagnosis.

The YOLO algorithm, or You Only Look Once, exemplifies such advancements in digital image processing.[7] It has gained acclaim as an effective object recognition system and has played a notable role in malaria diagnosis due to its rapid and accurate object tagging capabilities.[8] However, applying YOLO to the diverse and complex terrain of blood smear images presents specific challenges, such as distinguishing tiny malaria parasites from other elements like platelets and white cells, which can obscure the parasites.[9]

To address these challenges, this study employs a restricted image preprocessing strategy that includes a tiling technique. This method partitions an image into manageable parts, focusing primarily on the Region of Interest (ROI). By reducing image complexity, this approach enhances the YOLO algorithm's accuracy in identifying malaria parasites.[10] While it is recognized that both YOLO and image preprocessing can significantly aid in detecting malaria, there is a gap in the literature regarding the integration of these two approaches.[11]

This research aims to fill this gap by comparing current and advanced desktop-based deep learning models for small-object detection in microscopy. It evaluates both the individual and combined performance of these techniques to improve the accuracy of malaria diagnosis in thick and thin blood smear images.[12]

Improved diagnostic techniques have historically aided the effective control and potential prevention of malaria. Current diagnostic tools include traditional microscopy and rapid diagnostic tests, which suffer from drawbacks such as low sensitivity, limited throughput, and heavy reliance on human resources. This study explores how deep learning algorithms like YOLO can overcome these challenges when paired with image preprocessing techniques. By automating parts of the diagnosis process, this approach aims to remove time constraints and enhance the effectiveness of malaria eradication efforts. Ultimately, this research strives to contribute to the fight against malaria in some of the world's most vulnerable communities by implementing more effective technologies in the healthcare sector globally.

1.2 STATEMENT OF THE PROBLEM

Malaria is still a leading public health issue for people across the world, and the case is the same in Ethiopia through strains like *P. Vivax* and *P. Falciparum*. [13] Implementing these laboratories is a problem, and a lack of skilled people hampers the diagnosis of malaria parasites from blood samples in rural areas.[13] Methods commonly used in diagnosis, such as examining blood smears at the level of the glass slide, could miss tiny parasites because they are only visible in a minimal plane of focus. The problem is transferred by the general approaches of deep learning object detection, which encounter certain difficulties when applied to the specific domain of malaria image screening.[14] These models often have issues of losing significant spatial features because high-resolution microscopic image resolutions require downscaling to meet the input of the mentioned models. Thus, the detection results are relatively low.[15]

Driven by these problems, the present research intends to improve the applicability and practicality of diagnosing malaria parasites using digital microscopy and advanced deep-learning algorithms. The Researcher also proposes to reduce the detection of tiny, barely identifiable parasites by employing contemporary computational analysis methods in combination with high-definition digital imaging. This is alongside the researcher's greater mission of deploying state-of-the-art technologies in responding to critical public health issues: malaria prevention and decreasing diseases' effects on targeted communities.

This research uses data from blood smear images of malaria patients taken from the Worabe region, Chewaka, and Jimma for thick and open-source images of thin blood smears. This comprises identification and labeling by laboratory professionals from the Malaria Research Center at Jimma University and checking and solving labeling differences to make the data used to label the models as accurate as possible. This research aims to design and implement a desktop application prototype that uses sophisticated deep-learning algorithms for fast Malaria detection from blood smears. The prototype is at the heart of this research as it represents a proof-of-concept and an instrument that could potentially change diagnostics for malaria, especially in various healthcare settings. The study aims to substantially reduce the malaria burden by demonstrating reductions in diagnostic turnaround times and improved productivity of well-validated detection solutions upon integration into existing healthcare infrastructures.

1.3 MOTIVATIONS

To solve the problem of fighting malaria, the following breakthrough has been proposed: '*A Two-Stage Deep Learning Approach with Image-Tiling for Malaria Diagnosis from Microscopic Blood Smears*'. Due to the high prevalence of malaria in many regions across this globe, there is a need to move to the, which is based on standard diagnostic techniques. The primary purpose of this study is to make changes in the diagnostic process by increasing accuracy and speed and thus reducing the burden of this disease on affected populations worldwide. In line with the urgent global necessity for more efficient and accurate malaria diagnostics, this research explicitly addresses these needs by building AI-powered diagnostic technologies that not only have the potential to enhance health outcomes.

This research employs Deep learning and image processing, potentially impacting different areas, including diagnostics. The researcher determined to apply these advancements to

develop diagnostics that are as accurate and fast as current ones but available to all and accessible to various locations.

In addition to these technical contributions, the researcher's work is a synergy of healthcare, technology, and innovation that looks at the prospect of artificial intelligence revolutionizing medical diagnosis. This is where researchers aim to improve the environment, where great technologies are to be made available to everyone, with a focus on the medical sector and healthcare solutions.


1.4 RESEARCH QUESTIONS

Consequently, the problem of malaria diagnosis calls for creative solutions that increase accuracy and efficiency. So, the researcher framed well-defined research questions to guide our inquiry on a broad spectrum of capabilities and limitations in leveraging sophisticated deep learning models. These questions direct our research focus and are essential in assessing any proposed diagnostic models. By answering these questions, the researcher hopes to make significant advancements in medical diagnostics by using technology to take healthcare rights where traditional infrastructure may not be so profound.

1. What advanced deep learning techniques enhance the accuracy and efficiency of malaria detection in blood smear images?
2. What advancements in deep learning can reduce false positives in malaria detection from blood smear images?
3. How are image-tiling techniques integrated into deep learning models to improve the detection of malaria parasites as small objects in microscopic images?

1.5 RESEARCH OBJECTIVES

1.5.1 GENERAL OBJECTIVES

-  The general objective of this research is to develop and evaluate “A *Two-Stage Deep Learning Approach with Image-Tiling for Malaria Diagnosis from Microscopic Blood Smears*” techniques to improve diagnostic accuracy in laboratory settings.

1.5.2 SPECIFIC OBJECTIVES

The specific objective of this research is: -

- ✚ To develop a Deep Learning Framework equipped with enhanced object detection algorithms for high-accuracy malaria classification in blood smear images, aiming to improve diagnostic speed and efficiency.
- ✚ To optimize pre-train deep learning-based methods to reduce false positives in malaria detection from blood smear images with improved diagnostic accuracy.
- ✚ To investigate the integration of image-tiling techniques within deep learning frameworks to enhance the capabilities of small-object detection models for malaria parasites in microscopic images.

1.6 SIGNIFICANCE OF THE RESEARCH

The following are the significant advantages of this research to various vital stakeholders in highly affected malaria areas: Adopting sophisticated deep learning models will benefit the healthcare facilitators by allowing them to perform faster and more accurate malaria diagnoses. The enhancement of means for the diagnosis of diseases will help to lessen the disease incidence and provide efficient treatment. Government health workers can apply findings from this study to improve the strategies designed for malaria control and prevention, thereby improving the disease's health surveillance and health management. Similarly, the research community benefits because this study accrues much-needed knowledge on using deep learning in medical diagnostics so that it can be applied to other prevalent diseases. Furthermore, the malaria-infested population right from the grassroots level in the malaria-affected regions of the country is likely to benefit through facility improvement and enhanced diagnostic services, reducing the disease's health and socio-economic costs. Thus, signaling out those actual beneficiaries, the research makes a strong case for the study's worth and capacity to make a difference, positively influencing the health systems in those deprived areas.

1.7 SCOPE AND LIMITATION

1.7.1 SCOPE

This research focuses on improving malaria detection for Ethiopia's *P. vivax* and *P. Falciparum* strains. It encompasses aspects beginning with an extensive review of existing technologies

used for malaria detection. The goal is to understand procedures and identify improvements. The research starts the data gathering phase, during which complete information is acquired from the Worabe area, Chewaka, and Jimma for thick and open-source datasets for thin blood smears. This dataset will comprise many malaria cases, giving a solid foundation for further investigation. Experts from Malaria Research Center at Jimma University diagnosticians will rigorously label the information obtained for correctness, which is essential to the research. One exciting aspect of the study is the creation and use of an algorithm that checks the accuracy of expert labels. A third expert with experience will decide on discrepancies, ensuring the data remains reliable. Additionally, a new technique called image tiling will be developed as part of the research. This technique involves overlapping images by 30%. Additionally, the study entails the creation of a desktop application prototype that incorporates this approach of image tiling and deep learning algorithms in diagnosis. This prototype system is expected to show what kind of integration is necessary to incorporate the new methodologies into the current diagnosing processes to improve the efficiency of malaria identification. It will also assist in categorizing these parasites using thin blood films. The primary application area of this technology is enhancing accuracy and specificity in diagnosing malaria to differentiate between different strains. In addition, the research's implications extend to the global level since the study intended to set standards in malaria diagnosis that, if enhanced, will positively affect global public health and medical advancement.

1.7.2. LIMITATION

While the tiling approach with a 30% overlap is helpful for malaria detection utilizing models such as YOLOv9 and RT-DETR, the models' real-time detection skills strongly depend on the input data's quality and variety. Despite allowing for a more extensive inspection of blood smears, changes in malaria parasite growth stages and strain shape can make correct identification difficult. These models are primarily sensitive to changes in other datasets except the training dataset; hence, they may be limited in their application in a highly variant real-world environment. Furthermore, because training mainly uses samples from specific areas, the models may be biased toward identifying common local malaria strains. When the models are used in locations with diverse malaria variations, this bias may result in increased misdiagnosis rates. To improve the robustness and portability of the models, it is necessary to periodically update the datasets with diverse and representative examples and test the models in geographically diverse malaria-endemic regions.

CHAPTER TWO

2. REVIEW OF RELATED LITERATURE

2.1 CHALLENGES AND SOLUTIONS IN MALARIA DIAGNOSIS

2.2.1 THEORETICAL BACKGROUND AND LITERATURE REVIEW ON MALARIA DETECTION IN ETHIOPIA

In the past, Ethiopia has relied on two methods to identify malaria: microscopic analysis of blood smears and clinical diagnosis. Although symptom observation-based clinical diagnosis is a commonly available technique, its accuracy is compromised by overlap with other febrile disease symptoms.[16] The gold standard for identifying malaria parasites in blood smears is microscopic inspection, which provides excellent accuracy in determining the parasites.[17] However, the availability of trained workers, advanced laboratory equipment, and steady energy, frequently scarce in rural and resource-poor regions, is essential to the method's efficacy.[18]

Rapid diagnostic tests (RDTs), particularly in remote areas, have emerged as a viable option. Identifying malaria antigens in blood makes these point-of-care tests quick and straightforward, yielding findings in minutes. [19] However, variables like user skill and storage conditions might affect their accuracy. [20]

Future developments in digital imaging and machine learning techniques, such as YOLO, might provide new paths for automated parasite detection in blood smear pictures, thereby increasing speed and accuracy when conventional microscopy is not feasible. [28,7] However, there are challenges involved in implementing these technologies in Ethiopia. The country must enhance its infrastructure, provide training to healthcare professionals, and adapt algorithms to accommodate regional variations in parasite strains and imaging techniques.[29,30]

The existing literature highlights the need for diagnostic techniques and ongoing research when it comes to diagnosing malaria in Ethiopia.[24] It emphasizes how crucial it is to design solutions that are particular to the epidemiological and infrastructure realities of the nation's many regions.[1] Developments in diagnostic technology and current practices are critical to Ethiopia's campaign to eradicate malaria. Ethiopia must prioritize context-specific development, overcome infrastructure barriers, and adopt digital solutions to eliminate malaria and pave the path for a healthier future.

2.2.2. CURRENT METHODS OF MALARIA DIAGNOSIS

Ethiopia uses three diagnostic methods from this clinical diagnosis based on observing symptoms like fever, chills, and headache. However, these methods are limited in accuracy due to their overlap with other symptoms.[16] The microscopic study of blood smears, regarded as the gold standard, offers high accuracy in identifying malaria parasites. However, its effectiveness depends on the person's skills, sophisticated equipment, and stable electricity, especially when it is difficult in rural areas.[25] Rapid diagnostic tests (RDTs) are point-of-care tests that provide quick results within minutes through malaria antigen detection in blood. They are helpful for rural areas but can be affected by storage conditions and user proficiency.[19]

2.2.3 SYSTEMIC CHALLENGES AND TECHNOLOGICAL OPPORTUNITIES

Different assessments show three significant challenges to improving or effectively using the current system.

- 1) Limited healthcare infrastructure - Most rural areas often lack trained personnel, equipment, and stable electricity for conventional methods.[18]
- 2) Digital infrastructure gap - a tedious process of permitting digital solutions in the government (Minister of Health-Ethiopia).[26]
- 3) Adapting algorithms - Machine learning algorithms must be trained and optimized for Ethiopia's local parasite strains and image variations.

Table 2.1:Challenges and Solutions in Malaria Diagnosis

Study Reference	Challenges Identified	Impact on Diagnostics	Potential Solutions/Remark
[16] Opoku Afriyie et al., 2023	Limited access to advanced diagnostic equipment and reliable electricity	Delays in diagnosis and increased risk of incorrect treatment	Mobile and solar-powered diagnostic devices could mitigate some of these issues.

[18] Yalew, 2022	Shortage of trained healthcare personnel for malaria diagnosis	Reliance on less accurate methods like RDTs; potential for misdiagnosis	Training programs and AI-based diagnostic tools could help bridge the skill gap
[17] Hammami, Nuel, & Garcia, 2013	Inadequate health care infrastructure for storing and processing diagnostic samples.	Compromises the quality and timeliness of malaria diagnosis.	Developing robust, low-cost diagnostic technologies that require minimal infrastructure
[20] Boyce et al., 2018	Variability in the performance of RDTs due to storage conditions and handling.	This leads to decreased confidence in diagnostic outcomes, affecting treatment decisions	development of more resilient diagnostic tests.

2.2 DEVELOPMENTS IN ADVANCED NEURAL NETWORKS FOR MALARIA DETECTION.

Improving neural networks in malaria elucidates the use of deep learning to boost the odds of correct diagnosis. The recent developments categorize the use of more sophisticated models like Convolutional Neural Networks (CNNs) since they reap better image processing than conventional models for identifying malaria from blood samples. This progress speaks to the necessity of improving and evolving AI-driven methodologies in the medical diagnostic field, marking the tendency towards augmenting the identification methods while not highlighting the particular accuracy rates.[27]

The advancement of significant deep convolutional neural networks is most profound in developing neural networks for malaria detection. This technique advances the rapid diagnosis

of malaria parasites from red blood cell smears by allowing feature extraction and classification directly from picture patches. This strategy departs from the conventional feature engineering manual approach. This is a strength of deep learning and can enhance diagnosis using medical images.[28] Advances in neural network applications for malaria detection demonstrate the transition to automated diagnostic processes. Utilizing convolutional neural networks (CNNs), this research accelerates malaria parasite detection from red blood cell smears by speeding feature extraction and classification straight from segmented picture patches, indicating a considerable improvement over the standard human approach.[29]

Efficiencies in traditional malaria diagnosis by introducing a robust machine learning model using convolutional neural networks (CNNs) to classify and predict infected cells in blood smears automatically. The study compares three CNN models—Basic CNN, VGG-19 Frozen, and VGG-19 Fine Tuned—on 27,558 single-cell images to determine the most accurate model for malaria detection.[30] Combining two approaches, supervised learning, and segmentation in malaria diagnosis, brings an understanding of how applying deep learning in public health is feasible. It also reviews the existing limitations associated with present-day implementations of deep learning techniques, including issues in data preprocessing and computational requirements for the given diagnostic systems. It outlines the research advancements likely to improve real-time analysis and explain the ability of deep learning medical diagnosis applications.[31]

A new deep neural network model adapts a transfer learning method to improve the performance of the malaria diagnostic capacity in detecting infected falciparum parasites. By employing a hybrid of the pre-trained VGG layers and applying SVM to the extracted features from the VGG layers, this approach outperforms the conventional CNN model as a diagnostic tool for malaria diagnosis from microscopic images.[32]

Table 2.2: Developments in neural networks for malaria detection

Study Reference	Method	Advantages	Limitations
[27] Gourisaria et al., 2020	CNNs for malaria detection from blood samples	Utilizes sophisticated models for improved image processing.	Does not explicitly highlight accuracy rates.

		Enhances the accuracy and efficiency of diagnosis.	
[28] Rahman et al., 2019	Deep CNN for red blood cell malaria detection	Advances rapid diagnosis by allowing direct feature extraction and classification from image patches.	It departs from traditional feature engineering, which could limit the understanding of underlying features.
[29] Zhao et al., 2020	CNNs for automated malaria screening in low-resource settings	Speeds up feature extraction and classification, improving over manual methods.	General limitations of deep learning such as data dependency and computational demand.
[30] Shekar et al., 2020	Comparative analysis of Basic CNN, VGG-19 Frozen, VGG-19 Fine Tuned	Identifies the most accurate model for malaria detection from single-cell images.	It focuses only on technical efficacy without addressing real-world application challenges.
[31] Jdey et al., 2023	Deep learning for malaria detection: challenges and future directions	Reviews profound learning potential in public health, suggesting real-time analysis and	Highlights deep learning's current limitations in data preprocessing and computational needs.

		explanatory power improvements.	
[32] Vijayalakshmi and Rajesh Kanna, 2020	Transfer learning with VGG and SVM for malaria detection	Demonstrates superior performance over conventional CNN models in malaria diagnosis.	There is limited discussion on the integration challenges with existing medical diagnostic frameworks.

2.3 ACCURACY AND RELIABILITY OF AUTOMATED MALARIA DIAGNOSIS

Focuses on identifying malaria parasites in densely populated blood smear microscopic images using modified YOLOV3 and YOLOV4 models struggle armed with two potent weapons: adjusted versions of YOLOV3 and YOLOV4 deep learning algorithm. To accurately detect malaria parasites in microscopic blood smear samples, paving the way for faster, more accessible diagnosis. The authors meticulously modify these cutting-edge object detection algorithms, equipping them to home in on Plasmodium's tiny, telltale signs. They improve the models' tiny object competence by increasing feature sizes and adding detection layers, making them laser-focused on malaria parasites. As well as the outcomes go over. Among the models tested on an available dataset, the modified YOLOV4 stands out as the top performer, achieving an impressive 96.32% mean average precision in accurately identifying parasites. The modified YOLOV3 models demonstrate their value by surpassing the versions and other detection algorithms like SSD and Faster R CNN. However, despite these findings, the paper openly acknowledges room for improvement. To enhance the credibility of these models, additional information about the dataset, visualizations of results, and ablation studies to analyze modifications would be beneficial. Moreover, comparing detection techniques and assessing a wide range of datasets would further establish their prominence in this essential field. While there may be room for improvement, the paper exhibits strengths. It addresses a health issue directly by proposing deep learning models demonstrating exceptional

performance. By enhancing the speed, accuracy, and accessibility of diagnosing jungle fever, this research holds the potential to save lives and empower communities in their battle against this disease.[33]

A revolutionary paper proposes a smartphone app that tackles malaria head-on. This app, integrated with deep learning, automatically detects and counts malaria parasites in thick blood smears, offering a promising alternative. It works in two stages: an intensity-based screening identifies potential parasites, while a custom CNN classifies each candidate. Powered by a large, publicly available dataset, the app boasts impressive performance. However, it's not all sunshine and smears. Some key areas beg for further exploration. Comparing the app to existing methods would solidify its unique strengths and weaknesses. Understanding its computational demands is crucial for ensuring smooth smartphone compatibility. Finally, hearing from real users through studies would validate its practicality and identify potential improvement points. Despite these gaps, this research ignites hope. A future where accessible, AI-powered malaria diagnosis fits in the researcher's pocket is closer than ever, potentially saving lives and empowering communities. This paper paves the way, demanding further refinement to translate its brilliance into real-world impact.[8]

Traditionally, malaria diagnosis has danced to the microscope's tune, relying on skilled technicians to spot Plasmodium parasites in high-resolution, detail-rich smears. This paper proposes a bold leap forward: tile-based image processing that cuts these images into manageable chunks, empowering deep learning models like YOLOV4 to excel. The result? A staggering 95.3% recall and 87.1% precision, outperforming the old guard and generalizing adeptly to diverse datasets. This translates to faster, more accurate diagnoses, potentially reducing technicians' workload and opening doors to real-time, mobile phone-based malaria detection. This isn't a revolution in detecting malaria; it represents an advancement for global health, demonstrating how technology can make a difference in the ongoing battle against this persistent enemy.[15]

Aiming to revolutionize malaria diagnosis, this paper proposes a smartphone-integrated Microscope System for detecting parasites in dense blood smear samples in real time. It tackles the issue of limited skilled personnel by developing a low-cost, rapid screening solution. The heart lies in a modified YOLOv3 algorithm on a custom CNN, trained on diverse datasets from Ethiopia, India, and Thailand. This pipeline filters candidate parasites using an intensity-based method and classifies them with CNN. With impressive accuracy exceeding 97%, surpassing

even human experts, the technique showcases its potential to significantly aid disease screening and combat malaria effectively in regions struggling with high disease burden and limited resources. Future endeavors aim to refine robustness, expand compatibility with other parasite types, and develop a user-friendly interface for real-world deployment. This innovative approach paves the way for a future where affordable, accessible malaria diagnosis empowers resource-constrained communities to combat this deadly disease.[34]

Table 2.1: Accuracy and reliability of automated malaria diagnosis

Study Reference	Methods	Advantages	Disadvantage
[33] Abdurahman, Fante, & Aliy (2021)	Evaluated modified YOLOV3 and YOLOV4 models for detecting malaria parasites in blood smear images.	High accuracy (96.32% for YOLOV4) in identifying parasites. It improved tiny object detection.	Room for improvement in dataset diversity and model generalizability.
[35] Yang et al. (2020)	Developed a smartphone app using deep learning for malaria parasite detection in thick blood smears.	Offers a promising alternative for quick diagnosis. Utilizes a two-stage process for enhanced accuracy.	Computational demands and user experience need further exploration. Comparison with existing methods is required.
[15] Shewajo & Fante (2023)	Applied tile-based image processing for malaria screening using YOLOV4.	Achieved 95.3 % recall and 87.1 % precision, demonstrating the potential for mobile-based real-time diagnosis.	No post-analysis to reduce false positives, limiting reliability and clinical usability in real-world applications. Further

			refinement is needed.
[34] Chibuta & Acar (2020)	Proposed a smartphone-integrated microscope system using modified YOLOv3 for real-time screening of malaria parasites.	Showcased over 97 % accuracy, potentially exceeding human expert performance	Future work is necessary to refine robustness, compatibility with various parasite types, and user interface.

2.4 ETHICAL AND PRACTICAL CONSIDERATIONS IN IMPLEMENTING ARTIFICIAL INTELLIGENCE IN THE HEALTHCARE SECTOR

Because the rapid integration of AI in healthcare demands a thorough analysis of its ethics, it is necessary to unveil the ethical ramifications associated with this technology. A comprehensive scoping review was carried out between 2010 and July 21, and I got the answer by unleashing these ethical concerns. The study, which scrutinized a wide array of literature from databases like Medline (PubMed) and Embase (OVID), focused on fundamental concepts including individual autonomy, prevention of harm, equity, clarity, and confidentiality. Despite the considerable volume of articles, there was a noticeable lack of depth in the examination of these principles, particularly in AI's design or deployment, with harm prevention being the least discussed aspect.[36]

Simultaneously, the potential of AI to revolutionize healthcare is enormous, but its integration raises significant ethical and legal challenges. A review of 1238 articles, narrowed down to 16, highlighted these challenges, emphasizing the need for collaboration among policymakers, developers, medical practitioners, and patients for effective resolution.[37]

Furthermore, AI's revolutionary impact in the medical field in the 21st century, with its unique characteristics and inherent risks, has sparked various ethical and legal concerns. An early-stage comprehensive analysis suggests that while current frameworks are mainly capable of

addressing these challenges, sector-specific legal revisions, particularly concerning non-discrimination and product liability, might be necessary.[38]

A comprehensive examination addressing the Ethical, Legal, and Social Implications (ELSI) of artificial intelligence in the medical industry also advocates for an "Ethics by Design" (EbD) methodology. This approach integrates ELSI considerations early in the AI design process, targeting various stakeholders such as designers, engineers, and clinicians. The review, which analyzed 94 papers out of an initial 1108, revealed increasing academic interest in AI's ELSI across various clusters, including AI algorithms, physicians, patients, and the healthcare system. These discussions indicate that while AI promises significant improvements in patient care, the complex ELSI surrounding its implementation in healthcare remains crucial for consideration. The incorporation of Artificial Intelligence in decision-support systems. Suggests an evolution from a mutual patient-doctor relationship to a more complex, trilateral dynamic involving AI. Addressing these ethical and legal challenges requires a multi-faceted strategy involving a broad spectrum of stakeholders to guarantee the advantageous incorporation of AI into medical systems.[39]

This study elevates malaria diagnosis by combining existing methods with innovative enhancements. It advances YOLO-based detection by incorporating an image processing layer specifically tailored for blood smear images. The objective of this approach is not only to detect the presence of malaria parasites but also to identify their specific types. This strategy is particularly designed for regions with limited medical resources, offering significant improvements in diagnostic accuracy and processing speed. Consequently, this study aims to develop an effective and efficient solution for the healthcare system, predominantly in areas most affected by malaria. By focusing on enhancements in accuracy and inference time, this research represents a substantial advancement in the diagnostic techniques for malaria, potentially having a profound impact on public health.

Table 2.2: Ethical considerations in AI for healthcare

Study Reference	Ethical Issues	Impact/Concerns	Suggested Measures
[36] Karimian et al., 2022	Privacy and Data Security	Risk of sensitive patient data exposure.	Implement robust data encryption and access controls.

			Promote transparency in data handling
[37] Prakash et al., 2022	Bias in AI Models	AI algorithms may perpetuate or amplify existing healthcare disparities.	Incorporate diverse datasets in training. Regularly audit AI systems for bias.
[38] Schöonberger, 2019	Equitable Access	Disparities in access to AI-powered health care solutions.	Develop policies to ensure AI benefits are accessible to all populations, regardless of socioeconomic status.
[39] Cartolovni et al., 2022	Transparency and Accountability	Difficulty in understanding AI decisions can erode trust in healthcare.	Foster AI systems that are explainable and transparent. Establish clear accountability for AI decisions.
[40] Elyan et al., 2022	Informed Consent	Challenges in obtaining informed consent for using personal data in AI research.	Enhance patient understanding of AI use. Ensure consent processes are clear and comprehensive.

CHAPTER THREE

3. METHODOLOGY

3.1 RESEARCH DESIGN

The study also applies the Design Science Research (DSR) paradigm, a constructive methodology for creating artifacts to solve problems. The DSR framework is appropriate for this research since it helps design an original tool, the desktop application for malaria detection.

3.1.1 RESEARCH PARADIGM

The researcher adopts the Design Science Research (DSR) methodology, a paradigm centered on the life cycle and the iterative enhancement of the developed artifact. In Figure 3.1, the DSR framework is structured into three main stages: Input, Processor, and Output. Such a method is crucial when developing information systems solutions where this approach is systematic, with feedback at each step to improve the process further. These loops allow the data and information to flow from the Output stage to the Processor stage and from the Processor stage to the Input stage. This gives an iterative feedback process for each phase of the research, which is made to improve and become more specific with every phase, with decisions made based on previous phase results.

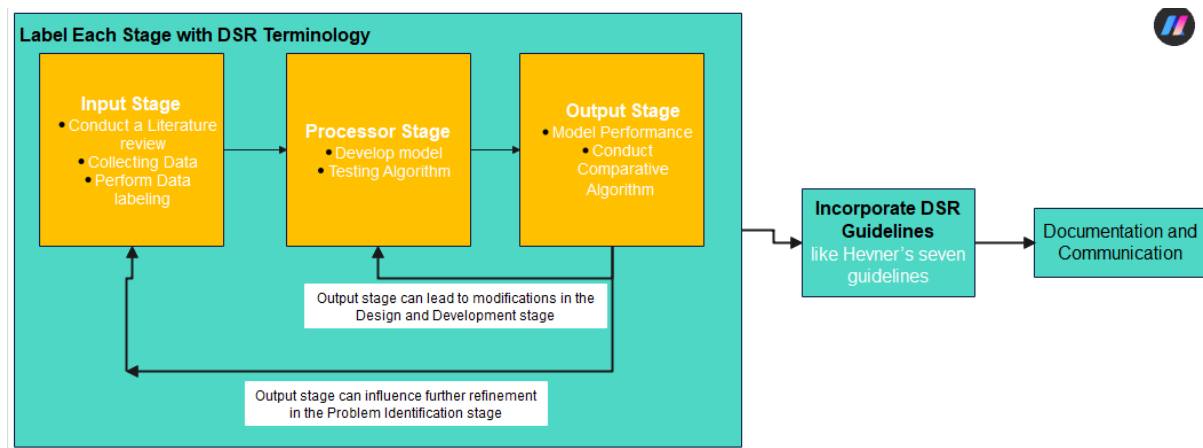


Figure 3.1: Research Paradigm

This diagrammatic representation signifies compliance with the DSR guidelines, which bring out the nature of the research, starting from the identification of the problem to the construction of the artifact, its testing, and, consecutively, its improvement. It was discovered that the DSR framework means not only adherence to the applied protocols but also developing

improvements relying on knowledge gained from empirical research and testing. This is particularly important in creating a sound artifact that best suits the dynamic context that characterizes application-driven environments. The above diagram shows the iterative nature of DSR, where the output of one stage is being used to enhance the next stage, and it forms a progression cycle that is inherent to design science research.

3.1.2 RESEARCH FLOW

As illustrated in Figure 3.2, the research flow chart presents a straightforward and elaborative completion procedure for every element of the study, thus supporting the research paradigm in removing the outlined stages.

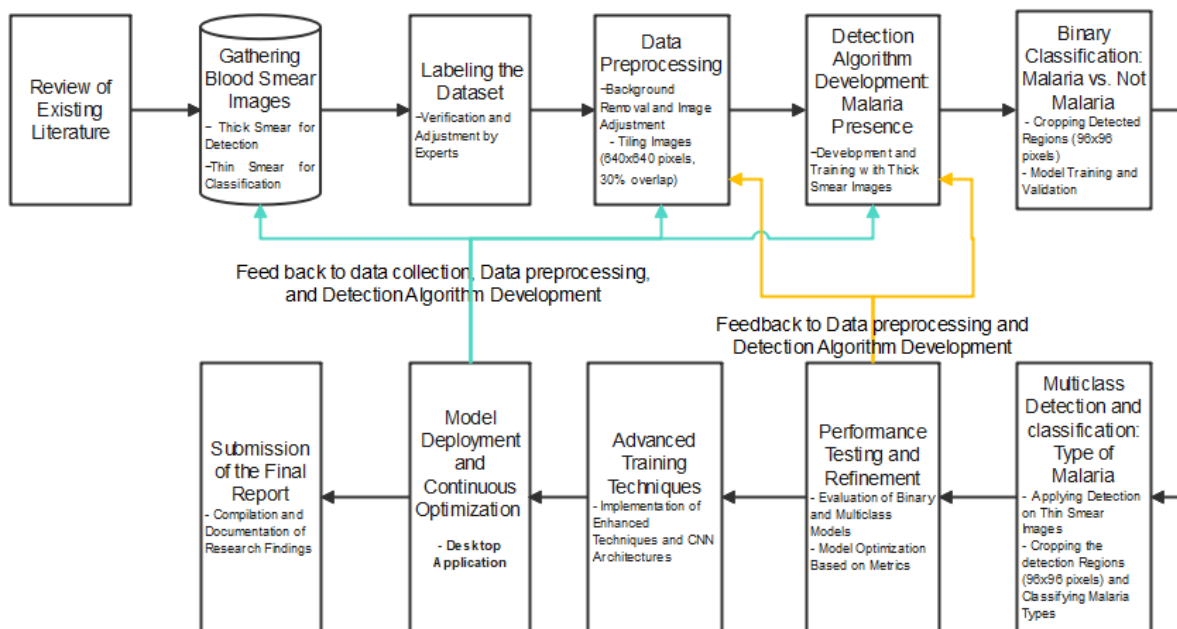


Figure 3.2: Research Flow

- *Initial Steps*: Data collection and literature review to create the background for using deep learning models.
- *Development and Testing*: The central component of the processor stage refers to the construction and improvement of models through a cyclical process to maximize their applicability and effectiveness in practice.
- *Final Evaluation*: This entails assessing the developed models against the standard models to check their efficiency and readiness for implementation.
- *Performance Testing and Refinement*: In this last evaluation stage, the models are compared with those benchmarks to evaluate their efficiency and practical application

ability. This feedback interacts with stages such as data preprocessing and model training, which modify the models and the deployment models.

- *Model Deployment and Continuous Optimization:* After that, models are released in a natural environment where constant improvement occurs depending on accurate data and users' opinions. It underscores the fact that the use of the tools must be followed by feedback and improvement as they are deployed continually from one setting to another.

The flow underpins the research paradigm combined with the arterial and step-by-step description of how the theoretical concepts of DSR are integrated into the research.

3.2 SOURCES OF DATA

The researcher carefully chose data sources that combine real-world relevance and academic rigor to create Deep Learning Models for Malaria Diagnosis.

3.2.1 PRIMARY SOURCE

1. **Local Clinical Blood Smear Images:** This research is based on blood smear images of malaria patients taken from the Worabe region, Chewaka, and Jimma for thick and open-source images of thin blood smears. These images are not representations. Field Actual samples of malaria were collected in various forms. The primary data source is essential because it supplies an original reference point for the research detection system.
2. **Annotations from Malaria Experts:** The researcher collaborates with malaria experts to boost the accuracy and detail of the information. They analyze the collected blood smear images and provide insights through annotations. This expert analysis enhances the accuracy and relevance of the researcher's data set.

3.2.2 SECONDARY SOURCE

1. **Academic Literature and Studies:** In this research, the researcher ventures into the treasure trove of extant literature on malaria detection. This includes observing published articles and findings from previously conducted research, which is the process that the researcher uses to incorporate the researcher's methodology into a

scientific context, like weaving a carpet from knowledge gathered by the research community.

2. The researcher has creatively used technological instruments, using existing resources like YOLO-V9 and RT-DETRs algorithms accompanied by trained malaria classification models. With their demonstrated capabilities, these instruments act as a starting point for the researcher, allowing the researcher to concentrate on adapting and improving them according to our research objectives.

In conclusion, the researcher's approach combines the authenticity of data with the richness of collective scientific knowledge and technological innovation. This combination enables our malaria detection research to be grounded in real-world data and on the front line of developments.

3.3 DATA COLLECTION

This study utilized a combination of controlled imaging techniques and systematic data gathering to assemble a robust dataset for training deep learning models. The data comprises both thick and thin blood smear images, specifically collected from the Worabe area, Chewaka, and Jimma, as well as supplemented by an Open-Source Dataset. The collection involved 1,378 thick blood smear images and 800 thin blood smear images, ensuring high-quality visibility of malaria parasites through a controlled environment.

Table 3.1: Data collection

Database	Data size	Image size	Image format
Worabe area, Chewaka, and Jimma	1378	2988x5312	JPG
Open-Source Dataset [41]	800	2752x2192	TIF, JPG

To ensure the computer systems could effectively analyze the images, all samples were converted into digital formats suitable for processing (JPG for thick smears and a combination of TIF and JPG for thin smears). The criteria for selecting blood samples involved a strict protocol aimed at maximizing the visibility of malaria parasites, including factors such as parasite density, blood cell condition, and smear quality.

The data was pre-processed through an image-tiling technique to create smaller, manageable segments for detailed analysis. For thick blood smear images, this process generated a total of 19,263 tiles, while for thin blood smears, a total of 5,971 tiles were produced. The thin smear tiles were further classified into two primary types of malaria parasites: *Plasmodium falciparum* (PF) and *Plasmodium vivax* (PV), with 2,890 tiles for PF and 2,896 tiles for PV.

Class Distribution and Data Balancing: The study specifically addressed potential class imbalances by ensuring a nearly equal distribution of tiles for PF and PV in the thin smear dataset to mitigate bias during model training. The class distribution for the thick blood smear dataset primarily focused on differentiating between malaria-infected and non-infected tiles, whereas the thin blood smear dataset emphasized the distinction between the two parasite types. This balanced approach helps in evaluating the quality and representativeness of the datasets, providing a clear understanding of the dataset's strengths and limitations.

By presenting the class distribution and details of the data collection process, this section aims to clarify the methodological rigor involved in preparing the datasets for subsequent analysis. This detailed documentation supports the assessment of dataset quality and reliability, essential for the robust training and evaluation of the employed models.

3.4 DATA ANNOTATION

To guarantee the precision of ground truth labels for Researcher images, they were annotated by two microscopy specialists (Lab. Expert Tirusew and Lab. Expert Fetya). Due to the subjective nature of hand tagging, an algorithm was created to automatically identify conflicts among these annotations. A third (Lab. Expert wondeade), a more experienced specialist decided on the latter discrepancy after comparing the marked images to the originals. This automated procedure, implemented before the expert assessment, improves the dependability of the annotation process, which is essential for providing precise data to our detection models. This systematic approach minimizes bias and successfully enhances the verification of data used in teaching recognition algorithms.

3.5 DATA PROCESSING

The unprocessed images received preprocessing to improve model training and performance. Consequently, the raw images underwent preprocessing to enhance the models' efficacy and attain a closer alignment with the preprocessing of the raw images utilized in model training.

- **Border Cleaning:** Whenever it works with an image, it performs an essential preprocessing operation before capturing a snapshot of the image. This method involves erasing the noise pixels, which have a minimal impact on the image's boundary region.
- **Image Tiling:** In this study, the image tiling process was applied with both horizontal and vertical overlaps of 30% to ensure that no critical information near the borders of the tiles was lost. This overlapping strategy was crucial for maintaining the integrity of the features within the blood smear images, which are essential for accurate malaria detection.

The tiling process divided each original thick and thin blood smear image into multiple sub-images of 640x640 pixels. With the original dimensions of the thick blood smear images being 2988x5312, a simplistic calculation without considering overlaps would indeed suggest a potential generation of $486 \times 8 = 48$ tiles per image, leading to $48 \times 1378 = 66,144$ tiles in total. However, the effective number of tiles was significantly reduced to 19,263 after **discarding tiles that did not contain any detectable parasites or relevant features**. For thin blood smear, 800 images were tiled to 5971 after removing parasite-free images. This reduction was due to the application of a precise labeling process that involved the removal of tiles lacking necessary diagnostic content, ensuring only the most informative tiles were retained for model training.

Justification for Overlapping Strategy: The choice to implement a 30% overlap both horizontally and vertically was driven by the need to prevent edge artifacts—a common issue in image processing where the information at the margins of an image is lost or distorted. Overlapping ensures that features near the edges of one tile are captured more centrally in adjacent tiles, thus preserving crucial information necessary for accurate analysis.

Formulation of Image Tiling: Consider an image with dimensions of $W \times H$ (W is the width, and H is the height). The image can be divided into tiles $w \times h$. An observer may ask how many tiles would be sufficient to cover the whole input source without overlapping. The answer to this question could be measured as.

$$\text{Number of horizontal tiles } \text{Tiles}_w = \frac{W}{w} \dots\dots\dots \text{Equation 3.1}$$

Number of vertical tiles $Tiles_H = \frac{H}{h}$ Equation 3.2

Where $\lceil \cdot \rceil$ denotes the ceiling function, ensuring that even if a part of the image extends beyond the whole tile dimension, an additional tile is used to cover it.

Keeping Track of Overlap: Tiles overlap to prevent edge artifacts and ensure complete analysis. When an overlap is added, you will lose the size of this overlap from effective tile dimensions:

Effective Tile Width (considering overlap $overlap_W$)

$$w_{eff} = w - overlap_W \text{ Equation 3.3}$$

Effective Tile Height (considering overlap $overlap_H$)

$$h_{eff} = h - overlap_H \text{ Equation 3.4}$$

Consequently, The adjusted number of tiles needed becomes

Horizontal tiles with overlap

$$Tiles_W = \lceil \frac{W - overlap_W}{w_{eff}} \rceil \text{ Equation 3.5}$$

Vertical tiles with overlap

$$Tiles_H = \lceil \frac{H - overlap_H}{h_{eff}} \rceil \text{ Equation 3.6}$$

Advantages of Image Tiling for YOLO Over Adjusting the Backbone CNN's

Stride: While adjusting the stride of the backbone CNN in YOLO allows the model to process the entire image in a single pass and potentially identify small objects, it can lead to a loss of global context and finer details critical in medical diagnostics. Image tiling, in contrast, allows for a more focused analysis of smaller regions, ensuring that local features are not overshadowed by the broader image context. This method is particularly advantageous in medical imaging where the clarity of local features can be paramount for accurate diagnosis. Additionally, tiling enhances the manageability of

high-resolution images, making them more tractable for deep learning models without the computational expense of processing very large inputs at once.

This section aims to clarify the methodology and rationale behind the image-tiling process used in this research, addressing the specific challenges of malaria parasite detection from microscopic images and enhancing the robustness of the dataset used for training the deep learning models.

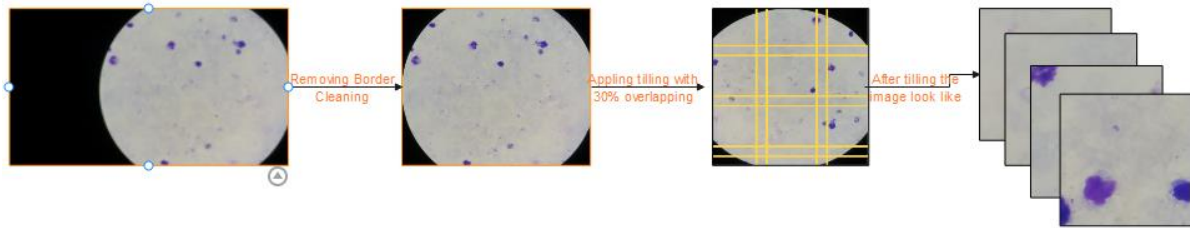


Figure 3.3 Data Preprocessing Method

3.6 DATASET SPLITTING

The prepared dataset was split into three subsets: Thus, the prepared dataset was divided into three sets:

- **Training Set (70%):** In creating the models in this study, the objective is to train the models to diagnose malaria from the blood smear image.
- **Validation Set (20%):** Applied also during the tuning of the hyperparameters, as well as to get hold of an external set that can be utilized in evaluating a fit during the training of the model.
- **Test Set (10%):** Employed to carry out the process of the final model's generalization to the new data.

3.7 FEATURE EXTRACTING

The extraction of features is a vital concept in deep learning techniques where the input information, including the images, is processed to produce structures that enable the model in facets such as object detection.[42] Current state-of-the-art models such as YOLOv9 and what seems to be a real-time DETR, which is the Detection Transformer, employ some of the most elaborate forms of neural networks to detect and recognize objects in an image efficiently.[43][44] YOLO (You Only Look Once) has a grid-based feature extraction feature

that utilizes convolution layers associated with spatial hierarchies to help in efficient object detection.[43] On the other hand, DETR proposes an entirely different solution, in which transformers are included in the detection framework. The self-attention mechanism allows the images to be processed as sequences, and direct object detection is achieved without using region proposal networks.[44]

This research utilizes advanced deep-learning models to enhance malaria detection in blood smear images. The chosen models YOLOv9, DETR, pre-trained Convolutional Neural Networks (CNNs), and Vision Transformers are applied to improve diagnostic accuracy and speed.

3.7.1 YOLOV9 FEATURE EXTRACTION AND DETECTION

YOLOv9 is applied based on the CNN architecture, which is fast and precise, especially in object detection. Next, just like in other YOLO versions, the feature extraction component works with a backbone network that processes input images and extracts rather vital features. This backbone is usually a version of the Darknet architecture that has been gradually developed to address various object detection practices efficiently.

YOLOv9 is implemented for its real-time detection capabilities, which are critical for identifying malaria parasites efficiently. This model's ability to process images swiftly aligns with our objective to enhance diagnostic speed without compromising accuracy, making it an ideal choice for immediate malaria identification in resource-limited settings.

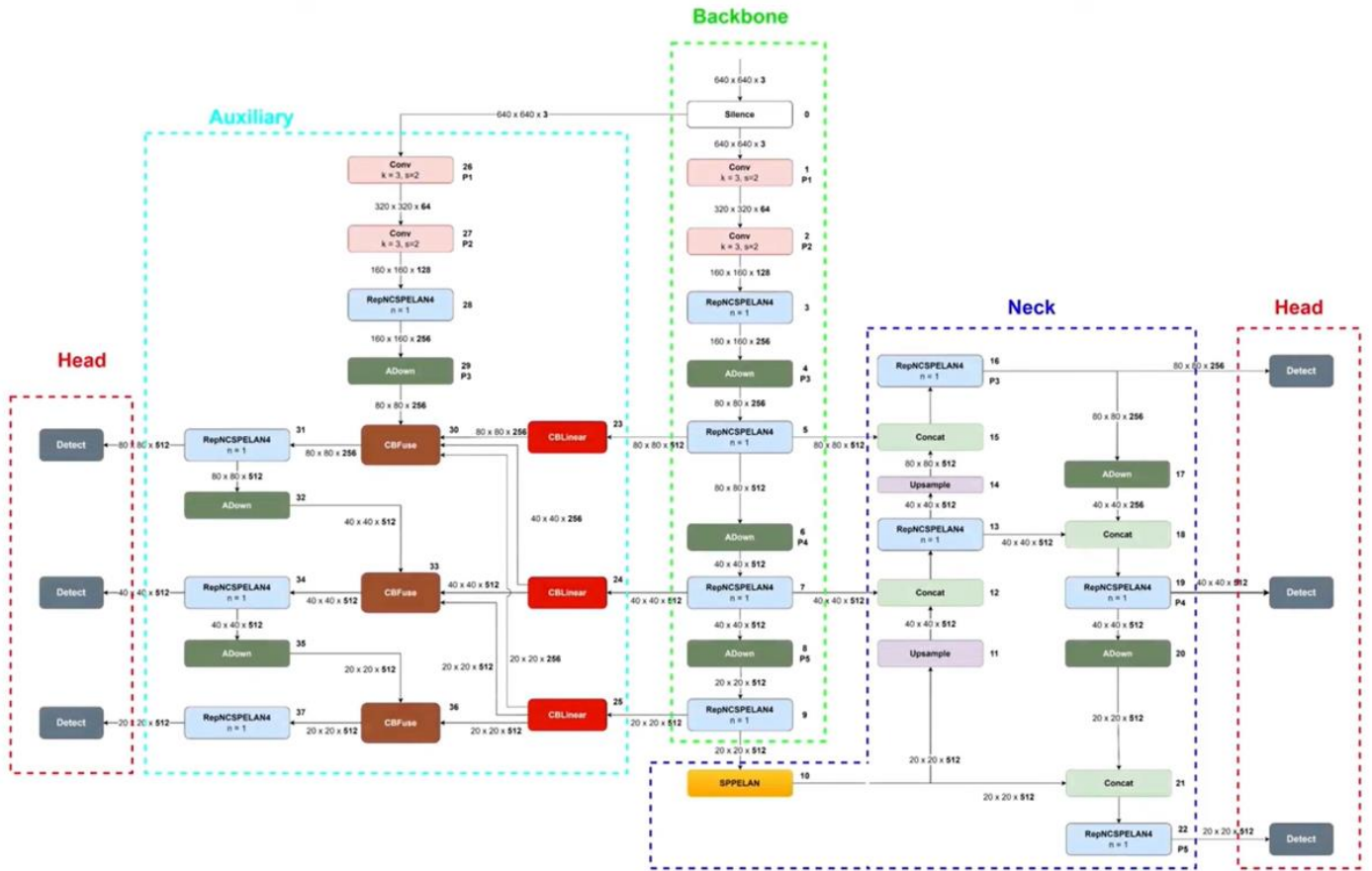


Figure 3.4: YOLOv9 Architecture [45]

Backbone Network

The backbone is the initial part of the network that deals with raw input images. In YOLOv9, this would typically be a convolutional neural network (CNN) designed to extract a hierarchy of features from the images. Each layer of the CNN captures different levels of detail:

This is the first layer of the network that receives images as raw inputs, also called feedback. In YOLOv9, this would have been a CNN used to feature extracting the captured images. Each layer of the CNN captures different levels of detail: Earlier, it was indicated that some of the layers in the CNN discover different features, including the following:

- Low-level features (like edges and colors) are captured in the early layers.
- Mid-level features (like textures and patterns) are captured in the middle layers.
- High-level features (like object parts) are captured in deeper layers.

These are passed through convolution and pooling layers and non-linear activation functions like ReLU or SiLU. These help the network learn the intricate details and features in the image's flow.

Detailed Backbone Architecture

1. Convolutional Layers:

YOLOv9 uses a hybrid of traditional and depth-wise separable convolutional layers to tailor the feature extraction process under efficiency constraints. These layers use kernels that pass through the input image to detect low-level features (e.g., edges and textures first, and then foundations of structures) and gradually build to more abstract features. With strategically placed depth-wise separable convolutions, the network makes up for a reduction in the number of parameters to be trained and computational load, which is desirable in mobile and edge computing scenarios.[46] YoloV9 uses auto padding, which is applied at the kernel level. The kernel strides are maintained by reducing the kernel size by a factor of 2.

$$P = K // 2, \text{ Where } P = \text{padding}, K = \text{kernel size} \quad \text{Equation 3.7}$$

2. Activation Functions:

The SiLU (Swish) activation functions are applied to each convolution layer in YOLOv9. For the same reason, SiLU is much better than the discontinuous ReLU simply because SiLU has a smooth non-monotonic curve that assists in how the gradients can flow (thus reducing variance in significance relevance for the vanishing gradient problem). This property makes SiLU very effective in deep networks, as retaining a solid gradient signal is essential for successful training. Using SiLU may help YOLOv9 better learn more complicated patterns/nuances in the data.[47]

$$x * \left(\frac{1}{1 + e^{-x}} \right) \quad \text{Equation 3.8}$$

3. Pooling Layers:

Pooling layers bring down the spatial dimensions (height & width, but not depth) of the coming input volume of the subsequent convolutional layer. The most commonly applied pooling function used for feature down sampling is the MAX operation applied in the MAX POOLING layer to ensure the detection of features from the scaling and orientation effects. [48] The max-

pooling layer has a window that analytically moves over the given input and, in doing so, saves the maximum value of the window in the output layer. The average-pooling layer shifts a window across the entire input, where the output value corresponds to the window's average values.

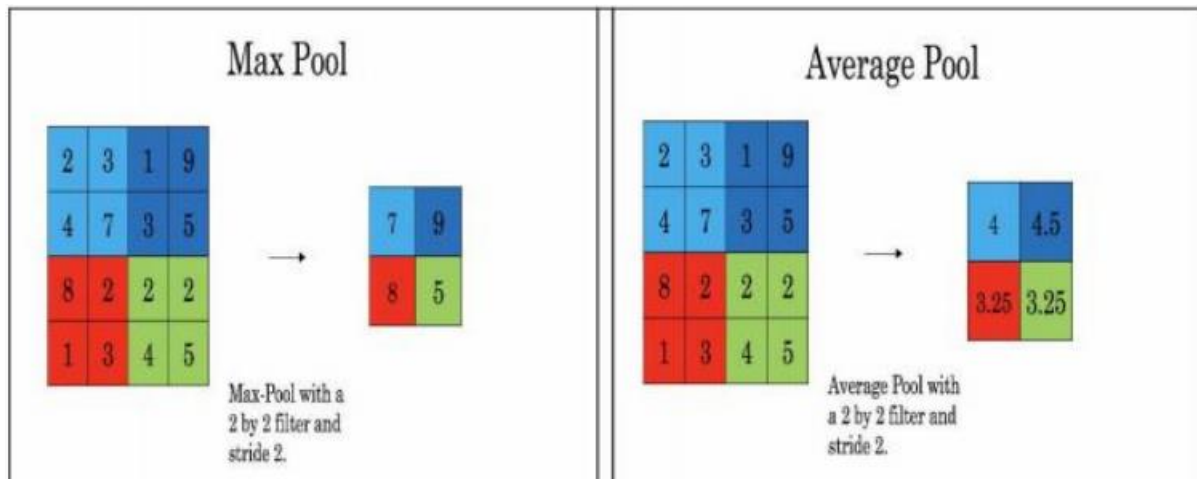


Figure 3.5 Max.Pooling vs Average Pooling [49]

4. Batch Normalization:

Batch Normalization is a technique for feeding any layer in a neural network with zero mean/unit variance inputs. It is applied after the convolution layers to enhance and speed up the training process. This also assists in reducing the internal covariate shift, often rife in most deep networks.[50]

5. Residual Connections:

The residual connections that the researcher finds in deeper networking and profound architectures assist in handling the vanishing gradient problem by enabling gradients to cascade through a shortcut connection that bypasses one or many layers. They are crucial in allowing training in intense networks.[51]

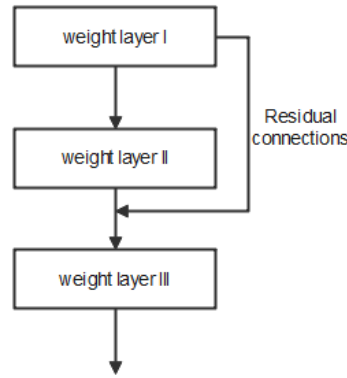


Figure 3.6: Residual Connections

6. Spatial Pyramid Pooling (SPP):

The purpose of the SPP layer is to make the network flexible to object size. It does this by pooling feature maps across scale and location to capture a larger context and conserve the spatial hierarchy of the input images in cases where the object part that is being detected has a varying size.[52]

The SPP is usually a two- or multi-layer structure with interleaved pooling and alternated convolution operations on the framework (by max pooling at different areas of the input feature map, the SPP obtains global contexts from input). The outputs are then cascaded and fed into the following layers together, producing a short vector output that is more robust in terms of feature discriminability.[52]

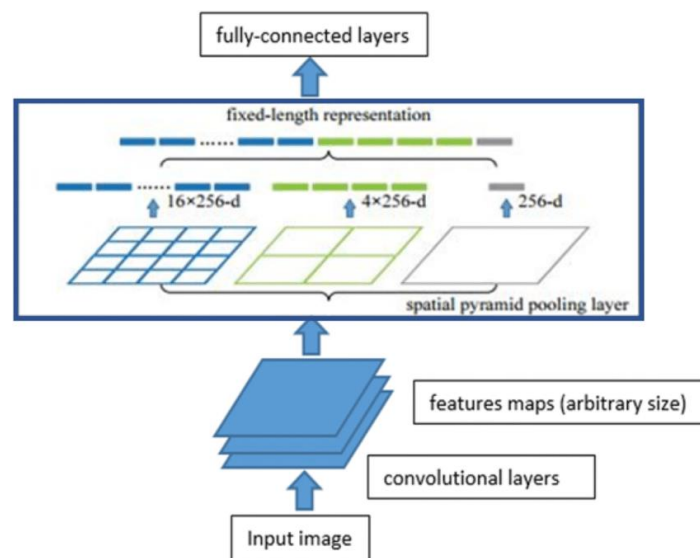


Figure 3.7: Spatial Pyramid Pooling [53]

Neck Architecture

The Neck is a crucial architectural element for further processing feature maps from the backbone and preparing them for the detection head (for detection tasks). Combining features from multiple network layers improves the model's ability to detect objects at different scales.[54]

1. Feature Pyramid Network (FPN):

FPN is part of many object detection models, like the YOLOv9, which improves the model to detect objects on more scales. This is done by creating a pyramid of feature maps at several scales, enabling the feature-extraction layers to handle fine-to-coarse features at corresponding scales and the exact fine-level details.[54]

The FPN takes feature maps from different depths of the backbone, corresponding to varying scales of features. These maps are then gradually up-sampled and combined with the deep features using element-wise addition. This combines the high-level semantic information with the low-level spatial details.[55]

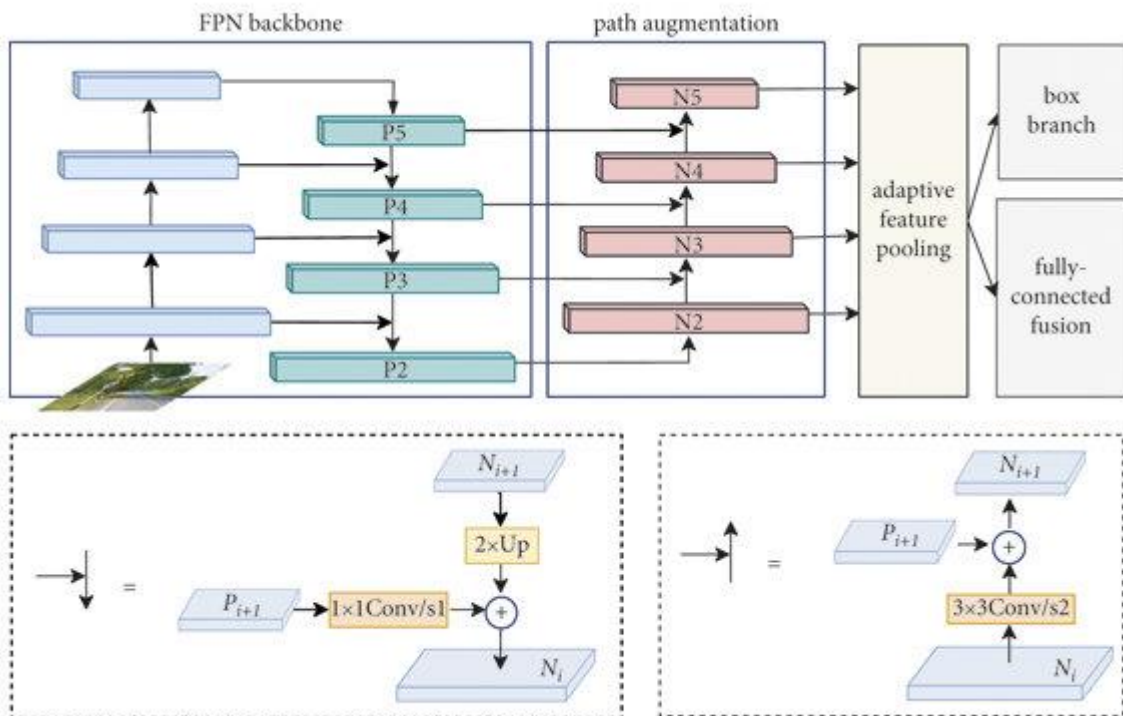


Figure 3.8: Feature Pyramid Network [53]

2. Path Aggregation Network (PAN):

In the present model, adding these extra pathways (in the PAN) while keeping the overall FPN intact enhances the flow of information and allows additional feature combinations to be computed. This structure enhances the localization capabilities for object detection, as it can maintain semantic solid features at all scales of the feature pyramid.[54]

They extend beyond the FPN mechanisms of initial top-down connections and operate over feature maps in an up-sample procedure. This is achieved by upscaling and combining feature maps from coarse to finest levels so that even the high-level (say semantic) features are effectively localized.[55]

Programmable Gradient Information:

An excellent feature in YOLOv9, programmable gradient information, allows us to alter the gradients at runtime during the backpropagation process. This property is used for learning rate changes using a regularization for feedback from real-time performance during the network training structured by the corresponding learning rate. This way, the network can sidestep common problems you may run into, such as exploding/vanishing gradients, and stabilize the training process as the network gets deeper.[56]

Auxiliary Reversible Branch

YOLOv9 also has an Auxiliary Reversible Branch, an advanced architectural feature. It was created to improve the network's detection capabilities, especially for smaller and tricky objects. This side chain to the main PERCEPTIO detection pipeline is optional, providing added compute and feature refinement opportunities.[56]

Purpose of Auxiliary Reversible Branch

The ARB is conveniently embedded in the YOLOv9 model for the following purposes

- **More accurate generalization for small object detection** can capture finer details in the images that may be lost later in deeper layers of the network.
- **Improve feature representation:** It offers an alternate information path to supplement the main detection pathways, resulting in a more profound decomposition of the input data.

- **Improving the robustness:** With better context and more details in the feature maps, the ARB can help the network maintain high performances along multiple scales and case levels of the input.

Structure of the Auxiliary Reversible Branch

1. Parallel Pathway:

- The ARB works parallel to the main detection branches, YOLOv9, and starts an intermediate backbone layer. In doing so, the branch can take advantage of both deep and shallow features of the network, capturing fine-grained textural information and higher-level context concurrently. [56]

2. Reversible Layers:

- The ARB core comprises reversible layers implementing forward and inverse operations with minimal computational overhead. These layers assist in maintaining more info during the network, avoiding the loss of essential details on the forward pass.[57]
- In each reversible stage, a pair of functions process the inputs in two streams. These functions' output is swapped and propagated to the next layer, uniting both refinements of input features without thus permanently transforming back and precisely reconstructing the input from the output.[57]

3. Feature Fusion:

- The processed features are recombined into the main detection pipeline after the ARB. This fusion is orchestrated so that the extra features from the ARB are gently joined and folded into the main feature maps without completely overtaking them.[56]
- Some fusion techniques can include concatenation, addition, or more advanced operations like attention mechanisms designed to emphasize relevant features for detection tasks.

It has the following advantages with the integration of the ARB in YOLOv9:

- More perceptive in identifying tiny and intricate objects thanks to augmented feature extraction abilities.[56]
- Improved gradient flow through the network, as the invertible operations enable gradients to flow more freely throughout the network, mitigating the issue of vanishing gradients.[57]
- Small memory footprint due to the reversible character of the computations (the memory required to compute outputs can be freed up by just keeping the input, as the outputs are a deterministic function from the inputs).[57]

Head Component

A "Head" component of YOLOv9 architecture is where the accurate detection stage is done. This part of the network takes the features generated after being processed by the backbone and neck parts. It uses them in object detection tasks like localizing objects and classifying them in the input images.

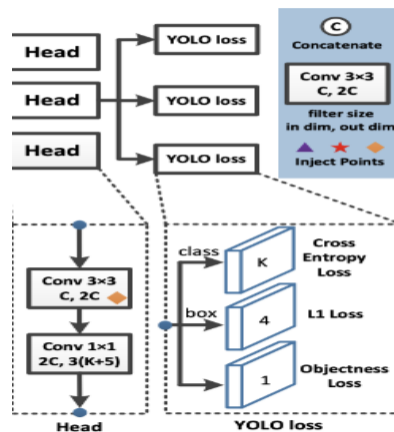


Figure 3.9: Head component of YOLOv9

Structure and Functionality of the YOLOv9 Head

1. Multiple Detection Layers:

- The YOLOv9 head consists of numerous detection layers to detect objects of different scales. This multi-scale methodology is critical for efficiently detecting objects of small to large sizes.

- Each detector layer outputs a set of bounding boxes, the corresponding object class probabilities, and objectness scores. The objectness score tells us about the presence of an object in the bounding box.[7]

2. Convolutional Layers:

- The head uses multiple convolutional layers to reduce the spatial dimensions of the feature maps, further refine the feature representation, and enable accurate object predictions.[7]
- Pooling layers apply specific kernel sizes to keep or diminish spatial measurements. They are critical for catching the spatial chain of importance of highlights required to recognize objects.

3. Anchor Boxes:

- Pretrained anchor boxes: This is a prior-defined box that the network can adjust to fit the size of the objects in the dataset.
- As the network is being trained, it learns how to predict the needed adjustments of these anchor boxes to fit the actual objects best.[7]

4. Bounding Box Regression:

- The head features some components for bounding box regression to refine the sizes of the anchor boxes to match the object boundaries exactly.[7]
- In detail, this process predicts four parameter-based corrections: position of bounding box centers, bounding box width, and height for the predicted bounding boxes to the anchor boxes.

5. Classification Layer:

- The head also does classification (it assigns the detected object to one of the learned probabilities) along with bounding box regression.[7]
- The softmax function is a crucial aspect that requires further investigation. It covers the number of classes and provides a probability of class membership.

Pros of YOLOv9 Head Architecture

- **Efficiency and Speed:** The head design's simplicity allows the head to process rapidly, an essential aspect of real-time detection applications.
- **Offset:** By training the head in detail, the head learns to apply acceptable corrections to anchor boxes and thus locate objects more accurately, increasing detection accuracy.
- **YOLOv9 can detect objects at multiple scales,** meaning that it can handle different object sizes (think tiny objects in images and videos), which is ideal when researcher data has objects on all scales, so you do not have to eliminate any part of the input data based on known object size.[7]

3.7.2 FEATURE EXTRACTING AND DETECTION FOR RT-DETR

Detection Transformer (DETR) is a radically different approach to solving object detection, which uses a transformer instead of the traditional fixed-shape backbone head architecture.[14] DETR relies on Transformers,[58] An architecture developed by researchers at Facebook AI to streamline object detection by removing the need for components as complex as anchor generation and non-max suppression that are otherwise widely recognized in classical methods. In this case, the transformer model speeds up the overall procedure by modeling the image as a collection of objects seen in one go.[14]

Transformers have become the new standard in NLP, as they handle sequences and long-range dependencies well.[58] By coupling these capabilities, DETR can reason about spatial relationships between objects in the image, parallelizing across the image and decoupling the image-wide context from the prediction. DETR re-phrases object detection as a direct set prediction problem by end-to-end adapting the transformer architecture (initially designed for sequences). This not only simplifies the task of detection but also increases the generalization of the model from the training data to the real world, resulting in enhanced detection performance.[14]

This release of DETR is a significant leap for object detection, demonstrating the potential, flexibility, and efficacy of transformer models outside text-based domains.[14] This is part of a more prominent theme I've noticed in AI research. How methods initially developed for one

kind of data are now employed for other unrelated applications, showing the utility and generalization of deep learning frameworks.[59]

The DETR model is applied to utilize its transformer-based architecture, which facilitates precise object detection without the need for the traditional region proposal networks. Its end-to-end training approach simplifies the learning pipeline and improves the detection of small objects such as malaria parasites, ensuring that each detected instance is classified with high reliability.

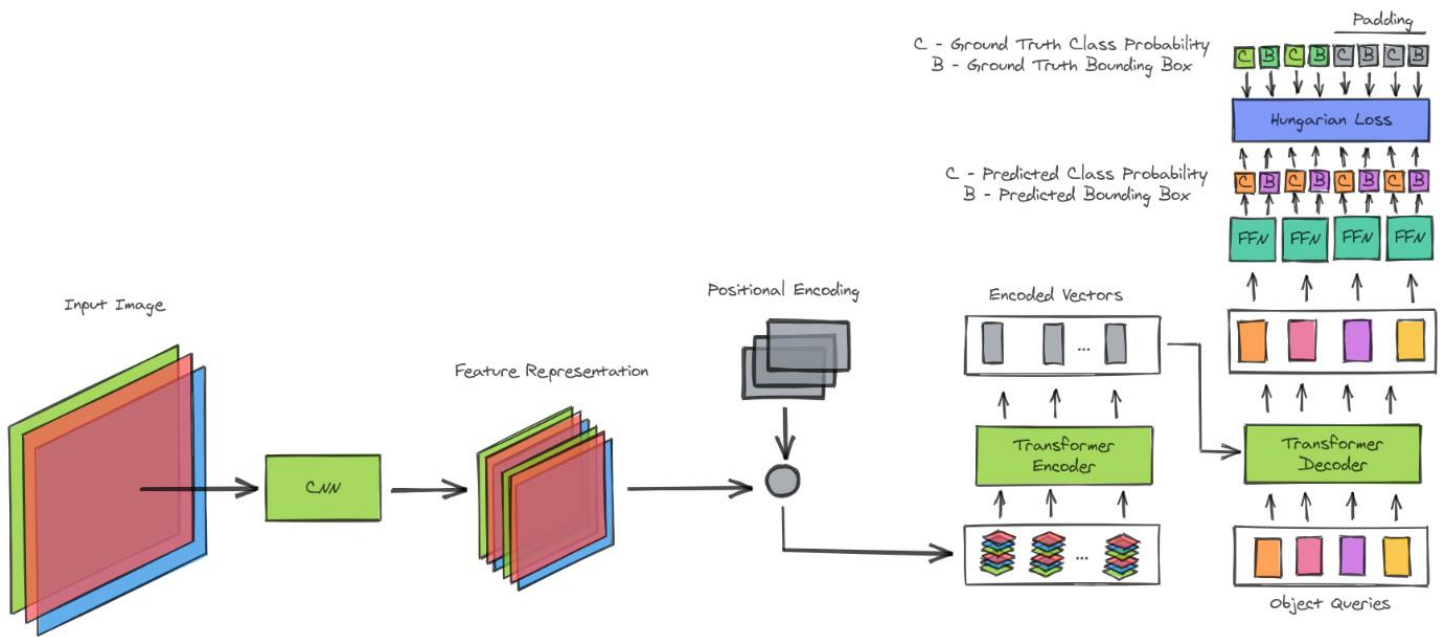


Figure 3.10: Detection Transformer Architecture

Backbone Network

DETR's backbone network is crucial in converting raw input images to flattened feature maps.[14] This is mainly made up of a Convolutional Neural Network (CNN) that is incredibly fine-tuned to capture hierarchical visual features from the input images. Since the detection pipeline in the DETR pipeline is initialized with a CNN backbone, the CNN generates the preprocessing of the input passed as input in the subsequent transformer module.[14]

Low-Level Features: A CNN tends to learn low-level features like edges, text, colors, etc., during the network's early layers.[42] These basic-ability features recognize elements such as simple edges and contours in an image. The convolutional layers operate over small receptive fields to capture exact local details and ignore the global structure.[42]

For example, images start as general visualizations and high-level features like edges, shapes, and so on are captured with depth inside the network. Such features are necessary to distinguish objects in different scenes and complex textures. Deeper layers have larger receptive fields, which can integrate information over larger image patches and capture high-level semantic information.[42]

The CNN backbone enables DETR to effectively work across various visual environments as the image surface processing is structured. The CNN baseline converts the raw data, resulting in a complex feature map set optimized as input to the transformer module to improve object detection efficiency and precision.[14]

The fact that DETR chooses to design CNNs as its backbone is also an emphatic affirmation of the power of convolutional architectures in modeling spatial hierarchies in visual data: CNNs are not just excellent image classification tasks; they also bring a significant weight into more complex detection tasks.[60]

Introduction to Transformers in DETR

A transformer is a particular type of neural network layer, and the Detection Transformer (DETR) [14] adopts the transformer architecture from natural language processing (NLP)[58] To object detection. The integration differs significantly from traditional detection systems with region proposal networks and post-processing steps with Non-Maximum Suppression (NMS).[14]

The transformer model in DETR is a powerful model that operates on visual data (images) and borrows from the transformer's encoder-decoder architecture, originally developed to address sequence-to-sequence prediction problems.[58] This structure allows the model to make global context-aware decisions about the image, which is extremely important for better object detection.

- **Encoder:** The encoder in DETR's transformer inputs the feature maps from the CNN backbone. These feature maps are flattened and incorporated with positional encodings to save spatial information.[14] The encoder has multiple self-attention layers to "read" the entire image simultaneously. This global view allows the encoder to model the long-range complex dependencies among various image parts.[58]

- **Decoder:** The decoder in DETR predicts the final set of objects and their bounding boxes. These are learned embeddings over object queries fed to the decoder one by one, and the decoder refines them gradually through its layers.[14] The decoder is one of the layers. It is responsible for following up the output from the encoder and the previous layer to combine information and result in the properties.

The transformer helps us to predict the set of objects directly and removes the need for heuristic methods like NMS (non-maximum suppression) in DETR.[14] This simplifies the detection pipeline, making the model more end-to-end and less dependent on hand-crafted parts, which is a cleaner and perhaps more moral solution.

Feature Processing by Transformer in DETR

The Detection Transformer (DETR) model uses the transformer architecture to handle features extracted from the input images. The architecture is divided into two main components: the encoder, where the feature maps are refined using self-attention mechanisms, and the decoder, which generates object predictions with the encoder outputs.[14]

Encoder's Role:

1. **Self-Attention Blocks:** The encoder performs self-attention blocks over the feature maps generated by the CNN Backbone.[14] The encoder now has a stack of N self-attention layers. Each self-attention layer looks at every feature simultaneously, thereby encoding the (interdependencies) information between different parts of the image.[58] This applies an attention mechanism, calculating attention scores over all pairs of features in the map to learn the relationship between each feature and every other feature across the image, capturing fine-grained details and contextual information.[58]
2. **Except for Global Contextual Understanding:** By taking all the features jointly through different self-attention layers, the encoder can have an overview of the complete image. This long-range view of context is essential for reasoning about complex scenes where local and global contexts play a crucial role in object identity and localization.[14] By understanding the context globally, predictions are no longer made in isolation but based on how all the features in a scene combine.

Decoder's Role:

1. **Cross-Attention Mechanisms:** In DETR, the decoder cross-attends over object queries (decoding) from image encoder forest object detection outputs (encoding). These object queries are generic embeddings at the beginning and are updated sequentially in each decoder layer.[14] Cross-attention achieves location-sensitive responses across the feature map for every query.
2. **Interact with the Encoder Outputs:** Layers in the decoder interact with the encoder outputs using the updated object queries to extract the information per the feature's location in the global feature map. This is accomplished through the cross-attention modules that connect the queries with the correct spatial locations in the feature map, translating the global understanding into detailed object predictions. Each query predicts a box and class of a different object of interest, and this is guaranteed by the fact that with this set-based global loss, the researcher has a one-to-one matching between predictions and ground truth objects.[14]

DETR is a powerful model for object detection. It can process features globally to make predictions and refine objects using the attention mechanism in the transformer. This rule saves from heavyweight post-processing steps traditionally used and makes the detection model's training end-to-end.[14]

Advantages of using Transformers for Feature Extraction in Object Detection

The Transformers, hailed for their breakthroughs in natural language processing, have also made the mark in object detection because of their unique features, primarily in feature generation. The Detection Transformer (DETR), which was introduced, shows an example where transformer architecture can be combined to improve feature extraction and where the learned features reflect their learnability and effectiveness concerning typical image patterns far from the olfactory patterns of a generic CNN built on top of convolutions.

Contextual Global info

1. **View of Image Features:** Transformers can access each part of the image through self-attention at the token level. By seeing an entire image, the model could learn global

relationships or dependencies between parts of it, making it easier to detect objects in context with their environment.

2. **Better Feature Representation:** While convolutional networks process information locally, transformers aggregate information from the entire image and learn better feature maps. This ability benefits small objects or objects with intricate interrelations with other objects in the scene.

End-to-End Differentiability:

1. **Differentiable Everything:** Transformers in DETR make the whole process (i.e., extracting features, computing queries, computing keys, performing attention operations, etc.) differentiable end-to-end trainable. This invariance means the researcher can backpropagate through the whole model, simplifying the training loop by cutting out the interstitial tasks or complex pipelines like RPN and NMS.
2. **Unified Model Architecture:** DETR with transformers uses a single, unified architecture without fragmented components, making the object detection system less complex. This simplification increases the interpretability of the model's operation and makes deployment and maintenance easier.

More accurate and efficient:

1. **Partial Heuristics and Post-processing:** Traditional object detection systems, on the other hand, frequently use heuristic methods to adjust the detection boxes, such as anchor box assignment and non-max suppression. By having transformers learn to predict boxes without the bounding box regression steps, fewer boxes are detected, resulting in cleaner and more accurate detections through set-based loss and bipartite matching.
2. **Scalability to complex scenes:** transformers are incredibly efficient at processing high amounts of interactions concurrently (writing $\sim N^2$), which is why they become efficient in complex scenes with many interacting objects-one-to-one or subtle event interactions are not their domain. Because self-attention is scalable, DETR can keep up with the increasing complexity of the scene without negatively affecting the performance

In conclusion, adding transformers in DETR offers a robust feature extraction approach that improves the model's capacity to decipher visual data. Besides enhancing the quality of the object detection models, this integration also dramatically improves the processes involved in training and performing inferences in such models. The advantages of this technique are well-described in different papers and benchmarks of object detection, which carry out the role of transformers in this regard.

3.7.3 FEATURE EXTRACTING FOR CLASSIFICATION

In this research, the researcher examined several convolutional neural network (CNN) architectures and the Vision Transformer (ViT), all of which provide a way to extract features from image data. Models like Mobile-ViT, Xception, InceptionV3, EfficientNetB0, DenseNet121, NASNetMobile, MobileNet V2, VGG16, ResNet50 and Vision transformer (ViT). Two variants of each model are selected to examine feature extraction abilities concerning different architectural designs:

Pretrained CNNs harness the power of transfer learning, where models developed on extensive datasets are fine-tuned for our specific task. This methodology significantly enhances the model's ability to recognize complex patterns in malaria-infected blood smears, thus improving the overall accuracy of the diagnosis.

The Vision Transformer is employed to capitalize on its ability to focus on critical parts of an image using a self-attention mechanism. This feature is particularly useful for detailed analysis of blood smears where the distinction between different types of malaria parasites requires examination of fine-grained visual details.

1. **Mobile-ViT:** A symbiosis of CNNs and Vision Transformers to perform efficient image processing.[61] It capitalizes on the spatial hierarchies learned by the convolutional layers and the long-range interactions across the image provided by the transformer layers, making it well-equipped for tasks that require reasoning over complex visual data with diverse textures and structures.[61]

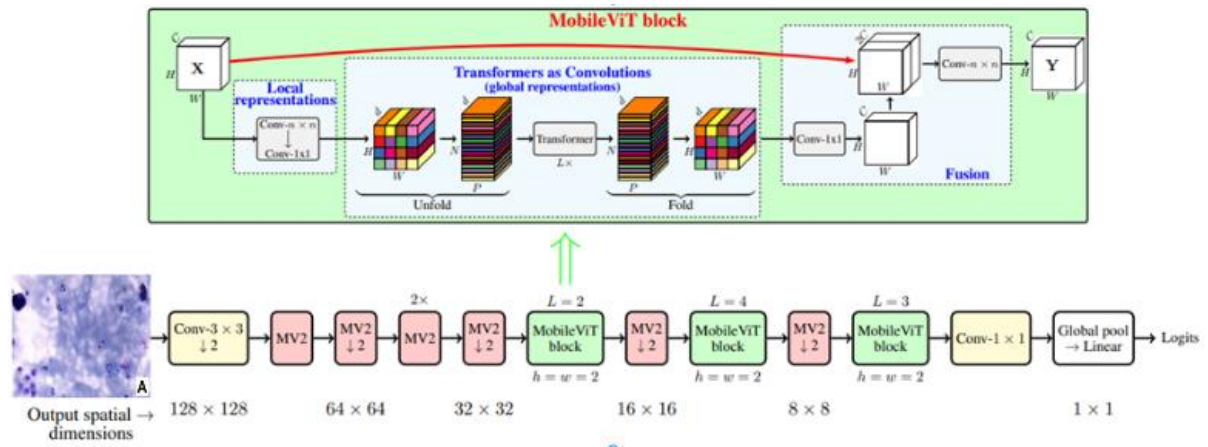


Figure 3.11 Mobile ViT Architecture [62]

Architecture: MobileViT combines the spatial processing capabilities of CNNs with the global contextual understanding of transformers. It is structured around blocks of MobileNet-style convolutions followed by transformer blocks that treat image patches as sequences for global information aggregation.[61]

Feature Extraction: MobileViT combines the spatial processing features of CNNs with the global contextual reasoning of transformers. It comprises MobileNet-style mobile convolution blocks followed by transformer blocks that operate on image patches as sequences while enabling interactions between the patches to incorporate global information.[61]

The combination of convolutional blocks and transformer blocks, on the other hand, enables MobileViT to keep its computational cost low while taking advantage of the large receptive field associated with the self-attention mechanism.

MobileViT takes the input image and runs it through a series of MobileNet-style convolutions that capture spatial features. Then, it applies a series of transformer blocks, treating the output of these convolutions as being sequenced. Convolution layers produce feature maps representing local visual patterns like textures, edges, etc.[61] The second wave of transformer layers up-sample these maps to capture global context; this combination of local and international image representations is carried through all later stages of the U-Net.

2. **Xception:** Internally, it uses depth-wise separable convolutions, which makes it easier to capture new patterns with fewer parameters.[63] An architecture that shines when it

comes to representing properties of object classes in datasets has considerable within-class variation.[63]

Architecture: Xception, short for "Extreme Inception," is a deeper version of the Inception model. All layers are replaced by Depth-wise Separable Convolution, which allows for computing channels and spatial separately.[63]

Feature Extraction: Since Xception utilizes depth-wise separable convolutions for its depth-based architecture, (it can more readily disentangle the joint dependencies between spatial and channel-wise relationships, leading it to identify better-detailed structures in lower layers and more complex context in deeper layers).[63]

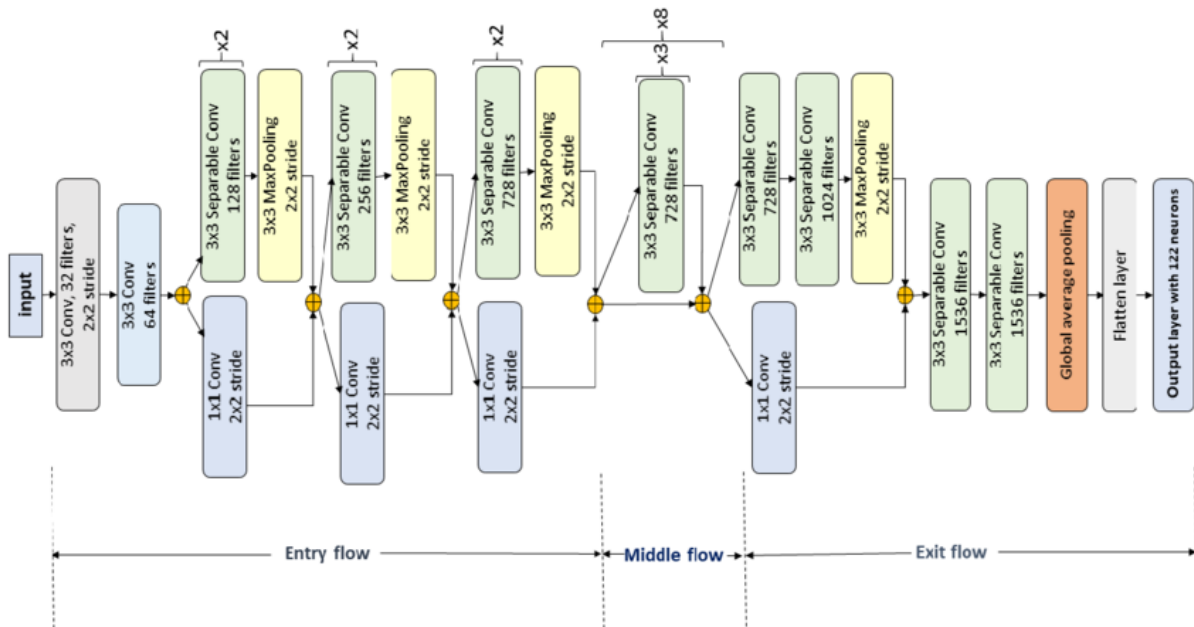


Figure 3.12: Xception Architecture [64]

The use of depth-wise separable convolutions, which decreased the complexity and operating costs of the model, substantially increased efficiency without compromising its functioning. The architecture is magnificent in high-resolution image processing with relatively fewer parameters.

The depth-wise separable convolution in Xception independently performs convolution operations on both spatial and channel-wise features.[63] This method permits the network's independence of spatial dimensions and feature channels. Depth-wise separable convolutions generate dense feature maps and are thus parameter—and

computation-efficient. The concatenated maps can capture features fine-granularly to learn features in different layers.[63]

3. **InceptionV3**: its architecture consists of multiple modules with parallel convolutions of separate sizes.[65] This will let the model learn information at different scales and levels of complexity, which will help it find and classify objects in an image correctly.[66]

Architecture: InceptionV3 improves upon previous generations by adding modules with multi-level parallel convolutional filters. It enables the network to select one of multiple scales of feature processing at each layer.[62]

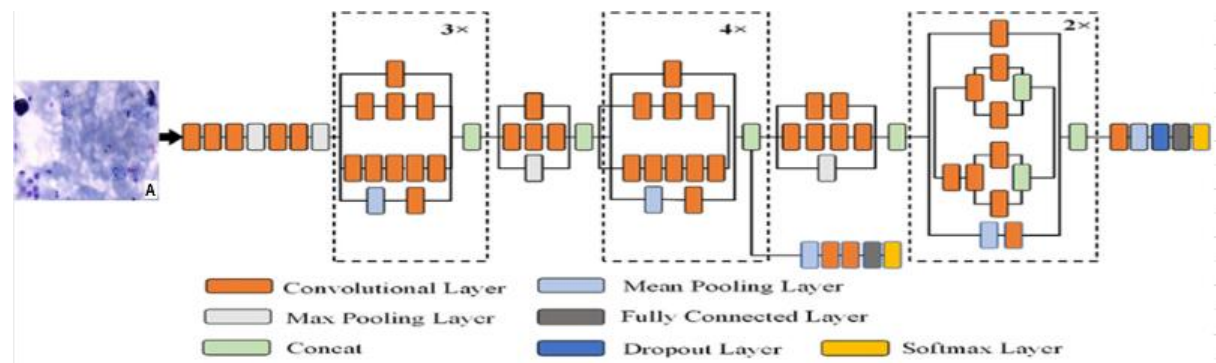


Figure 3.13: Inceptions Architecture [67]

Feature Extraction: Low-level features are processed in earlier layers with smaller convolutions, while higher-level features are captured in deeper layers through more extensive convolutions combined across the module outputs.[65]

InceptionV3 uses factorized convolutions to decrease network connections and parameters, decreasing feature network limitations at diverse scales while enhancing throughput.[65]

The InceptionV3 architecture uses parallel convolutional filters of different sizes to deal with multi-scale information simultaneously. The architecture generates various feature maps with unique resolutions and complexities at each module. The multi-scale processing allows us to capture multiple spatial hierarchies and object details.[65]

4. **EfficientNetB0** uses a compound scaling method that uniformly scales all depth, width, and resolution dimensions with fixed scaling coefficients.[68] Such balanced scaling

enhances the network's efficiency and capacity for extracting and processing features at different levels across the image.[68]

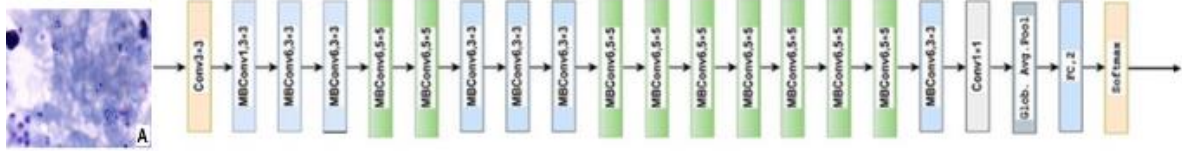


Figure 3.14: EfficientNetB0 Architecture [69]

Architecture: EfficientNetB0 uses a systematic scaling method that uniformly scales the dimensions of depth, width, and resolution with a balance between these dimensions, which has been demonstrated to be important in ConvNets.[68]

Feature Extraction: The model learns low-level features like texture and color in the earlier layers and progressively generalizes to object parts at higher layers.[70] This is achieved through the increased depth and resolution of the network, facilitated by the compound scaling method.[68] EfficientNetB0 dynamically adjusts its complexity to balance efficiency and accuracy, making it a versatile architecture for various image recognition tasks.[70]

EfficientNetB0 uses convolutional layers to extract features from the input images incrementally.[68] The compound scaling method ensures that feature maps grow more prosperous and detailed at each stage, with increased depth and resolution enhancing the network's ability to capture granular and high-level image features.[68]

5. **DenseNet121:** The network pattern described here is called dense, and it has dense connections from every layer to all subsequent layers in a feed-forward type.[71] The connectivity pattern ensures that the maximum amount of information between layers is retained so that higher layers can extract more refined features without losing essential details.[71]

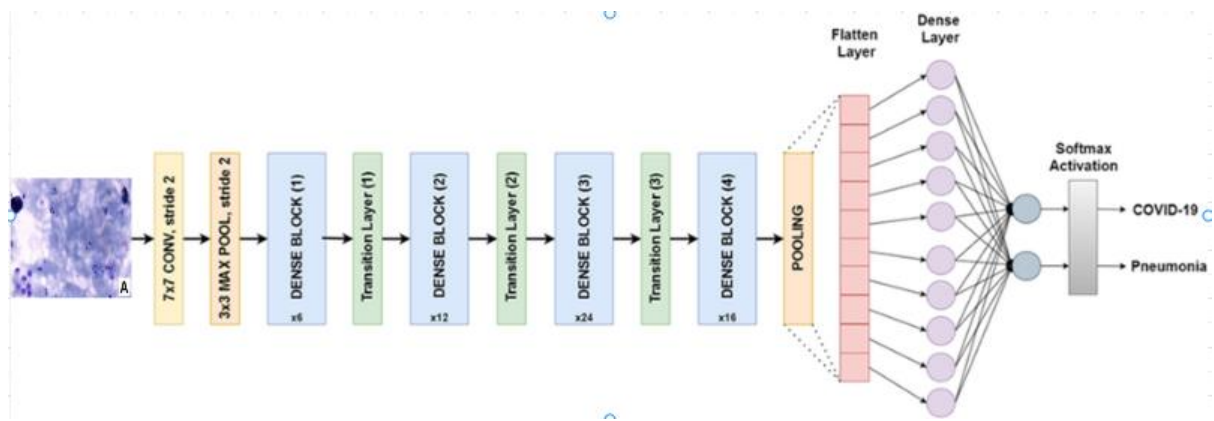


Figure 3.15: DenseNet121 Architecture [72]

Architecture: Instead, DenseNet121 introduces a mechanism to enforce this: dense connectivity patterns in which every layer receives inputs from all layers before it. This helps convolutional features propagate through the network and reuse features.[71]

Feature Extraction: This connectivity enables deep layers to access low-level feature information directly, thereby enriching the network with opportunities to integrate simple and complex features.[71] The dense connections also help to alleviate the vanishing gradient problem, permitting the network to be deeper and more efficient without a dramatic increase in parameters

DenseNet121 utilizes dense connectivity (each layer gets input from all prior layers, merges these, and then processes them further).[71] This causes each feature map to contain a lot of information as they combine the low-level details of the low layers with the high-level features from the high layers, increasing the reuse of features and their propagation across the network.[73]

6. **NASNetMobile:** Using Neural Architecture Search (NAS), a mobile-augmented architecture can be designed specifically for performance and computational efficiency in resource-constrained environments like edge computing.[74]

Architecture: NASNetMobile's architecture is derived through neural architecture search, which explores a vast search space of possible model configurations to identify those that maximize performance while adhering to computational constraints.[74] The resulting architecture is tailored for mobile devices. It combines separable convolutions and reduction blocks that downsample feature maps while extracting and refining relevant

information.[75] This lightweight yet robust design enables efficient inference of diverse image content on mobile platforms.

Feature Extraction: using separable convolutions and reduction blocks helps reduce spatial size while extracting and refining features. Designed to run on mobile devices, the architecture is lightweight yet capable of inferring different image contents in a performant manner.[74]

NASNetMobile is a mixture of reduction cells that downsample (reduce the spatial dimensions) and normal cells operating at a particular dimension. The encoder has reduction and normal cells where only the normal cells connect. This alternating structure gives the output the efficiency and diversity paramount for using feature maps in mobile deployments with computational constraints.[67]

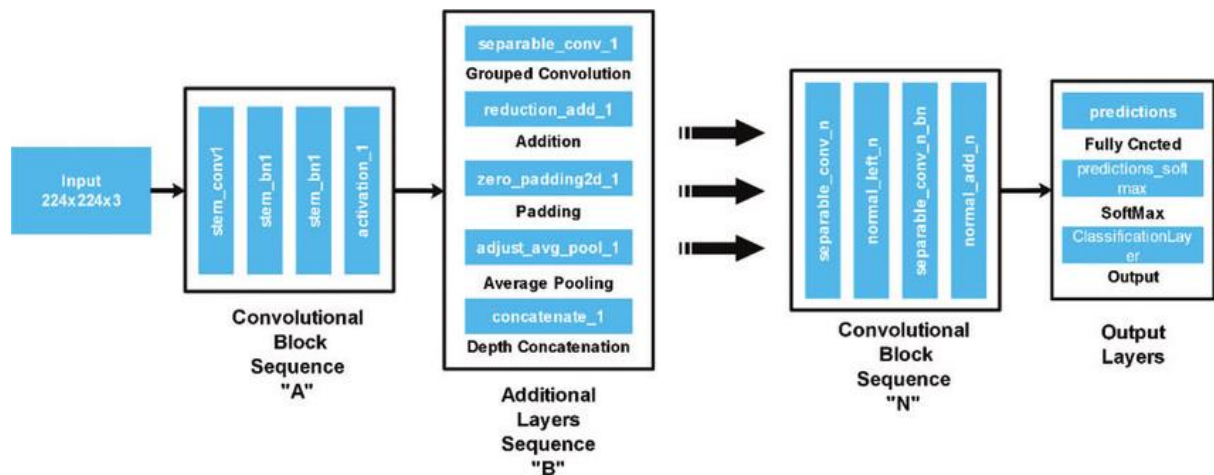


Figure 3.16: NASNet Mobile Architecture [76]

7. **MobileNet V2:** It uses inverted residual blocks with linear bottlenecks to efficiently capture the essence of feature maps. This model is optimized for mobile and edge devices, balancing model accuracy and inference time (speed).[77]

Architecture: MobileNet V2 utilizes an inverted residual structure where the input is first expanded into a high-dimensional space using a 1x1 convolution.[77] This is followed by a depth-wise convolution (3x3) to filter spatial information and a final 1x1 convolution to research back to a lower dimension.[77] Linear bottlenecks compress the representation between these layers, reducing computational costs. [77]. Shortcut connections between bottlenecks further enhance gradient flow and information propagation.[77]

Feature Extraction: Our model can efficiently extract spatial features using convolutional neural networks at a meager computational cost using lightweight depthwise convolutions, ensuring features at each layer are used to predict the target with precision.[77]

The MobileNet V2 is cleverly designed so information can flow through the system. This means it can keep track of irrelevant parts of data even as it analyzes data, leading to finding exciting patterns (often understanding the raw complexity of the data) without slowing it down.[78]

It's more of a detective trying to solve a mystery. Initially, the researcher noticed some obvious clues like shapes and colors. As it gets deeper, it starts correlating the dots to identify the more complicated patterns.[70] It does this effectively by performing a neat trick that scrutinizes the necessary stuff without overly bothering with the irrelevant math.[77]

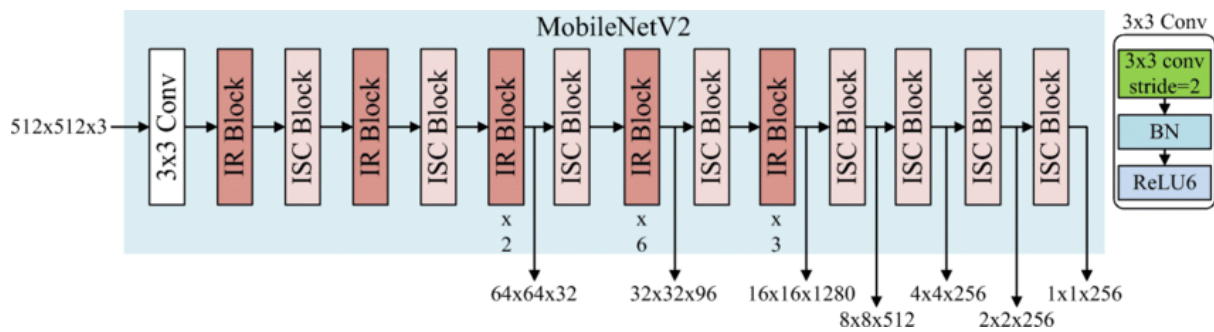


Figure 3.17: MobileNetV2 Architecture [79]

8. **VGG16:** A straightforward and deep conventional network. The repeated stacking up of convolution layers with small receptive fields captures the 'ingredients' of images well. Naturally, this leads to direct applications demanding powerful feature extraction.[80]

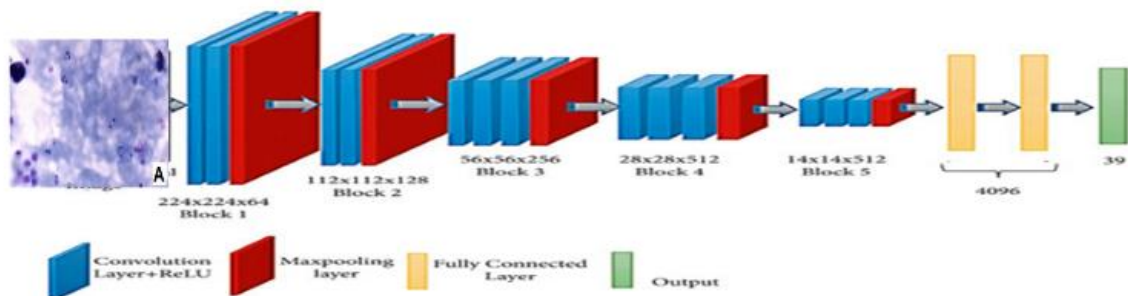


Figure 3.18: VGG16 Architecture [81]

Architecture: VGG16 features a straightforward architecture consisting of multiple convolutional layers with small (3x3) receptive fields followed by max-pooling layers [80]. This repetitive pattern creates a deep network that progressively extracts higher-level features from the input image. [82]

Feature Extraction: use a set of 3x3 convolutions followed by max-pooling layers for feature extraction up to a certain depth in the model. While simple conceptually, the depth and uniformity of VGG16 provide a robust and flexible feature extraction model that shows perfect transfer learning performance across many visual detection and identification tasks.[80]

For example, in the case of VGG16, this consists of several convolution layers back-to-back followed by max pooling. The fixed architecture leads to a deep stack of feature maps, where each layer captures higher-level abstractions. This simple chaining leads to complete feature extraction across the network.

9. **ResNet50:** It introduces long-range residual learning, capable of training up to intense layer networks.[51] ResNet50 can learn its robust feature representations by using skip connections, which enable the learner to jump over some layers, rectifying the vanishing gradients problem.[73]

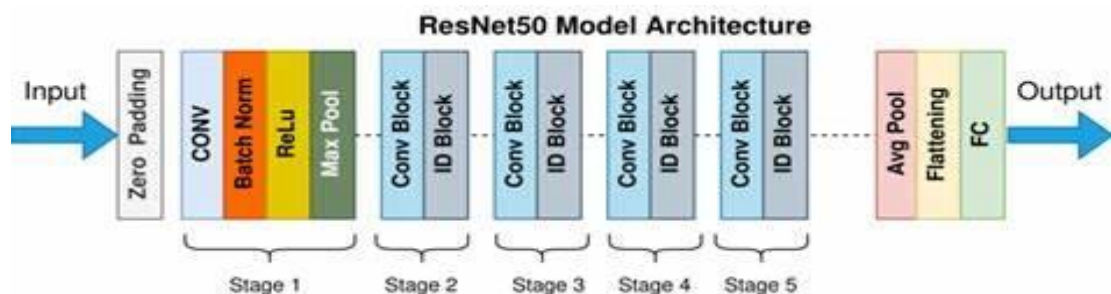


Figure 3.19: ResNet50 Architecture [83]

Residual Block: ResNet50 -where residual learning is employed-[51], introduces shortcut connections permitting direct gradient flows through the network. Hence, this approach helps to train deeper networks.[73]

Feature Extraction: The features are made gradually finer through the network, with residual connections to maintain the quality of information across the input and output.[51]

Skip connections are another critical milestone in the introduction of Resnet. They allow considerable network depth while vanishing gradients and enable substantial depth increase without degradation.[73]

Whereas ResNet50 introduces skip connections, convolutional layers can learn residual functions concerning the layer inputs.[51] These residual connections in ResNet50 improve the quality and depth of feature representation, allowing the network to learn intricate features without performance degradation.

10. Vision Transformer (ViT): This relatively new development applies the transformer mechanism (popular with language processing tasks) to image classification.[84] ViT divides plaintexts of images into patches and then outputs them as sequences, allowing the model to attend to long-range dependencies across the image, at least in the absence of the call to an oracle to select a predetermined number of tokens.[84]

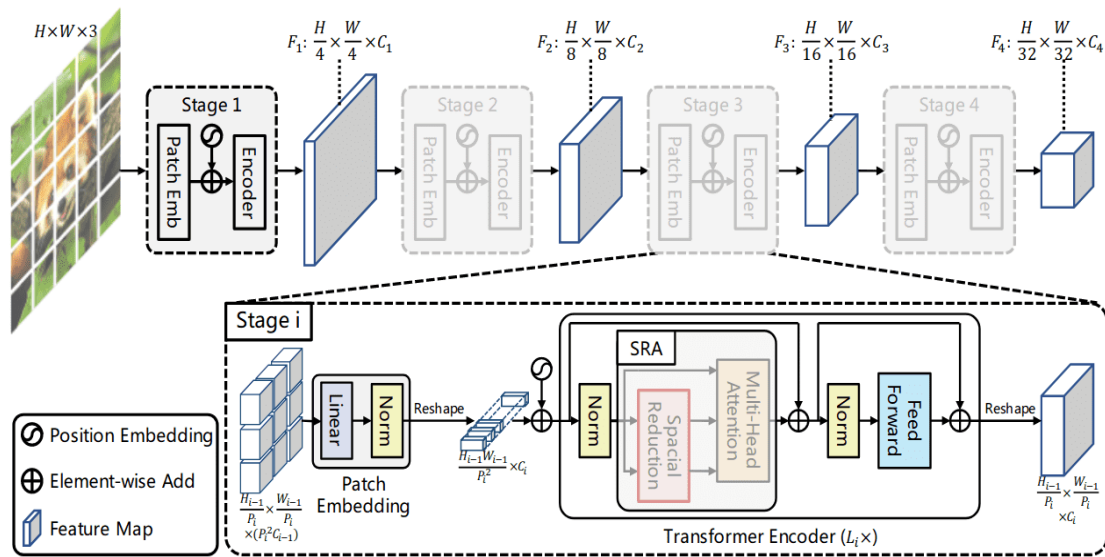


Figure 3.20: Vision Transform Architecture [85]

Architecture: With a vision for the transformer, vision transformer (ViT) uses a transformer architecture for image processing, breaking the image into patches regarded as tokens and passing them through a series of self-attention and feed-forward layers.[84]

Therefore, capturing long-range dependencies is crucial for the performance of different features and objects, which means feature extraction is the heart and soul of ViT.[84] To extract features, ViT uses a transformer to capture intricate relationships among distant

regions of an image and fuse these interactions across the entire image formed to increase the symbolic power of features and objects.[84]

Transformers provided a way to use attention over all elements of an input sequence in a single pass. While initially designed for language processing, bidirectional attention was promising for computer vision because attention can now reach long distances in the input.

So far, the Vision Transformer (ViT) method has been to treat individual image patches as tokens and run them through the traditional transformer blocks and self-attention mechanisms.[58] Since ViT generates feature maps that know the entire image, it can get a global view of researcher input data. These maps can attend to the entire input sequence using self-attention weights based on whole connectivity patterns and not simply local invariant patterns within the data.[84]

Any of these models might offer a unique solution to feature extraction, which is crucial for detecting and classifying a wide array of complex objects within images. This study seeks to evaluate and compare the efficacy and efficiency of these architectures on real-world imaging tasks, exploiting their different strengths in analyzing their traceability and use case scenarios.

3.8 FINE TUNING

The model weights need to be fine-tuned further to fit with the observations. In contrast to feature extraction, where the researcher removes the last fully connected layer and uses the output from the unit before that as the combinatoric (i.e., non-linear) function, fine-tuning is a much more extensive operation. The researcher transfers learning by modifying the CNN's architecture and object class output and re-training the modified CNN to recognize additional object classes.[86]

Several necessary steps take place in the fine-tuning process:[87]

- Remove the fully connected layers: The last layers of the neural network produce class-label predictions from the extracted features.
- This is done by replacing these layers with new, freshly initialized, fully connected layers.
- Freezing the initial convolutional (CONV) layers to retain the robust features learned by the network earlier on.

- Introducing training for the new fully connected layers
- An optional second phase of training aimed at adjusting deeper features by unfreezing a few of the earlier CONV layers

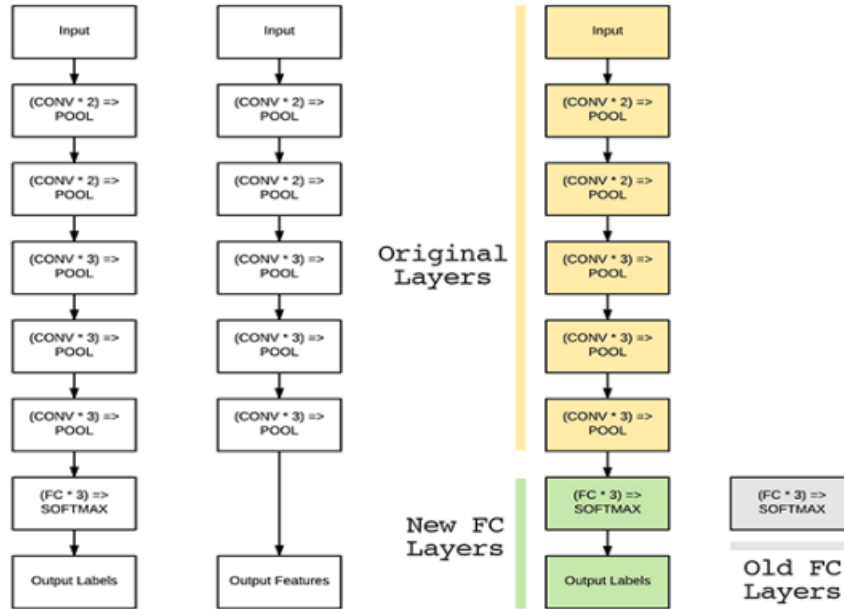


Figure 3.21 Left: The original VGG-16 Network Architecture. Middle: Removing the fully connected Layers. Right: Removing the Original Fully Connected Layers and Replacing Them with a Brand New Fully Connected [88]

Now, let's take an example of the VGG-16 network architecture. At first, the network has weights only for the fully connected layers at the end and nothing here for attention, referred to as the "head," and a softmax classifier. This head is removed during fine-tuning (as in feature extraction). Instead, a new, fully connected head is crafted and stacked on top of the former network architecture. This new head is then randomly initialized and connected to the frozen base of the network.[89]

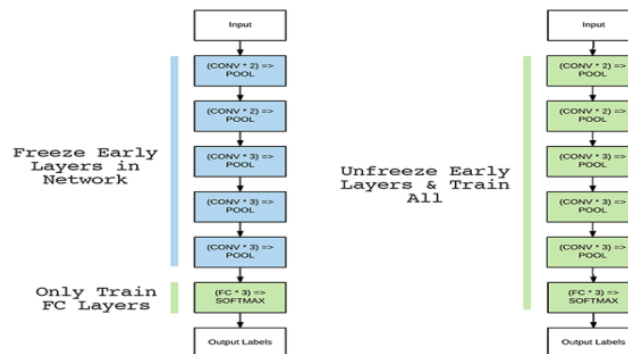


Figure 3.22: Sample Understanding of Fine-tuning Using VGG-16 [88]

During the initial training, it is common practice to freeze all CONV layers and to allow only the new, randomly initialized, fully connected layers to adjust to the dataset. This "warm-up" period helps the network adapt to patterns in the data without changing the CONV layers at the beginning. Once the results are stabilized for the additional classification, and the dataset requires it, some of the early layers tend to be unfrozen and get tuned, but through minimizing the architectural adjustment to the original convolutional filters so that they are not altered too far from their optimizable states.[90]

You train until the model performs well or until it attains acceptable accuracy. This is a compelling method for fine-tuning pre-trained CNNs on new datasets, allowing us to quickly get excellent results on a relatively small data set.

At the end of something like this, you should be able to:

- Fine-tune networks using frameworks like Keras.
- Use the fine-tuned model to make predictions in its upgraded state

3.9 MODEL TRAINING

Model training is one of the most critical steps in deep learning system development because this is where the design that you (theoretically imagined) is trained. This step includes tuning the model's parameters with the help of data to reduce errors and improve its prediction: Hyperparameter Tuning & The Model of Researcher Training Environment Shed.

3.9.1 HYPERPARAMETER TUNING

They are the settings of the training process; that is, Hyperparameters are the external properties of the network that are not the function of data. Hyperparameters are different from the other parameters and, instead of being directly learned during training processes, are set using heuristics (trial and error approaches). Because its values impact how well it trains the model and how efficiently it is generalized upon new data, tuning this hyperparameter is vital.

Key hyperparameters include:

- **Learning Rate:** This is arguably the most critical hyperparameter as it regulates the step size at each iteration while heading for a minimum of a loss function. A Correct

learning rate helps with fast and stable convergence, while an incorrect learning rate can cause the training process to diverge or stagnate.

- **Number of Epochs:** This hyperparameter defines how often the learning algorithm works through the entire training dataset. Fewer epochs may lead to underfitting and more to overfitting.
- **Batch Size** is the number of training examples used in one iteration. One key observation is that a smaller batch size has a regularization effect, which results in lower generalization error.
- **Regularization Techniques:** Regularization techniques like L1 and L2 regularization are the hyperparameters used to avoid overfitting a model by giving more weights to the larger weights.
- **Dropout Rate:** When training deep neural networks, Dropout Rate is the hyperparameter that controls the probability at which the ‘drop-out’ layer outputs are dropped.

There are three ways to fine-tune these hyperparameters: manual tuning via trial and error, grid search, where all the combinations of the parameters are evaluated, or more sophisticated techniques like random search and Bayesian optimization, which are far more efficient and often result in better performance.

3.9.2 TRAINING ENVIRONMENT

The hardware and software configurations you use to train the models are part of the context of the training environment. These include considerations such as:

- **Hardware Resources:** Hardware resources like graphics processing units (GPUs) or tensor processing units (TPUs) are essential to learning deep learning models. Thus, training a deep learning model is a computationally demanding task. In other words, the giant matrix operations performed during training do not have to be done on the CPU; instead, they can be handled on these hardware resources, making the training process significantly faster for the above reasons.

- **Software and Tools:** The choice of software frameworks influences how the researcher trains, so you should keep that in mind. Popular frameworks such as TensorFlow, PyTorch, and Keras provide flexibility, less coding time, and a vast library, making the different models more accessible and faster.
- **Distributed Training:** To reduce the training time, you can distribute the training on multiple machines/GPUs and use massive datasets or models. Some frameworks, like TensorFlow and PyTorch, support distributed training.
- **Data Accessibility:** It ensures that the training data is accessible in a timely and reliable manner (this holds when handling large datasets). Data must be pre-processed and saved to allow easy retrieval and avoid bottlenecks during training.
- **Environment Stability:** The training environment should be as static and reproducible. This will ensure that the models can be trained reliably over time. It is also essential to maintain uniformity with the versions of software and libraries on which the dependencies in training models are based.

Fine-tuned hyperparameters and a well-calibrated training environment can ensure that the model learns well from training data and can generalize its learning for unseen data. These models are more resistant to adversarial examples, and balance is necessary for creating working deep-learning models for natural products.

3.10 MODEL EVALUATION

Model evaluation employs several measures to comprehend a deep learning model's performance strengths and weaknesses. Evaluating a model's effectiveness in the early stages of research is critical and aids with model monitoring.

3.10.1 VALIDATION AND TESTING

One of the critical evaluation techniques in deep learning is using K-Fold cross-validation. When you have limited data samples, the researcher aims to ensure that a model performs well. It's not biased toward the training set. This method divides the data into k subsets, and a model is trained using k-1 of the folds as training data. This iterative procedure adjusts the model's hyperparameters and gives an unbiased estimate of the model's generalization ability on an independent dataset.

K-Fold cross-validation begins by dividing the entire dataset into smaller k sets. Commonly, the number of folds k is set to 5 or 10, but this can vary depending on the dataset size and the need to balance computational efficiency with evaluation thoroughness. The model is then trained on $k-1$ folds, with the remaining fold used as the test set. This cycle is repeated so that each fold serves as the test set once, allowing every data point to be used for training and validation.

Starting with the entire dataset, K-Fold cross-validation divides it into smaller k sets. Frequently, k is set to 5 or 10, but this can be changed based on the size of the dataset and the need to balance computational efficiency and evaluation thoroughness. The model is then fitted on the $k-1$ folds and tested on the remaining fold. It's a 'multi-holdout' (hence the whole part of the name) - multiple times, series of holdout sets are created, and this cycle is repeated so that every fold is the test at least once - and every data point is in both the training and validation set.

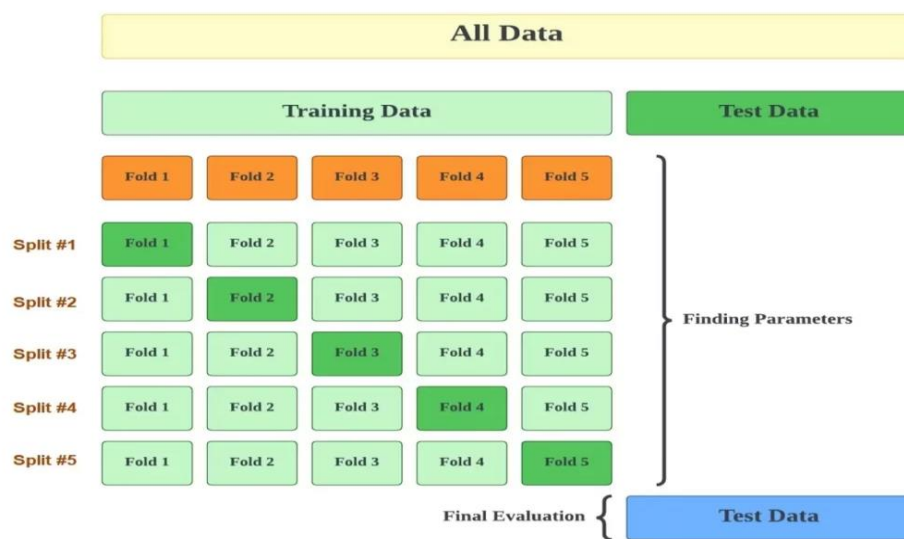


Figure 3.23: K-Fold cross-validation

It is most useful in areas such as healthcare, where higher prediction accuracy is critical. K-Fold Cross Validation can be very useful in games like malaria detection, where data could be limited and noisy. Therefore, it aids in creating a powerfully correct diagnostic classifier. In doing so, researchers aim to ensure their prediction model is sufficiently adapted to the particulars of the training data (thus being able to generalize well to different clinical settings).

These are the formulas to evaluate our model:

$$\text{Accuracy and recognition rate} = \frac{TP+TN}{P+N}$$

Equation 3.9

$$\text{Precision} = \frac{TP}{TP+FP}$$

Equation 3.13

$$\text{Error and Misclassification rate} = \frac{FP+FN}{P+N}$$

Equation 3.10

where.

P = Positive (whether true or false positive)

N = Negative (whether true or false

negative)

$$\text{Sensitivity, TP rate, and Recall} = \frac{TP}{P}$$

Equation 3.11

TP = True Positive

TN = True Negative

FP = False Positive

FN = False Negative

$$\text{Specificity and TN rate} = \frac{TN}{N}$$

Equation 3.12

3.11 ETHICAL CONSIDERATIONS

Ethics are almost always the first thing that comes to mind with medical research and application (and rightfully so), but specifically with deep learning research for disease diagnosis, such as malaria. The most common ethical issues, such as research, raise concerns about the use of data, patient confidentiality, and the accuracy of diagnoses.

1. Data Handling and Confidentiality:

Public health data processing, including patient-specific data, is subject to rigorous ethical and legal privacy requirements. All data from blood smear images collected for training and testing the AI models in this research are anonymized. This implies that data is used pred-de-recognized to remove any data that could be traced back to a person. This ensures patient privacy is maintained and reduces the potential risks of a data breach.

2. Informed Consent:

Also noteworthy is that data used to train models should be sourced under strict informed consent protocols. This means that the data was obtained from informed, consenting patients. This research involves working with healthcare institutions such as the Jimma University Malaria Center to deliver data ethically, informing patients about the study and what it will achieve.

3. Accuracy and Reliability of Diagnosis:

Another aspect of the ethical use of AI in healthcare refers to tools developed for proper and accurate diagnosis. This could cause more harm to a patient through treatment based on a misdiagnosis. Specifically, the research uses highly robust model evaluation and validation methods to help improve the precision and trustworthiness of the malaria detection models. Regular benchmark testing and testing on known data is performed to check the model's diagnostic quality. This ensures their high quality in diagnostics before they get deployed to clinical use.

4. Addressing Bias and Fairness:

AI systems for medical applications run the risk of perpetuating bias, which is one of the worst ethical violations the researcher could imagine. It is essential to ensure that the AI models are not learning or amplifying biases that might exist in the training data. This research addresses this by including many different datasets that reflect a mixture of malaria manifestations and individuals across various demographics. Further, the models are adjusted and retrained continuously, with the expansion into monitoring for bias in model predictions.

5. Transparency and Accountability:

Keeping patients in question how the AI model came up with the decisions and validating the result (just or unjust) to remain an epicenter of Ethical AI under the hood of the healthcare industry. It ensures that all the algorithms and their decisions are well-documented and readily comprehensible to a healthcare provider. This transparency is beneficial in generating trust among end users and enables responsible practices when the technology fails or does not work as expected.

Engaging thoroughly with these ethical issues ensures that the research meets high moral standards and can improve the acceptability and utility of AI diagnostics, potentially benefiting patient care and outcomes. Thus, it contributes to the goal of better health in the context of malaria detection.

3.12 SYSTEM ARCHITECTURE

This research proposes a state-of-the-art, two-stage malaria detection method that utilizes powerful deep-learning technology to address thick and thin blood smear examinations. This system is intended to fulfill the severe needs of malaria detection in places with inadequate healthcare infrastructure, concentrating on obtaining high diagnostic accuracy and operating efficiency.

Stage One: Detection in Thick Blood Smears - The first step of the diagnosis involves drawing at least one blood sample, which is essential in guaranteeing the adequacy and relevance of the smear images procured. A flexible imaging arrangement is deployed, with a smartphone or other suitable imaging device positioned behind a microscope eyepiece. This unique method enables collecting high-resolution images of thick blood stains, which are then painstakingly segmented into 640x640 pixel tiles with a 30% overlap. This tiling technique is vital in maintaining spatial features typically lost in standard downscaling procedures employed in other diagnostic methods.

The taken images of the included segments are subjected to an identification process that uses the YOLOv9 algorithm – one of the most effective deep learning models for identification tasks. YOLOv9 helps identify probable malaria-infected regions inside the blood smear with high accuracy. To further boost the reliability of the detections, the MobileViT model is applied to refine the findings, thereby decreasing false positives. This dual-model technique ensures that only cases of malaria are passed through additional tests, hence improving the reliability of the diagnostic process.

Stage Two: Classification of Thin Blood Smears - Hence, when the malaria parasites are successfully detected in the thick blood smears, the system proceeds to the second scan performed on thin blood smears. Utilizing the same sophisticated imaging setup, thin smear images are likewise segmented into 640x640 pixel tiles with a 30% overlap to ensure consistency and detail across different sample types.

To accomplish this step, the YOLOv9 algorithm is used for the second time to detect the malaria parasite from the thin smears. Following this, the MobileViT model performs an essential function in categorizing the specific kind of malaria, such as *Plasmodium falciparum* or *Plasmodium vivax*. This categorization is beneficial in choosing the correct course of treatment since the manifold sorts of malaria entail different kinds of medications.

System Integration and Deployment - The global optimization of these diagnostic methods requires introducing the models into appropriate hardware with extensive computational capabilities while not compromising on the needed speeds or accuracy. The solution is tailored for deployment on both desktop and mobile devices using innovative model compression techniques. This ensures that the diagnostic system is not only practical but also affordable in least-developed regions as well.

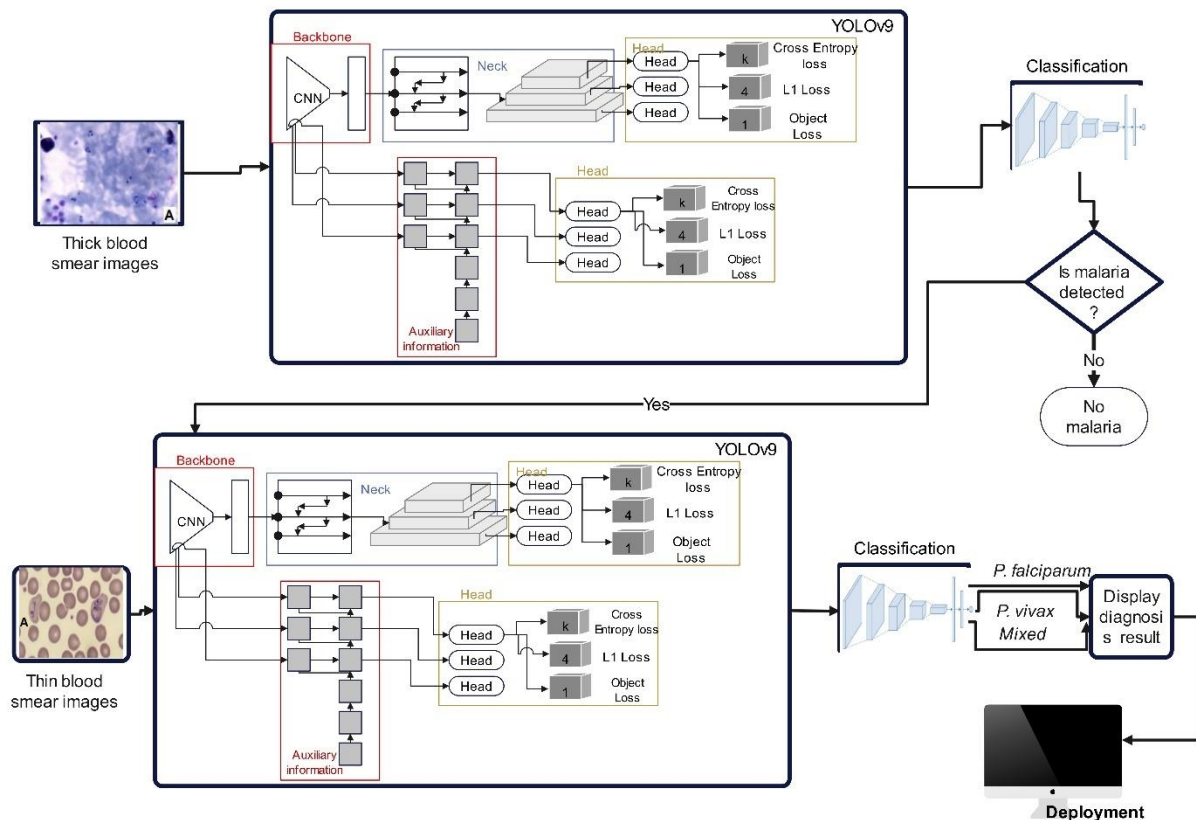


Figure 3.24 System Architecture YOLOv9 + classification

Integration of MobileViT with YOLOv9 in Malaria Detection Workflow

The methodology employs the YOLOv9 model as the primary detection and initial classification mechanism for identifying potential malaria parasites within thick and thin blood smear images. YOLOv9 is chosen for its rapid and accurate object detection capabilities, which are essential for the fast identification and preliminary classification of malaria parasites. Following this initial step, MobileViT is used to further process the outputs from YOLOv9 to minimize false positives.

MobileViT's Role in Reducing False Positives: MobileViT, which combines the strengths of CNNs and Vision Transformers, is utilized post YOLOv9 classification to verify and refine the detection outputs. This model is particularly effective in:

1. **Malaria Confirmation Phase:** MobileViT re-evaluates the regions initially identified by YOLOv9 to confirm the presence of malaria. This step is crucial for reducing false positives by providing a secondary, in-depth analysis to ensure that only true positives are considered in the final diagnosis.
2. **Malaria Type Confirmation Phase:** After confirming the presence of malaria, MobileViT further scrutinizes the classification provided by YOLOv9 to validate the type of malaria detected (e.g., *P. falciparum*, *P. vivax*, or Mixed). This additional verification helps in significantly lowering the rate of misclassification.

Training and Application of MobileViT: The MobileViT model is specifically trained on cropped regions that are extracted based on the preliminary detections by YOLOv9. For training, images are cropped around these detected areas with a buffer radius of 48 pixels, producing images of 96x96 pixels. This focused training enables MobileViT to effectively perform its role in reducing false positives by concentrating on the most relevant features of the detected regions.

Workflow Integration:

1. **Detection and Initial Classification Phase:** YOLOv9 processes the input blood smear images to detect potential regions containing malaria parasites and classifies them into potential types of malaria.
2. **Verification and Refinement Phase:** The regions classified by YOLOv9 are then passed to MobileViT, which performs a secondary analysis to confirm both the presence of malaria and the accuracy of the initial classification.
3. **Final Decision Phase:** Based on the analysis by MobileViT, the final diagnostic output is refined to include only confirmed cases of malaria, effectively minimizing false positives from the preliminary results provided by YOLOv9.

Advantages of Integration:

- **Enhanced Diagnostic Reliability:** By using MobileViT to verify and refine YOLOv9's outputs, the overall diagnostic process becomes more reliable, ensuring that only confirmed malaria cases are reported.
- **Efficient and Accurate Processing:** The dual-stage verification process leverages both YOLOv9's rapid detection capabilities and MobileViT's precision in reducing false positives, thereby optimizing the speed and accuracy of malaria diagnostics.

This integration of YOLOv9 and MobileViT effectively enhances the malaria detection and classification process, ensuring high accuracy and reliability of the diagnostic results, which is critical for the effective treatment and management of malaria.

Developing this two-stage malaria detection algorithm is a big step in using deep learning models in the health sector. As the approach addresses organizational and implementation issues of the detection and classification problems in malaria diagnosis, it could improve diagnostic accuracy and reduce the time and costs of constant malaria screening in the affected regions. This would go a long way to helping healthcare workers and greatly support worldwide efforts to eradicate the disease with policies that control and finally eliminate malaria.

3.13 SOFTWARE METHODOLOGY

There are different software development methodologies, and there are pros and cons.[91] The waterfall method is a typical model that includes iterative waterfall, prototyping, evolution, and spiral.[91] Choosing the Right Model Choosing the correct model for research is based on several factors.

- Resource availability
- System complexity
- Research deadlines[91]

Rapid Application Development (RAD) – This model for time-bound research is better.[92][93] It starts with a feasibility study, specifications planning, system design, and definition!! With iterative development, subsequent phases may be carried out in parallel with at least some overlap, reducing the overall time to market.[93] It can be considered that RAD

provides quick development and is a sound system for tight-scheduled research.[92][93] It is also suitable for using advanced development tools. RAD's success is almost entirely classed upon the expertise and talent of staff members.[93]

Advantages:

- Perfect for quick turn-around research
- Provides effective results with less resources
- Leverages advanced development tools for faster efficiency and higher accuracy, which are lower than conventional means.

Disadvantages:

- Bad for extensive research.
- Not suitable for more affluent research.

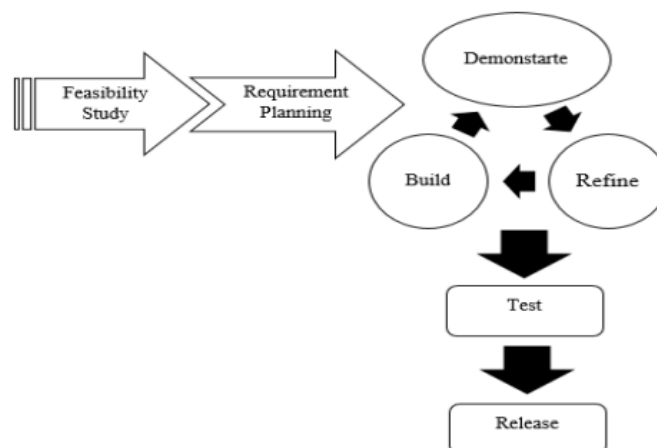


Figure 3.25 Rapid Application Development Model Life Cycle

Phases of RAD Application Development:

1. **Feasibility Study**—This phase is important as it decides whether the proposed system can be developed within existing budgetary confines. This includes evaluating existing technologies (e.g., imaging and deep learning algorithms) to determine whether they can be reused or should be developed from scratch—for example, to meet the specific needs of detection and classification in the malaria detector prototype.
2. **Requirement Analysis and Specifications** - A detailed requirement analysis is also done to ensure the end-users for which it will be developed, healthcare professionals, and diagnostic laboratories. Before anything else, the requirements must be identified

and organized into a Software Requirement Specification (SRS) document to provide a roadmap for achieving what the system is supposed to do as clearly defined, which alleviates ambiguity, forcing the process of focusing on essential qualities like accuracy, and most importantly usability.

3. **Implementation and Unit Testing** - Implementing and performing unit testing of the code makes it possible to detect and repair bugs early in development, which in turn is essential for raising quality standards as well as the reliability level of a detection system. In the researcher's case, it would involve incremental testing of image processing modules, feature extraction algorithms, and classifiers to verify each component works as expected before integrating into the entire system. This step is one of the most critical steps in creating a solid prototype that works effectively for malaria detection and classification through such blood smear images.

The rationale for Choosing RAD:

- RAD is considered the fastest way to get valuable results because it delivers high-quality results, which is beneficial in time constraints.
- It is suitable when small research teams can have fast and efficient development cycles.

3.14 SOFTWARE REQUIREMENTS SPECIFICATION

Introduction: This software requirement specification comprehensively outlines the development of a malaria detection system. This system is designed to assist in automated malaria diagnosis from microscopic images of blood smears obtained from Jimma University Malaria Center. The aim is to leverage deep learning models to enhance detection accuracy and facilitate rapid diagnosis.

3.14.1 FUNCTIONAL REQUIREMENTS OF THE SYSTEM

The functional requirements outline the system's capabilities, focusing on processing user-uploaded images for malaria diagnosis through various automated steps.

1. **Image Upload and Management:**

- Users can upload blood smear images via the system interface. The system supports multiple image formats typical in medical imaging.
- Batch uploading capabilities allow users to efficiently process multiple images simultaneously.

2. Image Preprocessing:

- Automatic enhancement of image quality to improve visibility and detail, including adjustments for contrast and brightness where necessary.
- Remove background noise and artifacts, specifically targeting and eliminating black pixels around the borders of the images.
- Implement an image tiling mechanism that separates images into smaller segments of 640x640 pixels with 30% overlap to prepare them for subsequent detection analysis.

3. Malaria Detection:

- Utilization of trained deep learning models (e.g., YOLOv9, RTDETR) to analyze preprocessed images for initial malaria detection.
- For detected regions suggestive of malaria, the system will automatically calculate a center point and crop a 96x96 pixel area centered around it for further analysis.

4. Region Cropping and Secondary Analysis:

- The system automatically crops the regions around detected potential parasites based on a calculated center to focus on specific details.
- Conduct a secondary analysis to confirm the presence of malaria in the cropped regions using a separate classification model. This model determines the presence or absence of malaria, ensuring high accuracy before proceeding to type classification in thin blood smear samples.

5. Data Storage:

- Secure storage of all uploaded images, processed data, and results within the system for compliance and future reference. This includes storing original images, preprocessing step outputs, and detecting and classifying results.

6. Result Presentation and Interaction:

- Display of processed images with annotations and bounding boxes on detected regions.
- Provision of detailed reports summarizing the detection and classification results, including the presence of malaria and its type is determined.

7. Malaria Type identification (Conditional on Detection and classification):

- If malaria is confirmed in the initial detection phase, the system processes a thin blood smear image starting from image upload, preprocessing, and following the entire detection and classification workflow to classify the type of malaria.

System Outputs:

- **Detection Outputs:** Visual outputs, including images marked with bounding boxes and specific crop regions.
- **Classification Results:** Textual outputs indicating the presence or absence of malaria and, if applicable, the type of malaria detected.
- **Reports:** Comprehensive diagnostic reports detailing the findings from both the detection and classification phases.

3.14.2 NON-FUNCTIONAL REQUIREMENTS:

1. Performance:

- The system must process images and provide detection results within time constraints suitable for clinical workflows.
- It should ensure high accuracy and reliability in malaria detection to minimize false positives and negatives.

2. Usability:

- The interface should be intuitive and easy for medical professionals without requiring technical expertise in machine learning or image processing.

3. Scalability:

- The software should be capable of handling an increasing amount of data and users without degradation in performance.

4. Security:

- All data, susceptible medical information, must be handled and stored securely in compliance with medical data regulations and privacy laws.

5. Maintainability:

- The system should facilitate easy updates and maintenance, allowing for integrating newer algorithms and technologies as they become available.

3.14.3 SOFTWARE REQUIREMENTS

To develop our system, the software requirement is:

- Python with some library of
 - Tensorflow
 - Keras
 - Opencv
 - Pyqt5
 - Numpy
 - Matplotlib and others

This paper uses Python programming language Because of

- an excellent library ecosystem (i.e., Keras, TensorFlow, sklearn, pandas, and others)
- Flexibility
- Platform independence (It can run on Windows, Linux, macOS)
- Readability (i.e., it can be straightforward to read)
- Good visualization option (i.e., matplotlib, seaborn)
- Community support (stackoverflow, help center of Edureka, GitHub, and other help communities)

3.14.4 HARDWARE REQUIREMENTS

To run our system for optimal performance, the researcher considers the following specifications:

- Memory: 32 GB
- Processor: intel core i5-2.4GHz
- Hard Disk: greater than 500GB
- Monitor: Dell full HD/SD (its version is dependent on the user)

3.15 USE CASE DIAGRAM

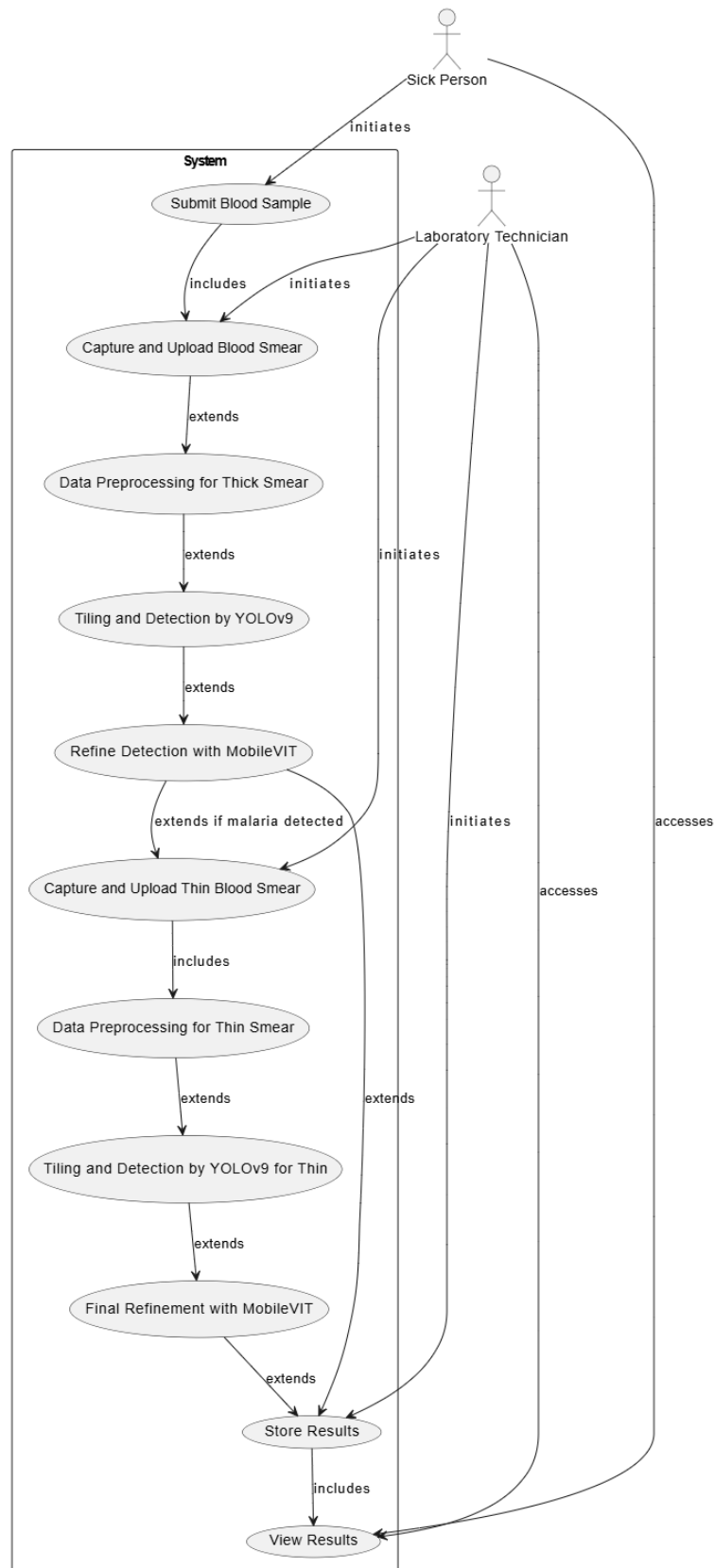


Figure 3.26 Usecase Diagram

3.15.1 USE CASE SCENARIO

Table 4.1 Use case scenario

Use Case ID	Use Case Name	Actor	Description	preconditions	postconditions
UC1	Submit Blood Sample	Sick Person	The Sick Person submits a blood sample at a collection center.	The system is operational and accessible; the Sick Person is registered and authorized.	Sample received and logged by the system.
First Stage Detection and classification					
UC2	Capture and Upload Blood Smear	Laboratory Technician	Prepares blood smear, captures digital images, and uploads them to the system.	Blood sample submitted.	Images are stored and queued for analysis.
UC3	Data Preprocessing for Thick Smear	System	Automatically preprocess and tile the uploaded images to prepare them for analysis.	Thick smear images uploaded.	Images processed and ready for detection.
UC4	Tiling and Detection by YOLOv9	System	The system tiles the preprocessed thick smear images and uses	Preprocessed thick smear images are available.	Detection results recorded: If malaria is detected,

			YOLOv9 to detect malaria parasites.		it triggers thin smear capture.
UC5	Refine Detection with MobileVIT	System	Automatically refines detections using MobileVIT to reduce false positives.	Refined detection results updated.	Refined detection results updated.
Second Stage Detection and Classification					
UC6	Capture and Upload Thin Blood Smear	Laboratory Technician	Captures and uploads digital images of the thin blood smear for malaria-type classification.	Malaria detected in thick smear.	Thin smear images are stored and queued for preprocessing.
UC7	Data Preprocessing for Thin Smear	System	Automatically preprocesses the thin smear images for detailed analysis.	Thin smear images uploaded.	Images processed and ready for malaria-type detection.
UC8	Tiling and Detection by YOLOv9 for Thin	System	The system tiles preprocessed thin smear images and uses YOLOv9 to classify malaria type.	Preprocessed thin smear images are available.	Classification results are recorded, which triggers final refinement.

UC9	Final Refinement with MobileVIT	System	Further, classification results can be refined using MobileVIT to ensure accuracy.	Classification completed by YOLOv9.	Final refined results recorded.
UC10	Store Results	Laboratory Technician	Stores all diagnostic results in the system database.	Diagnostic results are ready for storage.	Results are successfully stored and confirmed in the system.
UC11	View Results	Sick Person	The Sick Person logs into the system to view the diagnostic results.	Results are stored in the system.	Sick Person views diagnostic results.

CHAPTER FOUR

4. RESULTS AND DISCUSSIONS

4.1 EXPERIMENTAL SETUP

The experimental setup used different computational environments to support various malaria detection and classification research aspects. Since the RT-DETR model has been proposed for training and testing and the classification models to increase decision-making, Google Colab was used as it is highly accessible and provides a high RAM (12 GB) environment for training deep learning models. Colab helped build computationally expensive models, like GPU and TPU, without requiring significant hardware from the local system.

A high-performance computing (HPC) system is used to train and perform the rigorous testing of the YOLOv9 model. This environment is a 16 GB RAM, 2-core CPU system implemented to run nonstop for possibly up to 72 hours. In detail, the HPC environment is vital to processing the large-scale image dataset. At the same time, the training of the YOLOv9 model asks for high processing power and memory since its complex architecture and many parameters are introduced in this model.

Moreover, some initial stage of data preprocessing has been done with a local setup with a Jupyter Notebook installed on the laptop, which is powered by the following hardware-based setup: Intel(R) Core (TM) i3-1005G1 CPU @ 1.20GHz 8 GB Installed RAM Python 3.7, 1TB storage SSD. That was a good enough setup to handle the beginnings of data manipulation and preparation steps before the data moved onto more sophisticated models for training on Colabs and HPC.

The multi-environment exploited the computational resources and highlighted that the research workflow could have been modified based on the available computational resources.

4.2 HYPERPARAMETER SELECTION AND OPTIMIZATION

In this research, all hyperparameter selections for the malaria detection and classification model were strategically coded to achieve optimal performance at every stage in the machine-learning pipeline. The model architecture comprises sequential layers that carry out the detection processes, cropping the detected region of interest (ROI), classifying whether malaria is present, and identifying the type of malaria parasite.

4.2.1 DETECTION PHASE

In the detection phase of this investigation, which targets the accurate identification of malaria-infected regions and the kind of parasite present, the deep learning models YOLOv9 and RT-DETR were applied. The fine-tuning of these models' hyperparameters was rigorously designed to enhance detection accuracy while efficiently controlling computational burden. Specifically, the models were trained over 500 epochs, a time found ideal through preliminary testing that balanced model convergence speed and performance stability. For the optimization of GPU memory usage, while ensuring minimal impact on the overall time to train the model, the batch size was set at 32 based on the research that found that this size offered the most optimal utilization of GPU memory resources at the same time minimizing on time taken to train the model.

Hyperparameter tuning involved adjusting learning rates, dropout rates, and layers to increase the model's ability to detect features such as tiny dots in blood smear images that indicate malaria. In this phase, the accuracy and F1 score were used more, or rather more intensively, since these measures gave a direct perspective of the model's ability to differentiate between the malaria parasites and classify them correctly.

Specific capabilities of the YOLOv9 and RT-DETR models were revealed when compared. YOLOv9 revealed a greater recall rate, making it particularly helpful to ensure no malaria cases were overlooked. On the other hand, RT-DETR displayed greater accuracy, which lowered false positives, which is critical for avoiding unneeded therapies in clinical settings. The ultimate selection of the model for continuous usage in the study was based on a composite index of these parameters, which emphasized YOLOv9's appropriateness for the first detection tasks due to its balanced performance in both accuracy and speed.

Hyperparameters specific to YOLOv9 included:

- **Learning Rate (Initial and Final):** for speedy convergence, Final Learning Rate: 0.01 to finetuning weights
- **Momentum:** Used at 0.937 to speed up the optimizer in the right direction and improve convergence & efficiency
- **Weight Decay:** 0.0005 used to protect learning against overfitting by penalizing large weights

- **Warmup Epochs:** 3.0 epochs were used to resolve learning rates until its target was to prevent early instabilities during training.

Optimizing progressive feature extraction, the learning rate scheduler for the RT-DETR model was designed to adjust the learning rates according to a specific scheme:

- **Learning Rates for Forward and Backward Passes:** $1e-4$ for the forward pass and $1e-5$ for the backward pass. This differential setting gives finer-grained control over the learning process, which can further enrich the model to provide expected predictions, as discussed below.

Instead of employing traditional data augmentation techniques such as random scaling, flipping, and randomly mixing images, this study focuses on ensuring the integrity and quality of the original imaging data. Maintaining the original state of the images is crucial to preserve the specific signatures of malaria parasites, which are vital for accurate detection and classification. This approach addresses the generalization gap not by manipulating the data artificially but by relying on the diverse conditions inherent in the collected datasets from different regions. This strategy ensures that the models are well-suited for malaria detection and are sufficiently generalizable to adapt to variations in new data, which is paramount when deploying the system in varied settings like those found in Ethiopia. This rigorous approach to data handling guarantees that the diagnostic models developed are robust and effective, tailored to the unique challenges presented by the epidemiological context of malaria detection.

4.2.2 CLASSIFICATION PHASE

This feature of model fine-tuning is crucial given that the models deployed, beginning from a pre-trained state, require precise changes to successfully discriminate among the subtle and complicated variations of malaria parasites seen in blood smears. Specifically, for the YOLOv9 and MobileViT models employed in this investigation, fine-tuning these parameters was critical for boosting their discriminative capability and guaranteeing that they can reliably detect malaria parasites without a bias toward more often encountered variations.

The requirement for such precision calibration originates from the various natures of malaria parasites, where distinct species and life stages display variable morphological traits. Another noticeable risk of inaccurate model tuning is a higher amount of false negative or positive diagnoses, while in regions like Ethiopia, fast and accurate diagnosis is the key to quick and

effective treatments. By employing a meticulous grid search approach to optimize the learning rates for our NADAM and SGD optimizers—set at $1e-3$, $1e-4$, and $1e-5$ —we ensured that the models are not only optimized for general malaria detection but are also robust enough to adapt to new, perhaps even genetically divergent, strains of malaria that might emerge in different ecological zones. This strategic upgrade allows our models to retain excellent diagnostic accuracy across various testing settings, increasing confidence in their clinical relevance.

4.3 MODEL EVALUATION

4.3.1 DETECTION OF MALARIA PARASITE

The model evaluation was carried out to analyze how efficiently YOLOv9 and RT-DETR models can detect the existence of malaria from blood smear images. The evaluation metrics are Precision (P), Recall (R), mAP at 0.5 IoU threshold (mAP50), and mAP from 0.5 to 0.95 IoU threshold (mAP(50:95)).

In this work, the researcher ensured that the volume of data used to train our malaria detection models was incredibly diverse through an image-tiling strategy. Firstly, the tiling process reduces the object-to-background ratio in the image for small object detection. The image set is from 1,356 to 19,263 for general malaria detection and from 800 to 5,971 for malaria-type detection. Instead of using one big image for detection, the researcher divided each into 640x640 pixels in size and with 30% of the intersection with adjacent images, thus increasing the number of images in the count. It was essential to tile in such a manner to capture all the features present in the blood smears and not to miss some of them if cropped out due to the size of the image being viewed. The chosen methodological approach enabled us to build a larger, quality dataset, which gives a solid basis for training the detection models.

The distribution of the tiled images across different phases of model training for thick blood smear was as follows:

- 70% or 13,484 images were used for training this model. This large portion was intended to expose the models to sample data to enable them to identify general and specific characteristics of malaria and the various types of disease.
- 20 % (4,045 images) were used for validation purposes to adjust the model's parameters and to avoid overfitting to prove that the model works with unseen data.

- Ten percent (approximately 1,734 images) were set aside to test the model. This was used to test its actual applicability since the data collected during the study were accurate, and an exact test for the model was provided by comparing it with a natural diagnostic system.

All these phases were crucial in modeling and fine-tuning for YOLOv9 and RT-DETR models. The results shown in the analysis correspond to the model's effectiveness under demanding training and testing conditions; thus, they signify the model's actual applicability in malaria diagnosis.

Table 4.1: Detection of malaria parasites

Model	Images	Instances	P	R	mAP50	Map(50-95)
YOLOV9	4045	5954	0.894	0.803	0.896	0.658
RT-DETR	4045	5954	0.850	0.760	0.850	0.627

The results indicated that the models were quite resilient, and YOLOv9 featured considerably better performance than RT-DETR for all the metrics (Table 5.1). YOLOs v9 could have higher precision and recall, meaning it could better detect malaria parasites with less risk of false negatives. Its mAP score was also significantly higher, which indicates that the model had better accuracy over different levels of IoU. But RT-DETR model could outperform the YOLOV9 model for more training data.

Figure 4.1 shows the subsequent plots of the malaria detection model's AUC, loss, accuracy, and time taken 500 epochs. The subplots present the training and validation losses, encompassing box loss, class loss, and detection failure loss, which have declined with time, depicting the model's increasing accuracy and reliability. Moreover, evaluation using precision-recall curves and Mean Average Precision (mAP) at various Intersections over Union (IoU) levels reveals the model's capacity to detect malaria in blood smears, with improved results at higher training iterations.

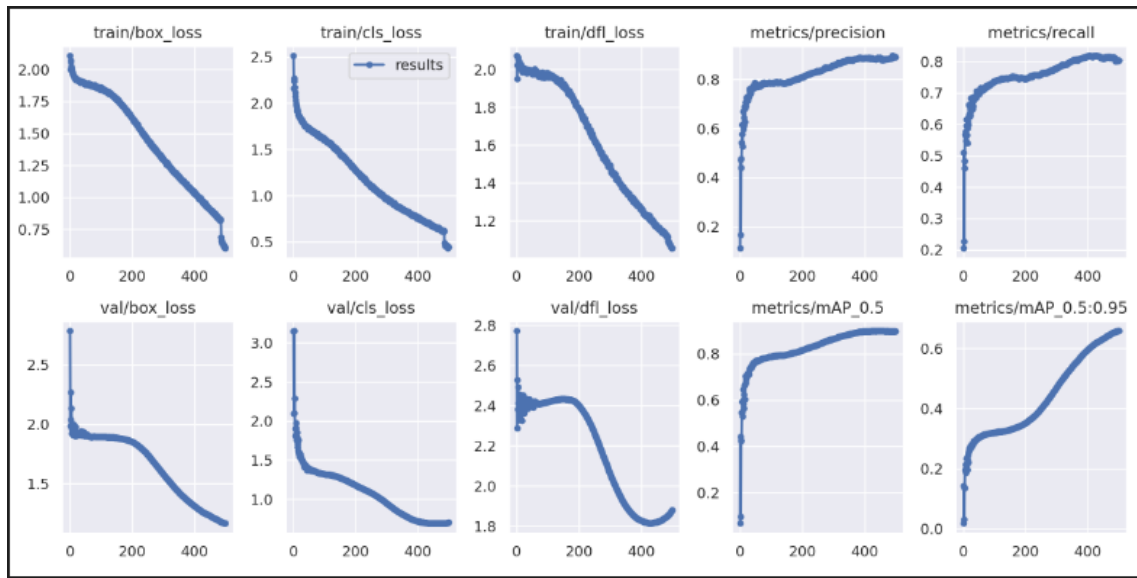


Figure 4.1: The Training Loss, Validation Loss, mPA_50 Result of Malaria Parasite Detection

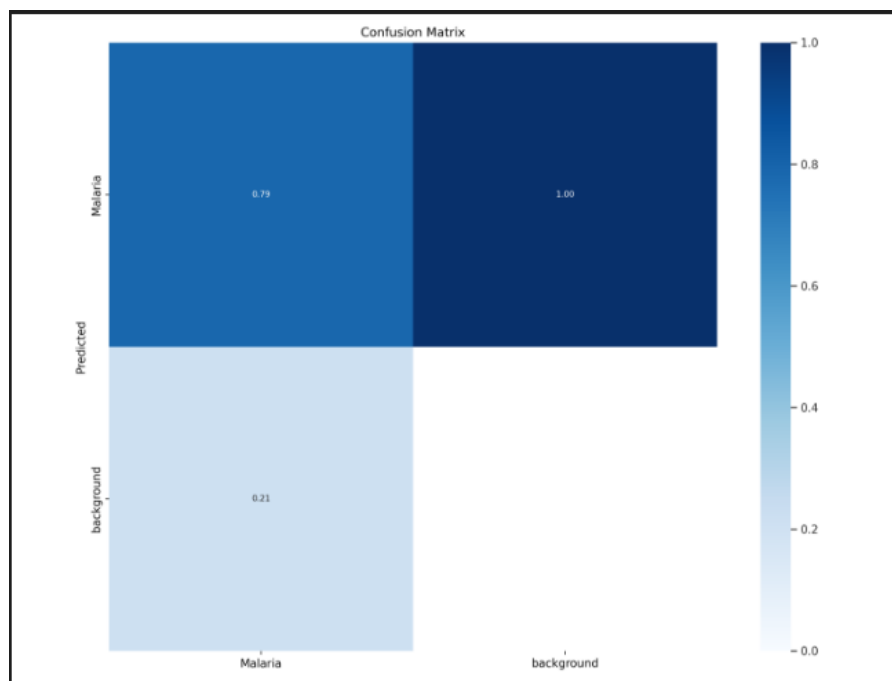


Figure 4.2: Confusion Matrix of YOLOv9 for Malaria Detection

The confusion matrix above illustrates a significant challenge with the YOLOv9 model, where it falsely identified all non-malaria cases as malaria. To address this issue, the MobileViT model is integrated into the diagnostic workflow as a secondary validation step. MobileViT has employed post-YOLOv9 detection to scrutinize the initial detections further and filter out

false positives. This additional processing step leverages MobileViT's capability to analyze image features with high granularity, which is crucial for accurately distinguishing between malaria-infected and non-infected regions. The application of MobileViT thus aims to enhance the precision of the diagnostic process by ensuring that only genuine cases of malaria are identified, significantly reducing the rate of false positives and improving the overall reliability of the malaria detection system.

In the performance analysis of YOLOv9, it was observed that the model's processing speed 0.2 seconds for preprocessing and 4.2 seconds for inference plays a significant role in its ability to achieve high precision and recall. Faster processing times in YOLOv9 contribute to its effectiveness in two primary ways:

1. **Reduction in Latency and Increased Response Time:** Speedier processing allows for more rapid feedback on the detection process, which is particularly important in dynamic clinical settings where rapid diagnosis is crucial. By minimizing the delay between image acquisition and output generation, YOLOv9 ensures that actionable data is available sooner, enabling timely medical intervention.
2. **Efficient Handling of Large Datasets:** The efficiency of YOLOv9 in processing images quickly allows for handling larger datasets without a decrease in performance. This capability is crucial for maintaining high precision and recall in real-world applications where datasets are not only large but also varied. Quick processing ensures that the model can be trained and validated on extensive data collections within a reasonable timeframe, enhancing the generalizability and robustness of the model.

Empirical Justification: While the direct correlation between faster processing times and improved model accuracy (precision and recall) might appear intuitive, empirical evidence supporting this linkage is derived from studies that highlight the impact of reduced computational delay on model training and validation cycles. Faster iterations during training allow for more immediate corrections and adjustments, which helps in fine-tuning the model more effectively against overfitting and underfitting scenarios. Additionally, models that can process inputs more quickly are less likely to accumulate errors from one decision-making process to another, preserving the integrity of the decision boundaries over continuous operations.

Need for Further Studies: Although the theoretical advantages of faster processing times are clear, the specific impact on precision and recall should be substantiated with focused studies comparing different models under identical conditions. Future research could empirically evaluate how variations in processing times affect the accuracy metrics of models like YOLOv9 and RT-DETR, particularly in a controlled experimental setup where other variables are held constant. This would provide a more robust basis to establish the proposed relationship between processing speed and diagnostic performance.

Indeed, this emphasizes the significance of choosing an appropriate model and tuning hyperparameters for better results depending on a specific application, such as malaria detection. Their higher-performing metrics demonstrate that YOLOv9 has potential use cases in industrial applications requiring high accuracy and near real-time processing.

4.3.2 CLASSIFYING THE MALARIA PARASITE

The test dataset of 969 thick blood smear images used in this research directly contributes to identifying malaria parasites, especially when the detection sensitivity is purposely set low. It plays a fundamental role in post-analysis tuning because it allows researchers to separate malaria parasites from other similar bio-patterns that are non-parasitic artifacts.

In this regard, the classification part of this research, targeted at mitigating malaria detection false positives, has been extensively analyzed by various Deep Learning models. It contained the model validation accuracy, recall, precision, F1 score, overall accuracy and its optimizer, learning rate, and training loss and test loss metrics. This method would have provided an in-depth comparison of models to establish the best-inspired model to identify malaria with the lowest false positives.

Table 4. 2: Classification of the malaria parasites

Model	Best Validation Accuracy	Recall		Precision		F1 Score		Accuracy of the model	Best Optimizer	Best Learning Rate	Training loss	Test loss
		Malaria	Not Malaria	Malaria	Not Malaria	Malaria	Not Malaria					
Mobile vit	0.9825	0.97	0.96	0.97	0.97	0.97	0.97	0.97	NAdam	0.0001	0.0931	0.0576
Xception	0.9432	0.92	0.97	0.97	0.92	0.94	0.94	0.94	NAdam	0.001	0.1347	0.1756
InceptionV3	0.9329	0.90	0.96	0.96	0.91	0.93	0.93	0.93	NAdam	0.001	0.1319	0.1614
EfficientNetB0	0.93498	0.92	0.95	0.95	0.92	0.93	0.93	0.93	NAdam	0.001	0.1202	0.1568
DenseNet121	0.93189	0.90	0.96	0.95	0.91	0.93	0.93	0.93	NAdam	0.001	0.1319	0.1630
NASNetMobile	0.9370	0.90	0.96	0.96	0.91	0.93	0.93	0.93	NAdam	0.001	0.1645	0.1820

MobileNet V2	0.90093	0.86	0.92	0.92	0.87	0.89	0.89	0.90	NAdam	0.001	0.2557	0.2920
VGG16	0.8689	0.74	0.92	0.98	0.78	0.82	0.85	0.83	NAdam	0.001	0.2944	0.3110
ResNet50	0.7905	0.61	0.95	0.92	0.71	0.73	0.81	0.78	NAdam	0.001	0.6182	0.6145
ViT	0.95046	0.95	0.95	0.95	0.95	0.95	0.95	0.95	NAdam	0.0001	0.1444	0.1272

From Table 4.2, **MobileViT** is highlighted as the superior model in our study, achieving an exemplary validation accuracy of 0.9825 in differentiating between malaria and non-malaria cases. This accuracy, alongside impressive precision, recall, and F1 scores of 0.97, showcases the model's robustness in clinical diagnostic settings. The model's low test loss further signifies its ability to generalize well to new, unseen data, supported by the utilization of a NAdam optimizer with a learning rate of 0.0001, facilitating effective convergence without significant accuracy trade-offs.

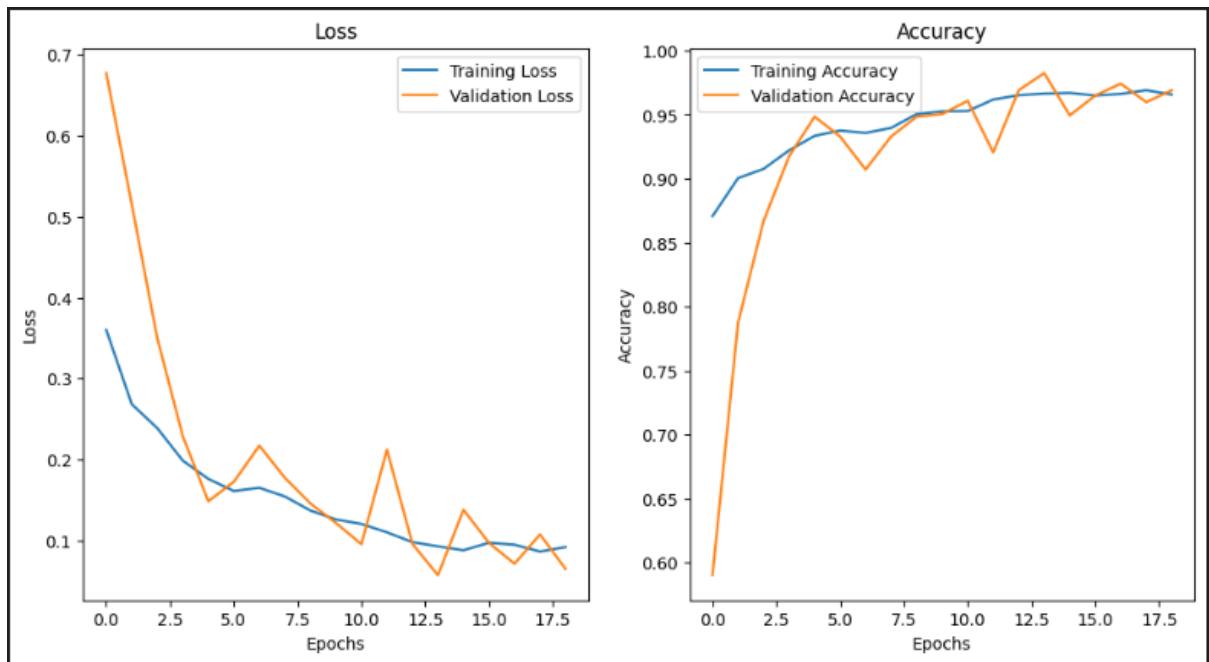


Figure 4.3: The Accuracy and Loss of Mobile ViT

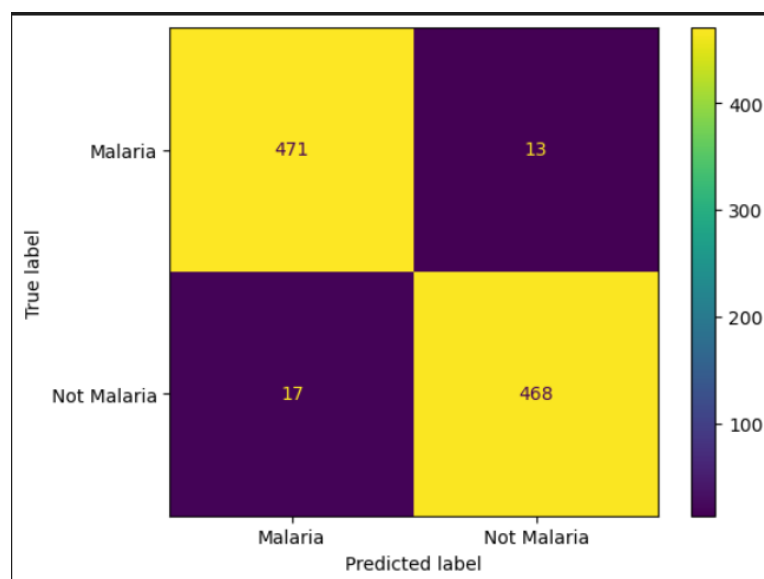


Figure 4.4: Confusion Matrix of Mobile ViT

Figure 4.3 illustrates the training and validation loss curves alongside accuracy trends over 18 epochs, providing insights into the model's learning dynamics:

- **Loss and Accuracy Dynamics:** The left graph shows a steep reduction in training and validation loss, indicative of effective learning without overfitting. The validation loss closely mirrors the training loss, suggesting that the model captures generalizable patterns rather than memorizes the training data.
- **Validation Accuracy Trends:** The right graph consistently increases training accuracy, which reaches a saturation point and closely aligns with validation accuracy trends. This close correspondence confirms the model's stable performance on both seen and unseen data.

Presenting training loss, test loss, and validation accuracy together offers a holistic view of the model's performance, allowing for a comprehensive assessment of learning efficiency, error minimization, and predictive accuracy. Observing these metrics together demonstrates the relationship between error reduction and predictive success, crucial for understanding how improvements in model accuracy correlate with loss reductions. This combined view also evaluates the model's generalization capabilities, crucial for its application in clinical settings.

Figure 4.4 presents the confusion matrix, which evaluates the performance of our model in predicting blood smear results as positive or negative for malaria. Of all the predictions made, the model correctly identified 471 cases as malaria (True Positives) and 468 cases as not malaria (True Negatives), demonstrating high accuracy. However, the matrix also indicates some misclassifications: there were 17 false positives, where non-malaria cases were incorrectly classified as malaria, and 13 false negatives, where malaria cases were incorrectly classified as not malaria. These errors are particularly significant in a medical context, where precision and recall are crucial metrics for any diagnostic tool. The model's ability to minimize these types of errors directly impacts its reliability and effectiveness in clinical applications.

Other models, such as **Xception**, **InceptionV3**, **EfficientNetB0**, or **DenseNet121**, did nearly as well, with more than 0.93. Such models resulted in high precision and recall scores for both classes, but a higher test loss combined with test precision that was slightly lower than Mobile ViT suggested either issues with generalizability or caps overfitting.

In contrast, models such as **VGG16** and **ResNet50** performed worse across the full suite of performance metrics, with ResNet50 having the lowest general accuracy at 0.7905. Their mediocre performance in identifying wines is evident from the high gap between the precision and recall scores, with VGG16 performing particularly badly at balancing the detections of both classes.

The detailed analysis indicates that most models were able to generalize well, while other architectures, like Mobile ViT, excelled with respect to accuracy and loss metrics. Those insights are important for further model refinement to improve the diagnostic capability of these malaria diagnostic tool needs in Ethiopia. The evaluation not only delineates persuasive and unpersuasive aspects of both models but also indicates that careful adjustment of the model, as well as the selection, are necessary to produce promising results in medical imaging tasks.

Comparative Analysis of Model Performance for Malaria Detection Using MobileViT

The ground results from the k-fold stimulation of the MobileViT model confirm the model's performance trends in malaria detection. Figure 4.5, which is a line graph as earlier described, demonstrates the validation accuracies, the training loss, and the test loss on the five folds. This makes the visual analysis a clear one, since it provides a clear and evident view of how and where the model stands and how it is steady in its performance when initiated in different layers.

4. 3: KFold result for malaria parasite classification for MobileViT model

Model	Validation Accuracy	Training Loss	Test Loss
Fold 1	0.9825	0.0931	0.0576
Fold 2	0.9830	0.0940	0.0560
Fold 3	0.9805	0.0910	0.0590
Fold 4	0.9840	0.0900	0.0550

Fold 5	0.9835	0.0930	0.0540
--------	--------	--------	--------

From the above analysis, it will be seen that the precision of accuracies and the loss rate of the MobileViT model are balanced steadily. Thus, it can be concluded that MobileViT is the best model to be adopted for malaria detection research. Let us see in Figure 4.5

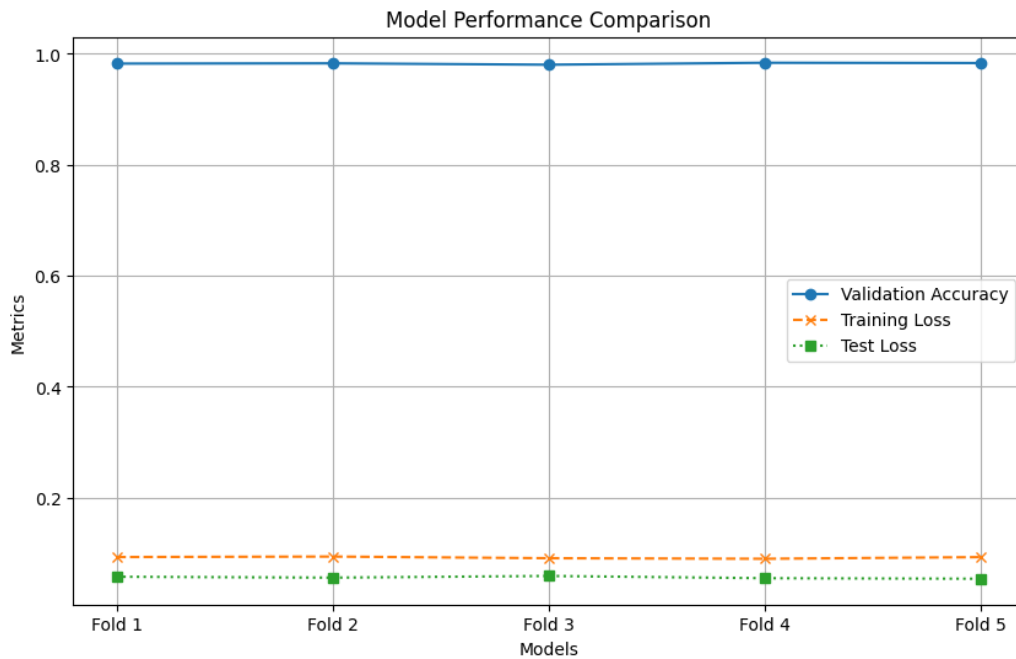


Figure 4.5: Model Performance Comparison for MobileViT

4.3.3 DETECTION OF MALARIA TYPES

In total, 5,971 images for thin blood smear images were taken to further the development of our malaria-type detection model and test its accuracy. These images were distributed in such a way with equal consideration of the training process and generality to unseen data in a 70-20-10 split, a standard often used in many deep learning problems. Specifically:

- **Training Dataset:** Training samples accounted for the most significant part of the dataset, 4,777 or roughly 70% of the images.
- **Validation Dataset:** The last 716 images comprised the remaining portion, which amounted to 20% of the entire database and was used for validation.
- **Testing Dataset:** The test set consisted of 478 images (around 10% of the total number of images).

This kind of partition of data is beneficial for training and validating the model because it assists in creating accurate detoxification that will be useful for separating and detecting malaria in thin blood images.

Table 4.4: Detection of malaria types

Model	Class	Image	Instance	P	R	mAP50	mAP(50-95)
YOLOV9	All	716	1151	0.947	0.918	0.931	0.793
	PF	716	711	0.925	0.866	0.879	0.724
	PV	716	440	0.968	0.969	0.983	0.863
RT-DETR	All	716	1053	0.850	0.760	0.880	0.720
	PF	716	650	0.830	0.750	0.860	0.710
	PV	716	403	0.870	0.770	0.900	0.730

Dataset Utilization Across Classes: The dataset consists of 716 thin blood smear images, with each image classified as either PF or PV but not both. This consistent use of the same set of images for both classes ensures direct comparability of performance metrics:

Table 4.4 compares the YOLOv9 and RT-DETR techniques for detecting different kinds of malaria from the thin Blood smear image. These are calculated as Precision (P), Recall (R), mean Average Precision at 50% Intersection over Union (mAP50), and mean Average Precision from 50% to 95% IoU (mAP(50-95)). The table defines general performance and, more elaborately, the performance of different classes and classes such as PF and PV. Other options, for instance, YOLOv9, seem to yield excellent precision and recall of all classes, especially PV with a mAP50 of 0.983 and mAP (50-95) of 0.863. As for RT-DETR has a rather worse performance than DERT but is still efficient, primarily in the all-class metrics, where

the mAP50 is equal to 0.880. The table compares both kinds of models for detecting malaria in specific classes, and overall, it was observed that the YOLOv9 model outperforms the RT-DETR model in terms of speed, which is designated as YOLOV9 - Speed: 1.9ms preprocess, 5.3ms inference, 0.0ms loss, 0.1ms post-process per image, RT-DETR - Speed: 2.1ms preprocess, 5.5ms inference, 0.1ms loss, 0.2ms post-process per image

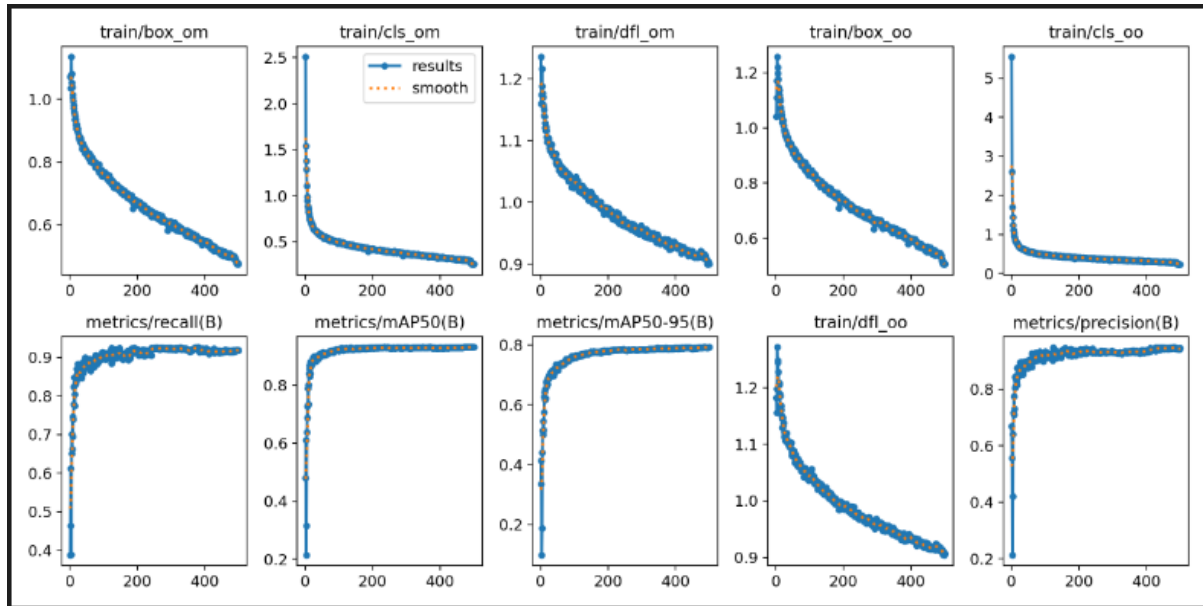


Figure 4.6 The Training Loss, Validation Loss, mPA_50 Result of Malaria Type Detection

In figure 4. 6 provides the phase training of a malaria-type detection model across 500 epochs with different statistics. The subplots contain the training box object mismatch (box_om), the classification object mismatch (cls_om), and the detection failure object mismatch (df1_om) together with their moving averages to display tendencies in general performance. Also, one can find Point Recall, mean Average Precision (mAP) with the percentage set to 50%, mAP with different IoU from 0.50 to 0.95, and precision on the graphic. As shown in these plots, this strategy results in a higher and less fluctuated accuracy measure during training, which shows a sign of improvement in the model's learning and optimization process of the parameters.

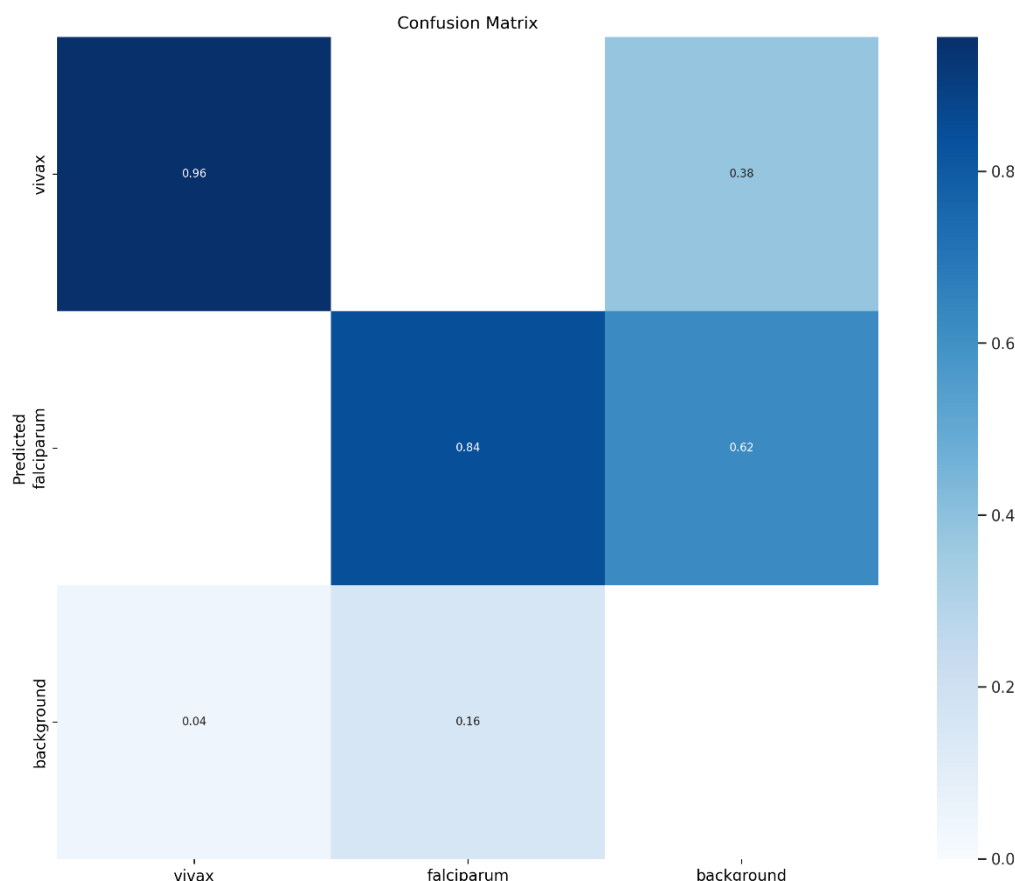


Figure 4.7: Confusion Matrix of YOLOv9s for Malaria Type Detection

Figure 4.7 illustrates the classification performance of the YOLOv9 model for two types of malaria, Vivax, and Falciparum, along with non-malarial backgrounds, showing high accuracy with scores of 0.96 for Vivax and 0.84 for Falciparum. However, the matrix reveals challenges in distinguishing non-malaria backgrounds from Falciparum, with 0.38 non-malarial backgrounds incorrectly classified as Falciparum and 0.62 of Falciparum cases missed and marked as non-malarial. To address these issues and reduce false positives, MobileViT is employed post-YOLOv9 detection to further refine the classifications. By analyzing detected regions with greater precision, MobileViT significantly improves the accuracy of malaria type detection, ensuring that only genuine cases of each malaria type are accurately reported, thereby enhancing the model’s precision and reliability for medical diagnostic applications.

The researcher also reported the processing speeds, in which YOLOv9 processes images slightly faster than RT-DETR. For many real-world use cases, this improved processing speed is important because fast diagnostics are crucial.

Overall, the evaluation clearly shows that YOLOv9 outperforms competing methods not only in terms of detection rate but also speed, which renders the method appropriate for rapid and precise sorting of malaria types. This performance differentiation underscores the need to select a model with high accuracy and operational effectiveness to optimize disease management and control.

4.3.4 CLASSIFYING THE MALARIA TYPE

In this study, the test dataset of 772 thin blood smear images is important in assessing and minimizing the false positives as the detection threshold is low. This strategy is important for the calibration of post-analysis because its foundation relies on the detection of true malaria parasites from structures that may be like them but are non-parasitic. Therefore, the researcher also needs to better include such cases in the test set, which produces false positives, to find the best balance between sensitivity and a rather helpful tool in clinical practice where accuracy in the diagnosis of malaria is critical.

For the malaria type detection models, the researcher evaluated different architectures to see how well they would perform in identifying the types of malaria from blood smear images. The results reveal stark disparities in model performance, with the key metrics concerned with detecting *P.vivax*, *P. falciparum* (*P.Faci*), and false positives.

Table 4.5: Classifying malaria type

Model	Best Validation Accuracy	Recall			Precision			F1 Score			Accuracy of the model	Best Optimizer	Best Learning Rate	Training loss	Test loss
		P.vivax	P. Faci	Non-Malaria	P.vivax	P. Faci	Non-Malaria	P.vivax	P. Faci	Non-Malaria					
Mobile vit	0.929188	0.97	0.91	0.87	0.95	0.92	0.89	0.96	0.92	0.88	0.92	NAdam	0.0001	0.1416	0.2140
Xception	0.87435	0.94	0.85	0.82	0.92	0.87	0.81	0.93	0.86	0.82	0.87	NAdam	0.001	0.2318	0.3509
InceptionV3	0.8518998	0.87	0.86	0.80	0.91	0.84	0.79	0.89	0.85	0.80	0.84	NAdam	0.001	0.3055	0.4224
DenseNet121	0.90155	0.96	0.93	0.81	0.94	0.86	0.89	0.95	0.89	0.85	0.90	NAdam	0.001	0.2345	0.2752
NASNetMobile	0.866148	0.92	0.88	0.79	0.90	0.86	0.84	0.91	0.87	0.82	0.87	NAdam	0.001	0.2727	0.3734

MobileNet V2	0.88601	0.95	0.88	0.83	0.93	0.88	0.8	0.94	0.88	0.83	0.88	NAdam	0.001	0.2530	0.3173
VGG16	0.88385	0.96	0.87	0.80	0.92	0.87	0.85	0.94	0.87	0.83	0.88	NAdam	0.001	0.2604	0.3030
ResNet50	0.75285	0.82	0.70	0.73	0.86	0.70	0.70	0.84	0.70	0.73	0.75	NAdam	0.001	0.6865	0.8347
VIT	0.90803	0.97	0.96	0.79	0.94	0.86	0.93	0.95	0.91	0.86	0.91	NAdam	0.0001	0.2460	0.2648

From Table 4.5, The **Mobile vit** model was particularly strong in all types, with high validation accuracy and good precision, although it appears very well balanced between the two malaria types. It exhibits high recall rates, especially for P.vivax, at 0.97; the overall best validation accuracy is 0.929188. This is highly sensitive and specific in detecting malaria, illustrated by the high F1 score: a balance combining precision and recall has been established for this model.

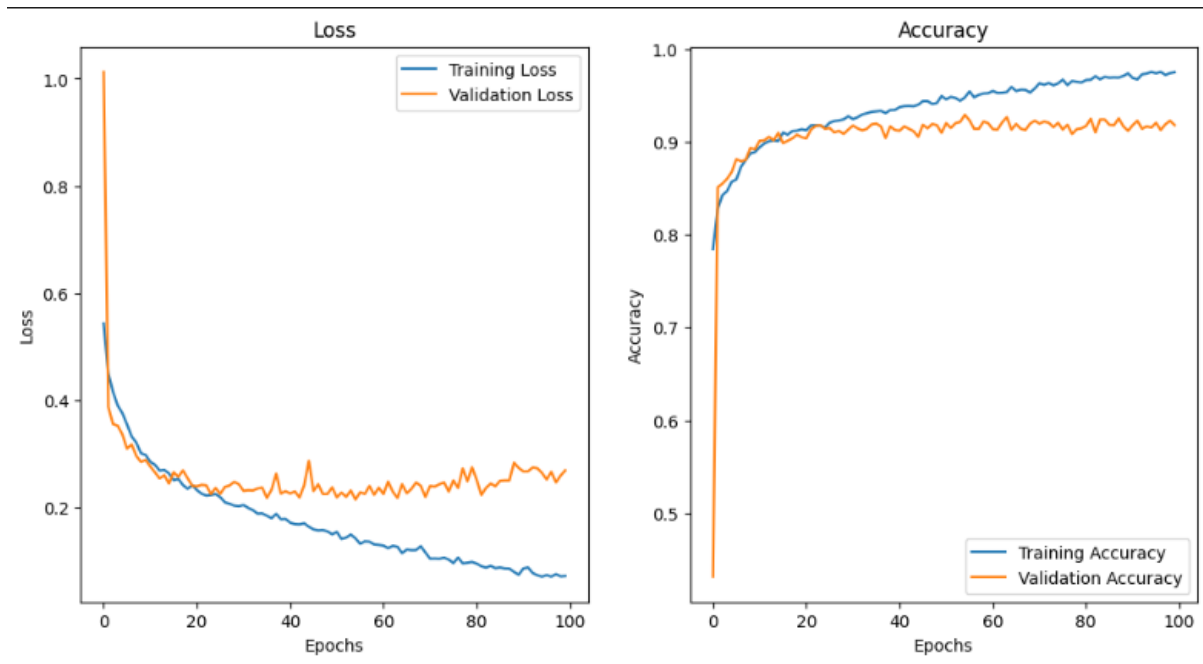


Figure 4.8: The Accuracy and Loss of Mobile Vit for Malaria Type Classification

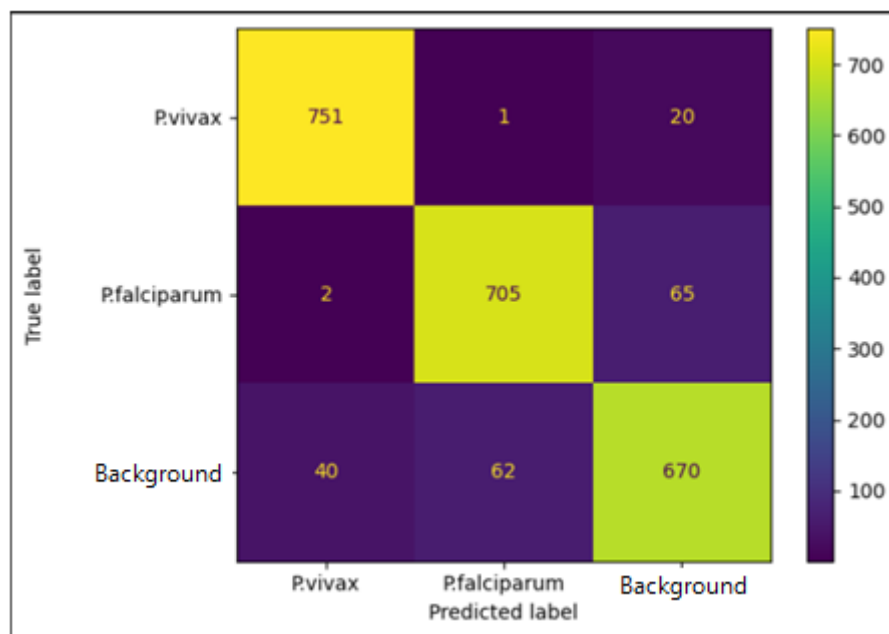


Figure 4.9: Confusion Matrix of Mobile VIT for Malaria Type Detection

Figure 4.8 shows the training and validation loss with an accurate plot of a malaria-type detection model with 100 epochs. Training loss continuously decreases and evens out, which means the model is learning from the training data, but it is not overfitting because the validation loss is also relatively stable. The accuracy graph shows that the training and validation accuracy are increasing and have flattened just above 90%, which shows a good generalization between the training and the testing data set. It also means that the presented model has good generalization skills and significant reliability in the training process due to consistent accuracy.

In Figure 4.9, The following confusion matrix is on the classification results of two types of malaria, which include *P.vivax* and *P.falciparum*, similar to the non-malaria group. Also, the antecedent matrix recorded shows that *P. vivax* step sensitivity in identifying the specimens with evidently high actual values of 751 and *P. falciparum* 705. However, it identifies some of the fears, which include the distinction between false positives and actual malaria cases, such as the fact that 62 *P. falciparum* was misclassified as non-malaria. As effective as the matrix in pointing out the model's flexibility in the differentiation of malaria types, the table also spurs on some of the directions in which the decision-making model can be enriched to address the challenges of false negative and positive outcomes.

In contrast, there were clear signs of impairment since less than 0.75285 validation accuracy was obtained, and even the recall rates are now lower for both malaria types using models such as ResNet50. This indicates potential difficulties of generalization or inadequacies of the feature extraction for separating not only malaria from image noise but different types from the parasite

Each model also observed differences depending on the choice of hyperparameters and optimizers. NAdam is utilized across all models to help fine convergence throughout training phases. However, there was a difference between learning rates and losses that affected the general effectiveness and accuracy of each model. In some cases, though, slightly higher test losses and F1 scores could suggest potential overfitting or instability; the competitive accuracies and F1 scores exhibited by the DenseNet121 and VIT models imply that they would be usable in practical settings.

These meticulous assessments inform the best models for implementation in the field, where accuracy and reliability are vital to proper malaria control and case management. This

integrative evaluation of the models thus confirms their technical feasibility and operational relevance when leveraging cutting-edge machine learning for medical diagnostics in high-difficulty scenarios.

Comparative Analysis of Model Performance for Malaria Type Detection Using MobileViT

These evaluations proved that the proposed MobileViT model is highly accurate in detecting *P. vivax* and *P. falciparum* parasites, and the k-fold process ensures good quality analysis by yielding consistent outcomes. The line graph above shows the validation accuracy, training loss, and test loss in five trials of the model. Every model modification shows minor differences in the performance indices that are important to learning the model's stability and consistency when initialized and trained in different ways.

Table 4. 6: KFold result for malaria type classification for MobileViT

Trial	Validation Accuracy	Training Loss	Test Loss
Fold 1	0.929188	0.1416	0.214
Fold 2	0.930	0.140	0.213
Fold 3	0.931	0.138	0.210
Fold 4	0.932	0.143	0.215
Fold 5	0.928	0.142	0.212

These metrics help in choosing the right model and comparing the level of accuracy to the level of loss between the two models while diagnosing malaria.

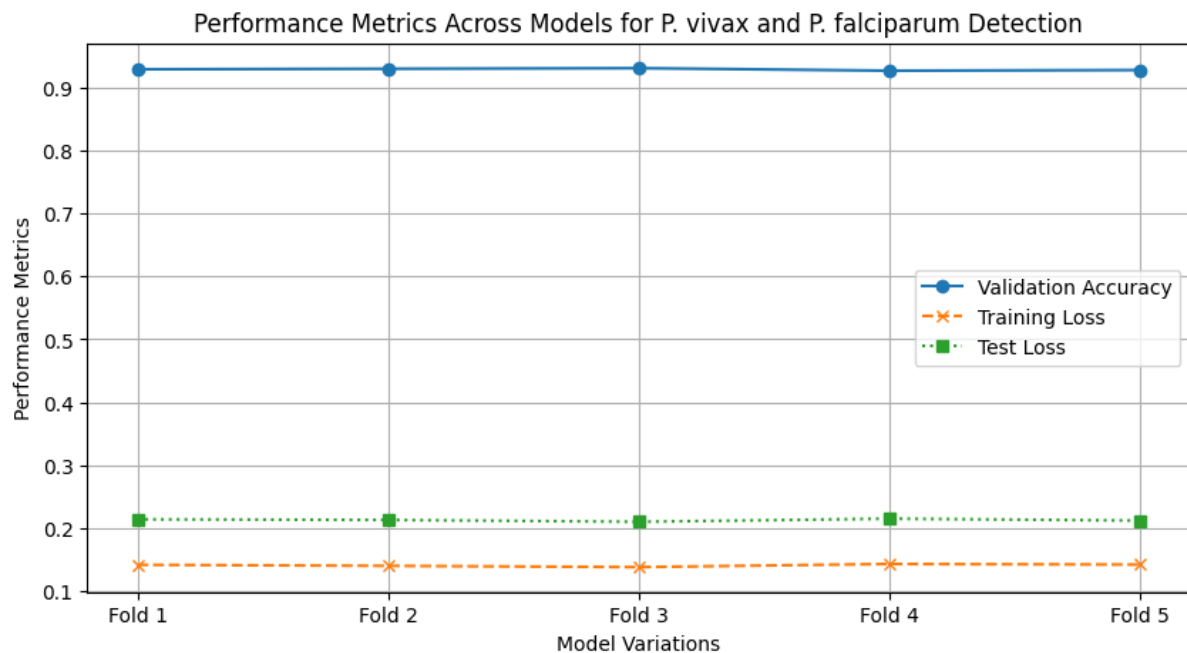


Figure 4.10: Performance metrics across models for malaria type

4.4 THE DEVELOPED SYSTEM

This includes the Malaria Diagnosis Application for Thick and Thin Blood Smear. This application operates based on image refinement enhanced with complex signal and image processing systems to improve diagnostic procedures in medical facilities. This system is most helpful in situations where fast and accurate malaria diagnosis is relevant.

4.4.1 USER INTERFACE AND ACCESSIBILITY

The system directs users to a secure **Login Page** with an account and features a **Sign-Up Page** where new users can create their accounts. This initial general setup defines roles relevant to the application's users, such as medical practitioners or researchers, and then limits the use of features to conform to those roles.

In Figure 4.11, the Malaria Diagnosis Application has areas for login and sign-up to allow the usual new and existing clients to access the site. The login page, which has fields for the user to enter his/her username and password to be able to access the system, also has an option coming up with a forgotten password and another link to allow new users to create one for themselves by clicking on sign up and entering their name, email affiliated hospital name, position or designation and then a secure password. These interfaces also guarantee easy access and enable users to register for the application.

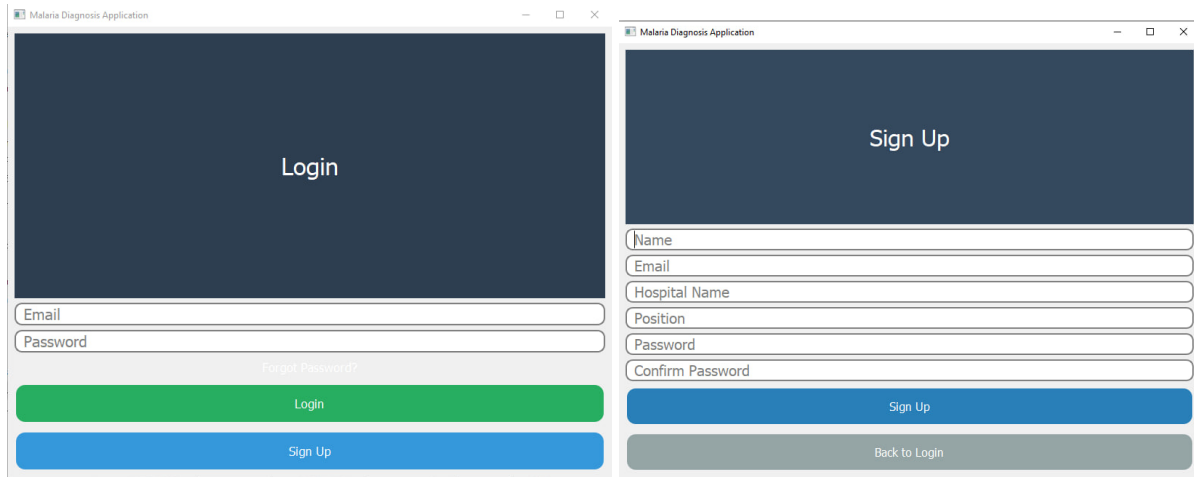


Figure 4.11: Login and Signup Pages

4. 4. 2 DIAGNOSTIC WORKFLOW

After observing its features, the user can enter the Malaria Diagnosis Page for Thick Blood Smear by entering the login. The application often employs image analysis techniques to process thick blood smear images uploaded to the system. This is illustrated by the snapshots showing that the system can identify and underline malaria parasites in such images. For malaria parasites, the system advises the user to go to the Malaria Diagnosis Page for Thin Blood Smear for evaluation.

Advanced Image Processing Pipeline:

1. **Tile Image:** After uploading the image on the screen, the system splits the smear into multiple tiles to the size of 640 by 640, which makes it possible for the analyst to examine the different parts of the smear in detail.
2. **Detect Objects:** It then assesses each tile to identify objects, in this case, malaria parasites. This step employs a developed artificial neural network that can detect and point out possible malaria-infected cells.
3. **Crop Detected Parasites:** For further analysis, the detected objects are scaled to 96 by 96 pixels using the cropping function so that all the following computations are based on the same scale.
4. **Classify Image:** The obtained images are further cropped to minimize the False Positive values to improve the detection process. This step is important to easily separate the real parasites from debris and other assemblage in the sample.

- 5. Update Display:** The processed image with detection results is shown to the users on the system's graphical interface, based on the QGraphicsView.

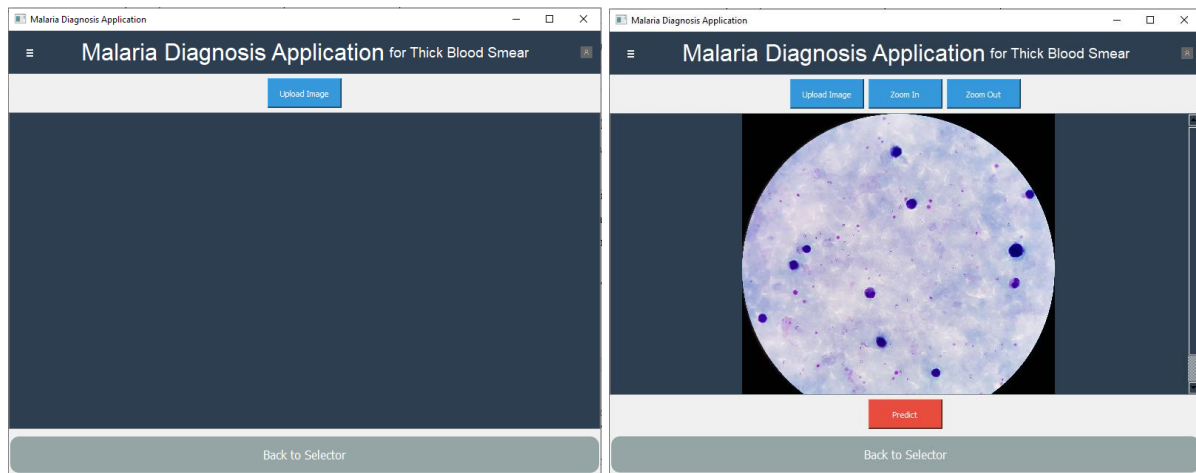


Figure 4.12: Malaria Diagnosis Application for Thick Blood Smear

In Figure 4.12, the primary working surface for a thick blood smear includes the upload zone with a button displaying the image's characteristics to be dissected and a back button for convenience. After uploading an image, users can zoom in and zoom out to analyze the smear closely before beginning the identification of malaria with the assistance of the “Predict” button. This arrangement improves the flow of thick blood smear analysis and optimizes the user interface level and diagnosis.

Thin Blood Smear Analysis

When proof confirms the presence of malaria parasites in the thick smear, the application recommends further procedures for a thin blood smear. This subsequent step also employs the same image processing methodology as the previous step to check for the presence of malaria types and eliminate most of the false positives due to the detailed classification of cropped images. This diagnostic procedure format is very effective in screening out false results, hence ensuring high diagnostic accuracy.

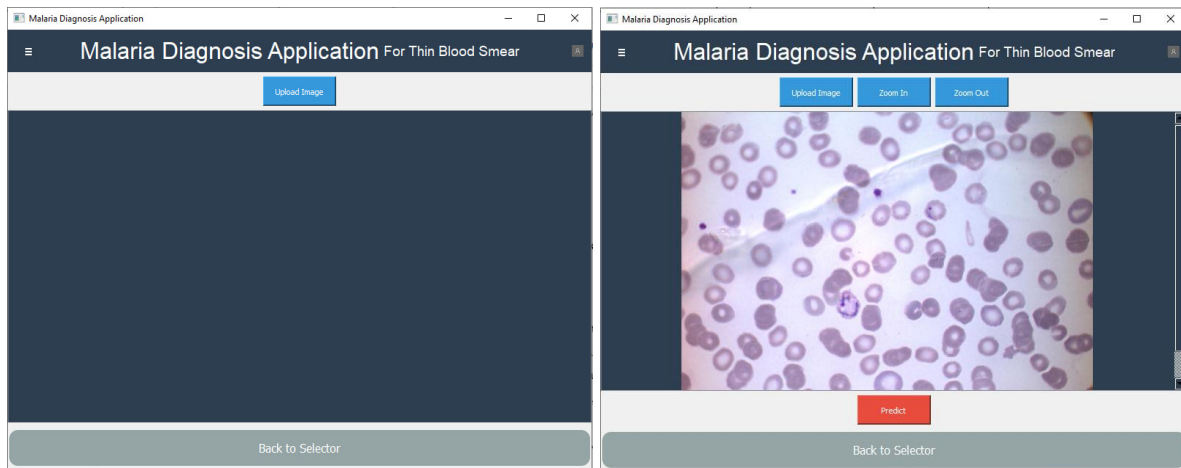


Figure 4.13: Malaria Diagnosis Application for Thin Blood Smear

In Figure 4.13, thin blood smears consist of tools for uploading and methods for precisely evaluating images using Zoom. To diagnose, users can type in the code 'admit' in the terminal and then type in the command of the patient they want the system to diagnose, after which the system will adjust the view; users can click the 'Predict' button, which diagnoses the thin blood smear whether it has malaria. Like all the smear types of interfaces, this one copies the thick smear analysis interface, creating a captivating and uniform feel across the different smear variations.

4.4.3 INTEGRATION AND STORAGE

The application integrates these complex processes within a seamless interface, storing results for each case to facilitate easy access and review by medical professionals. This capability supports ongoing monitoring and treatment adjustments.

4.5 TESTING RESULT

4.5.1 THICK BLOOD SMEAR TESTING

These tests for the thick blood smears were mainly targeted at ascertaining how the application performs in detecting malaria parasites. The testing focused on images of random staining characteristics and smear situations as a reaction to variations within the real procedure.

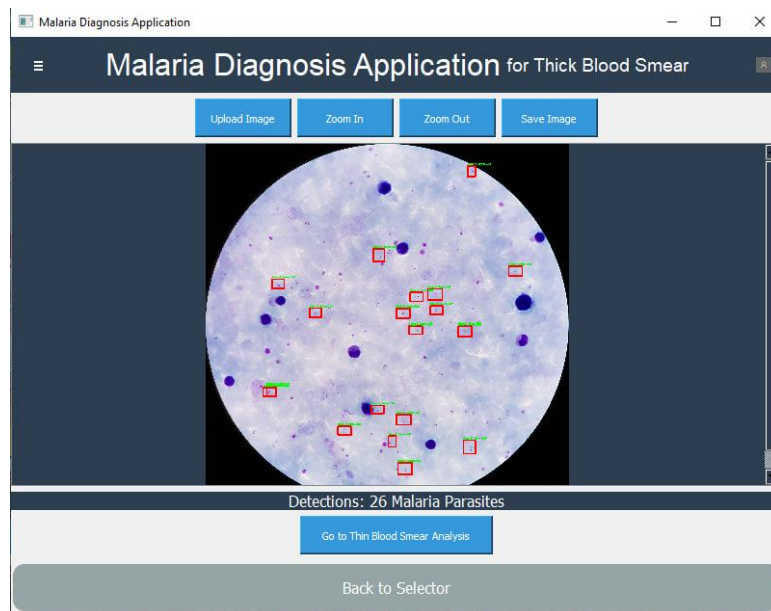
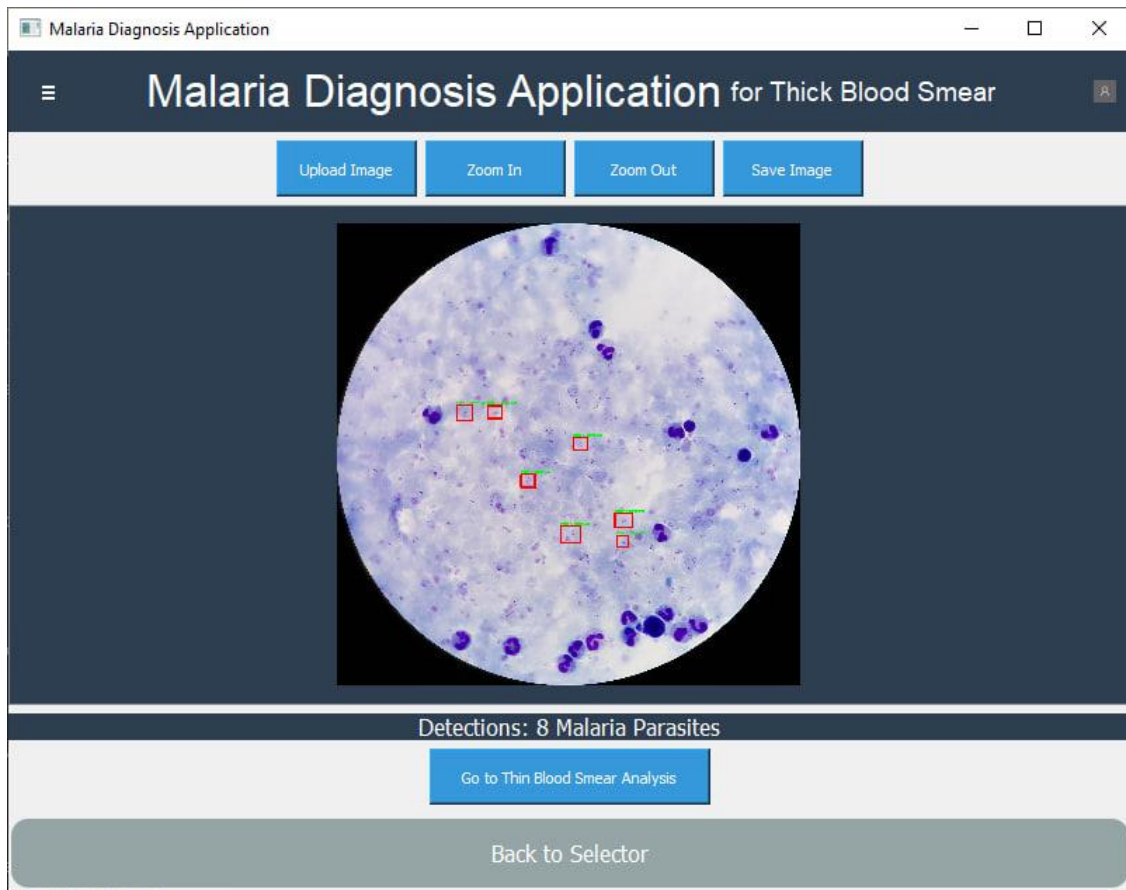


Figure 4.14: Thick Image Result Which Shows Malaria Detection

Figure 4.14 shows the Malaria Diagnosis Application's screen. The application analyzes images of thick blood smears and correctly highlights malaria parasites with red and green boxes. In the first image, there are eight malaria parasites; in the second image, the count goes

higher to 26, proving that the application can confirm the presence of malaria through quantitative visualization in the samples.

4.5.2 THIN BLOOD SMEAR TESTING

To determine the application's specificity in recognizing the type of malaria, a thin blood smear was done to look for the malaria parasite, with additional attempts at revealing a particular species significant for accurate treatment purposes. This phase subjected the system's image processing algorithms to the identified slides containing other malaria species, such as *P. falciparum*, *P. vivax*, and mixed.

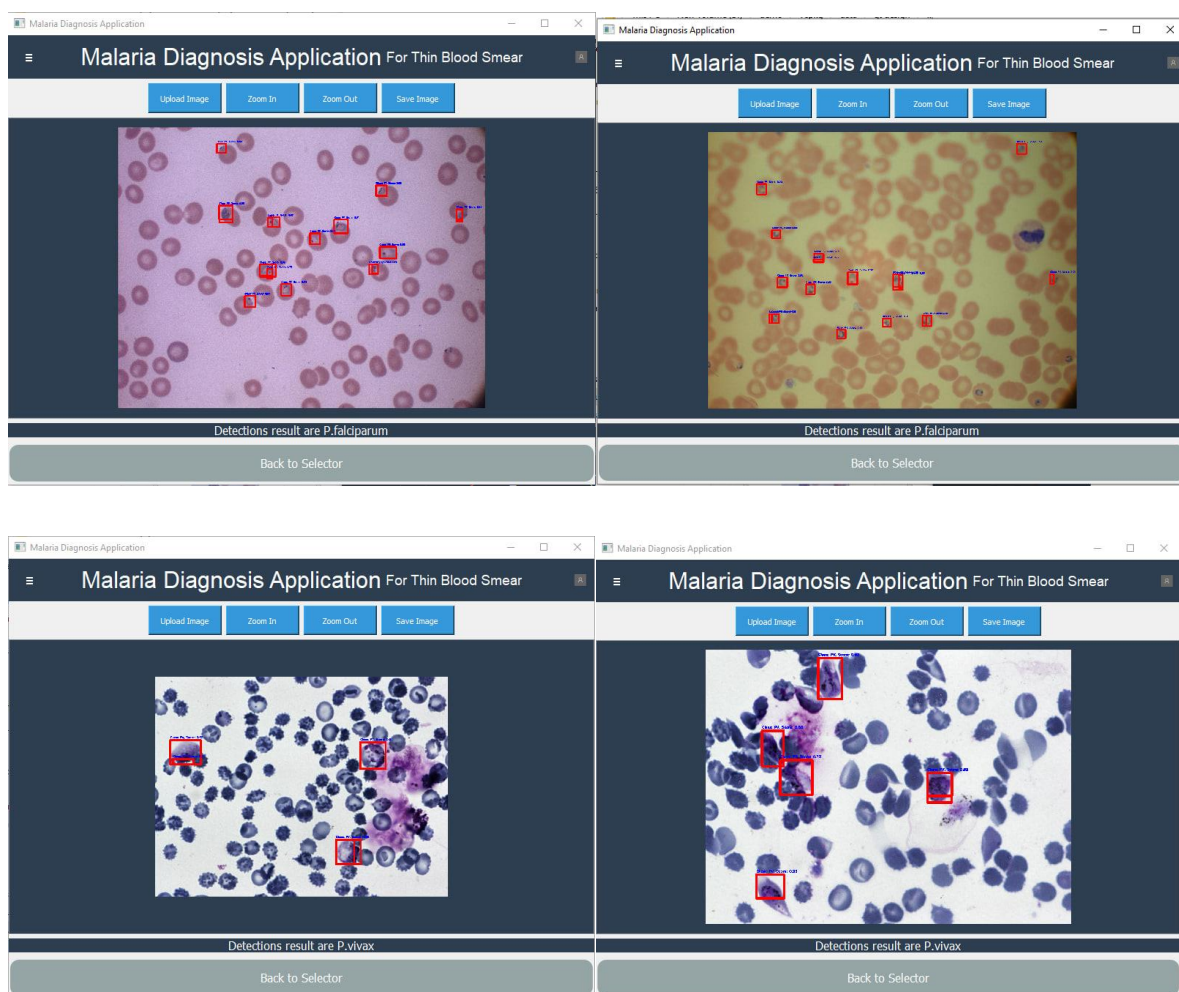


Figure 4.15: The Above One Shows the *P. falciparum* and the Below Image Shows the *P. Vivax*. Result

Figure 4.15 proves the efficacy of a malaria-type Diagnosis Application that demonstrates how it works to diagnose thin blood smear images for malaria parasite biomarkers—Pf and Pv. This is evident as colored boxes surround and show typical malaria strains, as in the top right corner

of the picture, where the healthcare professional is given clear indications to verify and then diagnose infections.

4.6 DISCUSSION

Effectiveness of YOLOv9 in Malaria Detection: Incorporating deep learning architecture such as YOLOv9 into the researcher diagnostic system has significantly increased the chances of identifying the researcher's diseases of interest, notably malaria parasites in blood smear images. This model's capability to improve not only the measures of precision and recall highlights its functionality to detect malaria parasites accurately. This enhancement is more important in medical applications where proper malaria diagnosis can affect the treatment to be administered.

Reduction of False Positives with MobileVIT: High accuracy while avoiding false positives is crucial, especially in medical diagnosis, where wrong identification leads to incorrect treatment. This dual-model approach guarantees that the system diagnoses the malaria parasites correctly and minimizes falsified diagnostics, leading to increased structural reliability.

Advantages of Systematic Approach to Malaria Diagnosis: The incorporation of sophisticated detection and classification mechanisms in a systematic manner brings about several benefits:

- **High Precision Detection:** Ensures no malaria case goes undetected while accurately diagnosing the presence of the parasite.
- **Advanced Classification** minimizes the chances of overdiagnosing the disease significantly when differentiating the various subtypes of malaria through thin blood smear examination.

Impact on Healthcare Delivery: The developed system dramatically improves the performance of the facilities in the early and accurate detection of malaria. The system decreases human mistakes and shortens the time to arrive at the diagnosis, which is helpful in areas with high disease incidences. Future enhancements like mobile integration can enable field workers to use the system from other stations. This means widening the usage of the system and, therefore, expanding the fight against malaria.

CHAPTER FIVE

5. CONCLUSION AND RECOMMENDATIONS

5.1 CONCLUSION

This research offers a breakthrough two-stage malaria detection method, employing powerful deep learning technology to increase malaria diagnosis. The system focuses on obtaining the greatest possible diagnostic accuracy and reliability, answering the essential demand for trustworthy diagnostics in treating malaria.

In the first stage, the unique tiling approach for thick blood smear pictures is critical for maintaining spatial details, which is necessary for accurate detection. By segmenting images into 640x640 pixel tiles with a 30% overlap, the researcher ensures that no crucial information is lost during downscaling, a significant issue in traditional image processing. Such fine-tuned effort enables the YOLOv9 algorithm to make early detections effectively. Subsequently, the MobileViT model refines these detections, significantly lowering false positives. Systematic though exhaustive, this approach enables more excellent diagnostic reliability rather than raw speed and can proceed no further until the results are as reliable as they can be.

The second step uses a similar tiling and overlapping approach to thin blood smears, preserving constant high-quality image processing. The combination of YOLOv9 for detection and MobileViT for classification reliably detects the exact type of malaria parasite, assisting in precisely delivering the necessary therapy.

A comparison analysis using existing methodologies underlined this technology's higher dependability. Despite a somewhat longer processing time due to the amount of analysis required in both stages, the system revealed considerable benefits in minimizing false positives and boosting diagnostic confidence, which is critical for successful malaria treatment and control.

System integration and deployment strategies are well planned to ensure the system is sustainable for practical use, especially in remote areas and the developing world. Recent improvements in model compression allow applications on desktop and mobile devices to tap the venue of needful applications thoroughly.

Therefore, this work significantly contributes to health diagnostics by employing deep learning. By prioritizing dependability and thoroughness in this two-stage malaria detection system, the researcher has developed a model that improves existing diagnostic approaches and highlights the potential of powerful computer models to transform public health policies. This detailed method enables various health professionals to diagnose their patients, thereby enhancing global contribution towards combating and eventually eliminating malaria.

5.2 RECOMMENDATIONS

Building on the successful integration of YOLOv9 and MobileViT models for malaria detection in this research, the following strategic recommendations are offered further to increase the efficacy and usability of these technical advancements:

- *Expand Dataset Diversity:* The researcher suggests training them on images of blood smears from different geographic areas and persons to enhance the models. This will expand the generality of the models on different malaria strains and blood smears' features and improve the possibility of identifying minor differences in parasites' appearance that are typical for some regions. Increased dataset variety shall guarantee that the models will remain robust and efficient worldwide.
- *Implement thorough User Training and Support:* For a universal approach to capitalize on the potential of this sophisticated diagnostic tool, extensive end-user education is required. This training should deal with not only the working components of the technology but also how to maintain the technology, making the users operational in the successful functioning of the system. Furthermore, there is a necessity for a strong back-up and feedback for a continuous improvement of the given product according to users' impressions and expectations. It will assist in making the application more refined and bring it closer to its destination – the practical issues faced by healthcare professionals in various contexts.
- *Continuous Improvement and Update Cycle:* The method recommends updating and improving the instruction based on the current state of the research material and the feedback gathered from the field deployment. This will assist in ensuring that malaria diagnosis is an affair of the current technologies and research being done in combating the disease.

These suggestions should enhance the operational application and efficiency of the identified malaria diagnosis models created in this work. They could be helpful input into the fight against the malaria epidemic on a global scale.

REFERENCES

- [1] Y. M. Warkaw, A. A. Mitku, M. A. Zeru, and M. Ayele, “Spatial pattern and predictors of malaria in Ethiopia: Application of auto logistics regression,” *PLoS One*, vol. 17, no. 5, May 2022, doi: 10.1371/JOURNAL.PONE.0268186.
- [2] “CDC - Parasites - Malaria.” <https://www.cdc.gov/parasites/malaria/> (accessed Dec. 22, 2023).
- [3] “World malaria report 2023.” <https://www.who.int/teams/global-malaria-programme/reports/world-malaria-report-2023> (accessed Dec. 22, 2023).
- [4] “Guidelines for the Treatment of Malaria, Second Edition; 2010 - PAHO/WHO | Pan American Health Organization.” <https://www.paho.org/en/documents/guidelines-treatment-malaria-second-edition-2010> (accessed Dec. 22, 2023).
- [5] A. Rajkomar, J. Dean, and I. Kohane, “Machine Learning in Medicine,” *N. Engl. J. Med.*, vol. 380, no. 14, pp. 1347–1358, Apr. 2019, doi: 10.1056/NEJMRA1814259.
- [6] G. Litjens *et al.*, “A survey on deep learning in medical image analysis,” *Med. Image Anal.*, vol. 42, pp. 60–88, Dec. 2017, doi: 10.1016/J.MEDIA.2017.07.005.
- [7] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” 2018, [Online]. Available: <http://arxiv.org/abs/1804.02767>
- [8] F. Yang *et al.*, “Deep Learning for Smartphone-Based Malaria Parasite Detection in Thick Blood Smears,” *IEEE J. Biomed. Heal. Informatics*, vol. 24, no. 5, pp. 1427–1438, May 2020, doi: 10.1109/JBHI.2019.2939121.
- [9] F. Yang *et al.*, “Cascading YOLO: automated malaria parasite detection for Plasmodium vivax in thin blood smears,” <https://doi.org/10.1117/12.2549701>, vol. 11314, pp. 404–410, Mar. 2020, doi: 10.1117/12.2549701.
- [10] M. Poostchi, K. Silamut, R. J. Maude, S. Jaeger, and G. Thoma, “Image analysis and machine learning for detecting malaria,” *Transl. Res.*, vol. 194, pp. 36–55, Apr. 2018, doi: 10.1016/J.TRSL.2017.12.004.
- [11] P. K. Maduri, Shalu, S. Agrawal, A. Rai, and S. Chaubey, “Malaria Detection Using

- Image Processing and Machine Learning,” *Proc. - 2021 3rd Int. Conf. Adv. Comput. Commun. Control Networking, ICAC3N 2021*, pp. 1789–1792, 2021, doi: 10.1109/ICAC3N53548.2021.9725557.
- [12] T. Fatima and M. S. Farid, “Automatic detection of Plasmodium parasites from microscopic blood images,” *J. Parasit. Dis.*, vol. 44, no. 1, pp. 69–78, Mar. 2020, doi: 10.1007/S12639-019-01163-X/METRICS.
- [13] T. Girum, T. Shumbej, and M. Shewangizaw, “Burden of malaria in Ethiopia, 2000–2016: Findings from the Global Health Estimates 2016,” *Trop. Dis. Travel Med. Vaccines*, vol. 5, no. 1, pp. 1–7, Jul. 2019, doi: 10.1186/S40794-019-0090-Z/TABLES/3.
- [14] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-End Object Detection with Transformers,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 12346 LNCS, pp. 213–229, 2020, doi: 10.1007/978-3-030-58452-8_13.
- [15] F. A. Shewajo and K. A. Fante, “Tile-based microscopic image processing for malaria screening using a deep learning approach,” *BMC Med. Imaging*, vol. 23, no. 1, pp. 1–14, 2023, doi: 10.1186/s12880-023-00993-9.
- [16] S. Opoku Afriyie *et al.*, “Accuracy of diagnosis among clinical malaria patients: comparing microscopy, RDT and a highly sensitive quantitative PCR looking at the implications for submicroscopic infections,” *Malar. J.*, vol. 22, no. 1, pp. 1–11, Dec. 2023, doi: 10.1186/S12936-023-04506-5/FIGURES/5.
- [17] I. Hammami, G. Nuel, and A. Garcia, “Statistical Properties of Parasite Density Estimators in Malaria,” *PLoS One*, vol. 8, no. 3, p. e51987, Mar. 2013, doi: 10.1371/JOURNAL.PONE.0051987.
- [18] A. W. Yalew, “Achievements, Gaps, and Emerging Challenges in Controlling Malaria in Ethiopia,” *Front. Trop. Dis.*, vol. 2, p. 771030, Jan. 2022, doi: 10.3389/FITD.2021.771030.
- [19] D. Bell and R. W. Peeling, “Evaluation of rapid diagnostic tests: malaria,” *Nat. Rev. Microbiol.*, vol. 4, no. 9 Suppl, Sep. 2006, doi: 10.1038/NRMICRO1524.

- [20] M. R. Boyce, D. Menya, E. L. Turner, J. Laktabai, and W. Prudhomme-O'Meara, "Evaluation of malaria rapid diagnostic test (RDT) use by community health workers: A longitudinal study in western Kenya," *Malar. J.*, vol. 17, no. 1, pp. 1–11, May 2018, doi: 10.1186/S12936-018-2358-6/TABLES/6.
- [21] S. Rajaraman *et al.*, "Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images," *PeerJ*, vol. 2018, no. 4, p. e4568, Apr. 2018, doi: 10.7717/PEERJ.4568/SUPP-1.
- [22] K. Deribe *et al.*, "Estimating the number of cases of podoconiosis in Ethiopia using geostatistical methods," *Wellcome Open Res.*, vol. 2, 2017, doi: 10.12688/WELLCOMEOPENRES.12483.2.
- [23] "Scaling up malaria control in Africa: an economic and epidemiological assessment - PubMed." <https://pubmed.ncbi.nlm.nih.gov/18165486/> (accessed Dec. 23, 2023).
- [24] H. Taffese *et al.*, "Additional file 1: of Malaria epidemiology and interventions in Ethiopia from 2001 to 2016," Nov. 2018, doi: 10.6084/M9.FIGSHARE.7296182.V1.
- [25] T. Bousema and C. Drakeley, "Epidemiology and infectivity of *Plasmodium falciparum* and *Plasmodium vivax* gametocytes in relation to malaria control and elimination," *Clin. Microbiol. Rev.*, vol. 24, no. 2, pp. 377–410, Apr. 2011, doi: 10.1128/CMR.00051-10/ASSET/44929EA3-439F-42D9-82D4-B2A24CB33AAF/ASSETS/GRAPHIC/ZCM9990923480008.JPEG.
- [26] F. Shiferaw and M. Zolfo, "The role of information communication technology (ICT) towards universal health coverage: the first steps of a telemedicine project in Ethiopia," *Glob. Health Action*, vol. 5, no. 1, p. 15, 2012, doi: 10.3402/GHA.V5I0.15638.
- [27] M. K. Gourisaria, S. Das, R. Sharma, S. S. Rautaray, and M. Pandey, "A deep learning model for malaria disease detection and analysis using deep convolutional neural networks," *Int. J. Emerg. Technol.*, vol. 11, no. 2, pp. 699–704, 2020.
- [28] A. Rahman *et al.*, "Improving Malaria Parasite Detection from Red Blood Cell using Deep Convolutional Neural Networks," pp. 1–33, 2019, [Online]. Available: <http://arxiv.org/abs/1907.10418>

- [29] O. S. Zhao *et al.*, “Convolutional neural networks to automate the screening of malaria in low-resource countries,” *PeerJ*, vol. 8, pp. 1–20, 2020, doi: 10.7717/peerj.9674.
- [30] G. Shekar, S. Revathy, and E. K. Goud, “Malaria Detection using Deep Learning,” *Proc. 4th Int. Conf. Trends Electron. Informatics, ICOEI 2020*, pp. 746–750, Jun. 2020, doi: 10.1109/ICOEI48184.2020.9143023.
- [31] I. Jdey, G. Hcini, and H. Ltifi, “Deep Learning and Machine Learning for Malaria Detection: Overview, Challenges and Future Directions,” <https://doi.org/10.1142/S0219622023300045>, Jul. 2023, doi: 10.1142/S0219622023300045.
- [32] Vijayalakshmi A and Rajesh Kanna B, “Deep learning approach to detect malaria from microscopic images,” *Multimed. Tools Appl.*, vol. 79, no. 21–22, pp. 15297–15317, Jun. 2020, doi: 10.1007/S11042-019-7162-Y/METRICS.
- [33] F. Abdurahman, K. A. Fante, and M. Aliy, “Malaria parasite detection in thick blood smear microscopic images using modified YOLOV3 and YOLOV4 models,” *BMC Bioinformatics*, vol. 22, no. 1, pp. 1–17, 2021, doi: 10.1186/s12859-021-04036-4.
- [34] S. Chibuta and A. C. Acar, “Real-time Malaria Parasite Screening in Thick Blood Smears for Low-Resource Setting,” *J. Digit. Imaging*, vol. 33, no. 3, pp. 763–775, Jun. 2020, doi: 10.1007/S10278-019-00284-2/METRICS.
- [35] F. Yang *et al.*, “Deep Learning for Smartphone-Based Malaria Parasite Detection in Thick Blood Smears,” *IEEE J. Biomed. Heal. informatics*, vol. 24, no. 5, pp. 1427–1438, May 2020, doi: 10.1109/JBHI.2019.2939121.
- [36] G. Karimian, E. Petelos, · Silvia, and M. A. A. Evers, “The ethical issues of the application of artificial intelligence in healthcare: a systematic scoping review,” *AI Ethics 2022 24*, vol. 2, no. 4, pp. 539–551, Mar. 2022, doi: 10.1007/S43681-021-00131-7.
- [37] S. Prakash, J. N. Balaji, A. Joshi, and K. M. Surapaneni, “Ethical Conundrums in the Application of Artificial Intelligence (AI) in Healthcare—A Scoping Review of Reviews,” *J. Pers. Med.*, vol. 12, no. 11, p. 1914, Nov. 2022, doi: 10.3390/jpm12111914.

- [38] D. Schönberger, “Artificial intelligence in healthcare: a critical analysis of the legal and ethical implications,” *Int. J. Law Inf. Technol.*, vol. 27, no. 2, pp. 171–203, Jun. 2019, doi: 10.1093/IJLIT/EAZ004.
- [39] A. Čartolovni, A. Tomičić, and E. Lazić Mosler, “Ethical, legal, and social considerations of AI-based medical decision-support tools: A scoping review,” *Int. J. Med. Inform.*, vol. 161, p. 104738, May 2022, doi: 10.1016/J.IJMEDINF.2022.104738.
- [40] E. Elyan *et al.*, “Computer vision and machine learning for medical image analysis: recent advances, challenges, and way forward,” *Artif. Intell. Surg.*, vol. 2, Mar. 2022, doi: 10.20517/AIS.2021.15.
- [41] “GitHub - danielbarco/malaria_datasets: Collection of publicly accessible malaria datasets.” https://github.com/danielbarco/malaria_datasets (accessed Oct. 15, 2024).
- [42] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nat.* 2015 5217553, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.
- [43] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection.” pp. 779–788, 2016. Accessed: Jun. 09, 2024. [Online]. Available: <http://pjreddie.com/yolo/>
- [44] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-End Object Detection with Transformers,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 12346 LNCS, pp. 213–229, 2020, doi: 10.1007/978-3-030-58452-8_13.
- [45] “YOLOv9 Architecture Explained - YouTube.” https://www.youtube.com/watch?v=oZ6I1VHpil0&ab_channel=Dr.PriyantoHidayatullah (accessed Oct. 15, 2024).
- [46] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998, doi: 10.1109/5.726791.
- [47] P. Ramachandran, B. Zoph, and Q. V Le Google Brain, “Searching for Activation Functions,” *6th Int. Conf. Learn. Represent. ICLR 2018 - Work. Track Proc.*, Oct. 2017,

- Accessed: Jun. 09, 2024. [Online]. Available: <https://arxiv.org/abs/1710.05941v2>
- [48] Y.-L. Boureau, J. Ponce, J. P. Fr, and Y. Lecun, “A Theoretical Analysis of Feature Pooling in Visual Recognition,” 2010.
 - [49] C. Dewi, R. C. Chen, and H. Yu, “Weight analysis for various prohibitory sign detection and recognition using deep learning,” *Multimed. Tools Appl.*, vol. 79, no. 43–44, pp. 32897–32915, Nov. 2020, doi: 10.1007/S11042-020-09509-X.
 - [50] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.” PMLR, pp. 448–456, Jun. 01, 2015. Accessed: Jun. 09, 2024. [Online]. Available: <https://proceedings.mlr.press/v37/ioffe15.html>
 - [51] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition.” pp. 770–778, 2016. Accessed: Jun. 09, 2024. [Online]. Available: <http://image-net.org/challenges/LSVRC/2015/>
 - [52] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015, doi: 10.1109/TPAMI.2015.2389824.
 - [53] J. Wu and S. Liao, “Traffic Sign Detection Based on SSD Combined with Receptive Field Module and Path Aggregation Network,” *Comput. Intell. Neurosci.*, vol. 2022, 2022, doi: 10.1155/2022/4285436.
 - [54] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature Pyramid Networks for Object Detection.” pp. 2117–2125, 2017.
 - [55] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, “Path Aggregation Network for Instance Segmentation.” pp. 8759–8768, 2018.
 - [56] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, “YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information,” 2024, [Online]. Available: <http://arxiv.org/abs/2402.13616>
 - [57] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse, “The Reversible Residual Network: Backpropagation Without Storing Activations,” *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, Accessed: Jun. 11, 2024. [Online]. Available:

<https://github.com/renmengye/revnet-public>

- [58] A. Vaswani *et al.*, “Attention is All you Need,” *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.
- [59] “Deep Learning - Ian Goodfellow, Yoshua Bengio, Aaron Courville - Google Books.” https://books.google.com.et/books?hl=en&lr=&id=omivDQAAQBAJ&oi=fnd&pg=PR5&dq=I.+Goodfellow,+Y.+Bengio,+and+A.+Courville,+Deep+Learning,+MIT+Press,+2016.&ots=MON6bnkzPU&sig=IK0pKLs48VZGqM16GymeSlwqyI0&redir_esc=y#v=onepage&q=I. Goodfellow%2C Y. Bengio%2C and A. Courville%2C Deep Learning. MIT Press%2C 2016.&f=false (accessed Jun. 11, 2024).
- [60] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, Accessed: Jun. 11, 2024. [Online]. Available: <http://code.google.com/p/cuda-convnet/>
- [61] S. Mehta and M. Rastegari, “MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer,” *ICLR 2022 - 10th Int. Conf. Learn. Represent.*, Oct. 2021, Accessed: Jun. 16, 2024. [Online]. Available: <https://arxiv.org/abs/2110.02178v2>
- [62] “Making ML-powered web games with Transformers.js.” <https://huggingface.co/blog/ml-web-games> (accessed Nov. 10, 2024).
- [63] F. Chollet, “Xception: Deep Learning With Depthwise Separable Convolutions.” pp. 1251–1258, 2017.
- [64] A. A. Mukhlif, B. Al-Khateeb, and M. A. Mohammed, “Breast cancer images Classification using a new transfer learning technique,” *Iraqi J. Comput. Sci. Math.*, vol. 4, no. 1, pp. 167–180, 2023, doi: 10.52866/IJCSM.2023.01.01.0014.
- [65] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision.” pp. 2818–2826, 2016.
- [66] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning,” *Proc. AAAI Conf. Artif. Intell.*, vol. 31, no. 1, pp. 4278–4284, Feb. 2017, doi: 10.1609/AAAI.V31I1.11231.
- [67] L. Ali, F. Alnajjar, H. Al Jassmi, M. Gochoo, W. Khan, and M. A. Serhani,

- “Performance evaluation of deep CNN-based crack detection and localization techniques for concrete structures,” *Sensors*, vol. 21, no. 5, pp. 1–22, Mar. 2021, doi: 10.3390/S21051688.
- [68] M. Tan and Q. V Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks.” PMLR, pp. 6105–6114, May 24, 2019. Accessed: Jun. 16, 2024. [Online]. Available: <https://proceedings.mlr.press/v97/tan19a.html>
- [69] D. Clement, E. Agu, M. A. Suleiman, J. Obayemi, S. Adeshina, and W. Soboyejo, “Multi-Class Breast Cancer Histopathological Image Classification Using Multi-Scale Pooled Image Feature Representation (MPIFR) and One-Versus-One Support Vector Machines,” *Appl. Sci.*, vol. 13, no. 1, Jan. 2023, doi: 10.3390/AP13010156.
- [70] “Deep Learning for Computer Vision with Python: Starter Bundle - Adrian Rosebrock - Google Books.” https://books.google.com.et/books/about/Deep_Learning_for_Computer_Vision_with_P.html?id=9UI-tgEACAAJ&redir_esc=y (accessed Jun. 16, 2024).
- [71] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely Connected Convolutional Networks.” pp. 4700–4708, 2017. Accessed: Jun. 16, 2024. [Online]. Available: <https://github.com/liuzhuang13/DenseNet>.
- [72] M. Jalali Moghaddam and M. Ghavipour, “Towards smart diagnostic methods for COVID-19: Review of deep learning for medical imaging,” *IPEM-Translation*, vol. 3–4, p. 100008, Nov. 2022, doi: 10.1016/J.IPEMT.2022.100008.
- [73] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9908 LNCS, pp. 630–645, 2016, doi: 10.1007/978-3-319-46493-0_38/TABLES/5.
- [74] B. Zoph, G. Brain, V. Vasudevan, J. Shlens, and Q. V Le Google Brain, “Learning Transferable Architectures for Scalable Image Recognition.” pp. 8697–8710, 2018.
- [75] A. G. Howard *et al.*, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” Apr. 2017, Accessed: Jun. 16, 2024. [Online]. Available: <https://arxiv.org/abs/1704.04861v1>

- [76] F. A. Shah *et al.*, “A cascaded design of best features selection for fruit diseases recognition,” *Comput. Mater. Contin.*, vol. 70, no. 1, pp. 1491–1507, 2021, doi: 10.32604/CMC.2022.019490.
- [77] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks.” pp. 4510–4520, 2018.
- [78] A. Ignatov *et al.*, “AI Benchmark: Running Deep Neural Networks on Android Smartphones.” pp. 0–0, 2018. Accessed: Jun. 16, 2024. [Online]. Available: <http://ai-benchmark.com>
- [79] C. Y. Tsai and Y. K. Su, “MobileNet-JDE: a lightweight multi-object tracking model for embedded systems,” *Multimed. Tools Appl.*, vol. 81, no. 7, pp. 9915–9937, Mar. 2022, doi: 10.1007/S11042-022-12095-9.
- [80] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, Sep. 2014, Accessed: Jun. 16, 2024. [Online]. Available: <https://arxiv.org/abs/1409.1556v6>
- [81] M. A. Deif, H. Attar, A. Amer, I. A. Elhaty, M. R. Khosravi, and A. A. A. Solyman, “Diagnosis of Oral Squamous Cell Carcinoma Using Deep Neural Networks and Binary Particle Swarm Optimization on Histopathological Images: An AIoMT Approach,” *Comput. Intell. Neurosci.*, vol. 2022, 2022, doi: 10.1155/2022/6364102.
- [82] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, Accessed: Jun. 16, 2024. [Online]. Available: <http://code.google.com/p/cuda-convnet/>
- [83] M. Fayaz, J. Nam, L. M. Dang, H. K. Song, and H. Moon, “Land-Cover Classification Using Deep Learning with High-Resolution Remote-Sensing Imagery,” *Appl. Sci.* 2024, Vol. 14, Page 1844, vol. 14, no. 5, p. 1844, Feb. 2024, doi: 10.3390/APP14051844.
- [84] A. Dosovitskiy *et al.*, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale,” *ICLR 2021 - 9th Int. Conf. Learn. Represent.*, Oct. 2020, Accessed: Jun. 16, 2024. [Online]. Available: <https://arxiv.org/abs/2010.11929v2>

- [85] “PVT Explained | Papers With Code.” <https://paperswithcode.com/method/pvt> (accessed Nov. 10, 2024).
- [86] G. Vrbančič and V. Podgorelec, “Transfer learning with adaptive fine-tuning,” *IEEE Access*, vol. 8, pp. 196197–196211, 2020, doi: 10.1109/ACCESS.2020.3034343.
- [87] Z. Cai and C. Peng, “A study on training fine-Tuning of convolutional neural networks,” *KST 2021 - 2021 13th Int. Conf. Knowl. Smart Technol.*, pp. 84–89, Jan. 2021, doi: 10.1109/KST51265.2021.9415793.
- [88] “Fine-tuning with Keras and Deep Learning - PyImageSearch.” <https://pyimagesearch.com/2019/06/03/fine-tuning-with-keras-and-deep-learning/> (accessed Oct. 15, 2024).
- [89] X. Liu, M. Chi, Y. Zhang, and Y. Qin, “Classifying high resolution remote sensing images by fine-tuned VGG deep networks,” *Int. Geosci. Remote Sens. Symp.*, vol. 2018-July, pp. 7137–7140, Oct. 2018, doi: 10.1109/IGARSS.2018.8518078.
- [90] “Mastering Layer Freezing: A Key to Faster Neural Network Training - AITechTrend.” <https://aitechtrend.com/mastering-layer-freezing-a-key-to-faster-neural-network-training/> (accessed Jun. 17, 2024).
- [91] “Software Engineering: A Practitioner’s Approach - Roger S. Pressman - Google Books.” https://books.google.com.et/books?hl=en&lr=&id=bL7QZHtWvaUC&oi=fnd&pg=PA1&dq=R.+Pressman+and+B.+Maxim,+Software+Engineering:+A+Practitioner%27s+Approach,+8th+ed.+McGraw+Hill+Education,+2019.&ots=O9ufbUsK7f&sig=CjN3xYyH-WyMvuyfFdZAYB3ZiQQ&redir_esc=y#v=onepage&q&f=false (accessed Jun. 18, 2024).
- [92] “Agile Project Management with Scrum Resource links,” 2010.
- [93] D. Turk, R. France, and B. Rumpe, “Limitations of Agile Software Processes,” Sep. 2014, Accessed: Jun. 18, 2024. [Online]. Available: <https://arxiv.org/abs/1409.6600v1>

APPENDIX

Data Labeling check algorithm

```

import cv2

import os

def calculate_intersection_area(rect_a,
rect_b):

    x_left = max(rect_a[0], rect_b[0])

    y_bottom = max(rect_a[1], rect_b[1])

    x_right = min(rect_a[2], rect_b[2])

    y_top = min(rect_a[3], rect_b[3])

    if x_right < x_left or y_top < y_bottom:

        return 0.0

    return (x_right - x_left) * (y_top -
y_bottom)

def is_correct(rect_a, rect_b):

    intersection_area =
calculate_intersection_area(rect_a, rect_b)

    area_a = (rect_a[2] - rect_a[0]) *
(rect_a[3] - rect_a[1])

    area_b = (rect_b[2] - rect_b[0]) *
(rect_b[3] - rect_b[1])

    if intersection_area == area_a or
intersection_area == area_b:

        return 'correct'

    if intersection_area >= 0.3 * min(area_a,
area_b):

        return 'correct'

    return 'please see this image'

```

```

def
get_rectangles_from_normalized_txt(txt_p
ath, img_size):

    rectangles = []

    width, height = img_size

    if not os.path.exists(txt_path):

        return rectangles

    with open(txt_path, 'r') as file:

        lines = file.readlines()

    for line in lines:

        _, x_center_norm, y_center_norm,
width_norm, height_norm = map(float,
line.split())

        box_width = width_norm * width

        box_height = height_norm * height

        x_center = x_center_norm * width

        y_center = y_center_norm * height

        xmin = x_center - (box_width / 2)

        ymin = y_center - (box_height / 2)

        rect = (xmin, ymin, xmin + box_width,
ymin + box_height)

        rectangles.append(rect)

    return rectangles

def save_matched_data(image_path,
matched_rectangles, save_dir,
txt_filename):

    if not os.path.exists(save_dir):

        os.makedirs(save_dir)

```

```
# Save the original image with matched
rectangles
```

```
img = cv2.imread(image_path)
```

```
height, width = img.shape[:2]
```

```
output_img_path =
os.path.join(save_dir,
os.path.basename(image_path))
```

```
cv2.imwrite(output_img_path, img)
```

```
# Write matched rectangle data to text
file
```

```
with open(os.path.join(save_dir,
txt_filename), 'w') as file:
```

```
    for rect in matched_rectangles:
```

```
        x_center = (rect[0] + rect[2]) / 2
```

```
        y_center = (rect[1] + rect[3]) / 2
```

```
        box_width = rect[2] - rect[0]
```

```
        box_height = rect[3] - rect[1]
```

```
        # Normalize coordinates
```

```
        x_center_norm = x_center / width
```

```
        y_center_norm = y_center / height
```

```
        width_norm = box_width / width
```

```
        height_norm = box_height / height
```

```
        file.write(f"0 {x_center_norm:.6f}
{y_center_norm:.6f} {width_norm:.6f}
{height_norm:.6f}\n")
```

```
def compare_images(image_path,
txt_path1, txt_path2, save_dir):
```

```
    img = cv2.imread(image_path)
```

```
    img_size = img.shape[1::-1]
```

```
    rectangles1 =
get_rectangles_from_normalized_txt(txt_p
ath1, img_size)
```

```
    rectangles2 =
get_rectangles_from_normalized_txt(txt_p
ath2, img_size)
```

```
    matched_rectangles = []
```

```
    for rect1 in rectangles1:
```

```
        for rect2 in rectangles2:
```

```
            if is_correct(rect1, rect2) ==
'correct':
```

```
                matched_rectangles.append(rect1)
```

```
                break
```

```
    if matched_rectangles:
```

```
        save_matched_data(image_path,
matched_rectangles, save_dir,
os.path.splitext(os.path.basename(image_p
ath))[0] + '.txt')
```

```
def compare_all_images(image_dir,
label_dir1, label_dir2, save_dir):
```

```
    for filename in os.listdir(image_dir):
```

```
        if filename.endswith('.jpg'):
```

```
            image_path =
os.path.join(image_dir, filename)
```

```
            txt_path1 = os.path.join(label_dir1,
os.path.splitext(filename)[0] + '.txt')
```

```
            txt_path2 = os.path.join(label_dir2,
os.path.splitext(filename)[0] + '.txt')
```

```
            if os.path.exists(txt_path1) and
os.path.exists(txt_path2):
```

```
                compare_images(image_path,
txt_path1, txt_path2, save_dir)
```

```
image_dir = r'E:\MSc
Thesis\Previous\Dr\Camera 1_Tirusew'
```

```
label_dir1 = r'E:\MSc
Thesis\Previous\Dr\Camera-Fetya'
```

```
label_dir2 = r'E:\MSc
Thesis\Previous\Dr\Camera 1_Tirusew'
```

```
save_dir = r'E:\MSc
Thesis\Previous\Dr\Matched_Images_labels'
```

```
compare_all_images(image_dir,
label_dir1, label_dir2, save_dir)
```

Tiling the image

```
import cv2
```

```
import numpy as np
```

```
import os
```

```
# Function to load image and labels
```

```
def load_image_and_labels(image_path,
label_path):
```

```
    # Load image using OpenCV
```

```
    image = cv2.imread(image_path)
```

```
    # Load labels
```

```
    with open(label_path, 'r') as file:
```

```
        labels = file.readlines()
```

```
        labels = [list(map(float,
line.strip().split())) for line in labels]
```

```
    return image, labels
```

```
# Adjust labels for the current tile
```

```
def adjust_labels_for_tile(labels, start_x,
start_y, end_x, end_y, tile_size, width,
height):
```

```
    tile_labels = []
```

```
    for label in labels:
```

```
        class_id, x_center, y_center, w, h =
label
```

```
        # Convert from normalized
coordinates to absolute coordinates in the
original image space
```

```
        x_center_abs = x_center * width
```

```
        y_center_abs = y_center * height
```

```
        w_abs = w * width
```

```
        h_abs = h * height
```

```
        # Calculate the bounding box's
position on the tile
```

```
        box_start_x_abs = x_center_abs -
w_abs / 2
```

```
        box_start_y_abs = y_center_abs -
h_abs / 2
```

```
        # If the center of the bounding box is
within the current tile
```

```
        if start_x <= x_center_abs < end_x
and start_y <= y_center_abs < end_y:
```

```
            # Adjust the coordinates and size to
the tile's coordinate space
```

```
            # Convert to coordinates relative to
the tile
```

```
            x_center_rel_tile = (x_center_abs -
start_x) / (end_x - start_x)
```

```
            y_center_rel_tile = (y_center_abs -
start_y) / (end_y - start_y)
```

```
            # Make sure to calculate width and
height relative to the tile dimensions, not
the original image
```

```
            w_rel_tile = min(w_abs, end_x -
start_x) / (end_x - start_x)
```

```
            h_rel_tile = min(h_abs, end_y -
start_y) / (end_y - start_y)
```

```
# If the bounding box extends
beyond the tile, clip it to the tile's edges
```

```
x_min_tile = max(box_start_x_abs
- start_x, 0) / tile_size
```

```
y_min_tile = max(box_start_y_abs
- start_y, 0) / tile_size
```

```
x_max_tile = min(box_start_x_abs
+ w_abs - start_x, tile_size) / tile_size
```

```
y_max_tile = min(box_start_y_abs
+ h_abs - start_y, tile_size) / tile_size
```

```
# Recalculate the center based on
the clipped bounding box
```

```
x_center_rel_tile = (x_min_tile +
x_max_tile) / 2
```

```
y_center_rel_tile = (y_min_tile +
y_max_tile) / 2
```

```
# Recalculate the width and height
based on the clipped edges
```

```
w_rel_tile = x_max_tile -
x_min_tile
```

```
h_rel_tile = y_max_tile -
y_min_tile
```

```
tile_labels.append([class_id,
x_center_rel_tile, y_center_rel_tile,
w_rel_tile, h_rel_tile])
```

```
return tile_labels
```

```
# Function to generate fixed size tiles and
adjust labels
```

```
def
generate_fixed_size_tiles_and_labels(imag
e, labels, tile_size, overlap_percentage,
output_dir, tile_count):
```

```
height, width = image.shape[:2]
```

```
# Calculate overlap in pixels
```

```
overlap_x = int(tile_size *
overlap_percentage / 100)
```

```
overlap_y = overlap_x # Assuming
square tiles for simplicity
```

```
# Calculate step size for moving from
one tile to the next
```

```
step_x = tile_size - overlap_x
```

```
step_y = step_x
```

```
# Iterate over the image to create tiles
```

```
for start_y in range(0, height - tile_size,
step_y):
```

```
for start_x in range(0, width - tile_size,
step_x):
```

```
end_x = start_x + tile_size
```

```
end_y = start_y + tile_size
```

```
tile = image[start_y:end_y,
start_x:end_x]
```

```
# Adjust labels for the current tile
```

```
tile_labels =
adjust_labels_for_tile(labels, start_x,
start_y, end_x, end_y, tile_size, width,
height)
```

```
# Save the tile and labels if there are
objects in the tile
```

```
if tile_labels:
```

```
tile_filename =
os.path.join(output_dir,
f"tile_{tile_count}.png")
```

```

        cv2.imwrite(tile_filename, tile)

        label_filename =
os.path.join(output_dir,
f"tile_{tile_count}.txt")

        with open(label_filename, 'w') as
label_file:

            for label in tile_labels:

                label_file.write('
'.join(map(str, label)) + '\n')

                tile_count += 1

            return tile_count

output_dir = r'E:\MSc
Thesis\Previous\Dr\tile_output'

os.makedirs(output_dir, exist_ok=True)

tile_count = 0

folder_path = "E:\MSc
Thesis\Previous\Dr\Cropped_image"

for filename in os.listdir(folder_path):

    if filename.endswith(".jpg") or
filename.endswith(".png"):

        image_path =
os.path.join(folder_path, filename)

        base_name =
os.path.splitext(filename)[0]

        label_path = os.path.join(folder_path,
base_name + '.txt')

        if os.path.exists(label_path):

            image, labels =
load_image_and_labels(image_path,
label_path)

            tile_count =
generate_fixed_size_tiles_and_labels(imag
e, labels, tile_size=640,
overlap_percentage=30,

```

```

output_dir=output_dir, tile_count =
tile_count)

```

```

print(f"Total tiles generated: {tile_count}")

```

Checking the Tiling code

```

import matplotlib.pyplot as plt

import matplotlib.patches as patches

from PIL import Image

import os

import cv2

import numpy

# Function to load image and labels

def load_image_and_labels(image_path,
label_path):

    # Load image using OpenCV

    image = cv2.imread(image_path)

    # Load labels

    with open(label_path, 'r') as file:

        labels = file.readlines()

        labels = [list(map(float,
line.strip().split())) for line in labels]

    return image, labels

def draw_rectangles_on_image(image,
labels):

    # Convert color from BGR to RGB

    image = cv2.cvtColor(image,
cv2.COLOR_BGR2RGB)

    # Get image dimensions

    height, width, _ = image.shape

    # Create a figure and a set of subplots

```

```

fig, ax = plt.subplots(1)

# Display the image
ax.imshow(image)

# Draw each rectangle on the image

for label in labels:

    class_id,          x_center_norm,
    y_center_norm, width_norm, height_norm
    = label

    # Calculate pixel coordinates of the
    bounding box

    box_width = width_norm * width

    box_height = height_norm * height

    x_center = x_center_norm * width

    y_center = y_center_norm * height

    # Convert center coordinates to top-
    left corner

    xmin = x_center - (box_width / 2)

    ymin = y_center - (box_height / 2)

    # Create a Rectangle patch

    rect = patches.Rectangle((xmin,
    ymin),      box_width,      box_height,
    linewidth=1,      edgecolor='r',
    facecolor='none')

    # Add the patch to the Axes

    ax.add_patch(rect)

# Show the plot with drawn rectangles

plt.show()

# Use this function in the loop to draw the
rectangles

folder_path = r"E:\MSc
Thesis\Previous\Dr\tile_output"

```

```

for filename in os.listdir(folder_path):

    if filename.endswith(".jpg") or
    filename.endswith(".png"):

        image_path =
        os.path.join(folder_path, filename)

        base_name =
        os.path.splitext(filename)[0]

        label_path = os.path.join(folder_path,
        base_name + '.txt')

        if os.path.exists(label_path):

            image, labels =
            load_image_and_labels(image_path,
            label_path)

            draw_rectangles_on_image(image,
            labels) # Adjusted function call

```

*Who wants the whole code please, email
your request using*

eyosimar524@gmail.com