

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

MOHAMED KHIDER UNIVERSITY OF BISKRA



Faculty of Exact Sciences, Natural and Life Sciences

Computer Science Department

Dissertation

Presented to obtain the academic master's degree in Computer Science

Option: Information Systems, Optimization, and Decision (SIOD)

Theme

Detecting Microorganisms in Water Using Deep Learning

Presented by **MOUSSAOUI Kaouthar**

Jury:

President	Belouaar Houcine	MCA
Supervisor	Mead Mohamed Nadjib	MCA
Examiner	Touil Keltoum	MAA

Academic year 2023-2024

Acknowledgements

*With great pride and humility, I extend my deepest gratitude to Allah Almighty, whose grace and mercy have enabled this significant achievement. His unwavering support has been the light guiding my path to success. I also express my profound appreciation to my esteemed supervisor, **Dr. Meadi Mohamed Nadjib**, who has generously shared his extensive knowledge and provided invaluable guidance throughout the preparation of this research. His insights and direction have significantly impacted my academic development and endowed me with essential mentorship skills.*

I must not forget to thank the head of the laboratory, whose kind treatment was like a beacon of hope that illuminated my path during dark times.

I also extend my gratitude to the esteemed committee members for accepting and evaluating my work, giving me the motivation to continue my academic journey with determination and persistence.

Finally, I would like to thank everyone who contributed to this journey, whether through their support, encouragement, or belief in my abilities. Their support has been the fuel that enabled me to achieve this accomplishment.

To all of you, I say from the depths of my heart: Thank you.

Moussaoui Kaouthar

Dedication

I dedicate this work To my dear father, who has always been my main pillar of support with his constant concern and limitless encouragement. His presence by my side has given me the strength and ability to face all challenges.

Special thanks go to my beloved mother, whose tender support created an ideal environment at home, helping me to focus and strive. Her boundless support has had a profound impact on me.

I also express my gratitude to my dear uncle, *Anouar MOUSSAOUI*, who has provided me with unwavering financial and moral support. His significant role in my success is unforgettable, and I am deeply thankful to him from the bottom of my heart.

Additionally, I extend my heartfelt thanks to my cousin *Messaoud MOUSSAOUI* for his invaluable support and encouragement throughout my academic journey.

To all my friends who have always encouraged me, and to whom I wish more success.

Thank you!

Kaouthar MOUSSAOUI

Abstract

Detecting microorganisms in water is crucial for ensuring water quality and public health. This study employs deep learning techniques to identify and classify microorganisms in water samples. The methodology involves collecting dataset included EM images, pre-processing the data using resizing and normalization, and then applying data augmentation techniques such as horizontal and vertical flipping, random rotation, Gaussian blur, CLAHE (Contrast Limited Adaptive Histogram Equalization), and cutout.

The study utilizes pre-trained models such as ResNet50, VGG19, and EfficientNet-B0 for feature extraction and classification. These models are combined with Support Vector Machines (SVM) and use also Vision Transformers (ViT-B16) to enhance accuracy. The models are evaluated based on precision, recall, F1-score, and accuracy.

The results show that for the first problem, the best models are the pre-trained models alone with EfficientNet-B0 and ResNet50 with SVM, achieving 98% accuracy, recall, and precision. For the second problem, the best model is the pre-trained ResNet50 alone, achieving 96% recall, 96% precision, and 95% accuracy. After training and evaluation, the model is deployed in a real-world environment and integrated into a web application via the Gradio framework.

This advancement offers rapid and accurate detection of harmful microorganisms, improving public health outcomes.

Keywords: *Deep learning, Water quality, Microorganism detection, CNN, ViT, Waterborne diseases, Computer-aided classification.*

ملخص

يُعدُّ الكشف عن الكائنات الحية الدقيقة في الماء أمراً بالغ الأهمية لضمان جودة المياه والصحة العامة. تستخدم هذه الدراسة تقنيات التعلم العميق لتحديد وتصنيف الكائنات الحية الدقيقة في عينات المياه. تتضمن المنهجية جمع مجموعة بيانات تحتوي على صور مجهر إلكتروني (EM)، ومعالجة البيانات مسبقاً من خلال تغيير حجمها وتطبيعها، ثم تطبيق تقنيات زيادة البيانات مثل الانقلاب الأفقي والعمودي، والتدوير العشوائي، والتشويش الغاوسي، وتعديل التباين التكيفي المحدود بالهستوجرام، والقطع الجزئي.

تستخدم الدراسة نماذج مدربة مسبقاً مثل VGG19، ResNet50، و EfficientNet-B0 لاستخراج الميزات والتصنيف. يتم دمج هذه النماذج مع آلات الدعم المتجهية (SVM) والمحولات البصرية (ViT-B16) لتعزيز الدقة. تُقيم النماذج بناءً على الدقة، الاسترجاع، مقياس F1، والدقة.

تظهر النتائج أنه بالنسبة للمشكلة الأولى، أفضل النماذج هي النماذج المدربة مسبقاً بمفردها مع EfficientNet-B0 و ResNet50 مع SVM، حيث تحقق نسبة دقة، واسترجاع، صحة تصنيف تصل إلى 98%. أما بالنسبة للمشكلة الثانية، فإن أفضل نموذج هو ResNet50 المدرب مسبقاً بمفرده، حيث يحقق نسبة استرجاع ودقة تبلغ 96%، صحة تصنيف 95%. بعد التدريب والتقييم، يتم نشر النموذج في بيئة واقعية ويتم دمجه في تطبيق ويب عبر إطار عمل Gradio.

يوفر هذا التقدم كشافاً سريعاً ودقيقاً للكائنات الدقيقة الضارة، مما يحسن نتائج الصحة العامة.

الكلمات المفتاحية: التعلم العميق، جودة المياه، كشف الكائنات الدقيقة، الشبكات العصبية الالتفافية، المحولات البصرية، الأمراض المنقولة بالماء، التصنيف بمساعدة الكمبيوتر.

Contents

Abstract

List of Figures

List of Tables

General introduction	1
1 General overview of water	3
1.1 Introduction	3
1.2 Water cycle	3
1.2.1 Evaporation	4
1.2.2 Condensation	4
1.2.3 Precipitation	4
1.2.4 Runoff	5
1.3 Water resources	5
1.3.1 Groundwater	6
1.3.2 Surface water	6
1.3.3 Atmospheric water	6
1.4 Water value	6
1.4.1 Human survival	7
1.4.2 Agricultural	7
1.4.3 Climate regulation	7
1.4.4 Industrial	8
1.5 Water properties	8
1.5.1 Universal solvent	8
1.5.2 Density & buoyancy	9

1.5.3	Cohesion and adhesion	9
1.6	Water pollution	10
1.6.1	Chemical pollution	10
1.6.2	Physical pollution	10
1.6.3	Microbial pollution	10
1.7	Biological Pollutants	11
1.7.1	Pathogenic bacteria	11
1.7.2	Viruses	12
1.7.3	Parasites	13
1.7.4	Parasitic worms	15
1.8	Water-borne diseases	16
1.8.1	Cholera disease	16
1.8.2	Typhoid fever disease	17
1.8.3	Shigellosis disease	17
1.8.4	Hepatitis A virus (HAV) disease	18
1.8.5	Poliomyelitis disease	18
1.8.6	Amoebiasis disease	19
1.8.7	Giardiasis	19
1.8.8	Ascariasis disease	20
1.9	Conclusion	20
2	Deep learning on microorganisms detection	21
2.1	Introduction	21
2.2	Machine learning	22
2.3	Machine learning models	22
2.3.1	Supervised learning	22
2.3.2	Unsupervised learning	25
2.3.3	Semi-supervised learning	27
2.3.4	Reinforcement	28
2.4	Deep learning	29
2.4.1	Convolutional neural networks (CNN)	29
2.4.2	Recurrent neural networks (RNNs)	31
2.4.3	Generative adversarial networks (GANs)	35
2.4.4	Vision transformer (ViT)	38

2.4.5	Transfer learning (TL)	40
2.5	Conclusion	44
3	System Design	45
3.1	Introduction	45
3.2	Global architecture	45
3.3	Detailed architecture	46
3.3.1	Data gathering	46
3.3.2	Pre-processing	47
3.3.3	Splitting	47
3.3.4	Train	48
3.3.5	Test	52
3.3.6	Evaluation metrics	53
3.3.7	Model deployment	54
3.4	Conclusion	55
4	Implementation and results	56
4.1	Introduction	56
4.2	Implementation tools and languages	56
4.2.1	Python	56
4.2.2	Kaggle	57
4.2.3	opencv	58
4.2.4	Pytorch	59
4.2.5	Matplotlib	59
4.2.6	Gradio	60
4.2.7	NVIDIA	60
4.2.8	CUDA	61
4.2.9	CuDNN	61
4.2.10	NumPy	62
4.2.11	Scikit-learn	62
4.3	Realization	63
4.3.1	Dataset description	63
4.3.2	Preprocessing steps & Split Dataset	67
4.3.3	Data Augmentation	69
4.4	Modelisation	73

4.4.1	Pretrained Models	73
4.4.2	Pretrained & ML Models	76
4.4.3	ViT Model	79
4.4.4	Evaluating the Models	83
4.5	Results and discussion	86
4.5.1	Pretrained Models	86
4.5.2	Pretrained Models & ML Models	101
4.5.3	ViT Model	111
4.5.4	discussion	116
4.5.5	Model deployment	117
4.6	Conclusion	120
	General conclusion	121
	Bibliography	122

List of Figures

1.1	Water Cycle	5
2.1	Classification techniques	24
2.2	Learning methods	28
2.3	Basic CNN Architecture	30
2.4	FRNN	32
2.5	Recursive Neural Network	33
2.6	Hopfield Network	33
2.7	SRN	34
2.8	Echo State Network	34
2.9	LSTM	34
2.10	GRUs	35
2.11	Basic GAN architecture	36
2.12	Basic VIT architecture	40
2.13	VGG-19 Architecture	42
2.14	EfficientNet-B0 architecture	42
2.15	Resnet50 architecture	43
2.16	PIX2PIX architecture	44
3.1	General architecture	46
3.2	Pre-processing steps.	47
3.3	Data partitioning: 80% for training and 20% for testing/validation.	48
3.4	Training Phase Architecture	48
3.5	Data Augmentation Processes	50
3.6	Feature Extraction and Classification Architecture	51
3.7	Vision Transformer Architecture	52

3.8	testing Phase.	53
3.9	Model Deployment.	55
4.1	Python version code and output.	57
4.2	Kaggle Logo	58
4.3	OpenCv logo	58
4.4	Pytorch logo	59
4.5	Matplotlib logo.	60
4.6	Gradio logo.	60
4.7	NVIDIA logo	61
4.8	CUDA logo	61
4.9	CuDNN logo	62
4.10	Numpy logo	62
4.11	Sklearn logo	63
4.12	Pathogenicity Distribution	64
4.13	Data Structure, Pathogenic, and Non-Pathogenic.	65
4.14	Percentage distribution of groups	67
4.15	Python code snippet for importing libraries.	67
4.16	Python code snippet for resizing images.	68
4.17	Python code snippet for normalization.	68
4.18	Python code snippet for splitting dataset.	69
4.19	Results of preprocessing steps.	69
4.20	Python code snippet for horizontal flip.	70
4.21	Python code snippet for vertical flip.	70
4.22	Python code snippet for rotation.	70
4.23	Python code snippet for Gaussian blur.	70
4.24	Python code snippet for CLAHE.	71
4.25	Python code snippet for Cutout.	72
4.26	Original image and its augmented versions.	73
4.27	Setting up the model using EfficientNet-B0 with ImageNet weights.	74
4.28	Setting up the loss function and optimizer.	75
4.29	Setting up training loop variables.	75
4.30	Epoch loop for training and validation phases.	75
4.31	Batch loop for each phase within each epoch.	76

4.32	Calculating and recording metrics for each epoch phase.	76
4.33	Loading and setting up the pretrained EfficientNet-B0 model.	77
4.34	Extracting features from training and test sets.	77
4.35	Training and evaluating multiple SVM classifiers.	78
4.36	Saving the best SVM model.	79
4.37	Importing Libraries and Setting up the Device	79
4.38	Loading Pretrained ViT Model and Freezing Base Parameters	80
4.39	Modifying the Classifier Head	80
4.40	Displaying Model Summary	80
4.41	Setting Up Data Transforms and Loaders	81
4.42	Creating Dataloaders	81
4.43	Defining Training and Evaluation Functions	82
4.44	Training Loop	83
4.45	Creating Optimizer and Loss Function	83
4.46	Training the Model	83
4.47	Loading the best model from the training phase.	84
4.48	Plotting training and validation accuracy and loss.	84
4.49	Model evaluation on the validation set.	85
4.50	Generating the confusion matrix.	85
4.51	Visualizing the confusion matrix using a heatmap.	85
4.52	Printing the classification report.	86
4.53	Confusion matrix showing the performance of the EfficientNet-B0 model (2 classes).	87
4.54	Accuracy and Loss Curves During Training and Validation Phases of EfficientNet-B0 (2 classes).	88
4.55	Confusion matrix showing the performance of the EfficientNet-B0 model (40 classes).	89
4.56	Accuracy and Loss Curves During Training and Validation Phases of the EfficientNet-B0 Model (40 classes).	91
4.57	Confusion matrix showing the performance of the ResNet-50 model (2 classes).	92
4.58	Accuracy and Loss Curves During Training and Validation Phases of ResNet-50 (2 classes).	93
4.59	Confusion matrix showing the performance of the ResNet-50 model (40 classes).	94

4.60 Accuracy and Loss Curves During Training and Validation Phases of the ResNet-50 Model (40 classes).	96
4.61 Confusion matrix showing the performance of the VGG19 model (2 classes).	97
4.62 Accuracy and Loss Curves During Training and Validation Phases of VGG19 (2 classes).	98
4.63 Confusion matrix showing the performance of the VGG19 model (40 classes).	99
4.64 Accuracy and Loss Curves During Training and Validation Phases of the VGG19 Model (40 classes).	101
4.65 Confusion Matrix for EfficientNet-B0 + SVM (2 classes)	102
4.66 Confusion Matrix for EfficientNet-B0 + SVM (40 classes).	103
4.67 Confusion Matrix for ResNet50 + SVM (2 classes).	105
4.68 Confusion Matrix for ResNet50 + SVM (40 classes).	106
4.69 Confusion Matrix for VGG19 + SVM (2 classes).	108
4.70 Confusion Matrix for VGG19 + SVM (40 classes).	109
4.71 Confusion matrix showing the performance of the VIT model (2 classes).	112
4.72 Accuracy and Loss Curves During Training and Validation Phases of the VIT model (2 classes).	113
4.73 Confusion matrix showing the performance of the VIT model (40 classes).	114
4.74 Accuracy and Loss Curves During Training and Validation Phases of the VIT model (40 classes).	116
4.75 Image Classification using EfficientNet and Gradio Interface	118
4.76 select the classification type	119
4.77 Application example First problematic.	119
4.78 Application example Second problematic.	120

List of Tables

2.1	Classification of unsupervised clustering methods	26
4.1	Image Counts per Class in EMDS-7 (2 Classes)	64
4.2	Image Counts per Class in EMDS-7	65
4.3	Hyperparameters used for training the VGG19 model.	74
4.4	Classification Report EfficientNet-B0 (2 classes).	88
4.5	Classification Report Detailing Precision, Recall, F1-Score, and Support of the EfficientNet-B0 Model (40 classes)	91
4.6	Classification Report ResNet-50 (2 classes).	93
4.7	Classification Report Detailing Precision, Recall, F1-Score, and Support of the EfficientNet-B0 Model (40 classes)	96
4.8	Classification Report VGG19 (2 classes).	98
4.9	Classification Report Detailing Precision, Recall, F1-Score, and Support of the VGG19 Model (40 classes)	101
4.10	Classification Report for EfficientNet-B0 + SVM (2 classes).	102
4.11	Classification Report for EfficientNet-B0 + SVM (40 classes).	105
4.12	Classification Report for ResNet50 + SVM (2 classes).	106
4.13	Classification Report for for ResNet50 + SVM (40 classes).	108
4.14	Classification Report for VGG19 + SVM (2 classes).	109
4.15	Classification Report for VGG19 + SVM (40 classes).	111
4.16	Classification Report Detailing Precision, Recall, F1-Score, and Support of the VIT model (2 classes)	112
4.17	Classification Report Detailing Precision, Recall, F1-Score, and Support of the VIT model (40 classes)	116
4.18	Comparison of models on two problematics	117

General introduction

Water quality and safety are crucial priorities for public health and the environment. Microbial contamination of water poses significant risks to human health and marine life. Therefore, detecting these microorganisms in water is crucial for maintaining water quality and preventing waterborne diseases. With technological advancements, deep learning techniques have become powerful tools for analyzing environmental data and detecting biological pollutants.

Research problem and motivation

Classifying microorganisms in water is an intricate task due to their vast diversity in sizes, densities, shapes, and morphologies. This challenge is heightened when microorganisms display similar characteristics yet possess distinct attributes, complicating the determination of their pathogenicity or non-pathogenicity. Traditional detection techniques are not only costly, time-consuming, and labor-intensive but also often unavailable to people in developing countries, further exacerbating the issue.

Objectives of the study

The primary objective of this study is to develop an efficient model for classifying microorganisms that can be easily used by everyone. This goal will be achieved through several stages. First, we will define the scope and specifications of the domain. Next, we will fine-tune models to perform the required task. After that, we will compare the obtained results. Finally, we will create a web application to utilize the most effective model among the various trained models.

Structure of the memo

This memo comprises an introduction, four chapters, and a general conclusion, organized as follows:

- **Chapter 1: Overview of Water:** Provides an introduction to water, water cycle, water sources, water importance, water characteristics, water pollution, biological pollutants, and waterborne diseases.
- **Chapter 2: Deep Learning for Microorganism Detection:** This chapter gives an overview of deep learning and artificial intelligence models, including Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Generative Adversarial Networks (GANs), Transformer (ViT), and Transfer Learning (TL).
- **Chapter 3: System Design:** Describes the proposed system architecture, including data collection, preprocessing, data splitting, training, testing, and performance evaluation.
- **Chapter 4: Implementation and Results:** This chapter presents the tools necessary for implementing the solution, explains the code, and discusses the final results.
- **General Conclusion:** Summarizes the key findings and provides recommendations for future research.

This study aims to contribute to improving water quality and protecting public health by applying deep learning techniques to accurately and effectively detect and classify microorganisms.

Chapter 1

General overview of water

1.1 Introduction

Water is considered the most abundant chemical compound on Earth's surface, covering a significant proportion of 71%. This prevalence is due to its crucial role in supporting life and its diverse forms in nature, which arise from its unique properties. Despite its abundance, ensuring safe water usage remains a significant challenge, largely due to pollution. Many people suffer and even die from using contaminated water, as they struggle to determine its suitability, highlighting the urgent need for effective water safety detection methods.

1.2 Water cycle

The Water Cycle is an open system that converts stored energy locally as well as thermal and chemical energy inside the system into kinetic energy and heat through interactions with energy molecules, water, air, and soil/rock. Complex processes including evaporation, condensation, precipitation, and runoff are a part of the hydrological cycle, often known as the water cycle. view the diagram in figure 1.1. Any modification (building of reservoirs, land cultivation, urbanization, industrialization, and river channelization) disturbs energy inputs and system components, which has an impact on the output [1].

Even though deteriorated water may lose qualities like cleanliness, enthalpy, and potential energy attraction, most of it is eventually restored by natural mechanisms in the moisture cycle (water cycle). Human demands can be satisfied by renewable water energy through careful management of this cycle [2].

1.2.1 Evaporation

The transition of water from a liquid to a vapor occurs naturally. The majority of it occurs near the surface of the Earth, when sunlight heats liquid water and turns it into vapor. The amount of liquid water lost from a closed system or the movement of water vapor away from the Earth's surface in an open system can be used to calculate evaporation. In essence, it's about water vaporizing and dissipating into the atmosphere. The energy from the sun or the air is what drives this process, literally making the water disappear [1] [3].

1.2.2 Condensation

The transition of water from a vapor to a liquid form is known as condensation. On exposed surfaces like those found in plants, lakes, streams, rocks, snow, and glaciers, this happens. On mountain slopes, condensation is a significant source of moisture that contributes to precipitation. The properties of the local plant foliage have a significant impact on the amount of condensation [4].

Clouds are formed when condensation condenses, reflecting sunlight and trapping heat to beautify the sky and control temperature. Rain is another effect of it that is essential to weather patterns and climate balance. Furthermore, condensation releases heat, which affects atmospheric circulation, particularly in tropical regions [5].

1.2.3 Precipitation

The main force behind the natural water cycle is thought to be precipitation, which is the process of water dropping in either liquid or solid form on the surface of the Earth and in the oceans. Since precipitation is necessary for all other hydrological processes, including evaporation, surface runoff, and recharge, rainfall is traditionally acknowledged as the beginning of the water cycle. It is impossible to overestimate the role that rainfall plays in the natural water cycle [6].

An examination of atmospheric temperatures is one technique used to assess the quality of surface precipitation. They demonstrate how the presence of a warm layer on top of a cold layer below frequently results in freezing rain and snow pellets. Precipitation types are predicted by using predictors based on average layer temperature and depth. Using statistical analysis of data from meteorological stations, criteria were devised to differentiate between freezing rain, ice pellets, snow, and rain. Using temperature data from overhead weather measurements or weather forecast models, meteorologists can readily apply this strategy [7].

1.2.4 Runoff

Water droplets are moved from one place to another during the flow phase by gravity and the land's natural curves, preferring paths with few obstacles and dirt. When there are no obstructions, gravity pulls water droplets toward the center of the Earth. This gravitational force changes the terrain, which in turn modifies the path of water flow. Erosion and sedimentation are terms used to describe the processes by which water moves sediment in river systems. Through the removal of dirt or the creation of new routes, these occurrences actively shape the landscape. The water's velocity determines how effective these processes are; higher flows cause soil particles to be suspended, while lower flows cause silt to deposit. Furthermore, there is a direct relationship between the velocity of the water and the amount of silt in it [8].

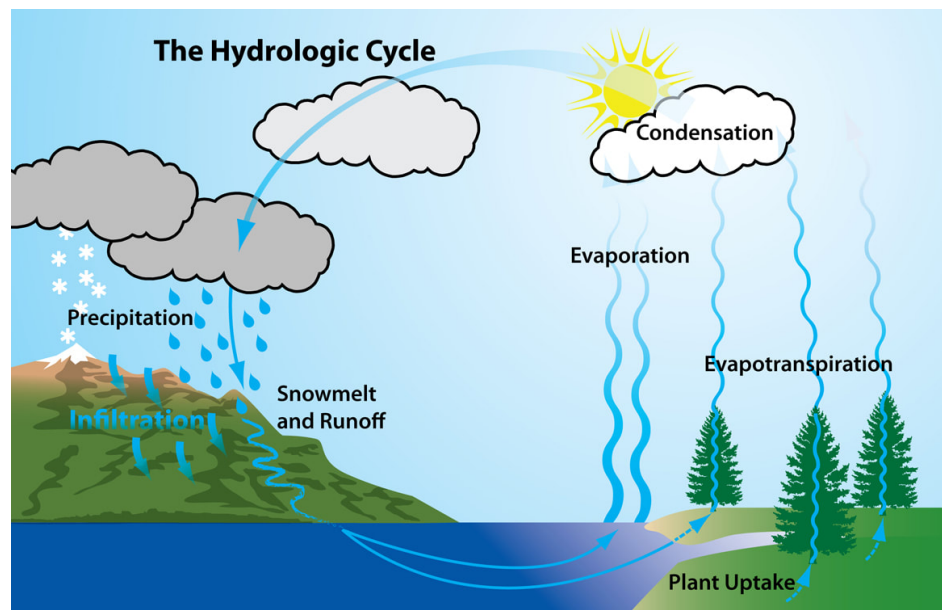


Figure 1.1: Water Cycle [9]

1.3 Water resources

There are two main categories in the field of water resources: conventional and non-conventional. Natural sources such as groundwater, atmospheric water, and surface water are considered conventional water resources. However, treated wastewater and desalinated water are examples of non-conventional water supplies.

In the past, nations mainly depended on traditional resources. Yet, due to contemporary

needs, planning and management of water resources now incorporates both conventional and non-conventional water resources [10].

1.3.1 Groundwater

One of the main sources of fresh water below the surface of the Earth is groundwater. It is characterized by porosity and permeability and is found in rocks and soil. Groundwater flows through porous rocks and sediments, while crystalline rocks and impermeable clay can act as barriers. The main direction of groundwater flow is from high-porosity to low-porosity regions. It is essential for sustaining environmental balance and protecting water supplies during dry spells. It is utilized for industry, agriculture, and human consumption [11].

Alluvial Aquifers, Glacial Terrains, Sandstone-Shale Aquifers, Carbonate Rocks, Volcanic Terrains, Crystalline Rocks, Coastal Aquifers, and Arid Regions are examples of groundwater settings [12].

1.3.2 Surface water

Surface water is that which accumulates naturally on the Earth's surface, such as lakes, rivers, ponds, and marshes. Surface water is affected by changes in precipitation, evaporation and discharge, and forms an essential part of the natural water cycle. Surface water is an important resource for human, animal and agricultural use, and constitutes a vital environment for many organisms and plants [13].

1.3.3 Atmospheric water

The water that exists in the Earth's atmosphere in a variety of forms, such as water vapor, clouds, fog, and precipitation, is referred to as the "water of atmosphere." Through processes including evaporation, condensation, and precipitation, this atmospheric water is essential to controlling Earth's climate and weather patterns [14].

1.4 Water value

For all living things to have access to enough food and a healthy environment, water and its resources are necessary. Freshwater is becoming more and more in demand worldwide due to population growth and economic expansion. Water scarcity not only endangers human

food supplies but also drastically lowers ecological biodiversity on land and in water. Vital water supplies are under rising pressure in many countries as a result of changing lifestyles, climate change consequences, and growing global population. As a result, the pressing need to conserve water is becoming more widely recognized [15].

Water is essential for preserving public health and quality of life, but its distribution is uneven around the globe, making it more difficult to access and utilize effectively [15].

1.4.1 Human survival

Water is essential to all of the bodily processes that take place in it. Since the body's cells make up its components, it plays a crucial function in the development and lubrication of human cushions and joints. Moreover, it helps nutrients to be transported anew as general physiological processes including digestion and blood pressure, as well as body temperature, are balanced and reregulated. Studies highlight the various roles that developing cells play, guaranteeing the discovery of the heart and its functions, and expanding our understanding and diversity of temperature regulation [16].

1.4.2 Agricultural

Agriculture is a vital component of both human life and the global economy, providing a significant portion of food and rural income. Globally, the amount of water used for irrigation has significantly grown, which has enhanced crop productivity and farmers' livelihoods. But there are drawbacks to this expansion as well: irrigation uses a lot of energy and leads to water leaks and soil salinization. Adopting techniques that improve water use efficiency and lessen adverse environmental effects is necessary for agriculture to become sustainable. This can be accomplished by implementing cutting-edge irrigation methods and better managing water resources [17].

1.4.3 Climate regulation

Built wetlands with a free water surface that operate as carbon sinks and have the capacity to absorb carbon from the atmosphere help regulate the climate. Additionally, they support the overall balance of greenhouse gases by boosting nitrous oxide emissions and decreasing methane emissions [18].

1.4.4 Industrial

The various industrial applications of water are described in this paragraph, with the main ones being cooling, transportation, washing, and solvent use. Due to cooling requirements, the production of thermal and atomic power is heavily consumed by industries such as chemical plants, metallurgy, pulp and paper, petroleum refining, and machinery manufacturing. Climate, water supply systems, and technology all affect how much water is used. Even though it makes up a small portion of the total intake, industrial water use might be important for some operations. Reducing withdrawals and pollution is necessary because rapid industrialization has resulted in untreated wastewater discharge contaminating water supplies. There has been a tendency since the 1970s toward stabilizing or reducing the demand for industrial water, and future advancements will probably concentrate on more effective water usage technology [19].

1.5 Water properties

Water's unique chemical makeup makes it essential for supporting life in all of its forms on Earth. Water is essential to many biological and industrial processes because of its unique physical and chemical characteristics [20]. The following are some of these attributes:

1.5.1 Universal solvent

Because it can sustain the complex chemical chemistry that gives rise to life, water is a popular solvent that is vital to maintaining life. Studies suggest that various organic solvents might have different chemistries. Because water is so abundant in the universe, it makes an excellent solvent, which is why biospheres were invented. High constants, scaffolding effects support, and structural stability across the temperature range that live systems are exposed to are requirements for a solvent that is able to support life. Although water provides a good example of these qualities, there are still potential to find substitutes. Aqueous molecules interact with one another through atom-dependent forces that can be broadly categorized as splice interactions and electrostatic interactions. For polar or ionic compounds, polar solvents like water work well; nonpolar solvents offer fewer possibilities. Furthermore, water facilitates cleavage interactions between nonpolar molecules, meaning that certain cleavage effects are required for any prospective solvent to support life. Water is the only solvent that successfully satisfies these conditions, which are necessary for the sustaining of life, but there

may be other solvents that are not as well understood [21].

1.5.2 Density & buoyancy

Understanding water behavior requires an understanding of water density (ρ), which is the ratio of water mass to volume. Freshwater has a density of about 1000 kg/m^3 at 4°C and atmospheric pressure, whereas seawater has a denser density of about 1027 kg/m^3 because of dissolved salts. With only slight changes in pressure and temperature, its density is comparatively stable. Water defies gravity due to its density, which makes Archimedes' principle apply. This is especially true in colder climates and ocean. Essentially, density and buoyancy are related: as a body submerges, water is displaced, producing buoyant force that, when combined with the body's weight, determines whether the object floats or sinks [22].

1.5.3 Cohesion and adhesion

A fundamental phenomena, water cohesion reflects its physical and chemical properties and is essential to many chemical and biological processes. This word describes how molecules of water interact with one another. In terms of water qualities, it means that molecules of water can stick together because of hydrogen bonds. Due to their polarity, which allows for hydrogen bond formation and mutual interaction, water molecules have little positive and negative charges close to their hydrogen atoms and modest negative charges near their oxygen atoms. Water's surface tension is a result of cohesiveness, which acts as a cohesive force and makes the surface behave like an elastic membrane. Certain species, including water insects, may move on the water's surface without sinking thanks to this characteristic. Furthermore, cohesiveness is essential for the movement of water through plant tissues, as water molecules stick together to form a continuous column that runs from the roots to the leaves. The attraction of molecules of various substances, on the other hand, is referred to as adhesion. In the case of water, this attraction happens when molecules of water are drawn to other surfaces, like the surfaces of plant vessels or fibers. Water can ascend or spread on surfaces thanks to adhesion, which is necessary for activities like plant water intake and other natural occurrences. In conclusion, adhesion and cohesiveness are complementary forces that influence critical biological processes and give water its special qualities [23].

1.6 Water pollution

Global water pollution is becoming a bigger issue, so strategies for managing water resources must be continuously assessed to meet these new demands. Approximately 14,000 individuals worldwide pass away from diseases and fatalities brought on by water contamination every day. Water pollution is a problem in both wealthy and developing nations. Water quality is influenced by a wide range of elements, including groundwater, precipitation, climate, soil type, vegetation, geology, flow conditions, and human activity. Although mining, urbanization, and agriculture all have an impact on water quality, point sources like towns and industries offer the most threat. Sediments, hazardous pollutants, and nutrients are examples of non-point source pollution [24].

Pollution manifests in three distinct types:

1.6.1 Chemical pollution

The term "chemical pollution" describes the presence of heavy metals, organic and inorganic pollutants, and chemical toxins in water. These pollutants can originate from man-made activities like leaking chemicals from various sources or from natural processes like the chemical breakdown of organic matter. In addition to endangering human health and welfare, this pollution prevents society and the economy from developing sustainably. In order to address the effects on the environment and ecology and guarantee that the public has access to clean drinking water, the current situation necessitates ongoing monitoring [25] [26].

1.6.2 Physical pollution

When organic and inorganic materials are suspended in water, they alter the water's color, flavor, and odor, leading to physical pollution. The rise in water temperature brought on by the cooling water leakage from nuclear reactors and companies into bodies of water is one instance of physical pollution. Because of the decrease in dissolved oxygen caused by this temperature rise, aquatic life may be harmed [25].

1.6.3 Microbial pollution

The term "microbial contamination" describes the introduction of potentially harmful microorganisms into water, soil, or the atmosphere, including bacteria, viruses, and protozoa.

Natural ecosystems are contaminated by this type of pollution, which originates from a variety of sources such as untreated sewage, agricultural runoff, and industrial discharge. Elevated levels of total aerobic bacteria and fecal coliforms, particularly *Escherichia coli*, have been reported in water bodies such as Golbasi Lake, indicating microbial pollution and possible health hazards [27].

1.7 Biological Pollutants

Biological pollutants primarily refer to invasive and non-native species of microorganisms in water samples. These pollutants can cause adverse effects at multiple levels of biological organization. At the ecosystem level, they can alter the flow of organic materials and energy, while at the habitat level, they can modify chemical and physical conditions. At the community level, biological pollutants can cause structural changes, such as the dominance, elimination, or replacement of native species by invasive species. At the population level, they can induce genetic changes, such as hybridization between invasive and native species, and at the individual organism level, they can lead to contamination by internal parasites or pathogens. Biological pollutants can result in economic losses, human health problems, and a decline in the authenticity of nature conservation areas. The concept of "biological pollutants" proposed by Elliott, is recognized in invasion biology. This concept is used to develop bio-pollution level assessments and standards for environmental status descriptors in European marine strategy frameworks [28]. Examples of biological pollutants include various types that can be classified as follows:

1.7.1 Pathogenic bacteria

Microorganisms known as pathogenic bacteria are responsible for infectious diseases in humans, animals, or plants. These microbes can cause illness or even death in their hosts. Important characteristics include their capacity to infiltrate and colonize host cells, create toxins that impair normal function, elude immune responses, and transfer across hosts through a variety of channels [29].

1.7.1.1 *Vibrio cholerae*

The Gram-negative bacterium *Vibrio cholerae* is the source of the serious gastrointestinal illness cholera. The bacteria is a well-known worldwide waterborne infection that is

mainly spread via sewage and tainted water. Cholera can be prevented and controlled by maintaining good personal hygiene and making sure food is safe to eat, which includes thoroughly boiling meat, drinking pasteurized milk, and using chlorinated water. The relevance of *Vibrio cholerae* is found in its capacity to induce severe diarrheal illness, which can result in significant morbidity and mortality, particularly in impoverished nations [30].

1.7.1.2 Salmonella

The gram-negative bacterium salmonella is the source of enterocolitis and gastritis. In order for Salmonella to be deadly, complex combinations of lethality factors that enable the bacterium to elude the host immune system must be coordinately expressed. Every species of Salmonella has the capacity to enter the host by triggering bacterial autophagy within intestinal epithelial cells. Furthermore, species linked to gastritis cause the intestines to become inflamed and secretory, whereas species that cause enteric fever cause a systemic infection by living and multiplying in mononuclear phagocytes [31]. Salmonella Infantis, Salmonella Newport, Salmonella Hadar, Salmonella Bareilly, Salmonella Nchanga, Salmonella Montevideo, Salmonella Tennessee, Salmonella Oranienburg, Salmonella Typhi, and Salmonella Paratyphi A are a few names for different serotypes of Salmonella [32].

1.7.1.3 Shigella

Shigella is a rod-shaped, gram-negative enteric bacteria that is spread through the fecal-oral pathway. It causes shigellosis, also known as bacillary dysentery, which is characterized by symptoms like fever, abdominal pain, and bloody diarrhea in humans. There are four primary species: *Shigella dysenteriae*, *Shigella sonnei*, *Shigella flexneri*, and *Shigella Boydii* [33].

1.7.2 Viruses

A class of diseases known as enteric viruses can be harmful to both human and animal health in aquatic environments. These viruses produce a wide range of illnesses and symptoms in humans and other animals because they are host-specific [34]. As an illustration:

1.7.2.1 Adenoviruses (AdVs)

AdVs, or adenoviruses, are members of the Adenoviridae family, which includes a range of illnesses that impact different kinds of animals. Six genes in this family can infect any

animal, including fish and humans. The seven species (HAdV A-G) and up to 113 recognized kinds of human adenoviruses (HAdVs) are grouped under the Mastadenovirus genus. AdVs are important in environmental and medicinal contexts [35]. They can result in a variety of illnesses in humans, ranging from minor respiratory infections to serious ailments. AdVs are also important environmental pathogens because they can contaminate water supplies and perhaps spread waterborne diseases [36].

1.7.2.2 Hepatitis A virus (HAV)

The fecal-oral pathway is the main means by which the hepatitis A virus (HAV) spreads, and tainted water is a major factor in this process. It may survive for several months in the environment and shows tolerance to certain situations like as low pH and high concentrations of chlorine. Elderly people, people with impaired immune systems, and people living in crowded environments are high-risk categories. Adults are more likely to develop serious disease from infections, whereas children may only show no symptoms [37].

1.7.2.3 F-specific (F+) RNA phages

Enteric viruses and fecal contamination in water are frequently detected using F-specific (F+) RNA phages as markers. The monitoring of microbiological sources is made possible by identifying particular subgroups of these phages. Temperature, pH, and the presence of organic materials are some of the variables that affect their survival and length of time in river water. The differential persistence patterns exhibited by different subgroups could skew assessments of their abundance in surface water. It's important to take water features into account when using F+ RNA phages for identifying microbial sources since differences in subgroup persistence can provide false information about their initial proportions. In conclusion, even though F+ RNA phages are useful markers, it is essential to comprehend the dynamics of their persistence in order to appropriately interpret the data [38].

1.7.3 Parasites

Parasites can be transmitted to humans through various means, including the direct consumption of contaminated water, causing 842,000 deaths annually [39]. They can find the following examples:

1.7.3.1 Giardia

When consumed through tainted food or water, the protozoan parasite *Giardia* can cause the gastrointestinal disease Giardiasis in both humans and animals. It is among the most prevalent parasites spread by water and can infect drinking water. Research indicates that traditional methods of treating water, such as filtration, sedimentation, and coagulation, are usually successful in eliminating *Giardia* from potable water. Advanced treatment techniques like UV disinfection and membrane filtration can also offer extra barriers to help remove and inactivate *Giardia*. *Giardia* removal methods for water are determined by a number of criteria, including raw water quality, treatment goals, and legal restrictions. To guarantee that the treatment procedures are successful in eliminating *Giardia* from the final drinking water supply, testing and monitoring are essential [40].

1.7.3.2 Cryptosporidium

The diarrheal disease known as "cryptosporidiosis" is brought on by the tiny protozoan parasite *Cryptosporidium*. Consuming food or water tainted with resistant cysts, or "oocysts," is how this parasite spreads. Severe diarrhea, vomiting, and abdominal pain can be symptoms of a *Cryptosporidium* infection, particularly in young children and immunocompromised people. Rarely, it can result in fatalities or very serious complications such as acute dehydration. The basic course of treatment for a *Cryptosporidium* infection is to replace lost fluids and electrolytes using intravenous and dietary fluids; however, there are no proven treatments for this infection. While some treatments can be used to treat symptoms, they don't really work to get rid of the parasite. By enhancing personal cleanliness habits and water quality, one can prevent illness better than treating it [41].

1.7.3.3 Cyclospora

is a parasite that can lead to cyclosporiasis, an intestinal ailment. It is a parasite with a single cell that can only reproduce in human hosts. Consuming food or water tainted with *Cyclospora* oocysts causes infection. Watery diarrhea, appetite loss, significant weight loss, exhaustion, and other flu-like symptoms are among the symptoms. If treatment is not received, the sickness may linger for several weeks, a month, or longer. People with impaired immune systems, such as those living with HIV/AIDS, may be more vulnerable to more serious or protracted illnesses. Antibiotics, such as trimethoprim-sulfamethoxazole, are commonly used to treat cyclosporiasis. As part of supportive treatment, it's crucial to stay hydrated

[42].

1.7.4 Parasitic worms

Many low-income communities and developing countries face serious health difficulties due to parasitic worm or helminth infections. Prior studies have detected the existence of parasitic worms in water sources in different nations. Humans who contract these infections may experience a variety of health problems, such as diarrhea, vomiting, nausea, skin rashes, dry coughs, and swelling and pain in the abdomen [43]. The following helminths are some of the most common ones found in water sources:

1.7.4.1 Ascaris lumbricoides

The huge parasitic roundworm *Ascaris lumbricoides* is a prominent cause of ascariasis, a parasitic illness that is common throughout the world. Symptoms of ascariasis include vomiting, diarrhea, congestion in the nose, and abdominal pain. Severe cases may result in biliary or intestinal obstruction, which could cause problems for the liver or lungs, among other organs. Usually, anti-worm drugs such as mebendazole and albendazole are used in treatment. Surgical intervention may be required in difficult cases. Promoting good personal hygiene, sanitation, and waste disposal techniques are among the prevention tactics; these are especially important for the health and wellbeing of children [44].

1.7.4.2 Strongyloides stercoralis

Strongyloides stercoralis, also referred to as the little intestinal worm, is a parasitic nematode that can cause serious health problems if it becomes infected. Strongyloidiasis is an illness that can be fatal, especially in people with impaired immune systems. Diarrhea, abdominal pain, rash, and inflammation of the lungs are among the symptoms. Antiparasitic drugs such as albendazole or ivermectin are commonly used in treatment, with an emphasis on the significance of prompt action to avoid serious consequences. For early detection and management, high-risk populations must undergo routine screening and treatment. Prevention is key to reducing the chance of contracting this parasite, including good personal hygiene, secure waste disposal, and sufficient water disinfection [45].

1.7.4.3 Trichuris trichiura

Humans harbor the parasitic worm *Trichuris trichiura* in their large intestines, which is spread by consuming tainted food or drink. Diarrhea, abdominal pain, anemia, malnutrition, distension in the abdomen, weight loss, and decreased productivity can all result from this infection. Anti-worm drugs, such as mebendazole or albendazole, are used as part of the treatment under a doctor's supervision. Strict attention to the specified treatment plan, taking medication with food, getting enough fluids, and seeing a doctor right away if symptoms worsen or don't go away are all advised in order to reduce complications and side effects [46].

1.8 Water-borne diseases

A variety of disorders caused by bacterial, viral, or parasite pathogens found in contaminated water sources are referred to as waterborne diseases. Water that has been contaminated is frequently the transit medium. A description of the pathogens that cause diseases that are transmitted by water is provided in Section 1.7 **Types of biological pollutants**. When they spread quickly through contaminated water, diseases might come out. While bacterial indicators are frequently employed in the evaluation of water quality, virus contamination may not always be reflected by them. Studies have shown that disease outbreaks can originate from water that satisfies local quality standards. The Centers for Disease Control (CDC) gather information and publish epidemiological surveillance reports on a regular basis to supervise the tracking and monitoring of outbreaks of waterborne diseases. However, a large number of outbreaks go unreported, which results in insufficient surveillance [47].

It is crucial to recognize that notable waterborne diseases include:

1.8.1 Cholera disease

The World Health Organization estimates that cholera, a waterborne illness, causes between three and five million cases and more than 100,000 deaths yearly, making it a serious worldwide health concern. Current cholera outbreaks in Haiti, Zimbabwe, Angola, and South Africa highlight the critical need for a more thorough understanding of these and related aquatic illnesses. A popular model for studying disease dynamics is the Susceptible-Infected-Recovered (SIR) model, in which water sources provide the means of transmission for cholera. The length of time the pathogen remains in water determines how long illnesses last. Taking into account both direct and indirect transmission pathways, the Susceptible-

Infected-Water-Recovered (SIWR) model offers important insights into these dynamics and aids in directing public health initiatives. Studies show that several transmission channels are important for disease localization, and the SIWR model helps to make sense of these pathways. Although the model is fundamentally recognizable, determining useful parameters from real-world data may provide difficulties. Thus, adding more data regarding illness distribution and water quality over time can improve the model's identifiability and increase its usefulness for managing and preventing disease [48].

1.8.2 Typhoid fever disease

The *Salmonella typhi* bacteria, which causes typhoid fever, normally takes one to three weeks to mature. Mild exhaustion, appetite loss, and mild muscle aches are among the early symptoms. Severe symptoms including chills, coated tongue, nosebleeds, coughing, sleeplessness, nausea, and diarrhea develop later. A defining feature of the illness is severe fever, which frequently reaches 105°F (40.6°C). At their worst, three weeks after incubation, patients may have blood-tinged stools, psychosis, and emaciation; one in five also develop gastrointestinal hemorrhage. Because of the high fever, neuropsychiatric abnormalities such as hallucinations, aberrant behavior, tremors in the nervous system, and enlargement of the brain are common. Ninety to ninety-five percent of people who contract typhoid disease survive. The effects of typhoid disease go beyond direct mortality since survivors are more likely to die from other causes, especially heart failure and tuberculosis. Typhoid-affected pregnant women are more likely to miscarry and give birth before term. Typhoid disease increases morbidity in young children and can affect cognitive performance in adulthood. Low death rates from typhoid indicate pure water, while high rates indicate contamination. Typhoid acts as an indicator of water quality. Typhoid rates are significantly decreased by chlorinating and filtering water; data indicates that large reductions in mortality occur when purification technologies are used. The benefits of water purification efforts for public health are further highlighted by the fact that these improvements in water quality also lower mortality from other diseases [49].

1.8.3 Shigellosis disease

The severe diarrheal illness known as shigellosis, which is primarily responsible for millions of illnesses and thousands of fatalities each year in Asia, is brought on by bacteria belonging to the genus *Shigella*. Despite its seriousness and widespread effects, there is cur-

rently no approved vaccination against *Shigella*, and the underlying mechanisms of protection are still unknown. The requirement for a multivalent vaccination to protect against the various serotypes of *Shigella* complicates vaccine development even more. There are four species in the genus *Shigella*; *Shigella dysenteriae* was formerly the cause of significant outbreaks but is now quite uncommon. Rarely, *S. boydii* is also isolated. *S. flexneri* is widespread throughout the world and comes in several serotypes, especially in nations with little resources. On the other hand, conversely, *S. sonnei*, which has a single serotype, is more common in high-income areas, though it's unknown why it's so dominant there. The rising percentage of *S. sonnei*-caused shigellosis cases in developed nations is correlated with rising economic growth. As a result, *S. flexneri* has decreased proportionately while *S. sonnei* has emerged in quickly developing nations. The precise causes of this change are yet unknown, though, and *S. sonnei*'s increasing antibiotic resistance has alarming consequences for world public health [50].

1.8.4 Hepatitis A virus (HAV) disease

Hepatitis A virus (HAV) disease, also known as HAV, is a serious global health risk due to its waterborne nature. Although direct contact between individuals remains the most common mode of transmission, outbreaks of HAV associated with drinking contaminated water have been reported in many different countries. Understanding the environmental behavior of HAV only became possible with the development of cultivation and enumeration techniques in tissue culture. HAV has been reported in concentrated wastewater, natural water sources, and often observed post-outbreak. It shows remarkable resilience, lasting at least a month at room temperature and months in colder climates. Factors affecting its survival include humidity, pH, temperature, salt concentration, microbial activity, and parallels with other enteric viruses. To completely remove HAV from water, effective removal techniques such as coagulation, high-rate filtration, and disinfection are essential [51].

1.8.5 Poliomyelitis disease

Poliomyelitis, commonly referred to as polio, is an infectious viral disease that mainly affects the neurological system. Even while the theory connecting polio to water is not new and has been debated by numerous organizations for about 50 years, it did not get popular acceptance until 1937, when it attracted a large number of supporters throughout the preceding five years. It's unknown if the poliovirus can spread by food and drink, contaminated

objects, or airborne particles, even though laboratory tests suggest that it enters the body through the digestive system. According to studies, the poliovirus can be found in the feces of infected people, even those who don't exhibit any symptoms, and human feces are thought to be a fertile reservoir of the virus. Moreover, the virus has been found in sewage water samples, indicating the possibility of feces contaminated with viruses getting into sewage systems. But just because a virus is found in sewage water doesn't indicate that people can contract it by drinking it. It is currently unknown how long the poliovirus may stay in water and how contagious it is, despite laboratory tests suggesting that it can persist for a while outside of the human or animal body under certain circumstances. The idea that polio may spread through water is supported by some evidence, although it is inconsistent with the epidemiological pattern of the disease. The idea that water is a major factor in the spread of the poliovirus is not supported by enough evidence, in contrast to other waterborne illnesses like cholera and typhoid [52].

1.8.6 Amoebiasis disease

Entamoeba histolytica in particular is the cause of the serious bowel illness known as amoebic colitis, sometimes known as amoebiasis. The parasitic cysts that are present in food and water might spread infection when consumed. Clinical signs include ulcers forming in the large intestine, skin ulceration in the perianal region, fever, diarrhea, dysentery, and stomach pain. Metronidazole, secnidazole, and ornidazole are just a few of the drugs used to treat the condition, which can be identified through laboratory testing on stool samples. As of right now, there is no vaccine to prevent amoebiasis, but it can be managed with good personal hygiene, sanitation, and public awareness campaigns about the disease's causes [53].

1.8.7 Giardiasis

Among parasite disorders, gastrointestinal infections like giardiasis are common around the world. These epidemics are mostly caused by *Giardia*, of which 60 percent of cases are waterborne. Fecal matter deposition from humans and animals is the cause of this parasite's presence in water; contamination levels can range from 0.3 to 100 cysts per 100 liters of surface water. *Giardia* cysts in treated drinking water must be eliminated in surface water by at least 99.9 percent according to regulatory requirements. The maximum daily infection rates for systems using contaminated water, however, are five times greater than for those using clean sources. The attack rate of *Giardia* in outbreaks linked to untreated surface

water can range from 0.5 to 16 percent, contingent on the degree of cyst contamination. Even while Giardia infections are common and asymptomatic, eating as few as 10 cysts might cause disease. In order to guarantee adequate protection against waterborne illnesses, risk assessment models are essential for assessing the relevance of Giardia contamination in water and directing public health initiatives [54].

1.8.8 Ascariasis disease

Ascariasis is a frequent health issue that mostly affects youngsters in warm, semi-tropical climates. The helminthic sickness is caused by eating the eggs of *Ascaris lumbricoides* roundworms, which can contaminate food, water, or soil. Following ingestion, the eggs hatch into adult worms in the intestines, which can cause a variety of clinical symptoms, including fever, vomiting, coughing, wheezing, appetite loss, shortness of breath, stomach pain, malnutrition, and stunted growth, particularly in children. Clinical signs and symptoms, blood and stool testing, and chemical medicines are used in the diagnosis and treatment processes. It is crucial to adhere to preventive and control measures, such as drinking cleaned water, avoiding contact with polluted soil, and enhancing sanitary systems, in order to prevent this neglected disease [55].

1.9 Conclusion

Ultimately, water is a natural resource of incomparable beauty and importance. Its harmonious cycle, many forms, and unique properties make it an essential element for life on Earth. However, pollution is seriously threatening this vital resource, exposing humanity to serious water-borne diseases.

In the face of these challenges, it is everyone's duty to preserve the purity of water and ensure healthy access to this substance that surrounds us with so much grace. Through concerted monitoring, treatment and awareness efforts, we can hope to conserve this precious resource and ensure a bright future for future generations. Water, in its magnificence, deserves our deepest respect and greatest vigilance.

Chapter 2

Deep learning on microorganisms detection

2.1 Introduction

The study of microorganisms is crucial for scientists in various fields such as clinical microbiology, agriculture, medical science, and food production due to their importance. Observing microorganisms under a microscope and using cultivation techniques is necessary to understand their biological, genetic, and physiological properties. However, traditional methods are labor-intensive and expensive. Microorganisms can have similar morphological features, making classification challenging. To streamline the classification process, a machine learning (ML) powered recognition tool can be developed to reduce the need for extensive human intervention [56].

The dissertation's Chapter 2 provides a thorough summary of the various machine learning models. The first section of the chapter defines machine learning and how it can help computers learn and adapt without the need for explicit programming. After that, it explores the different kinds of learning models, such as supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. The chapter concludes with a summary of deep learning that covers its definition, features, and several varieties of deep learning networks.

This chapter provides a solid basis for comprehending the various artificial intelligence methodologies.

2.2 Machine learning

Within computer science, machine learning is a specialized discipline that focuses on creating algorithms and methods to automatically solve complicated problems that are challenging to program using conventional approaches. Machine Learning algorithms use a set of labeled data and analyze it to create a model or set of rules that can predict outcomes for fresh data, rather than depending on the creation of precise designs beforehand. Using labeled data as a source, this learning model is used to forecast future outcomes. This method, called supervised learning, works better than conventional techniques in terms of accuracy and efficiency and may be used to a wide range of difficult tasks. The difficulty, though is in understanding how issues are resolved, particularly in neural network-based algorithms [57].

Furthermore, there are several formal definitions of machine learning in the literature. "A field of study that gives computers the ability to learn without being explicitly programmed" is how Arthur Samuel described machine learning in his groundbreaking work. "A computer program is said to learn from experience (E) with respect to some class of tasks (T) and performance measure (P), if its performance at tasks in T, as measured by P, improves with experience E" according to Tom Mitchell's explanation in computer science terminology. Machine learning is described as "programming computers to optimize a performance criterion using example data or past experience" by Ethem Alpaydin in his book. The concept of teaching computers to do things beyond simple computations intelligently is shared by these several definitions [58].

2.3 Machine learning models

The most prominent learning models along with their various algorithms or applications that will be considered are illustrated in Figure 2.2 and explained below:

2.3.1 Supervised learning

In supervised learning, a knowledge base is built using preclassified patterns, which support the classification of new patterns. The primary task of this type of learning is to link input features to an output defined as a class. The result of this learning is to create a model that can be used to classify unseen instances correctly. In general, the model is represented as a function $f(x)$, where x represents the input patterns and y represents the resulting class. The preclassified patterns are referred to as the "training set" which consists

of pairs of input and output, while the unseen patterns are referred to as the "test set" which contains only input patterns. Where $DS = \{ \langle X_1, y_1 \rangle, \langle X_2, y_2 \rangle, \dots, \langle X_n, y_n \rangle \}$, where n is the number of patterns or observations and p represents the number of classes. The general algorithm for supervised learning is presented as shown in Algorithm 1 below [59]. There are several proposed supervised learning algorithms, which we can classify into two categories:

Algorithm 1 Generic Supervised Learning [59]

Require: N training examples with labels dataset: $\{X - Y\}$, $\{ \langle x_1, y_1 \rangle, \langle x_2, y_2 \rangle, \dots, \langle x_n, y_p \rangle \}$, $k \pm 10$; //cross validation

Ensure: M - training model based on probabilistic approach

- 1: $i \leftarrow 0$
 - 2: **for** each i in k **do**
 - 3: $dataset_samples \leftarrow dataset/k$ // training dataset
 - 4: $dataset_samples[i]$
 - 5: $M_i \leftarrow Classifier(training)$
 - 6: $M \leftarrow M_i$
 - 7: **end for**
 - 8: **return** M
-

2.3.1.1 Classification

Classification is a fundamental method in machine learning and data mining, used to predict the group membership of data instances. Despite the availability of various techniques in these fields, classification remains the most widely used approach, given its vital role in future planning and knowledge discovery. Researchers in the fields of machine learning and data mining extensively explore the details of classification, considering it a thoroughly studied problem. Figure 2.1 illustrates a comprehensive model of supervised learning, focusing on various classification techniques employed. However, despite its prevalence, classification faces challenges, especially in dealing with missing data. The absence of values within datasets poses challenges during both the training and classification phases. The reasons for missing data vary, ranging from misunderstandings leading to the non-entry of records, to data being deemed invalid at the time of entry, and even data removal due to inconsistencies with other documented data or equipment malfunction [60].

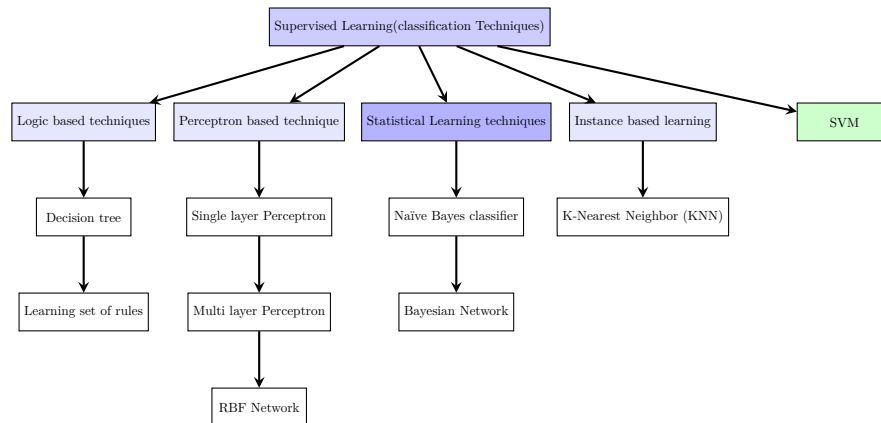


Figure 2.1: Classification techniques

2.3.1.2 Regression

Regression analysis is a technique used for two purposes. Firstly, regression analyses are commonly employed for forecasting and prediction, where their applications heavily intersect with the field of machine learning. Secondly, regression analysis can be used in some cases to determine the causal relationships between independent and dependent variables [61]. And this is the 3 main types of regression:

- **Simple Linear Regression:** This model involves a single independent variable predicting the dependent variable. The relationship is defined by the equation $y = \beta_0 + \beta_1x + \epsilon$, where β_0 and β_1 are coefficients and ϵ represents the error term [61].

- **Multiple Linear Regression (MLR):** MLR uses multiple independent variables to predict the dependent variable. The model aims to establish a linear relationship between the independent variables x and the dependent variable y [61], expressed as:

$$y = \beta_0 + \beta_1x_1 + \dots + \beta_mx_m + \epsilon[61] \quad (2.1)$$

- **Polynomial Regression:** Polynomial regression involves modeling the relationship between the independent and dependent variables using a polynomial equation of n th degree. It extends beyond linear relationships to capture curvilinear interactions. The model is expressed as:

$$y = \beta_0 + \beta_1x + \beta_2x^2 + \dots + \beta_hx^h + \epsilon[61] \quad (2.2)$$

where h is the polynomial degree [61].

2.3.2 Unsupervised learning

Unsupervised learning algorithms rely on unlabeled data to build models. These models aim to extract the underlying relationships inherent in the structure of the features themselves, with examples including clustering and principal component analysis [62]. The general algorithm for unsupervised learning is presented as shown in Algorithm 2 below. There are several proposed unsupervised learning algorithms, which we can classify into four categories:

Algorithm 2 Generic Unsupervised Learning [59]

Input: N training examples without labels dataset: $\{X \rightarrow?\}$

2: Output: M_c - returns model with k number of clusters and center of each cluster

$k \leftarrow 5$ // number of clusters

4: $cv \leftarrow 10$ // cross-validation

$i \leftarrow 0$

6: // iterate cv times

for i in cv **do**

8: // iterate over cv times

$dataset_samples \leftarrow dataset/k$

10: $training_i \leftarrow dataset \setminus dataset_samples[i]$

$M_i \leftarrow \text{Cluster}(training_i)$

12: $c \leftarrow c_i$

$M \leftarrow M_i$

14: **end for**

2.3.2.1 Clustering

Clustering is an exploratory technique aimed at identifying groups or clusters of high density, where observations within each cluster are more similar to each other than to observations in different clusters. This process relies on quantifying the degree of similarity or dissimilarity between observations, and the results of the analysis are greatly influenced by the type of similarity metric used [63].

In the realm of deep learning applications, several clustering algorithms stand out for their suitability and effectiveness. Among the most notable are those listed in the table below:

Class	Methods
Distance measure based	K-means clustering Hierarchical clustering Fuzzy clustering Support vector machines Spectral clustering Decision trees
Statistical	Expectation maximization algorithm
Neural networks	Self organizing maps (Kohonen networks) Adaptive resonance theory Autoencoders Co-localization Generative models

Table 2.1: Classification of unsupervised clustering methods [63]

2.3.2.2 Association rule mining

After clustering, association rules mining is a crucial unsupervised data mining technique that uncovers intriguing associations (dependencies, links) in sizable data sets. The data is kept in the form of transactions, which might come from relational databases or data warehouses or be created by an outside process. Association rules are a crucial data mining tool for knowledge extraction from data because of their strong scalability features and the constantly expanding volume of gathered data. Finding intriguing correlations opens up a supply of data that organizations frequently use to inform their decisions. Market-basket data analysis, cross-marketing, catalog design, loss-leader analysis, clustering, data preprocessing, genomics, and other fields are some of the application areas of association rules [64].

2.3.2.3 Dimensionality Reduction

In order to improve computing efficiency and analysis accuracy, dimensionality reduction (DR) techniques are applied as a preprocessing step for data, identifying a low-dimensional version of the original data. Finding a lower-dimensional representation that retains as much of the original data's substance as feasible is the mathematical definition of the dimension reduction problem. Based on the learning process, DR approaches are divided into supervised and unsupervised categories. In order to learn the lower-dimensional representation and predict class labels on unknown data, supervised algorithms make use of a training set. The Linear Discriminant Analysis (LDA), Maximum Margin Criterion (MMC),

and Orthogonal Centroid (OC) method are a few instances of supervised techniques. Singular Value Decomposition (SVD) and other unsupervised techniques project the original data into a new lower-dimensional space without the need for label information. DR approaches work by either picking a subset of the current features or by changing the existing features into a new reduced set of features. Through the use of linear or nonlinear combinations of vector coordinates in the original dimensions, feature transformation approaches seek to reduce the dimensionality of data to a minimal number of dimensions. It is thought that these methods are effective in revealing latent structures within datasets. Principal component analysis (PCA), independent component analysis (ICA), projection pursuit, and factor analysis are a few examples of feature transformation methods. Techniques for unsupervised feature selection are far more difficult than those under supervision. Techniques for data reduction using linear algebra are predicated on projections, in which the data matrix is transformed or multiplied to decrease the dimensionality of the matrix. SVD, ICA, and Non-negative Matrix Factorization (NMF) are a few of these methods. The focus of the literature is on text retrieval applications of DR, including PCA, ICA, Random Mapping (RM), and FastMap [65].

2.3.2.4 Anomaly detection

An anomaly is an occurrence that results in a percentage of network traffic deviating from what is considered normal or typical. These instances eloquently demonstrate the necessity of detecting such incidents [66]. Unsupervised anomaly detection algorithms can be applied if the dataset has anomalies and abnormal data together with no labels at the same time. The key takeaway here is to just employ intrinsic information, such a dataset's density estimate. The density of the region each instance dwells in is then used to assign a score. The most adaptable approach is unsupervised anomaly detection, particularly in real-world scenarios when data has been gathered and needs to be examined without any further information. However, it is very dependent on the input data. Success requires appropriate preprocessing and the creation of data views [67].

2.3.3 Semi-supervised learning

Semi-supervised learning bridges the gap between supervised and unsupervised learning, offering a balanced approach that harnesses the strengths of both methods. By leveraging both labeled and unlabeled data, semi-supervised learning aims to outperform purely super-

vised methods in terms of learning performance. This approach is particularly valuable in fields where collecting labeled data is challenging or costly, whereas unlabeled data is abundant and easily accessible. By employing semi-supervised learning algorithms, it becomes possible to achieve better or comparable results with fewer labeled instances, thereby reducing the expenses and efforts associated with data annotation. Moreover, semi-supervised learning provides a computational model that emulates certain aspects of human learning, illuminating how humans learn from a combination of labeled and unlabeled information [68].

2.3.4 Reinforcement

Reinforcement learning is the process of figuring out how to optimize a numerical reward signal by mapping situations to actions. Unlike most forms of machine learning, the learner is not informed which actions to do. Instead, they must experiment to determine which activities provide the greatest reward. The most intriguing and difficult situations are those in which decisions can impact not just the immediate reward but also the circumstance that arises later and all benefits that follow. The two key aspects of reinforcement learning that set it apart are trial-and-error search and delayed reward [69].

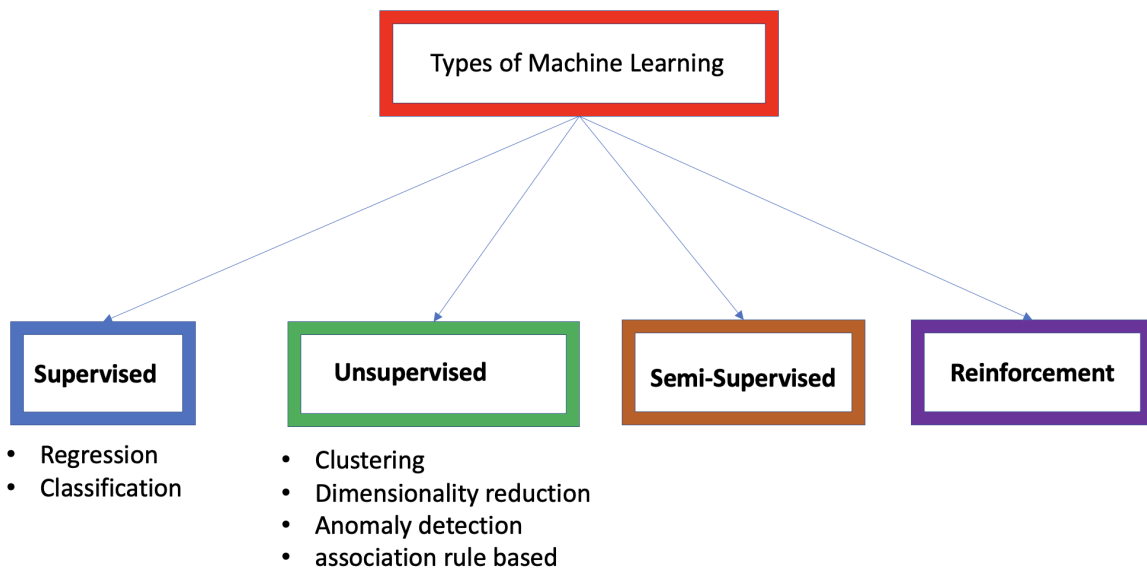


Figure 2.2: Learning methods [70]

2.4 Deep learning

Deep learning (DL), a specialized subset of machine learning (ML), excels in adaptability and learning capacity by structuring concepts hierarchically. In this structure, simple concepts form the foundation for more abstract representations. The architecture of DL involves multiple hidden layers, where categories are progressively learned. Each node in the network represents a part of the whole system, and together they form a complete picture. These nodes have weights that indicate the strength of their connections to the output, which are adjusted as the model learns. The widespread popularity and effectiveness of DL are largely due to its ability to utilize large amounts of data. The rise of big data has opened up new opportunities for DL advancements. Andrew Ng famously compared DL models to rocket engines, with vast amounts of data serving as the fuel that powers these algorithms [71]. In the next section, we will explore the different types of deep learning models:

2.4.1 Convolutional neural networks (CNN)

Convolutional Neural Networks (CNN) have emerged as one of the most intriguing methods recently, and they have played a pivotal role in a number of successful and difficult machine learning applications. Convolutional neural networks (CNNs), a type of deep artificial neural network, are utilized in numerous applications requiring visual data [72].

2.4.1.1 Overall architecture

Convolutional neural networks (CNNs) are composed of three main types of layers: convolutional layers, fully-connected layers, and pooling layers. Stacking these layers results in a CNN architecture, as seen in Figure 2.3.

1. **Convolutional Layer:** This layer uses the scalar product of the weights of the neurons connected to particular sections of the input and the region connected to the input volume to calculate the output of those neurons. The output of the activation created by the preceding layer is subjected to an elementwise activation function by the Rectified Linear Unit, or ReLU for short.
2. **Pooling Layer:** This layer further reduces the number of parameters inside that activation by performing downsampling along the spatial dimensions of the provided input.

3. **Fully-Connected Layers:** These layers try to generate class scores from the activations to be utilized for classification, carrying out the same tasks as ordinary ANNs. To enhance performance in between these layers, ReLU can also be applied.

CNNs use convolutional and upsampling techniques to process the original input through the previously mentioned successive layers and produce class scores for tasks involving regression and classification [73].

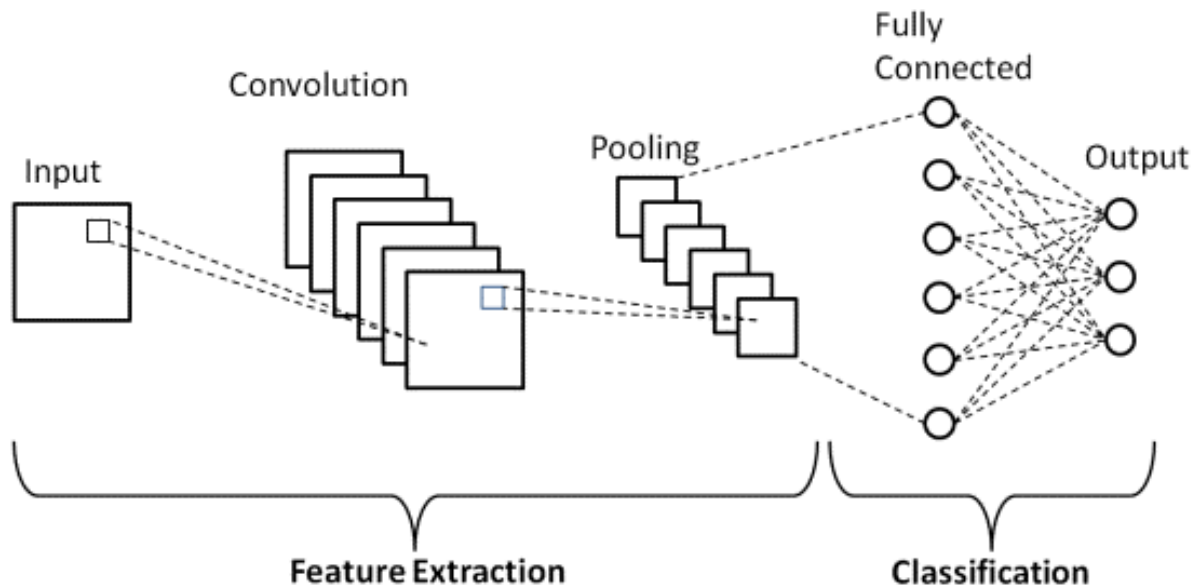


Figure 2.3: Basic CNN Architecture [74]

2.4.1.2 Evolution of Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are modeled after the architecture of the human cerebral cortex, which comprises cells sensitive to distinct regions of the visual field. These networks are composed of layers ordered according to their functions. CNNs are said to have their origins in the "Neocognitron" a device that Fukushima unveiled in 1980 and which transformed images by using local connections between neurons [75]. Subsequently, significant prominent constructions have been listed chronologically with increased accuracy. These include:

1. **LeNet:** Yann LeCun created LeNet-5, which was mostly used for character recognition jobs like reading numbers and zip codes. LeCun also presented the MNIST database, a common reference point for digit identification.

2. **AlexNet**: The ImageNet ILSVRC challenge was won in 2012 by AlexNet, a system developed by Alex Krizhevsky et al., with top-1 and top-5 error rates of 37.5% and 17.0%, respectively. It popularized CNNs in computer vision and is composed of five convolutional layers followed by three fully connected layers.
3. **ZFNet**: A proposal by Rob Fergus and Matthew Zeiler, took home the 2013 ILSVRC title. By modifying the hyperparameters, it outperformed AlexNet, especially by enlarging the middle convolutional layers and decreasing the stride and filter size of the first layer.
4. **VGGNet**: With 19 layers, VGGNet—the runner-up in the ILSVRC 2014—from the Oxford VGG group proved that performance increases with network depth. Nevertheless, its high processing demands render it ineffective for implementation on low-end GPUs.
5. **GoogLeNet**: Developed at Google by Szegedy et al., it was awarded the 2014 ILSVRC. With the addition of the inception module, which approximates a sparse CNN, the number of parameters was greatly decreased. Additionally, it increased computational efficiency and achieved 93.3% top-5 accuracy on ImageNet by substituting global average pooling for fully connected layers.
6. **ResNet**: A 152-layer network with batch normalization and skip connections, designed by Kaiming He et al., took first place in the 2015 ILSVRC. With a classification layer and global average pooling, it achieves 95.51% top-5 accuracy and is computationally more economical than VGGNet.
7. **DenseNet**: Published by Gao Huang et al. and recognized as the best article at CVPR 2017, establishes a feed-forward connection between every layer and every other layer. On benchmark tasks like CIFAR-10, CIFAR-100, SVHN, and ImageNet, it demonstrated notable gains.

These networks are strong instruments in the field of computer vision, exhibiting notable improvements in accuracy and performance over time [76].

2.4.2 Recurrent neural networks (RNNs)

Recurrent Neural Networks (RNNs) belong to a category of supervised machine learning models comprising artificial neurons featuring one or more feedback loops [77]. In the context

of a recurrent neural network (RNN), a discrete-time dynamical system is represented by an input x_t , an output y_t , and a hidden state h_t . time is denoted by the subscript t . The formulation of the dynamical system is expressed as follows:

$$h_t = f_h(x_t, h_{t-1}) [78] \tag{2.3}$$

$$y_t = f_o(h_t) [78] \tag{2.4}$$

When an output function is denoted by f_o and a state transition function by f_h , respectively. A pair of parameters, θ_h and θ_o , define each function. To estimate the parameters of an RNN, given a collection of N training sequences $D = \{(x_1^{(n)}, y_1^{(n)}), \dots, (x_{T_n}^{(n)}, y_{T_n}^{(n)})\}_{n=1}^N$, the cost function can be minimized as follows:

$$J(\theta) = \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} d(y_t^{(n)}, f_o(h_t^{(n)})), [78] \tag{2.5}$$

where $h_t^{(n)} = f_h(x_t^{(n)}, h_{t-1}^{(n)})$ as well as $h_0^{(n)} = 0$. A predetermined divergence measure, such as cross-entropy or Euclidean distance, between a and b is denoted by the symbol $d(a, b)$ [78].

2.4.2.1 RNN architecture types

The consists of the following components: The most prominent architecture types of an RNN that will be explained below:

1. **Fully Recurrent Neural Network (FRNN):** The 1980s saw the development of the FRNN, which has two layers—input and output—connected by movable weights. Through feeding back activations to input layer units, it is able to learn temporal sequences. Mapping input sequences to output sequences across several time steps is the process of learning [79].

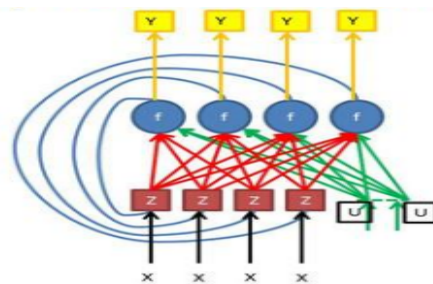


Figure 2.4: FRNN [79]

2. **Recursive Neural Network:** This network, which is frequently used in natural language processing to process distributed representations of structures, applies the same set of weights recursively in a graph-like form. The Recursive Neural Tensor Network is one variant that makes use of tensor-based composition functions [79].

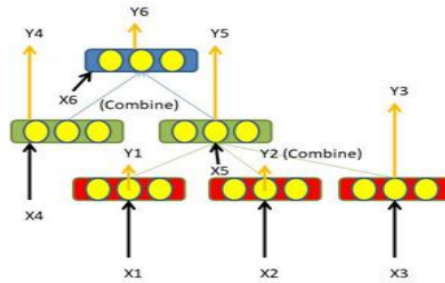


Figure 2.5: Recursive Neural Network [79]

3. **Hopfield Network:** It was created in 1982 by John Hopfield and has symmetric connections between neurons. Neurons individually and asynchronously adjust their activity values. It is taught utilizing either the Hebbian or the Storkey learning principles, and it is utilized for Content Addressable Memory (CAM) [79].

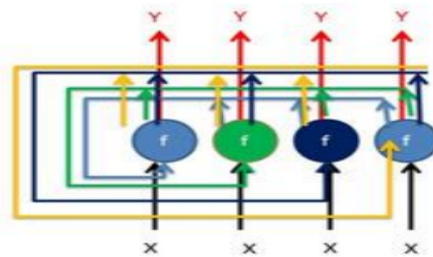


Figure 2.6: Hopfield Network [79]

4. **Elman Networks and Jordan Networks (Simple Recurrent Network):** Elman networks store past values of hidden units and have an extra context layer. Similar to hidden layer networks, Jordan networks use feedback context units from the output layer [79].

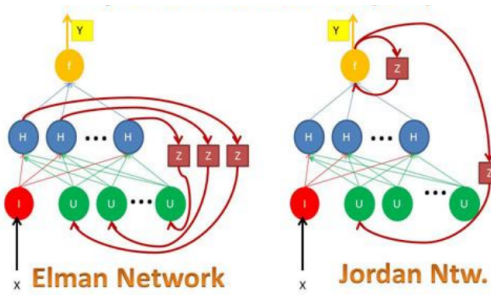


Figure 2.7: SRN [79]

5. **Echo State Network:** This network's hidden layer has a fixed random weight assignment and sparse connection. It is possible to learn the weights of output neurons to replicate particular time-based patterns [79].

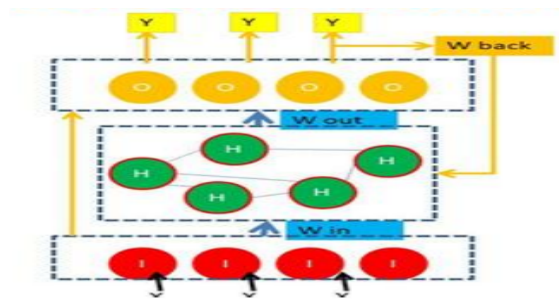


Figure 2.8: Echo State Network [79]

6. **Long Short-Term Memory (LSTM):** The vanishing gradient issue is avoided in the task-learning process by LSTM. To preserve recollection of previous occurrences, it has recurrent gates such as "forget" gates [79].

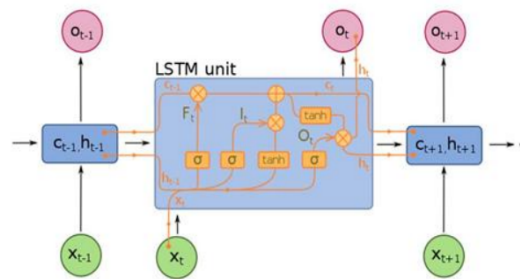


Figure 2.9: LSTM [79]

7. **Gated Recurrent Units (GRUs):** GRUs, which Kyunghyun Cho first introduced in 2014, have gating processes akin to LSTM but with fewer parameters. Activities like as speech signal modeling and polyphonic music are areas in which they excel [79].

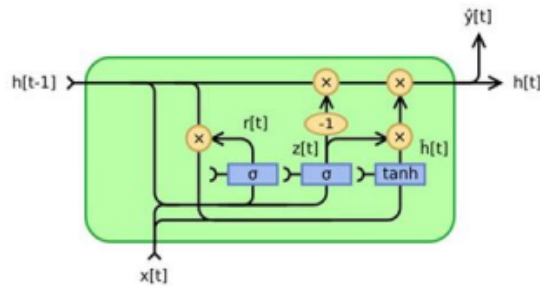


Figure 2.10: GRUs [79]

2.4.3 Generative adversarial networks (GANs)

With the use of two neural networks staged as opponents to one another, generative adversarial networks, or GANs, create fresh synthetic sample data that closely mimics real sample data and has a high chance of being accepted as real inputs. They are widely employed in the production of speech, films, and photographs. Due to their superior performance in image tasks, GANs are especially well-suited for image processing. They are employed in many different applications and are thought to be the most effective method for creating images. The discriminator's primary objective is to determine if a sample is true or false in its distribution. In the meantime, the generator generates a false trial distribution in an attempt to trick the discriminator. The discriminator determines the likelihood or not that a given sample is authentic. If the likelihood value is higher, the sample is more likely to be representative of the population. If the value is around 0, the sample is bogus. The optimal answer is provided when the probability value is close to 0.5, which illustrates the absence of difference between synthetic and real sample data [80].

2.4.3.1 Network architecture and learning

As illustrated in Figure 2.11, the architecture of Generative Adversarial Networks (GANs) comprises of a Generator (G) and a Discriminator (D). The generator uses a random noise vector (Z) as input and outputs an image ($G(z)$) in an attempt to create an image. After that, the Discriminator receives the created image and uses its output to update the Generator's parameters. The Discriminator is a binary classifier that checks samples produced by

the Generator simultaneously, distinguishing between those that are real and those that are fraudulent. The Discriminator simulates the likelihood that a sample image, X , is authentic or fraudulent. The Generator then receives these probabilities back as feedback. The word "adversarial" in Generative Adversarial Networks originates from the competition that both the Generator and the Discriminator have over time as they attempt to outsmart one another. The minimax game problem forms the basis of the optimization procedure. In order to enable the Generator to generate realistic-looking images and the Discriminator to get increasingly better at differentiating between real and fake generated images, backpropagation is used during training to update the parameters of both the Generator and the Discriminator. The Minimax loss function, which was first proposed by Goodfellow et al., is used by GANs. Whereas the Discriminator seeks to increase the following function, the Generator seeks to decrease it [81]. The Minimax loss is represented by:

$$\min_G \max_D f(D, G) = \mathbb{E}_x[\log(D(x))] + \mathbb{E}_z[\log(1 - D(G(z)))] [81] \quad (2.6)$$

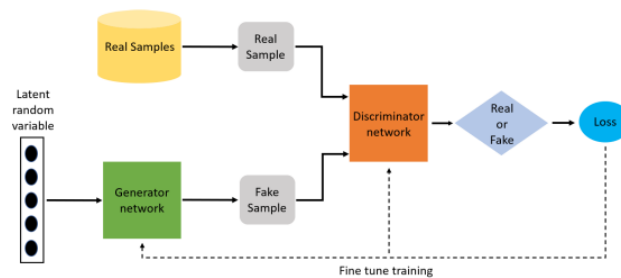


Figure 2.11: Basic GAN architecture [81]

2.4.3.2 Types of GANs

The following is an explanation of the most common Types of GANs:

1. **Vanilla GANs:** is characterized as a kind of generative model in which the discriminator and generator are multi-layer perceptrons. The minimax optimization problem must be solved in order to formulate a Vanilla GAN mathematically. While the discriminator D attempts to discern between the genuine samples from μ and the generated samples from G , the generator G generates samples from a latent space distribution γ to mimic the data distribution μ [82]. Given is the objective function:

$$V(D, G) = \mathbb{E}_{x \sim \mu}[\log D(x)] + \mathbb{E}_{z \sim \gamma}[\log(1 - D(G(z)))] [82] \quad (2.7)$$

The optimization problem is then:

$$\min_G \max_D V(D, G) [82] \quad (2.8)$$

The generator iteratively tries to make its output better in an attempt to trick the discriminator, and the discriminator adjusts itself in order to become more adept at differentiating between produced and genuine samples. The goal is to get to a point of equilibrium where the discriminator cannot tell the difference between the generated and genuine samples [82].

- 2. Deep Convolutional GAN (DCGAN):** The generator G and discriminator D are two deep neural networks that make up a deep convolutional generative adversarial network (dCGAN). Fake molecular fingerprints are produced by G, which begins with random noise inputs and goes through the reverse convolutional process to capture the data distribution. In order to determine whether or not a sample fingerprint originates from G, D is trained with real data using a chosen CNN architecture. The idea of dCGAN is to simultaneously train these two models iteratively and enhance their performance. When D is unable to identify the data produced by G from real data, it means that G has already mastered the hidden pattern needed to sample fresh fingerprints. In the section on convolutional neural networks, the construction process of D was described. In general, the method of developing G is the opposite of that of building a CNN model. A reshape layer was inserted using reshape in between the first convolution layer and the fully linked layer. Between each convolution layer, UpSampling1D was applied to gradually change the data's output shape [83].
- 3. Conditional Generative Adversarial Networks (CGANs):** are regarded as the main method for producing images from text; they enable content control by giving the discriminator and generator an extra condition. Controllable and semantically matched visuals are produced as a result, matching the supplied text. Innovative solutions, like the LD-CGAN model, have been proposed in spite of obstacles like unstable training processes and nonsensical results. This model uses effective guidance of semantic information and partitioning to simplify the network and enhance the quality of generated images. Enhancement of spatial characteristics in multi-scale contexts, differentiation of semantic processes in producing images ranging from low to high resolution, and use of numerous losses to improve the quality of generated images are other notable aspects of LD-CGAN [84].

4. **CycleGAN**: is a new method for image-to-image translation. CycleGAN's unique power comes from the fact that it just needs unpaired instances from the X and Y picture domains. In order to meet the following two requirements, CycleGAN trains two transformations, $F : X \rightarrow Y$ and $G : Y \rightarrow X$, in parallel: 1. $F(x) \sim p(y)$ for $x \sim p(x)$, and $G(y) \sim p(x)$ for $y \sim p(y)$; 2. $GF(x) = x$ for all $x \in X$, and $F(G(y)) = y$ for all $y \in Y$. The distributions of the two picture domains, X and Y , are described by the expressions $p(x)$ and $p(y)$. The first criterion, which is implemented by training two discriminators on X and Y , respectively, makes sure that the output images seem to originate from the desired domains. The second requirement, which is imposed by a cyclic consistency loss of the form $\|GF(x) - x\| + \|F(G(y)) - y\|$, makes sure that the information about a source image is encoded in the generated image. Semantically encoding the information from the source picture x into the components of the output image $F(x)$ is the aim [85].

2.4.4 Vision transformer (ViT)

Perspective Since their initial introduction by Dosovitskiy et al., Transformers (ViTs) have shown greater performance in image classification applications when trained on large-scale datasets when compared to state-of-the-art (SOTA) Convolutional Neural Networks (CNNs). On the other hand, ViTs need a lot of processing power and training data. Touvron et al. addressed this problem by introducing a data-efficient ViT that made use of regularization and data augmentation strategies that were previously used for CNNs. They further enhanced performance by implementing a Transformers-based teacher-student method. ViTs have been used by researchers to tackle a variety of vision problems, including object detection and image segmentation. For example, Carion et al. developed a new architecture for object detection on the COCO dataset that uses a Transformer encoder-decoder and a set-based global loss to achieve competitive results. Transformers have been applied to 3D medical picture segmentation, leveraging global attention features by combining the benefits of Transformer networks with U-Net. Choosing the right pre-trained model for transfer learning (TL) remains a difficulty, even with the many advancements made to ViT models and the abundance of pre-trained ViT models available. To efficiently attain the greatest performance for new tasks, Steiner et al. advised selecting a small number of well performing pre-trained models for fine-tuning rather than adjusting all pre-trained Transformers [86].

2.4.4.1 Vision transformer's operation

The following processes, which are all essential to the overall operation of the Vision Transformer, can be used to break down how it operates as shown in Figure 2.12:

1. **Patch Embedding:** One way to partition the incoming image is into square patches of fixed size. Learnable linear projection is used to turn each patch into a vector. This generates a series of patch embeddings that act as input tokens for layers that follow [87].
2. **Positional Embedding:** The intentional provision of positional information makes up for the Transformer's inbuilt lack of spatial awareness. In order for the model to discern between different places in the image and capture spatial relationships, positional encodings are usually added to patch embeddings during the input stage and are often learned [87].
3. **Encoder Layers:** Multiple encoder layers, each with two main sub-layers—multi-head self-attention and feedforward neural networks—make up the core of the Vision Transformer [87].
4. **Multi-Head Self-Attention:** Relationships between various patches in the input sequence are captured by this method. For every patch, it calculates a weighted sum of all patch embeddings, with weights based on importance. Using numerous sets of learnable characteristics (attention heads), multi-head attention captures different kinds of relationships [87].
5. **Feedforward Neural Networks:** The output of each patch is fed via a feedforward neural network after self-attention. Usually, this network consists of ReLU activation and a completely connected layer. By doing this, non-linearity is introduced, allowing the model to pick up intricate patch correlations [87].
6. **Layer Normalization and Residual Connections:** Layer normalization and residual connections follow both feedforward network outputs and self-attention network outputs. By normalizing sub-layer inputs, layer normalization stabilizes training. During training, residual connections help gradient flow, which reduces the risk of disappearing gradient problems [87].

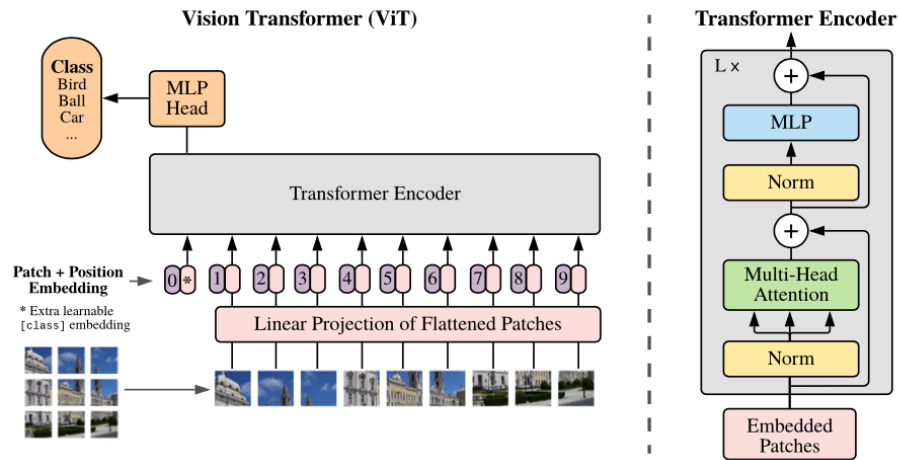


Figure 2.12: Basic ViT architecture [88]

2.4.5 Transfer learning (TL)

The concept of Transfer Learning (TL) involves describing the notion of transferring knowledge across tasks and domains through TL. To enhance learning in a target domain and task, it entails leveraging past knowledge from a source domain and task. For example, since riding a bike and learning to ride a motorbike are in the same domain, learning to ride a motorbike can benefit from prior cycling experience. TL is a useful strategy in a variety of learning contexts where it reduces the need for large amounts of training data and laborious, from-scratch model training [89].

Its known as A machine learning technique involves developing and training a model for one job, then using it for a related secondary task. It describes the circumstance in which knowledge gained in one context is applied to improve performance in another. When a fresh dataset is available that is smaller than the one that was initially used to train the pre-trained model, transfer learning is usually used. Instead of beginning the learning process from scratch with random weight initialization, it enables us to start with the learned features on the ImageNet dataset and modify these features, as well as maybe the model's structure, to suit the new dataset/task. This makes Transfer Learning possible for picture classification. Even with limitations in terms of time and computer power, we are able to identify the variables influencing classification accuracy by testing the network and adjusting its topology (i.e., parameters) and dataset properties [90].

2.4.5.1 Pre-trained models categories

Pretrained models are classified based on the kind of data they utilize and tasks they accomplish. The following three fall into these categories:

1. **Image Classification Models:** possess a range of alternatives at their disposal for classifying images:

- **VGG19:** VGG19 is a Convolutional Neural Network (ConvNet) model developed by the Visual Geometry Group (VGG) at the University of Oxford. This model is part of the VGG models family, which was introduced in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2014.

The VGG19 model consists of 19 deep layers, including Convolutional Layers, Max-Pooling Layers, and Fully Connected Layers. The model is characterized by using small filters of size 3×3 in the convolutional layers, allowing it to capture fine details in images while maintaining relatively low computational complexity. VGG19 starts with an input of size $224 \times 224 \times 3$ (the original image is resized to this size) and passes through several convolutional and pooling layers before the output is passed through three fully connected layers and then to a Softmax layer for image classification. The model uses 2×2 max-pooling after each block of convolutional layers to reduce the spatial feature size.

The key features of VGG19 include:

- Simplicity in design, as only small (3×3) convolutional filters are used.
- Significant depth with 19 convolutional and fully connected layers.
- Excellent performance in image classification tasks.

VGG19 has been widely used in various research and commercial applications due to its high performance and ease of modification and extension to other deep learning models [91]. Its architecture is shown in the figure 2.13.

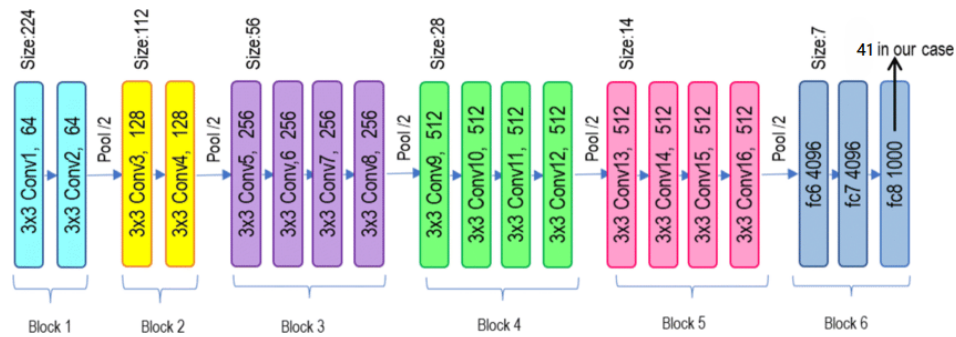


Figure 2.13: VGG-19 Architecture [92]

- EfficientNet-B0:** One Convolutional Neural Network (CNN) architecture that use the compound scaling technique to boost accuracy is called EfficientNet. Using a compound coefficient, this method uniformly scales the three dimensions (width, depth, and resolution). EfficientNet comes in a variety of models, from B0 to B7. The elements of MobileNetV2, which comprise the Mobile Inverted Bottleneck Conv (MBConv) with the inclusion of Squeeze and Excitation (SE) optimization, form the foundation of the EfficientNet-B0 architecture. Studies have indicated that the utilization of MBConv and SE blocks can improve precision while reducing the quantity of parameters, which qualifies this architecture for use in mobile applications [93]. The EfficientNet-B0 architecture’s layer arrangement is shown in Figure 2.14.

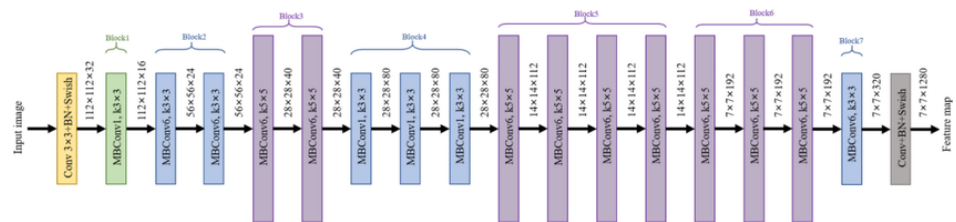


Figure 2.14: EfficientNet-B0 architecture [94]

- Resnet50:** Is deep convolutional neural network with 50 layers, addresses the challenges of training deep networks, such as vanishing or exploding gradients and accuracy degradation. This architecture includes groups of identical layers, highlighted in different colors, and identity blocks that use outputs from previous layers in subsequent layers. The initial layer features 64 filters with a 7x7 kernel, followed by a 3x3 max-pooling layer. The first group has three identical blocks, the second and third groups have four, and the fourth group has three. Blue

curves represent identity blocks connecting layers of different sizes. The network culminates in 38 fully connected layers for classification [95]. As shown in Figure 2.15.

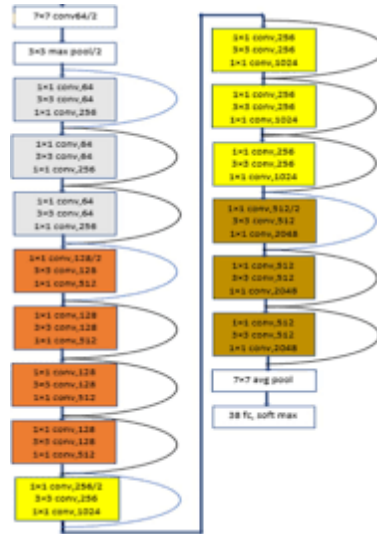


Figure 2.15: resnet50 architecture [95]

2. **Generative Models:** There are many such options for generative models. For example, Pix2Pix is an image-to-image translation technique that transfers images between domains. It uses the generative adversarial network framework: a discriminator model decides whether an image is actual or generated, and a generator converts input images into new images. Unlike a plain GAN, the discriminator in Pix2pix is conditioned on image pairs: one image comes from the generator's output and the other from the target image instead of drawing the generator from a noise vector. The pix2pix loss is a concatenation of an L1 loss and an adversarial loss. The L1 does not bring up blurriness as it measures the dissimilarity between the generated and target images. The adversarial loss makes sure there is enough realism in the generated images to be able to fool the discriminator. These two losses are then summed to arrive at the final loss function, weighted by the parameter lambda (λ) [96]. The Pix2pix architecture is shown in Figure 2.16.

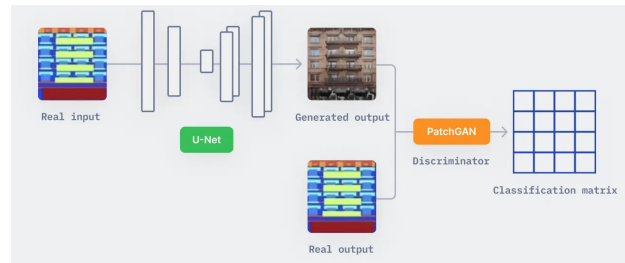


Figure 2.16: PIX2PIX architecture [97]

3. **VIT Models:** There are numerous options available for ViT models, such as ViT-B/16, a variant of the Vision Transformer (ViT) architecture. The "B" stands for "Base" indicating the model size, and "16" refers to the patch size (16x16 pixels) used in the model. Vision Transformers, or ViTs, are a type of neural network architecture designed for image recognition tasks, leveraging the transformer model originally designed for natural language processing.

ViT-B/16 employs self-attention mechanisms to process image patches, enabling it to capture long-range dependencies and spatial information across the entire image. This approach contrasts with traditional Convolutional Neural Networks (CNNs) that rely on convolutional operations. ViT-B/16 has demonstrated competitive performance across various image classification benchmarks and is part of a family of models that vary by the number of parameters and patch sizes, including ViT-L/16 for "Large" and ViT-H/14 for "Huge" [98].

2.5 Conclusion

This chapter has examined the theoretical foundations and real-world applications of deep learning methods for the identification of microorganisms in water. The study examined the use of neural network techniques and their diverse applications in enhancing the precision and velocity of biological pollutant detection in potable water.

In the upcoming chapter, the focus will be on the techniques used to achieve Electron Microscopy (EM) detection of microorganisms in drinking water. Key models to be developed, data preparation methods, and performance testing will be emphasized to ensure accurate and reliable results. This will enable the presentation of innovative and effective solutions to improve the quality of drinking water and ensure its safety.

Chapter 3

System Design

3.1 Introduction

The significant advancements in the field of artificial intelligence and its applications in various sectors, including environmental monitoring, have paved the way for innovative systems capable of predicting trends and aiding in decision-making processes. This chapter provides an overview of the system designed for the identification and classification of EM using microscopic images.

Firstly, we will discuss the general architecture of our classification model. Following this, we will delve into the specific functionalities of our model, including dataset preparation, model training, and performance evaluation, and model deployment.

3.2 Global architecture

The global architecture describes the microscopic images classification process. This process begins with the collection of a dataset comprising electron microscope images, which then undergo pre-processing to enhance their quality. The pre-processed data is subsequently split into training, testing, and validation sets. The training set is used to train the model to detect patterns and extract significant features. After training, the model's performance is analyzed using the testing and validation sets to ensure accuracy and generalizability. Based on this analysis, a decision is made: if the model meets the performance criteria, it is accepted and used to create a web application for real-time water quality assessment. If the model does not meet the criteria, it undergoes further training and refinement. This iterative process ensures the development of a robust and effective system for identifying and

classifying waterborne pathogens. As shown in the figure 3.1.

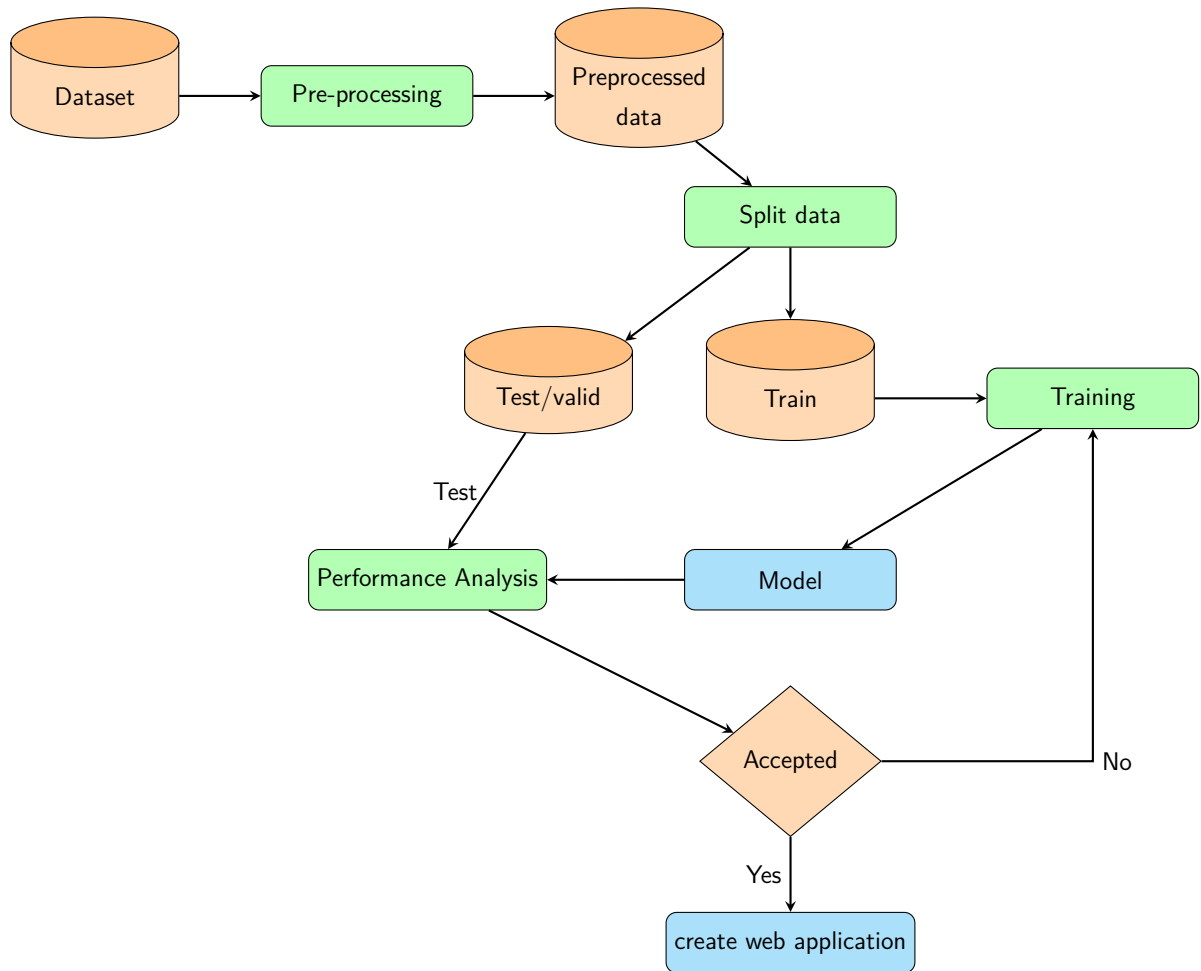


Figure 3.1: General architecture

3.3 Detailed architecture

The comprehensive architecture and procedures for managing the dataset, from collection to preprocessing, are described in this section. The objective is to guarantee that the data is adequately prepared for training machine learning models through adherence to a methodical and comprehensive procedure.

3.3.1 Data gathering

The gathered dataset included PNG-formatted EM images from various classes. There are 41 folders (classes) in each part, except for one class that we did not put in a folder

because it does not contain any images, which is named "unknown".

3.3.2 Pre-processing

The pre-processing phase is a critical stage in image classification tasks, where we modify the data through processes such as resizing, applying filters, and removing noise before passing the data to the training model. In our system, we resized all the images to 224x224 pixels and performed normalization (see Figure 3.2).

Resizing: The original dataset images had varying dimensions. To standardize these dimensions, we resized the images to (224x224x3), where the first two numbers correspond to the width and height of the image, and the third number refers to the image channels, indicating that the images are RGB.

Normalization: After resizing, the images were normalized to ensure consistent pixel value ranges across all images. This process involves scaling the pixel values to a range of [0, 1] by dividing each pixel value by 255 (the maximum pixel value for an 8-bit image). Normalization helps in speeding up the convergence of the training process and improves the overall performance of the machine learning model.

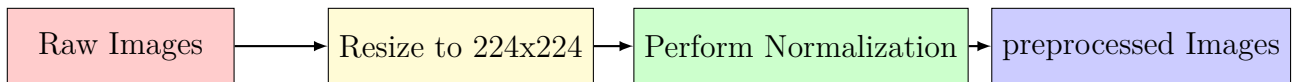


Figure 3.2: Pre-processing steps.

3.3.3 Splitting

When partitioning a dataset in deep learning, we typically divide it into subsets. The common practice involves three partitions: a training set, a test set, and a validation set. However, due to the small size of the dataset in this scenario, we opted for only two partitions: one for training and one for testing. In this setup, 80% of the data is used for training, while 20% is allocated for testing. The division is performed accordingly. In this arrangement, the test set also serves as the validation set, meaning the same dataset is used to evaluate the model's performance and validate its generalization ability (see 3.3).

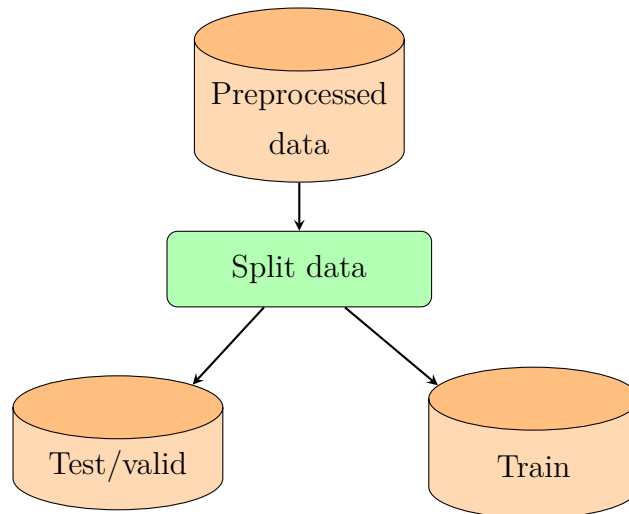


Figure 3.3: Data partitioning: 80% for training and 20% for testing/validation.

3.3.4 Train

In deep learning, the training phase is essential for developing a model that can make accurate predictions on new data. This phase involves multiple steps, starting with data preparation and ending with model fitting. Data augmentation is a crucial technique in this process, used to artificially expand the size of a training dataset by creating modified versions of existing images. This enhances the model’s performance by making it more robust to variations in the data. The typical architecture of the training phase includes collecting training data, applying data augmentation techniques, using the augmented data for training, and ultimately obtaining a fitted model.

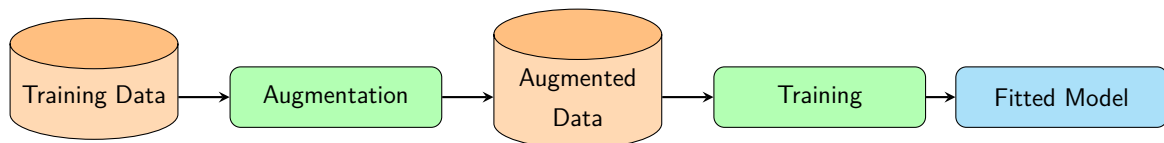


Figure 3.4: Training Phase Architecture

3.3.4.1 Augmentation

Data augmentation is a technique used to increase the size and diversity of training data for deep learning networks, which helps improve the performance of neural networks by reducing overfitting and enhancing generalization. This technique includes several methods such as traditional geometric transformations (e.g., rotation, flipping, scaling), color adjust-

ments (e.g., increasing contrast, adjusting white balance, sharpening), and using Generative Adversarial Networks (GANs) to create new images. Additionally, it involves texture transfer and style transfer techniques that enable the creation of high-quality perceptual images that combine the content of one image with the style of another [99]. These are the methods we employed in my work to augment and enhance the train dataset (see figure 3.5):

Flipping: Horizontal flipping is frequently used to increase the variety of training data. It helps mitigate misclassification issues caused by uniform image orientations. However, it might not be effective for datasets with asymmetrical or directionally sensitive data, like characters or digits, as it can lead to incorrect or opposing classifications [100].

Random Rotations: The images are rotated at random by multiples of ninety degrees (90°, 180°, 270°) to guarantee that the model becomes invariant to various object orientations.

Blurring the Image: A regression technique used as a preprocessing step in many computer vision algorithms to smooth, blur, and remove noise from an image is included in the Gaussian blur technique (Chauhan, 2018). Gaussian blur is a kind of linear low-pass filter in which the Gaussian function is used to determine the pixel value (Novák et al., 2012). Equation 3.1 (Novák et al., 2012) defines the two-dimensional Gaussian function as the sum of two one-dimensional Gaussian functions:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} [101] \quad (3.1)$$

Where the coordinates are (x, y) , and the Gaussian distribution's standard deviation is ' σ ' [101].

Contrast Limited Adaptive Histogram Equalization (CLAHE): CLAHE is an image enhancement technique that modifies the histogram's structure to ensure a more evenly distributed intensity level across an image. Unlike Adaptive Histogram Equalization (AHE), which can overly enhance contrast, CLAHE limits the histogram to a boundary value to avoid this issue. By dividing the image into contextual sections and applying histogram equalization to each pixel value, CLAHE makes hidden features in the image more visible [102].

Cutout Augmentation: According to DeVries and Taylor (2017), cutout is an image enhancement method that removes contiguous areas of data from images. For multivariate time series (MTS), temporal cutout randomly selects a time segment and a number of channels, setting the chosen values to zero. The size of the time segment is randomly determined between specified maximum and minimum hyperparameter values. The probability of each channel being selected is controlled by a hyperparameter [103].

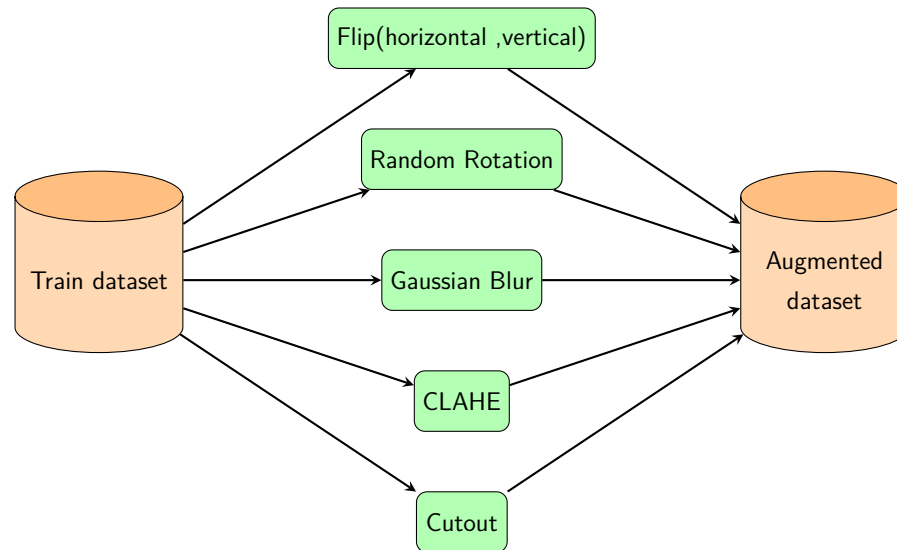


Figure 3.5: Data Augmentation Processes

3.3.4.2 Used Models

In this work, we employ a variety of models to tackle the challenges of microorganism detection in water samples. These models are presented in the following subsection:

1. **Pretrained Models:** we will use well-known pretrained models in my project as they provide substantial knowledge gained from training on large datasets. We will utilize ResNet50, VGG19, and EfficientNet-B0 to achieve my goal of image classification. These models excel in accurately and effectively classifying images into specific categories.

Additionally, We will leverage these pretrained models for feature extraction. The hidden layers in ResNet50, EfficientNet-B0, and VGG19 provide rich representations of images, which can be used in various applications. Using these models saves time and resources, and enhances model performance and accuracy across different tasks.

2. **Pretrained & Machine Learning Models:** We propose utilizing pretrained models for feature extraction, while employing traditional machine learning algorithms for classification. For this purpose (see figure 3.6), we have selected the following algorithms:

Support Vector Machines (SVM): A collection of supervised machine learning techniques called Support Vector Machines (SVM) are utilized in regression and classification. SVM is characterized as a predictive tool that prevents overfitting to the

data by utilizing machine learning theory to improve prediction accuracy. SVM, which allows for control over complexity and avoids problems with high-dimensional data, is frequently used in text analysis, handwriting recognition, and image classification. It plays a significant role in pattern classification and addressing regression problems. By transforming data into a high-dimensional feature space where linear classification is carried out, the kernel technique allows SVM to extend this method to non-linear boundaries. SVM operates by determining the ideal hyperplane that divides various data classes with the widest feasible margin. The ideal hyperplane is defined by a collection of "support vectors" that are included in the final model. Since its initial introduction at the COLT-92 conference in 1992, SVM has developed into a vibrant field of machine learning research. Its remarkable efficacy in a wide range of real-world applications, including text categorization and handwriting recognition, has contributed to its growing popularity. SVM is a potent and adaptable method for regression and classification that combines real-world applications with mathematical theory to produce precise and broadly applicable findings [104].

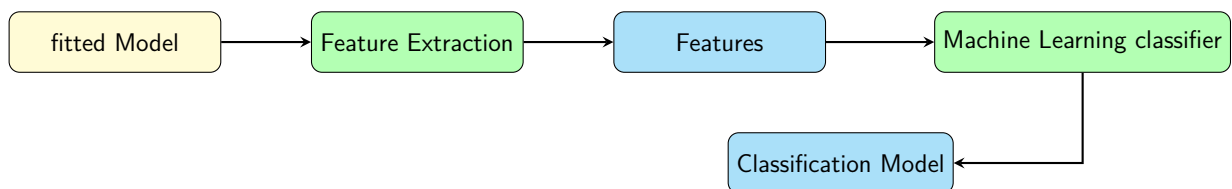


Figure 3.6: Feature Extraction and Classification Architecture

- 3. Vit Models:** Vit Models are a promising new technique that we would like to use in this research on the detection of microorganisms in water. This technology is part of the ongoing development in the field of artificial intelligence and deep learning, and is primarily used for analyzing large datasets and enhancing prediction accuracy and processing. And this is our vit architecture we used ViT-B16 as pretrained model (see Figure 3.7).

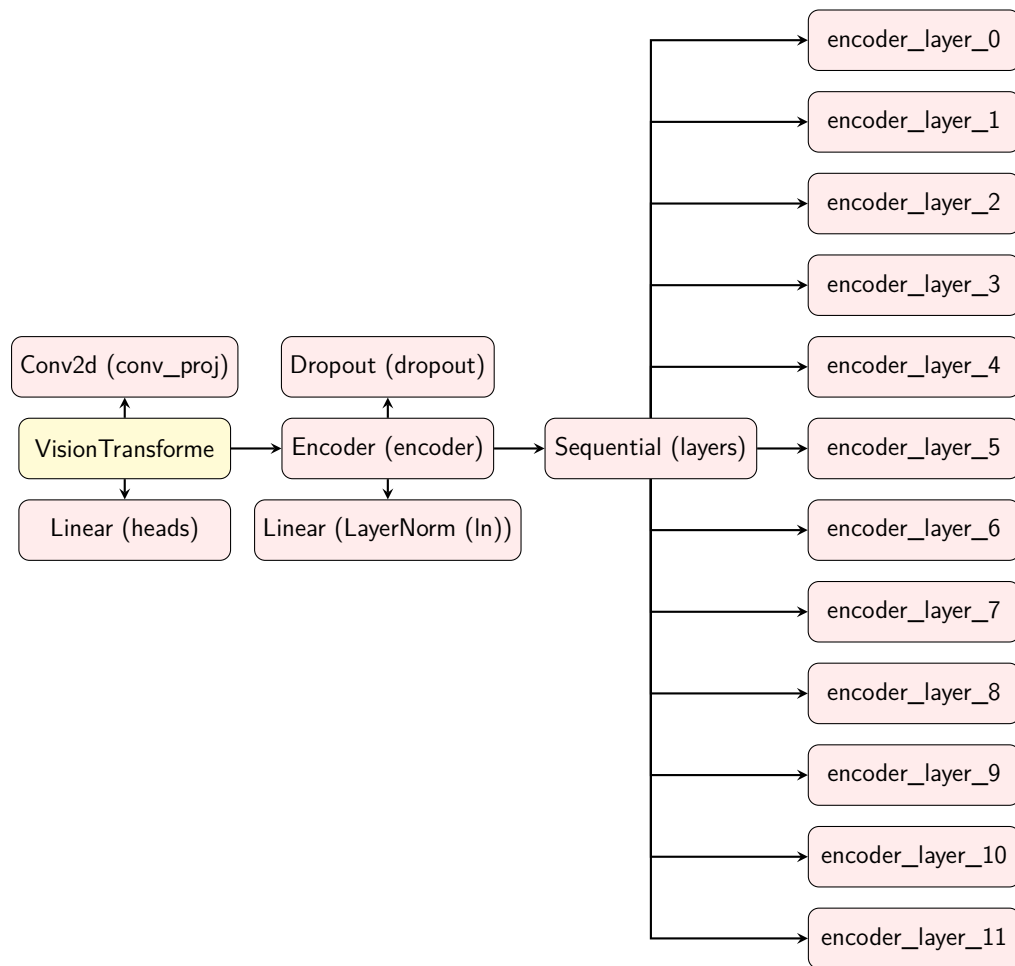


Figure 3.7: Vision Transformer Architecture

3.3.5 Test

The testing phase is when the trained model is assessed using the test set to gauge its performance. This step is crucial as it is the last one before the model is deployed in a real-world setting.

To get a class prediction from the trained model, an image of the same dimensions as the training image is fed into the network (see figure 3.8).

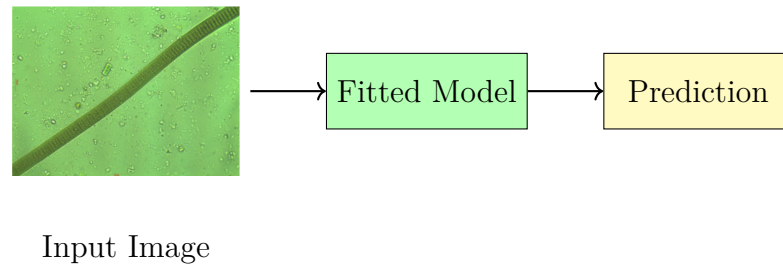


Figure 3.8: testing Phase.

3.3.6 Evaluation metrics

We've put in place a number of indicators to make sure our system is sturdy and dependable. By comparing expected and actual results, these measures generate scores that indicate how effective the model is. They are used in a traditional data classification procedure in both the training and testing stages. Generally speaking, scores fall between 0 and 1, where 0 denotes subpar performance and 1 (or 100%) denotes perfect performance.

Confusion Matrix

A table known as a confusion matrix is frequently used to explain how well a classification model performs when applied to a set of test data for which the true values are known. It makes it possible to visualize how well an algorithm performs. In the matrix, the examples in a predicted class are represented by each column, and the occurrences in an actual class are represented by each row. There are four parts to the confusion matrix:

- True Positives (TP): The number of observations correctly predicted as positive.
- True Negatives (TN): The number of observations correctly predicted as negative.
- False Positives (FP): The number of observations incorrectly predicted as positive.
- False Negatives (FN): The number of observations incorrectly predicted as negative.

The confusion matrix is often presented in the following format:

	Actual Positive	Actual Negative
Predicted Positive	<i>TP</i>	<i>FP</i>
Predicted Negative	<i>FN</i>	<i>TN</i>

precision

The positive patterns that are successfully predicted from all of the projected patterns in a positive class are measured using precision. The equation for precision is given by:

$$\text{Precision (p)} = \frac{TP}{TP + FP} \quad (3.2)$$

Recall

The percentage of positive patterns that are correctly categorized is measured by recall. The recall equation is provided by:

$$\text{Recall (r)} = \frac{TP}{TP + FN} \quad (3.3)$$

F1-Score

The harmonic mean of recall and precision yields the F1-score, which strikes a balance between the two measures. It is computed with the following formula:

$$\text{F1-Score (F1)} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3.4)$$

Accuracy

The ratio of accurate predictions to all instances analyzed is the measure of accuracy. It is computed with the following formula:

$$\text{Accuracy (acc)} = \frac{TP + TN}{TP + FP + FN + TN} \quad (3.5)$$

3.3.7 Model deployment

Training a model is not the end goal in deep learning. A deep learning project only gains interest when it is widely adopted, regardless of its application—for example, object detection or image categorization. Herein lies the role of deployment. The process of putting a completed machine learning model into a real-world setting where it may be utilized for the intended goals is known as deployment. Models can be implemented in many different contexts, and in order to make them available to end users, they are frequently connected into applications using an API [105].

Deploying our model requires multiple processes, which are frequently completed concurrently, after it has been trained and evaluated. The model must first be moved to the deployment environment so that it may access the required data sources and hardware resources. The model must be incorporated into the operational procedure second. Making it available to end users via a web browser is part of this (see figure 3.9).

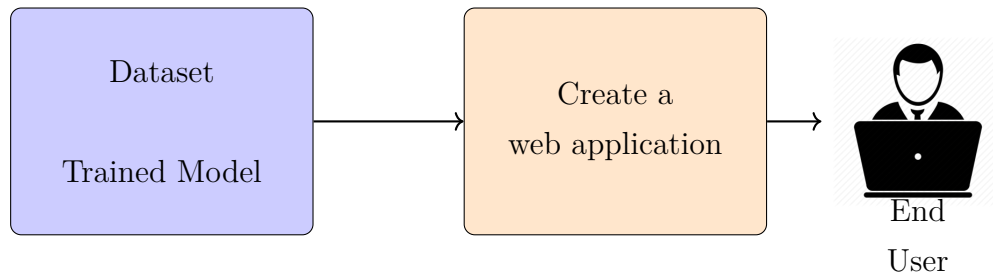


Figure 3.9: Model Deployment.

3.4 Conclusion

In conclusion, the aim of this chapter was to present a detailed system design for leveraging electron microscopy images to detect and classify microorganisms in water. We covered the methodologies involved in data collection, preprocessing, model training, and performance evaluation, demonstrating how these elements contribute to the development of an effective real-time water quality assessment tool. The system's ability to enhance the accuracy and speed of microorganism detection offers a scalable solution for various water monitoring scenarios.

The iterative process of reevaluation and adjustment highlighted our commitment to achieving optimal model performance. In the next chapter, we will focus on the implementation of this system and the practical applications of our findings in real-world scenarios, providing insights into the operational aspects and potential improvements for future developments in water safety technology.

Chapter 4

Implementation and results

4.1 Introduction

In this chapter, we discuss the implementation of deep learning models for detecting microorganisms in water and evaluate the performance of these models using a specific dataset. The study focuses on comparing several advanced models, including pre-trained models and models combined with traditional machine learning algorithms such as SVM, in addition to Vision Transformer (ViT) models. In this chapter, we will review the implementation steps, from data collection and preprocessing to model training and evaluation. Additionally, we will discuss the performance of different models based on various metrics such as precision, recall, and classification accuracy. Finally, we will cover the deployment process of the model using tools like Gradio to facilitate model access and application in a real-world environment.

4.2 Implementation tools and languages

We were able to familiarize ourselves with a variety of development methodologies and tools during the project's implementation phase, which required the use of specific programming domains. These are listed below:

4.2.1 Python

Python is a high-level, interpreted, object-oriented language with dynamic semantics. Its dynamic typing and dynamic binding, along with its high-level built-in data structures, make it an appealing language for Rapid Application Development and for usage as a scripting

or glue language to join existing components. Because of its straightforward, basic syntax, Python promotes readability, which lowers software maintenance costs. Python's support for packages and modules promotes code reuse and program modularity. The large standard library and the Python interpreter are freely distributable and accessible for free on all major platforms in source or binary form [106].

The version of Python used in this project is 3.10.13, as shown in Figure 4.1.

```
1 from platform import python_version
2 python_version()
3 '3.10.13'
4
```

Figure 4.1: Python version code and output.

4.2.2 Kaggle

Kaggle is the world's largest community of over 18 million data scientists and machine learning/data science enthusiasts [107][108]. Founded in 2010 by Anthony Goldbloom and Ben Hamner, the company is headquartered in the San Francisco Bay Area . Kaggle provides a vibrant platform where data scientists compete to solve complex, real-world problems using the latest machine learning techniques [108].

The platform offers a plethora of resources, including over 338,000 public datasets, more than 1,080,000 public notebooks, and over 4,900 pre-trained models. Kaggle hosts more than 27,000 competitions to help users develop their skills and learn new methods. It also provides free educational courses on programming, machine learning, and data manipulation, complete with certificates. The supportive and inclusive learning environment encourages interaction through forums and mentorship programs [107].

Kaggle has a proven track record of delivering cutting-edge business results, especially in the Energy sector, as well as solving challenging problems across a diverse array of industries including life sciences, financial services, aviation, information technology, and retail. The dedicated team, led by D. Sculley, Jeff Moser, and William Cukierski, works diligently to ensure Kaggle remains at the forefront of innovation and education in the field of data science and machine learning [107].



Figure 4.2: Kaggle Logo [107]

4.2.3 opencv

A comprehensive open-source software library for computer vision and machine learning applications is called OpenCV, or Open Source Computer Vision Library. It was established in 2010 and offers a shared infrastructure to hasten the application of machine perception in items for sale. OpenCV provides both traditional and cutting-edge methods for applications like object identification, face recognition, human activity classification in films, and much more. It has over 2500 optimized algorithms. With more than 47,000 users and an estimated 18 million downloads, it is widely used by businesses, academic institutions, and governments all around the world. The library leverages specific hardware instructions and supports a variety of operating systems and programming languages, with a bias toward real-time vision applications. Furthermore, it fosters community involvement and offers multiple channels for assistance and cooperation, guaranteeing ongoing advancements and novelty in the domain of computer vision and machine learning [109].



Figure 4.3: OpenCv logo [109]

4.2.4 Pytorch

The PyTorch Foundation serves as a collaborative hub for the deep learning community, fostering the growth of the open-source PyTorch framework. PyTorch was created in September 2016 by Meta AI (formerly Facebook AI). Led by experienced AI/ML experts from major tech companies, the Foundation facilitates community engagement and supports the development of the PyTorch ecosystem. As part of The Linux Foundation, the PyTorch community works on initiatives like training, events, tooling, and research to enable PyTorch usage at scale. The Foundation's guiding principles focus on democratizing state-of-the-art AI/ML tools and making them accessible to all. The Foundation has a governance structure and encourages contributions from developers and member companies. Developers can join the community, access comprehensive resources, and collaborate with the PyTorch Foundation representatives [110].



Figure 4.4: Pytorch logo [110]

4.2.5 Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It makes easy things easy and hard things possible. It allows you to create publication-quality plots, make interactive figures that can zoom, pan, and update, customize visual styles and layouts, export to many file formats, and embed plots in JupyterLab and graphical user interfaces. You can also use a rich array of third-party packages built on Matplotlib. Matplotlib was created by neurobiologist John Hunter to work with EEG data, and it has grown to be used in many other fields [111].



Figure 4.5: Matplotlib logo [111].

4.2.6 Gradio

Gradio is an open-source Python library used to create machine learning and data science demos, as well as web applications. With Gradio, you can quickly build a user-friendly interface around your machine learning models or data workflows, allowing users to interact with your demo by dragging and dropping images, pasting text, recording audio, and more, all through their web browser. Gradio is useful for rapidly deploying models with shareable links, gathering feedback on model performance, and debugging models interactively using built-in processing and interpretation tools [112].



Figure 4.6: Gradio logo [112].

4.2.7 NVIDIA

American-born NVIDIA is a multinational technology company with headquarters in Santa Clara, California. It is a leading manufacturer of graphics processing units (GPUs) and high-performance computer equipment. The GPU was developed by NVIDIA, which is well known for its innovative products and technologies. These include gaming, artificial intelligence, robotics, autonomous vehicles, and high-performance computing. On April 5, 1993, Jensen Huang, Chris Malachowsky, and Curtis Priem established Nvidia [113][114].



Figure 4.7: NVIDIA logo [113]

4.2.8 CUDA

CUDA (Compute Unified Device Architecture) is a parallel computing platform and API developed by Nvidia, enabling the use of GPUs for general-purpose computing. Launched in 2006, it supports programming languages such as C, C++, Fortran, and Python, facilitating accelerated processing. Key features include parallel computing, explicit memory management, and efficient thread management. Widely adopted in machine learning, scientific research, and other fields, CUDA's extensive tools and libraries are vital for enhancing computational performance [115][116][117][118].



Figure 4.8: CUDA logo[117]

4.2.9 CuDNN

cuDNN (CUDA Deep Neural Network) is a GPU-accelerated library by NVIDIA for deep learning, offering optimized implementations for convolution, attention, matmul, pooling, and normalization. It supports multi-threading and CUDA streams for enhanced perfor-

mance. NVIDIA provides extensive resources including installation guides, API references, a developer guide, and troubleshooting documentation to assist with using cuDNN effectively. These resources are available on the NVIDIA cuDNN documentation hub [119].



Figure 4.9: CuDNN logo [119]

4.2.10 NumPy

NumPy is a fundamental Python open-source project that provides robust numerical computing capabilities. Established in 2005, it extends the functionality of earlier libraries while maintaining a commitment to open accessibility. Governed by a Steering Council, NumPy promotes community collaboration via GitHub, with various teams dedicated to development, documentation, and optimization. Funding from foundations and institutional partners ensures its continued growth. NumPy's significance lies in its pivotal role in scientific computing, offering users a vast array of manipulation functions and mathematical operations, rendering it an indispensable tool for numerical analysis and data manipulation tasks [120].



Figure 4.10: Numpy logo [120]

4.2.11 Scikit-learn

An open-source machine learning library for the Python programming language is called Scikit-learn. David Cournapeau started the idea in 2007 as a Google Summer of Code project,

and Matthieu Brucher expanded on it for his thesis. The first public release occurred on February 1st, 2010, after INRIA's Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, and Vincent Michel assumed leadership in 2010. Numerous supervised and unsupervised learning methods for tasks like classification, statistical analysis, dimensionality reduction, and model selection are included in the collection. The NumFOCUS group provides financing for Scikit-learn, which is also funded by INRIA, Microsoft, and Nvidia. Donations are used to pay for organizational budget and other costs, such workshop travel [121].



Figure 4.11: Sklearn logo [121]

4.3 Realization

Before sending the final images to the following steps, we will do preprocessing operations on the dataset that was utilized for this project. Tumor binary classification will be denoted as "Pathogenic & Non-Pathogenic Detection" whereas multiclass classification will be called "EMs Detection".

4.3.1 Dataset description

The dataset used in this project is the EMDS-7, which includes 2,365 PNG images across 42 classes. We removed the "unknown" class because it was empty and since class 2 and class 11 share the same name and content, they have been combined. We can classify the objects into two categories: Pathogenic or Non-Pathogenic. Between 2018 and 2019, environmental biologists in Shenyang, China, took the photos from different lakes and rivers using a 400× optical microscope. From 2020 to 2021, bioinformatics experts manually annotated the items in XML format. [122].

4.3.1.1 First problematic

Table 4.1 provides the number of images for each category in the EMDS-7 dataset. Each row consists of the folder name, category name, and the number of images. This information is necessary to understand data distribution and organization for preprocessing, model training, and data analysis.

Folder Name	Class Name	Image Count
Pathogenic	Pathogenic	1063
Non-Pathogenic	Non-Pathogenic	1302

Table 4.1: Image Counts per Class in EMDS-7 (2 Classes)

The Figure 4.12 provides a graph illustrating the pathogenicity distribution.

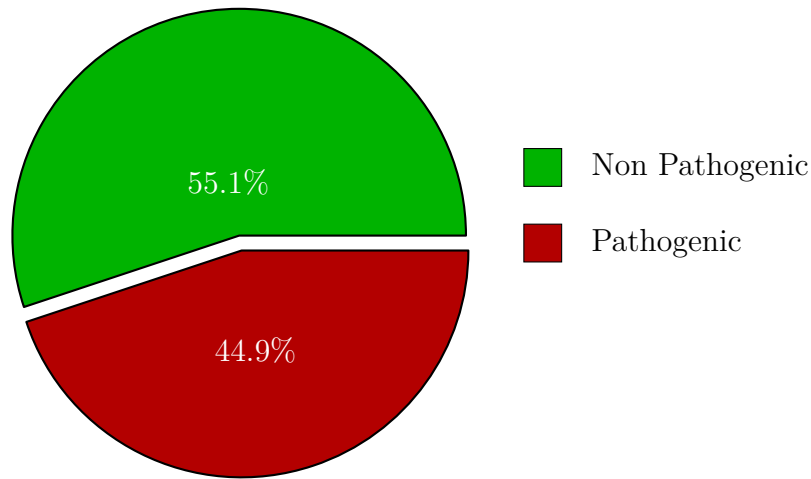


Figure 4.12: Pathogenicity Distribution

My database is stored on the Kaggle platform. The Figure 4.13 below illustrates the data organization as it is structured on Kaggle, facilitating effective information management and ease of access for various analyses.

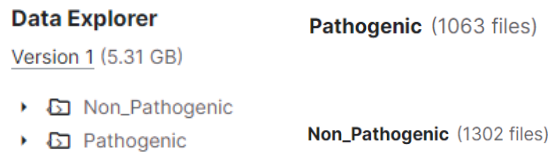


Figure 4.13: Data Structure, Pathogenic, and Non-Pathogenic.

4.3.1.2 Second problematic

The table 4.2 provides the count of images per class in the EMDS-7 dataset, along with information on whether each class is harmful (pathogenic) or not. The table is ordered to list all harmful classes first, followed by non-pathogenic ones.

Table 4.2: Image Counts per Class in EMDS-7

Index	Harmful	Folder Name	Class Name	Image Count
1	1	EMDS7-G001	Oscillatoria	41
2	1	EMDS7-G003	Microcystis	307
3	1	EMDS7-G014	Pediastrum	95
4	1	EMDS7-G021	Staurastrum	9
5	1	EMDS7-G022	Phormidium	276
6	1	EMDS7-G023	Fragilaria	55
7	1	EMDS7-G024	Anabaenopsis	22
8	1	EMDS7-G025	Coelosphaerium	77
9	1	EMDS7-G031	Merismopedia	33
10	1	EMDS7-G034	Raphidiopsis	9
11	1	EMDS7-G035	Gomphosphaeria	58
12	1	EMDS7-G036	Euglena	81
13	0	EMDS7-G002	Ankistrodesmus	69
14	0	EMDS7-G004	Gomphonema	87
15	0	EMDS7-G005	Sphaerocystis	55
16	0	EMDS7-G006	Cosmarium	17
17	0	EMDS7-G007	Cocconeis	14
18	0	EMDS7-G008	Tribonema	49
19	0	EMDS7-G009	Chlorella	80

Index	Harmful	Folder Name	Class Name	Image Count
20	0	EMDS7-G010	Tetraedron	25
21	0	EMDS7-G012	Brachionus	113
22	0	EMDS7-G013	Chaenea	6
23	0	EMDS7-G015	Spirulina	18
24	0	EMDS7-G016	Actinastrum	23
25	0	EMDS7-G017	Navicula	75
26	0	EMDS7-G018	Scenedesmus	86
27	0	EMDS7-G019	Golenkinia	60
28	0	EMDS7-G020	Pinnularia	36
29	0	EMDS7-G026	Crucigenia	9
30	0	EMDS7-G027	Achnanthes	18
31	0	EMDS7-G028	Synedra	77
32	0	EMDS7-G029	Ceratium	23
33	0	EMDS7-G030	Pompholyx	49
34	0	EMDS7-G032	Spirogyra	89
35	0	EMDS7-G033	Coelastrum	29
36	0	EMDS7-G037	Euclanis	14
37	0	EMDS7-G038	Keratella	65
38	0	EMDS7-G039	diversicornis	89
39	0	EMDS7-G040	Surirella	22
40	0	EMDS7-G041	Characium	5

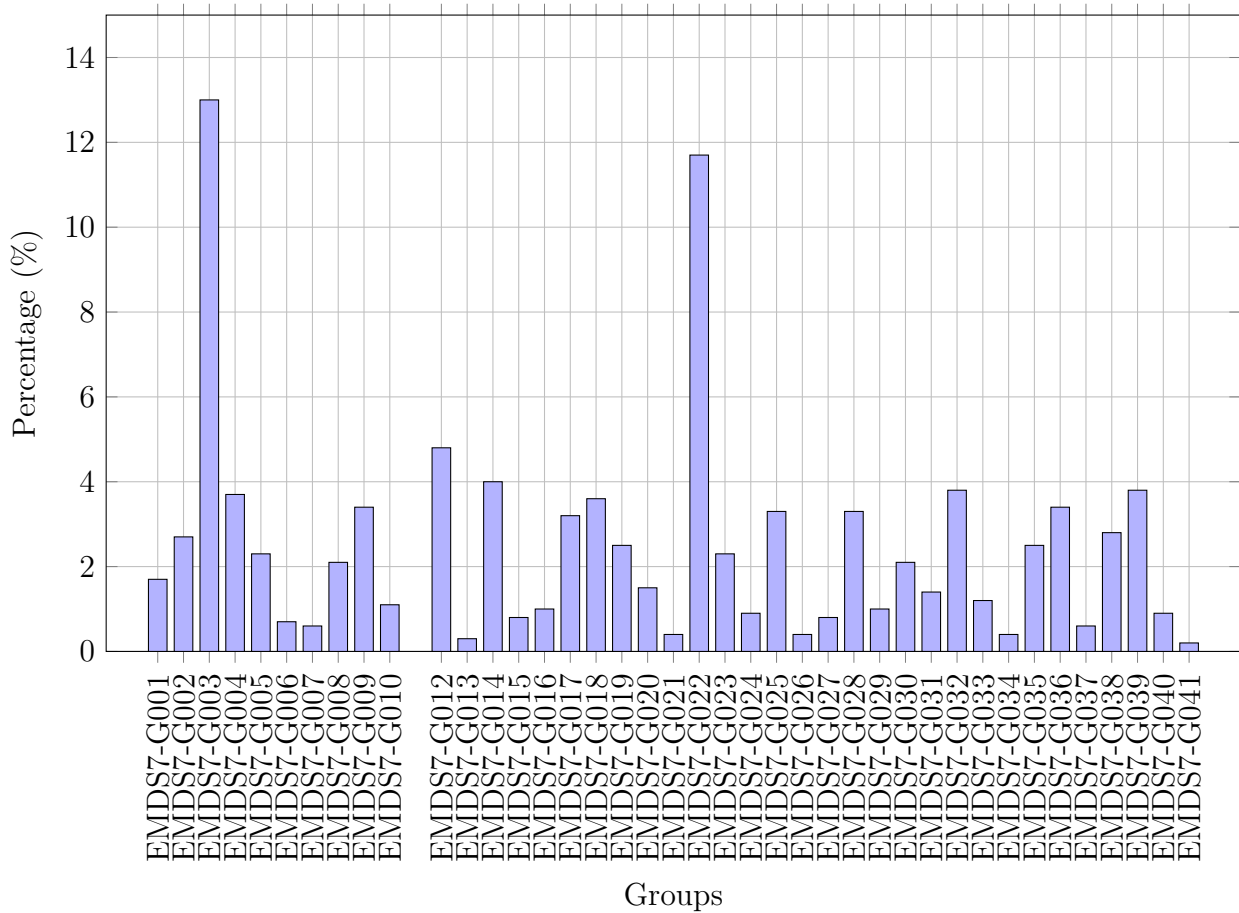


Figure 4.14: Percentage distribution of groups

4.3.2 Preprocessing steps & Split Dataset

The preprocessing steps and dataset splitting are common to both problematics. There is no difference in this stage; only the inputs and outputs vary according to the specific dataset for each problematic. To begin, we imported the necessary libraries in Python.

```

1 import numpy as np
2 import cv2
3 from PIL import Image
4 from torchvision import transforms
5

```

Figure 4.15: Python code snippet for importing libraries.

4.3.2.1 Data loading

in this phase is to load the images from the dataset directory using a custom dataset class. This method ensures that images are organized by class labels and can handle different formats and structures.

4.3.2.2 Resizing the Images

The first step in preprocessing is to resize the images to a uniform size. This is crucial as models require fixed input dimensions. We use `transforms.Resize()` from the `torchvision.transforms` module to resize the images. The syntax is:

```
1 # Define the target image size
2 image_size = (224, 224)
3
4 transforms.Resize(image_size)
5
```

Figure 4.16: Python code snippet for resizing images.

4.3.2.3 Normalization

Normalization adjusts the pixel values to a common scale, typically using the mean $([0.485, 0.456, 0.406])$ and standard deviation $([0.229, 0.224, 0.225])$ of the dataset. This step is important for efficient and stable training. We use `transforms.Normalize()` to standardize the pixel values. The syntax is:

```
1 #transforms.Normalize(mean, std)
2 transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
3
4
```

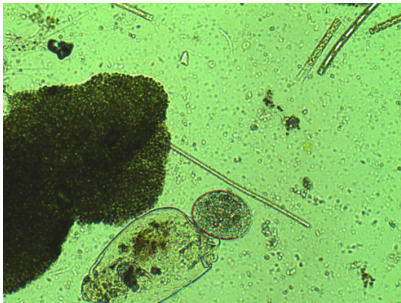
Figure 4.17: Python code snippet for normalization.

4.3.2.4 Split Dataset

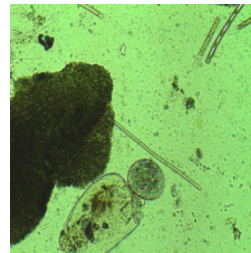
Splitting the dataset is a crucial step in preparing the data for model training and evaluation. Typically, the dataset is divided into training and testing sets to evaluate the model's performance on unseen data. Here, we use an 80-20 split, where 80% of the data is used for training and 20% for testing. The syntax is:

```
1 from torch.utils.data import random_split
2
3 # Split dataset into training and testing sets (80-20 split)
4 train_size = int(0.8 * len(dataset))
5 test_size = len(dataset) - train_size
6 train_dataset, test_dataset = random_split(dataset, [train_size, test_size])
7
```

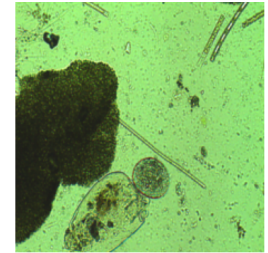
Figure 4.18: Python code snippet for splitting dataset.



(a) Original Image



(b) Resized



(c) Normalization

Figure 4.19: Results of preprocessing steps.

4.3.3 Data Augmentation

Data augmentation is an important technique to enhance the dataset by applying various transformations. Here, we describe and demonstrate the code for each augmentation method used. The difference lies in how the augmentation technique is implemented for the two problematics.

In the first problematic, as observed in Image 4.12, the distribution among classes is equal. Therefore, we applied each upcoming augmentation technique to every image and saved it.

However, in the second problematic, as illustrated in Image 4.14, the image distribution among classes is not consistent at all. Because of this, we had to adapt our augmentation strategy differently. For the second problematic, we applied specific augmentation techniques selectively based on the class distribution to ensure a more balanced dataset.

4.3.3.1 Horizontal Flip

This transformation flips the image horizontally with a probability of 1.0.

```
1 import torchvision.transforms as transforms
2
3 # Horizontal flip
4 horizontal_flip = transforms.RandomHorizontalFlip(p=1.0)
5
```

Figure 4.20: Python code snippet for horizontal flip.

4.3.3.2 Vertical Flip

This transformation flips the image vertically with a probability of 1.0.

```
1 import torchvision.transforms as transforms
2
3 # Vertical flip
4 vertical_flip = transforms.RandomVerticalFlip(p=1.0)
5
```

Figure 4.21: Python code snippet for vertical flip.

4.3.3.3 Rotation

This transformation rotates the image by 30 degrees.

```
1 import torchvision.transforms as transforms
2
3 # Rotation
4 rotation = transforms.RandomRotation(30)
5
```

Figure 4.22: Python code snippet for rotation.

4.3.3.4 Gaussian Blur

This transformation applies Gaussian blur with a kernel size of 3.

```
1 import torchvision.transforms as transforms
2
3 # Gaussian blur
4 gaussian_blur = transforms.GaussianBlur(3)
5
```

Figure 4.23: Python code snippet for Gaussian blur.

4.3.3.5 CLAHE

CLAHE (Contrast Limited Adaptive Histogram Equalization) is a custom transformation.

```
1 import cv2
2 import numpy as np
3 from PIL import Image
4
5 # Custom CLAHE transformation
6 class CLAHE:
7     def __init__(self, clip_limit=2.0, tile_grid_size=(8, 8)):
8         self.clip_limit = clip_limit
9         self.tile_grid_size = tile_grid_size
10
11     def __call__(self, img):
12         img = np.array(img)
13         if len(img.shape) == 2:
14             img = cv2.cvtColor(img, cv2.COLOR_GRAY2BGR)
15         lab = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)
16         lab_planes = list(cv2.split(lab))
17         clahe = cv2.createCLAHE(clipLimit=self.clip_limit, tileGridSize=self.tile_grid_size)
18         lab_planes[0] = clahe.apply(lab_planes[0])
19         lab = cv2.merge(lab_planes)
20         img = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)
21         return Image.fromarray(img)
22
23 clahe = CLAHE()
24
```

Figure 4.24: Python code snippet for CLAHE.

4.3.3.6 Cutout

Cutout is a custom transformation that masks out square regions of the input image.

```
1 import numpy as np
2 from PIL import Image
3
4 # Custom Cutout transformation
5 class Cutout:
6     def __init__(self, num_holes=1, size=50):
7         self.num_holes = num_holes
8         self.size = size
9
10    def __call__(self, img):
11        img = np.array(img)
12        h, w, _ = img.shape
13        mask = np.ones((h, w), np.float32)
14
15        for _ in range(self.num_holes):
16            y = np.random.randint(h)
17            x = np.random.randint(w)
18
19            y1 = np.clip(y - self.size // 2, 0, h)
20            y2 = np.clip(y + self.size // 2, 0, h)
21            x1 = np.clip(x - self.size // 2, 0, w)
22            x2 = np.clip(x + self.size // 2, 0, w)
23
24            mask[y1: y2, x1: x2] = 0
25
26        mask = np.expand_dims(mask, axis=-1)
27        img = img * mask
28        img = img.astype(np.uint8) # Convert to uint8
29        return Image.fromarray(img)
30
31 cutout = Cutout()
32
```

Figure 4.25: Python code snippet for Cutout.

The figure 4.26 below showcases the results obtained after applying the previously mentioned augmentation techniques to the original image.

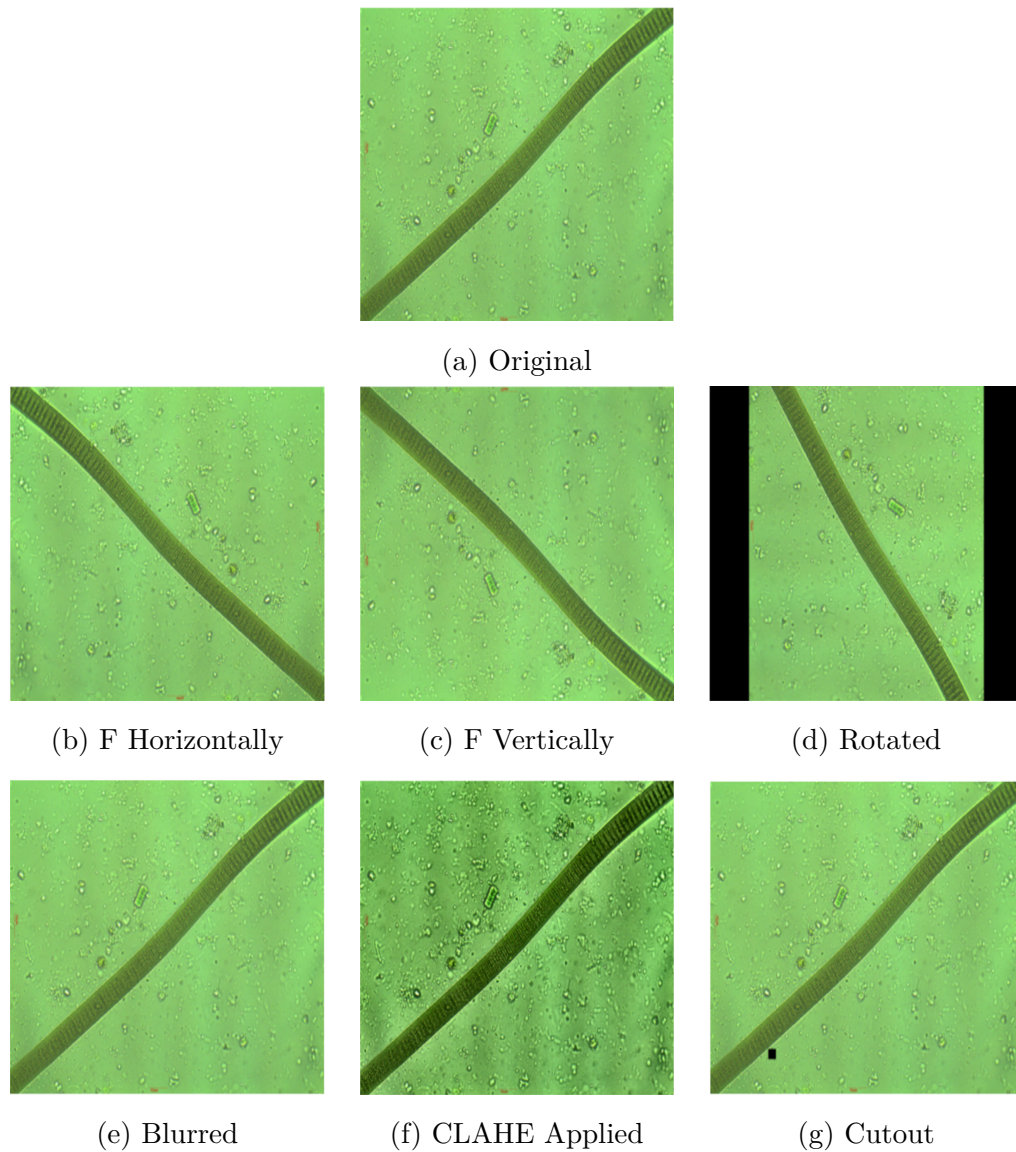


Figure 4.26: Original image and its augmented versions.

4.4 Modelisation

In this section, we will present the code used in the creation, training, and evaluation phases of the various models.

4.4.1 Pretrained Models

The PyTorch library includes models such as ResNet50, VGG19, and EfficientNet-B0 to assist with transfer learning. The final layer of the model (`_fc`) was adjusted to match the

number of classes in our dataset: 2 for the first problematic (Pathogenic & Non-Pathogenic Detection) and 40 for the second (EMs Detection).

Models built on PyTorch are trained using custom training loops. This process allows us to evaluate our model during the training phase. Here, we utilize the split data, specifically the training data, to adjust the model.

4.4.1.1 Model Setup

In this stage, we configure the device (GPU or CPU) to be used and load the pre-trained EfficientNet-B0 model. We modify the final layer of the model to align with the number of classes in our dataset (2 classes in this case). Finally, we move the model to the appropriate device.

```
1 # Model Setup
2 device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
3 model = EfficientNet.from_pretrained('efficientnet-b0') # Using EfficientNet-B0 with ImageNet weights
4 num_ftns = model._fc.in_features # Getting the number of input features for the final layer
5 model._fc = nn.Linear(num_ftns, 2) # Modifying the final layer to match the number of classes
6 model = model.to(device) # Moving the model to the appropriate device (GPU or CPU)
7
```

Figure 4.27: Setting up the model using EfficientNet-B0 with ImageNet weights.

4.4.1.2 Training

The parameters were selected to optimize the model's performance and ensure stable convergence during training.

Hyperparameters	Value
Loss Function	CrossEntropyLoss
Optimizer	SGD
Metrics	Accuracy
Epochs	100
Batch Size	32
Learning Rate	0.001
Momentum	0.9

Table 4.3: Hyperparameters used for training the VGG19 model.

Loss Function and Optimizer Setup: At this stage, we define the loss function as `CrossEntropyLoss` and use the `SGD` optimizer with a learning rate of 0.001 and momentum of 0.9.

```
1 criterion = nn.CrossEntropyLoss()
2 optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)
3
```

Figure 4.28: Setting up the loss function and optimizer.

Training Loop: In this stage, we define variables to store the training and validation accuracy and loss histories, and specify the best validation accuracy to track and save the best model.

```
1 num_epochs = 100
2 train_acc_history = []
3 train_loss_history = []
4 val_acc_history = []
5 val_loss_history = []
6
7 best_val_acc = 0.0
8 best_model_path = 'best_model.pth'
9
```

Figure 4.29: Setting up training loop variables.

Epoch Loop: In this stage, we iterate over epochs for both training and validation. In each epoch, we set the model to either train or evaluate mode based on the current phase.

```
1 for epoch in range(num_epochs):
2     print(f'Epoch {epoch}/{num_epochs - 1}')
3     print('-' * 10)
4
5     for phase in ['train', 'val']:
6         if phase == 'train':
7             model.train()
8             dataloader = dataloaders['train']
9         else:
10            model.eval()
11            dataloader = dataloaders['val']
12
```

Figure 4.30: Epoch loop for training and validation phases.

Batch Loop: In this stage, we define a loop to iterate over batches within each phase. We compute the loss, perform weight updates during training, and accumulate accuracy and loss metrics.

```
1     running_loss = 0.0
2     running_corrects = 0
3
4     for inputs, labels in dataloader:
5         inputs = inputs.to(device)
6         labels = labels.to(device)
7
8         optimizer.zero_grad()
9
10        with torch.set_grad_enabled(phase == 'train'):
11            outputs = model(inputs)
12            _, preds = torch.max(outputs, 1)
13            loss = criterion(outputs, labels)
14
15            if phase == 'train':
16                loss.backward()
17                optimizer.step()
18
19        running_loss += loss.item() * inputs.size(0)
20        running_corrects += torch.sum(preds == labels.data)
21
```

Figure 4.31: Batch loop for each phase within each epoch.

Calculating and Recording Metrics: In this stage, we calculate the loss and accuracy for each epoch and record them. If the validation accuracy is the best encountered so far, we save the model.

```
1     epoch_loss = running_loss / len(dataloader.dataset)
2     epoch_acc = running_corrects.double() / len(dataloader.dataset)
3
4     if phase == 'train':
5         train_loss_history.append(epoch_loss)
6         train_acc_history.append(epoch_acc)
7     else:
8         val_loss_history.append(epoch_loss)
9         val_acc_history.append(epoch_acc)
10
11    # Save the best model
12    if epoch_acc > best_val_acc:
13        best_val_acc = epoch_acc
14        torch.save(model.state_dict(), best_model_path)
15        print("Saved the best model with validation accuracy: {:.4f}".format(best_val_acc))
16
```

Figure 4.32: Calculating and recording metrics for each epoch phase.

4.4.2 Pretrained & ML Models

We used the pre-trained EfficientNet-B0 model as an example, but the same process will be applied to all models resulting from the previous stage. This process involved extracting features from the images and modifying the final layer to match the number of classes (2/40 depending on the problematic). We then used the outputs from the penultimate layer as

feature vectors for training the SVM model.

4.4.2.1 Feature Extraction

First, we load the pretrained EfficientNet-B0 model and prepare it for feature extraction. The model's last layer is adjusted to match the number of classes in our dataset.

```
1 # Model Setup
2 device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
3 model = EfficientNet.from_pretrained('efficientnet-b0')
4 num_fttrs = model._fc.in_features
5 model._fc = nn.Linear(num_fttrs, 40) # Adjust the last layer for 40 classes
6 model_path = '/kaggle/input/models40c/efficientnet-b0_0.9384.pth' # Replace with your model path
7 model.load_state_dict(torch.load(model_path))
8 model = model.to(device)
9 model.eval()
10
```

Figure 4.33: Loading and setting up the pretrained EfficientNet-B0 model.

Next, we define a function to extract features from the penultimate layer of the model. This function processes the inputs in batches and extracts features from the specified layer.

```
1 # Define a feature extraction function
2 def extract_features(model, dataloader):
3     model.eval()
4     features = []
5     labels = []
6
7     with torch.no_grad():
8         for inputs, targets in dataloader:
9             inputs = inputs.to(device)
10            targets = targets.to(device)
11
12            # Extract features from the penultimate layer
13            x = model.extract_features(inputs)
14            x = nn.functional.adaptive_avg_pool2d(x, 1).squeeze(-1).squeeze(-1)
15            features.extend(x.cpu().numpy())
16            labels.extend(targets.cpu().numpy())
17            print(f"Processed {len(features)} samples")
18
19        return np.array(features), np.array(labels)
20
21 # Extract features from training and test sets
22 print("Extracting features from training set...")
23 train_features, train_labels = extract_features(model, train_loader)
24 print("Extracting features from test set...")
25 test_features, test_labels = extract_features(model, test_loader)
26
```

Figure 4.34: Extracting features from training and test sets.

4.4.2.2 Training SVM Classifier

To improve the performance of the classifier, we first standardize the extracted features. We then define a parameter grid for the SVM and search for the best hyperparameters through cross-validation.

```
1 # Standardize features
2 scaler = StandardScaler()
3 train_features = scaler.fit_transform(train_features)
4 test_features = scaler.transform(test_features)
5
6 # Parameter grid for SVM
7 param_grid = {
8     'C': [0.1, 1, 10, 100, 1000],
9     'kernel': ['linear', 'rbf', 'poly'],
10    'degree': [2, 3, 4],
11    'gamma': ['scale', 'auto']
12 }
13
14 # Training and evaluating multiple SVM classifiers
15 best_accuracy = 0
16 best_params = None
17 best_model = None
18
19 for C in param_grid['C']:
20     for kernel in param_grid['kernel']:
21         for degree in param_grid['degree']:
22             for gamma in param_grid['gamma']:
23                 # Skip degrees for non-polynomial kernels
24                 if kernel != 'poly' and degree != param_grid['degree'][0]:
25                     continue
26
27                 print(f"Training SVM with C={C}, kernel={kernel}, degree={degree}, gamma={gamma}...")
28                 svm_classifier = svm.SVC(C=C, kernel=kernel, degree=degree, gamma=gamma)
29                 svm_classifier.fit(train_features, train_labels)
30
31                 print(f"Predicting on test set with SVM (C={C}, kernel={kernel}, degree={degree}, gamma={gamma})...")
32                 predictions = svm_classifier.predict(test_features)
33                 accuracy = accuracy_score(test_labels, predictions)
34                 print(f"Accuracy: {accuracy}")
35
36                 if accuracy > best_accuracy:
37                     best_accuracy = accuracy
38                     best_params = {'C': C, 'kernel': kernel, 'degree': degree, 'gamma': gamma}
39                     best_model = svm_classifier
40
```

Figure 4.35: Training and evaluating multiple SVM classifiers.

After identifying the best SVM model, we save it for future use.


```
1 # Save the best SVM model
2 joblib.dump(best_model, 'best_svm_model.pkl')
3 print("Best SVM model saved to 'best_svm_model.pkl'")
4
5 # Evaluation of the best model
6 print(f"Best Model Parameters: C={best_params['C']}, kernel={best_params['kernel']}, degree={best_params['degree']}, gamma
   = {best_params['gamma']}")
7 print(f"Best Model Accuracy: {best_accuracy}")
8
```

Figure 4.36: Saving the best SVM model.

4.4.3 ViT Model

This section describes the steps taken to fine-tune a pretrained Vision Transformer (ViT) model for a custom classification task.

4.4.3.1 Import Libraries and Setup Device

First, the necessary libraries for deep learning, data manipulation, and performance metrics are imported. The computing device is also set up.

```
1 import os
2 import matplotlib.pyplot as plt
3 import torch
4 import torchvision
5 from torch import nn
6 from torchvision import datasets, transforms
7 from torch.utils.data import DataLoader
8 import numpy as np
9 import seaborn as sns
10 from sklearn.metrics import classification_report, confusion_matrix, precision_score, recall_score
11 from tqdm.auto import tqdm
12 from torchinfo import summary
13
14 device = "cuda" if torch.cuda.is_available() else "cpu"
15
```

Figure 4.37: Importing Libraries and Setting up the Device

4.4.3.2 Load Pretrained ViT Model and Freeze Base Parameters

The pretrained ViT model with its default weights is loaded and transferred to the selected device. The base model parameters are then frozen to prevent them from being updated during training.

```
1 pretrained_vit_weights = torchvision.models.ViT_B_16_Weights.DEFAULT
2 pretrained_vit = torchvision.models.vit_b_16(weights=pretrained_vit_weights).to(device)
3
4 for parameter in pretrained_vit.parameters():
5     parameter.requires_grad = False
6
```

Figure 4.38: Loading Pretrained ViT Model and Freezing Base Parameters

4.4.3.3 Modify the Classifier Head

The classifier head of the ViT model is modified to match the number of classes in the custom dataset (`class_names`).

```
1 class_names = [f'EMDS7-G{i:03}' for i in range(1, 42) if i != 11]
2 pretrained_vit.heads = nn.Linear(in_features=768, out_features=len(class_names)).to(device)
3
```

Figure 4.39: Modifying the Classifier Head

4.4.3.4 Display Model Summary

The model architecture, including input and output sizes, number of parameters, and trainable parameters, is displayed using `torchinfo`.

```
1 summary(model=pretrained_vit,
2         input_size=(32, 3, 224, 224),
3         col_names=["input_size", "output_size", "num_params", "trainable"],
4         col_width=20,
5         row_settings=["var_names"])
6
```

Figure 4.40: Displaying Model Summary

4.4.3.5 Setup Data Transforms and Loaders

The data directories are set, and automatic transforms are retrieved from the pretrained ViT weights to preprocess the data consistently. The number of CPU cores available is also set.

```
1 train_dir = '/kaggle/input/finalds40c/EMDS7_Preprocessed/train'
2 test_dir = '/kaggle/input/finalds40c/EMDS7_Preprocessed/test'
3 pretrained_vit_transforms = pretrained_vit_weights.transforms()
4 print(pretrained_vit_transforms)
5 NUM_WORKERS = os.cpu_count()
6
```

Figure 4.41: Setting Up Data Transforms and Loaders

4.4.3.6 Create Dataloaders

PyTorch dataloaders for the training and testing datasets are created, applying the necessary transformations and setting batch size and worker count.

```
1 def create_dataloaders(train_dir: str, test_dir: str, transform: transforms.Compose, batch_size: int, num_workers: int =
   NUM_WORKERS):
2     train_data = datasets.ImageFolder(train_dir, transform=transform)
3     test_data = datasets.ImageFolder(test_dir, transform=transform)
4     class_names = train_data.classes
5     train_dataloader = DataLoader(train_data, batch_size=batch_size, shuffle=True, num_workers=num_workers, pin_memory=True
   )
6     test_dataloader = DataLoader(test_data, batch_size=batch_size, shuffle=False, num_workers=num_workers, pin_memory=True)
7     return train_dataloader, test_dataloader, class_names
8
9 train_dataloader_pretrained, test_dataloader_pretrained, class_names = create_dataloaders(train_dir=train_dir, test_dir=
   test_dir, transform=pretrained_vit_transforms, batch_size=32)
10
```

Figure 4.42: Creating Dataloaders

4.4.3.7 Define Training and Evaluation Functions

The training (`train_step`) and testing (`test_step`) procedures are defined. `train_step` handles the forward pass, loss computation, backward pass, and parameter updates, while `test_step` evaluates the model's performance on the test set without updating parameters.

```
1 def train_step(model, dataloader, loss_fn, optimizer, device):
2     model.train()
3     train_loss, train_acc = 0, 0
4     for batch, (X, y) in enumerate(dataloader):
5         X, y = X.to(device), y.to(device)
6         y_pred = model(X)
7         loss = loss_fn(y_pred, y)
8         train_loss += loss.item()
9         optimizer.zero_grad()
10        loss.backward()
11        optimizer.step()
12        y_pred_class = torch.argmax(torch.softmax(y_pred, dim=1), dim=1)
13        train_acc += (y_pred_class == y).sum().item() / len(y_pred)
14    train_loss = train_loss / len(dataloader)
15    train_acc = train_acc / len(dataloader)
16    return train_loss, train_acc
17
18 def test_step(model, dataloader, loss_fn, device):
19     model.eval()
20     test_loss, test_acc = 0, 0
21     with torch.inference_mode():
22         for batch, (X, y) in enumerate(dataloader):
23             X, y = X.to(device), y.to(device)
24             test_pred_logits = model(X)
25             loss = loss_fn(test_pred_logits, y)
26             test_loss += loss.item()
27             test_pred_labels = test_pred_logits.argmax(dim=1)
28             test_acc += ((test_pred_labels == y).sum().item() / len(test_pred_labels))
29     test_loss = test_loss / len(dataloader)
30     test_acc = test_acc / len(dataloader)
31     return test_loss, test_acc
32
```

Figure 4.43: Defining Training and Evaluation Functions

4.4.3.8 Training Loop

The training and validation process is orchestrated over a specified number of epochs, saving the best model based on validation accuracy. The loss and accuracy for both training and validation sets are stored and printed after each epoch.

```
1 def train(model, train_dataloader, test_dataloader, optimizer, loss_fn, epochs, device):
2     results = {"train_loss": [], "train_acc": [], "test_loss": [], "test_acc": []}
3     best_accuracy = 0.0
4     best_model_path = 'best_model.pth'
5     for epoch in tqdm(range(epochs)):
6         train_loss, train_acc = train_step(model, train_dataloader, loss_fn, optimizer, device)
7         test_loss, test_acc = test_step(model, test_dataloader, loss_fn, device)
8         if test_acc > best_accuracy:
9             torch.save(model.state_dict(), best_model_path)
10            best_accuracy = test_acc
11            results["train_loss"].append(train_loss)
12            results["train_acc"].append(train_acc)
13            results["test_loss"].append(test_loss)
14            results["test_acc"].append(test_acc)
15            print(f"Epoch {epoch+1}/{epochs}")
16            print(f"Train Loss: {train_loss:.4f}, Train Accuracy: {train_acc:.4f}")
17            print(f"Validation Loss: {test_loss:.4f}, Validation Accuracy: {test_acc:.4f}\n")
18    return results
19
```

Figure 4.44: Training Loop

4.4.3.9 Create Optimizer and Loss Function

The Adam optimizer is created to update model parameters, and CrossEntropyLoss is used as the loss function.

```
1 optimizer = torch.optim.Adam(params=pretrained_vit.parameters(), lr=1e-3)
2 loss_fn = torch.nn.CrossEntropyLoss()
3
```

Figure 4.45: Creating Optimizer and Loss Function

4.4.3.10 Train the Model

The training process is started using the `train` function defined earlier.

```
1 pretrained_vit_results = train(model=pretrained_vit, train_dataloader=train_dataloader_pretrained, test_dataloader=
2     test_dataloader_pretrained, optimizer=optimizer, loss_fn=loss_fn, epochs=50, device=device)
```

Figure 4.46: Training the Model

4.4.4 Evaluating the Models

Evaluation is a common stage across all types of models. In the following sections, we will explain the evaluation process we performed, along with the code used to implement it.

Loading the Best Model

After training, we load the best model saved during the training process.

```
1 # Load the best model
2 best_model = EfficientNet.from_name('efficientnet-b0')
3 num_ftrs = best_model._fc.in_features
4 best_model._fc = nn.Linear(num_ftrs, 2)
5 best_model.load_state_dict(torch.load(best_model_path))
6 best_model = best_model.to(device)
7
```

Figure 4.47: Loading the best model from the training phase.

Plotting Metrics

Visualize the training and validation accuracy and loss over epochs.

```
1 # Plot training and validation accuracy and loss
2 plt.figure(figsize=(12, 4))
3 plt.subplot(1, 2, 1)
4 plt.plot(train_acc_history_cpu, label='Training Accuracy')
5 plt.plot(val_acc_history_cpu, label='Validation Accuracy')
6 plt.xlabel('Epoch')
7 plt.ylabel('Accuracy')
8 plt.legend()
9 plt.subplot(1, 2, 2)
10 plt.plot(train_loss_history_cpu, label='Training Loss')
11 plt.plot(val_loss_history_cpu, label='Validation Loss')
12 plt.xlabel('Epoch')
13 plt.ylabel('Loss')
14 plt.legend()
15 plt.savefig('training_validation_metrics.jpg')
16 plt.show()
17
```

Figure 4.48: Plotting training and validation accuracy and loss.

Evaluate the model on the validation set which in at the same time test set and compute the test accuracy.

```
1 # Evaluate the model
2 best_model.eval()
3 test_corrects = 0
4 all_labels = []
5 all_preds = []
6
7 for inputs, labels in dataloaders['val']:
8     inputs = inputs.to(device)
9     labels = labels.to(device)
10
11     outputs = best_model(inputs)
12     _, preds = torch.max(outputs, 1)
13     test_corrects += torch.sum(preds == labels.data)
14     all_labels.extend(labels.cpu().numpy())
15     all_preds.extend(preds.cpu().numpy())
16
17 test_acc = test_corrects.double() / len(dataloaders['val'].dataset)
18 print(f'Test Acc: {test_acc:.4f}')
19
```

Figure 4.49: Model evaluation on the validation set.

Generating the confusion matrix

Compute the confusion matrix to evaluate model performance.

```
1 # Confusion Matrix
2 cm = confusion_matrix(all_labels, all_preds)
3 class_names = train_dataset.classes
4
```

Figure 4.50: Generating the confusion matrix.

Visualizing the confusion matrix

Visualize the confusion matrix using a heatmap.

```
1 # Visualize the Confusion Matrix
2 plt.figure(figsize=(8, 6))
3 sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_names, yticklabels=class_names)
4 plt.ylabel('Actual')
5 plt.xlabel('Predicted')
6 plt.title('Confusion Matrix')
7 plt.savefig(f'confusion_matrix_{test_acc:.4f}.jpg')
8 plt.show()
9
```

Figure 4.51: Visualizing the confusion matrix using a heatmap.

Printing the Classification Report

Print the classification report for detailed performance metrics.

```
1 # Classification Report
2 print('Classification Report')
3 print(classification_report(all_labels, all_preds, target_names=class_names))
4
```

Figure 4.52: Printing the classification report.

4.5 Results and discussion

Model accuracy rates, validation precision rates, and changes in loss functions were analyzed and recorded at every step of the training process. In each investigation, these modifications were looked at independently for each model, for each problematic.

4.5.1 Pretrained Models

After training the models for 100 epochs, the resulting metrics were evaluated by testing the system's reliability using the test dataset. The evaluation includes a confusion matrix illustrating the model's performance in data classification, alongside various performance metrics such as precision, recall, and F1 score. These results are presented based on the two problematics as shown below:

4.5.1.1 EfficientNet-B0

The EfficientNet-B0 architecture was chosen in this study for several reasons related to its efficiency and balanced performance. EfficientNet-B0 is considered a resource-efficient model, offering an excellent balance between accuracy and computational complexity, making it capable of achieving high-quality results with fewer resources. This makes it an ideal choice for handling small datasets, as it can achieve remarkable performance without intensive computations.

The following illustrates how its application to our dataset produced the desired results:

First problematic

The confusion matrix (Figure 4.53) provides a detailed breakdown of the model's performance in classifying images into two categories: pathogenic and non-pathogenic. The matrix shows that the model correctly classified 255 instances as non-pathogenic and 207 instances as pathogenic. However, there were some misclassifications: 6 pathogenic instances were incorrectly classified as non-pathogenic, and 5 non-pathogenic instances were incorrectly classified as pathogenic.

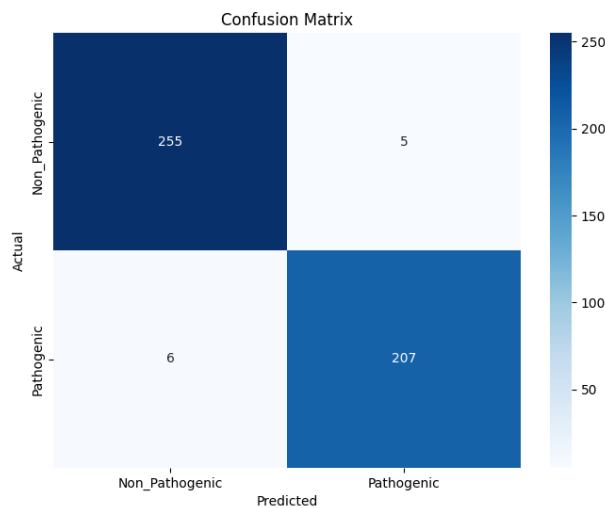


Figure 4.53: Confusion matrix showing the performance of the EfficientNet-B0 model (2 classes).

The left graph in Figure 4.54 shows the model's accuracy for each epoch. We observe that the accuracy increases rapidly at the beginning and then stabilizes at around 99.7% in training and about 95% in validation. The highest accuracy reached is approximately 97.67%.

The right graph shows the model's loss for each epoch. We observe that the loss decreases rapidly at the beginning and stabilizes at a very low value in training (around 0.005) and is slightly higher in validation (around 0.25), indicating that the model learns effectively but experiences some variance between training and validation performance.

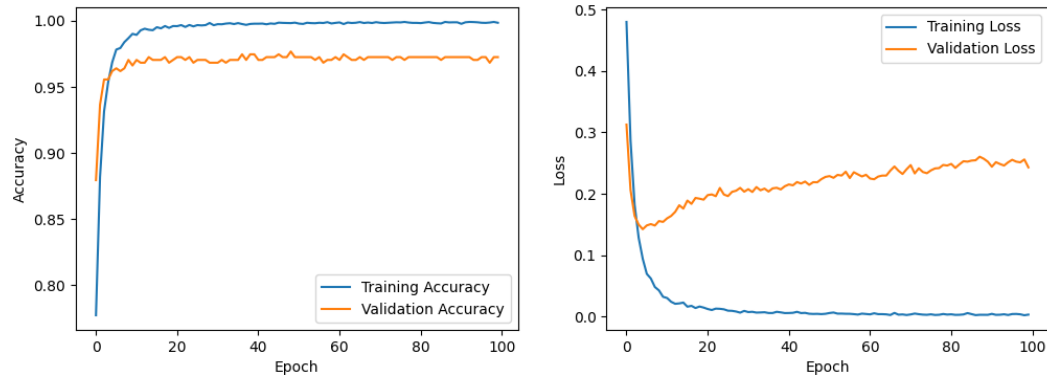


Figure 4.54: Accuracy and Loss Curves During Training and Validation Phases of EfficientNet-B0 (2 classes).

This table (Table 4.4) summarizes the performance metrics for the "Non_Pathogenic" and "Pathogenic" classes in terms of precision, recall, F1-score, and support.

Class	Precision	Recall	F1-Score	Support
Non_Pathogenic	0.98	0.98	0.98	260
Pathogenic	0.97	0.97	0.97	213
Accuracy	0.98 (473)			

Table 4.4: Classification Report EfficientNet-B0 (2 classes).

These results demonstrate the effectiveness and efficiency of the EfficientNet-B0 model in data classification, confirming its merit as an excellent choice for AI applications that require a balance between accuracy and resource utilization.

Second problematic

The confusion matrix (Figure 4.55) provides a detailed breakdown of the model’s performance in classifying images into multiple categories. The matrix shows the number of correctly classified instances and misclassifications for each category.

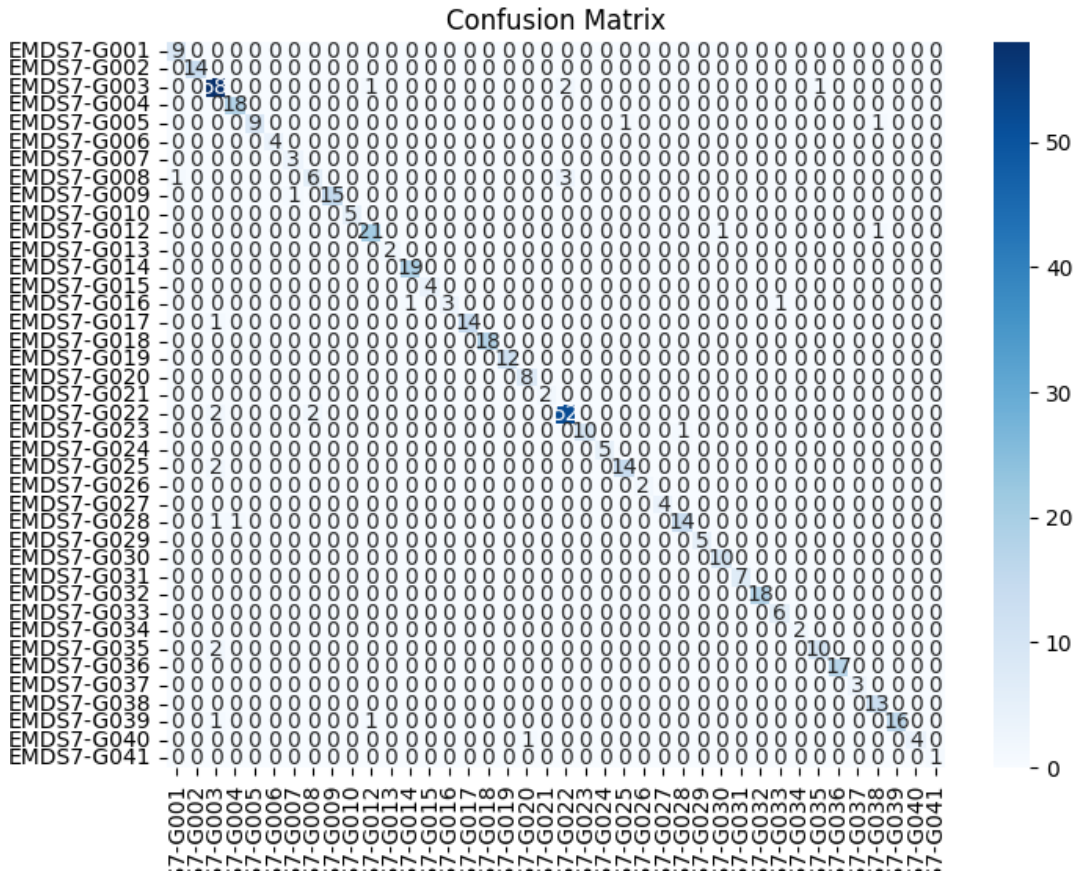


Figure 4.55: Confusion matrix showing the performance of the EfficientNet-B0 model (40 classes).

The table below (Table 4.5) provides more details for comprehending the confusion matrix, showing precision, recall, f1-score, and support for each category.

Category	Precision	Recall	F1-Score	Support
EMDS7-G001	1.00	1.00	1.00	9
EMDS7-G002	0.93	1.00	0.97	14
EMDS7-G003	0.91	0.95	0.93	62
EMDS7-G004	0.95	1.00	0.97	18
EMDS7-G005	1.00	0.82	0.90	11
EMDS7-G006	1.00	1.00	1.00	4

Category	Precision	Recall	F1-Score	Support
EMDS7-G007	0.75	1.00	0.86	3
EMDS7-G008	0.78	0.70	0.74	10
EMDS7-G009	1.00	0.88	0.93	16
EMDS7-G010	1.00	1.00	1.00	5
EMDS7-G012	0.88	0.96	0.92	23
EMDS7-G013	1.00	1.00	1.00	2
EMDS7-G014	0.95	1.00	0.97	19
EMDS7-G015	0.80	1.00	0.89	4
EMDS7-G016	1.00	0.80	0.89	5
EMDS7-G017	0.93	0.93	0.93	15
EMDS7-G018	1.00	0.89	0.94	18
EMDS7-G019	0.92	1.00	0.96	12
EMDS7-G020	0.89	1.00	0.94	8
EMDS7-G021	1.00	1.00	1.00	2
EMDS7-G022	0.95	0.93	0.94	56
EMDS7-G023	1.00	0.91	0.95	11
EMDS7-G024	0.83	1.00	0.91	5
EMDS7-G025	0.93	0.88	0.90	16
EMDS7-G026	1.00	1.00	1.00	2
EMDS7-G027	1.00	1.00	1.00	4
EMDS7-G028	0.93	0.81	0.87	16
EMDS7-G029	1.00	1.00	1.00	5
EMDS7-G030	0.91	1.00	0.95	10
EMDS7-G031	1.00	1.00	1.00	7
EMDS7-G032	1.00	1.00	1.00	18
EMDS7-G033	0.86	1.00	0.92	6
EMDS7-G034	1.00	1.00	1.00	2
EMDS7-G035	0.92	0.92	0.92	12
EMDS7-G036	1.00	1.00	1.00	17
EMDS7-G037	1.00	1.00	1.00	3
EMDS7-G038	0.87	1.00	0.93	13
EMDS7-G039	1.00	0.89	0.94	18

Category	Precision	Recall	F1-Score	Support
EMDS7-G040	1.00	0.60	0.75	5
EMDS7-G041	1.00	1.00	1.00	1
accuracy	-	-	0.94	487

Table 4.5: Classification Report Detailing Precision, Recall, F1-Score, and Support of the EfficientNet-B0 Model (40 classes)

The left graph in Figure 4.56 shows the model’s accuracy for each epoch. We observe that the accuracy increases rapidly at the beginning and then stabilizes at around 99.7% in training and about 93% in validation. The highest accuracy reached is approximately 93.84%.

The right graph shows the model’s loss for each epoch. We observe that the loss decreases rapidly at the beginning and stabilizes at a very low value in training (around 0.005) and is slightly higher in validation (around 0.30), indicating that the model learns effectively but experiences some variance between training and validation performance.

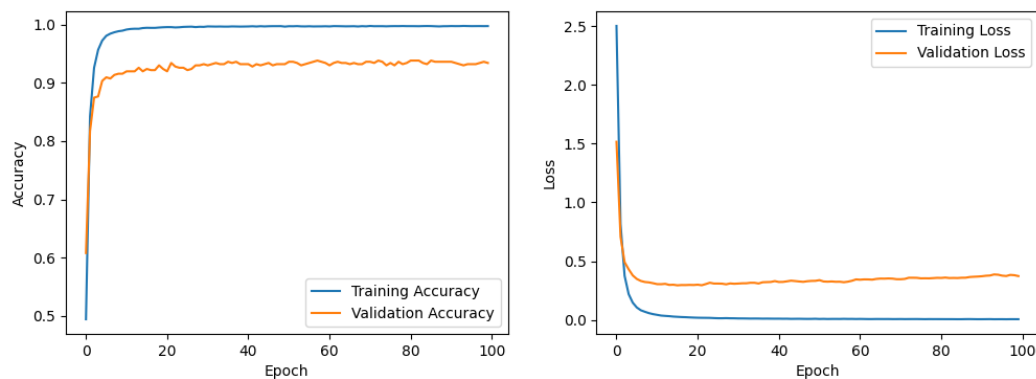


Figure 4.56: Accuracy and Loss Curves During Training and Validation Phases of the EfficientNet-B0 Model (40 classes).

4.5.1.2 ResNet-50

ResNet-50 is another deep learning model that is in use. This architecture was chosen because ResNet-50 offers a novel solution to the vanishing gradients problem. ResNet-50 avoids these layers by stacking several identity mappings, which are convolutional layers that do nothing at first, and reuses activations from the layer before it.

The results of ResNet-50 are clearly shown below:

First problematic

The confusion matrix (Figure 4.57) is very close to that of EfficientNet-B0 (Figure 4.53). The difference is that the current model correctly classified 208 cases as pathogenic compared to 207 in EfficientNet-B0, and correctly classified 254 cases as non-pathogenic compared to 255 in EfficientNet-B0. While the number of misclassifications of non-pathogenic cases as a Pathogenic remained the same (5 cases), as did the number of misclassifications of Pathogenic cases as Non-Pathogenic (6 cases).

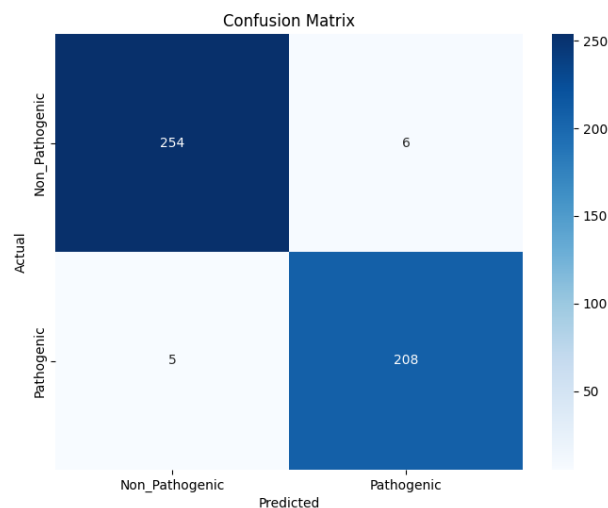


Figure 4.57: Confusion matrix showing the performance of the ResNet-50 model (2 classes).

The Figure 4.58 shows the training and validation accuracy and loss curves for the model over 100 epochs. The training accuracy (blue line in the left plot) quickly reaches around 100% and remains stable, while the validation accuracy (orange line) reaches around 96-97% with minor fluctuations. Regarding loss, the training loss (blue line in the right plot) quickly drops to very low levels close to zero and remains stable, whereas the validation loss (orange line) fluctuates significantly. This indicates some variability in performance on the validation set, which may suggest issues such as overfitting or variability in the validation data set.

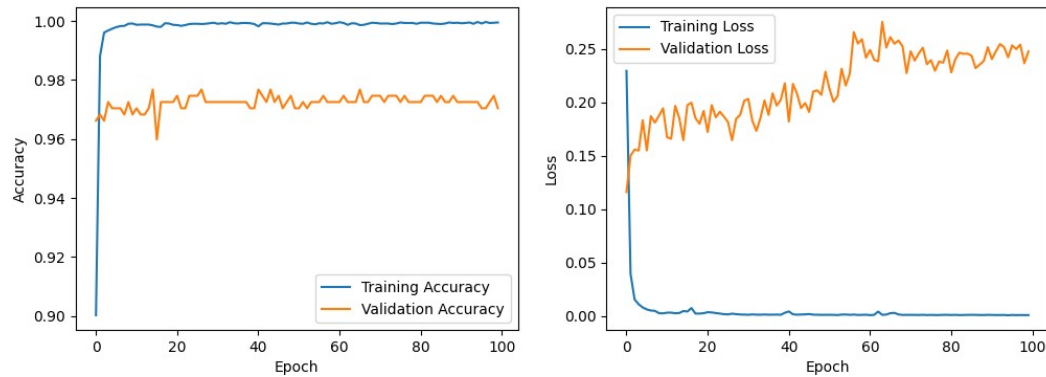


Figure 4.58: Accuracy and Loss Curves During Training and Validation Phases of ResNet-50 (2 classes).

The accuracy, recall, F1-score, and support performance metrics for the "Non_Pathogenic" and "Pathogenic" classes are compiled in this table:

Class	Precision	Recall	F1-Score	Support
Non_Pathogenic	0.98	0.98	0.98	260
Pathogenic	0.97	0.98	0.97	213
Accuracy	0.98 (473)			

Table 4.6: Classification Report ResNet-50 (2 classes).

Second problematic

Figure 4.59 presents a comprehensive analysis of the model's performance in categorizing images into several groups. The number of correctly categorized and misclassified incidents for each category is displayed in the matrix.

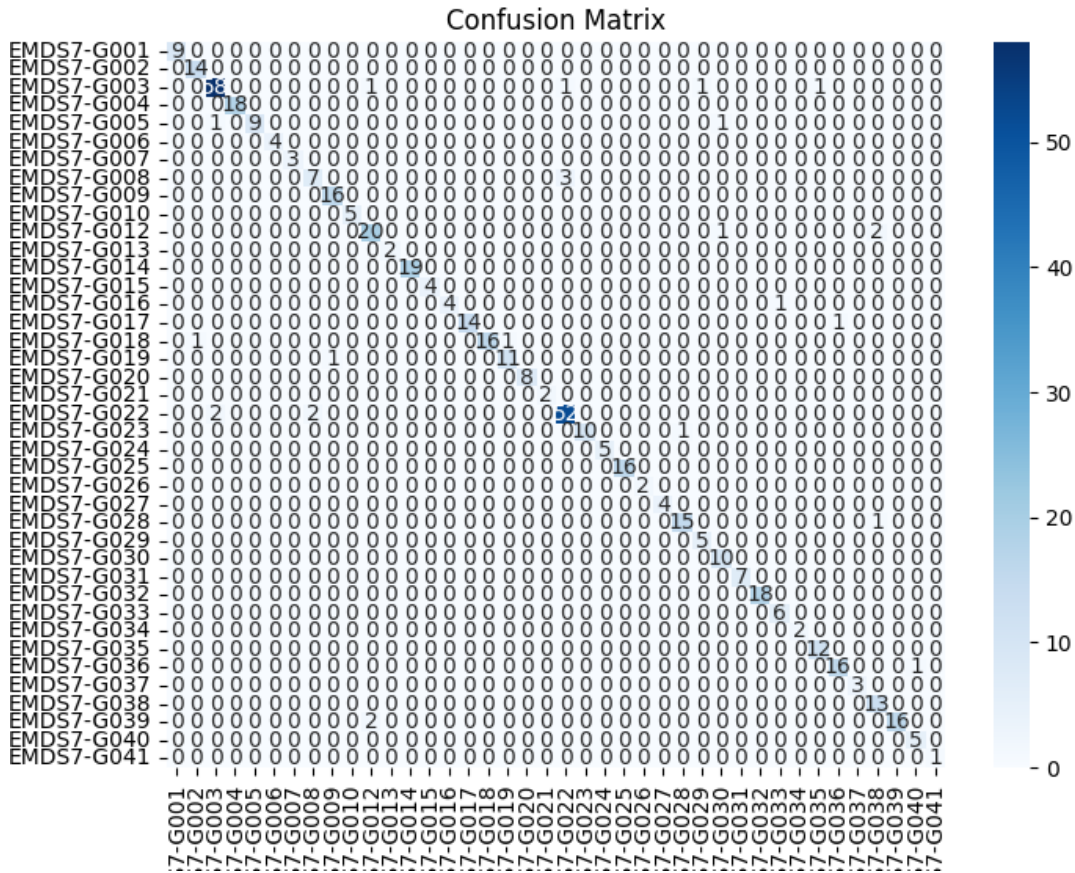


Figure 4.59: Confusion matrix showing the performance of the ResNet-50 model (40 classes).

More information on understanding the confusion matrix can be found in the table below (Table 4.7), which displays the precision, recall, f1-score, and support for each category.

Category	Precision	Recall	F1-Score	Support
EMDS7-G001	1.00	1.00	1.00	9
EMDS7-G002	0.93	1.00	0.97	14
EMDS7-G003	0.95	0.94	0.94	62
EMDS7-G004	1.00	1.00	1.00	18
EMDS7-G005	1.00	0.82	0.90	11
EMDS7-G006	1.00	1.00	1.00	4
EMDS7-G007	1.00	1.00	1.00	3

Category	Precision	Recall	F1-Score	Support
EMDS7-G008	0.78	0.70	0.74	10
EMDS7-G009	0.94	1.00	0.97	16
EMDS7-G010	1.00	1.00	1.00	5
EMDS7-G012	0.87	0.87	0.87	23
EMDS7-G013	1.00	1.00	1.00	2
EMDS7-G014	1.00	1.00	1.00	19
EMDS7-G015	1.00	1.00	1.00	4
EMDS7-G016	1.00	0.80	0.89	5
EMDS7-G017	1.00	0.93	0.97	15
EMDS7-G018	1.00	0.89	0.94	18
EMDS7-G019	0.92	0.92	0.92	12
EMDS7-G020	1.00	1.00	1.00	8
EMDS7-G021	1.00	1.00	1.00	2
EMDS7-G022	0.93	0.93	0.93	56
EMDS7-G023	1.00	0.91	0.95	11
EMDS7-G024	1.00	1.00	1.00	5
EMDS7-G025	1.00	1.00	1.00	16
EMDS7-G026	1.00	1.00	1.00	2
EMDS7-G027	1.00	1.00	1.00	4
EMDS7-G028	0.94	0.94	0.94	16
EMDS7-G029	0.83	1.00	0.91	5
EMDS7-G030	0.83	1.00	0.91	10
EMDS7-G031	1.00	1.00	1.00	7
EMDS7-G032	1.00	1.00	1.00	18
EMDS7-G033	0.86	1.00	0.92	6
EMDS7-G034	1.00	1.00	1.00	2
EMDS7-G035	0.92	1.00	0.96	12
EMDS7-G036	0.94	0.94	0.94	17
EMDS7-G037	1.00	1.00	1.00	3
EMDS7-G038	0.81	1.00	0.90	13
EMDS7-G039	1.00	0.89	0.94	18
EMDS7-G040	0.83	1.00	0.91	5

Category	Precision	Recall	F1-Score	Support
EMDS7-G041	1.00	1.00	1.00	1
accuracy	-	-	0.95	487

Table 4.7: Classification Report Detailing Precision, Recall, F1-Score, and Support of the EfficientNet-B0 Model (40 classes)

The Figure 4.60 shows the training and validation accuracy and loss curves for the model over 100 epochs. The training accuracy (blue line in the left plot) quickly reaches around 100% and remains stable, while the validation accuracy (orange line) stabilizes around 94.6% with minor fluctuations. Regarding loss, the training loss (blue line in the right plot) quickly drops to very low levels close to zero and remains stable, whereas the validation loss (orange line) fluctuates significantly between 0.2 and 0.3.

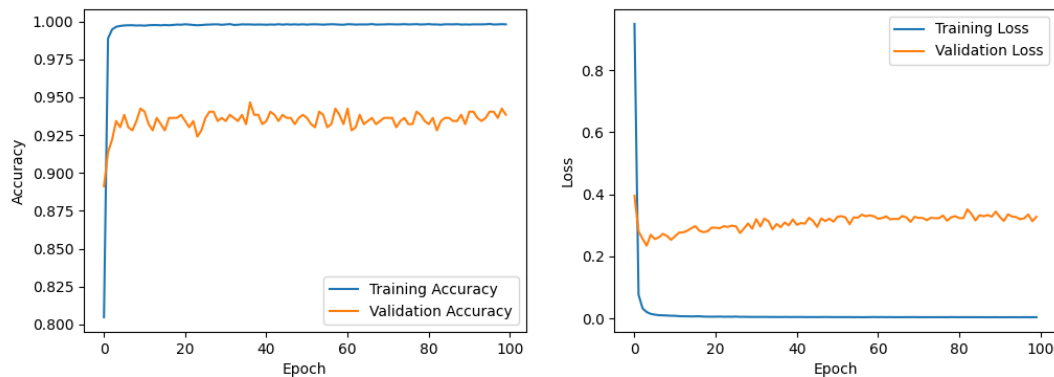


Figure 4.60: Accuracy and Loss Curves During Training and Validation Phases of the ResNet-50 Model (40 classes).

4.5.1.3 VGG19

Because of its deep and reliable architecture for extracting features from images, the VGG19 model was selected. The reason behind VGG19's reputation at handling fine details in photos is because it makes use of small (3x3) convolutional layers, which enable it to extract intricate patterns from the data. VGG19 is a great option for challenging classification problems since it has demonstrated its high efficacy in multiple contests and real-world computer vision applications. The following clearly displays the VGG19 results:

First problematic

The confusion matrix (Figure 4.61) illustrates the performance of the VGG19 model. This model shows a strong performance, similar to the other models discussed. In this case, the VGG19 model correctly classified 256 cases as non-pathogenic and 206 cases as pathogenic. The number of misclassifications of non-pathogenic cases as pathogenic was 4, while the number of misclassifications of pathogenic cases as non-pathogenic was 7.

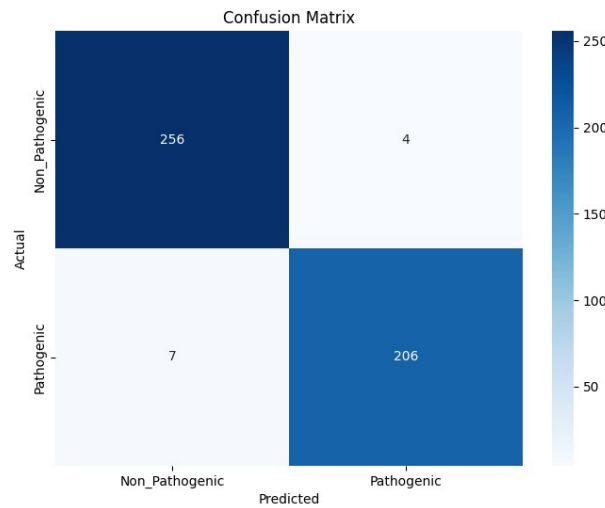


Figure 4.61: Confusion matrix showing the performance of the VGG19 model (2 classes).

The Figure 4.62 shows the training and validation accuracy and loss curves for the model over 100 epochs. The training accuracy (blue line in the left plot) quickly reaches around 100% and remains stable, while the validation accuracy (orange line) reaches around 96-97% with minor fluctuations. Regarding loss, the training loss (blue line in the right plot) quickly drops to very low levels close to zero and remains stable, whereas the validation loss (orange line) fluctuates significantly. This indicates some variability in performance on the validation set, which may suggest issues such as overfitting or variability in the validation data set.

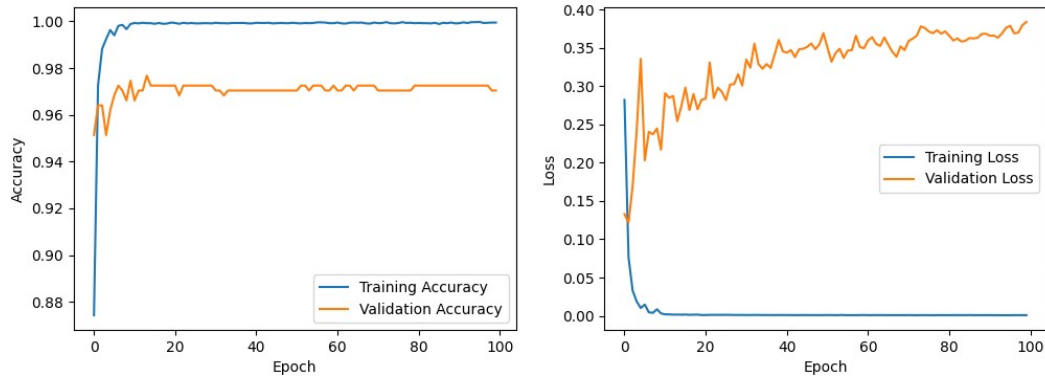


Figure 4.62: Accuracy and Loss Curves During Training and Validation Phases of VGG19 (2 classes).

The accuracy, recall, F1-score, and support performance metrics for the "Non_Pathogenic" and "Pathogenic" classes are compiled in this table:

Class	Precision	Recall	F1-Score	Support
Non_Pathogenic	0.97	0.98	0.98	260
Pathogenic	0.98	0.96	0.97	213
Accuracy	0.97 (473)			

Table 4.8: Classification Report VGG19 (2 classes).

Second problematic

Figure 4.63 presents a comprehensive analysis of the VGG19 model's performance in categorizing images into several groups. The number of correctly categorized and misclassified incidents for each category is displayed in the matrix.

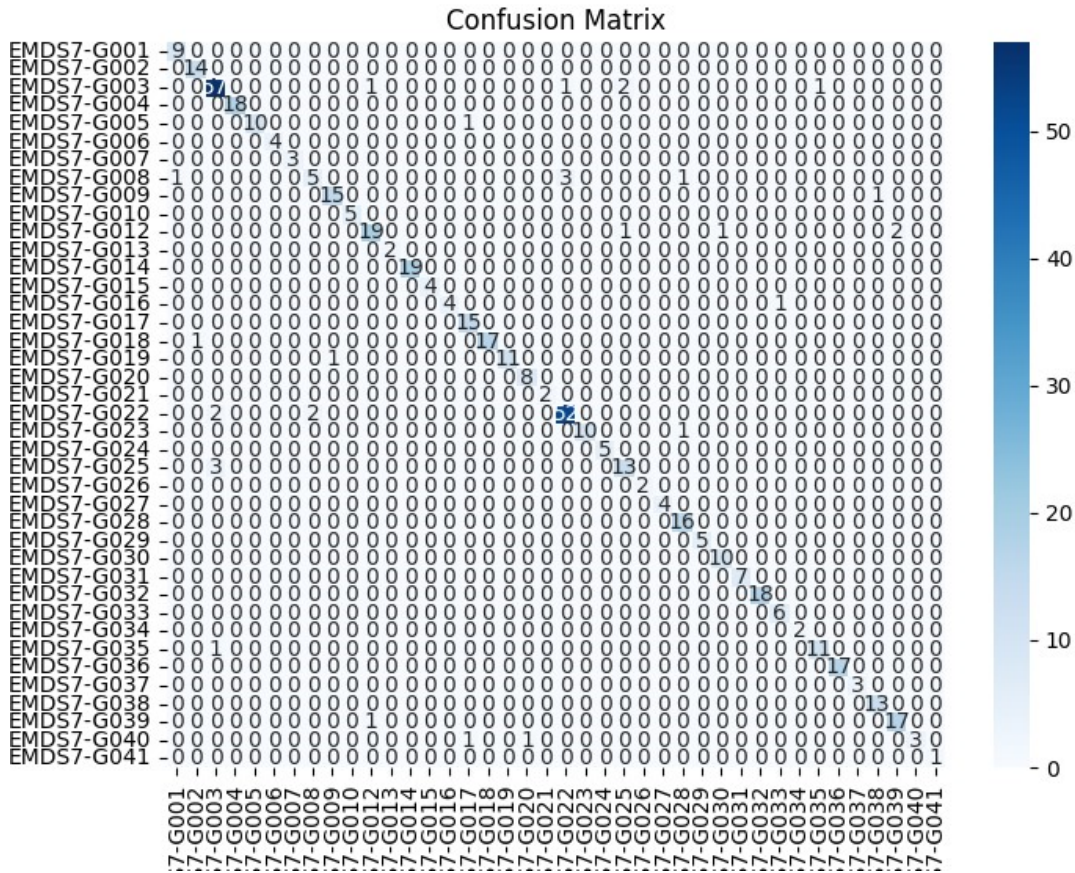


Figure 4.63: Confusion matrix showing the performance of the VGG19 model (40 classes).

More information on understanding the confusion matrix can be found in the table below (Table 4.9), which displays the precision, recall, f1-score, and support for each category.

Category	Precision	Recall	F1-Score	Support
EMDS7-G001	0.90	1.00	0.95	9
EMDS7-G002	0.93	1.00	0.97	14
EMDS7-G003	0.90	0.92	0.91	62
EMDS7-G004	1.00	1.00	1.00	18
EMDS7-G005	1.00	0.91	0.95	11
EMDS7-G006	1.00	1.00	1.00	4
EMDS7-G007	1.00	1.00	1.00	3

Category	Precision	Recall	F1-Score	Support
EMDS7-G008	0.71	0.50	0.59	10
EMDS7-G009	0.94	0.94	0.94	16
EMDS7-G010	1.00	1.00	1.00	5
EMDS7-G012	0.90	0.83	0.86	23
EMDS7-G013	1.00	1.00	1.00	2
EMDS7-G014	1.00	1.00	1.00	19
EMDS7-G015	1.00	1.00	1.00	4
EMDS7-G016	1.00	0.80	0.89	5
EMDS7-G017	0.88	1.00	0.94	15
EMDS7-G018	1.00	0.94	0.97	18
EMDS7-G019	1.00	0.92	0.96	12
EMDS7-G020	0.89	1.00	0.94	8
EMDS7-G021	1.00	1.00	1.00	2
EMDS7-G022	0.93	0.93	0.93	56
EMDS7-G023	1.00	0.91	0.95	11
EMDS7-G024	1.00	1.00	1.00	5
EMDS7-G025	0.81	0.81	0.81	16
EMDS7-G026	1.00	1.00	1.00	2
EMDS7-G027	1.00	1.00	1.00	4
EMDS7-G028	0.89	1.00	0.94	16
EMDS7-G029	1.00	1.00	1.00	5
EMDS7-G030	0.91	1.00	0.95	10
EMDS7-G031	1.00	1.00	1.00	7
EMDS7-G032	1.00	1.00	1.00	18
EMDS7-G033	0.86	1.00	0.92	6
EMDS7-G034	1.00	1.00	1.00	2
EMDS7-G035	0.92	0.92	0.92	12
EMDS7-G036	1.00	1.00	1.00	17
EMDS7-G037	1.00	1.00	1.00	3
EMDS7-G038	0.93	1.00	0.96	13
EMDS7-G039	0.89	0.94	0.92	18
EMDS7-G040	1.00	0.60	0.75	5

Category	Precision	Recall	F1-Score	Support
EMDS7-G041	1.00	1.00	1.00	1
accuracy	-	-	0.94	487

Table 4.9: Classification Report Detailing Precision, Recall, F1-Score, and Support of the VGG19 Model (40 classes)

Figure 4.64 displays the accuracy of the model for each epoch on the left graph. Our findings show that accuracy rises quickly in the beginning before stabilizing at 99.7% in training and roughly 93% in validation. Roughly 93.63% accuracy is the maximum that was attained.

The model's loss for each epoch is displayed on the right graph. We see that the loss starts off very low and stabilizes at a very low value in training (about 0.003). In validation, the loss is slightly greater (around 0.60), suggesting that the model learns well but exhibits some variation in performance between training and validation.

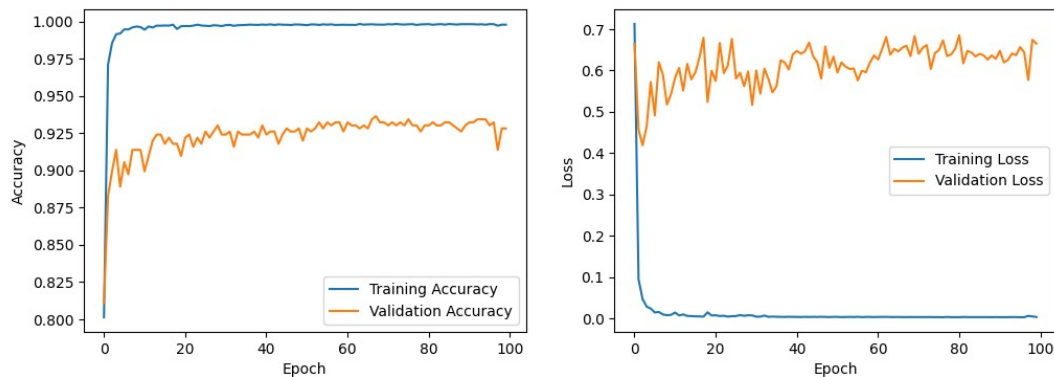


Figure 4.64: Accuracy and Loss Curves During Training and Validation Phases of the VGG19 Model (40 classes).

4.5.2 Pretrained Models & ML Models

This section will present the results of our code implementation shown in details in 4.4.2, we carried out a number of studies into both problematic categories. The comprehensive results are displayed as follows:

4.5.2.1 EfficientNet-B0 + SVM

First, we started by implementing the code using the pre-trained EfficientNet-B0 model to extract features, and then we used the SVM model to classify these features. The results of this process are shown below:

First problematic

The results we obtained after using SVM did not change anything, as the best result was the same as the one before using SVM. Even the confusion matrix (see Figure 4.65) and classification report (see Table 4.10) were exactly the same.

The best hyperparameters are: C=0.1, kernel=linear, degree=2, gamma=scale.

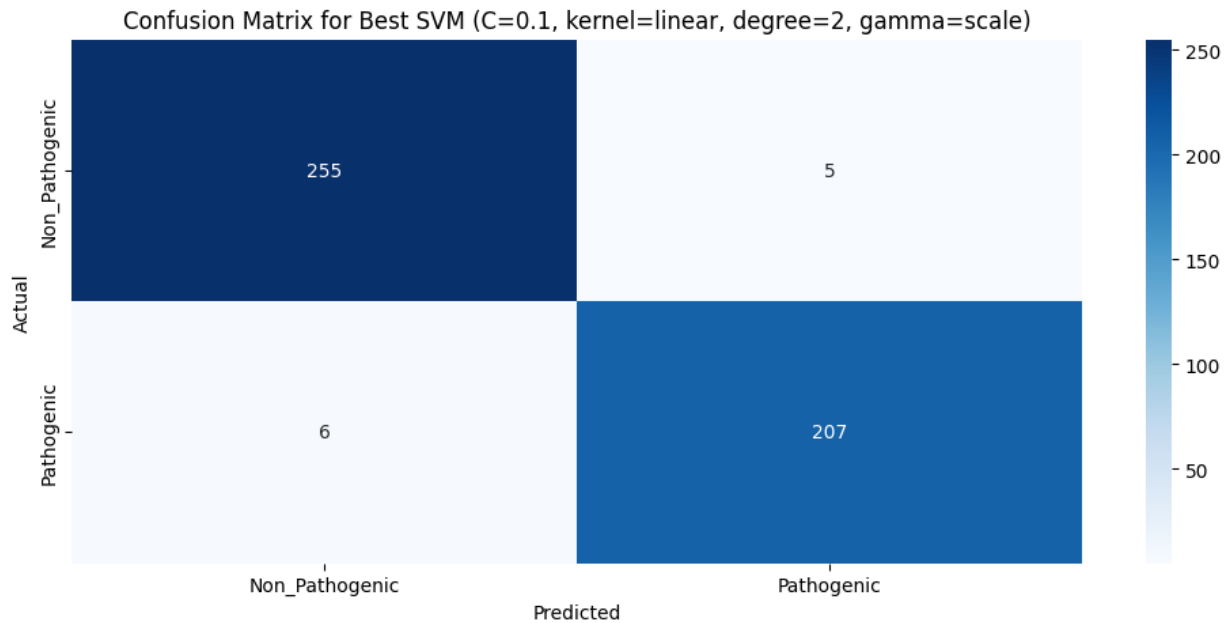


Figure 4.65: Confusion Matrix for EfficientNet-B0 + SVM (2 classes)

	Precision	Recall	F1-Score	Support
Non_Pathogenic	0.98	0.98	0.98	260
Pathogenic	0.98	0.97	0.97	213
Accuracy	0.98 (473)			

Table 4.10: Classification Report for EfficientNet-B0 + SVM (2 classes).

Second problematic

There is a slight improvement in accuracy after using the SVM algorithm. This improvement may be difficult to notice through accuracy alone, but it will be slightly noticeable in the confusion matrix (see Figure 4.66) and the classification report results (see Table 4.11). The settings that yielded the best results were as follows: $C=0.1$, $\text{kernel}=\text{rbf}$, $\text{degree}=2$, $\text{gamma}=\text{scale}$.

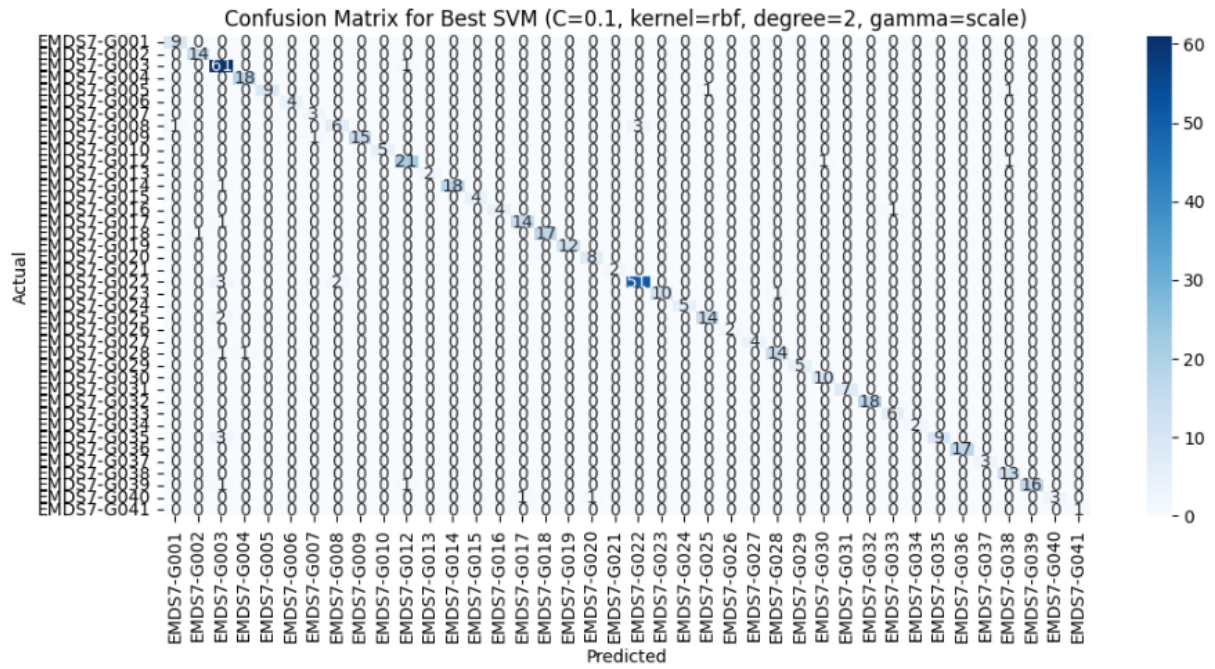


Figure 4.66: Confusion Matrix for EfficientNet-B0 + SVM (40 classes).

Category	Precision	Recall	F1-Score	Support
EMDS7-G001	0.90	1.00	0.95	9
EMDS7-G002	0.93	1.00	0.97	14
EMDS7-G003	0.84	0.98	0.90	62
EMDS7-G004	0.95	1.00	0.97	18
EMDS7-G005	1.00	0.82	0.90	11
EMDS7-G006	1.00	1.00	1.00	4
EMDS7-G007	0.75	1.00	0.86	3
EMDS7-G008	0.75	0.60	0.67	10

Category	Precision	Recall	F1-Score	Support
EMDS7-G009	1.00	0.94	0.97	16
EMDS7-G010	1.00	1.00	1.00	5
EMDS7-G012	0.91	0.91	0.91	23
EMDS7-G013	1.00	1.00	1.00	2
EMDS7-G014	1.00	0.95	0.97	19
EMDS7-G015	1.00	1.00	1.00	4
EMDS7-G016	1.00	0.80	0.89	5
EMDS7-G017	0.93	0.93	0.93	15
EMDS7-G018	1.00	0.94	0.97	18
EMDS7-G019	1.00	1.00	1.00	12
EMDS7-G020	0.89	1.00	0.94	8
EMDS7-G021	1.00	1.00	1.00	2
EMDS7-G022	0.94	0.91	0.93	56
EMDS7-G023	1.00	0.91	0.95	11
EMDS7-G024	1.00	1.00	1.00	5
EMDS7-G025	0.93	0.88	0.90	16
EMDS7-G026	1.00	1.00	1.00	2
EMDS7-G027	1.00	1.00	1.00	4
EMDS7-G028	0.93	0.88	0.90	16
EMDS7-G029	1.00	1.00	1.00	5
EMDS7-G030	0.91	1.00	0.95	10
EMDS7-G031	1.00	1.00	1.00	7
EMDS7-G032	1.00	1.00	1.00	18
EMDS7-G033	0.86	1.00	0.92	6
EMDS7-G034	1.00	1.00	1.00	2
EMDS7-G035	1.00	0.75	0.86	12
EMDS7-G036	1.00	1.00	1.00	17
EMDS7-G037	1.00	1.00	1.00	3
EMDS7-G038	0.87	1.00	0.93	13
EMDS7-G039	1.00	0.89	0.94	18
EMDS7-G040	1.00	0.60	0.75	5
EMDS7-G041	1.00	1.00	1.00	1

Category	Precision	Recall	F1-Score	Support
accuracy	-	-	0.94	487

Table 4.11: Classification Report for EfficientNet-B0 + SVM (40 classes).

4.5.2.2 Resnet50 + SVM

In this section, we present the performance results of our classification model, which combines the features extraction using ResNet50 with an SVM classifier. The following evaluation results illustrate the effectiveness of the ResNet50 + SVM combination.

First problematic

The results of the confusion matrix (Figure 4.67) and classification report (Table 4.12) for the best SVM model, with parameters $C=0.1$, $\text{kernel}=\text{linear}$, $\text{degree}=2$, and $\text{gamma}=\text{scale}$, are also consistent with the performance observed using ResNet50 before the application of SVM.

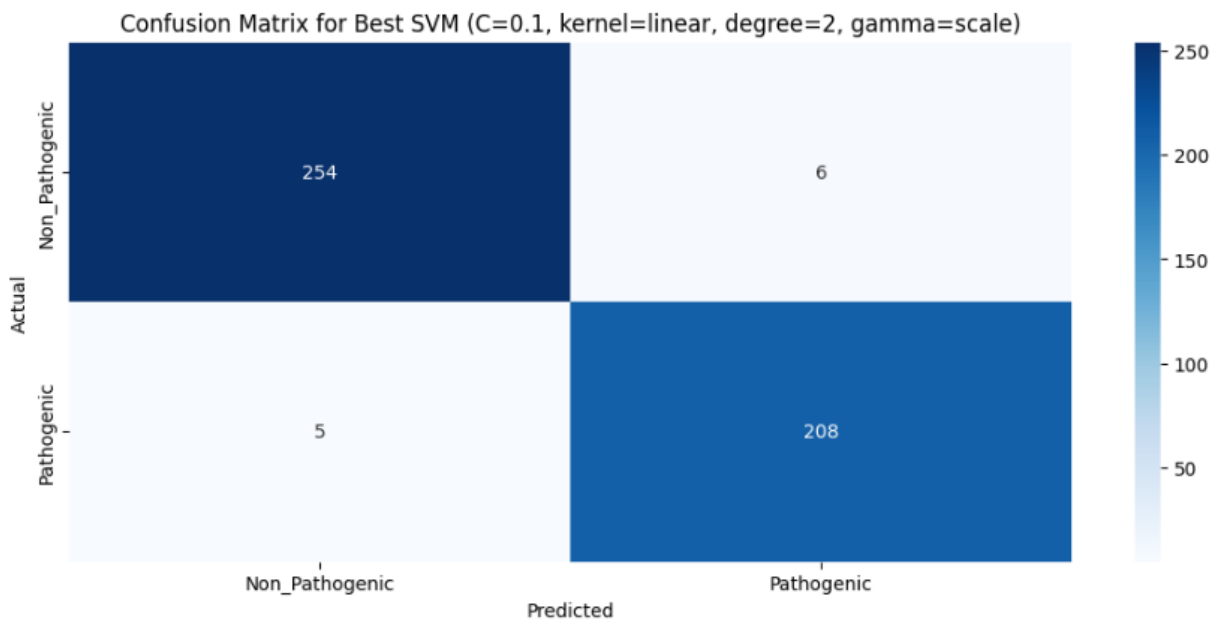


Figure 4.67: Confusion Matrix for ResNet50 + SVM (2 classes).

Class	Precision	Recall	F1-Score	Support
Non_Pathogenic	0.98	0.98	0.98	260
Pathogenic	0.97	0.98	0.97	213
Accuracy	0.98 (473)			

Table 4.12: Classification Report for ResNet50 + SVM (2 classes).

Second problematic

The following evaluation results illustrate the effectiveness of the ResNet50 + SVM combination. These results are consistent with the performance observed using ResNet50 before the application of SVM, indicating that the SVM had no positive or negative impact on these results.

However, we will show the confusion matrix (see Figure 4.68) and Classification Report (see Table 4.13) again with the Parameters: C=0.1, kernel=linear, degree=2, gamma=scale that gave the results that are considered the best.

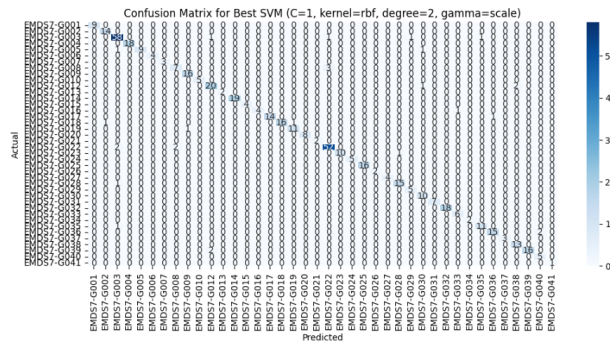


Figure 4.68: Confusion Matrix for ResNet50 + SVM (40 classes).

Category	Precision	Recall	F1-Score	Support
EMDS7-G001	1.00	1.00	1.00	9
EMDS7-G002	0.93	1.00	0.97	14
EMDS7-G003	0.92	0.94	0.93	62
EMDS7-G004	1.00	1.00	1.00	18
EMDS7-G005	1.00	0.82	0.90	11
EMDS7-G006	1.00	1.00	1.00	4

Category	Precision	Recall	F1-Score	Support
EMDS7-G007	1.00	1.00	1.00	3
EMDS7-G008	0.78	0.70	0.74	10
EMDS7-G009	0.94	1.00	0.97	16
EMDS7-G010	1.00	1.00	1.00	5
EMDS7-G012	0.87	0.87	0.87	23
EMDS7-G013	1.00	1.00	1.00	2
EMDS7-G014	1.00	1.00	1.00	19
EMDS7-G015	1.00	1.00	1.00	4
EMDS7-G016	1.00	0.80	0.89	5
EMDS7-G017	1.00	0.93	0.97	15
EMDS7-G018	1.00	0.89	0.94	18
EMDS7-G019	0.92	0.92	0.92	12
EMDS7-G020	1.00	1.00	1.00	8
EMDS7-G021	1.00	1.00	1.00	2
EMDS7-G022	0.93	0.93	0.93	56
EMDS7-G023	1.00	0.91	0.95	11
EMDS7-G024	1.00	1.00	1.00	5
EMDS7-G025	1.00	1.00	1.00	16
EMDS7-G026	1.00	1.00	1.00	2
EMDS7-G027	1.00	1.00	1.00	4
EMDS7-G028	0.94	0.94	0.94	16
EMDS7-G029	0.83	1.00	0.91	5
EMDS7-G030	0.83	1.00	0.91	10
EMDS7-G031	1.00	1.00	1.00	7
EMDS7-G032	1.00	1.00	1.00	18
EMDS7-G033	0.86	1.00	0.92	6
EMDS7-G034	1.00	1.00	1.00	2
EMDS7-G035	0.92	0.92	0.92	12
EMDS7-G036	0.94	0.88	0.91	17
EMDS7-G037	1.00	1.00	1.00	3
EMDS7-G038	0.87	1.00	0.93	13
EMDS7-G039	1.00	0.89	0.94	18

Category	Precision	Recall	F1-Score	Support
EMDS7-G040	0.71	1.00	0.83	5
EMDS7-G041	1.00	1.00	1.00	1
Accuracy	0.94 (487)			

Table 4.13: Classification Report for ResNet50 + SVM (40 classes).

4.5.2.3 VGG19 + SVM

In the end, we applied the same procedure to the resulted model VGG19 and obtained the following results:

First problematic

There is a slight decline in the results after using SVM with this parameters: $C=1$, $\text{kernel}=\text{rbf}$, $\text{degree}=2$, $\text{gamma}=\text{scale}$, but this decline does not make a noticeable difference in the confusion matrix (see Figure 4.69) compared to the results of VGG19 before using SVM. However, the classification report (see Table 4.14) remains unchanged.

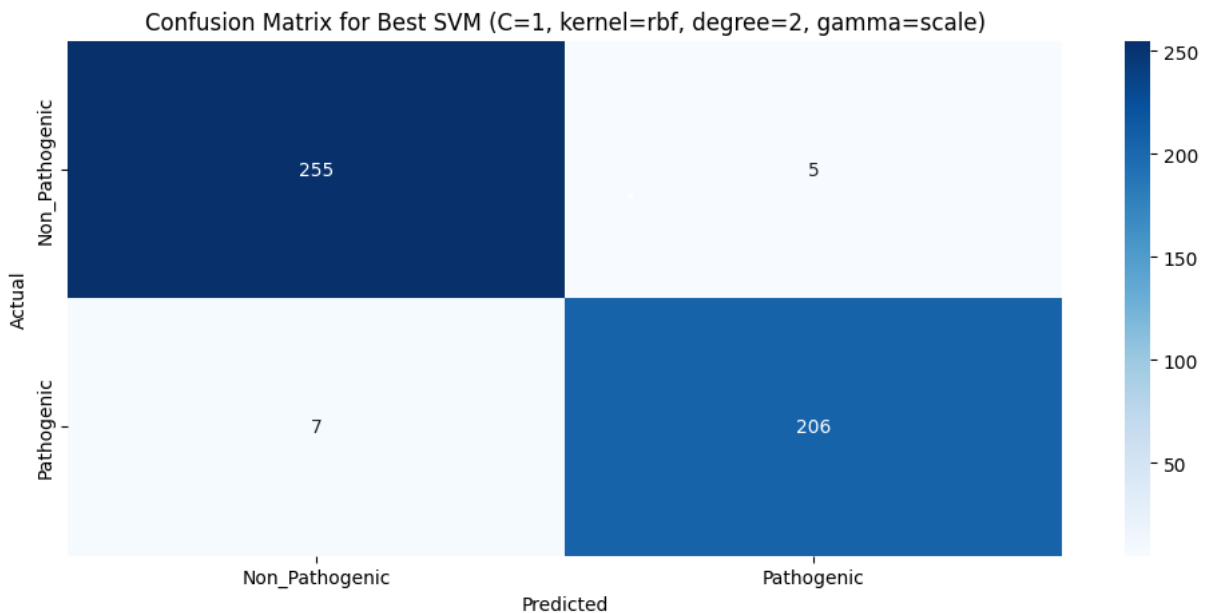


Figure 4.69: Confusion Matrix for VGG19 + SVM (2 classes).

	Precision	Recall	F1-Score	Support
Non_Pathogenic	0.97	0.98	0.98	260
Pathogenic	0.98	0.97	0.97	213
Accuracy	0.97 (473)			

Table 4.14: Classification Report for VGG19 + SVM (2 classes).

Second problematic

When compared to applying the model without SVM, the results in this instance are inferior because to the degradation in performance caused by the usage of SVM. The classification report (see Table 4.15) and the confusion matrix (see Figure 4.70) are shown next. The accuracy decreased to 88% from 94%, and this is the best model that could be found. The settings are degree=2, gamma=scale, kernel=rbf, and C=1.

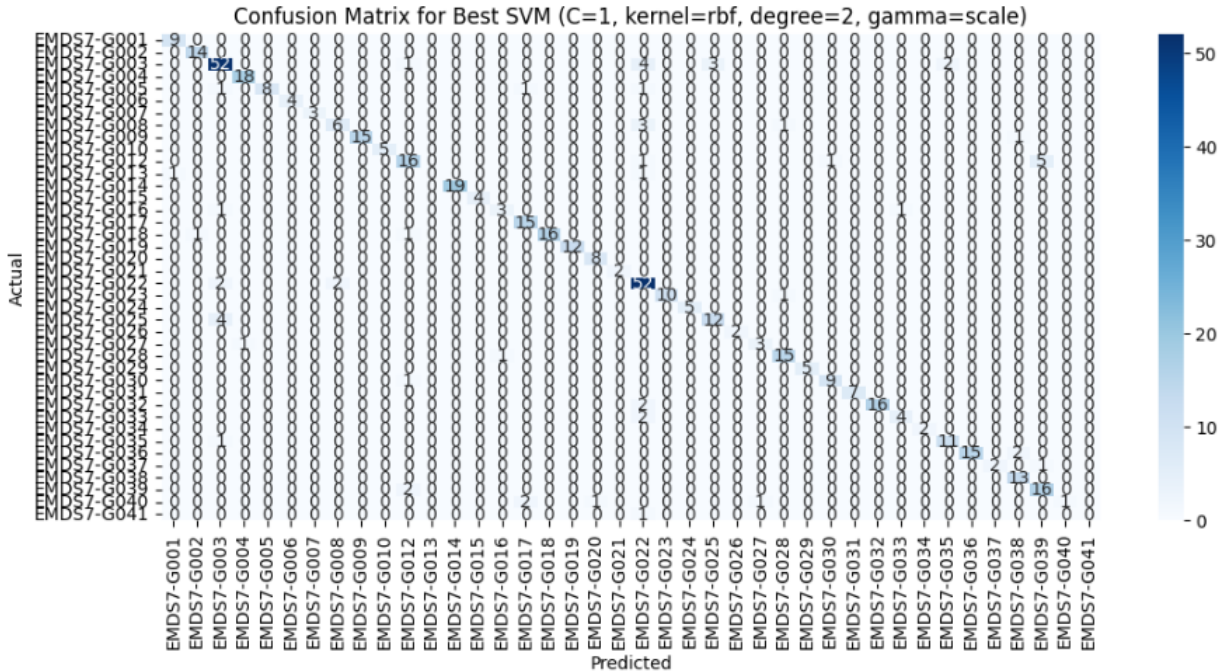


Figure 4.70: Confusion Matrix for VGG19 + SVM (40 classes).

Category	Precision	Recall	F1-Score	Support
EMDS7-G001	0.90	1.00	0.95	9
EMDS7-G002	0.93	1.00	0.97	14
EMDS7-G003	0.85	0.84	0.85	62
EMDS7-G004	0.95	1.00	0.97	18
EMDS7-G005	1.00	0.73	0.84	11
EMDS7-G006	1.00	1.00	1.00	4
EMDS7-G007	1.00	1.00	1.00	3
EMDS7-G008	0.75	0.60	0.67	10
EMDS7-G009	1.00	0.94	0.97	16
EMDS7-G010	1.00	1.00	1.00	5
EMDS7-G012	0.76	0.70	0.73	23
EMDS7-G013	0.00	0.00	0.00	2
EMDS7-G014	1.00	1.00	1.00	19
EMDS7-G015	1.00	1.00	1.00	4
EMDS7-G016	0.75	0.60	0.67	5
EMDS7-G017	0.83	1.00	0.91	15
EMDS7-G018	1.00	0.89	0.94	18
EMDS7-G019	1.00	1.00	1.00	12
EMDS7-G020	0.89	1.00	0.94	8
EMDS7-G021	1.00	1.00	1.00	2
EMDS7-G022	0.78	0.93	0.85	56
EMDS7-G023	1.00	0.91	0.95	11
EMDS7-G024	1.00	1.00	1.00	5
EMDS7-G025	0.80	0.75	0.77	16
EMDS7-G026	1.00	1.00	1.00	2
EMDS7-G027	0.75	0.75	0.75	4
EMDS7-G028	0.88	0.94	0.91	16
EMDS7-G029	1.00	1.00	1.00	5
EMDS7-G030	0.90	0.90	0.90	10
EMDS7-G031	1.00	1.00	1.00	7
EMDS7-G032	1.00	0.89	0.94	18
EMDS7-G033	0.80	0.67	0.73	6

Category	Precision	Recall	F1-Score	Support
EMDS7-G034	1.00	1.00	1.00	2
EMDS7-G035	0.85	0.92	0.88	12
EMDS7-G036	1.00	0.88	0.94	17
EMDS7-G037	1.00	0.67	0.80	3
EMDS7-G038	0.81	1.00	0.90	13
EMDS7-G039	0.73	0.89	0.80	18
EMDS7-G040	1.00	0.20	0.33	5
EMDS7-G041	0.00	0.00	0.00	1
Accuracy	0.88 (487)			

Table 4.15: Classification Report for VGG19 + SVM (40 classes).

4.5.3 ViT Model

Before presenting the results, it is important to note that the results we obtained were not what we expected. we anticipated better outcomes from using the ViT (Vision Transformer) model; however, unfortunately, the current results did not meet the expected goals. Below, we will present and analyze these results in detail.

First problematic

The confusion matrix in Figure 4.71 provides a detailed analysis of the model's performance in classifying images into two classes. It shows the number of instances for each class that were correctly and incorrectly classified.

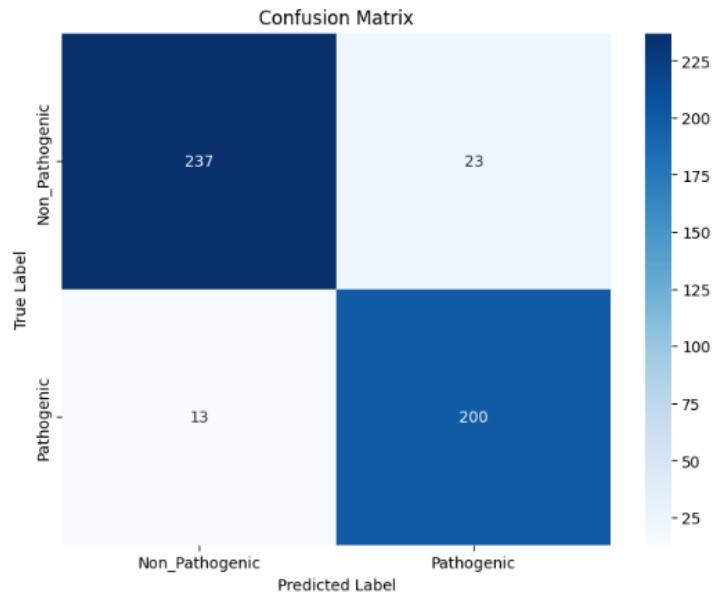


Figure 4.71: Confusion matrix showing the performance of the VIT model (2 classes).

Table 4.16 displays the precision, recall, f1-score, and support for each class.

	Precision	Recall	F1-Score	Support
Non_Pathogenic	0.95	0.91	0.93	260
Pathogenic	0.90	0.94	0.92	213
accuracy	-	-	0.92	473

Table 4.16: Classification Report Detailing Precision, Recall, F1-Score, and Support of the VIT model (2 classes)

Training and Validation Metrics Figure 4.72 displays the accuracy and loss of the model for each epoch during training and validation phases.

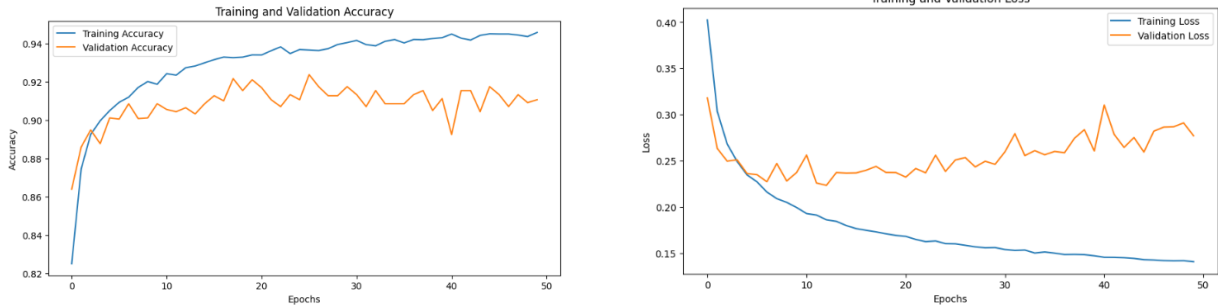


Figure 4.72: Accuracy and Loss Curves During Training and Validation Phases of the VIT model (2 classes).

The left graph in Figure 4.72 shows that accuracy increases rapidly in the beginning before stabilizing at nearly 94% in training and roughly 90-92% in validation. The right graph shows that the loss starts off high and stabilizes at a very low value in training, while the validation loss is slightly greater, indicating good learning with some variation in performance between training and validation.

Second problematic

The confusion matrix shown in Figure 4.73 provides a thorough study of the model's performance in classifying images into multiple classes by showing the number of incidents for each class that were properly and incorrectly classified.

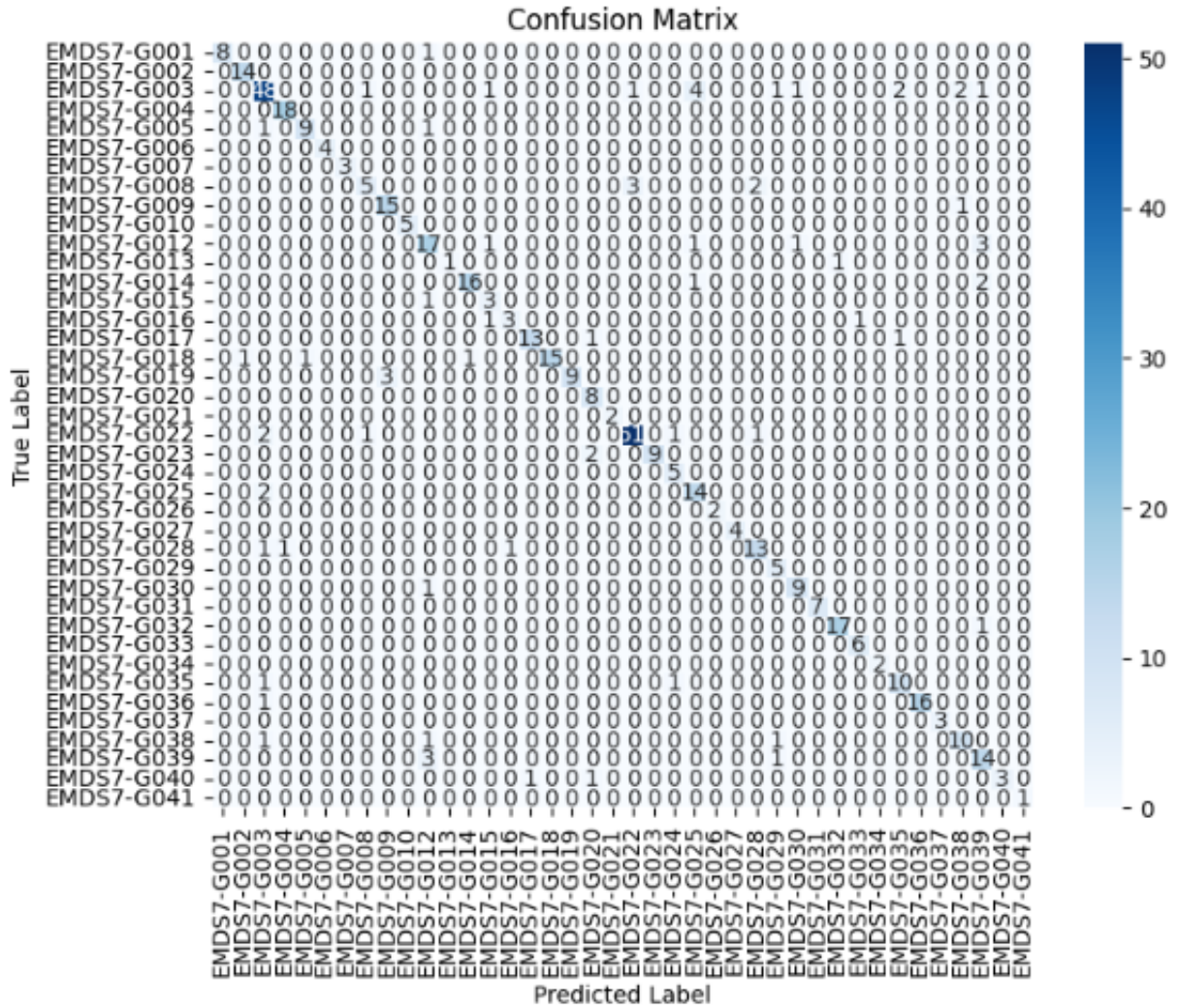


Figure 4.73: Confusion matrix showing the performance of the VIT model (40 classes).

More information on understanding the confusion matrix can be found in the table below (Table 4.17), which displays the precision, recall, f1-score, and support for each class.

Category	Precision	Recall	F1-Score	Support
EMDS7-G001	1.00	0.89	0.94	9
EMDS7-G002	0.93	1.00	0.97	14
EMDS7-G003	0.84	0.77	0.81	62
EMDS7-G004	0.95	1.00	0.97	18
EMDS7-G005	0.90	0.82	0.86	11

Category	Precision	Recall	F1-Score	Support
EMDS7-G006	1.00	1.00	1.00	4
EMDS7-G007	1.00	1.00	1.00	3
EMDS7-G008	0.71	0.50	0.59	10
EMDS7-G009	0.83	0.94	0.88	16
EMDS7-G010	1.00	1.00	1.00	5
EMDS7-G012	0.68	0.74	0.71	23
EMDS7-G013	1.00	0.50	0.67	2
EMDS7-G014	0.94	0.84	0.89	19
EMDS7-G015	0.50	0.75	0.60	4
EMDS7-G016	0.75	0.60	0.67	5
EMDS7-G017	0.93	0.87	0.90	15
EMDS7-G018	1.00	0.83	0.91	18
EMDS7-G019	1.00	0.75	0.86	12
EMDS7-G020	0.67	1.00	0.80	8
EMDS7-G021	1.00	1.00	1.00	2
EMDS7-G022	0.93	0.91	0.92	56
EMDS7-G023	1.00	0.82	0.90	11
EMDS7-G024	0.71	1.00	0.83	5
EMDS7-G025	0.70	0.88	0.78	16
EMDS7-G026	1.00	1.00	1.00	2
EMDS7-G027	1.00	1.00	1.00	4
EMDS7-G028	0.81	0.81	0.81	16
EMDS7-G029	0.62	1.00	0.77	5
EMDS7-G030	0.82	0.90	0.86	10
EMDS7-G031	1.00	1.00	1.00	7
EMDS7-G032	0.94	0.94	0.94	18
EMDS7-G033	0.86	1.00	0.92	6
EMDS7-G034	1.00	1.00	1.00	2
EMDS7-G035	0.77	0.83	0.80	12
EMDS7-G036	1.00	0.94	0.97	17
EMDS7-G037	1.00	1.00	1.00	3
EMDS7-G038	0.77	0.77	0.77	13

Category	Precision	Recall	F1-Score	Support
EMDS7-G039	0.67	0.78	0.72	18
EMDS7-G040	1.00	0.60	0.75	5
EMDS7-G041	1.00	1.00	1.00	1
accuracy	-	-	0.86	487

Table 4.17: Classification Report Detailing Precision, Recall, F1-Score, and Support of the VIT model (40 classes)

Figure 4.74 displays the accuracy of the model for each epoch on the left graph. Our findings show that accuracy rises quickly in the beginning before stabilizing at nearly 100% in training and roughly 80-85% in validation.

The model’s loss for each epoch is displayed on the right graph. We see that the loss starts off very low and stabilizes at a very low value in training. In validation, the loss is slightly greater, suggesting that the model learns well but exhibits some variation in performance between training and validation.

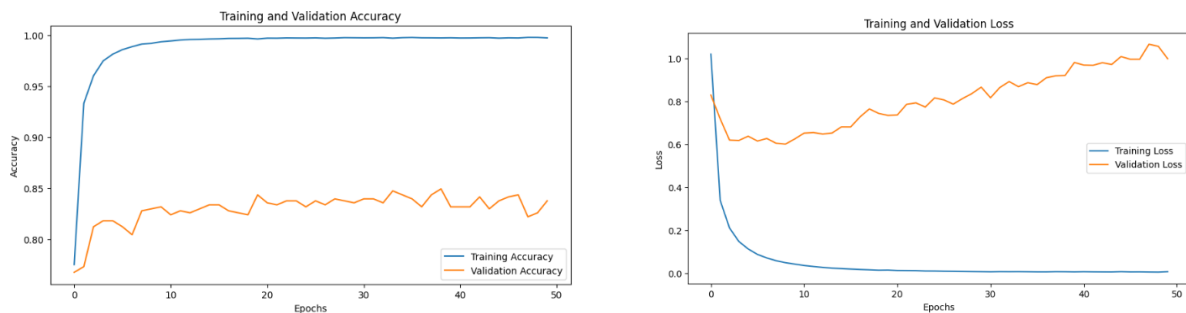


Figure 4.74: Accuracy and Loss Curves During Training and Validation Phases of the VIT model (40 classes).

4.5.4 discussion

The table 4.18 presents a comparison of various models on two different problematics based on three performance measures: precision, recall, and accuracy. Below is the discussion for each problematic:

Measure	Problematic	Problematic 1			Problematic 2		
		Precision	Recall	Accuracy	Precision	Recall	Accuracy
Pretrained model	Efficientnetb0	0.98	0.98	0.98	0.96	0.95	0.94
	Resnet50	0.98	0.98	0.98	0.96	0.96	0.95
	VGG19	0.98	0.98	0.98	0.96	0.95	0.94
Pretrained + svm	Efficientnetb0	0.98	0.98	0.98	0.96	0.94	0.94
	Resnet50	0.98	0.98	0.98	0.95	0.96	0.94
	VGG19	0.97	0.97	0.97	0.87	0.84	0.88
VIT	VITB16	0.88	0.86	0.85	0.92	0.93	0.92

Table 4.18: Comparison of models on two problematics

Problematic 1: The best is the pre-trained models alone (Efficientnetb0, Resnet50, and VGG19) also the Efficientnetb0, and Resnet50 with SVM as they achieved equal and high performance.

Problematic 2: The best is the pre-trained Resnet50 alone, which achieved high, consistent performance with minimal variances compared to other models.

4.5.5 Model deployment

Multiple tools and strategies are available for implementing an automatic learning model. Our choice of the open-source Python package Gradio framework is based on its ability to quickly create user-customizable and intuitive interface components for our model. Additionally, Gradio provides the ability to create shareable links with a 72-hour validity period.

The code sets up a web interface using Gradio to classify images with a pre-trained EfficientNet model. It imports necessary libraries, defines image transformation steps (resize and normalize), and loads the EfficientNet model modified for binary classification ("Pathogenic" and "Non_Pathogenic"). The model weights are loaded, and the model is set to evaluation mode. The 'classify_image' function preprocesses input images, performs inference with the model, and returns class probabilities. A Gradio interface is created to upload images, classify them, and display the results. Finally, 'demo.launch(share=True)' starts the interface for public use. And here is the code:

```
1 import gradio as gr
2 import torch
3 import torchvision.transforms as transforms
4 from PIL import Image
5 from efficientnet_pytorch import EfficientNet
6 import torch.nn as nn
7
8 # Define the transformation: resize to 224x224 and normalize
9 image_size = (224, 224)
10 data_transforms = transforms.Compose([
11     transforms.Resize(image_size),
12     transforms.ToTensor(),
13     transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
14 ])
15
16 # Load the trained model
17 device = torch.device("cpu")
18 model = EfficientNet.from_name('efficientnet-b0')
19 num_ftrs = model._fc.in_features
20 model._fc = nn.Linear(num_ftrs, 2)
21 model.load_state_dict(torch.load('C:/Users/kaouthar/Desktop/FINAL/2CLASS/efficientnetb0/efficientnet-b0_0.9767.pth',
22     map_location=device))
23 model = model.to(device)
24 model.eval()
25
26 # Image preprocessing and classification function
27 def classify_image(image):
28     image = data_transforms(image).unsqueeze(0).to(device)
29     outputs = model(image)
30     percentages = torch.nn.functional.softmax(outputs, dim=1)[0] # Calculate percentages
31     class_names = ["Pathogenic", "Non_Pathogenic"]
32     result = {class_names[i]: round(percentages[i].item(), 2) for i in range(len(class_names))}
33
34     return result
35
36 # Gradio interface
37 demo = gr.Interface(
38     fn=classify_image,
39     inputs=gr.Image(type="pil"),
40     outputs=gr.Label(num_top_classes=2),
41     title="Image Classifier",
42     description="Upload an image to classify it."
43 )
44 demo.launch(share=True)
45
```

Figure 4.75: Image Classification using EfficientNet and Gradio Interface

After running the code, you will get two URLs: Local URL, and public URL.

Additionally, you will get a message stating that the public share link expires in 72 hours. If you want free permanent hosting and GPU upgrades, you can run the command `gradio deploy` from the Terminal to deploy your project to Hugging Face Spaces.

When the user clicks on the public URL, the app's web interface opens, allowing him to select the classification type (see Figure 4.76) he needs and upload an image to get results (see Figure 4.77 & 4.78).

Choose a model to classify images

EMs detection Pathogenic & Non_Pathogenic detection

Figure 4.76: select the classification type

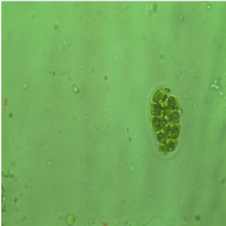
Select the classification type

EMs detection Pathogenic & Non_Pathogenic detection

Aqua Eye

Upload an image to classify it.

image



output

Non_Pathogenic

Non_Pathogenic	100%
Pathogenic	0%

Flag

Clear Submit

Figure 4.77: Application example First problematic.

Select the classification type

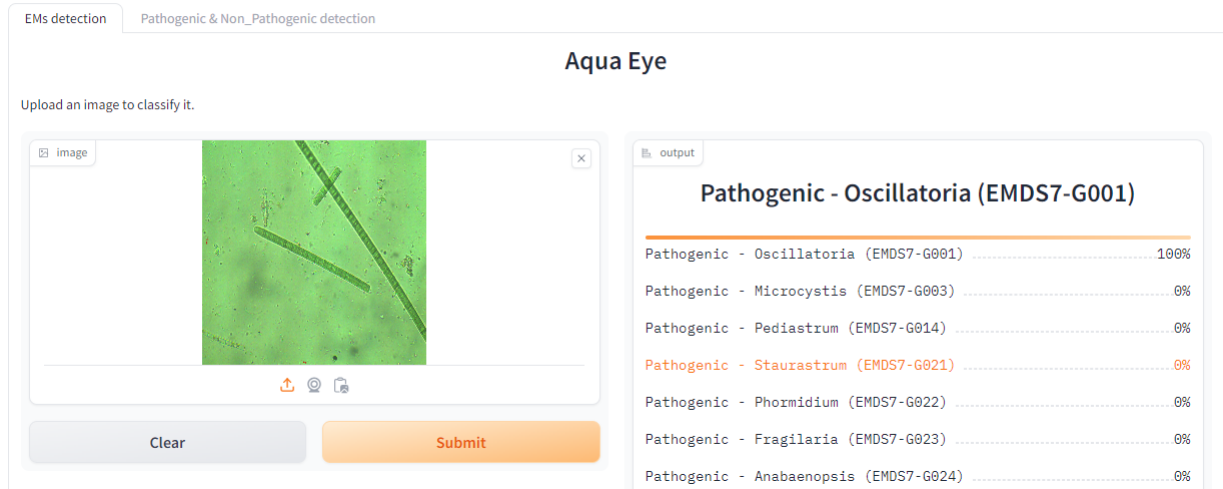


Figure 4.78: Application example Second problematic.

4.6 Conclusion

In this chapter, we presented a detailed system for using electron microscope images to detect and classify microorganisms in water. We covered the methodologies for data collection, processing, model training, and performance evaluation, demonstrating how these elements contribute to developing an effective real-time water quality assessment tool. The results show that pre-trained models such as EfficientNetB0 and ResNet50 achieve high performance in classifying microorganisms, offering improved accuracy and speed compared to traditional methods.

The deployment process, implemented using the Gradio framework, demonstrates how end-users can easily access the model through a dedicated web interface. This technological advancement represents a significant step towards improving public health outcomes by providing rapid and accurate detection of harmful microorganisms in water sources.

General conclusion

Detecting and classifying microorganisms in water is crucial to ensure water quality and human health. This study aims to develop and test effective models using deep learning techniques to identify and classify microorganisms in water.

Both studies focused on using a variety of deep learning models such as ResNet50, VGG19, EfficientNet-B0, and Vision Transformer (ViT), in addition to enhancing these models using SVM classifiers. The results showed that the pretrained models (ResNet50, VGG19, and EfficientNet-B0) performed excellently in the first classification problem (classifying microorganisms as pathogenic or non-pathogenic), achieving an accuracy of 98%. In the second classification problem (classifying microorganisms into 40 different categories), the pretrained ResNet50 showed the best performance, achieving the highest accuracy and stability. The pretrained EfficientNet-B0 model also demonstrated good performance in both problems, making it a favorable choice due to its efficiency and smaller size.

Regarding the Vision Transformer (ViT) model, it showed good performance but was less stable and accurate compared to traditional models like ResNet50 and EfficientNet-B0, limiting its effectiveness in some classification scenarios.

Although the results were encouraging, there are many challenges that need to be addressed in the future. Expanding the dataset to ensure that the models can recognize a wider range of microorganisms is essential. Improving preprocessing techniques can also enhance model performance. Additionally, using advanced detection techniques such as YOLO can improve the accuracy of pinpointing microorganism locations in images. This study represents a significant step towards improving tools for detecting microorganisms in water using deep learning techniques, paving the way for further research to enhance the accuracy and efficiency of these models in the future.

In the future, we will not rely on microscopes; instead, we will capture images directly from mobile phones using microscopic slides.

Bibliography

- [1] Milan K. Jermar. *Water Resources and Water Management*. Elsevier Science Pub. Co, Distributors for the United States and Canada, 1987.
- [2] T. Oki. Global hydrological cycles and world water resources. *Science*, 313(5790):1068–1072, 2006.
- [3] W. J. Shuttleworth. Evaporation. *NORA (NERC Acoustics)*, 1979.
- [4] G. K. Vallis. The trouble with water: Condensation, circulation and climate. *The European Physical Journal Plus*, 135(6), 2020.
- [5] C. De Jong. The contribution of condensation to the water cycle under high-mountain conditions. *Hydrological Processes*, 19(12):2419–2435, 2005.
- [6] Thomas Pagano and Soroosh Sorooshian. Hydrologic cycle. *Encyclopedia of Global Environmental Change*, 1:450–464, 2002. PDF.
- [7] Pierre Bourgooin. A method to determine precipitation types. *Weather and Forecasting*, pages 583–592, 2000.
- [8] Ahmad Wedyan, Jacqueline Whalley, and Ajit Narayanan. Hydrological cycle algorithm for continuous optimization problems. *Journal of Optimization*, 2017:Article ID 3828420, 2017.
- [9] NotesYCHS. The hydrological cycle. <https://notesychs.weebly.com/the-hydrological-cycle.html>, Accessed May 12, 2024.
- [10] A. A. Murad, H. Al Nuaimi, and M. Al Hammadi. Comprehensive assessment of water resources in the united arab emirates (uae). *Water Resources Management*, 21(9):1449–1463, 2006.

- [11] Jean Margat and Thierry Ruf. *Are Groundwater Resources Eternal? 90 Keys to Understanding Groundwater*. Editions Quae, 2014.
- [12] S. Mandel and Z. L. Shiftan. *Groundwater Resources: Investigation and Development*. Academic Press, New York London Toronto Sydney San Francisco, 1981.
- [13] National Geographic Education. Surface water. <https://education.nationalgeographic.org/resource/surface-water/>, Accessed April 12, 2024.
- [14] Bjorn Stevens and Sandrine Bony. Water in the atmosphere. *Physics Today*, 66(6):30, 2013. Published Online: May 31, 2013; Published Print: June 1, 2013.
- [15] Z Kılıç. The importance of water and conscious use of water. *International Journal of Hydrology*, 2020.
- [16] Mohammad Zakir Hossain. Water: The most precious resource of our life. *College of Economics and Political Science, Sultan Qaboos University, Oman*, 2:1436–1445, 2015.
- [17] Iván Francisco García-Tejero, Víctor Hugo Durán-Zuazo, José Luis Muriel-Fernández, and Carmen Rocío Rodríguez-Pleguezuelo. *Water and Sustainable Agriculture*. Springer Netherlands, 1st edition, 2011.
- [18] Ülo Mander, Jean Tournebize, Kuno Kasak, and William J Mitsch. Climate regulation by free water surface constructed wetlands for wastewater treatment and created riverine wetlands. *Ecological Engineering*, 72:103–115, 2014.
- [19] Igor A. Shiklomanov. World water resources: A new appraisal and assessment for the 21st century. *State Hydrological Institute, St Petersburg, Russia*, page 18, 2000. A summary of the monograph prepared in the framework of the International Hydrological Programme.
- [20] Molly Sargen. Biological roles of water: Why is water necessary for life? <https://notesychs.weebly.com/the-hydrological-cycle.html>, June 2023. Figures by Daniel Utter.
- [21] Andrew Pohorille and Lawrence R. Pratt. Is water the universal solvent for life? *Origins of Life and Evolution of Biospheres*, 2012.
- [22] L. Torres-Ronda and X. Schelling i del Alcázar. The properties of water and their applications for training. *Journal of Human Kinetics*, 44(1):237–248, 2014.

- [23] F. Oztas and E. Bozkurt. Biology teacher candidates' misconceptions about surface tension, adhesion and cohesion. *Energy Education Science and Technology Part B*, 2011.
- [24] FN Chaudhry and MF Malik. Factors affecting water pollution: A review. *Journal of Ecosystem & Ecography*, 7:1, 2017.
- [25] SHH Al-Taai. Water pollution: Its causes and effects. *IOP Conference Series: Earth and Environmental ...*, 2021, 2021.
- [26] Chao Wang and Chenxu Yu. Detection of chemical pollutants in water using gold nanoparticles as sensors: a review. *Reviews in Analytical Chemistry*, 32(1):1–14, 2013.
- [27] Emin Toroglu and Sevil Toroglu. Microbial pollution of water in golbasi lake in adiyaman, turkey. *J. Environ. Biol.*, 30(1):33–38, 2009.
- [28] Hessamaddin Sohrabi, Afsaneh Hemmati, Mir Reza Majidi, Shirin Eyvazi, Ali Jahanban-Esfahlan, Behzad Baradaran, Roshanak Adlpour-Azar, Ahad Mokhtarzadeh, and Miguel de la Guardia. Recent advances on portable sensing and biosensing assays applied for detection of main chemical and biological pollutant agents in water samples: A critical review. *TrAC Trends in Analytical Chemistry*, 143:116344, 2021.
- [29] João P. S. Cabral. Water microbiology. bacterial pathogens and water. *Int. J. Environ. Res. Public Health*, 7(10):3657–3703, 2010.
- [30] Mani Maheshwari and Bindu Kiranmayi. Vibrio cholerae - a review. *Veterinary World*, September 2011.
- [31] M. E. Ohl and S. I. Miller. Salmonella: A model for bacterial pathogenesis. *Annual Review of Medicine*, 52(1):259–274, February 2001.
- [32] Shu-Kee Eng, Priyia Pusparajah, Nurul-Syakima Ab Mutalib, Hooi-Leng Ser, Kok-Gan Chan, and Learn-Han Lee. Salmonella: A review on pathogenesis, epidemiology and antibiotic resistance. *Frontiers in Life Science*, 8(3):284–293, 2015.
- [33] A. F. Maheux, L. Bissonnette, M. Boissinot, J.-L. T. Bernier, V. Huppé, F. J. Picard, and M. G. Bergeron. Rapid concentration and molecular enrichment approach for sensitive detection of escherichia coli and shigella species in potable water samples. *Applied and Environmental Microbiology*, 77(17):6199–6207, 2011.

- [34] Theng-Theng Fong and Erin K. Lipp. Enteric viruses of humans and animals in aquatic environments: Health risks, detection, and potential water quality assessment tools. *Microbiology and Molecular Biology Reviews*, 69(2):357–371, 2005.
- [35] S. C. Jiang. Human adenoviruses in water: Occurrence and health implications: A critical review. *Environmental Science & Technology*, 40(23):7132–7140, 2006.
- [36] G. R. Takuissua, S. Kenmoeb, J. T. Ebogo-Beloboc, C. Kengne-Ndéd, D. S. Mbagae, A. Bowo-Ngandjie, J. L. Ondigui Ndzee, R. Kenfack-Momof, S. Tchatchouangg, J. Kenfack-Zanguimf, R. Lontuo Fogangh, E. Zeuko’o Menkemi, G. I. Kame-Ngasse, J. N. Magoudjou-Pekamf, E. Suffredinij, C. Venerik, P. Mancinik, G. Bonanno Ferrarok, M. Iaconellik, M. Veranil, I. Federigil, A. Carduccil, and G. La Rosaka. Exploring adenovirus in water environments: a systematic review and meta-analysis. *International Journal of Environmental Health Research*, 34(6):2504–2516, 2024.
- [37] J. M. E. Venter, J. van Heerden, J. C. Vivier, W. O. K. Grabow, and M. B. Taylor. Hepatitis a virus in surface water in south africa: what are the risks? *Journal of Water and Health*, 5(2):229, 2007.
- [38] Yongheng Yang and Mansel W. Griffiths. Comparative persistence of subgroups of f-specific rna phages in river water. *Applied and Environmental Microbiology*, 79(15):4564–4567, 2013.
- [39] R.A. Kristanti, T. Hadibarata, M. Syafrudin, et al. Microbiological contaminants in drinking water: Current status and challenges. *Water, Air, & Soil Pollution*, 233(299), 2022.
- [40] Walter Q. Betancourt and Joan B. Rose. Drinking water treatment processes for removal of cryptosporidium and giardia. *Veterinary Parasitology*, 126(1-2):219–234, 2004.
- [41] M. W. LeChevallier, W. D. Norton, and R. G. Lee. Giardia and cryptosporidium spp. in filtered drinking water supplies. *Applied and Environmental Microbiology*, 57(9):2617–2621, 1991.
- [42] Ana Luz Galván, Angela Magnet, Fernando Izquierdo, Soledad Fenoy, Cristina Rueda, Carmen Fernández Vadillo, Nuno Henriques-Gil, and Carmen del Aguila. Molecular characterization of human-pathogenic microsporidia and cyclospora cayetanensis isolated from various water sources in spain: a year-long longitudinal study. *Applied and Environmental Microbiology*, 79(2):449–459, 2013.

- [43] Risky Ayu Kristanti, Tony Hadibarata, Muhammad Syafrudin, Murat Yilmaz, and Shakila Abdullah. Microbiological contaminants in drinking water: Current status and challenges. *Water Air Soil Pollution*, 233, 2022.
- [44] M. E. Scott. *Ascaris lumbricoides*: A review of its epidemiology and relationship to other infections. *Annales Nestlé (English Ed.)*, 66(1):7–22, 2008.
- [45] A. A. El-Badry, D. A. Hamdy, and W. M. Abd El Wahab. Strongyloides stercoralis larvae found for the first time in tap water using a novel culture method. *Parasitology Research*, 2018.
- [46] L. S. Stephenson, C. V. Holland, and E. S. Cooper. The public health significance of trichuris trichiura. *Parasitology*, 121(S1):S73, 2000.
- [47] R. G. Sinclair, E. L. Jones, and C. P. Gerba. Viruses in recreational water-borne disease outbreaks: a review. *Journal of Applied Microbiology*, 107(6):1769–1780, 2009.
- [48] M. C. Eisenberg, S. L. Robertson, and J. H. Tien. Identifiability and estimation of multiple transmission pathways in cholera and waterborne disease. *Journal of Theoretical Biology*, 324:84–102, 2013.
- [49] B. Beach, J. Ferrie, M. Saavedra, and W. Troesken. Typhoid fever, water quality, and human capital formation. *The Journal of Economic History*, 76, 2016.
- [50] C. N. Thompson, P. Thanh Duy, and S. Baker. The rising dominance of shigella sonnei: An intercontinental shift in the etiology of bacillary dysentery. *PLOS Neglected Tropical Diseases*, 9, 2015.
- [51] A. M. Nasser. Prevalence and fate of hepatitis a virus in water. *Critical Reviews in Environmental Science and Technology*, 24(4):281–323, 1994.
- [52] Kenneth F. Maxcy. Hypothetical relationship of water supplies to poliomyelitis. *American Journal of Public Health*, 33(1):42–45, 1943.
- [53] Mahendra Pal. Amoebiasis: an important foodborne disease of global public health concern. *Opinion Article*, 2, 2020. Received: January 24, 2020; Published: February 14, 2020.
- [54] Joan B. Rose, Charles N. Haas, and Stig Regli. Risk assessment and control of water-borne giardiasis. *American Journal of Public Health*, 81(6), 1991.

- [55] Angesom Hadush and Mahendra Pal. Ascariasis: Public health importance and its status in ethiopia. *Air & Water Borne Diseases*, 5(1), 2016.
- [56] Priya Rani, Shallu Kotwal, Jatinder Manhas, and Vinod Sharma. Machine learning and deep learning based computational approaches in automatic microorganisms image recognition: Methodologies, challenges, and developments. *Archives of Computational Methods in Engineering*, 29:1801–1837, 2022.
- [57] Gopinath Rebala, Ajay Ravi, and Sanjay Churiwala. Machine learning definition and basics. In *An Introduction to Machine Learning*, pages 1–17. SpringerLink, 2019.
- [58] Issam El Naqa and Martin J. Murphy. Machine learning in radiation oncology. In *Machine Learning in Radiation Oncology*, pages 3–11. SpringerLink, 2024.
- [59] Y C A Padmanabha Reddy, P Viswanath, and B Eswara Reddy. Semi-supervised learning: a brief review. *International Journal of Engineering & Technology*, 7(1.8):81–85, 2018. Website: www.sciencepubco.com/index.php/IJET.
- [60] Aized Amin Soofi and Arshad Awan. Classification techniques in machine learning: Applications and issues. *Journal of Basic & Applied Sciences*, 13:459–465, 2017.
- [61] Dastan Hussen Maulud and Adnan Mohsin Abdulazeez. A review on linear regression comprehensive in machine learning. *Journal of Applied Science and Technology Trends*, 01(02):140–147, 2020.
- [62] Mir Henglin, Gillian Stein, Pavel V. Hushcha, Jasper Snoek, Alexander B. Wiltschko, and Susan Cheng. Machine learning approaches in cardiovascular imaging. *Circulation: Cardiovascular Imaging*, 10(10):e005614, 2017. Originally published on 27 Sep 2017.
- [63] Artúr István Károly, Róbert Fullér, and Péter Galambos. Unsupervised clustering for deep learning: A tutorial survey. *Antal Bejczy Center for Intelligent Robotics*, 1(2), 2018.
- [64] Krzysztof J. Cios, Witold Pedrycz, Roman W. Swiniarski, and Lukasz A. Kurgan. *Data Mining - A Knowledge Discovery Approach*. Springer, 2007.
- [65] Kumar Aswani Ch. Analysis of unsupervised dimensionality reduction techniques. *Computer Science and Information Systems*, 6(2):217–227, 2009.

- [66] Johan Mazel. Unsupervised network anomaly detection. *Networking and Internet Architecture [cs.NI]*, 2011. fNNT: ff. fftel-00667654f.
- [67] Markus Goldstein and Seiichi Uchida. Behavior analysis using unsupervised anomaly detection. *The 10th Joint Workshop on Machine ...*, 2014.
- [68] Xiaojin Zhu and Andrew B. Goldberg. Overview of semi-supervised learning. In *Introduction to Semi-Supervised Learning*, Synthesis Lectures on Artificial Intelligence and Machine Learning, pages 9–19. Springer, 2009.
- [69] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, Massachusetts, 1998. PDF.
- [70] LinkedIn. Image from linkedin. https://media.licdn.com/dms/image/D5612AQHjM8zM31Zq9A/article-inline_image-shrink_1500_2232/0/1680157867002?e=1720051200&v=beta&t=HM-tAyjlUSrmCzpNLRNMYBX4m3WesbXPc-y0PIqlffE, Accessed May 12, 2024.
- [71] Laith Alzubaidi, Jinshuai Bai, Aiman Al-Sabaawi, Jose Santamaría, A. S. Albahri, Bashar Sami Nayyef Al-dabbagh, Mohammed A. Fadhel, Mohamed Manoufali, Jinglan Zhang, Ali H. Al-Timemy, Ye Duan, Amjed Abdullah, Laith Farhan, Yi Lu, Ashish Gupta, Felix Albu, Amin Abbosh, and Yuantong Gu. A survey on deep learning tools dealing with data scarcity: definitions, challenges, solutions, tips, and applications. *Journal of Big Data*, 10(1):46, 2023.
- [72] Md. Anwar Hossain and Md. Shahriar Alam Sajib. Classification of image using convolutional neural network (cnn). *Global Journal of Computer Science and Technology: D Neural & Artificial Intelligence*, 19(2), 2019.
- [73] Keiron O’Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015. 10 pages, 5 figures.
- [74] MK Gurucharan. Basic cnn architecture: Explaining 5 layers of convolutional neural network. <https://www.upgrad.com/blog/basic-cnn-architecture/>, Jul 2022. Blog post, last updated 27th July, 2022.
- [75] Shadman Sakib, Nazib Ahmed, Ahmed Jawad Kabir, and Hridon Ahmed. An overview of convolutional neural network: Its architecture and applications. *Preprints*, February 2019. NOT PEER-REVIEWED.

- [76] M. Coşkun, Ö. Yıldırım, A. Uçar, and Y. Demir. An overview of popular deep learning methods. *European Journal of Technique (EJT)*, 7(2):165–176, 2017.
- [77] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*, 2018. Submitted on 29 Dec 2017 (v1), last revised 22 Feb 2018 (this version, v3).
- [78] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*, 2013. Submitted on 20 Dec 2013 (v1), last revised 24 Apr 2014 (this version, v5).
- [79] Trupti Katte. Recurrent neural network and its various architecture types. *International Journal of Research and Scientific Innovation (IJRSI)*, V:124, March 2018.
- [80] S.P. Porkodi, V. Sarada, and V. et al. Maik. Generic image application using gans (generative adversarial networks): A review. *Evolving Systems*, 14:903–917, 2023.
- [81] A. Dash, J. Ye, and G. Wang. A review of generative adversarial networks (gans) and its applications in a wide variety of disciplines: From medical to remote sensing. *IEEE Access*, 12:18330–18357, 2024.
- [82] Yang Wang. A mathematical introduction to generative adversarial nets (gan). <https://doi.org/10.48550/arXiv.2009.00169>, 2020.
- [83] Yuemin Bian, Junmei Wang, Jaden Jungho Jun, and Xiang-Qun Xie. Deep convolutional generative adversarial network (dcgan) models for screening and design of small molecules targeting cannabinoid receptors. *Molecular Pharmaceutics*, 16(11):4451–4460, 2019.
- [84] L. Gao, D. Chen, Z. Zhao, J. Shao, and H.T. Shen. Lightweight dynamic conditional gan with pyramid attention for text-to-image synthesis. *Pattern Recognition*, 107384, 2020.
- [85] Casey Chu, Andrey Zhmoginov, and Mark Sandler. Cyclegan, a master of steganography. *arXiv preprint arXiv:1712.02950*, 2017. NIPS 2017, workshop on Machine Deception.
- [86] Behnaz Gheflati and Hassan Rivaz. Vision transformer for classification of breast ultrasound images. *arXiv preprint arXiv:2107.11892*, October 2021.

- [87] Hansa Hettiarachchi. Unveiling vision transformers: Revolutionizing computer vision beyond convolution. *Medium*, August 2023. <https://medium.com/@hansahettiarachchi/unveiling-vision-transformers-revolutionizing-computer-vision-beyond-convolution>
- [88] D. Yao and Y. Shao. A data efficient transformer based on swin transformer. *Vis Comput*, 40:2589–2598, 2024.
- [89] Hee E Kim, Alejandro Cosa-Linan, Nandhini Santhanam, Mahboubeh Jannesari, Mate E Maros, and Thomas Ganslandt. Transfer learning for medical image classification: a literature review. *BMC Medical Imaging*, 22(1):69, 2022.
- [90] M. Hussain, J.J. Bird, and D.R. Faria. A study on cnn transfer learning for image classification. In A. Lotfi, H. Bouchachia, A. Gegov, C. Langensiepen, and M. McGinnity, editors, *Advances in Computational Intelligence Systems*, volume 840 of *Advances in Intelligent Systems and Computing*. Springer, Cham, 2019.
- [91] Umair Muneer Butt, Sukumar Letchmunan, Fadratul Hafinaz Hassan, Sultan Zia, and Anees Baqir. Detecting video surveillance using vgg19 convolutional neural networks. *International Journal of Advanced Computer Science and Applications*, 11(2), 2020.
- [92] Anuradha Khattar and Syed Quadri. Generalization of convolutional network to domain adaptation network for classification of disaster images on twitter. *Multimedia Tools and Applications*, 81, September 2022.
- [93] Dyah Ajeng Pramudhita, Fatima Azzahra, Ikrar Khaera Arfat, Rita Magdalena, and Sofia Saidah. Strawberry plant diseases classification using cnn based on mobilenetv3-large and efficientnet-b0 architecture. *Jurnal Ilmiah Teknik Elektro Komputer dan Informatika*, 9(3):522–534, 2023.
- [94] Aihua Zhou, Yujun Ma, Wanting Ji, Ming Zong, Pei Yang, Min Wu, and Mingzhe Liu. Multi-head attention-based two-stream efficientnet for action recognition. *Multimedia Systems*, 29:1–12, 06 2022.
- [95] Ishrat Zahan Mukti and Dipayan Biswas. Transfer learning based plant diseases detection using resnet50. In *2019 4th International Conference on Electrical Information and Communication Technology (EICT)*, pages 1–6, 2019.

- [96] Yu-Keun Han, Sung-Woon Jung, Hyuk-Ju Kwon, and Sung-Hak Lee. Rainwater-removal image conversion learning with training pair augmentation. *Entropy*, 25:118, 2023. (This article belongs to the Special Issue Information Network Mining and Applications).
- [97] V7 Labs. Generative adversarial networks: A complete guide. <https://www.v7labs.com/blog/generative-adversarial-networks-guide>.
- [98] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.
- [99] Adam Mikolajczyk and Michał Grochowski. Data augmentation for improving deep learning in image classification problem. In *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, pages 117–122. IEEE, 2018.
- [100] Khaled Alomar, Halil Aysel, and Xiaohao Cai. Data augmentation in classification and segmentation: A survey and new strategies. *Journal of Imaging*, 9:46, 02 2023.
- [101] Nahla M Ibrahim, Ahmed Abou ElFarag, and Rania Kadry. Gaussian blur through parallel computing. In *IMPROVE*, pages 175–179, 2021.
- [102] Muhammad Fauzan Rahman, Febryanti Sthevanie, and Kurniawan Nur Ramadhani. Face recognition in low lighting conditions using fisherface method and clahe techniques. In *2020 8th International Conference on Information and Communication Technology (ICoICT)*, pages 1–6, 2020.
- [103] Hong Yang and Travis Desell. Robust augmentation for multivariate time series classification. *arXiv preprint arXiv:2201.11739*, 2022.
- [104] Vikramaditya Jakkula. Tutorial on support vector machine (svm). *School of EECS, Washington State University*, 37(2.5):3, 2006.
- [105] David Weedmark. A 4-step guide to machine learning model deployment. <https://domino.ai/blog/machine-learning-model-deployment>, 2021. Accessed: 2024-06-19.

- [106] Guido van Rossum. Python: A programming language for readability and simplicity. <https://www.python.org/doc/essays/blurb/>, 1996. Accessed: 2024-06-01.
- [107] Kaggle. <https://www.kaggle.com/>. Accessed: June 19 2024.
- [108] Kaggle on crunchbase. <https://www.crunchbase.com/organization/kaggle#/entity>. Accessed: June 2024.
- [109] OpenCV Team. Opencv about page. <https://opencv.org/about/>. Accessed: June 19 2024.
- [110] The PyTorch Foundation. Pytorch foundation. <https://pytorch.org/foundation>, Accessed: 2024-06-19.
- [111] Matplotlib Development Team. Matplotlib: Visualization with python. <https://matplotlib.org/>, 2024. Accessed: June 19, 2024.
- [112] Gradio. <https://www.gradio.app/>. Accessed: 2024-06-19.
- [113] NVIDIA. Nvidia. <https://www.nvidia.com/en-us/>, Accessed: 2024-06-19.
- [114] NVIDIA. Nvidia geforce. <https://www.nvidia.com/en-me/geforce/>, Accessed: 2024-06-19.
- [115] NVIDIA. What is cuda. <https://blogs.nvidia.com/blog/what-is-cuda-2/>, 2021. Accessed: 2024-06-04.
- [116] NVIDIA. Cuda documentation. <https://docs.nvidia.com/cuda/doc/index.html>, 2023. Accessed: 2024-06-04.
- [117] NVIDIA. Cuda toolkit. <https://developer.nvidia.com/cuda-toolkit>, 2023. Accessed: 2024-06-04.
- [118] NVIDIA. Cuda zone. <https://developer.nvidia.com/cuda-zone>, 2023. Accessed: 2024-06-04.
- [119] NVIDIA. Nvidia cudnn documentation hub. <https://developer.nvidia.com/cudnn>, 2024. Accessed: 2024-06-04.
- [120] NumPy Contributors. Numpy project. <https://numpy.org/>, Accessed: 2024-06-04.

- [121] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [122] Hechen Yang, Chen Li, Xin Zhao, Bencheng Cai, Jiawei Zhang, Pingli Ma, Peng Zhao, Ao Chen, Tao Jiang, Hongzan Sun, Yueyang Teng, Shouliang Qi, Xinyu Huang, and Marcin Grzegorzek. Emds-7: Environmental microorganism image dataset seventh version for multiple object detection evaluation. *Frontiers in Microbiology*, 14:1084312, 2023.