Enhancing Gastrointestinal Diseases Classification Using Gastroenterology Imaging and Transfer Learning

Tadiwos Workeye Simegnew

A Thesis Submitted to the Department of Computer Science and Engineering,

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering (Artificial Intelligence)

Office of Graduate Studies

Adama Science and Technology University

September, 2024

Adama, Ethiopia

Enhancing Gastrointestinal Diseases Classification Using Gastroenterology Imaging and Transfer Learning

Tadiwos Workeye Simegnew

Advisor: Mesfin Abebe (Ph.D)

A Thesis Submitted to the Department of Computer Science and Engineering,

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering (Artificial Intelligence)

Office of Graduate Studies

Adama Science and Technology University

September, 2024

Adama, Ethiopia

DECLARATION

I hereby declare that this Master Thesis entitled "Enhancing Gastrointestinal Diseases Classification Using Gastroenterology Imaging and Transfer Learning" is my original work. That is, it has not been submitted for the award of any academic degree, diploma or certificate in any other university. All sources of materials that are used for this thesis have been duly acknowledged through citation.

_____	_____	_____
Name of Student	Signature	Date

RECOMMENDATION OF ADVISORS

I, the advisor of this thesis, hereby certify that I have read the revised version of the thesis entitled “Enhancing Gastrointestinal Diseases Classification Using Gastroenterology Imaging and Transfer Learning” prepared under my guidance by Tadiwos Workeye submitted in partial fulfillment of the requirements for the degree of Mater’s of Science in Computer Science and Engineering. Therefore, I recommend the submission of revised version of the thesis to the department following the applicable procedures.

Major Advisor/Supervisor	Signature	Date

APPROVAL PAGE

I, the advisor of the thesis entitled "Enhancing Gastrointestinal Diseases Classification Using Gastroenterology Imaging and Transfer Learning" and developed by Tadiwos Workeye, hereby certify that the recommendation and suggestions made by the board of examiners are appropriately incorporated into the final version of the thesis.

Major Advisor/Supervisor	Signature	Date

We, the undersigned, members of the board of examiners of the thesis by Tadiwos Workeye have read and evaluated the thesis entitled "Enhancing Gastrointestinal Diseases Classification Using Gastroenterology Imaging and Transfer Learning" and examined the candidate during open defense. This is, therefore, to certify that the thesis is accepted for partial fulfillment of the requirement of the degree of Master of Science in Computer Science and Engineering.

Chair Person	Signature	Date
Internal Examiner	Signature	Date
External Examiner	Signature	Date

Finally, approval and acceptance of the thesis is contingent upon submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department Graduate Council (DGC) and School Graduate committee (SGC).

Department Head	Signature	Date
School Dean	Signature	Date
Office of Postgraduate Studies, Dean	Signature	Date

ACKNOWLEDGEMENTS

First and foremost, I wish to convey my deepest gratitude to my supervisor, Dr. Mesfin Abebe, for his invaluable assistance, support, and motivation throughout this study. His expertise and constructive feedback have been instrumental in shaping this thesis and ensuring its successful completion.

I also wish to extend my thanks to the faculty members and colleagues at Adama Science and Technology University who provided insightful comments and suggestions that greatly enhanced the quality of this work. Their support and camaraderie made the research process more enjoyable and enriching.

I would like to extend my heartfelt thanks to Adera Medical Center for providing GI imaging, and to Dr. Kalieb and Dr. Besufikad for their invaluable support and efficient data labeling, which played a crucial role in the success of this research.

I am deeply grateful to my family and friends for their unwavering support, patience, and understanding throughout my studies. Their belief in me has been a source of encouragement and inspiration.

Finally, I want to thank everyone who, in some way or another, contributed to the success of this thesis. Your encouragement and support have been deeply appreciated.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

ACC	Accuracy
AI	Artificial Intelligence
Bi-GRU	Bidirectional GRU
CNN	Convolutional Neural Networks
CT	Computed Tomography
EWT	Empirical Wavelet Transform
GA	Genetic Algorithm
GI	Gastrointestinal
GIT	Gastrointestinal Tract
GRU	Gated Recurrent Unit
IQI	Information Quality Index
KNN	K-nearest Neighbors
LR	Logistic Regression
LSVM	Linear Support Vector Machine
MCC	Matthews Correlation Coefficient
MLP	Multi-Layer Perceptron
MSE	Mean Square Error
PSNR	Peak signal-to-noise ratio
ResNeSt	Split Attention Network
ResNet	Residual Neural Network
SI	Similarity Index
VIF	Visual Information Fidelity
WCE	Wireless Capsule Endoscopy
FCL	Fully Connected Layer

ABSTRACT

Gastrointestinal (GI) diseases present a significant global health challenge, requiring the advancement of enhanced diagnostic tools for accurate and efficient classification. This study explores the enhancement of GI disease classification through the application of transfer learning, utilizing various deep learning architectures, including ResNeSt50, ResNeSt101, ResNeSt200, ResNet50, and VGG16, applied to the Kvasir V2 dataset and a local prepared Ethiopian dataset. Initially, various models were evaluated using the Kvasir V2 dataset, with ResNeSt50 emerging as the top performer. The best results with ResNeSt50 were achieved using a configuration with a learning rate of 0.01, a dropout rate of 0.5, a weight decay of 1e-3, unfreezing of the last two layers and FCL, scheduler factor of 0.2. This setup achieved a validation set accuracy of 93.33%, a test set accuracy of 93.22%, and a local test set accuracy of 88.75%. To further enhance the model's robustness and accuracy, the Kvasir V2 dataset was incorporated with our local dataset during training. With this combined dataset, ResNeSt50 achieved a validation accuracy of 93.22% and a test accuracy of 93.11%. The model also maintained consistently high metrics across F1 score, precision, and recall, all within the range of 93.32% to 93.39%. This incorporation of the local dataset significantly improved the local test set accuracy from 88.75% to 93.17%, highlighting its strong generalization capabilities across different datasets. The findings underscore the effectiveness of the ResNeSt50 model, particularly with these specific hyperparameter settings and data augmentation strategies, in enhancing GI disease classification. This work emphasizes the power of transfer learning in improving diagnostic accuracy and provides and establishes a basis for future research, including the expansion of datasets and the exploration of other deep learning architectures.

***Keywords*:** Split Attention, Gastrointestinal, ResNeSt, Imaging Techniques, Local Ethiopian Dataset, Transfer Learning, Regional Nuance

CHAPTER ONE

1. INTRODUCTION

1.1. Background of the Study

Gastrointestinal (GI) diseases encompass a diverse array of pathological conditions affecting the digestive system, which mainly encompasses gastrointestinal motility disorders, infectious inflammation (such as Helicobacter pylori infection, cholera, and intestinal parasites), non-infectious inflammation (such as chronic gastritis and Crohn's disease), and gastrointestinal cancers (Wang et al., 2020). These ailments impose a substantial global health burden, with estimations from the Global Burden of Disease Study highlighting their prevalence and impact on morbidity and mortality rates worldwide (Alatab et al., 2020). The complexity and heterogeneity of GI diseases pose significant challenges for accurate diagnosis and timely intervention, emphasizing the critical need for precise disease classification methodologies.

In the identification of gastrointestinal diseases, a spectrum of techniques is employed, encompassing a range of diagnostic modalities. These encompass pH monitoring (Butt & Kasmin, 2020), manometry (Pham, 2020), ultrasound elastography (Ślósarz et al., 2021), serologic testing (Rashid & Lee, 2016), nuclear medicine scans (Sureshkumar et al., 2020), biopsies (Tran-Nguyen et al., 2021), breath tests (Rana & Malik, 2014), genetic testing (Goodman & Chung, 2016), stool tests (Kasırga E., 2019), laboratory tests (Hooshyar & Rostamkhani, 2022), physical examinations, comprehensive medical histories (Daly & Zarate-Lopez, 2021) and smart pill(Cummins G., 2021). While each technique has its own significance in diagnosing specific gastrointestinal conditions, they are not without limitations. These limitations include difficulty interpreting the results, the high cost of testing, and radiation exposure. Additionally, some tests are not widely available or require specialized equipment.

Advancements in medical imaging technologies have redefined disease diagnosis and monitoring in gastroenterology. Modalities like endoscopy, computed tomography (CT), and capsule endoscopy offer invaluable insights into the structural and pathological changes

within the gastrointestinal tract, enabling clinicians to visualize, characterize, and diagnose various GI conditions (Ionescu et al., 2022; Nehra et al., 2022). In spite of these developments in technology, the interpretation of imaging data and accurate disease classification remain complex tasks, often requiring expert clinical judgment and a comprehensive understanding of disease patterns.

Machine learning, particularly deep learning models, has become a potentially fruitful route in healthcare for automating disease classification tasks. Convolutional Neural Networks (CNNs) and their sophisticated architectures have demonstrated remarkable success in medical image analysis, offering the potential to revolutionize GI disease classification (Puttagunta & Ravi, 2021; Suganyadevi et al., 2022). Transfer learning, an adaptation of these models using pre-trained networks and domain-specific data, holds promise for effectively leveraging knowledge from diverse datasets to enhance classification accuracy (Yu et al., 2022).

In the Ethiopian context, gastrointestinal diseases contribute significantly to the country's healthcare burden, but limited access to specialized expertise and diagnostic tools often impedes timely and accurate disease diagnosis (Melak et al., 2023). Bridging this gap through technological innovations like automated disease classification systems tailored to local disease profiles is critical to improving healthcare outcomes and interventions.

This study aims to address the challenges inherent in gastrointestinal disease classification by harnessing the power of transfer learning, particularly the Split Attention Network (ResNeSt), to develop a robust and adaptable model. By integrating local Ethiopian datasets with publicly available resources and applying transfer learning techniques, this research endeavors to create a precise and versatile classification system tailored to the Ethiopian context (Zhang et al., 2020).

1.2. Motivation of the Study

There are significant promises for improving GI diseases classification through transfer learning with the ResNeSt architecture in the context of gastroenterology imaging. The urgent need for a precise and effective diagnosis of gastrointestinal disorders, which place a

significant burden on healthcare systems throughout the world, particularly Ethiopia, is what spurred this study project. Through the use of transfer learning techniques, primarily with the ResNeSt model, the goal is to optimize and localize the learning process using carefully selected Ethiopian data while also utilizing the strength of pre-trained models on publicly available datasets. This method seeks to address the distinct symptoms and prevalence of gastrointestinal disorders in the Ethiopian population, moreover improving the robustness and accuracy of disease classification in the local context. Moreover, by integrating local data into the model, there's an opportunity to create a more tailored and culturally relevant diagnostic tool, potentially improving healthcare outcomes and treatment strategies for Ethiopian patients. Ultimately, this research seeks to bridge the gap between global advancements in AI-driven disease classification and the specific healthcare needs of Ethiopian populations, thereby contributing to more accessible and effective healthcare solutions for gastrointestinal diseases on a local and global scale.

1.3. Statement of the Problem

There is a significant problem in the field of gastroenterology when it comes to imaging techniques for accurately diagnosing and classifying the disease. According to the current studies, as evidenced by Bhardwaj et al. (2023) and Malik et al. (2023), the existing methodologies are limited when it comes to treating a wide range of diseases. This raises concerns regarding the overall capacity to comprehensively diagnose a variety of gastrointestinal conditions. An important yet nuanced concern is whether models are lightweight rather than not, as the aspect is not explicitly addressed in the studies by Gunasekaran et al. (2023), Mohapatra et al. (2023), and Yogapriya et al. (2021).

The study by Hmoud et al. (2021) highlights a potential limitation concerning its dependence on the small dataset of 5,000 images from the Kvasir dataset. This factor may affect the accuracy of training the convolutional neural networks (CNNs), as they cannot guarantee the proper representation of the entire spectrum of gastrointestinal diseases based on the analysis of their features and generalizations. Moreover, the use of a limited number of images inhibits the ability of CNNs to catch all the details and alterations from a broader range of

gastrointestinal disorders. Hence, it will limit the robustness of the model and its diagnostic accuracy.

Moreover, a distinct issue arises concerning the lack of models explicitly tailored to regional nuances and variations, particularly in Ethiopia. Local disease characteristics in Ethiopia may deviate from those in publicly available datasets, as highlighted by Sivari et al., (2023), Gunasekaran et al. (2023) and Mohapatra et al. (2023), necessitating the development of a classification system that is not only precise but also adaptable to the unique disease patterns prevalent in the Ethiopian context. This disparity underscores the urgency of closing the gap between global advancements and the specific healthcare demands of the Ethiopian population, warranting a more nuanced and context-specific approach in disease identification within gastroenterology imaging.

1.4. Research Questions

RQ1. Which pre-trained model is suitable for achieving effective classification of gastrointestinal diseases?

RQ2. Which fine-tuning techniques and transfer learning effectively adapt the model to local Ethiopian diseases while maintaining performance on the publicly available dataset?

RQ3. What impact does integrating local Ethiopian disease data have on the model's performance in classifying gastrointestinal diseases?

1.5. Objective of the Study

1.5.1 General Objective

The general objective of the study is to implement an enhanced Gastrointestinal Diseases Classification using Gastroenterology Imaging through Transfer Learning with established base models.

1.5.2 Specific Objectives

- To collect local gastroenterology imaging data from the Ethiopian Medical Center and integrate it with the publicly available (Kvasir V2) dataset for comprehensive model training.
- To preprocess both gastroenterology imaging data sets by employing suitable techniques, ensuring data optimization and readiness for subsequent model training.
- To select appropriate pre-trained ResNeSt model, trained on a comprehensive dataset like ImageNet, best suited for the varied classes within the gastrointestinal imaging dataset.
- To apply transfer learning and different fine-tuning techniques, adapt the ResNeSt model to kvasier v2 gastroenterology imaging datas and leverage pre-existing knowledge to enhance disease classification accuracy.
- To apply transfer learning and different fine-tuning techniques, adapt the ResNeSt model to combined gastroenterology imaging datas and leverage pre-existing knowledge to enhance disease classification accuracy.
- To assess the developed model's effectiveness on publicly available dataset and combined dataset using standardized metrics, employ a distinct test dataset to assess its efficacy in accurately classifying diverse gastrointestinal diseases depicted in gastroenterology imaging.

1.6. Scope and Limitations of the Study

1.6.1 Scope of Study

The primary aim of this research is to advance the classification of gastrointestinal diseases utilizing gastroenterology imaging through transfer learning. The study will specifically address the following key aspects within the scope of disease classification:

- **Gastrointestinal Disease Classification:** focusing on the accurate and efficient categorization of various gastrointestinal disorders seen in gastroenterology imaging and ensuring the light weightiness of the model.
- **Utilization of Transfer Learning with ResNeSt:** Leveraging transfer learning strategies using ResNeSt architecture specifically adapted to Ethiopian disease data, optimizing the model's accuracy for classifying locally prevalent gastrointestinal conditions.

- **Integration of Local Ethiopian Data:** Incorporating and emphasizing the significance of local Ethiopian gastroenterology imaging datasets to enhance the model's understanding and precision in classifying Ethiopian-specific gastrointestinal diseases.

1.6.2 Limitation of Study

While the suggested model holds promise for enhancing gastrointestinal disease classification, the following limitations should be acknowledged:

- **Limited Generalizability to Rare Diseases:** The model's performance may vary concerning new or rare disease instances not adequately represented in the training dataset.
- **Resource Constraints:** Resource constraints could potentially limit the scalability and implementation of the proposed model.
- **Ethiopian Dataset Representativeness:** Challenges may arise if the collected data does not comprehensively capture the diversity of gastrointestinal conditions in all regions of Ethiopian population.

1.7 Significance of the Study:

In the field of medicine, this work is highly significant. This study intends to enhance disease classification techniques by fusing state-of-the-art technology with a broad dataset that includes both publicly available (Kvasir V2) and local Ethiopian gastrointestinal imaging data. By customizing classification models to the unique disease profiles of the Ethiopian region, this method promises to improve diagnostic efficacy and precision. This study aims to enable early and accurate disease identification through the use of transfer learning techniques with the ResNeSt architecture. It also provides important insights into the patterns and characteristics of diseases unique to gastrointestinal conditions in Ethiopia, thereby improving healthcare outcomes. The consequences encompass refining scientific knowledge in gastrointestinal imaging as well as streamlining healthcare procedures and resource management.

1.8 Organization of the Thesis

Chapter 1: Introduction

This chapter outlines the research topic and offer contextual information on gastrointestinal diseases classification. It outlines the motivation behind the study, the research problems addressed, the objectives, scope, limitations, and potential applications of the study. Furthermore, it presents an overview of the organization of the thesis.

Chapter 2: Literature Review

In this chapter, relevant literature on gastrointestinal diseases classification is reviewed. It covers background information on various gastrointestinal diseases, existing methods and algorithms for disease classification, as well as a review of related works in the field.

Chapter 3: Research Methodology

This chapter outlines the methodology employed in the study to enhance gastrointestinal diseases classification. It describes the methods used for dataset collection, preprocessing techniques, deep learning algorithms employed for classification, and the evaluation metrics employed to measure model performance.

Chapter 4: Proposed Approach for enhancing gastrointestinal diseases classification

The proposed approach for enhancing gastrointestinal diseases classification using transfer learning is presented in detail. This includes a description of the novel methods or techniques proposed for improving classification accuracy and effectiveness, such as innovative preprocessing methods, transfer learning and fine tuning methods.

Chapter 5: Implementation of the proposed approach

This chapter focuses on the practical implementation of the proposed approach. It describes the experimental setup, including the software tools and libraries used, dataset characteristics, preprocessing steps applied, model training procedures, and any specific considerations in the implementation process.

Chapter 6: Results and Discussions

In this chapter, the results of the experiments conducted to evaluate the proposed approach are presented and analyzed. It discusses the performance of the enhanced classification system compared to existing methods, highlighting any improvements achieved and addressing any challenges encountered during the experimentation process.

Chapter 7: Conclusion

The important findings, contributions, and implications of the study are compiled in this last chapter, which also serves to end the thesis. It also discusses potential future research directions and areas for further investigation in the field of gastrointestinal diseases classification.

CHAPTER TWO

2. LITERATURE REVIEW

2.1 Overview of Gastrointestinal (GI) diseases

The term "gastrointestinal" (GI) illnesses refer to a broad category of medical conditions affecting the digestive tract. These include a range of illnesses, including intestinal parasites, cholera, Helicobacter pylori infections, and abnormalities affecting the motility of the gastrointestinal tract. This wide range of digestive disorders is also greatly influenced by gastrointestinal malignancies and non-infectious inflammations like Crohn's disease and chronic gastritis. All of these diseases present significant difficulties for gastroenterologists in terms of diagnosis, therapy, and general management (Wang et al., 2020).

Symptoms of GI diseases range from abdominal pain, bloating, and diarrhea to more severe complications such as rectal bleeding and unintended weight loss. These symptoms can significantly impact patients' quality of life and often require timely and accurate diagnosis for effective management and treatment (Zeng et al., 2022).

Despite advances in medical science, diagnosing GI diseases accurately remains a complex task due to the overlap of symptoms and the need for comprehensive evaluation. Therefore, understanding the intricacies of GI disorders is crucial for developing innovative approaches, such as utilizing medical imaging and deep learning, to enhance diagnostic accuracy and improve patient outcomes.

2.2. Role of Medical Imaging in Gastroenterology:

Medical imaging plays a pivotal role in the diagnosis and management of gastrointestinal (GI) diseases by providing clinicians with valuable insights into the structure and function of the digestive tract (Badea, et al., 2014). Various imaging modalities are utilized in gastroenterology, each offering unique strengths and limitations.

Endoscopy, including upper and lower gastrointestinal endoscopy, allows for direct visualization of the GI tract, enabling the detection of abnormalities such as ulcers, polyps, and tumors (Di Giulio et al., 2016). Computed tomography (CT) and magnetic resonance

imaging (MRI) provide detailed cross-sectional images of the abdomen (De Backer et al., 2006), aiding in the diagnosis of conditions such as inflammatory bowel disease (IBD), pancreatic disorders, and liver tumors. Ultrasonography offers a non-intrusive technique for assessing abdominal organs and detecting conditions such as gallstones and liver cirrhosis.

While medical imaging plays a crucial role in diagnosing GI diseases, it is not without limitations. Interpretation of imaging findings can be subjective, and certain conditions may require multiple imaging modalities for accurate diagnosis. Additionally, concerns regarding radiation exposure and contrast agent usage underscore the need for careful consideration when selecting imaging techniques for individual patients (Sun et al., 2023).

2.3. Machine Learning and Deep Learning in Medical Imaging:

Machine learning (ML) techniques have revolutionized medical imaging by enabling automated analysis of complex imaging data, leading to improved diagnostic accuracy and efficiency (Varoquaux & Cheplygina, 2023). In the context of gastroenterology, ML algorithms are increasingly being applied to interpret gastrointestinal (GI) images and aid in disease classification and diagnosis (van der Sommen et al., 2020).

Deep learning (DL) is an advanced approach of machine learning (ML) that use learning techniques to develop specific knowledge on how machines should respond to humans. One of the key advantages of DL in medical imaging is its ability to extract meaningful patterns and features from large datasets of GI images, allowing for the detection of subtle abnormalities that may be challenging for human observers to identify (Rana, et al., 2023). Convolutional Neural Networks (CNNs), a type of deep learning architecture, have emerged as particularly powerful tools for image analysis due to their ability to automatically learn hierarchical representations of image features (Sharma, et al., 2018).

However, challenges remain in the application of ML and DL in medical imaging, including the need for large and diverse training datasets, potential biases in data collection, and the interpretability of ML models. Additionally, ethical considerations such as patient privacy and data security must be carefully addressed to ensure the responsible deployment of ML algorithms in clinical practice. Despite these challenges, the integration of ML techniques

into medical imaging holds immense promise for advancing the field of gastroenterology and improving patient care.

2.4 Convolutional Neural Networks (CNNs)

CNNs are a class of deep learning models specifically designed for processing and analyzing visual data, including images and videos. CNNs are inspired by the organization of the visual cortex in the human brain, with layers of neurons that progressively extract features from raw pixel inputs (Sharma, et al., 2018).

Convolutional layers, which apply a number of learnable filters to the input data to find local patterns and features, are one of the primary innovations of CNNs. These filters conduct convolutions as they move across the input image, creating feature maps that capture various facets of the spatial organization of the input. Other layer types that are commonly used in CNNs include pooling layers, which use feature maps that have been downscaled to reduce computational complexity, and fully connected layers, which use the extracted features to do classification or regression.

CNNs have demonstrated remarkable success in various computer vision tasks, including image classification, object detection, segmentation, and more recently, medical image analysis (Chen, et al., 2022). They excel at tasks requiring the comprehension of intricate visual patterns and relationships because of their hierarchical architecture, which enables them to automatically learn from raw image data and derive meaningful representations of characteristics.

In the area of medical imaging, CNNs have shown great promise in assisting healthcare professionals with tasks include identifying lesions, diagnosing diseases, and organizing treatments. By leveraging large datasets of annotated medical images, CNNs can learn to identify subtle abnormalities and patterns indicative of specific diseases or conditions, ultimately aiding in more accurate and timely diagnosis (Sarvamangala et al., 2022); Puttagunta & Ravi, 2021; Suganyadevi et al., 2022).

2.5 Transfer Learning in Healthcare

Transfer learning has manifested as a groundbreaking technique in machine learning, offering significant advantages in various domains. By leveraging knowledge gained from pre-trained models on big datasets like image net, transfer learning allows for the adaptation and fine-tuning of models for new tasks with limited datasets, significantly reducing the computational demand and time required for model training. This approach enables researchers and practitioners to build highly accurate and efficient machine learning models without starting from scratch, thus facilitating faster experimentation and iteration in model development (Yu et al., 2022).

In the healthcare setting, transfer learning holds immense promise for revolutionizing medical image analysis and diagnostic processes (Kim et al., 2022). By fine-tuning pre-trained models with domain-specific medical imaging data, transfer learning enables the development of highly accurate and robust diagnostic tools tailored to specific healthcare applications. This approach expedites the development and deployment of machine learning models for medical image analysis, enhancing diagnostic accuracy, efficiency, and generalization capabilities. Furthermore, transfer learning facilitates the adaptation of models to new imaging modalities, diseases, or patient populations, ultimately improving patient care and clinical outcomes in a variety of healthcare settings.

2.6 Attention mechanisms in deep learning

Attention mechanisms play a pivotal role in enhancing deep learning models by enabling them to selectively focus on pertinent information within input data while disregarding irrelevant details (Niu, et al., 2021). Originally inspired by human visual attention, these mechanisms have become integral components across various deep learning architectures, particularly in tasks involving sequential or spatial data. By giving useful weights to different parts of the input data, attention mechanisms empower the model's ability to change its focus based on the relevance of each element to the task at hand. For instance, in machine translation tasks, attention mechanisms aid in pinpointing relevant words in the source sentence to generate accurate translations in the target language.

Attention mechanisms, including self-attention and split-attention, serve as versatile tools for enhancing the efficiency of deep learning models. In self-attention mechanisms, neural network layers compute attention weights based on input data, enabling the model to selectively emphasize relevant information (Hernández & Amigó, 2021). Conversely, split-attention involves Dividing the feature maps into multiple groups and performing attention mechanism within each group. This enables the model to focus on various feature map regions independently and capture diverse information. These mechanisms have showcased notable effectiveness various uses, including image captioning, machine translation, and medical image analysis, highlighting their pivotal role in elevating the performance of deep learning models across diverse domains.

2.7 Recent Advances in Transfer Learning Architectures

Transfer learning has become increasingly prevalent in the field of machine learning, particularly in the domain of computer vision and medical imaging (Kim et al,. 2022). Recent years have seen the development of and refined a variety of transfer learning architectures that have significantly improved the model's performance on new tasks with limited labeled data. In this section, we explored some of the most notable recent advances in transfer learning architectures, with a focus on their applications in medical imaging and healthcare.

Inception Networks (GoogLeNet): Inception Networks, also known as GoogLeNet, were presented by Szegedy et al. in 2014 as an alternative architecture to traditional convolutional neural networks (CNNs). The key innovation of Inception Networks is the employment of inception modules, which are made up of several parallel convolutional filters with varying strides and widths. This allows the network to capture both local and global features at multiple scales simultaneously, leading to improved representation learning and feature extraction (Kapoor et al., 2022).

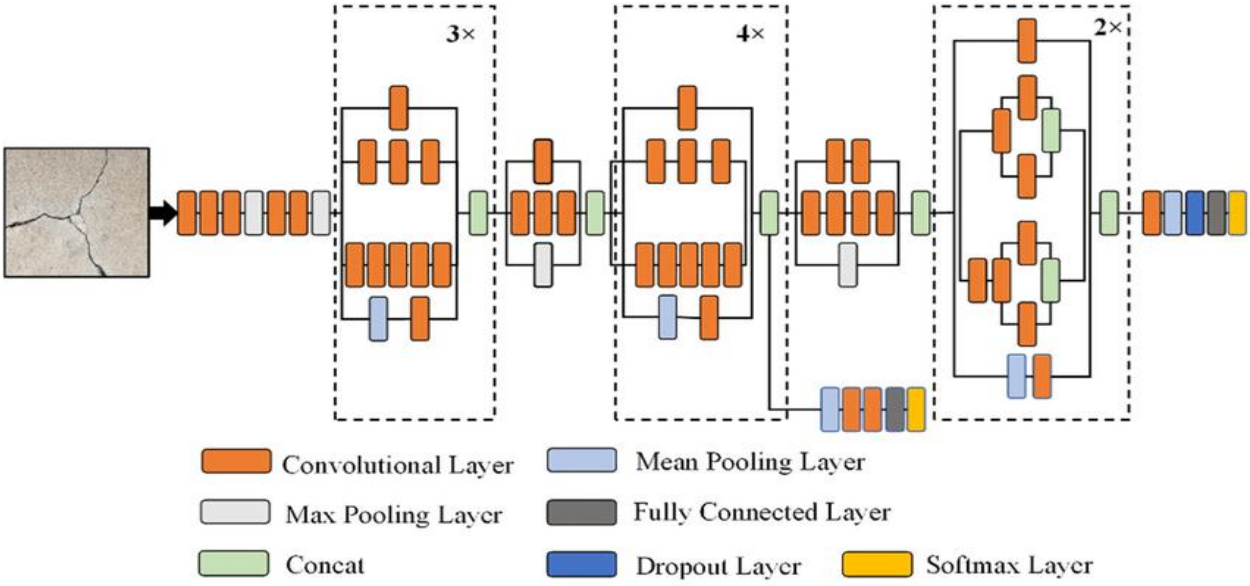


Figure 2.1: Architecture of Inception v3 Network (Ali et al., 2021)

In transfer learning, pre-trained Inception models, such as InceptionV3 and InceptionResNetV2, have demonstrated strong performance in medical imaging tasks (Kapoor et al., 2022). These models are particularly well-suited for tasks that require fine-grained feature extraction and spatial hierarchies, such as tumor segmentation in MRI or CT scans. By leveraging the hierarchical representations learned by pre-trained Inception models, researchers can effectively transfer knowledge from generic image datasets to domain-specific medical imaging datasets, leading to improved performance and generalization.

Efficient Net: Efficient Net, introduced by Tan et al. in 2019, represents a new variety of CNN architectures that achieve cutting-edge results with a great reduction in the number of parameters and processing power when compared to earlier models. The key idea behind Efficient Net is compound scaling, which uniformly scales the network's depth, width, and resolution to achieve optimal performance across a wide range of resource constraints(Tan & Le, 2019)

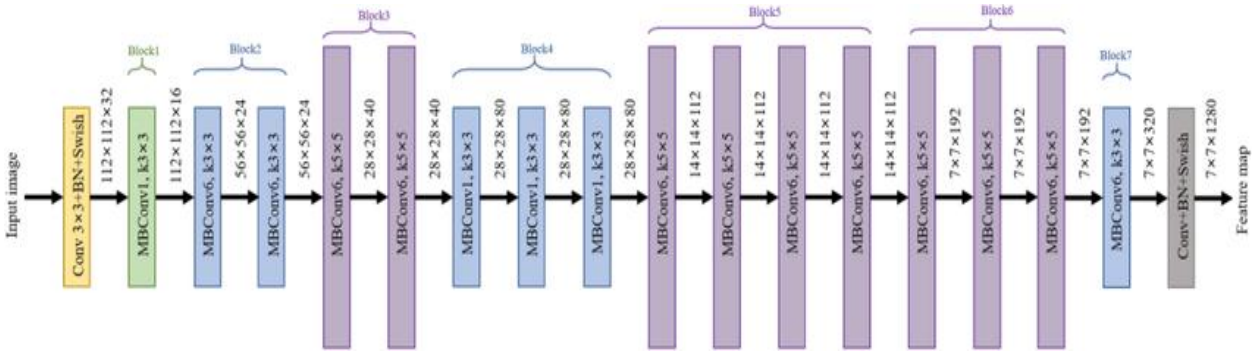


Figure 2.2: Architecture of Efficient Net-B0 (Zhou et al., 2023)

In transfer learning, EfficientNet models have shown promise for medical imaging tasks, where computational efficiency and model size are often critical considerations (Raza et al., 2023). By fine-tuning pre-trained EfficientNet models on medical imaging datasets, researchers can achieve high levels of accuracy while minimizing computational costs and memory requirements. This makes EfficientNet particularly ideal to be used on devices with limited resources, like smartphones or edge computing devices, where computational resources are constrained.

Residual Networks (ResNet): Residual Networks, or ResNets, represent a breakthrough in deep learning architectures that have revolutionized image classification tasks. Introduced by He et al. in 2015, ResNets introduce skip connections, also known as residual connections, which allow the network to learn residual mappings rather than the desired underlying mappings directly, so addressing the difficulty of training very deep neural networks. This lessens the issue caused by vanishing gradients and enables the training of much deeper networks with hundreds or even thousands of layers (He et al.,2015).

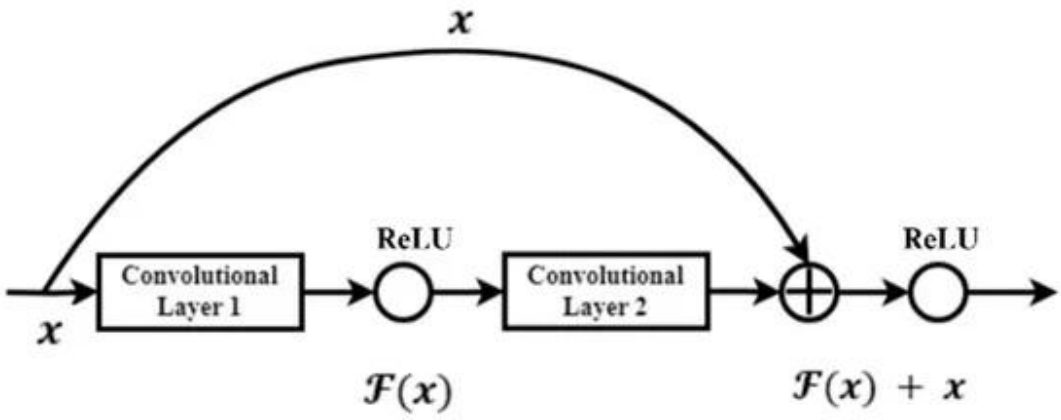


Figure 2.3: Residual block or skip connection on ResNet. (Hasanah et al., 2023)

In the perspective of transfer learning, pre-trained ResNet models, like ResNet-50 and ResNet-101, have become popular choices for fine-tuning on medical imaging datasets. These models, which have been pre-trained on extensive image classification datasets like ImageNet, capture rich hierarchical representations of visual features that are transferable to a variety of tasks using medical imaging. By optimizing these previously trained ResNet models on smaller, domain-specific datasets of medical images, researchers can achieve cutting edge effectiveness in tasks such as disease classification, lesion prediction, and anatomical segmentation.

ResNeSt: In recent years, ResNeSt has emerged as a powerful architecture for transfer learning in medical imaging and computer vision tasks. ResNeSt, short for Split Attention Network, builds upon the foundation of ResNet by introducing a novel attention mechanism that enhances feature representation learning. Introduced by Zhang et al. in 2020, ResNeSt incorporates a modified attention module called the "Residual Attention Module" (RAM), which captures rich spatial dependencies and contextual information within feature maps.

The ResNeSt architecture leverages the hierarchical structure of split attention mechanisms to obtain both local and public features effectively. This allows the model to attend to relevant regions of the input image adaptively, focusing on informative features while suppressing irrelevant distractions. By incorporating attention mechanisms at multiple levels of the network, ResNeSt enhances the discriminative power of the learned representations,

leading to improved performance in image classification, object detection, and segmentation tasks (Zhang et al., 2020)

In transfer learning, pre-trained ResNeSt models have demonstrated strong performance in medical imaging applications, where accurate feature representation learning is crucial for diagnostic accuracy. By fine-tuning pre-trained ResNeSt models on medical imaging datasets, researchers can leverage the model's ability to capture intricate spatial relationships and contextual information, leading to improved performance and generalization on tasks such as disease classification, lesion detection, and anatomical segmentation.

Moreover, ResNeSt's efficient attention mechanisms and hierarchical feature extraction make it ideal for implementation on devices with limited resources, like mobile phones or edge computing devices, where computational resources are limited. This allows for the advancement of efficient and extendable diagnostic tools that can be launched in a wide range of clinical settings, improving access to healthcare and patient outcomes.

2.7.1 Architecture Overview of ResNeSt

ResNeSt (Split Attention Network) is a CNN architecture that builds upon the foundation of ResNet by introducing a novel attention mechanism called the Residual Attention Module (RAM). ResNeSt enhances feature representation learning by incorporating split attention mechanisms, empowering the architecture to capture both local and global features effectively.

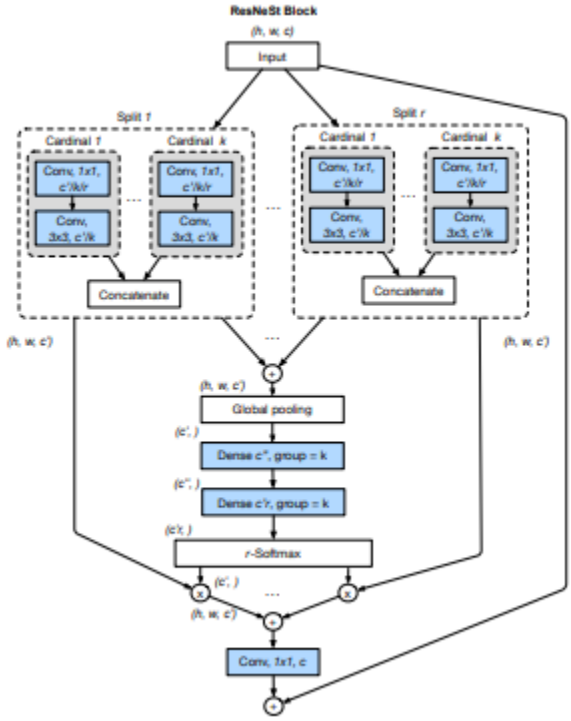


Figure 2.4: Implementation ResNeSt blocks (Zhang et al., 2020)

2.7.2 Common Term of ResNeSt

Residual Blocks: It is the fundamental components of the ResNeSt. Multiple convolutional layers and shortcut connections that omit a layer or layers are present in every residual block. This makes it possible to train very deep networks and helps to alleviate the issue of disappearing gradients.

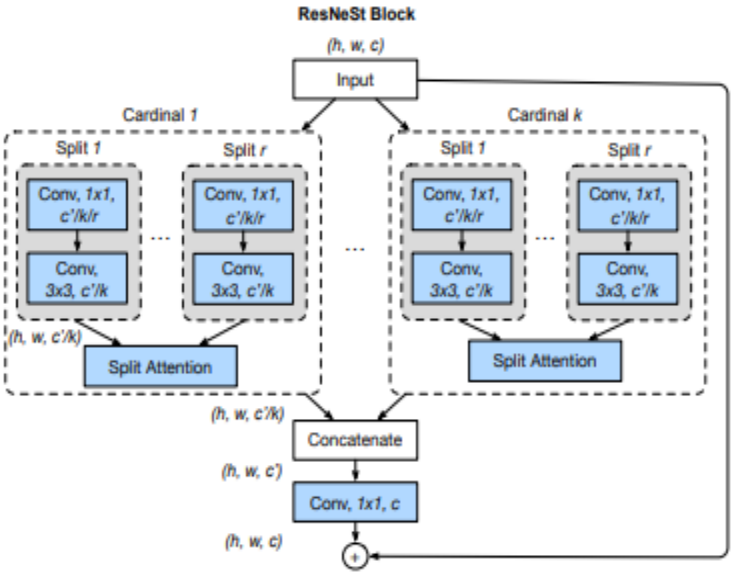


Figure 2.5: ResNeSt block (Zhang et al., 2020)

Cardinal: refers to a group of feature maps that are divided into multiple branches for split attention. Each cardinal group consists of a subset of the feature maps within a given stage or block of the network. Split attention is performed independently within each cardinal group, allowing the model to attend to different parts of the feature maps simultaneously and capture diverse information.

Convolutional Layers: These are the layers in CNN responsible for extract features from the images, such as edges, textures, and patterns, by applying convolutional procedures to the input data.

Split Attention: Split attention, involves dividing the feature maps into multiple groups and performing attention mechanisms within each group. This allows the model to focus on different parts of the feature maps independently and capture diverse information. In ResNeSt, split attention is typically implemented within each stage or block of the network.

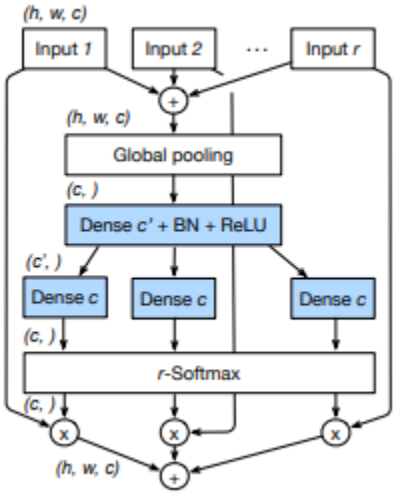


Figure 2.6 : Illustrates the concept of split attention within a cardinal group (Zhang et al., 2020).

Global pooling: often referred to as global average pooling (GAP), is is a method for reducing each feature map's spatial dimensions to a single value. It achieves this by taking the average of all the values in each feature map, resulting in a fixed-size representation regardless of the input image size. Global pooling helps to enforce translational invariance and minimize the number of parameters in the network, facilitating faster training and inference

Softmax activation function: is commonly employed to transform the output of the previous layer into a probability distribution over the classes in a classification task. Softmax normalizes the output scores across all classes, ensuring that they sum up to one and can be interpreted as probabilities.

Pooling Layers: Using downsampled feature maps, pooling layers minimize the spatial dimensions of the feature maps while preserving the most pertinent data.This helps decrease computational complexity and parameter size, making the network more efficient.

Fully Connected Layers: Fully connected layers perform classification based on the extracted features. They map the high-dimensional feature representations to the output classes, enabling the network to make predictions.

In conclusion, recent advances in transfer learning architectures have significantly advanced the state-of-the-art in medical imaging and healthcare. From ResNets and Inception Networks to EfficientNet, and others, researchers have developed a diverse array of models that leverage pre-trained representations to achieve superior performance on a wide range of tasks. By fine-tuning these pre-trained models on domain-specific medical imaging datasets, researchers can leverage the knowledge encoded in large-scale image datasets to improve diagnostic accuracy, efficiency, and generalization in healthcare applications. As transfer learning continues to evolve, we can expect further advancements in medical imaging and healthcare, leading to improved patient outcomes and quality of care.

2.8 Related Works

(Gunasekaran et al., 2023) The research utilized the KVASIR v2 dataset, comprising 8 classes and 8000 samples, to propose the GIT-Net ensemble model. This model amalgamated predictions from three pre-trained models—DenseNet201, InceptionV3, and ResNet50—employing two methods: model averaging and weighted averaging. Utilizing unmodified weights from the ImageNet dataset, the researchers froze the initial ten layers of the pre-trained models to expedite training time. The weighted average ensemble model emerged as the top performer, achieving a notable accuracy of 95%, surpassing both the model average ensemble and individual models. These findings underscore the effectiveness of employing an ensemble approach to classify features that individual base learners might inaccurately discern. Evaluation was based on accuracy, with the model averaging ensemble at 92.96% and the weighted average ensemble at 95.00%.

(Sivari et al., 2023) utilized two publicly available datasets, Kvasir and HyperKvasir, comprising numerous endoscopic images depicting various gastrointestinal tract findings. To ensure model compatibility and consistency, the images underwent preprocessing, including resizing and normalization. Focusing on three distinct CNN models based on the VGG16 architecture as base learners, the researchers leveraged transfer learning by incorporating pre-trained weights from ImageNet, thereby expediting model training and bolstering feature extraction capabilities. Each base learner underwent independent training on the prepared datasets, cultivating diverse insights into the image data. Subsequently, the researchers

constructed four unique stacking ensemble models, with each ensemble integrating the three trained base learners. To synthesize insights from the base learners, each ensemble employed a distinct meta-learner, such as a multi-layer perceptron (MLP), logistic regression (LR), linear support vector machine (LSVM), and K-nearest neighbors (KNN). The primary role of the meta-learners was to amalgamate and refine knowledge derived from the base learners, culminating in a unified classification approach. The stacking ensemble models surpassed the performance of individual deep learning models, exhibiting notably high accuracy and sensitive measurements. In the KvasirV2 dataset, the stacking ensemble models achieved an accuracy of 98.42% (ACC) and a Matthews correlation coefficient (MCC) of 98.19%, while in the HyperKvasir dataset, they attained 98.53% ACC and 98.39% MCC. This approach notably enhanced the performance of deep learning models, presenting objective and reliable outcomes compared to prevailing state-of-the-art studies. Evaluation metrics encompassed accuracy and the Matthews correlation coefficient (MCC), supplemented by McNemar's statistical test to reinforce the findings.

(Mohapatra et al., 2023) utilized the publicly available HyperKvasir dataset for their experimental work, employing a method comprising five stages for endoscopic image classification. These stages encompassed image resizing, pre-processing, augmentation, empirical wavelet transform (EWT) for pattern extraction, and classification through a deep convolutional neural network (CNN). By decomposing images using EWT to extract specific patterns, the deep CNN facilitated disease classification at two levels. The proposed model demonstrated an accuracy of 96.65% with a Matthews correlation coefficient (MCC) of 0.9298 in the first level of classification and 94.25% accuracy with an MCC of 0.8108 in the second level. Additionally, the authors conducted a comparative analysis with other contemporary techniques to establish the efficacy of their proposed method.

(Nouman et al., 2023) employed a dataset containing endoscopic images depicting gastrointestinal (GI) tract diseases to propose a technique comprising two primary steps. Initially, an optimized brightness-controlled contrast enhancement method, facilitated by a genetic algorithm (GA), was applied to enhance the contrast of wireless capsule endoscopy (WCE) images. This involved adjusting brightness and contrast values within the images. Subsequently, data augmentation techniques were implemented on the WCE images,

followed by transfer learning using a modified pre-trained model. The extracted features underwent classification via multiple machine-learning classifiers to facilitate disease classification.

Results revealed an overall enhancement in accuracy by 15.26%, precision by 13.3%, recall rate by 16.77%, and F-measure by 15.18% compared to the original dataset. Notably, the proposed framework outperformed state-of-the-art techniques for the classification of GI tract diseases.

Evaluation of the technique included reporting on the quality enhancement of WCE images using qualitative measures such as peak signal-to-noise ratio (PSNR), mean square error (MSE), visual information fidelity (VIF), similarity index (SI), and information quality index (IQI). Additionally, the classification performance was assessed based on accuracy, precision, recall rate, and F-measure, providing comprehensive insights into the efficacy and improvements achieved by the proposed method.

(Malik et al., 2022) utilized publicly available databases to conduct a study focused on diagnosing ulcerative colitis, polyps, and dyed-lifted polyps through wireless capsule endoscopy (WCE) images. They proposed and applied four multi-classification deep learning (DL) models to the dataset: Vgg-19 + CNN, ResNet152V2, Gated Recurrent Unit (GRU) + ResNet152V2, and ResNet152V2 + Bidirectional GRU (Bi-GRU). Remarkably, the Vgg-19 + CNN model outperformed the other models, achieving an impressive accuracy of 99.45%. The findings showcased the superior performance of the Vgg-19 + CNN model compared to recent state-of-the-art classifiers in accurately classifying gastrointestinal (GI) diseases using WCE images. Evaluation metrics encompassed various parameters such as accuracy, loss, Matthew's correlation coefficient (MCC), recall, precision, negative predictive value (NPV), positive predictive value (PPV), and F1-score, providing a comprehensive assessment of the DL classifiers' classification performance**.**

(Hmoud et al., 2021) utilized the Kvasir dataset, comprising 5,000 images evenly distributed among five types of lower gastrointestinal diseases, for their study. They developed a diagnostic system for gastrointestinal disease diagnosis employing deep learning techniques, specifically GoogleNet, ResNet-50, and AlexNet. Prior to input into the deep learning

networks, the images underwent enhancement and noise removal. Transfer learning was applied to fine-tune the pretrained convolutional neural network (CNN) models, while classification utilized the softmax activation function. Remarkably, all CNN models yielded exceptional results, with AlexNet achieving notable performance metrics: an accuracy of 97%, sensitivity of 96.8%, specificity of 99.20%, and an impressive AUC (area under the curve) of 99.98%. The evaluation metrics encompassed accuracy, sensitivity, specificity, and AUC, underscoring the robustness of the models' diagnostic potential for gastrointestinal diseases.

(Yogapriya et al., 2021) combined traditional image processing algorithms and a data augmentation technique with pre-trained deep convolutional neural network (CNN) models to classify gastrointestinal tract diseases from wireless endoscopy images. Utilizing pretrained models such as VGG16, ResNet-18, and GoogLeNet with adjusted fully connected and output layers, the proposed models are validated using a dataset containing 6702 images across 8 classes of gastrointestinal tract diseases. In the results and discussion section, the VGG16 model outperformed others, achieving 96.33% accuracy, 96.37% recall, 96.5% precision, and 96.5% F1-measure. Compared to other state-of-the-art models, VGG16 demonstrated the highest Matthews correlation coefficient value of 0.95 and Cohen's kappa score of 0.96. Evaluation of model performance relied on the Matthews Correlation Coefficient (MCC), which serves as a statistical measure, offering a higher rate when prediction results encompass true positive, false positive, true negative, and false negative values.

(Bhardwaj et al., 2023) conducted a comprehensive analysis of deep learning-based approaches for the prediction of gastrointestinal diseases using multi-class endoscopy images . They utilized the Kvasir dataset, which consists of endoscopic images categorized into five classes: dyed-lifted polyps, esophagitis, normal cecum, dyed resection margins, and normal colon . Deep transfer learning models such as DenseNet201, EfficientNetB4, Xception, InceptionResNetV2, and ResNet152V2 were employed for disease detection and classification. The models were evaluated based on various performance metrics, including precision, loss, accuracy, F1 score, root mean square error, and recall. Inception ResNetV2 achieved the highest testing accuracy of 97.32% for detecting dyed-lifted polyps, while

Xception performed efficiently in detecting dyed resection margins (95.88%), esophagitis (96.88%), normal cecum (97.16%), and normal colon (98.88%) . The research aims to address the limitations of previous methodologies and proposes the development of an application for patients to self-diagnose gastrointestinal diseases.

Table 2.1: Summary of Related Works

Study	***DataSets***	***Methodology***	***Gaps***
(Gunasekaran et al., 2023)	Kvasir V2	This model combined results from DenseNet201, InceptionV3, and ResNet50 using averaging (simple & weighted).	Dependability on the performance of the base learners. If the base learners are not accurate, the ensemble model(GIT-NET) will not be accurate either
(Sivari et al., 2023)	Kvasir V-2 & Hyper-Kvasir.	The research employed two gastrointestinal image datasets, preprocessed for model consistency. It then utilized VGG16-based CNNs, implementing transfer learning and ensemble methods with diverse meta-learners for unified classification refinement.	Kvasir and HyperKvasir predominantly reflect Western populations, potentially limiting the generalizability and accuracy of models for populations like Ethiopians, characterized by distinct disease manifestations and genetic factors.
(Mohapatra et al., 2023)	hyper-Kvasir	Mohapatra et al. (2022) used the HyperKvasir dataset, employing five stages for endoscopic image classification. Their method achieved high accuracy (96.65% and 94.25%) in two classification levels, utilizing empirical wavelet transform and deep CNN, while also comparing	Combining EWT with CNNs can be computationally demanding, especially for resource-constrained environments like some clinical facilities. The paper does not address optimizations or alternative network architectures for

		its efficacy against contemporary techniques.	improving efficiency
(Nouman et al., 2023)	Kvasir V-2 & Hyper-Kvasir.	The methodology involves enhancing image contrast using a genetic algorithm for wireless capsule endoscopy images, followed by data augmentation and transfer learning with modified pre-trained models.	MobileNet-V2 was utilized for feature extraction, yet there was no comparative analysis conducted with alternative pre-trained models.
(Yogapriya et al., 2021)	Kvasir V2	The proposed models utilize pretrained models VGG16, ResNet-18, and GoogLeNet, with adjusted fully connected and output layers .	Limited generalizability in diverse healthcare settings, especially resource-constrained areas like Ethiopia, due to varied factors.
(Malik et al., 2023)	CVC-ClinicDB,Kvasir	They employed four deep learning models to classify GI diseases from wireless capsule endoscopy images, with Vgg-19 + CNN yielding the highest accuracy of 99.45%, surpassing other models.	The study only included a small number of diseases, and the models may not be able to accurately diagnose other GI conditions.

(Hmoud et al., 2021)	Kvasir	Comparison among three networks, GoogleNet, ResNet-50, and AlexNet, resulted in achieved accuracies of 96.7%, 95%, and 97%, respectively.	The study's dependence on a small dataset of 5,000 images from the Kvasir dataset could limit the ability to effectively train Convolutional Neural Networks (CNNs) to comprehensively represent diverse gastrointestinal diseases through feature extraction and generalization.
(Bhardwaj et al,. 2023)	Kvasir	Assessed deep learning models on Kvasir dataset images, with InceptionResNetV2 achieving 97.32% accuracy in detecting dyed-lifted polyps and Xception efficiently identifying diverse gastrointestinal conditions, aiming for a patient-oriented self-diagnosis application.	The study only included a small number of diseases, and the models may not be able to accurately diagnose other GI conditions.

Generally speaking, using imaging techniques to effectively diagnose and categorize intestinal diseases presents a substantial challenge for the profession of gastroenterology. Current approaches usually encounter difficulties in handling the variety of diseases and the need for light weightiness. Moreover, there are insufficient models that are explicitly designed to account for regional subtleties and differences, particularly in Ethiopia, where local illness patterns and characteristics may differ from those found in datasets that are made publically available. In order to improve the accuracy and efficacy of disease identification in gastroenterology imaging within this particular context, this disparity calls for the development of a more accurate and flexible classification system that not only addresses the complexities of publicly available datasets but also integrates local Ethiopian disease datasets.

CHAPTER THREE

3. RESEARCH METHODOLOGY

3.1. Overview of Methodology

The chapter discussed the methodology which encompasses a multi-tiered approach aimed at developing an advanced model for gastrointestinal disease classification in gastroenterology imaging. It involves the curation of a diverse dataset incorporating both publicly available and local Ethiopian gastroenterology imaging data, followed by meticulous preprocessing to refine the dataset for model training. Employing transfer learning techniques coupled with ResNeSt architecture, the pre-trained model undergoes adaptation to the specific attributes of gastrointestinal diseases. The model's efficacy is systematically evaluated using standardized metrics on distinct test datasets to gauge its accuracy in delineating various gastrointestinal conditions.

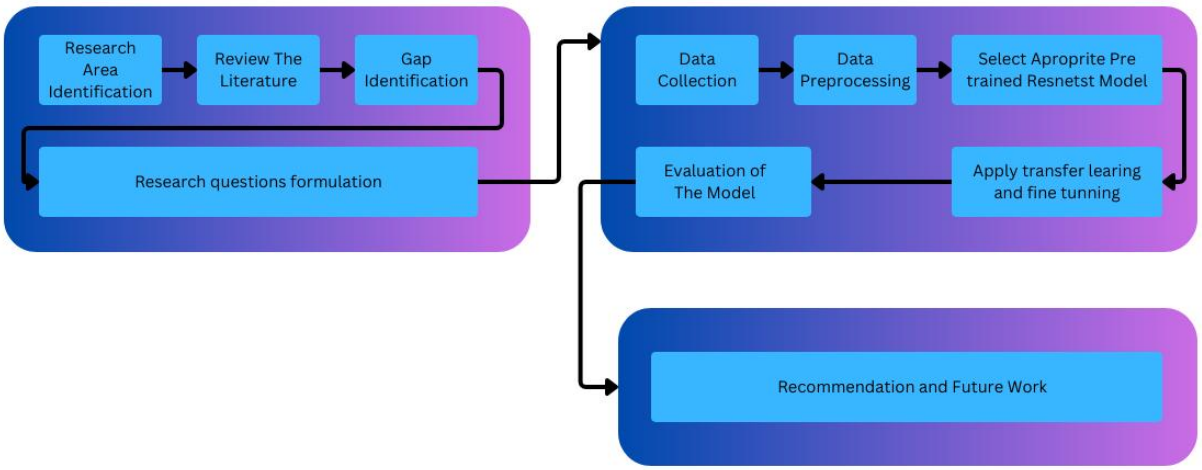


Figure 3.1: Overall process of the study

3.2. Datasets Collection

The datasets utilized in this research comprise a combination of a globally recognized dataset, Kvasir v2, and locally sourced data from Medical Center in Ethiopia (Adera Medical Center), aimed at enhancing the diversity and specificity of gastrointestinal imaging for

disease classification. The approach involves meticulous collection and curation to ensure a comprehensive representation of gastrointestinal pathologies.

3.2.1. Kvasir v2 Dataset

The Kvasir v2 dataset is a meticulously curated collection of gastrointestinal (GI) endoscopy images, meticulously annotated and verified by experienced medical professionals, specifically adept endoscopists. It encompasses eight different classes, each class consisting of 1000 images showcasing various aspects within the GI tract. These classes range from anatomical landmarks like the Z-line, pylorus, and cecum to pathological findings such as esophagitis, polyps, and ulcerative colitis. This comprehensive dataset offers an abundance of images for each class, facilitating diverse applications like image retrieval, machine learning, deep learning, and transfer learning tasks. The dataset comprises images of different resolutions, varying from 720x576 to 1920x1072 pixels, systematically organized into distinct folders based on their content. Notably, certain image classes include a supplementary green picture within the image, illustrating the endoscope's position and configuration within the bowel, by use of an electromagnetic imaging system (ScopeGuide, Olympus Europe) that may support the interpretation of the image.

Table 3.1: Description of KvasirV2 Dataset

Classes	***Description***
Anatomical Landmarks	
Z-line	The Z-line, marking the esophagus-stomach transition, aids in diagnosing conditions like gastro-esophageal reflux. Recognizing it during endoscopy is vital for assessing gastrointestinal health.
Pylorus	The pylorus, situated at the stomach's exit into the small intestine, controls food passage with circular muscles. Essential for endoscopic access to the duodenum, inspecting it is crucial during gastroscopy. A typical view reveals a smooth, circular opening amidst pink stomach mucosa, as shown in the accompanying image.

Cecum	The cecum, the initial section of the large intestine, is pivotal for confirming a thorough colonoscopy, serving as a quality benchmark. Its recognition and documentation are paramount, with the appendiceal orifice being a notable feature. Often, documentation of this orifice, along with scope configuration, validates cecum intubation. An example of the appendiceal orifice, resembling a crescent-shaped slit, is depicted in the accompanying image, with the green inset showcasing the scope's configuration to confirm the cecum's position.
Pathological Findings	
Esophagitis	Esophagitis, characterized by inflammation of the esophagus, manifests as disruptions in the esophageal mucosa near the Z-line. In the accompanying image, red mucosal tongues are evident, protruding from the white esophageal lining. Severity is gauged by the length of these breaks and the extent of mucosal involvement. Typically triggered by factors like gastroesophageal reflux or vomiting, prompt detection is crucial for symptom alleviation and preemptive measures against potential complications. Computer-assisted detection holds promise for evaluating severity and automating reporting processes, offering enhanced clinical management.
Polyps	Polyps, abnormal mucosal growths in the bowel, vary in shape and can potentially develop into colorectal cancer. Early detection and removal are crucial for prevention. Automated detection aids in diagnosis and assessment, enhancing clinical management.
Ulcerative Colitis	Ulcerative colitis, a chronic inflammatory condition affecting the large bowel, ranges from mild to severe, with distinct endoscopic features. Diagnosis relies on colonoscopy, where findings vary from swollen and red mucosa in mild cases to prominent ulcerations in moderate cases. Automated

	assessment systems enhance accuracy in grading disease severity, as illustrated in the image to the left depicting bleeding, swelling, and ulceration, with fibrin covering the wounds.
Polyp Removal	
Dyed and Lifted Polyps	polyps in the gastrointestinal tract that have been stained with dye and lifted from the surrounding mucosa during endoscopy. These polyps typically appear as raised lesions with distinct margins against the surrounding tissue, often exhibiting a change in color due to the application of dye.
Dyed Resection Margins	showing the resection margins of polyps or lesions in the gastrointestinal tract that have been stained with dye during endoscopic resection procedures

3.2.2. Local Data

The Local Ethiopian Dataset, collected from Adera Medical Center, consists of images representing various gastrointestinal anatomical landmarks and pathological findings. This dataset includes images of key anatomical structures in gastroenterology, such as the Z-line, Pylorus, and Cecum. Additionally, it features pathological findings commonly encountered in gastrointestinal endoscopy, including Esophagitis, Polyps, and Ulcerative Colitis. Each class within the dataset is composed of a diverse range of images captured during endoscopic examinations of 7,890 patients. These images were meticulously collected from Adera Medical Center to provide a comprehensive representation of gastrointestinal conditions prevalent in the Ethiopian population, thereby facilitating research and diagnostic efforts in gastroenterology.

Out of the entire dataset, 4,180 images were labeled, categorized into six classes, and meticulously annotated by Ethiopian gastrointestinal experts, Dr. Besufikad Girma and Dr. Kalieb Assefa. The classes and the number of images in each are as follows:

- **Cecum:** 500 images
- **Esophagitis**: 761 images
- **Polyps**:705 images
- **Pylorus**: 715 images
- **Ulcerative Colitis:** 728 images
- **Z-Lines**: 771 images

This diverse dataset provides a comprehensive representation of gastrointestinal conditions prevalent in the Ethiopian population, facilitating accurate and contextually relevant model training.

Expert-Driven Labeling Process

Initial Labeling by Dr. Besufikad Girma:

- *Role*: Dr. Besufikad Girma, a respected physician at Adera Medical Center, led the initial labeling process.
- *Procedure*: Dr. Girma meticulously examined each image, using his extensive medical expertise to assign accurate labels. The dataset was divided into six classes, including "Cecum," "Esophagitis," "Polyps," "Pylorus," "Ulcerative Colitis," and "Z-Lines." Specifically, 500 images were labeled for the "Cecum" class, while the other five classes received between 700 and 771 images each, ensuring comprehensive coverage of gastrointestinal disease manifestations in the Ethiopian population.

Verification by Dr. Kalieb Assefa:

- *Role*: To ensure the highest accuracy, the initial labels assigned by Dr. Girma were reviewed and confirmed by Dr. Kalieb Assefa, another esteemed expert in the field.
- *Procedure*: Dr. Kalieb conducted a thorough review of the labeled images, providing an additional layer of validation. This verification process solidified the dataset's reliability and accuracy across all classes.

This expert-driven labeling process ensures that the dataset is not only diverse but also highly accurate, making it a valuable resource for developing reliable gastrointestinal disease classification models.

3.2.3. The Importance of Collecting Local Data in Ethiopia

We conducted an analysis comparing the Kvasir v2 dataset with local Ethiopian gastrointestinal data, collaborating with local medical experts to identify several variations specific to the Ethiopian context. For instance, we observed a higher prevalence of granulomatous inflammation in the cecum, often linked to schistosomiasis due to exposure to contaminated water sources. Additionally, malnutrition-induced changes in the pylorus, such as atrophy and hypertrophy, were noted, reflecting the high rates of malnutrition in the country. We also found significant cases of tuberculosis-related esophagitis, characterized by mucosal thickening and ulcerations, influenced by the prevalence of tuberculosis in the region. Furthermore, our local dataset revealed features related to gastrointestinal complications from endemic diseases, which are more pronounced in this population. Lastly, unique imaging characteristics associated with traditional dietary habits indicated how specific foods impact gut flora and inflammation patterns. These variations underscore the necessity of incorporating local health contexts into diagnostic models to accurately address the specific health challenges faced by the Ethiopian population.

3.3. Gastrointestinal Diseases Classification Modeling

3.3.1. Dataset Preparation and Preprocessing

In our study, preprocessing served as a critical phase in the development of our classification model. This stage involved transforming the collected data, characterized by noise and extensive volume, into relevant and refined datasets suitable for model training. Given that the dataset encompassed missing values and noisy data, preprocessing addressed these issues to ensure the dataset's compatibility with machine-learning algorithms. Handling missing values and categorical data was paramount to prepare the dataset in a format conducive to model utilization. The significance of preprocessing lay in its ability to mitigate inconsistencies within the data, thereby enhancing the accuracy and efficacy of the

subsequent machine-learning models. By undertaking preprocessing steps for both datasets, we ensured that the data fed into the model were clean and optimized for GI disease classification. Consequently, preprocessing stood as an indispensable component in the completion of our prediction model, facilitating accurate and reliable disease predictions

Data Cleaning and Standardization:

In our research, we employed rigorous data cleaning and normalization techniques for both the global and local datasets to ensure data consistency, eliminate noise, and standardize formats, thereby facilitating seamless integration. Data cleaning involved meticulous scrutiny of the datasets to identify and rectify inconsistencies, anomalies, and inaccuracies that could potentially compromise the integrity of the data. This process included tasks such as identifying and handling missing values, correcting data entry errors, and removing outliers to enhance the overall quality of the datasets.

Data Augmentation Techniques:

In our study, we employed advanced data augmentation techniques to enhance the variability and robustness of our datasets, thereby improving the generalization capability of our models. These augmentation methodologies encompassed a range of transformative operations, including rotation, flipping, and color adjustments, among others. By systematically applying these techniques to the existing dataset, we aimed to introduce diverse variations and permutations of the original data, effectively expanding the dataset's scope and richness.

Data Labeling for Local Ethiopian Data: To enhance the classification accuracy of gastrointestinal diseases within the Ethiopian context, the dataset was meticulously labeled by local medical experts. This process ensured high-quality and contextually relevant labels, reflecting the expertise of professionals familiar with the specific medical conditions prevalent in the region.

Balanced Sampling Strategies Balanced Sampling Strategies: We implemented sampling strategies to ensure equitable representation of both global and local datasets across the training, validation, and test sets, thereby preventing bias toward any particular dataset source.

To address class imbalance and bias, we sampled 500 images from each class across both datasets. By incorporating an equal number of images (500) from all classes, we maintained balance and fairness in our model training process.

3.3.2. Appropriate Model

Following a comprehensive systematic review encompassing a wide array of deep learning architectures, the ResNeSt (Split-Attention Network) model emerged as the optimal choice for our specific task of gastrointestinal disease classification in gastroenterology imaging. This selection process involved a meticulous examination of various factors, including model performance metrics, architectural complexity, computational efficiency, and applicability to our dataset and research objectives.

The ResNeSt architecture, characterized by its innovative incorporation of split-attention mechanisms within residual blocks, stood out for its exceptional performance in image classification tasks. This architectural design enhances feature representation by enabling the network to attend to multiple subspaces within each feature map, thereby promoting more effective feature learning and discrimination. Moreover, the ResNeSt architecture's scalability and flexibility make it well-suited for handling diverse and complex datasets, such as those encountered in medical imaging research.

Furthermore, the ResNeSt model's proven track record in achieving state-of-the-art results across a range of benchmark datasets and competitions underscores its efficacy and reliability in real-world applications. Its robustness to variations in input data and ability to generalize well to unseen samples make it particularly well-suited for our task of gastrointestinal disease classification, where the variability and complexity of clinical images pose significant challenges.

Overall, the ResNeSt architecture's combination of architectural innovation, performance excellence, and adaptability to our research context positions it as the most appropriate model for achieving our objectives of accurate and reliable gastrointestinal disease classification in gastroenterology imaging. Its selection reflects a strategic decision grounded in empirical

evidence and aligns with our commitment to leveraging cutting-edge deep learning techniques to advance medical research and clinical practice.

3.3.3. Rationale for Choosing ResNeSt Variants, ResNet50, and VGG16 Over Other State-of-the-Art Architectures

We conduct the systematic review for The selection of ResNeSt50, ResNeSt101, ResNeSt200, ResNet50, and VGG16 in this study is driven by several key considerations, including their performance, computational efficiency, adaptability for transfer learning, interpretability, and widespread use in medical imaging research. While other advanced architectures, such as Inception-ResNet v2/v4, Inception v4, DenseNet, and Vision Transformers, offer impressive results in specific contexts, ResNeSt and ResNet architectures were chosen for their balanced trade-offs, particularly in a clinical and research context focused on medical imaging.

Balance of Performance and Computational Efficiency

ResNeSt models, particularly ResNeSt50, ResNeSt101, and ResNeSt200, enhance the classic ResNet architecture by introducing a Split Attention mechanism that improves feature representation without drastically increasing computational complexity. Compared to newer models like Vision Transformers or DenseNet, which are computationally heavy and resource-demanding, ResNeSt strikes an optimal balance, making it ideal for tasks such as gastrointestinal disease classification, where computing resources may be limited.

Transfer Learning Adaptability

VGG16, ResNet50, and ResNeSt models are highly suited for transfer learning, especially in medical imaging where labeled data is often limited. These architectures have well-established pre-trained weights on large datasets (e.g., ImageNet), enabling fine-tuning for specific tasks. In contrast, more complex models like Vision Transformers, though powerful, are less mature for transfer learning in the medical imaging domain and require substantial computational resources to fine-tune effectively.

Model Interpretability and Clinical Application

In medical imaging, interpretability is critical, especially when models are used to inform clinical decisions. ResNeSt and ResNet architectures maintain simplicity while enhancing performance through attention mechanisms, making them more interpretable than architectures like Vision Transformers or DenseNet. Although models like Vision Transformers can capture complex patterns, their decision-making processes are often opaquer, making their clinical adoption and validation challenging.

Implementation and Infrastructure

ResNeSt, ResNet50, and VGG16 benefit from established libraries and robust support in major machine learning frameworks such as TensorFlow and PyTorch. This widespread support provides access to pre-trained weights, detailed documentation, and an active community, significantly simplifying model implementation. In contrast, models like Vision Transformers and DenseNet, while powerful, are more computationally expensive in terms of training time and hardware requirements. For research projects with constrained computational resources, such as this one, ResNet and ResNeSt variants offer a more practical balance between performance and resource needs.

Model Maturity and Proven Track Record

VGG16 and ResNet50 have been extensively studied, benchmarked, and used in a wide range of clinical and medical imaging research, including disease classification tasks. Their proven performance and widespread adoption make them highly reliable for replicability and robustness, which are essential for research in this thesis.

ResNeSt, while a more recent variant, improves on the standard ResNet models by incorporating split attention, enhancing feature representations while avoiding the complexity of models like Vision Transformers. This positions ResNeSt as an ideal compromise, leveraging state-of-the-art advancements while maintaining practical feasibility for deployment in medical imaging tasks.

3.3.4. Transfer Learning and Fine-Tuning

In our study, we explored several transfer learning techniques and fine-tuning strategies to identify the most effective approach for classifying gastrointestinal (GI) diseases accurately. Transfer learning involves leveraging knowledge gained from pre-trained models on large datasets and adapting them for new tasks with limited labeled data. For our study, we selected transfer learning techniques that have shown success in similar medical imaging tasks and adapted them to our specific dataset of GI images.

One effective transfer learning technique we considered is feature extraction, where we leverage the pre-trained ResNeSt model to extract relevant features from our GI image dataset which can represent both local and global datasets. This approach allows us to benefit from the ResNeSt model's ability to capture high-level features relevant to disease classification while fine-tuning only the final layers of the model for our specific task. By freezing the earlier layers of the ResNeSt model during training, we can retain the learned representations and adapt them to our GI image classification problem.

In addition to feature extraction, we also explored fine-tuning strategies such as differential learning rates., where we adjust the learning rates for different layers of the pre-trained ResNeSt model during fine-tuning. Instead of using a uniform learning rate across all layers, we assign higher learning rates to the later layers of the model (closer to the output layer) and lower learning rates to the earlier layers (closer to the input layer). This allows us to fine-tune the later layers more aggressively, focusing on adapting them to the specific features of our GI image dataset, while maintaining stability in the earlier layers where more generic features are learned.

By carefully selecting a suitable transfer learning technique, such as feature extraction, and complementing it with fine-tuning strategies like gradual unfreezing, we aim to develop a highly effective model for GI disease classification. This approach allows us to leverage the pre-trained knowledge of the ResNeSt model while adapting it to our specific dataset, ultimately improving the model's performance and accuracy in diagnosing various GI conditions.

3.4. Evaluation

3.4.1. Model Evaluation

Evaluating the performance of a machine learning model is a critical aspect of ensuring its accuracy and reliability in real-world applications. Model evaluation techniques provide insights into how well the model generalizes to unseen data and effectively learns patterns from the training dataset. In our study, we employed a comprehensive approach to model performance evaluation, utilizing both holdout and cross-validation methods.

Holdout evaluation involves partitioning the dataset into distinct subsets: a training set, a validation set, and a test set. This partitioning allows us to train the model on one subset, validate its performance on another, and finally test its generalization ability on the remaining subset. While holdout evaluation offers simplicity and flexibility, it may suffer from high variability due to differences between the training and test datasets.

To address this limitation and ensure robust model evaluation, we employed two data splitting techniques. Specifically, we divided our dataset using two different splits: 60/40 and 70/30. In the 60/40 split, we allocated 60% of the data for training, 30% for validation, and 10% for testing. For the 70/30 split, we used 70% of the data for training, 15% for validation, and 15% for testing. This approach allowed us to evaluate the model's performance on separate training, validation, and test sets, providing a reliable assessment of its generalization ability across different data partitions.

Overall, our approach to model performance evaluation combines the strengths of multiple holdout methods, allowing us to confidently assess the effectiveness and reliability of our machine learning models in accurately predicting gastrointestinal diseases from imaging data.

3.4.2. Model Performance Evaluation Metrics

Evaluating the performance of prediction models is essential for understanding how well they generalize to unseen data and effectively classify instances. In our study, we employed a diverse range of performance evaluation metrics to comprehensively assess the performance of our classifier algorithms.

A. Accuracy:

As a basic measure of the model's overall prediction accuracy, accuracy provides a broad picture of the model's performance by expressing its capacity to classify all of the dataset's classes correctly. It does this by computing the ratio of correctly predicted samples to the total number of samples.

$$\text{Accuracy} = \frac{\text{True Negative} + \text{True Positive}}{\text{True Positive} + \text{False Positive} + \text{True Negative} + \text{False Negative}}$$

Equation 1: Equation of accuracy

B. F1-Score:

An indicator used to assess how well the model balances recall and precision in its predictions is the F1-score. Because it takes into account both positive and negative classes, the F1-score offers important information on how well the model is able to identify various gastrointestinal disorders. As the harmonic mean of precision and recall, it provides a fair evaluation of the model's effectiveness in various settings and classes.

$$\text{F1} = \frac{2}{recall^{-1} + precision^{-1}}$$

Equation 2: Equation of F1 score

C. Precision:

Precision refers to the percentage of correctly predicted positive instances out of all instances predicted as positive, and it is a metric used to assess how accurate the model is at making positive predictions. It is especially useful in situations where reducing false positives is important, like in medical diagnosis, as it shows how well the model can prevent misclassifying negative instances as positive.

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

Equation 3: Equation of precision

D. Recall:

Recall, also known as sensitivity, assesses the model's ability to identify all actual positive instances within the dataset. It calculates the proportion of true positive cases that the model correctly detects out of all the actual positives. By measuring how effectively the model locates all relevant instances of a given class and minimizing false negatives, recall is essential for ensuring comprehensive disease diagnosis.

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

Equation 4: Equation of recall

3.5. Development Tools

The proposed thesis work for this topic was designed and implemented using a variety of development tools. These tools included platforms and tools for developing prototypes, UML modeling tools, and other appropriate tools needed to conduct the research.

These development tools were briefly described in the sections that followed.

3.5.1. Design Tools

Design tools are platforms that are used for creating, presenting, and comprehending design ideas. As a flexible and powerful graphic design tool, Enterprise Architect was used in the development of the suggested system. This tool provided a lightweight yet effective way to produce professional-looking visual representations, including network diagrams, flowcharts, and many other kinds of diagrams.

3.5.2. Hardware Tools

Table 3.2: Hardware tools

No.	***Tools***	***Used for***
1.	RAM	Facilitating efficient data processing and analysis, ensuring optimal computational performance throughout the research endeavor.
2.	GPU	Boosting the processing speed and efficiency of image and analysis tasks, accelerating the training process.
3.	Hard Disk	Providing reliable storage capacity.

3.5.3. Software Tools

Table 3.3: Software tools

No.	***Software Tools***	***Description***
1.	python	Python, a versatile and high-level programming language, is renowned for its readability and extensive library support, making it widely favored for data science, machine learning, and web development.
2.	skit-learn	Scikit-learn, a Python ML library, leverages NumPy and SciPy for efficient data analysis and modeling. Its user-friendly interface suits both beginners and experts, offering diverse machine learning algorithms.
3.	Anaconda	Anaconda, a Python distribution for data science, streamlines package management and environment setup. Pre-packaged with essential tools, it ensures compatibility and reproducibility in projects.
4.	Jupyter notebook	Jupyter Notebook is an open-source web app for creating and sharing documents with live code, visualizations, and text. Widely used in data science, it enables interactive and collaborative environments for reproducible analyses.
5.	Pytorch	PyTorch, developed by Facebook, is a flexible and user-friendly open-source deep learning framework.
6.	PyTorch	PyTorch, an open-source machine learning framework by Google, is widely recognized for its flexibility and scalability in building and training deep learning models, with a robust ecosystem for seamless deployment across platforms.
7.	Keras	Keras, a user-friendly Python API, simplifies building and training neural network models on top of frameworks like TensorFlow.

CHAPTER FOUR

4. Proposed Approach for Enhancing Gastrointestinal Diseases Classification

4.1. Introduction

This chapter presents the envisioned solution for enhancing gastrointestinal disease classification through the utilization of transfer learning techniques. The primary objective is to address the identified problem by curating a comprehensive dataset on gastrointestinal diseases, incorporating both publicly available Kvasir v2 data and data sourced from Adera Health Center. The dataset is meticulously prepared, considering the variety of gastrointestinal conditions, to facilitate multiclass classification and enable a deeper understanding and utilization of the machine learning model.

Following this, the model undergoes rigorous testing and validation procedures to ensure its robustness and reliability in real-world clinical settings. The aim is to refine and optimize the model, enhancing its accuracy and efficacy in gastrointestinal disease classification tasks. Through iterative testing and refinement, the prototype evolves into a sophisticated tool capable of aiding practitioners in diagnosing gastrointestinal diseases with confidence and precision.

The data collection process involved collaboration with Adera Health Center, where gastrointestinal imaging data were obtained, complemented by publicly available data from the Kvasir v2 dataset. Leveraging the methodology outlined in Chapter Three, the dataset was meticulously curated and prepared to ensure its suitability for prediction tasks.

This chapter provides a comprehensive overview of the proposed approach to gastrointestinal disease classification using transfer learning techniques. It outlines the data collection process, dataset preparation methodology, and deployment of the pre-trained deep learning model, laying the foundation for further discussions on the implementation and evaluation of the proposed solution.

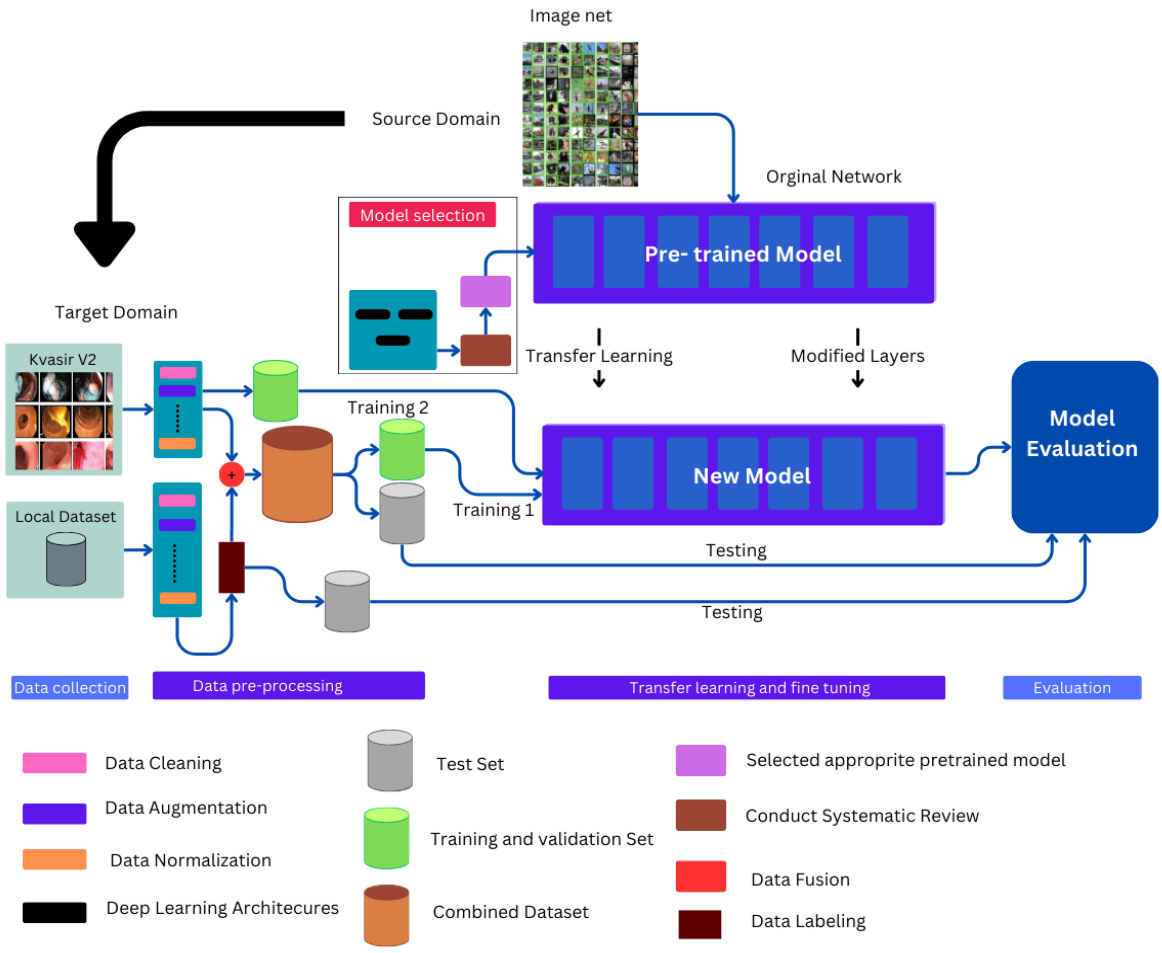


Figure 4.1: Architecture of the Proposed Method

4.2. Data Preprocessing

Data preprocessing plays a crucial role in preparing data for machine learning models, encompassing tasks like data cleaning, data labeling, filling missing values, organizing raw data, and handling categorical values. In this study, our dataset comprises gastrointestinal imaging data from Adera Medical Center, which is unlabeled, along with the publicly available Kvasir v2 dataset.

4.2.1. Data Labeling

To ensure the highest quality and relevance of the data, the labeling was performed by experienced local medical professionals. This meticulous process involved annotating images

from the dataset with specific gastrointestinal conditions, leveraging the expertise of local experts to reflect the unique characteristics of the diseases prevalent in the Ethiopian context. By employing this expert-driven approach, we aimed to achieve a high degree of accuracy in the labeled data, thereby enhancing the effectiveness and reliability of the proposed system in diagnosing and classifying gastrointestinal conditions.

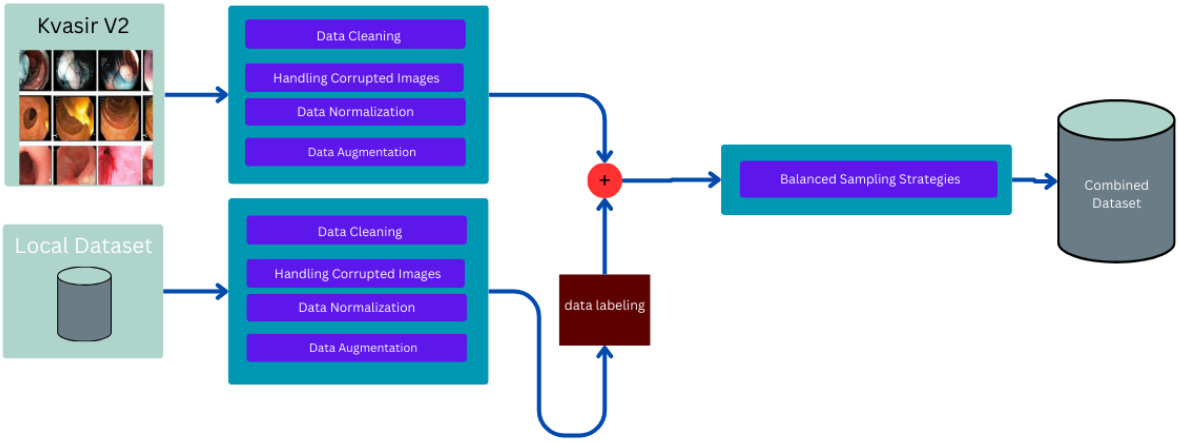


Figure 4.2: Data Preprocessing

4.2.2. Data Cleaning

In the data cleaning process, we aim to address issues related to corrupted and noisy images within the dataset. Here's a description of the steps involved:

Identifying Corrupted Images: The first step is to detect any corrupted images within the dataset. These may include images that are partially or entirely damaged due to various reasons such as file corruption, incomplete downloads, or formatting errors.

Removing Corrupted Images: Once a corrupted image is identified, it is removed from the dataset. This ensures that only intact and usable images remain for further analysis and processing.

Detecting Noisy Images: Noisy images are those that contain unwanted artifacts or disturbances, often caused by factors like sensor imperfections, compression artifacts, or environmental interference.

Denoising Images: For images identified as noisy, a denoising algorithm is applied to clean and enhance the image quality. Denoising techniques aim to reduce or eliminate unwanted noise while preserving important features and details in the image.

By systematically addressing corrupted and noisy images, we ensure that the dataset is clean and of high quality, setting the stage for accurate and reliable analysis and modeling in subsequent stages of the project.

Pseudocode

```
function clean_data(dataset):
    for each image in dataset:
        if image is corrupted:
            remove_corrupted_image(image)
        if image is noisy:
            denoise_image(image)
    return cleaned_dataset
function remove_corrupted_image(image):
    remove image from dataset
function denoise_image(image):
    apply denoising algorithm to remove noise from image
```

4.2.3. Data Normalization

In our preprocessing pipeline, we applied normalization to the images using the transforms.Normalize function. This step adjusted the pixel values to a standard scale, with the mean and standard deviation values set to [0.485, 0.456, 0.406] and [0.229, 0.224, 0.225], respectively. This normalization process helped stabilize and accelerate the training of the model by standardizing the input image data.

Pseudocode

```
# Define normalization parameters
mean = [0.485, 0.456, 0.406]
std = [0.229, 0.224, 0.225]
# Apply normalization transformation to the dataset
def normalize_image(image):
```

```
return Normalize(image, mean=mean, std=std)
```

4.2.4. Data Augmentation

We employed data augmentation techniques, including flipping, rotating, and resizing, to enrich our dataset and enhance the robustness of our machine learning model. These methods introduced variations that improved the model's generalization and performance.

Pseudocode

```
# Define augmentation operations
augmentation_operations = [
    Flip(),         # Flip the image horizontally or vertically
    Rotate(angle),  # Rotate the image by a specified angle
    Resize(width, height)  # Resize the image to a fixed width and height
]
```

4.2.5. Balance Sampling Strategy

In our thesis we employ a balanced sampling strategy to mitigate the challenges posed by class imbalance within our dataset. Class imbalance, where certain gastrointestinal disease classes may be underrepresented compared to others, can adversely affect the performance of machine learning models by biasing them towards the majority class.

To address this issue, we implement a balanced sampling strategy that ensures each gastrointestinal disease class is adequately represented during the training process. This approach involves randomly sampling instances from each class to match the size of the smallest class, effectively creating a balanced distribution of instances across all classes. By doing so, we prevent the model from being disproportionately influenced by the majority class and enable it to learn from all classes equally.

4.3. Model Building

In our thesis on "Enhancing Gastrointestinal Disease Classification using Transfer Learning," we undertake model building using the ResNeSt architecture, comparing its variants with depths. Leveraging transfer learning and fine-tuning techniques, we incorporate both locally

sourced gastrointestinal imaging data from Adera Medical Center and publicly available data from the Kvasir v2 dataset.

Transfer learning allows us to leverage knowledge gained from pre-trained models on large-scale datasets, such as ImageNet, and adapt it to our specific task of gastrointestinal disease classification. Fine-tuning further refines the pre-trained model's parameters to better suit our target domain, ensuring optimal performance on our combined dataset.

By combining data from multiple sources and employing transfer learning and fine-tuning, we aim to develop a robust classification model capable of accurately identifying various gastrointestinal diseases. This approach not only enhances the model's ability to generalize across different datasets but also improves its effectiveness in real-world clinical settings, where data variability and domain-specific nuances are prevalent. Through our comparative analysis of ResNeSt variants and transfer learning strategies, we seek to identify the most effective model configuration for our specific task, ultimately contributing to advancements in gastrointestinal disease diagnosis and treatment.

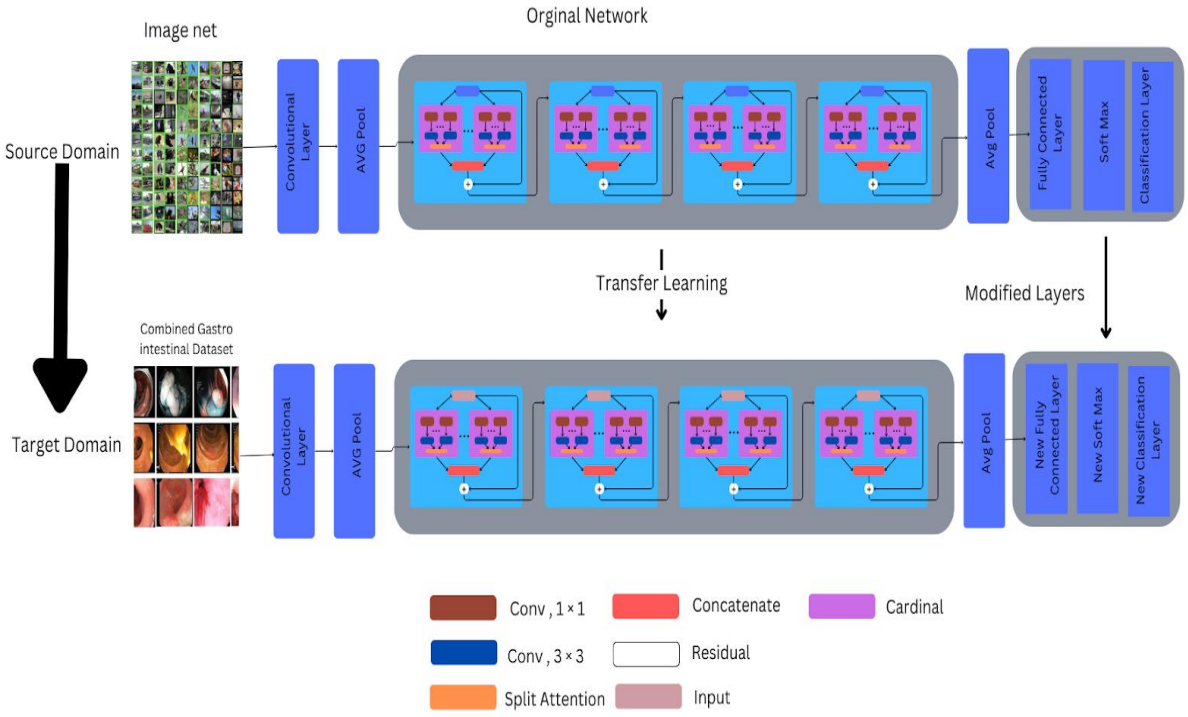


Figure 4.3: Architecture of proposed model

4.4. Model Evaluation

We rigorously evaluate our developed model to assess its performance and efficacy in accurately classifying gastrointestinal diseases. Model evaluation is a critical step in ensuring the reliability and effectiveness of our classification system.

We employ a comprehensive set of evaluation metrics to measure various aspects of our model's performance, including accuracy, precision, recall, F1-score, and confusion matrix analysis. These metrics provide valuable insights into the model's ability to correctly classify different gastrointestinal diseases and its overall performance across multiple classes.

Furthermore, we ensure the robustness and generalization capability of our model through careful data splitting rather than cross-validation. We employ two types of data splits: a 60/40 split and a 70/30 split. In the 60/40 split, the dataset is divided into 60% for training, 30% for validation, and 10% for testing. In the 70/30 split, the dataset is divided into 70% for training, 15% for validation, and 15% for testing. By training and evaluating the model using these different splits, we assess its performance consistency and ability to generalize to new, unseen data.

Through meticulous model evaluation, we aim to ascertain the strengths and weaknesses of our classification system, identify areas for improvement, and ultimately ensure the reliability and accuracy of our model in real-world clinical settings. By adhering to rigorous evaluation standards, we strive to contribute valuable insights to the field of gastrointestinal disease diagnosis and treatment.

CHAPTER FIVE

5. IMPLEMENTATION AND EXPERIMENT

In this chapter, we detail the implementation and experimental evaluation of our proposed system for enhancing gastrointestinal disease classification through transfer learning techniques. The primary objective was to develop a robust classification model capable of accurately identifying various gastrointestinal diseases using medical imaging data. By leveraging transfer learning and incorporating local Ethiopian data, our aim was to improve the diagnosis of gastrointestinal diseases and facilitate early detection, ultimately enhancing patient care and treatment outcomes.

The proposed system is specifically designed to meet the needs of healthcare professionals by offering a reliable diagnostic tool based on insights derived from medical imaging data. Utilizing transfer learning and advanced machine learning algorithms, the system sought to enhance the early detection of gastrointestinal conditions, thereby improving diagnostic accuracy and contributing to better patient care outcomes.

To achieve these objectives, we employed a combination of software tools and packages tailored to our specific requirements. These included machine learning frameworks such as PyTorch, as well as libraries like pandas and matplotlib for data preprocessing and visualization. Additionally, we ensured a conducive working environment that emphasized compatibility and efficiency throughout the implementation process.

Before presenting the actual implementation of the proposed system, we provide a comprehensive overview of the tools, software packages, and working environment employed. This overview establishes the foundation for the subsequent chapters, where we explore the practical aspects of developing and evaluating the classification model for gastrointestinal disease prediction.

5.1. Working Environment

To fulfill the research objectives effectively, the following hardware and software specifications were utilized:

Hardware Specification:

Desktop with high-performance configuration

- **Processor**: Intel(R) Core(TM) i7-10700K CPU @ 3.80GHz
- **Installed RAM:** 8.00 GB (Gigabyte)
- **System**: 64-bit operating system, x64-based processor
- **GPU**: NVIDIA GeForce RTX 3080 (Installed for accelerated computations)
- **Hard Disk**: 1TB (Terabyte)

Software Specification:

- Operating System: Windows 10
- Anaconda Distribution for Python (with Jupyter Notebook)
- NVIDIA CUDA Toolkit (for GPU acceleration)
- Microsoft Office Suite (for documentation and data preparation)
- Enterprise architecture (for diagram creation and visualization)

5.2. Implementation Environment:

To facilitate the realization of the research objectives, Python was chosen as the primary programming language due to its widespread adoption and extensive support within the machine learning and data science communities. Python offers a rich ecosystem of libraries, frameworks, and tools that are essential for various stages of data processing and model development. Leveraging Python's versatility and robustness, the implementation of the research involves the utilization of several key packages and libraries tailored to specific tasks.

Table 5.1: Implementation tools

Tools	***Version***	***Description***
Anaconda	**1.10.0**.	Anaconda is utilized as the primary integrated development environment (IDE) for managing Python packages and dependencies, providing a convenient platform for executing Python scripts and managing project environments.

Jupyter Notebook	7.0.1	Jupyter Notebook is employed for interactive data analysis and documentation, allowing for the creation of executable documents containing code, visualizations, and narrative text.
Python Programming Language	3.11	Python serves as the core language for implementing the research, providing a flexible and intuitive environment for data processing and analysis.
NumPy	1.26.4	NumPy is utilized for numerical computations and array manipulation, offering efficient data structures and mathematical functions essential for scientific computing.
Pandas	2.2	Pandas is employed for data manipulation and analysis, enabling seamless handling of structured data and integration with other Python libraries.
Scikit-learn	1.4.2	Scikit-learn is utilized for machine learning tasks, providing a comprehensive set of tools for data preprocessing, model selection, and evaluation.
Pytorch	2.16.1	It is utilized for deep learning tasks, offering a flexible framework for building and training neural networks.
Matplotlib and Seaborn	3.8.4 and 0.13.2 respectively	Matplotlib and Seaborn are employed for data visualization, allowing for the creation of insightful plots and charts to convey research findings effectively.

5.3. Training Procedure

Data Preparation

Dataset 1: Kvasir V2

- ✓ *Preprocessing***:** Preprocessing steps specific to the Kvasir V2 dataset are applied, including normalization and augmentation for training.
- ✓ *Data Loading:* Data loaders are set up for the training, validation, and test sets.

Dataset 2: Local Data

- ✓ *Preprocessing***:** Preprocessing steps are performed on the local dataset, ensuring consistency with the Kvasir V2 dataset in terms of normalization and resizing.
- ✓ *Data Loading***:** Data loaders are set up for this dataset as well.

Combined Dataset

- ✓ *Data Combination:* The Kvasir V2 and local datasets are merged for final testing. Proper balancing and preprocessing of the combined dataset are ensured.

Model Initialization

- ✓ *Model Setup:* The ResNeSt model is initialized, with pre-trained weights for transfer learning.
- ✓ *Optimizer***:** The Adam optimizer is chosen and configured with an appropriate learning rate and parameters.
- ✓ *Loss Function:* The loss function suitable for the classification task, such as Cross-Entropy Loss, is selected.

Model Training on Kvasir V2 Dataset

- ✓ *Training Procedure:* The model is trained using only the Kvasir V2 dataset.
- ✓ *Validation***:** Performance on the Kvasir V2 validation set is monitored, and early stopping is applied as needed.

- ✓ *Testing***:** The model is evaluated on the Kvasir V2 test set to assess performance.

Test on Local Data Test Set

- ✓ *Testing***:** The model is evaluated on the local data test set to assess performance.

Model Training on Combined Datasets

- ✓ *Data Combination:* The Kvasir V2 and local datasets are merged, ensuring balanced sampling and proper preprocessing.
- ✓ *Training Procedure:* The model is trained on the combined dataset.
- ✓ *Validation***:** The model is validated using the combined dataset's validation set.
- ✓ *Testing***:** The model is evaluated on the combined test set to assess overall performance.

Model Check pointing

- ✓ *Save Best Model:* Checkpoints are saved during each experiment to retain the best-performing model based on validation metrics.

Post-Training

- ✓ *Final Evaluation:* After completing all experiments, the final model is evaluated on each test set and the combined test set.
- ✓ *Result Analysis:* Results are analyzed to compare performance across different datasets and to determine the effectiveness of the model in various scenarios.

5.4. Pre-processing and Data Set Preparation

In the pre-processing implementation phase, we meticulously prepared the dataset to ensure its quality and suitability for subsequent analysis and modeling. This involved several key steps, including data cleaning to address issues such as corrupted images and noise reduction, data normalization, and data augmentation to enrich the dataset with additional examples

5.4.1. Load Data

In this step, the datasets required for training and evaluating the model were loaded. The provided code snippet illustrates how two datasets were loaded from specified directories: the

Kvasir v2 dataset and the Local Ethiopian dataset. Using the `datasets.ImageFolder` method from the `torchvision` library, images were read from their respective folders and organized into a format suitable for further processing. This facilitated efficient access and manipulation of image data for subsequent stages of model development.

```
dataset_dir = 'C:\\Users\\model\\Documents\\Tadiwos Data\\Datasets\\'
full_dataset = datasets.ImageFolder(os.path.join(dataset_dir, 'Kvasir2'))
```

Figure 5.1: Implementation of loading kvasirv2 dataset

```
dataset_dir = 'C:\\Users\\model\\Documents\\Tadiwos Data\\Datasets\\'
full_dataset = datasets.ImageFolder(os.path.join(dataset_dir, 'Local'))
```

Figure 5.2: Implementation of loading kvasirv2 data set

5.4.2. Data Cleaning

In this stage, we addressed issues related to data quality by implementing data cleaning techniques. This involved handling corrupted images and reducing noise to ensure that the dataset was free from artifacts that could affect subsequent analysis. Corrupted images were identified and removed, while noise reduction filters were applied to enhance the clarity of important structures and patterns within the images.

```
def is_corrupted(image_path):
    try:
        img = Image.open(image_path)
        img.verify()  # Verify the image integrity
        img.close()
        return False
    except (IOError, SyntaxError) as e:
        print(f"Corrupted image detected: {image_path}, error: {e}")
        return True
```

Figure 5.3: Implementation of identifying corrupted images

```
# Function to remove corrupted images
def remove_corrupted_images(directory):
    i = 0;
    for root, _, files in os.walk(directory):
        for file in files:
            image_path = os.path.join(root, file)
            if is_corrupted(image_path):
                os.remove(image_path)
                print(f"Removed corrupted image: {image_path}")
            else:
                i= 1
    if(i == 1):
        print(f'No Corrupted Image Found');
```

Figure 5.4: Implementation of corrupted images

```
def denoise_image(image_path):
    img = cv2.imread(image_path)
    if img is not None:
        img_tensor = torch.tensor(img).float().to(device)  # Move image to GPU
        dst_tensor = torch.nn.functional.interpolate(img_tensor.permute(2, 0, 1).unsqueeze(0), scale_factor=0.5, mode='bilinear', align_corners=False)
        denoised_img = dst_tensor.squeeze().permute(1, 2, 0).cpu().numpy().astype(np.uint8)  # Move back to CPU and convert to uint8
        return img, denoised_img
    return None, None
```

Figure 5.5: Implementation of de-noising

5.4.3. Data Split

We carefully divided the dataset into training, validation, and test sets with proportions of 70%, 15%, and 15% in one scenario, and 60%, 30%, and 10% in another. These splits were chosen to balance the need for a sufficient amount of training data while ensuring enough validation and test data to evaluate the model's performance comprehensively. In the 70-15-15 split, the larger training set maximizes the amount of data the model can learn from, which is particularly important when dealing with complex patterns in medical images. The validation and test sets, each representing 15% of the data, are large enough to reliably assess the model's performance without significantly reducing the data available for training.

The 60-30-10 split was used in another scenario to emphasize validation. This approach is particularly useful when fine-tuning hyperparameters and conducting more rigorous validation, allowing us to observe how well the model generalizes on unseen data during

training. The 30% validation set provides ample opportunity to detect overfitting and ensure the model's adaptability to new, unseen data, which is critical for reliable deployment in real-world clinical settings.

We avoided other common splits, such as 80-10-10 or 90-5-5, because they would result in smaller validation and test sets. This could lead to less reliable assessments of the model's performance, particularly when working with smaller datasets. Larger validation and test sets are necessary in our case to ensure that the evaluation metrics are stable and reflective of the model's true capabilities. This systematic approach to data splitting was crucial for ensuring robust and unbiased assessments during model training and evaluation.

```python
# Define the root directory and subdirectories for the split data
root_dir = os.path.join(dataset_dir, 'split_data_local_60_30_10')
output_dirs = {
    'train': os.path.join(root_dir, 'Train'),
    'val': os.path.join(root_dir, 'Val'),
    'test': os.path.join(root_dir, 'Test')
}

# Create the root directory if it doesn't exist
if not os.path.exists(root_dir):
    os.makedirs(root_dir)

# Create the subdirectories for each phase (train, val, test)
for dir_name in output_dirs.values():
    if not os.path.exists(dir_name):
        os.makedirs(dir_name)

# Split and save images class-wise
for class_id, img_paths in class_images.items():
    total_images = len(img_paths)
    train_size = int(0.6 * total_images)
    val_size = int(0.3 * total_images)

    # Shuffle image paths for random splitting
    random.shuffle(img_paths)

    # Split the data
    train_paths = img_paths[:train_size]
    val_paths = img_paths[train_size:train_size + val_size]
    test_paths = img_paths[train_size + val_size:]

    class_dir = full_dataset.classes[class_id]

    # Copy images to their respective directories
    for phase, paths in zip(['train', 'val', 'test'], [train_paths, val_paths, test_paths]):
        class_phase_dir = os.path.join(output_dirs[phase], class_dir)
        if not os.path.exists(class_phase_dir):
            os.makedirs(class_phase_dir)
        for img_path in paths:
            shutil.copy(img_path, os.path.join(class_phase_dir, os.path.basename(img_path)))

print("Dataset splitting and saving completed.")
```

Figure 5.6: Implementation of data split

5.4.4. Data Normalization

We normalized the images by adjusting their pixel values to match a standard distribution. This was done using `transforms.Normalize`, setting the mean to `[0.485, 0.456, 0.406]` and

the standard deviation to `[0.229, 0.224, 0.225]`, aligning the images with the ImageNet dataset.

```
# Define normalization
normalize = transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                 std=[0.229, 0.224, 0.225])
```

Figure 5.7: Implementation of data normalization

5.4.5. Data Augmentation

We defined data transformations for the training, validation, and test sets to ensure consistency and enhance the model's performance. For the training set, we applied several transformations including resizing to 224x224 pixels, random affine transformations (with shear and scale adjustments), and random horizontal flips. These augmentations aimed to improve the model's robustness by introducing variability. For the validation and test sets, we resized the images to 224x224 pixels without additional augmentations to maintain consistency during evaluation.

```
data_transforms = {
    'train':
    transforms.Compose([
        transforms.Resize((224,224)),
        transforms.RandomAffine(0, shear=10, scale=(0.8,1.2)),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        normalize
    ]),
    'validation':
    transforms.Compose([
        transforms.Resize((224,224)),
        transforms.ToTensor(),
        normalize
    ]),
    'test':
    transforms.Compose([
        transforms.Resize((224,224)),
        transforms.ToTensor(),
        normalize
    ]),
}
```

Figure 5.8: Implementation of data augmentation

5.5. Build Classification Model

5.5.1. Model Setup

The ResNeSt model was initialized with the option to load pre-trained weights, leveraging the benefits of transfer learning. This setup enables the model to utilize previously learned features from large-scale datasets, providing a strong starting point for fine-tuning on our specific dataset. The model's architecture was adapted by freezing the earlier layers and fine-tuning only the classifier layer and the preceding layer, optimizing it for our task while reducing the risk of overfitting.

```
# Load ResNeSt model
model = resnest101(pretrained=True)


class ResNeStTransfer(nn.Module):
    def __init__(self, model, num_classes):
        super(ResNeStTransfer, self).__init__()
        self.features = nn.Sequential(*list(model.children())[:-2])
        self.pool = nn.AdaptiveAvgPool2d(1)
        self.dropout = nn.Dropout(p=0.5)
        num_ftrs = model.fc.in_features
        self.fc = nn.Linear(num_ftrs, num_classes)
    def forward(self, x):
        x = self.features(x)
        x = self.pool(x)
        x = torch.flatten(x, 1)
        x = self.dropout(x)
        x = self.fc(x)
        return x

for param in model.parameters():
    param.requires_grad = False

for param in model.layer3.parameters():
    param.requires_grad = True
for param in model.layer4.parameters():
    param.requires_grad = True

num_classes = 6
model = ResNeStTransfer(model, num_classes)
```

Figure 5.9: ResNeSt model setup

5.5.2. Optimizer

The Adam optimizer was selected and configured with an appropriate learning rate to balance convergence speed and stability. Adam's adaptive learning rate properties were particularly beneficial for fine-tuning the ResNeSt model, allowing for efficient optimization of the classifier and preceding layers. The optimizer's parameters, including beta values and weight decay, were carefully chosen to enhance model performance while minimizing the risk of overfitting.

```
optimizer = optim.Adam(filter(lambda p: p.requires_grad, model.parameters()), lr=learning_rate, weight_decay=1e-3)
```

Figure 5.10: Definition of optimizer and regularization

5.5.3. Loss Function

The Cross-Entropy Loss function was selected as it is well-suited for multi-class classification tasks. This loss function measures the performance of the classification model by comparing the predicted probabilities with the actual class labels, penalizing incorrect predictions more heavily. By minimizing the Cross-Entropy Loss, the model's predictions become increasingly accurate over time, making it an ideal choice for our classification objectives. The loss function was defined in the implementation using the `nn.CrossEntropyLoss()` function.

Cross-entropy loss offers several advantages over other loss functions, particularly for classification tasks. It is specifically designed for this purpose, making it more appropriate than loss functions like mean squared error (MSE), which are better suited for regression problems. Cross-entropy loss is sensitive to the confidence of predictions, penalizing incorrect classifications more heavily when the model is confident, which facilitates effective learning and faster convergence during training.

```
criterion = nn.CrossEntropyLoss()
```

Figure 5.11: Loss function definition

5.5.4. Learning Rate Scheduler

A learning rate scheduler was integrated to adjust the learning rate throughout the training process. This helps in fine-tuning the model's learning process, allowing for more precise convergence as the model approaches optimal performance.

```
# Learning Rate Scheduler with ReduceLROnPlateau
exp_lr_scheduler = lr_scheduler.ReduceLROnPlateau(optimizer, mode='min', patience=3, factor=0.2)
```

Figure 5.12: Learning rate scheduler definition

5.5. Model Training and Check Pointing

The model was transferred to the GPU for accelerated training, utilizing CUDA if available. The training process was conducted using the Kvasir V2 dataset, with the model, optimizer, loss function, and learning rate scheduler all integrated into the training loop. Upon completion, the model's best weights, identified through monitoring validation accuracy, were saved as a checkpoint in the file best_model.pth. This allows for easy restoration and further experimentation or deployment with the most effective version of the model.

```
# Train the model
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
model = model.to(device)
model, train_losses, val_losses, train_accuracies, val_accuracies = train_model(
    model, dataloaders, dataset_sizes, criterion, optimizer, exp_lr_scheduler, num_epochs=num_epochs
)

# Save the model checkpoint
torch.save(model.state_dict(), 'best_model.pth')
```

Figure 5.13: Model training and check pointing

5.6. Model Evaluation and Visualization

The model evaluation is conducted using the `evaluate_model` function, which measures the model's performance on both the Kvasirv2 test set and the local data test set. During this process, the model is switched to evaluation mode, and predictions are made without altering

the model weights. The function gathers predictions and true labels to compute essential performance metrics, including accuracy, F1 score, precision, and recall. A confusion matrix is also generated to visualize the model's classification performance across various classes. The evaluation results are displayed, and visual representations such as the confusion matrix and plots of training history are created to offer a detailed understanding of the model's effectiveness on the test data. This thorough evaluation process ensures that the model's performance is comprehensively analyzed and documented.

```
test_accuracy, test_f1, test_precision, test_recall, test_specificity, test_conf_matrix = evaluate_model(
    model, dataloaders['test'], class_names
)
```

Figure 5.14: Evaluation of model on kvasirv2 test set

```
# Test On local data
test_accuracy = evaluate_model(
    model, dataloaders_local['test'], class_names
)
```

Figure 5.15: Evaluation of model on local test set

```
plot_training_results(train_losses, val_losses, train_accuracies, val_accuracies,num_epochs)
```

Figure 5.16: Plotting training cur for training history

5.7. Data Incorporation and Model Retraining and Re Evaluation

The data incorporation process involved merging the local and Kvasirv2 datasets to create a unified dataset that addresses class imbalance issues. By identifying common classes between the two datasets and selecting a balanced number of images from each class based on the minimum number found across both datasets, we ensured that the combined dataset was both representative and balanced. This incorporated data was then subjected to the complete preprocessing pipeline, including normalization, augmentation, and splitting into training, validation, and test sets with specified proportions. Following this, we built and trained our model using this balanced dataset. The model's performance was rigorously evaluated on the test data, with key metrics such as accuracy, precision, recall, and F1 score

computed. Additionally, visualizations such as confusion matrices and training curves were generated to provide a comprehensive understanding of the model's effectiveness. This thorough approach ensured that the model was trained and evaluated on a well-prepared and balanced dataset, leading to reliable and insightful results.

5.8. Lightweight Model Optimization via TorchScript

In our Study, the PyTorch model was converted to TorchScript using `torch.jit.script` to create a lightweight version for efficient deployment. This optimization makes the model more portable and suitable for resource-constrained environments.

```
# Convert your PyTorch model to TorchScript
scripted_model = torch.jit.script(model)

# Save the scripted model
scripted_model.save("model_scripted.pt")
```

Figure 5.17: lightweight model optimization via TorchScript

CHAPTER SIX

6. RESULT AND DISCUSSIONS

6.1. Chapter Overview

In this chapter, the results of training and evaluating different versions of the ResNeSt architecture ResNeSt-50, ResNeSt-101, and ResNeSt-200 are analyzed in comparison with ResNet and VGG16 for the classification of gastrointestinal diseases. The chapter begins by discussing the selection of ResNeSt based on a systematic review of relevant literature. Subsequently, it compares the performance of the various ResNeSt versions with each other and with ResNet and VGG16 using the local dataset. This comparison includes an assessment of classification accuracy and efficiency. The effects of combining data from different sources for training are also explored, with a focus on identifying the best-performing ResNeSt version. The chapter concludes with a detailed discussion on the relative effectiveness of these models and their implications for enhancing disease classification in clinical practice.

6.2. Hyper-parameter Configurations for Model Training

We conducted a series of experiments using various learning rates, starting from 0.01 and decreasing to 0.0001, to explore their impact on model performance. Given the relatively small size of our dataset, we chose to train the model for 30 and 60 epochs, ensuring sufficient training time for convergence without overfitting. Additionally, we experimented with different dropout rates and weight decay values (L2 regularization) to prevent overfitting and enhance the model's generalization capabilities. Various scheduler factors were also applied to adjust the learning rate dynamically during training, helping the model to converge more effectively. Additionally, we conducted experiments with various batch sizes, specifically 24, 32, and 40. These experimental variations allowed us to identify the optimal combination of hyper parameters for the classification task.

We also conducted further experiments by adjusting the number of unfrozen layers. Specifically, we explored scenarios where only the final fully connected layer (FCL) was

unfrozen, as well as cases where the last layers or the last two layers, including the FCL, were unfrozen. This approach allowed us to assess the impact of fine-tuning different depths of the model on its performance.

Table 6.1: Hyper-parameter configurations.

Cases	Epoch	Learning Rate	Drop Out	Weight Decay (L2)	Unfreeze Layer	Scheduler Factor
Case 1.	30	0.01	0.5	1e-3	Last 2 layers, FCL	0.2
Case 2.	30	0.0001	0.5	1e-3	Last Layer, FCL	0.2
Case 3.	30	0.01	0.25	1e- 3	Last 2 layers,, FCL	0.5
Case 4.	30	0.00001	0.5	1e-4	Last 2 layers,, FCL	0.2
Case 5.	60	0.01	0.5	1e-3	Last 2 layers,, FCL	0.2
Case 6.	60	0.01	0.25	1e-3	Last layer, FCL	0.2
Case 7.	60	0.00001	0.5	1e-4	Last 2 layers,, FCL	0.5

6.3. Discussion of ResNeSt Results

Throughout our training process, we employed two key data splits: **40%-60%** (30% validation, 10% testing, 60% training) and **30%-70%** (15% validation, 15% testing, 70% training) on the KvasirV2 dataset. These splits were strategically chosen to assess how different allocations of data for validation, testing, and training influence the model's overall performance. The **40%-60%** split was designed to provide a larger validation set, thereby enhancing the model's generalization capabilities through a broader range of examples for fine-tuning. In contrast, the **30%-70%** split aimed to maximize the amount of data used for training, ensuring robust learning and model development.

6.3.1. Discussion of ResNeSt50 Results

Among the evaluated cases, Case 5 (30%-70%) emerged as the first top-performing model, achieving a validation accuracy of 93.33% and a test accuracy of 93.22%. It also maintained consistently high metrics across F1 score, precision, and recall, all within the range of 93.32% to 93.39%. Additionally, this configuration achieved the highest local data accuracy on the test set at 88.75%, underscoring its strong generalization capabilities across different datasets. Following closely, Case 1 (40%-60%) ranked second, with a validation accuracy of 93.31%, a test accuracy of 93.17%, and a local data accuracy of 88.20%, along with similarly strong F1 score, precision, and recall. These results highlight the effectiveness of both the 30%-70% and 40%-60% data splits in fine-tuning and evaluating the ResNest-50 model, resulting in optimal performance across multiple metrics.

Table 6.2: Model evaluation of ResNeSt50

Model Evaluation									
Cases	**Data Split**		**KvasirV2 Accuracy**			**F1 Score (%)**	**Precision (%)**	**Recall (%)**	**Local Data Accuracy**
	Val	**Train**	**Train**	**Val**	**Test**				**Test Set**
Case1	30%	70%	94.79%	93.22%	93.11%	93.05	93.62	93.11	88.17%
	40%	***60%***	***95.14%***	***93.31%***	***93.17%***	***93.17***	***93.28***	***93.17***	***88.20%***
Case2	30%	70%	98.48%	91.56%	91.11%	91.08	91.23	91.11	86.20%
	40%	60%	98.44%	91.89%	91.33%	91.29	91.42	91.33	86.21%
Case3	30%	70%	93.83%	91.78%	91.11%	91.03	91.21	91.11	86.14%
	40%	60%	94.14%	92.56%	90.57%	90.37	90.86	90.50	87.21%
Case4	30%	70%	96.17%	91.00%	91.13%	91.09	91.11	91.11	86.15%
	40%	60%	97.33%	92.28%	91.00%	90.93	91.01	91.00	87.23%
Case5	**30%**	**70%**	**94.76%**	**93.33%**	**93.22%**	**93.32**	**93.39**	**93.33**	**88.75%**
	40%	60%	94.58%	92.78%	92.67%	92.61	92.93	92.67	87.15%
Case6	30%	70%	96.02%	92.44%	92.44%	92.38	92.88	92.44	87.11%

	40%	60%	94.92%	93.00%	93.17%	93.12	93.46	93.17	86.11%
Case7	30%	70%	98.60%	89.78%	90.00%	89.96	90.07	90.00	82.10%
	40%	60%	93.06%	90.83%	88.17%	87.93	88.29	88.17	81.80%

Model Loss of ResNeSt50

The model loss evaluation provides valuable insights into the training and validation losses across different cases and data splits. Case 5, with a 30%-70% data split, emerged as the first most effective configuration, achieving a training loss of 0.1355 and a validation loss of 0.1901 These loss values correspond well with the model's strong performance metrics, indicating efficient learning and excellent generalization to unseen data. Case 1(40%-60%), which ranked second in validation accuracy, recorded a training loss of 0.1354 and a validation loss of 0.2136, further demonstrating its robustness. These findings reinforce Case 5 and Case 1 as the first and second top-performing configurations in this study.

Table 6.3: Model loss of ResNeSt50

Model Loss				
	Data Split		**Training**	**Validation**
Case 1	30%	70%	0.1352	0.2310
	40%	***60%***	***0.1354***	***0.2136***
Case 2	30%	70%	0.0507	0.2829
	40%	60%	0.0318	0.2832
Case 3	30%	70%	0.1499	0.2484
	40%	60%	0.1585	0.2132
Case 4	30%	70%	0.1037	0.2276
	40%	60%	0.0802	0.2343
Case 5	**30%**	**70%**	**0.1355**	**0.1901**
	40%	60%	0.1372	0.2020
Case 6	30%	70%	0.1037	0.2276
	40%	60%	0.1269	0.2058
Case 7	30%	70%	0.1978	0.2707
	40%	60%	0.1730	0.2531

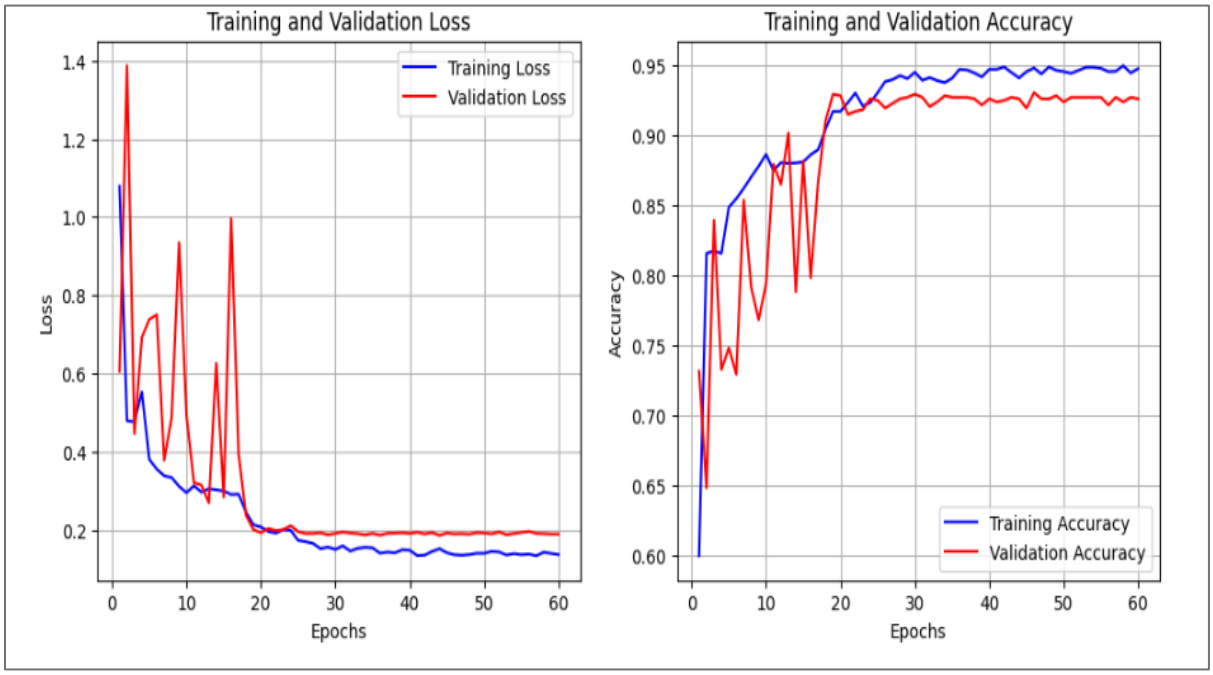


Figure 6.1: Learning Curve for top performer of ResNeSt50 (case 5-30% 70% Data Split)

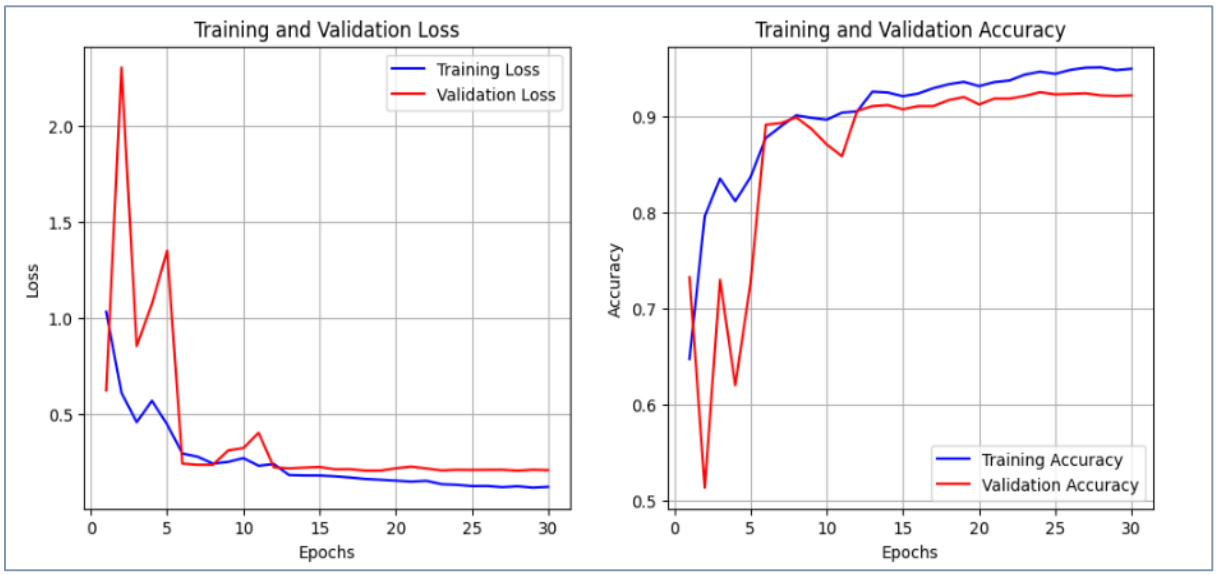


Figure 6.2: Learning curve of the second top perform of resnest50 (Case 1 - 40% 60% data Split)

6.3.2. Discussion of ResNest101 Results

In the model evaluation, **Case 3 (30%-70%)** stands out as the top-performing model, achieving a validation accuracy of **93.22%** and a test accuracy of **93.11%**. The model maintains high performance across other metrics as well, with F1 score, precision, and recall

all falling within the **93.07% to 93.33%** range. Additionally, it achieved a strong local data accuracy on the test set at **87.17%**, reflecting its robust generalization capabilities. **Case 5 (30%-70%)** is the next best performer, with a validation accuracy of **93.00%**, a test accuracy of **92.78%**, and a local data accuracy of **87.07%**. This case also shows strong results in F1 score, precision, and recall, all within the **92.76% to 92.89%** range. The effectiveness of these configurations highlights the success of the **30%-70%** data split in optimizing the ResNeSt101 model for high performance across diverse metrics.

Table 6.4: Model evaluation of ResNeSt101

Model Evaluation									
Cases	**Data Split**		**KvasirV2 Accuracy**			**F1 Score (%)**	**Precision (%)**	**Recall (%)**	**Local Data Accuracy**
	Val	**Train**	**Train**	**Val**	**Test**				**Test Set**
Case1	30%	70%	92.76%	93.00%	90.89%	90.85	90.93	90.89	84.83%
	40%	60%	95.14%	93.00%	91.83%	91.83	91.94	91.83	**85.11**%
Case2	30%	70%	95.98%	91.67%	91.33%	91.31	91.56	91.33	85.07%
	40%	60%	97.78%	92.61%	92.67%	92.63	92.63	92.67	86.11%
Case3	**30%**	**70%**	**94.02**%	**93.22**%	**93.11**%	**93.07**	**93.33**	**93.11**	**87.17**%
	40%	60%	95.00%	91.11%	91.17%	91.17	91.20	91.17	84.83%
Case4	30%	70%	99.57%	93.00%	92.68%	92.6	92.83	91.78	87.07%
	40%	60%	97.94%	92.89%	91.67%	91.60	91.90	91.67	84.90%
Case5	***30%***	***70%***	***94.76***%	***93.00***%	***92.78***%	***92.76***	***92.89***	***92.78***	***87.07***%
	40%	60%	94.08%	93.56%	92.50%	92.49	92.58	92.50	84.88%
Case 6	30%	70%	93.71%	92.44%	92.44%	92.38	92.88	92.44	84.60%
	40%	60%	95.92%	93.00%	91.83%	91.80	92.06	91.83	86.56%
Case 7	30%	70%	92.60%	89.78%	90.00%	89.96	90.07	90.00	86.22%
	40%	60%	92.03%	90.83%	89.67%	89.54	90.01	89.67	84.96%

Model Loss of ResNeSt101

The model loss evaluation underscores **Case 3(30%-70%)** as the top-performing configuration, boasting the training loss of **0.1524** and a validation loss of **0.1976**. These low loss values align with the model's exceptional accuracy metrics, indicating highly efficient learning and strong generalization to unseen data. **Case 5(30%-70%)**, which also performed well, recorded a training loss of **0.1366** and a validation loss of **0.1901**. This further confirms its robustness and ability to avoid overfitting. Overall, the results highlight Case 3 as the leading configuration in terms of minimizing loss, followed closely by Case 5.

Table 6.5: Model loss of ResNeSt101

Model Loss				
	Data Split		**Training**	**Validation**
Case 1	30%	70%	0.1857	0.2020
	40%	60%	0.1327	0.1891
Case 2	30%	70%	0.1053	0.2174
	40%	60%	0.0589	0.2350
Case 3	**30%**	**70%**	**0.1524**	**0.1976**
	40%	60%	0.1261	0.2032
Case 4	30%	70%	0.0128	0.2570
	40%	60%	0.0535	0.2339
Case 5	***30%***	***70%***	***0.1366***	***0.1901***
	40%	60%	0.1482	0.1858
Case 6	30%	70%	0.1586	0.2181
	40%	60%	0.1112	0.2017
Case 7	30%	70%	0.1978	0.2707
	40%	60%	0.2134	0.2490

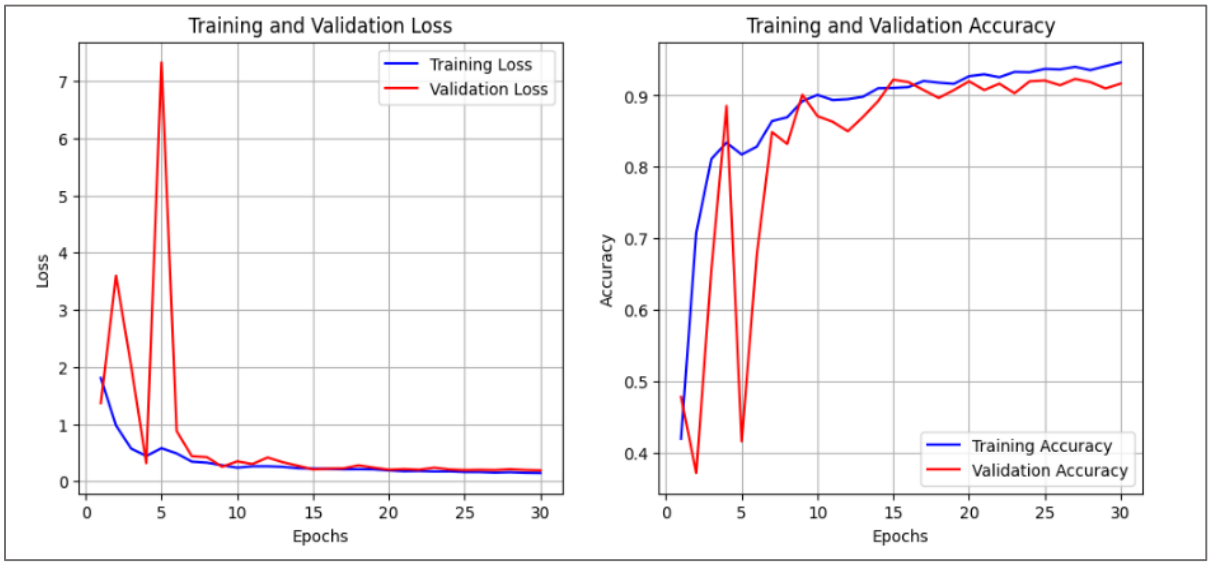


Figure 6.3: Learning Curve for top performer of ResNeSt101 (case 3-30% 70% Data Split)

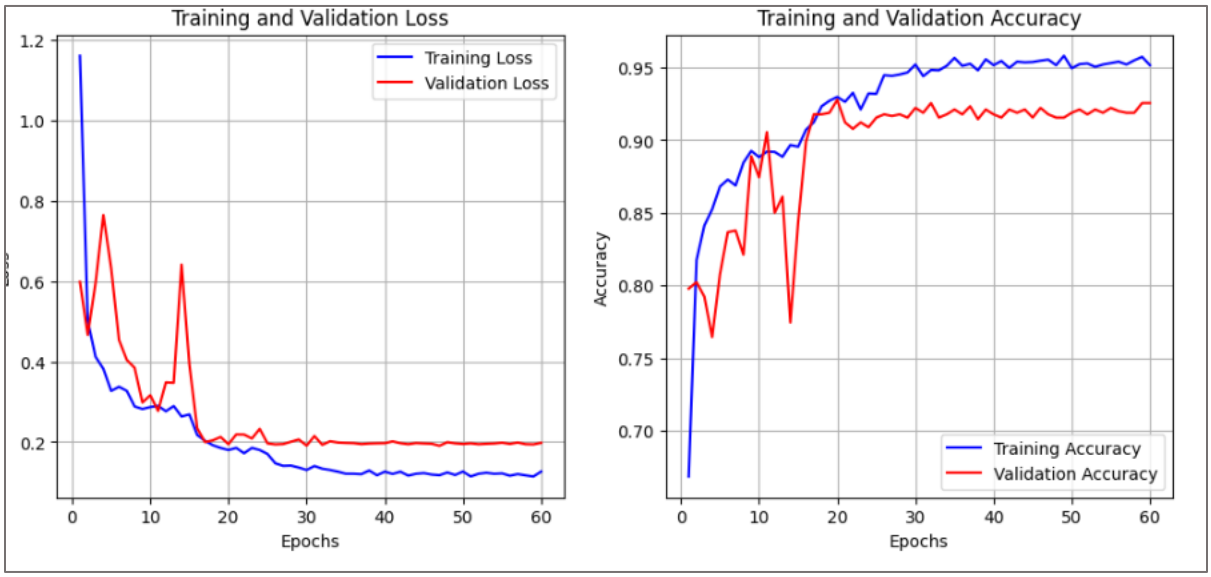


Figure 6.4: Learning Curve for second top performer of ResNeSt50 (case 5-30% 70% Data Split)

6.3.3. Discussion of ResNeSt200 Results

Among the evaluated cases for the ResNest-200 model, Case 3(40%-60%) emerged as the top-performing model. It achieved a validation accuracy of 92.89% and a test accuracy of 92.83% on the KvasirV2 dataset. This configuration consistently displayed high performance metrics, with an F1 score of 92.82%, precision of 92.86%, and recall of 92.83%. Additionally, it recorded the highest local data accuracy on the test set at 87.89%,

underscoring its strong generalization capabilities across different datasets. Following closely, Case 6(40%-60%) ranked second, with a validation accuracy of 92.72%, a test accuracy of 92.67%, and a local data accuracy of 87.67%. It also maintained robust metrics across F1 score, precision, and recall, all within the range of 92.66% to 92.77%. These results highlight the effectiveness of the 40%-60% data split in fine-tuning and evaluating the ResNest-200 model, resulting in optimal performance across multiple metrics.

Table 6.6: Model Evaluation of ResNeSt200

Model Evaluation									
Cases	**Data Split**		**KvasirV2 Accuracy**			**F1 Score (%)**	**Precision (%)**	**Recall (%)**	**Local Data Accuracy**
	Val	**Train**	**Train**	**Val**	**Test**				**Test Set**
Case1	30%	70%	95.58%	92.33%	**92.22%**	**92.20**	**92.39**	**92.22**	87.17%
	40%	60%	95.50%	92.39%	**92.17%**	**92.13**	**92.31**	**92.17**	87.05%
Case2	30%	70%	97.43%	91.89%	91.56%	91.53	91.62	91.56	86.17%
	40%	60%	96.83%	91.28%	91.17%	91.15	91.35	91.17	86.11%
Case3	30%	70%	95.36%	92.11%	92.56%	92.51	92.78	92.56	87.15%
	40%	**60%**	**94.67%**	**92.89%**	**92.83%**	**92.82**	**92.86**	**92.83**	87.89%
Case4	30%	70%	97.55%	91.89%	91.33%	91.29	91.43	91.33	86.39%
	40%	60%	95.50%	91.27%	91.17%	91.17	91.49	91.17	86.62%
Case5	30%	70%	94.38%	92.56%	92.22%	92.19	92.26	92.22	86.35%
	40%	60%	92.78%	92.72%	92.50%	92.47	92.58	92.50	87.17%
Case6	30%	70%	95.05%	92.89%	91.89%	91.89	91.90	91.89	86.56%
	40%	***60%***	***97.39%***	***92.72%***	***92.67%***	***92.66***	***92.77***	***92.67***	***87.67%***
Case7	30%	70%	93.05%	89.56%	89.50%	89.47.	89.78	89.79	84.25%
	40%	60%	89.97%	89.72%	89.17%	89.05	89.34	89.17	84.11%

Model Loss Evaluation for ResNeSt200

The model loss evaluation provides valuable insights into the training and validation losses across different cases and data splits. Case 3(40%-60%) emerged as the first most effective configuration, achieving a training loss of 0.1454 and a validation loss of 0.2175. These loss values align well with the model's strong performance metrics, indicating efficient learning

and excellent generalization to unseen data. Case 6(40%-60%), which ranked second in test accuracy, recorded a training loss of 0.0814 and a validation loss of 0.2205. Despite the slightly higher training loss, the validation loss remains controlled, further supporting its robustness.These findings reinforce Case 3 and Case 6 as the first and second top-performing configurations in this study of the ResNest-200 model.

Table 6.7: Model loss of ResNeSt200

Model Loss				
	Data Split		**Training**	**Validation**
Case 1	30%	70%	0.1185	0.2165
	40%	60%	0.1170	0.2028
Case 2	30%	70%	0.0703	0.2816
	40%	60%	0.0822	0.2511
Case 3	30%	70%	0.1309	0.2179
	40%	**60%**	**0.1454**	**0.2175**
Case 4	30%	70%	0.0690	0.2509
	40%	60%	0.1173	0.2426
Case 5	30%	70%	0.1514	0.2071
	40%	60%	0.1835	0.2119
Case 6	30%	70%	0.1242	0.2117
	40%	***60%***	***0.0814***	***0.2205***
Case 7	30%	70%	0.1871	0.2595
	40%	60%	0.2735	0.2686

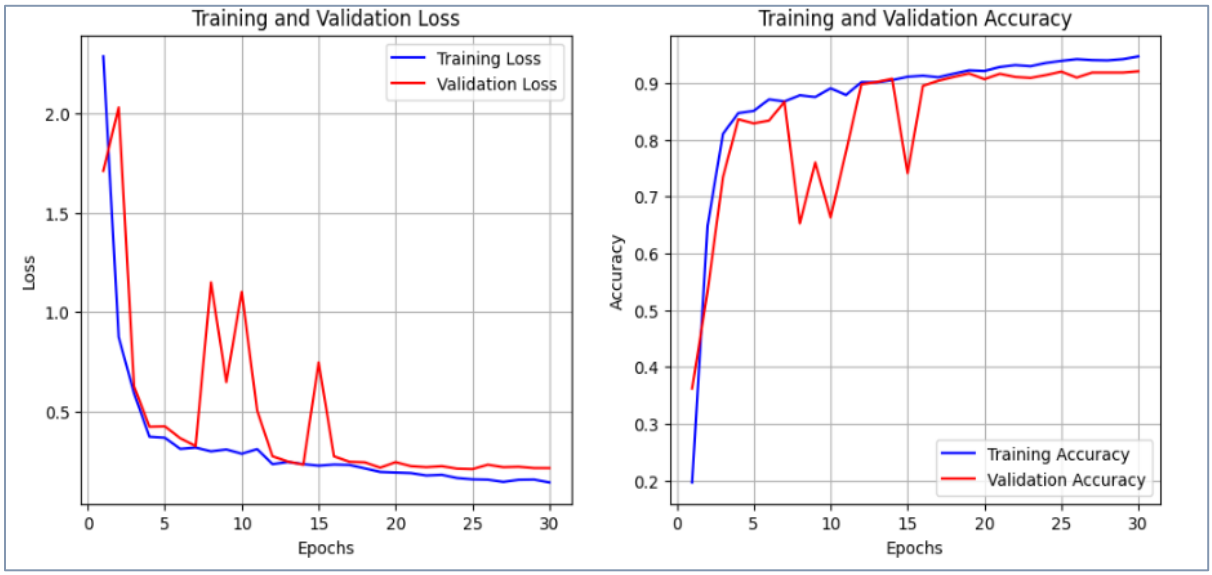


Figure 6.5: Learning Curve for top performer of ResNeSt101 (case 3-40% 60% Data Split):

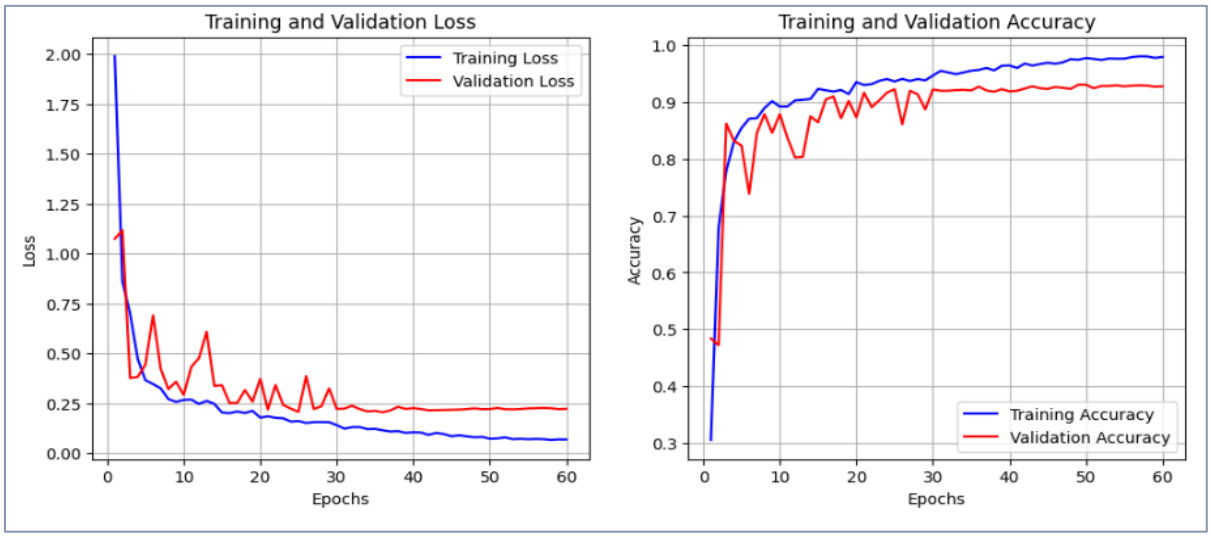


Figure 6.6: Learning Curve for the second top performer of ResNeSt200 (case 6-40% 60% Data Split)

6.3.4. Discussion of ResNeSt50 on Local Dataset

Among the evaluated cases, Case 5 (30%-70%) stood out as the top-performing model, achieving a validation accuracy of 93.33% and a test accuracy of 93.22%. It also demonstrated consistently strong metrics, with F1 score, precision, and recall all ranging between 93.32% and 93.39%. Moreover, this configuration recorded the highest accuracy on

local data in the test set, reaching 88.75%, highlighting its robust generalization across different datasets.

Table 6.8: Model evaluation of ResNeSt50 on local data

Model Evaluation								
Cases	**Data Split**		**KvasirV2 Accuracy**			**F1 Score (%)**	**Precision (%)**	**Recall (%)**
	Val	**Train**	**Train**	**Val**	**Test**			
Case1	30%	70%	95.98%	91.67%	91.33%	91.31	91.56	91.33
	40%	60%	97.78%	92.61%	92.67%	92.63	92.63	92.67
Case2	30%	70%	95.36%	92.11%	92.56%	92.51	92.78	92.56
	40%	**60%**	**94.67%**	**92.89%**	**92.83%**	**92.82**	**92.86**	**92.83**
Case3	30%	70%	93.83%	91.78%	91.11%	91.13	91.11	91.21
	40%	60%	94.14%	92.56%	90.67%	90.47	90.66	90.45
Case4	30%	70%	97.27%	91.15%	91.22%	91.19	91.22	91.00
	40%	60%	98.44%	91.38%	91.12%	91.12	91.33	91.78
Case5	**30%**	**70%**	**95.74%**	**93.15%**	**93.11%**	**93.12**	**93.33**	**93.23**
	40%	60%	94.58%	92.78%	92.67%	92.61	92.93	92.67
Case6	30%	70%	93.86%	91.56%	90.89%	90.86	90.90	9089
	40%	60%	95.67%	91.44%	90.67%	90.65	90.94	90.67

Model Loss of ResNeSt50

The model loss evaluation provides valuable insights into the training and validation losses across different cases and data splits. Case 5, with a 40%-60% data split, emerged as the first most effective configuration, achieving a training loss of 0.1355 and a validation loss of 0.1901 These loss values indicating efficient learning and excellent generalization to unseen data.

Table 6.9: Model loss of ResNeSt50 on local data

Model Loss				
	Data Split		Training	Validation
Case 1	30%	70%	0.1053	0.2174
	40%	***60%***	0.0589	0.2350
Case 2	30%	70%	0.1319	0.2119
	40%	60%	**0.1454**	**0.2175**
Case 3	30%	70%	0.1401	0.2486
	40%	60%	0.1585	0.2132
Case 4	30%	70%	0.1545	0.2268
	40%	60%	0.0876	0.2317
Case 5	**30%**	**70%**	**0.1365**	**0.2001**
	40%	60%	0.1272	0.2020
Case 6	30%	70%	01728	0.2217
	40%	60%	0.1322	0.2122

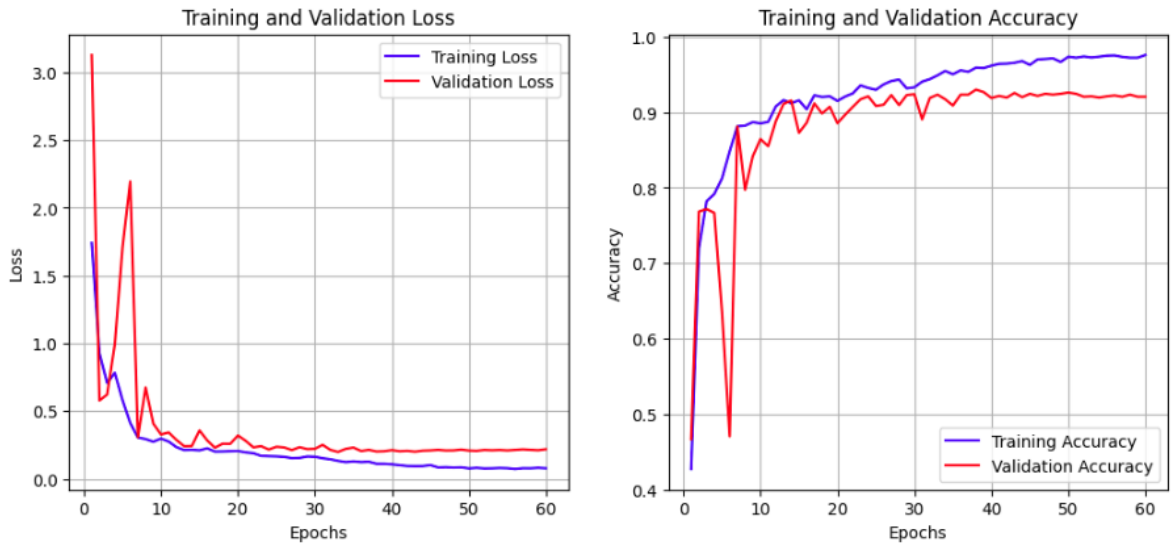


Figure 6.7: Learning Curve for top performer of ResNeSt50 On local data (case 5-30% 70% Data Split)

6.4. Comparison of ResNeSt with other Deep Learning Architecture

In this section, we present a comparative analysis of the ResNeSt architecture against other prominent deep learning models. The evaluation focuses on various performance metrics, including accuracy, F1 score, precision, recall, and model loss across different datasets and data splits. By systematically comparing ResNeSt with architectures such as ResNet50 and, and VGG16, we aim to highlight the unique advantages of ResNeSt in terms of efficiency, generalization ability, and overall model robustness. This comparison provides valuable insights into the suitability of ResNeSt for complex tasks in the field of deep learning.

6.4.1. Discussion of ResNet50 Results

In the model evaluation, Case 5(40%-60%) stands out as the top performer, achieving the highest validation accuracy of 91.97% and a test accuracy of 91.83%. This configuration also excels in other metrics, with an F1 score of 91.81%, precision of 91.17%, and recall of 83.17%. Additionally, Case 5 achieves a strong local data accuracy of 86.89% on the test set. These results confirm that the 40%-60% data split is particularly effective for fine-tuning the ResNet50 model, leading to optimal performance across multiple evaluation criteria.

Table 6.10 Model Evaluation of ResNet50

Model Evaluation									
Cases	**Data Split**		**KvasirV2 Accuracy**			**F1 Score (%)**	**Precision (%)**	**Recall (%)**	**Local Data Accuracy**
	Val	**Train**	**Train**	**Val**	**Test**				**Test Set**
Case1	30%	70%	90.30%	90.22%	88.67%	86.63	89.06	88.67	85.06%
	40%	60%	90.78%	90.78%	90.83%	90.81	91.27	90.83	85.17%
Case2	30%	70%	91.64%	91.19%	91.00%	91.19	90.20	91.10	86.25%
	40%	60%	99.03%	91.06%	91.00%	90.99	91.20	91.00	86.11%
Case3	30%	70%	90.81%	91.33%	90.44%	90.36	91.15	90.44	85.83%
	40%	60%	90.03%	89.56%	88.00%	87.82	89.00	88.00	82.66%
Case4	30%	70%	89.60%	91.44%	90.56%	90.41	90.68	90.56	85.00%
	40%	60%	99.72%	91.50%	91.67%	91.66	91.79	91.67	85.22%
Case5	30%	70%	91.83%	91.22%	91.11%	91.11	91.18	91.11	84.89%

	40%	***60%***	***92.19%***	***91.97%***	***91.83%***	***91.81***	***91.17***	***91.83***	**86.89%**
Case6	30%	70%	93.52%	91.44%	91.22%	91.20	91.35	91.22	84.44%
	40%	60%	95.14%	91.56%	91.67%	90.66	91.78	90.67	86.15%
Case7	30%	70%	93.86%	91.56%	90.89%	90.86	90.90	9089	86.00%
	40%	60%	95.67%	91.44%	90.67%	90.65	90.94	90.67	88.84%

Model Loss of ResNet50

The model loss analysis identifies **Case 5(40%-60%)** as the top-performing configuration, with a training loss of **0.2065** and a validation loss of **0.2263**. These low loss values reflect the model's efficient learning and strong generalization capabilities. The consistent performance across both training and validation losses underscores the effectiveness of this configuration, making it the best choice for achieving balanced and reliable results.

Table 6.11: Model Loss of ResNet50

Model Loss				
	Data Split		**Training**	**Validation**
Case 1	30%	70%	0.2194	0.2473
	40%	60%	0.2414	0.2506
Case 2	30%	70%	0.1623	0.2164
	40%	60%	0.0362	0.2590
Case 3	30%	70%	0.2230	0.2329
	40%	60%	0.2329	0.2545
Case 4	30%	70%	0.2075	0.2460
	40%	60%	0.0142	0.2492
Case 5	30%	70%	0.2113	0.2217
	40%	**60%**	**0.2065**	**0.2263**
Case 6	30%	70%	0.1680	0.2377
	40%	60%	0.1299	0.2367
Case 7	30%	70%	01738	0.2277
	40%	60%	0.1339	0.2111

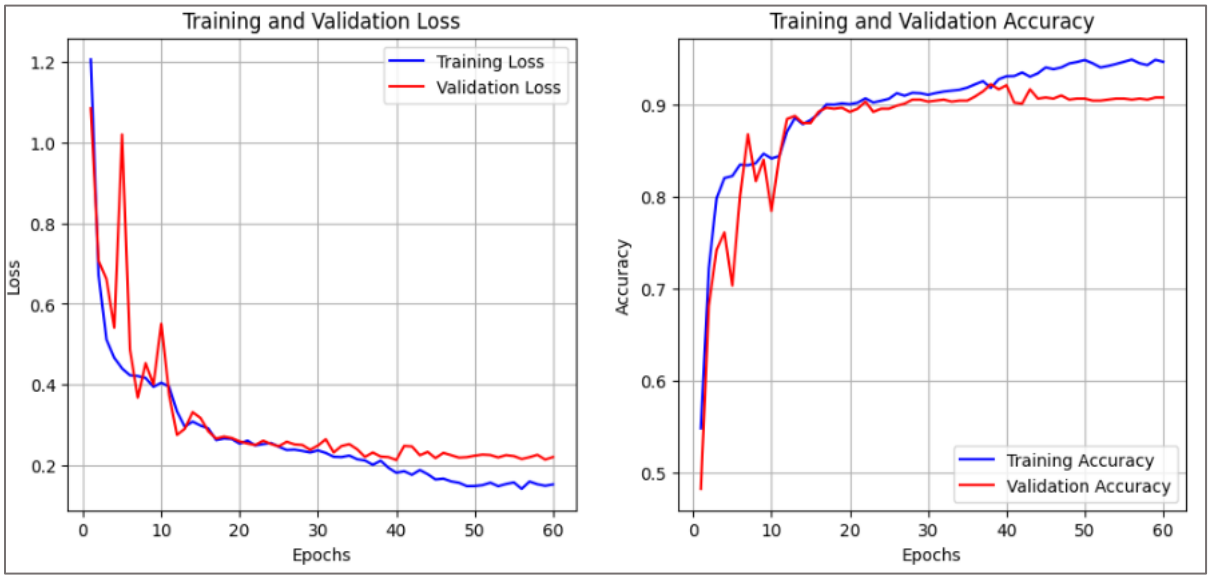


Figure 6.8: Learning Curve for the top performer of ResNet50 (case 5-40% 60% Data Split)

6.4.2. Discussion of VGG16 Results

The model evaluation for VGG-16 was conducted using two data splits: **30%-70%** (15% validation, 15% testing, 70% training) and **40%-60%** (30% validation, 10% testing, 60% training) on the KvasirV2 dataset. These splits were designed to explore how different allocations of data for training, validation, and testing affect the model's performance. **Case 6** with a **40%-60%** split emerged as the top-performing model, achieving a validation accuracy of **92.83%** and a test accuracy of **92.78%**. This case also showed consistently high F1 score (**92.77%**), precision (**92.75%**), and recall (**92.77%**), indicating strong model performance across all metrics. Additionally, **Case 6** achieved the highest local data accuracy of **87.11%** on the test set, further demonstrating its effectiveness.

Table 6.12: Model Evaluation of VGG16

Model Evaluation									
Cases	**Data Split**		**KvasirV2 Accuracy**			**F1 Score (%)**	**Precision (%)**	**Recall (%)**	**Local Data Accuracy**
	Val	**Train**	**Train**	**Val**	**Test**				**Test Set**
Case 1	30%	70%	96.67%	92.56%	92.44%	92.43	92.45	92.44	86.74%
	40%	60%	97.64%	92.67%	92.11%	92.06	92.12	92.11	86.02%
Case 2	30%	70%	90.79%	91.89%	91.33%	91.21	92.14	91.33	85.64%
	40%	60%	91.39%	90.89%	90.17%	90.04	91.09	90.17	85.89%
Case 4	30%	70%	92.48%	92.00%	91.78%	91.74	91.84	91.78	86.11%
	40%	60%	92.11%	91.06%	91.67%	91.49	92.92	91.67	85.45%
Case 5	30%	70%	93.90%	91.89%	92.44%	92.42	92.60	92.44	86.89%
	40%	60%	92.93%	91.56%	92.44%	92.41	92.64	92.44	85.48%
Case6	30%	70%	94.88%	91.67%	92.56%	92.52	92.53	92.56	87.02%
	40%	**60%**	**94.76%**	**92.83%**	**92.78%**	**92.75**	**92.77**	**92.78**	**87.11%**
Case7	30%	70%	92.02%	91.44%	92.50%	92.75	92.83	92.78	87.02%
	40%	60%	91.79%	92.56%	92.44%	92.41	92.56	92.44	86.89%

Model Loss Description

The model loss analysis provides insights into the training and validation losses for each configuration. **Case 6** with a **40%-60%** split recorded a relatively low training loss of **0.1380** and a validation loss of **0.2125**. These values indicate that the model learned effectively and generalized well to unseen data, which is reflected in its high accuracy and other performance metrics. This strong performance across various evaluation metrics reinforces **Case 6** as the optimal configuration for the VGG-16 model in this study.

Table 6.13: Model loss of VGG16

Model Loss				
	Data Split		Training	Validation
Case 1	30%	70%	0.0939	0.2598
	40%	60%	0.0666	0.2807
Case 2	30%	70%	0.2274	0.2499
	40%	60%	0.2176	0.2672
Case 3	30%	70%	0.1896	0.2280
	40%	60%	0.1948	0.2291
Case 4	30%	70%	0.1530	0.2293
	40%	60%	0.1714	0.2155
Case 5	30%	70%	0.1506	0.2112
	40%	**60%**	**0.1380**	**0.2125**
Case 6	30%	70%	0.2049	0.2170
	40%	60%	0.2114	0.2278

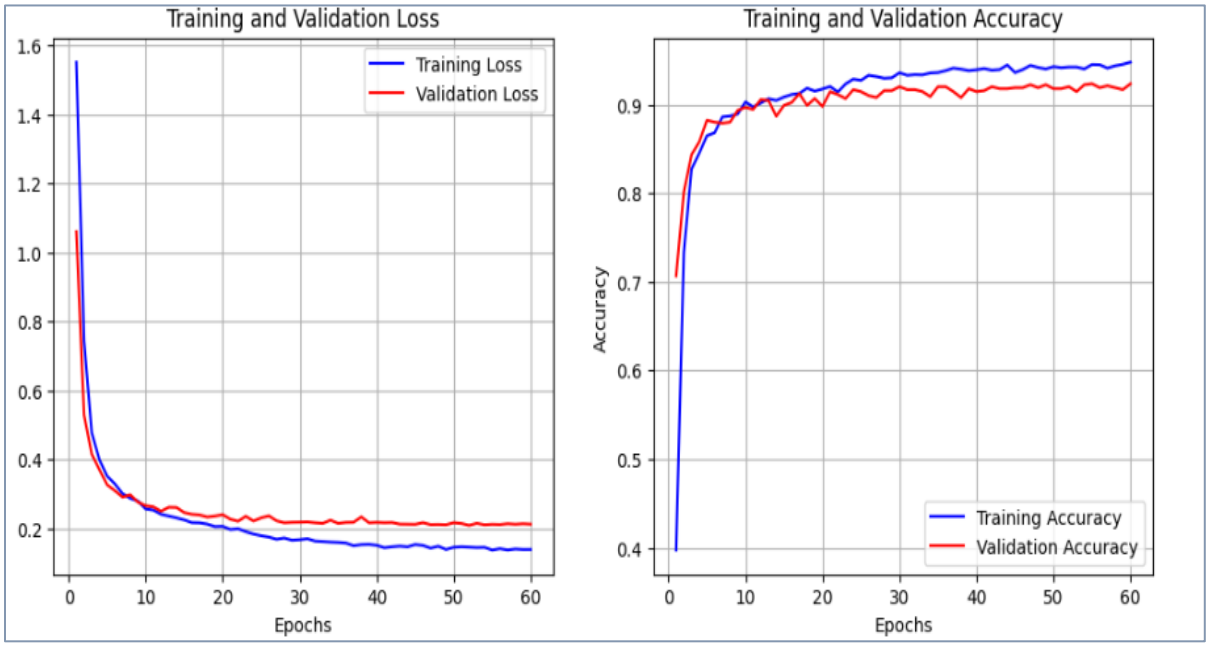


Figure 6.9: Learning Curve for the top performer of VGG16 (case 6-40% 60% Data Split)

The comparative analysis of various deep learning architectures reveals that ResNeSt50 consistently outperforms other models, including ResNeSt101, ResNeSt200, ResNet50, and VGG16, particularly in terms of generalizability and accuracy. ResNeSt50's superior performance can be attributed to its balanced architecture, which effectively captures multi-scale features while maintaining computational efficiency. Unlike ResNeSt101 and ResNeSt200, which offer deeper layers and more complex structures, ResNeSt50 strikes an optimal balance between depth and computational cost, enabling it to generalize better to unseen data. This is especially evident in scenarios where model overfitting can be a concern, as the simpler yet powerful architecture of ResNeSt50 avoids this pitfall, leading to more robust performance across diverse datasets.

Moreover, when compared to ResNet50 and VGG16, ResNeSt50 demonstrates a clear advantage in handling complex image recognition tasks. While ResNet50 and VGG16 are well-established models known for their reliability, they fall short in comparison to ResNeSt50's ability to adapt to intricate patterns and variations in the data. The architectural innovations in ResNeSt50, such as split-attention blocks, allow for more efficient feature recalibration, resulting in improved accuracy across multiple benchmarks.

6.5. Model Training Result of Incorporated Data

Through extensive experimentation on the Kvasir V2 dataset and subsequent testing on our local dataset, it was consistently observed that ResNeSt50 outperformed other deep learning models in terms of accuracy and generalization. To further enhance the accuracy of our local dataset, we incorporated Kvasir V2 data with our local dataset, leveraging the strengths of ResNeSt50 as the most effective architecture identified in our experiments. By training on this combined dataset, we successfully improved the accuracy of our local dataset's test set while maintaining the high level of performance achieved previously. This approach demonstrates the efficacy of data incorporation in boosting model accuracy, particularly when using a well-optimized architecture like ResNeSt50.

Discussion of **ResNeSt50 Result on Incorporated Dataset**

Among the cases evaluated using the incorporated data, Case 5 (30%-70%) emerged as the top performer, achieving a validation accuracy of 93.33% and a test accuracy of 93.22%. This model consistently delivered high performance across key metrics, with F1 score, precision, and recall all ranging between 93.32% and 93.39%. Additionally, it demonstrated exceptional generalization capabilities, significantly improving the local test set accuracy from 88.75% to 93.17%.

Table 6.14: Model evaluation of ResNeSt50 on incorporated dataset

Model Evaluation									
Cases	**Data Split**		**KvasirV2 Accuracy**			**F1 Score (%)**	**Precision (%)**	**Recall (%)**	**Local Data Accuracy**
	Val	**Train**	**Training**	**Val**	**Test**				**Test Set**
Case1	30%	70%	94.79%	93.11%	93.00%	93.15	93.52	93.22	92.89%
	40%	60%	95.14%	93.22%	93.17%	93.11	93.17	93.11	93.10%
Case2	30%	70%	98.48%	91.14%	90.89%	91.00	91.11	90.89	91.20%
	40%	60%	98.44%	91.89%	91.11%	91.12	91.33	91.22	91.26%
Case3	30%	70%	93.80%	91.68%	90.89%	91.00	91.24	91.16	91.16%
	40%	60%	96.89%	92.55%	90.55%	90.67	90.74	90.68	90.21%
Case4	30%	70%	97.27%	91.15%	91.22%	91.19	91.22	91.00	90.22%
	40%	60%	98.44%	91.38%	91.12%	91.12	91.33	91.78	91.36%
Case5	**30%**	**70%**	**96.76%**	**93.22%**	**93.11%**	**93.12**	**93.29**	**93.22**	**93.17%**
	40%	60%	94.58%	92.78%	92.67%	92.61	92.93	92.67	91.15%
Case6	30%	70%	96.02%	92.44%	92.44%	92.38	92.88	92.44	91.22%
	40%	60%	94.92%	93.00%	93.17%	93.12	93.46	93.17	90.11%
Case7	30%	70%	98.60%	89.78%	90.00%	89.96	90.07	90.00	90.10%
	40%	60%	93.06%	90.83%	88.17%	87.93	88.29	88.17	91.80%

Model Loss of ResNeSt50

The model loss evaluation provides valuable insights into the training and validation losses across different cases and data splits. Case 5, with a 40%-60% data split, emerged as the first most effective configuration, achieving a training loss of 0.1355 and a validation loss of 0.1901 These loss values correspond well with the model's strong performance metrics, indicating efficient learning and excellent generalization to unseen data.

Table 6.15: Model loss of ResNeSt50 on incorporated dataset

Model Loss				
			Training	**Validation**
Case 1	30%	70%	0.2310	0.1352
	40%	***60%***	***0.1354***	***0.2136***
Case 2	30%	70%	0.1825	0.2536
	40%	60%	0.0315	0.2536
Case 3	30%	70%	0.1452	0.2478
	40%	60%	0.1519	0.2132
Case 4	30%	70%	0.1536	0.2276
	40%	60%	0.0839	0.2345
Case 5	**30%**	**70%**	**0.1322**	**0.2001**
	40%	60%	0.1315	0.2025
Case 6	30%	70%	0.0785	0.2145
	40%	60%	0.1124	0.2236
Case 7	30%	70%	0.1936	0.2893
	40%	60%	0.1236	0.2785

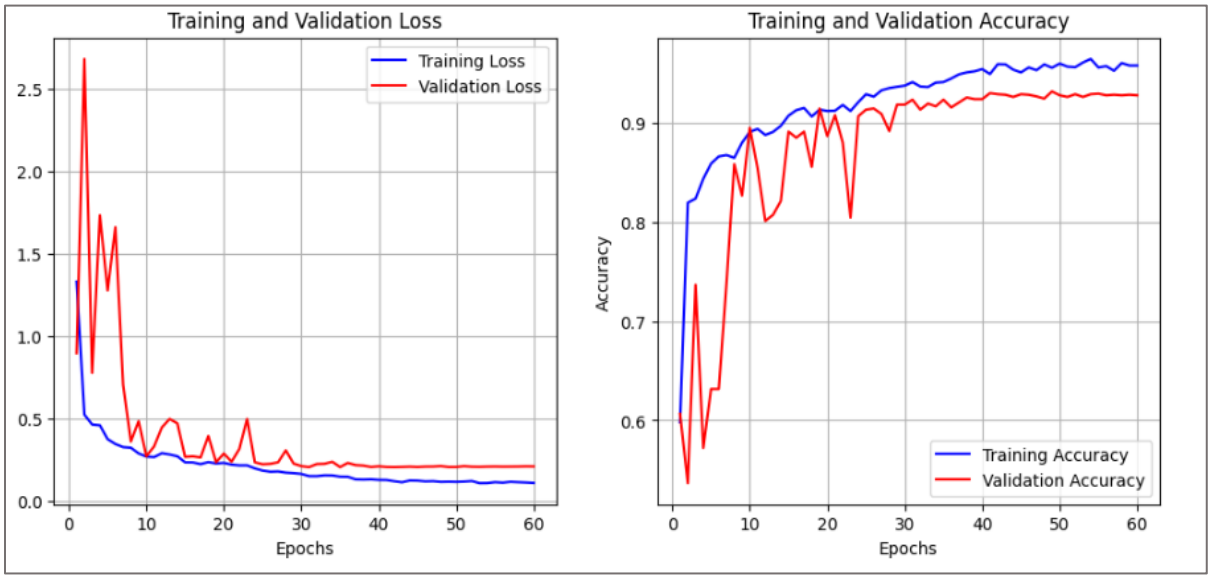


Figure 6.10: Learning Curve for the top performer of ResNeSt50 on incorporated Dataset (case 5-30% 70% Data Split)

6.6. Comparison of Results with Related works

In the realm of gastroenterology, effectively diagnosing and categorizing intestinal diseases through imaging techniques poses significant challenges. Current methodologies often struggle to accommodate the diverse spectrum of gastrointestinal disorders while also they are not often lightweight enough for practical application. This is particularly evident in Ethiopia, where local disease patterns and characteristics may diverge markedly from those reflected in publicly available datasets. Most existing models do not adequately address these regional nuances, which can hinder diagnostic accuracy and efficacy. This gap in the literature underscores the pressing need for a tailored classification system that not only grapples with the complexities presented by global datasets but also integrates local Ethiopian disease data.

In contrast to the existing approaches, our study offers a novel contribution by classifying six distinct gastrointestinal disease classes while implementing a lightweight model specifically designed for efficiency and effectiveness in resource-constrained environments. This lightweight model not only enhances computational efficiency but also ensures that it can be deployed in clinical settings with limited computational resources. Furthermore, by

incorporating a local dataset, my methodology effectively addresses the regional variances in disease manifestation, ensuring that the classification system is relevant and applicable to the Ethiopian context.

The effectiveness of our model is evidenced by its performance metrics, which reflect significant improvements over previous state-of-the-art methods. While models like GIT-Net achieved a notable good accuracy using the KVASIR v2 dataset, my approach demonstrates an ability to generalize better to local conditions, thus providing a more robust diagnostic tool for clinicians in Ethiopia. Additionally, the integration of local data ensures that the model is well-calibrated to the unique challenges faced by healthcare providers in the region, ultimately enhancing the quality of patient care and fostering a more nuanced understanding of gastrointestinal diseases in diverse populations. This comprehensive approach not only fills the existing gaps in the literature but also paves the way for future research and development in the field of gastrointestinal imaging and diagnostics.

In comparison to the study conducted by Malik et al. (2022), which utilized publicly available databases to focus on diagnosing ulcerative colitis, polyps, and dyed-lifted polyps with a multi-classification deep learning (DL) approach, our study offers a more comprehensive framework by classifying six distinct gastrointestinal disease classes. While Malik et al. employed four models, achieving an impressive accuracy with their VGG-19 + CNN model, my lightweight model not only aims for high accuracy but also ensures efficiency in environments where computational resources may be limited. The incorporation of a local Ethiopian dataset allows my model to better adapt to regional disease manifestations, thereby enhancing its diagnostic relevance and applicability. Unlike the approach taken by Malik et al., which focused on a narrower set of diseases, my work broadens the scope of classification, addressing a wider array of conditions that may be encountered in clinical practice. This enables a more holistic understanding of gastrointestinal diseases and reinforces the need for tailored solutions that reflect the unique healthcare challenges faced in Ethiopia, ultimately contributing to improved patient outcomes and more effective healthcare delivery.

Hmoud et al. (2021) utilized the Kvasir dataset, comprising 5,000 images evenly distributed among five types of lower gastrointestinal diseases, to develop a diagnostic system for GI disease classification using deep learning techniques. They employed pretrained CNN models—GoogleNet, ResNet-50, and AlexNet—applying transfer learning for fine-tuning, with AlexNet achieving the best performance: 97% accuracy, 96.8% sensitivity, 99.20% specificity, and an AUC of 99.98%. However, their use of the original Kvasir dataset, which contains fewer and less diverse disease types, contrasts with our study that employed the more comprehensive Kvasir V2 dataset, which represents a broader spectrum of GI diseases, making our model more applicable to real-world clinical scenarios. By using Kvasir V2, alongside a local Ethiopian dataset, our approach achieved high accuracy and generalizability across diverse populations, further enhancing the model's robustness and clinical relevance.

6.7. Research Question Discussion

Answer for Q1: Among the various pre-trained models evaluated, ResNeSt50 emerged as the most suitable for effective classification of gastrointestinal diseases. The ResNeSt50 model outperformed other architectures like ResNeSt101, ResNeSt200, ResNet50, and VGG16 in terms of accuracy. This superiority can be attributed to its advanced neural network design that incorporates split-attention blocks, allowing it to capture more complex features and achieve higher classification accuracy. The model's architecture facilitates better feature extraction and representation, which is crucial for distinguishing between various gastrointestinal conditions in medical imaging. The choice of fine-tuning parameters such as learning rate and dropout rate directly affects the model's ability to learn new features from the local Ethiopian dataset without overfitting. Unfreezing the last two layers allows the model to adjust its high-level features to better fit the local data while retaining the foundational knowledge acquired from the Kvasir V2 dataset. The scheduler factor of 0.2 helps in managing the learning rate schedule, which can be crucial in preventing the model from converging too quickly or getting stuck in local minima. These techniques collectively help in transferring the model's learned features to new, diverse data while maintaining its performance on previously seen data.

Answer for Q2: Effective fine-tuning techniques for adapting the model to local Ethiopian diseases while preserving performance on the Kvasir V2 dataset included using a learning rate of 0.01, a dropout rate of 0.5, a weight decay of 1e-3, unfreezing the last two layers, and fully connected layer and scheduler factor of 0.2. These configurations were pivotal in optimizing the model's ability to generalize across different datasets. The choice of fine-tuning parameters such as learning rate and dropout rate directly affects the model's ability to learn new features from the local Ethiopian dataset without overfitting. Unfreezing the last two layers and FCL allows the model to adjust its high-level features to better fit the local data while retaining the foundational knowledge acquired from the Kvasir V2 dataset. The scheduler factor of 0.2 helps in managing the learning rate schedule, which can be crucial in preventing the model from converging too quickly or getting stuck in local minima. These techniques collectively help in transferring the model's learned features to new, diverse data while maintaining its performance on previously seen data.

Answer for Q3: Integrating local Ethiopian disease data significantly enhanced the model's performance, improving the local test set accuracy from 88.75% to 93.17%. This demonstrates that incorporating diverse data sources can help improve the model's ability to generalize and accurately classify diseases that might be underrepresented in the local dataset. The integration of local Ethiopian data highlights the benefits of using a more diverse and representative dataset for training. By including local data, the model gains exposure to a wider variety of disease presentations and conditions specific to the Ethiopian population, which improves its generalization capabilities. This improvement in accuracy on local data underscores the importance of dataset diversity in training robust and effective diagnostic models. The results suggest that local datasets should be considered in the training process to enhance the applicability and reliability of machine learning models in different geographic and demographic contexts.

CHAPTER SEVEN

7. Conclusions and Future works

7.1. Conclusions

This thesis aimed to enhance the classification of gastrointestinal (GI) diseases using gastroenterology imaging through the application of transfer learning. The increasing prevalence of GI disorders underscores the need for accurate and efficient diagnostic tools, and this research sought to meet this demand by leveraging advanced deep learning models within the framework of transfer learning.

In our comparative analysis, we evaluated several pretrained deep learning models, including ResNeSt50, ResNeSt101, ResNeSt200, ResNet50, and VGG16. Among these, ResNeSt50 emerged as the top-performing model. Our extensive experiments, conducted on the Kvasir V2 dataset and a local dataset, identified Case 3 (40%-60% split) which was configured with a learning rate of 0.01, a dropout rate of 0.5, a weight decay of 1e-3, unfreezing of the last two layers, and FCL, scheduler factor of 0.2 as the highest achiever. This configuration attained the highest validation accuracy of 93.33% and a test accuracy of 93.22%, along with strong performance across key metrics such as F1 score, precision, and recall, all exceeding 93%. Additionally, Case 3 demonstrated superior generalization, achieving a local data test accuracy of 88.75%.

To further enhance the model's robustness and accuracy, we incorporated the Kvasir V2 dataset with our local dataset during training. Among the evaluated cases on this incorporated data, Case 5 (30%-70%) again proved to be the top performer, achieving a validation accuracy of 93.33% and a test accuracy of 93.22%. It also maintained consistently high metrics across F1 score, precision, and recall, all within the range of 93.32% to 93.39%. This configuration significantly improved the local test set accuracy from 88.75% to 93.17%, underscoring its strong generalization capabilities across different datasets.

These results validate the effectiveness of the ResNeSt50 architecture in comparison to other models, demonstrating its potential for significantly improving diagnostic accuracy in GI disease classification when trained on a combined dataset. Future research could focus on

expanding the dataset, exploring additional model architectures, and further refining transfer learning techniques to continue advancing automated GI disease diagnosis

7.2. Future Works

Future work in this field could focus on several key areas to build upon the findings of this thesis. Expanding the dataset to include more diverse and larger samples would enhance the model's generalization capabilities, particularly across different GI diseases and imaging conditions. Exploring advanced architectures like Inveption-ResNet v2/v4, Inceptionv4, DenseNet or Vision Transformer and hybrid models could offer further improvements in accuracy and efficiency. Additionally, to facilitate the practical use of the trained model in real-world healthcare, it is crucial to develop strategies for seamless integration into clinical settings. This includes ensuring the model is compatible with existing medical imaging systems and designing a user-friendly interface that allows healthcare professionals to interact with the technology easily. Furthermore, incorporating explainable AI techniques will enhance the transparency and interpretability of the model’s predictions, helping clinicians understand how decisions are mad. These avenues collectively aim to advance the reliability and effectiveness of GI disease classification models, contributing to improved diagnostic tools in healthcare.

REFERENCES

[1] Wang, T., Liu, K., Wen, L., Yang, Y., Yin, X., Liu, K., ... & Chen, D. (2020). Autophagy and gastrointestinal diseases. *Autophagy: Biology and Diseases: Clinical Science*, 529-556.

[2] Alatab, S., Sepanlou, S. G., Ikuta, K., Vahedi, H., Bisignano, C., Safiri, S., ... & Naghavi, M. (2020). The global, regional, and national burden of inflammatory bowel disease in 195 countries and territories, 1990–2017: a systematic analysis for the Global Burden of Disease Study 2017. *The Lancet gastroenterology & hepatology*, *5*(1), 17-30.

[3] Butt, I., & Kasmin, F. (2020). Esophageal pH Monitoring.

[4] Zhang, H., et al. (2020). "ResNeSt: Split-Attention Networks." arXiv preprint arXiv:2004.08955.

[5] Malik, H., Naeem, A., Sadeghi-Niaraki, A., Naqvi, R. A., & Lee, S. W. (2023). Multi-classification deep learning models for detection of ulcerative colitis, polyps, and dyed-lifted polyps using wireless capsule endoscopy images. *Complex & Intelligent Systems*, 1-21.

[6] Bhardwaj, P., Kumar, S., & Kumar, Y. (2023). A Comprehensive Analysis of Deep Learning-Based Approaches for the Prediction of Gastrointestinal Diseases Using Multi-class Endoscopy Images. *Archives of Computational Methods in Engineering*, 1-18.

[7] Yogapriya, J., Chandran, V., Sumithra, M. G., Anitha, P., Jenopaul, P., & Suresh Gnana Dhas, C. (2021). Gastrointestinal tract disease classification from wireless endoscopy images using pretrained deep learning model. *Computational and mathematical methods in medicine*, *2021*.

[8] Mohapatra, S., Pati, G. K., Mishra, M., & Swarnkar, T. (2023). Gastrointestinal abnormality detection and classification using empirical wavelet transform and deep convolutional neural network from endoscopic images. *Ain Shams Engineering Journal*, *14*(4), 101942.

[9] Hmoud Al-Adhaileh, M., Mohammed Senan, E., Alsaade, W., Aldhyani, T. H. H., Alsharif, N., Abdullah Alqarni, A., ... & Jadhav, M. E. (2021). Deep learning algorithms for detection and classification of gastrointestinal diseases. *Complexity*, *2021*, 1-12.

[10] Gunasekaran, H., Ramalakshmi, K., Swaminathan, D. K., & Mazzara, M. (2023). GIT-Net: an ensemble deep learning-based GI tract classification of endoscopic images. *Bioengineering*, *10*(7), 809.

[11] Sivari, E., Bostanci, E., Guzel, M. S., Acici, K., Asuroglu, T., & Ercelebi Ayyildiz, T. (2023). A New Approach for Gastrointestinal Tract Findings Detection and Classification: Deep Learning-Based Hybrid Stacking Ensemble Models. *Diagnostics*, *13*(4), 720.

[12] Nouman Noor, M., Nazir, M., Khan, S. A., Song, O. Y., & Ashraf, I. (2023). Efficient gastrointestinal disease classification using pretrained deep convolutional neural network. *Electronics*, *12*(7), 1557.

[13] Pham, T., Telias, I., & Beitler, J. R. (2020). Esophageal manometry. *Respiratory Care*, *65*(6), 772-792.

[14] Rashid, M., & Lee, J. (2016). Serologic testing in celiac disease: Practical guide for clinicians. *Canadian Family Physician*, *62*(1), 38-43.
[15] Ślósarz, D., Poniewierka, E., Neubauer, K., & Kempiński, R. (2021). Ultrasound elastography in the assessment of the intestinal changes in inflammatory bowel disease—Systematic review. *Journal of Clinical Medicine*, *10*(18), 4044.
[16] Sureshkumar, A., Hansen, B., & Ersahin, D. (2020, February). Role of nuclear medicine in imaging. In *Seminars in Ultrasound, CT and MRI* (Vol. 41, No. 1, pp. 10-19). WB Saunders.
[17] Tran-Nguyen, T., Fernandez-Esparrach, G., Ash, S., Balaguer, F., Bird-Lieberman, E. L., Córdova, H., ... & Kupcinskas, J. (2021). Biopsy sampling in upper gastrointestinal endoscopy: a survey from 10 tertiary referral centres across europe. *Digestive Diseases*, *39*(3), 179-189.
[18] Rana, S. V., & Malik, A. (2014). Hydrogen breath tests in gastrointestinal diseases. *Indian Journal of Clinical Biochemistry*, *29*, 398-405.
[19] Goodman, R. P., & Chung, D. C. (2016). Clinical Genetic Testing in Gastroenterology. *Clinical and Translational Gastroenterology*, *7*(4), e167.
[20] Kasırga, E. (2019). The importance of stool tests in diagnosis and follow-up of gastrointestinal disorders in children. *Turkish Archives of Pediatrics/Türk Pediatri Arşivi*, *54*(3), 141.
[21] Hooshyar, H., & Rostamkhani, P. (2022). Accurate laboratory diagnosis of human intestinal and extra-intestinal amoebiasis. *Gastroenterology and Hepatology From Bed to Bench*, *15*(4), 343.
[22] Daly, M., & Zarate-Lopez, N. (2021). Functional gastrointestinal disorders: History taking skills in practice. *Clinical Medicine*, *21*(5), e480.
[23] Cummins, G. (2021). Smart pills for gastrointestinal diagnostics and therapy. *Advanced Drug Delivery Reviews*, *177*, 113931.
[24] Ionescu, A. G., Glodeanu, A. D., Ionescu, M., Zaharie, S. I., Ciurea, A. M., Golli, A. L., ... & Vere, C. C. (2022). Clinical impact of wireless capsule endoscopy for small bowel investigation. *Experimental and Therapeutic Medicine*, *23*(4), 1-9.
[25] Nehra, A. K., Sheedy, S. P., Johnson, C. D., Flicek, K. T., Venkatesh, S. K., Heiken, J. P., ... & Fidler, J. L. (2022). Imaging Review of Gastrointestinal Motility Disorders. *RadioGraphics*, *42*(7), 2014-2036.
[26] Suganyadevi, S., Seethalakshmi, V., & Balasamy, K. (2022). A review on deep learning in medical image analysis. *International Journal of Multimedia Information Retrieval*, *11*(1), 19-38.
[27] Puttagunta, M., & Ravi, S. (2021). Medical image analysis based on deep learning approach. *Multimedia tools and applications*, *80*, 24365-24398.
[28] Yu, X., Wang, J., Hong, Q. Q., Teku, R., Wang, S. H., & Zhang, Y. D. (2022). Transfer learning for medical images analyses: A survey. *Neurocomputing*, *489*, 230-254.
[29] Meng, T., Jing, X., Yan, Z., & Pedrycz, W. (2020). A survey on machine learning for data fusion. *Information Fusion*, *57*, 115-129.
[30] Melak, W., Asmare, W., Bane, A., & Erkie, M. (2023). Predictive value of alarm features in diagnosing upper gastrointestinal malignancies among dyspeptic patients: A cross-sectional study in Ethiopia. *Gastroenterol Hepatol Res*, *5*(3), 13.

[31] Badea, R., Sudoł-Szopińska, I., Mueller, S., Ştefănescu, H., & Lupsor Platon, M. (2014). Imaging in gastroenterology. *Gastroenterology Research and Practice*, *2014*.
[32] Zeng, W., Qi, K., Ye, M., Zheng, L., Liu, X., Hu, S., ... & Wei, Y. (2022). Gastrointestinal symptoms are associated with severity of coronavirus disease 2019: a systematic review and meta-analysis. *European journal of gastroenterology & hepatology*, *34*(2), 168-176.
[33] Di Giulio, E., Angeletti, S., & Desideri, F. (2016). Upper and lower gastrointestinal endoscopy. *Recenti Progressi in Medicina*, *107*(6), 278-284.
[34] De Backer, A. I., Mortelé, K. J., De Keulenaer, B. L., Henckaerts, L., & Verhaert, L. (2006). CT and MR imaging of gastrointestinal tuberculosis. *Jbr-btr*, *89*(4), 190-4.
[35] Sun, B., Liu, J., Li, S., Lovell, J. F., & Zhang, Y. (2023). Imaging of Gastrointestinal Tract Ailments. *Journal of imaging*, *9*(6), 115.
[36] Varoquaux, G., & Cheplygina, V. (2022). Machine learning for medical imaging: methodological failures and recommendations for the future. *NPJ digital medicine*, *5*(1), 48.
[37] van der Sommen, F., de Groof, J., Struyvenberg, M., van der Putten, J., Boers, T., Fockens, K., ... & Bergman, J. J. (2020). Machine learning in GI endoscopy: practical guidance in how to interpret a novel field. *Gut*, *69*(11), 2035-2045.
[38] Rana, M., & Bhushan, M. (2023). Machine learning and deep learning approach for medical image analysis: diagnosis to detection. *Multimedia Tools and Applications*, *82*(17), 26731-26769.
[39] Sharma, N., Jain, V., & Mishra, A. (2018). An analysis of convolutional neural networks for image classification. *Procedia computer science*, *132*, 377-384.
[40] Chen, F., & Tsou, J. Y. (2022). Assessing the effects of convolutional neural network architectural factors on model performance for remote sensing image classification: An in-depth investigation. *International Journal of Applied Earth Observation and Geoinformation*, *112*, 102865.
[41] Sarvamangala, D. R., & Kulkarni, R. V. (2022). Convolutional neural networks in medical image understanding: a survey. *Evolutionary intelligence*, *15*(1), 1-22.
[42] Kim, H. E., Cosa-Linan, A., Santhanam, N., Jannesari, M., Maros, M. E., & Ganslandt, T. (2022). Transfer learning for medical image classification: a literature review. *BMC medical imaging*, *22*(1), 69.
[43] Niu, Z., Zhong, G., & Yu, H. (2021). A review on the attention mechanism of deep learning. *Neurocomputing*, *452*, 48-62.
[44] Hernández, A., & Amigó, J. M. (2021). Attention mechanisms and their applications to complex systems. *Entropy*, *23*(3), 283.
[45] Kapoor, A., Shah, R., Bhuva, R., & Pandit, T. (2020). Understanding inception network architecture For image classification.
[46] Ali, L., Alnajjar, F., Jassmi, H. A., Gocho, M., Khan, W., & Serhani, M. A. (2021). Performance evaluation of deep CNN-based crack detection and localization techniques for concrete structures. *Sensors*, *21*(5), 1688.
[47] Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning* (pp. 6105-6114). PMLR.

[48] Raza, R., Zulfiqar, F., Khan, M. O., Arif, M., Alvi, A., Iftikhar, M. A., & Alam, T. (2023). Lung-EffNet: Lung cancer classification using EfficientNet from CT-scan images. *Engineering Applications of Artificial Intelligence*, *126*, 106902.

[49] Zhou, A., Ma, Y., Ji, W., Zong, M., Yang, P., Wu, M., & Liu, M. (2023). Multi-head attention-based two-stream EfficientNet for action recognition. *Multimedia Systems*, *29*(2), 487-498.

[50] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

[51] Gyasi, E. K., & Swarnalatha, P. (2023). Cloud-MobiNet: an abridged Mobile-Net convolutional neural network model for ground-based cloud classification. *Atmosphere*, *14*(2), 280.

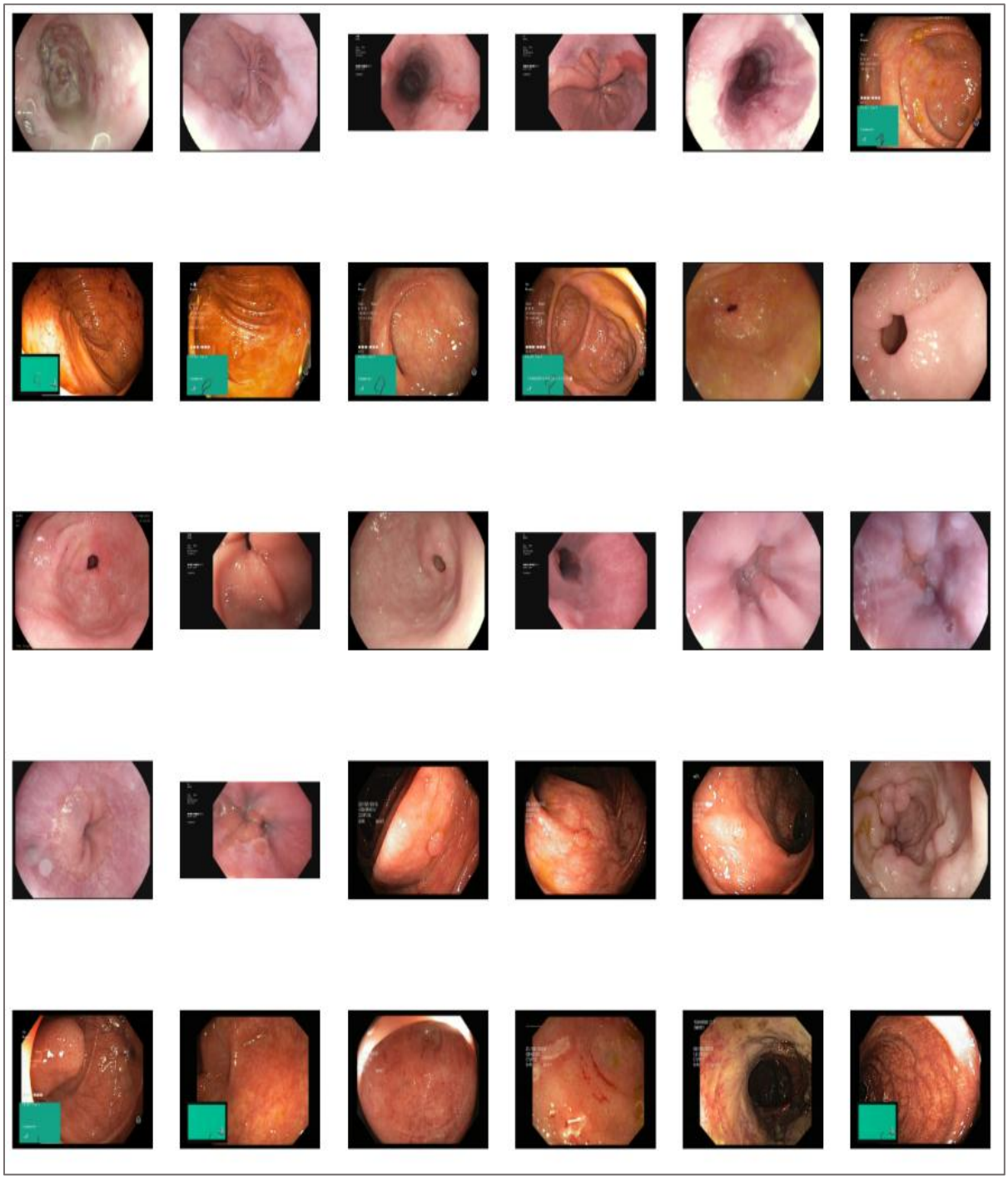
[52] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[53] Hasanah, S. A., Pravitasari, A. A., Abdullah, A. S., Yulita, I. N., & Asnawi, M. H. (2023). A deep learning review of resnet architecture for lung disease Identification in CXR Image. *Applied Sciences*, *13*(24), 13111.

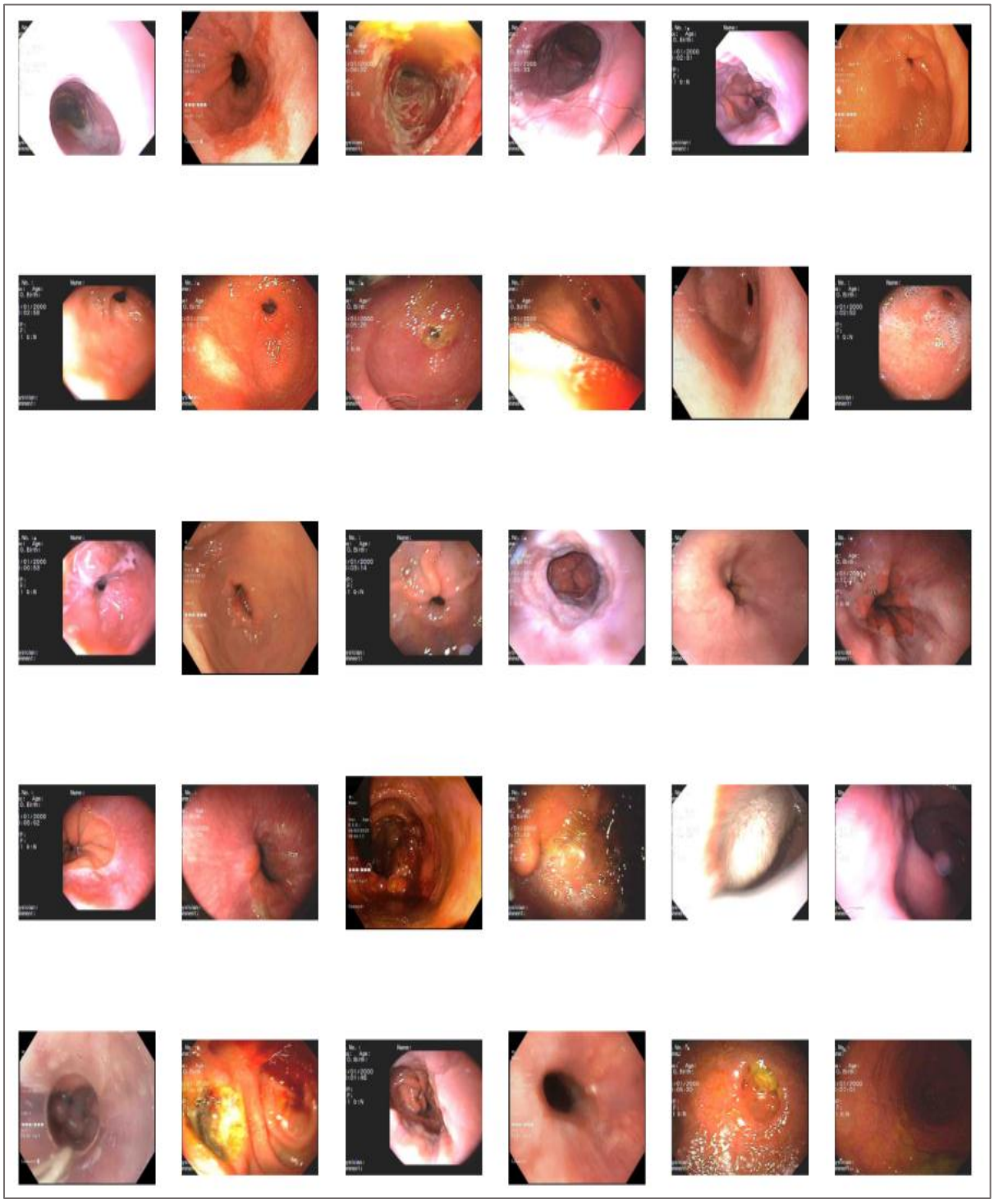
APPENDIXES

Appendix A: Sample Dataset's

Sample images from KvasirV2 Dataset



Sample Image from Local Dataset



Appendix B: Sample Code

Import necessary Libraries

```
import os
import random
import numpy as np
import cv2
import time
import torch
from PIL import Image
from torchvision import transforms
from torchvision import transforms,datasets,models
from sklearn.model_selection import train_test_split
from torch.utils.data import Dataset, DataLoader
import matplotlib.pyplot as plt
import torch.nn as nn
import torch.optim as optim
import copy
import pickle
import resnest.torch as resnest  # Import the resnest module
from torchvision.transforms import functional as F
from skimage import exposure
from collections import defaultdict
import shutil
from torchvision.datasets import ImageFolder
from torch.utils.data import DataLoader, Dataset, random_split
from resnest.torch import resnest50
#from torchvision import models
from sklearn.model_selection import train_test_split
from torch.optim import lr_scheduler
```

```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, confusion_matrix, classification_report
import seaborn as sns
```

```
import os
from torchvision import datasets
from collections import defaultdict
import random
import shutil
```

Code to Define GPU Device

```
# Define GPU device
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

Code for Normalization and Data Augmentation

```
# Define normalization
normalize = transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                 std=[0.229, 0.224, 0.225])

data_transforms = {
    'train':
    transforms.Compose([
        transforms.Resize((224,224)),
        transforms.RandomAffine(0, shear=10, scale=(0.8,1.2)),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        normalize
    ]),
    'validation':
    transforms.Compose([
        transforms.Resize((224,224)),
        transforms.ToTensor(),
        normalize
    ]),
    'test':
    transforms.Compose([
        transforms.Resize((224,224)),
        transforms.ToTensor(),
        normalize
    ]),
}
```

Code for Image Transformation and Data Loading

```
splittedData = 'C:\\Users\\model\\Documents\\Tadiwos Data\\Datasets\\split_data_kvasir_70_15_15\\'
# Apply transforms and create DataLoaders
image_datasets = {
    'train':  datasets.ImageFolder(splittedData+'Train', data_transforms['train']),
    'val': datasets.ImageFolder(splittedData + 'Val', data_transforms['validation']),
    'test': datasets.ImageFolder(splittedData+'Test', data_transforms['test'])
}

dataloaders = {
    'train': DataLoader(image_datasets['train'], batch_size=40, shuffle=True, num_workers=0),
    'val': DataLoader(image_datasets['val'], batch_size=40, shuffle=False, num_workers=0),
    'test': DataLoader(image_datasets['test'], batch_size=40, shuffle=False, num_workers=0)
}
```

Code for ResNeSt50 Model setup

```
# Load ResNeSt model
model = resnest50(pretrained=True)
class ResNeStTransfer(nn.Module):
    def __init__(self, model, num_classes):
        super(ResNeStTransfer, self).__init__()
        # Extract all layers except the final classification layer
        self.features = nn.Sequential(*list(model.children())[:-2])
        # Adaptive pooling to ensure output size is compatible
        self.pool = nn.AdaptiveAvgPool2d(1)
        # Dropout layer to reduce overfitting
        self.dropout = nn.Dropout(p=0.5)
        # Access the number of features in the penultimate layer
        num_ftrs = model.fc.in_features
        # New fully connected layer for the number of target classes
        self.fc = nn.Linear(num_ftrs, num_classes)
    def forward(self, x):
        # Pass input through feature extraction layers
        x = self.features(x)
        # Adaptive average pooling
        x = self.pool(x)
        # Flatten the output to feed into the fully connected layer
        x = torch.flatten(x, 1)
        # Apply dropout for regularization
        x = self.dropout(x)
        # Final classification layer
        x = self.fc(x)
        return x

# Freeze model parameters
for param in model.parameters():
    param.requires_grad = False

for param in model.layer3.parameters():
    param.requires_grad = True
for param in model.layer4.parameters():
    param.requires_grad = True
num_classes = 6
model = ResNeStTransfer(model, num_classes)
# Modify the last layer for transfer learning
num_ftrs = model.fc.in_features
model.fc = nn.Linear(num_ftrs, num_classes)
# Define loss function and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(filter(lambda p: p.requires_grad, model.parameters()), lr=learning_rate, weight_decay=1e-3)
# Learning Rate Scheduler with ReduceLROnPlateau
exp_lr_scheduler = lr_scheduler.ReduceLROnPlateau(optimizer, mode='min', patience=3, factor=0.2)
# Use SGDR
#exp_lr_scheduler = lr_scheduler.CosineAnnealingWarmRestarts(optimizer, T_0=10, T_mult=2)
```

Code for ResNeSt50 Model Summary

```
model.to(device)
from torchsummary import summary

# Assuming your model is named `model`
summary(model, (3, 224, 224))  # Adjust input size as needed
```

ResNeSt50 Model Summary

```
            Conv2d-515           [-1, 2048, 7, 7]       1,048,576
       BatchNorm2d-516           [-1, 2048, 7, 7]           4,096
         AvgPool2d-517           [-1, 1024, 7, 7]               0
            Conv2d-518           [-1, 2048, 7, 7]       2,097,152
       BatchNorm2d-519           [-1, 2048, 7, 7]           4,096
              ReLU-520           [-1, 2048, 7, 7]               0
        Bottleneck-521           [-1, 2048, 7, 7]               0
            Conv2d-522            [-1, 512, 7, 7]       1,048,576
       BatchNorm2d-523            [-1, 512, 7, 7]           1,024
              ReLU-524            [-1, 512, 7, 7]               0
            Conv2d-525           [-1, 1024, 7, 7]       2,359,296
       BatchNorm2d-526           [-1, 1024, 7, 7]           2,048
              ReLU-527           [-1, 1024, 7, 7]               0
            Conv2d-528            [-1, 256, 1, 1]         131,328
       BatchNorm2d-529            [-1, 256, 1, 1]             512
              ReLU-530            [-1, 256, 1, 1]               0
            Conv2d-531           [-1, 1024, 1, 1]         263,168
         rSoftMax-532                 [-1, 1024]               0
       SplAtConv2d-533            [-1, 512, 7, 7]               0
            Conv2d-534           [-1, 2048, 7, 7]       1,048,576
       BatchNorm2d-535           [-1, 2048, 7, 7]           4,096
              ReLU-536           [-1, 2048, 7, 7]               0
        Bottleneck-537           [-1, 2048, 7, 7]               0
            Conv2d-538            [-1, 512, 7, 7]       1,048,576
       BatchNorm2d-539            [-1, 512, 7, 7]           1,024
              ReLU-540            [-1, 512, 7, 7]               0
            Conv2d-541           [-1, 1024, 7, 7]       2,359,296
       BatchNorm2d-542           [-1, 1024, 7, 7]           2,048
              ReLU-543           [-1, 1024, 7, 7]               0
            Conv2d-544            [-1, 256, 1, 1]         131,328
       BatchNorm2d-545            [-1, 256, 1, 1]             512
              ReLU-546            [-1, 256, 1, 1]               0
            Conv2d-547           [-1, 1024, 1, 1]         263,168
         rSoftMax-548                 [-1, 1024]               0
       SplAtConv2d-549            [-1, 512, 7, 7]               0
            Conv2d-550           [-1, 2048, 7, 7]       1,048,576
       BatchNorm2d-551           [-1, 2048, 7, 7]           4,096
              ReLU-552           [-1, 2048, 7, 7]               0
        Bottleneck-553           [-1, 2048, 7, 7]               0
AdaptiveAvgPool2d-554           [-1, 2048, 1, 1]               0
           Dropout-555                 [-1, 2048]               0
            Linear-556                    [-1, 6]          12,294
================================================================
Total params: 46,238,310
Trainable params: 44,548,998
Non-trainable params: 1,689,312
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 590.10
Params size (MB): 176.39
Estimated Total Size (MB): 767.05
----------------------------------------------------------------
```

Code for ResNeSt50 Model Training

```
def train_model(model, dataloaders, dataset_sizes, criterion, optimizer, scheduler, num_epochs=25, patience=20, device='cuda'):
    best_model_wts = model.state_dict()
    best_acc = 0.0

    train_losses = []
    val_losses = []
    train_accuracies = []
    val_accuracies = []
    early_stop = False
    patience_counter = 0
    min_val_loss = float('inf')
    model.to(device)  # Move model to device

    for epoch in range(num_epochs):
        if early_stop:
            print("Early stopping")
            break

        print(f'Epoch {epoch}/{num_epochs - 1}')
        print('-' * 10)
        for phase in ['train', 'val']:
            if phase == 'train':
                model.train()
            else:
                model.eval()

            running_loss = 0.0
            running_corrects = 0

            for inputs, labels in dataloaders[phase]:
                inputs = inputs.to(device)
                labels = labels.to(device)

                optimizer.zero_grad()

                with torch.set_grad_enabled(phase == 'train'):
                    outputs = model(inputs)
                    _, preds = torch.max(outputs, 1)
                    loss = criterion(outputs, labels)
                    if phase == 'train':
                        loss.backward()
                        optimizer.step()

                running_loss += loss.item() * inputs.size(0)
                running_corrects += torch.sum(preds == labels.data)
```

```
            epoch_loss = running_loss / dataset_sizes[phase]
            epoch_acc = running_corrects.double() / dataset_sizes[phase]

            if phase == 'train':
                train_losses.append(epoch_loss)
                train_accuracies.append(epoch_acc.item())  # Convert to Python float
            else:
                val_losses.append(epoch_loss)
                val_accuracies.append(epoch_acc.item())  # Convert to Python float

                # Update the learning rate scheduler based on validation loss
                if isinstance(scheduler, lr_scheduler.ReduceLROnPlateau):
                    scheduler.step(epoch_loss)
                else:
                    scheduler.step()

                if epoch_loss < min_val_loss:
                    min_val_loss = epoch_loss
                    best_acc = epoch_acc
                    best_model_wts = model.state_dict()
                    patience_counter = 0
                else:
                    patience_counter += 1
                    if patience_counter >= patience:
                       # early_stop = True

            print(f'{phase} Loss: {epoch_loss:.4f} Acc: {epoch_acc:.4f}')

        print()

    print(f'Best val Acc: {best_acc:.4f}')
    model.load_state_dict(best_model_wts)

    return model, train_losses, val_losses, train_accuracies, val_accuracies
```

Code for plotting Training Curve

```
import matplotlib.pyplot as plt

def plot_training_results(train_losses, val_losses, train_accuracies, val_accuracies, num_epochs):
    epochs = range(1, num_epochs + 1)

    # Ensure all inputs are lists or NumPy arrays and have the correct length

    plt.figure(figsize=(12, 5))

    # Plotting training and validation loss
    plt.subplot(1, 2, 1)
    plt.plot(epochs, train_losses, 'b', label='Training Loss')
    plt.plot(epochs, val_losses, 'r', label='Validation Loss')
    plt.title('Training and Validation Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.grid(True)  # Enable grid
    # Plotting training and validation accuracy
    plt.subplot(1, 2, 2)
    plt.plot(epochs, train_accuracies, 'b', label='Training Accuracy')
    plt.plot(epochs, val_accuracies, 'r', label='Validation Accuracy')
    plt.title('Training and Validation Accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.grid(True)  # Enable grid
    plt.show()
```

Code for evaluate model performance

```
# Evaluate the model
def evaluate_model(model, dataloader, class_names):
    model.eval()

    all_preds = []
    all_labels = []

    with torch.no_grad():
        for inputs, labels in dataloader:
            inputs = inputs.to(device)
            labels = labels.to(device)

            outputs = model(inputs)
            _, preds = torch.max(outputs, 1)

            all_preds.extend(preds.cpu().numpy())
            all_labels.extend(labels.cpu().numpy())

    all_preds = np.array(all_preds)
    all_labels = np.array(all_labels)

    accuracy = accuracy_score(all_labels, all_preds)
    f1 = f1_score(all_labels, all_preds, average='weighted')
    precision = precision_score(all_labels, all_preds, average='weighted')
    recall = recall_score(all_labels, all_preds, average='weighted')

    conf_matrix = confusion_matrix(all_labels, all_preds)
    specificity = calculate_specificity(conf_matrix, len(class_names))

    print_evaluation_metrics(accuracy, f1, precision, recall, specificity, all_labels, all_preds, class_names)
    plot_confusion_matrix(conf_matrix, class_names)
    plot_metrics_bar_chart(accuracy, precision, recall, f1, specificity)

    return accuracy, f1, precision, recall, specificity, conf_matrix
```

Code for Defining Evaluation Metrics

```
# Calculate specificity for each class
def calculate_specificity(conf_matrix, num_classes):
    specificity = []
    for i in range(num_classes):
        tn = conf_matrix.sum() - (conf_matrix[i, :].sum() + conf_matrix[:, i].sum() - conf_matrix[i, i])
        fp = conf_matrix[:, i].sum() - conf_matrix[i, i]
        specificity.append(tn / (tn + fp))
    return specificity

# Print evaluation metrics
def print_evaluation_metrics(accuracy, f1, precision, recall, specificity, all_labels, all_preds, class_names):
    print(f'Accuracy: {accuracy:.4f}')
    print(f'F1-Score: {f1:.4f}')
    print(f'Precision: {precision:.4f}')
    print(f'Recall: {recall:.4f}')
    print(f'Specificity: {specificity}')
    print('\nClassification Report:')
    print(classification_report(all_labels, all_preds, target_names=class_names))

# Plot confusion matrix
def plot_confusion_matrix(conf_matrix, class_names):
    plt.figure(figsize=(10, 8))
    sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=class_names, yticklabels=class_names)
    plt.ylabel('Actual')
    plt.xlabel('Predicted')
    plt.title('Confusion Matrix')
    plt.show()

# Plot metrics bar chart
def plot_metrics_bar_chart(accuracy, precision, recall, f1, specificity):
    metrics = ['Accuracy', 'Precision', 'Recall', 'F1-Score', 'Specificity']
    values = [accuracy, precision, recall, f1, np.mean(specificity)]

    plt.figure(figsize=(10, 5))
    plt.bar(metrics, values, color=['skyblue', 'orange', 'green', 'red', 'purple'])
    plt.title('Evaluation Metrics')
    plt.ylim(0, 1)
    plt.ylabel('Score')
    plt.show()
```

Code for data split

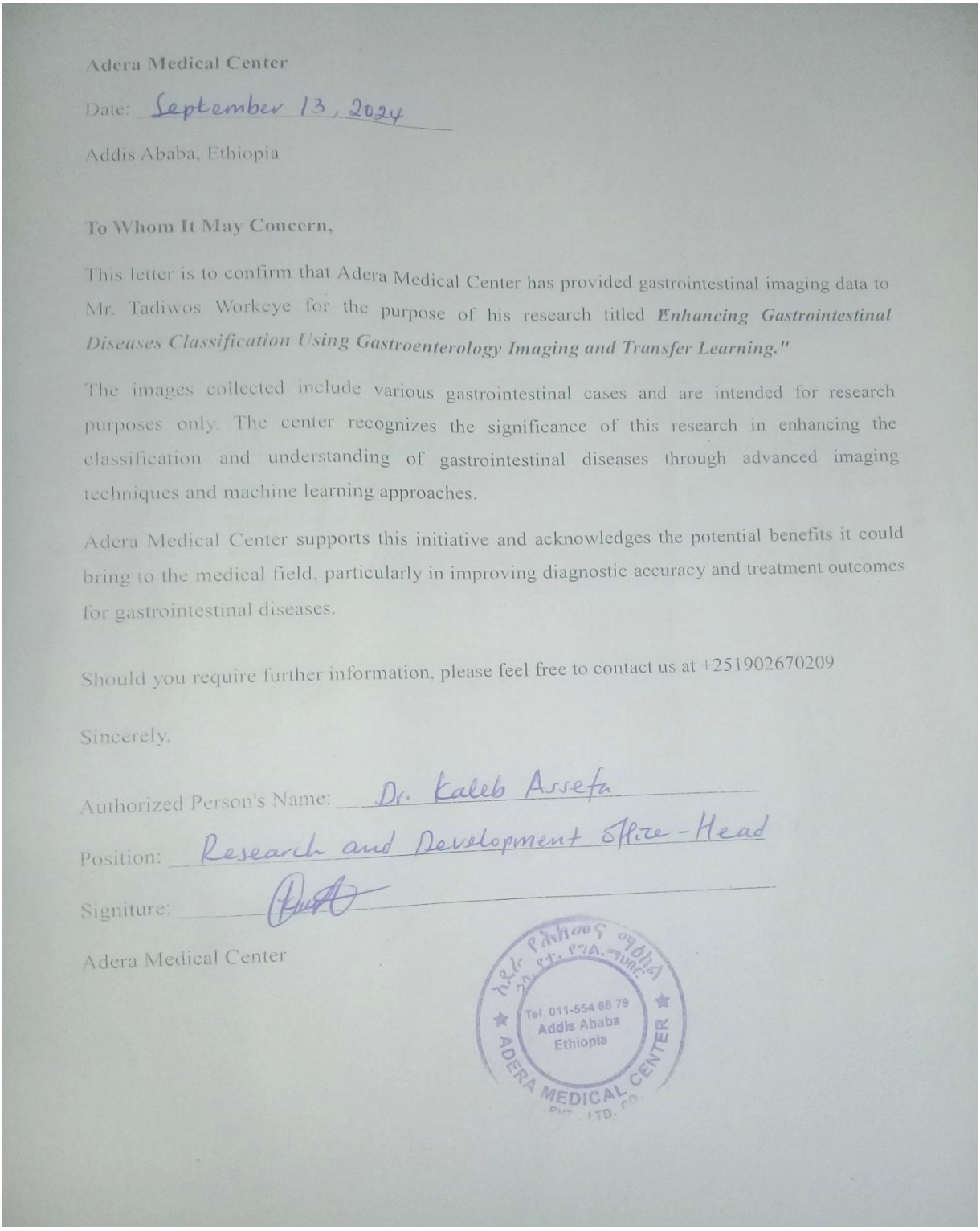
```
dataset_dir = 'C:\\Users\\model\\Documents\\Tadiwos Data\\Datasets\\'
full_dataset_local = datasets.ImageFolder(os.path.join(dataset_dir, 'Local'))
```

```
# Organize images by class
class_images = defaultdict(list)
for img_path, label in full_dataset.imgs:
    class_images[label].append(img_path)
```

```
# Define the root directory and subdirectories for the split data
root_dir = os.path.join(dataset_dir, 'split_data_local_60_30_10')
output_dirs = {
    'train': os.path.join(root_dir, 'Train'),
    'val': os.path.join(root_dir, 'Val'),
    'test': os.path.join(root_dir, 'Test')
}

# Create the root directory if it doesn't exist
if not os.path.exists(root_dir):
    os.makedirs(root_dir)

# Create the subdirectories for each phase (train, val, test)
for dir_name in output_dirs.values():
    if not os.path.exists(dir_name):
        os.makedirs(dir_name)

# Split and save images class-wise
for class_id, img_paths in class_images.items():
    total_images = len(img_paths)
    train_size = int(0.6 * total_images)
    val_size = int(0.3 * total_images)

    # Shuffle image paths for random splitting
    random.shuffle(img_paths)

    # Split the data
    train_paths = img_paths[:train_size]
    val_paths = img_paths[train_size:train_size + val_size]
    test_paths = img_paths[train_size + val_size:]

    class_dir = full_dataset.classes[class_id]

    # Copy images to their respective directories
    for phase, paths in zip(['train', 'val', 'test'], [train_paths, val_paths, test_paths]):
        class_phase_dir = os.path.join(output_dirs[phase], class_dir)
        if not os.path.exists(class_phase_dir):
            os.makedirs(class_phase_dir)
        for img_path in paths:
            shutil.copy(img_path, os.path.join(class_phase_dir, os.path.basename(img_path)))

print("Dataset splitting and saving completed.")
```

Appendix C: Confirmation Letter

Confirmation Letter from Adera Medical Center



Adera Medical Center

Date: September 13, 2024

Addis Ababa, Ethiopia

To Whom It May Concern,

This letter is to confirm that Adera Medical Center has provided gastrointestinal imaging data to Mr. Tadiwos Workeye for the purpose of his research titled ***Enhancing Gastrointestinal Diseases Classification Using Gastroenterology Imaging and Transfer Learning."***

The images collected include various gastrointestinal cases and are intended for research purposes only. The center recognizes the significance of this research in enhancing the classification and understanding of gastrointestinal diseases through advanced imaging techniques and machine learning approaches.

Adera Medical Center supports this initiative and acknowledges the potential benefits it could bring to the medical field, particularly in improving diagnostic accuracy and treatment outcomes for gastrointestinal diseases.

Should you require further information, please feel free to contact us at +251902670209

Sincerely,

Authorized Person's Name: Dr. Kaleb Assefa

Position: Research and Development Office-Head

Signiture:

Adera Medical Center

Tel. 011-554 68 79
Addis Ababa
Ethiopia
ADERA MEDICAL CENTER