# Colorectal Polyp Detection from Local Features Leveraging Deep CNN

by

Readhwana Reaz Adrin
19201034
Mahmuda Junainah
19201060
Antu Chowdhury
19201077
Niladri Saha
19201078

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
Summer 2023

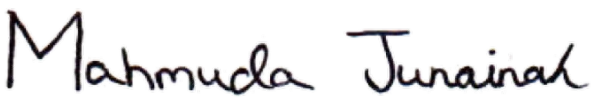
# Declaration

It is hereby declared that

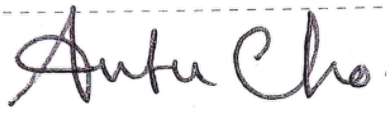
1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

**Student's Full Name & Signature:**

Readhwana Reaz Adrin
19201034



Mahmuda Junainah
19201060



Antu Chowdhury
19201077

Niladri Saha
19201078

# Approval

The thesis titled "Colorectal Polyp Detection from Local Features Leveraging Deep CNN" submitted by

1. Readhwana Reaz Adrin (19201034)
2. Mahmuda Junainah (19201060)
3. Antu Chowdhury (19201077)
4. Niladri Saha ( 19201078)

Of Summer, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on September 17, 2023.

**Examining Committee:**

Supervisor:
(Member)

Rafeed Rahman
Lecturer
Department of Computer Science and Engineering
BRAC University

Co-Supervisor:
(Member)

Dibyo Fabian Dofadar
Lecturer
Department of Computer Science and Engineering
BRAC University

Program Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

# Abstract

A colorectal polyp is a small cell cluster that develops on the lining of the colon. The majority of colon polyps are harmless, but some can develop into tumors later on and can cause colorectal cancer. Though the exact causes of polyp formation are not known, some factors contributing to polyp formation are smoking, excess alcohol consumption, obesity, lack of exercise, consumption of processed food, or having a family history of colorectal polyps. However, if the polyp is detected at an early stage, then it is possible to prevent the tumor or colorectal cancer. In this study, our aim will be to train a CNN model and evaluate its performance. Analysis of medical images using deep learning has been confirmed to be quite successful. CNN showed notably higher accuracy in the domain of medical image classification than traditional machine learning algorithms. CNN is a cutting-edge method of deep learning that is widely used in image or object recognition and classification.



**Keywords:** Colorectal Polyp; Colorectal cancer; Deep Learning; Convolutional Neural Network (CNN); Image recognition and classification.

# Acknowledgement

First and foremost, all praise belongs to the Great Almighty, without whom our thesis couldn't have been finished. Secondly, we would like to thank our supervisor, Mr. Rafeed Rahman, and co-supervisor, Mr. Dibyo Fabian Dofadar, for their helpful suggestions and assistance during our work. They were there for us whenever we needed them.
Finally, without our parents' ongoing support, it might not have been possible.

# Table of Contents

# Nomenclature

Several symbols & abbreviations that will be used later in the document's body are described in the following list.

$\phi$ Phi

$CAD$ Computer-aided Design

$CNN$ Convolutional Neural Networks

$CRF$ Conditional Random Fields

$f(x)$ Function

$HMM$ Hidden Markov model

$IOU$ Intersection Over Union

$max(0, x)$ Maximum between 0 and x

$RPN$ Region Proposal Network

$EDA$ Exploratory Data Analysis

$FN$ False Negative

$FP$ False Positive

$TN$ True Negative

$TP$ True Positive

# Chapter 1

# Introduction

## 1.1 Thoughts behind the Prediction Model

Colorectal polyp is the growth of irregular tissues on the inner membrane of the colon. Colorectal polyps have the possibility of turning into colorectal cancer over time if they are not diagnosed at the right time. In 2020, colorectal cancer impacted over 1.9 million individuals, making it the second most prevalent disease in women and the third most prevalent in men worldwide, according to data from the World Cancer Research Fund International (WCRF)[16]. Therefore, it is necessary to remove colorectal polyps from the moment they are initially detected in order to prevent colorectal cancer. There are various ways to detect colorectal polyps and among them, the most frequently used method is colonoscopy. Colonoscopy is a procedure in which a colonoscope, containing a camera, is inserted into the rectum to examine the interior of the colon. The effectiveness of this process depends on a few human elements such as the operator's dedication and attention to detail. Hence, this process is prone to errors and there are possibilities of failing to detect colorectal polyps during colonoscopy. In an attempt to reduce the chances of missing colorectal polyps during examination, in recent times, Convolutional Neural Network (CNN) is being used to identify and locate colorectal polyps.
Convolutional Neural Networks, also known as CNN, is a class of artificial neural networks commonly used in deep learning for object and image detection and categorization. Recently, CNN has demonstrated a key role in a range of activities and tasks, involving speech recognition in NLP, video processing, obstacle recognition for self-driving cars, and computer vision tasks like localization and segmentation. Now, this technique is also being applied in the medical field to detect colorectal polyps. Colorectal polyps can be detected in various ways using CNN; among them one of the ways is to apply CNN in colonoscopy videos in order to identify polyps. Other ways of applying CNN to identify colorectal polyps include joining two MASK R-CNNs using distinct backbone architectures. This process is considered to be one of the most advanced CNN methods to detect colorectal cancer. Following this approach, in the initial stage, the RPN supplants the selective search algorithm from R-CNN and Fast R-CNN and boosts the detection speed by combining all the content into a single network. Then it uses two parallel branches: Bounding Box Branch and Mask Branch. The bounding box branch is used for detection, and the mask branch is used for segmentation. In further research [7], this technique gets improved for more accuracy when the researchers train the CNN model on post-

learning methods: false positive and offline training. These post-learning methods boost the efficiency and performance of the model and it ultimately achieves more accuracy than before.
In this research, we have come up with the proposal to use different types of Convolutional Neural Network (CNN) based transfer learning models such as Inception-V3, Res2Net, ResNet-152, Resnet-50 and MobileNet-V2 on our selected dataset to determine a model that can identify the polyps in our dataset accurately. In order to compare the accuracy of the CNN based transfer learning models with the attention-based transfer learning model, we have also come up with the proposal to use Swin Transformer, which is an attention-based model. Additionally, we have proposed to ensemble these models and determine whether the combination of all these models can help us to get a better accuracy while detecting polyps using our dataset.

## 1.2 Problem statement

Colorectal cancer is considered a critical public health issue that can be caused by the growth of colorectal polyps if it is not detected on time. According to 'Cancer Today', in terms of prevalence and death, colorectal cancer has risen to the third place among all cancer types in the year 2020 [15]. Since polyps can be considered an early stage of causing colorectal cancer, the early detection of it is supposed to increase the survival rate of the patients. Colonoscopy is the most popular approach for detecting colorectal polyps among the several techniques available. This procedure's efficacy is reliant on a few human factors, such as the endoscopists' commitment and thoroughness. As a result, there is a chance that the procedure will go wrong and colorectal polyps will not be found during the examination. However, the progression of CAD technologies to facilitate endoscopists may be a viable approach to the situation's rehabilitation. CNN has been used to diagnose polyps in recent years in an effort to decrease the likelihood of overlooking them throughout evaluation. Many researchers have tried many methods to create a colorectal polyp detection model without any drawbacks. But it is a super rare case to find such research work without any drawbacks, especially when the topic is a crucial matter like colorectal cancer. In a number of research works, polyps are identified without localization which indicates that the model is unable to identify the polyp's precise position. Again, some research is unable to handle the rate of false positive and false negative outputs. Furthermore, some research works perform way too well theoretically, but practically, these fail to execute because they are not capable of processing at 25 FPS (Frame Per Second). Additionally, some research works utilized datasets of carefully chosen images, typically of greater quality than the frames obtained in typical colonoscopy recordings. Such biased shortlisting may lead to performance outcomes that are overstated yet not feasible in actual use. However, in this situation, it seems like it is nearly impossible to create a model without any drawbacks, so it is necessary to figure out how each model performs on a particular dataset. Based on the performance of these models, researchers will be able to determine which model is the best for detecting polyps accurately. In this thesis work, we aim to determine the type of transfer learning model that can detect polyps accurately from our dataset. By comparing the CNN based transfer learning models, the attention-based model, and ensembling all the transfer learning models, we aim to determine a model that can detect polyps accurately.

## 1.3 Aims and Objectives

The objective of our research is to determine the type of machine learning model that can detect polyps accurately from our dataset. This research also aims to help researchers choose a machine learning algorithm to detect polyps accurately by providing evaluation metrics across the trained machine learning models.

# Chapter 2

# Literature Review

Many studies have been conducted in this domain and in the following papers, researchers have found that -

In paper [5], the authors discussed the miss-detection rate of colonoscopy of 25%, which can lead to a late detection of polyp, dropping the survival rate of the patient to 10% due to colorectal cancer. They also explained the necessity of computer-aided detection (CAD) and described several types of deep convolutional neural network (CNN) structures, including R-CNN, Fast R-CNN and Faster R-CNN. They expressed their contribution in this paper [5], dividing it into four different segments. Firstly, they claimed it to be the first paper to detect polyps using the region-based object detection technique. Secondly, they applied several augmentation strategies to evaluate the dataset such as scaling, flipping, brightening, blurring etc. Thirdly, they suggested two post-training approaches: false positive training and offline training, and finally, they claimed that in the colonoscopy image dataset, their diagnosis model outperformed previous CNN-based research in terms of detection accuracy. For this research, they used publicly accessible polyp-frame datasets from several sources which include CVC-CLINIC, ETIS-LARIB, two datasets of colonoscopy videos, the ASU-Mayo Clinic Colonoscopy Video dataset and the CVC-ClinicVideoDB dataset. Images were extracted from colonoscopy recordings showing polyps in a variety of sizes, shapes, and colors using the CVC-CLINIC and ETIS-LARIB datasets. To assess the accuracy of the polyp diagnosis in colonoscopy clips, the additional datasets were utilized. In order to execute the model, they used seven different methods such as image augmentation, region proposal method, fast R-CNN detector, pre-trained deep CNN for image transfer learning (Intercept ResNet), training detector, false positive training and offline training for video detection. At first, they augmented the input images and passed them through a deep CNN model that had been trained beforehand, which uses RPN depending on the polyp classification and regional regression. Then, after resizing the data proposed by the RPN, the detector generates an output using Fast R-CNN. After that, the output goes through the post-learning methods (false positive and offline learning) and repeats the entire process to achieve more accuracy. They concluded the result based on four parameters: FP, FN, TP and TN. Depending on these four criteria, they calculated the performance metrics—precision, recall and specificity—which made them claim this model to be the most accurate and precise model for polyp detection. However, the model has only one limitation which is that the processing time of polyp detection for each frame is about 0.39 seconds which is more than

average. If real-time detection is necessary like in a normal colonoscopy, this can be a drawback of the model.

In paper [7], the authors used the robust object identification neural network Mask R-CNN. In this paper, they combined two Mask R-CNNs with various backbone configurations such as ResNet50 and ResNet101 in order to boost efficiency. The authors' contribution can be divided into three segments. Firstly, the claim that the process of segmenting polyps using Mask R-CNN is being done for the first time in this research. Secondly, they combined two Mask R-CNNs with various backbone configurations in order to boost efficiency. Finally, they show that their suggested strategy performs better than cutting-edge methods. They collected their dataset from the MICCAI 2015 polyp detection challenge. For the following research, they used publicly accessible polyp-frame datasets from several sources which include CVC-ClinicDB, ETIS-Larib and CVC-ColonDB. They utilized the CVC-ClinicDB dataset as a training set for the model and the other two datasets as testing sets. In order to execute the model, they divided the method into three different components: augmentation of data, two Mask R-CNNs with various backbone configurations and the bitwise combination of the Mask CNNs. There are two different phases to the Mask R-CNN structure. The initial phase is RPN which supplants the selective search algorithm from R-CNN and Fast R-CNN and boosts the detection speed by combining all the content into a single network. Bounding Box Branch and Mask Branch are two parallel branches found in the other phase. For detection, the bounding box branch is utilized, and for segmentation, the mask branch. On the basis of the performance metrics, they claimed that their test results outperformed cutting-edge techniques for polyp detection. However, the research has some drawbacks as well. The tests were done on limited datasets. They also did not train the model on FP datasets which can reduce the accuracy rate of the model.

Deep learning architectures are utilized in this work [6] to classify breast cancer using histological pictures. Convolutional Neural Networks are the foundation for the purpose of picture categorization among the existing techniques. There are several people researching in different divisions and creating this successful paper. Like Tomas's applied capsule net architecture obtained 87% accuracy using 400 images for this classification problem. Carlos's pre-trained Inseeption Resnet V2 achieved 76% which is based on CNN feature extraction and vector machines. Rakhlim used deep neural networks and gained 87.2% as classification of breast cancer by benign and malignant. Alexander Brook automates the disease from microscopic biopsy images and Swapna introduced CNN-LSTM which is a machine learning model. They proposed a three-layer architecture to classify breast cancer. The histology pictures are supplied into the capsule net architecture in the suggested approach. The three layers are the input, hidden and classification (output) layers. From the BATCH 18 grand challenges, they collected the database. Out of 400 images, they only take 285 images of different types of cancer. These data have undergone stain normalization and patch extraction as pre-processing processes. After their experiment, the result was pretty good. There are 5 tables shown in this paper [6]. With Deep CNN, they get the highest accuracy among all others. It is evident from our study that data pre-processing and parameter adjustment can boost the performance of traditional designs.

In this study [1], they provide a modified CNN network for classifying lung

pixels. Instead of manually defining a set of features to perform classification and extract discriminant characteristics from the training data, they developed a fully autonomous neural-based machine learning system. Their approach isn't problem-specific, so it can be simply used in any other imaging domain. Better classification results should be anticipated because CNN utilizes supervised learning algorithms as opposed to unsupervised RBM neural networks. In this division [1], for picture databases, CNN dramatically enhanced the best performance. MNIST, CIFAR10 and NORB databases are used here. The normalized lung imaging patch with a mean of zero and a variance of one is the input for the network architecture approach. The first layer is a convolutional neural network with a total of sixteen output channels and a kernel size of 7 x 7 pixels. The layer going to follow that is situated at the pooling layer with a kernel size of 22. The next 3 layers are made up of 100–50–5 neurons apiece and are completely linked neural layers. In ILD lung image patch classification, networks with only one convolution kernel outclass networks with multiple convolutional layers. Feature Extraction for Images Convolutional neural networks may be used to directly input picture pixels to conventional neural networks that feed information forward to solve photo classification issues. The learning model is significantly simplified by CNN models, which integrate considerably smaller kernel filters using weights. Additionally, CNN networks outperform traditional fully connected neural networks in terms of speed and dependability. The foundation of CNN is a layer of convolutional neurons. Visual classification systems that use one or more two-dimensional matrices or channels as input to the convolutional layer generate multiple two-dimensional matrices. There could be a discrepancy in the number of both input and output matrices. For CNN to reduce feature dimensions, the pooling layer is crucial. By altering the convolution output matrices, pooling algorithms should be employed to combine the neighboring components in order to reduce the amount of outputs in the CNN. This CNN model's convolutional layer makes use of the ReLU activation function. The network converges far more quickly when using the ReLU activation function than when using the sigmoid activation function, according to test findings. They process 128 input samples in a batch before changing the connection weights once for the whole batch, as opposed to updating them after each back propagation. In order to increase performance, the drop-out method randomly disables neurons from every layer in terms of training [1]. Each training cycle starts with a random initialization of a drop-out map that marks the on or off state of the related neuron and has the same size as the neurons in each layer. In this experiment [1], the drop-out rate of that training was set at 0.6, and the output of the activation of the testing neuron was multiplied by 0.4. They have created their own neural network framework, which can be used to train a variety of neural systems, including CNN, RBM, autoencoders and more, as opposed to utilizing the open-source CNN implementation that is currently available. The assessment makes use of the ILD database which is open to the public [1].

There are 113 sets of HRCT pictures in the collection, and 2062 2D regions of interest (ROI) have been identified using the ILD classification. Their classification outcomes were compared to those of three additional feature extraction techniques: (1) Rotation-invariant where three resolutions are available for the LBP feature, (2) the SIFT key-point feature at the patch center and (3) RBM is used for unsupervised feature learning [1]. SVM classification models are combined with all three of the studied methodologies. To categorize ILD pattern patches in HRCT lung

image patches, they have presented a modified CNN architecture. Their method, which only uses one convolution layer, uses training data to efficiently learn DCT-like patterns and generates precise classification results. Results demonstrate that their method outperforms benchmarks and can automatically extract discriminative features without manual feature design.

In paper [2], the authors introduced an effective way to detect colorectal polyps using CNN and automatic polyp detection. There are a number of ways to detect and remove colorectal polyps. One of the most common and effective ways to remove colorectal polyps is colonoscopy. Colonoscopy is the procedure in which a colonoscope is passed into the rectum or colon and a camera is attached to the colonoscope in order to see the membrane of the colon. The process of colonoscopy is effective in detecting colorectal polyps; however, this process is operator-dependent and depends on the observation skills of the operator [2]. Apart from colonoscopy, another efficient method to detect colorectal polyps is automatic polyp detection using convolutional neural networks (CNN). According to a 2015 study [2], colorectal polyps can be found using automatic polyp detection by first inserting an image of the polyp into the system, then converting that image into an edge map, then refining that edge map by deleting the non-polyp border pixels. After removing the non-polyp boundaries, a refined edge map is obtained and this refined edge map is passed into a voting program, which determines the location of a polyp and then a voting map is generated by the program. The voting map generated contains the location of the polyp. Once the polyp's position has been identified, three kinds of multi-scale patches are found and collected. The CNN then produces a probability score for every input group and the highest score for every group after processing the groups of the patches. At the end of this process [2], the maximum scores of all three sets are combined and a mean is calculated in order to compute the final probabilistic output for the polyp.

The main contribution of this research [2], is to combine different features of colorectal polyps into one polyp detection system so that it becomes easier to detect a colorectal polyp from its initial stage. Another contribution of this research is that it introduced a system through which polyps can be detected from colonoscopic videos using CNN. The dataset used in this research is 40 short colonoscopic videos of which 7000 frames contained images of polyps and 28,000 frames did not contain any images of polyps. This data set was divided into training and testing data sets in order to train and test the system. The findings of this study demonstrated that this method of detecting polyps, utilizing automatic polyp detection and CNN, can lower the latency and quantity of false positives. However, this technique's inability to determine the time it took to find the first colorectal polyp is a research limitation.

In paper [10], the authors discussed a common problem faced during colonoscopy operations and detecting polyps. The problem faced during colonoscopy operation is that the CNN based computer-aided system cannot always detect colorectal polyps during colonoscopy as most of the time the images obtained from colonoscopy are blurry and the system cannot distinguish between the actual polyp and the non-polyp-like structures. Another problem faced is that there is a difference between the images of actual colonoscopy and the images available in the public datasets; hence, the system cannot always detect polyps that have a different shape, color or texture than the regular colorectal polyps available in the public datasets. The authors

propose a new polyp detection system using a blur detection module, two CNN detectors, and an ensemble module to address this issue. The two CNN detectors (AFP-Net and U-Net with dilation convolution detector) detect colorectal polyps, and the ensemble combines the results obtained from the two CNN detectors to produce a new image showing the location of the polyp. The pipeline's processing time is reduced by the blurry detection module, which filters the unclear colonoscopy pictures. The authors' contributions to this paper include creating a private dataset of 7,313 images from 224 full colonoscopies [10] for the purposes of training the model and assessing the system's performance in use, presenting a system architecture that can aid in addressing polyp detection and colonoscopy issues, and training and testing their suggested model on colonoscopy videos. In this research [10], they used three different types of datasets, which are the blurry detection dataset, the public dataset and the private dataset. The public dataset, which also includes the GIANA (Gastrointestinal Image Analysis) 2018 and ETIS-Larib Polyp DB datasets, is used to train and test the system; the blurry detection data set is used to train and test the blurry detection module; and a potent model that may be applied during a colonoscopy is developed using the private dataset.

In their experiment [10], the blurry detection used by the authors is a CNN based binary classifier and they have also used different backbones in this experiment in order to check the sensitivity, specificity, and running time and then they compared the blurry detection module with the backbones. They used the pre-trained SqueezeNet as the model's backbone during its training phase and fine-tuned it on the training set. After carrying out the experiment, it was found that the SqueezeNet backbone was more accurate and less time-consuming compared to the other backbones (Resnet-50 and VGG16), as the SqueezeNet backbone achieved more than 95% on both sensitivity and specificity and the processing time of SqueezeNet for each frame was 3 ms.

In this experiment [10], the AFP-Net and U-Net with dilation convolution detectors are used in this model as CNN detectors, as the AFP-Net detector helps to detect the colorectal polyps, localizes them and can detect polyps on different scales, whereas the U-Net with dilation convolution detector is used to maintain the resolution of the image, enlarge the receptive field and produce a one-dimensional segmentation map. In the last step, the ensemble combines the results obtained from the two CNN detectors and produces a new image showing the location of the polyp. The findings of the experiment using the publicly available datasets showed that there is a considerable discrepancy between the colonoscopy pictures from real medical operations and the publicly available datasets; as a result, the system needs to be trained using the privately available datasets. The findings revealed that the model can detect polyps reliably and that the frequency of false positives decreased after training the model using the private data set. Hence, this proved that this model can be used in real-world clinics during colonoscopy, as this system can detect colorectal polyps accurately in real time. However, the limitation of this research [10] is that this system cannot completely eliminate the number of false positives detected during the colonoscopy.

Medical and computer science collaboration is not playing a vital role in the case diagnosis of some significant medical issues. This paper [4], focused on the development of an automated method for categorizing polyps as malignant or benign

in endoscopic images. They did, however, train a CNN model with a fully linked layer for polyp classification. During the task, their main challenge was to find the data on the medical image for the classification tasks. We can also observe from the research that lighting has a significant impact on picture categorization such as with low and high illumination, differing classification results can be yielded. To overcome this problem, they found a standardized lighting condition that has shown the highest accuracy. They selected the region of interest pooling (ROI) for each polyp; for this, the polyp image was taken from the publicly available videos. To have better results, they resized the image by $128 \times 128 \times 3$ and normalized it. They identified them manually using Kudo's pit pattern categorization. To have computational efficiency, they used a pooling technique called max pooling to consider the maximum number of output values.

This paper's [4] main contribution is that CNN was fully trained using the extended endoscopic image data set, and they used the transfer learning approach to learn low-level CNN features from a nonmedical database. To see the output, they used CNN layers to extract low-level features. The Eqn. 2.1 is -

$$h_{i,j} = f\left(\sum_{m}^{M}\sum_{n}^{N} w_{m,n}x_{i+m,j+n} + b_{m,n}\right) \tag{2.1}$$

To take the hidden layer, they took the higher dimensional ReLU activation feature in the Eqn.2.2

$$f(x) = max(0, x) \tag{2.2}$$

They also used the max pooling technique to get the maximum output of the neuron. This paper [4], consists of a CNN with a whole linked network for polyp categorization. They made the image RGB to have better output and used the max pool to reduce the dimension of the feature maps. After passing through the 64 and 128 filters, the output was passed through the maxpool3 with the filters equal to 128 filters where every neuron is connected to each other.

The result of this paper [4] is significant; they automated the data using flipping, rotation, and translation operations. They made the dataset contain 31525 images and the classification contains 15875 images. To deal with the disappearing gradient, they employed ReLU. The last FCN is a softmax layer that converts the probability to classify the specific situation. In this research [4], vector encoding proved to be effective for adopting the labeling classification. During this kind of research, gradient desensitization was one of the common problems. To solve this, they used the Adam optimizer where weights were initialized as normal variables. While doing the experiment in Intel i5, the results showed 87% accuracy.

In this research [4], they used the ROI technique whose main advantage is that we may utilize the same feature map for all suggestions, allowing us to transmit the full image to CNN rather than passing each proposal separately. In other ways, this technique has problems as well. It can cause performance issues in terms of processing speed; it is suboptimal and it's not possible to do end-to-end training during the system run. They used max pooling to see the outcome of this research [4] and the main disadvantage of max pooling is that the pooling operator only considers the largest element from the pooling region and disregards smaller components. If

the bulk of the components in the pooling region are of high magnitude, the distinguishing traits vanish after the maximum pooling process. However, it may also be a useful tool when max pooling is employed to facilitate over-fitting by providing an abstracted form of the representation. It also reduces processing costs by reducing the number of parameters to learn and provides the internal representation with basic translation invariance. They used ReLU to deal with the decreasing gradient. ReLU is an essential tool in deep learning, but it has problems as well. The ReLU has some limitations. One of the main disadvantages of ReLU is that heavily weighted updates can indicate that the sum input of the activation function is always negative, no matter what the network input is. In other words, it offers several advantages. The key advantage of using the ReLU function over other activation functions is that it does not simultaneously excite all neurons.

Medical image analysis has become a famous trend in past years in computer science. In this paper [3], we will see the colonoscopy polyp detection using the CAD algorithm. From the available video sources, they collected the polyp images and generated results. The CRF model was utilized as a classification technique. The CNN-derived features outperform the eigenimage model in terms of invariance to viewing angles and image quality factors. As previously stated, the CNN features include sent into the CRF classifier. For this study, they used data from human and mouse colonoscopies [3].

During the study [3], they utilized a quality evaluation method that used the hidden Markov model (HMM) technique to filter out low-quality footage. They employed deinterlacing and noise reduction filters to remove noise. They employed several various angle photos for polyps, lumen, vasculature, diverticula, feces, blood, and normal mucosa in this paper[3]. Their convolutional neural network is made up of three convolutional parts that alternate with two subsampling layers. The final layer is a connected layer; a backpropagation algorithm is used to train the network. In each of the layers, we have a different feature map and the role of the input image grouping. There are filters and the filters changed their characteristics according to the weights and activation function according to eqn 2.3.

$$y = \phi\left(\sum_{i=1}^{I}\sum_{j=1}^{J} w_{ij}x_{ij} + c\right) \tag{2.3}$$

A bigger variety of filters can produce better results, but it also increases the risk of overtraining and limits generalization accuracy. They employed a conditional random classifier (CRF) for spatial and temporal association in colonoscopy footage in this work.

During this study [3], they employed the CRF model's Markov property to address the spatial and temporal correlations inside the frames in order to perform CRF classification. Because it doesn't require that anymore, the input data can be conditionally independent and the CRF model has become robust to image fluctuations. Using some parameters, we were able to access the probability that an image could contain a classifier.

As a consequence, basic $3 \times 3$ midpoint filters were employed to remove noise from the dataset, and the image distribution was normalized by subtracting the mean and dividing by the standard variation. Following CNN training, the feature was

adopted from CNN and employed CRF categorization. The polyp was tested in low and high light, and the edge detection findings from the original photographs were utilized to fill the screening window. The edge data was used to identify acceptable screening windows, and then polyp detection was accomplished using classification.

This research [3] used the HMM technique. This technique has both advantages and disadvantages. HMM offers a solid statistical basis and rapid learning methods, allowing learning to occur directly from raw sequence data. This is the main advantage of this technique. On the contrary, in most cases, Markov analysis is poor at describing events; therefore, it cannot represent the true model of the underlying situation. Another important tool they used is CRF classification. The essential components of CRF are the feature functions, which produce the real value. The polyp lighting condition has a significant impact on the research results, so setting up the proper lighting condition can be a challenging task.

Another way to detect colorectal polyps is to use the image pre-processing technique and enhanced faster R-CNN. In this paper [12], the authors used an image pre-processing technique and an enhanced faster R-CNN technique to detect colorectal polyps. The objective of this study was to enhance the detection accuracy of colorectal polyps. In this study [12], the authors obtained the colonoscopy images from the colonoscopy image database which is available on the Internet. The names of the colonoscopy image databases are CVC-ClinicDB, CVC-ColonDB and ETIS-LaribPolypDB [12]. After obtaining the colonoscopy images, the authors first preprocessed the images in order to remove specular reflections present in the colonoscopy images [12]. Specular reflections in the colonoscopy images are caused due to the bubbles present in the colon during colonoscopy and these bubbles reflect light when the colposcopic probe casts light on them. This causes specular reflection and results in false polyp detection. Hence, it is necessary to remove these specular reflections in order to detect polyps accurately. For the purpose of this study [12], the authors took help from the medical professionals and identified the position of the specular reflections using an empirical procedure and decided to use the HSV color space for the pre-processing of the colonoscopy images. Instead of using the RGB and YUV color models, the HSV color space/model is used in this study [12], as this color space corresponds well with the way humans see colors. In the beginning of image preprocessing, the color space of the images was changed from RGB/YUV to HSV. After changing the color space of the images, the specular reflections were removed from the image by calculating a threshold value of the S-space and V-space of the image [12]. The threshold values were calculated using a trial-and-error method and after performing the thresholding operation on the images, a few dispersed dots were visible on the images. These dispersed dots were removed by performing a dilation operation on the images and patching the images. In this study [12], in order to remove specular reflections, the authors used the image patching algorithm suggested by Telea [12]. A training dataset was also created by the authors by applying a number of different techniques to the images obtained from the publicly available datasets. To enlarge the sample size of the training data, various approaches were used on the images. These techniques included rotation, translation, scaling, flipping, tailoring, and adding Gauss noise. After applying the aforementioned techniques, the authors labeled the data.

After the images have been preprocessed, the authors select a feature extraction

network for their model, design the anchor for the RPN network and design the activation and loss functions of the model. The feature extraction network selected for the model of this study [12] is VGGNet as it has a lesser number of convolutional layers and this network is referred to as VGG16 in this study as it has 16 weight layers. To detect various sizes of polyps, the authors fixed the magnitude of the anchor of the RPN network to 4, 8, 16, and 32 and modified and designed the VGG16 feature extraction network to produce a feature map. They also designed the activation function of this model in accordance with the convolutional layer, the classification and regression layers in the feature extraction network, and the loss function for this model was divided into the classification loss function and the bounding loss function.

The experimental model was built by the authors in the Caffe framework [12] with an initial learning rate of 0.001 and a progressive rise in learning rate. As their chosen dataset was small and inadequate to train the model, the authors of this study [12] used the transfer learning method. The training segment of the model was developed by the authors using numerous datasets from ImageNet, followed by the separation of the relating full connection layer and use of the residual feature extraction network, and eventually the back propagation to fine-tune the pre-training model in accordance with the properties of their data set. In this study [12], the polyp was located, categorized into a category, and then a threshold value of Intersection Over Union (IOU) was calculated to see if the polyp was a positive sample. A mean average precision was also computed in order to assess the performance of this model's algorithm against algorithms from other models. The results of this study [12] show that a new polyp detection model using faster R-CNN has a detection rate of 91% , hence this model can identify polyps satisfactorily. However, the limitation of this study is that the algorithm used in this model cannot automatically detect certain features of the polyp and categorize it into adenomatous polyps and non-adenomatous polyps. Additionally, the pre-processing technique of this model can be improved, and the robustness of this model has not been determined.

In paper [8], the authors described some techniques for polyp detection and localization automatically in colonoscopy videos. Polyps are found and located using single-frame object detection or a segmentation network such as U-Net. Here, they are using CNN for more accuracy. They have adopted CNN and trained it instead of using CAD to detect polyps. Notice that the vast majority of existing CNN algorithms identify the polyp on a frame-by-frame basis. Jittering can occur due to minor intensity fluctuations between frames. The changing sight of polyps across video frames is a major hurdle while training a CNN network to identify polyps efficiently. However, their findings show that the polyp characteristics for each sufferer are relatively stable, with little movement shift between successive frames. They have used three types of methods to train their algorithm and four types of sub-methods. All together, it is detecting polyps with high accuracy. For the first method, they adopted U-NET for identifying polyps. It is called a single-frame polyp detector. This method is known as an end-to-end CNN, so it can use an image as input and generate a segmentation strategy for the image's object of interest. The single-frame structure has been widely utilized and successfully implemented in several CAD systems.

To gain a better understanding of single-frame detectors, they first utilized erosion

to flatten the spur on the separation borders. After that, they estimate important components and individually order them by dimensions and the ratio of surface area. If the excellent and greatest ellipse-like shape regions share the same element, it is centrally calculated and used as a new seed called the polyp center. The second method is a polyp tracker. It has four sub-methods. The initial submethod, the pattern of moving objects across two consecutive frames of video, is known as optical flow. The pixel value of an item remains constant between frames. The number of pixels moves in the same direction. A fuzzy picture or visual artifact, on the other hand, may cause this approach to fail, resulting in a premature halt to optical flow. The motion regression model is intended to calculate migration for a polyp center depending on its motions in image sequences. Seeing that any new site is within the video frame, a CNN model must be used to assess whether it is a polyp center for future monitoring or not, after which tracking will be stopped. The third submethod provides an on-the-fly trained CNN-based structure for distinguishing polyps from origins and deciding whether or not to cease the traceability proposed by the regression model after they realized that the presence of a polyp continues to remain compatible among films. For the fourth sub-method, it is tracking stop criteria because the same polyp can be traced by many seeds obtained from different photos. They stop tracking after more than ten frames to decrease calculation time and avoid unnecessary monitoring. They created a spatial voting technique in the third method to minimize outliers based on the assumption that the proper polyp center will gather inside a tiny space. The closely coupled product was then estimated, and the ultimate polyp center was found by utilizing the largest product's center. They used various data sets in this examination. Most importantly, the data used in this work is made up of 18 separate clip serials. In every clip, at least one polyp can be seen throughout the series. In the end, the results were good for them, as their proposed algorithm has an efficiency of 97% in recall for polyp detection. Moreover, they get up to 96% for polyp localization.

# Chapter 3

# Methodology

## 3.1 Flowchart

Our proposal for detecting polyps completely depends on our pre-trained transfer learning model and attention-based classifier. In the transfer learning model, we use 5 models and then ensemble them to get an accurate result to detect the polyp. The CNN transfer learning model has multiple types, but in our proposed deep-trained transfer learning model, we will use 5 models: ResNet152, InceptionV3, MobileNetV2, ResNet50, and Res2Net, which can give us a higher accuracy result by using the image datasets more and more. Also, we are doing ensemble in the transfer learning part to get the mean accuracy of detection. Moreover, in another section, we are using an attention-based classifier model named Swin Transformer. We compare this model with the ensemble and transfer learning models so that we can see the better accuracy and results.

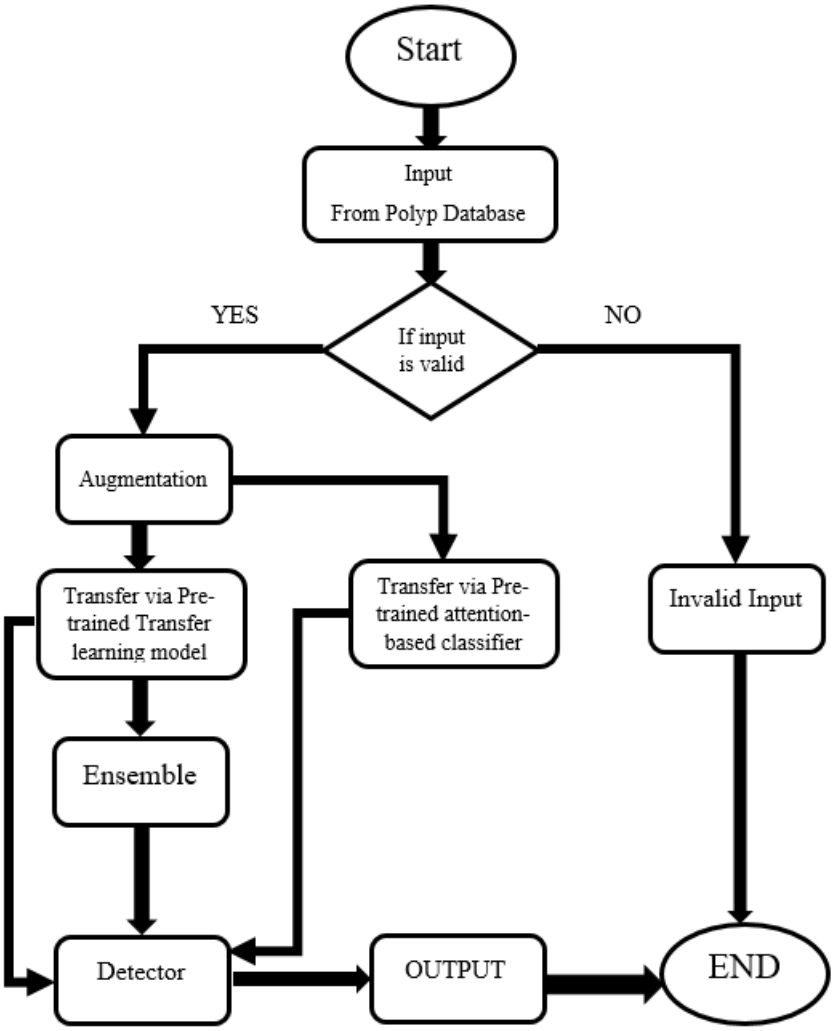



Figure 3.1: Flowchart

## 3.2 Workflow

First of all, input images from the polyp database, which must be valid images. If this is not a valid image, the program will end. After checking the validity of the images, the images get augmented. Augmentation is a method where the database images will increase in number on the basis of different positions, angles, and color combinations. After augmentation, it will go into two different algorithms. The first algorithm is going into our pre-trained transfer learning model algorithm which is based on 5 transfer learning models: ResNet50, InceptionV3, MobileNetV2, ReS2Net, and ResNet152 for image classification. Then the models will go through the ensemble part. A detector will detect the polyps. Using the ensemble, the algorithm will detect the polyp with the highest accuracy, show the output and end

the program. Moreover, from augmentation, there will be another algorithm which is transfer via a pre-trained attention-based classifier. We used the Swin Transformer algorithm in the attention-based classifier. After the execution of the program, the algorithm will go to the detector and show the output and accuracy. We are running our programs for 30 epochs.

## 3.3 Description of the model

### 3.3.1 Transfer Learning

For our thesis, we used a transfer learning method in order to classify the images of our dataset. In the machine learning method called transfer learning, a pre-trained model is utilized as the basis for a new prototype. A model that has been trained for one work is then used for another, similar work as an optimization to facilitate quick modeling development for the second work. One can accomplish a new activity with considerably improved performance by using transfer learning as opposed to training with a model from scratch with the enormous resources needed to train the deep CNN [14]. Transfer learning is frequently carried out using deep learning models that have been pre-trained for a significant and difficult image categorization activity. Transfer learning can be used to retrain the last layer of an existing model, greatly decreasing the time and data used for training. With an initial learning rate of 1e-5, the Adam optimizer is used for optimization. In this paper, we used the pre-trained Inception-v3 architecture, ResNet-152 architecture, Res2Net architecture, MobileNet V2 architecture and ResNet50 architecture. For all the architectures, the tail portion of the network is removed and given a new classifier, such as Softmax, to suit the requirements of the new task in order to transfer the highly learned features from the models such as InceptionV3, to the new image classification task. With the exception of the final few layers, the structure of the freshly created transfer network is identical to that of the pre-trained model. While training the newly built structure, the weights in the transferred layers are kept fixed. As a result, the previously acquired information can be applied to the assigned task[11].

### 3.3.2 CNN Structure

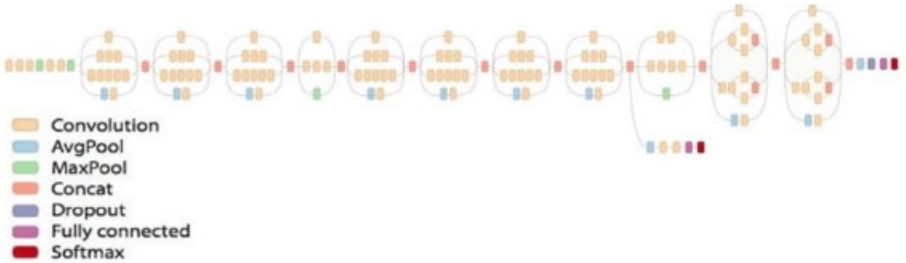



Figure 3.2: CNN Structure

Convolutional neural networks are multi-layer artificial neural networks that combine feature extraction and classification. They also accept multiple raw images as input and output classification. With picture input of depth, height, and breadth, the neurons inside a layer are arranged in 3 dimensions. Whereas the final layer, generates class scores by shrinking the whole image to a vector and each area of its preceding layer is related to every area of that layer. With the use of our collected dataset, the CNN transfer learning models were trained up in this thesis [9].

### 3.3.3 ResNet-152 Architecture

ResNet-152 is a variant of the ResNet architecture, and it has 152 layers in its architecture. The architecture of ResNet-152 consists of a number of residual blocks, each of which has several convolutional layers and skip connections. The skip connections present in the architecture of ResNet-152 allow the model to efficiently learn the residual mappings, which helps the model train deep networks without facing the degradation problem. The skip connections present in the architecture of ResNet-152 also help the model handle the vanishing gradient problem [18].
ResNet-152 uses a model that has already been trained on a large dataset, such as ImageNet, and fine-tunes it on a smaller dataset. This method of pre-training ResNet-152 on ImageNet helps ResNet-152 recognize complex patterns and representations in images and other types of data, and the model can avoid having to start over on the target dataset by using the pre-trained weights. This technique of pre-training ResNet-152 on ImageNet also helps to reduce the training time and the usage of computational resources.

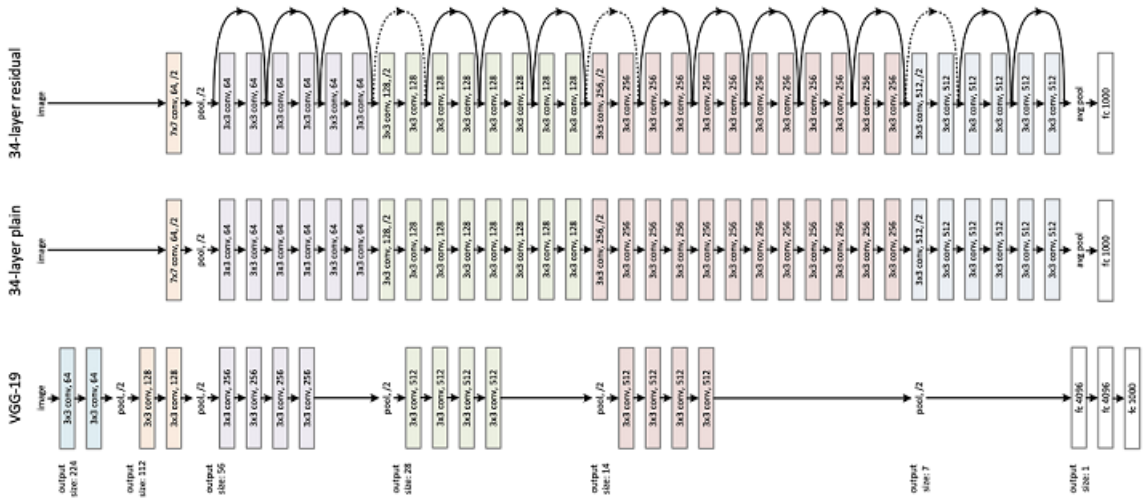



Figure 3.3: ResNet-152 Structure

### 3.3.4 InceptionV3 Architecture

The Inception V3 deep learning model, which is used to categorize images, is built on convolutional neural networks.It is a model that Google has already trained using more than 1.4 million photos and over 1000 different classes on ImageNet. CNN is

used in the Inception V3 model, an image recognition algorithm, to extract features from images. The proposed Inception V3 model is made up of the Fully Connected Layer, AvgPool, MaxPool, Concat Layer, Convolution, Softmax Function, and Dropout. By sharing weights and having sparse connections, the convolution layer is characterized as it calculates the neuronal output that is linked to the local regions that are now present in the previous layer. Furthermore, in the exact same layer, it correlates to kernels, and under each particular feature map, it shares the weights of the neurons. Then, in the classification section, a fully linked layer uses the outputs of both the custom-generated segmented feature and the Inception-V3 feature. [9]

### 3.3.5 MobileNet-V2 Architecture

MobileNetV2 is an optimized and resource-efficient deep learning architecture specifically tailored for image recognition tasks on mobile-embedded devices. It represents a significant advancement over MobileNetV1, incorporating novel techniques to improve its computational efficiency while maintaining high accuracy. Notably, the network employs inverted residuals and linear bottlenecks, effectively reducing compassion while preserving critical feature representation. Inverted residuals utilize a distinctive bottleneck structure, initiating with a limited number of channels and gradually increasing them in the bottleneck layers. This approach contrasts with conventional residual blocks. Depth-wise separable convolutions are extensively employed, splitting the convolution operation into depth-wise and point-wise convolutions, thereby significantly reducing computational complexity. Additionally, the model offers flexibility through the wide multiplier hyperparameter, and expansion factor, enabling users to fine-tune the trade-off between model size and performance. The result is an exceptionally efficient architecture that strikes an optimal balance between accuracy and resource utilization, rendering it eminently suitable for real-time applications on resource-constrained devices.

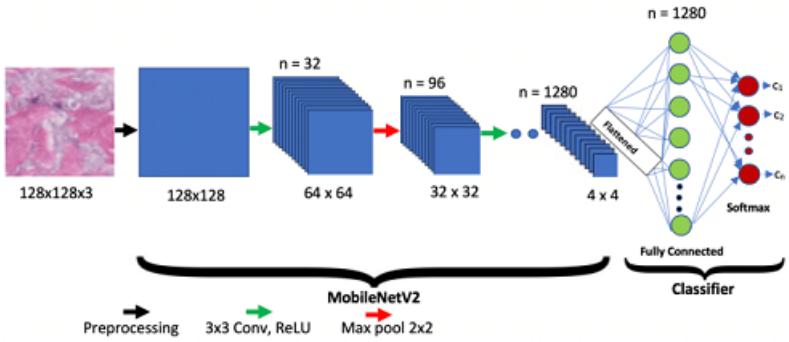



Figure 3.4: MobileNet V2

### 3.3.6 Res2Net Architecture

Res2Net is a deep learning model architecture designed to enhance the representation power of CNN by effectively capturing multi-scale features. The concept of Res2Net is to address the limitations of typical CNN in handling multi-scale information. In many computer vision works, objects in an image can vary significantly in size, and the model needs to understand both fine-grained details and border contextual information to perform well.

There are two parts of Res2Net which are:

1. Standard Residual Block (ResNet): In a standard residual block, the input features are passed through two consecutive convolutional layers, and the output of the second layer is added to the original input. The addition of residual information enables the effective training of very deep networks.

2. Res2Block(Res2Net) : In contrast to the single pathway in the standard Residual Block, the Res2Block splits the input feature into multiple groups. Each group then undergoes a separate convolutional transformation with different kernel sizes, capturing information at multiple scales.

After the conventional transformation in each group, the outputs are aggregated by concatenation, resulting in a fused feature map. This fused feature map combines multi-scale information from all the pathways within the Res2block. Finally, the fused feature map is added to the original input through a shortcut connection to create the final output of the Res2Block.

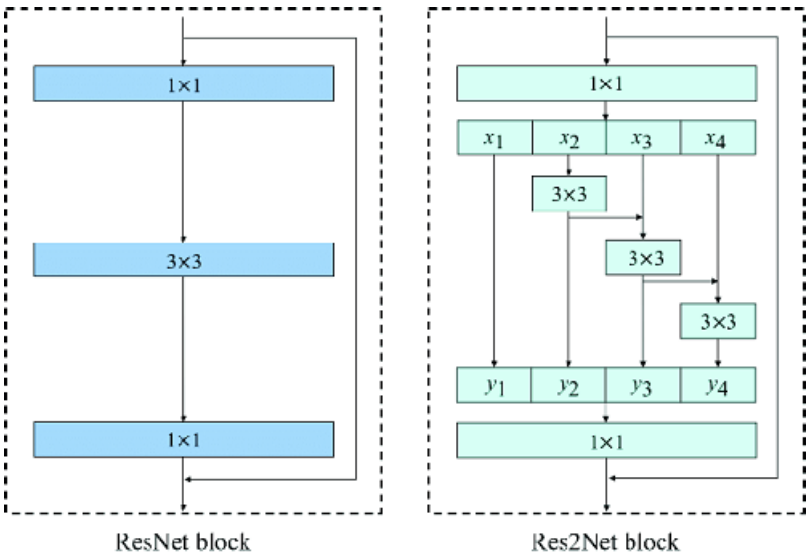



Figure 3.5: Res2Net Architecture

### 3.3.7 ResNet-50 Architecture

ResNet50 works by utilizing a deep CNN architecture with a focus on residual learning. The fundamental building block of Resnet-50 is the residual block, which

introduces a shortcut connection to enable efficient gradient flow during training. The network is able to learn the residual information, or the variation between each block's input and output, due to the shortcut connections that skip one or more convolutional layers. This design mitigates the vanishing gradient problem, and enables the training of a significantly deeper network. ResNet-50 further employs bottleneck blocks, which consist of three convolutional layers with reduced channel dimensions, effectively balancing computational complexity and representational capacity. The network depth extends to 50 layers, enabling it to capture intricate features from the input image. Global average pooling and fully connected layers follow the stack of convolutional layers to generate the final predictions for image classification tasks. Overall, ResNet 50 remarkable success lies in its ability to train very deep networks effectively and attain excellent performance in several different computer vision tasks.

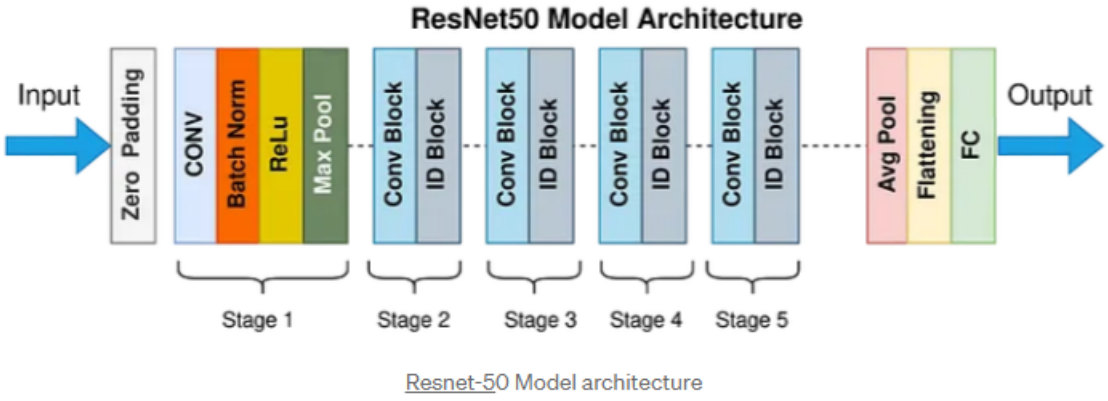



Figure 3.6: ResNet50 Architecture

### 3.3.8 Swin Transformer Architecture

The Swin transformer is designed for computer vision tasks specifically object detection, image classification and segmentation. It builds upon the popular vision transformer model and introduces several modifications to address the challenges of handling large image resolution efficiently.

The architecture of Swin Transformer consists of the following:

1. Shifted Windows: The core idea behind the Swin transformer is the use of Shifted windows. Unlike the traditional non-overlapping or image patches used in ViT, Swin Transformer divides the image into non-overlapping windows, each of which is processed individually. However, to capture interactions between adjacent windows, the windows are shifted in specific patterns during different stages of the model.

2. Hierarchical Architecture: Swin Transformer employs a hierarchical architecture, dividing the model into multiple stages or layers. Each layer processes the window at a particular resolution and gradually reduces the

spatial dimensions as the data flows through the network. This hierarchical approach helps to model efficiently and handle larger images.

3. Token to Token (T2T) Vision Transformers: In the Swin transformer, the hierarchical architecture uses a Token to Token mechanism to reduce computational complexity. Instead of flattening the spatial dimensions into one-dimension tokens, T2T rearranges the window into tokens. This enables the model to maintain spatial information within the hierarchical structure, reducing the overall number of tokens.

4. Shifted Transformer Block: The basic building block of the Swin Transformer is the shifted transformer block. It incorporates shifted self-attention, where an attention mechanism can capture long-range dependencies between windows without requiring global self-attention. The shifted self-attention operation reduces the computation complexity significantly compared to the full self-attention mechanism used by traditional transformers.

By using these key modifications, the Swin Transformer can efficiently process large images while still capturing long-range dependencies effectively. This makes it suitable for different computer vision tasks that require handling high-resolution images.

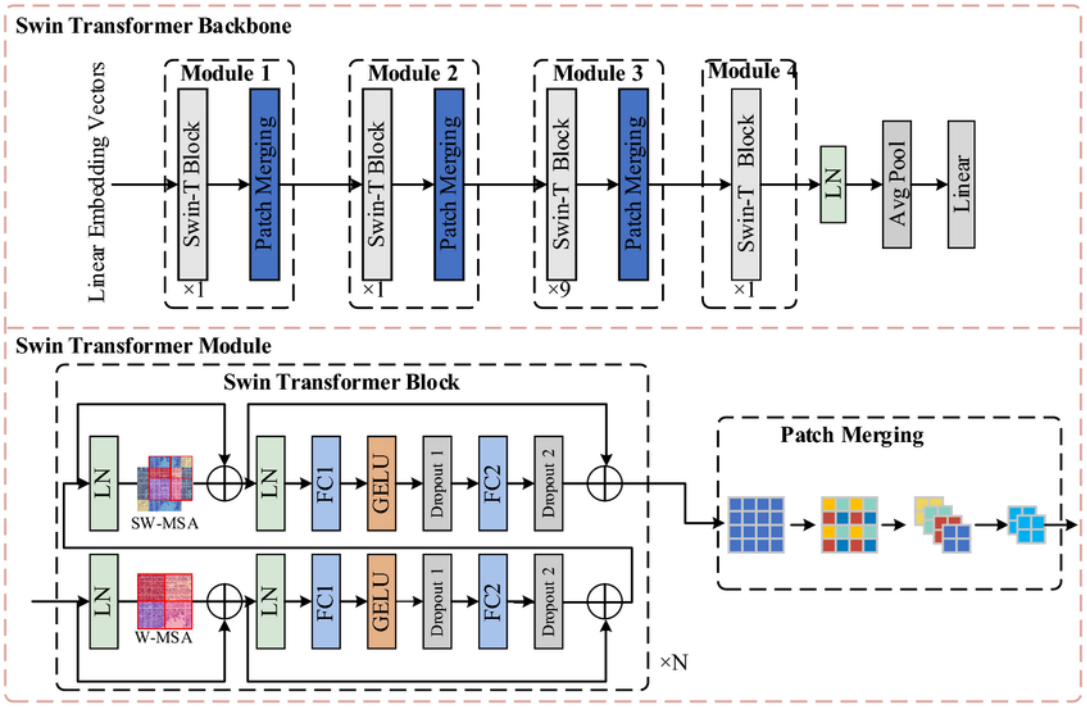



Figure 3.7: Swin Transformer Architecture

### 3.3.9 Ensemble

Ensemble is a technique in which multiple models are trained instead of training a single model, and the predictions of these multiple models are combined together to get a better accuracy. The aim of ensemble is to combine multiple models together and improve the accuracy of a model [19].

In ensemble, at first a set of models is selected which have been trained on a particular dataset or domain. After selecting the models, these models are loaded with pre-trained weights and fine-tuned on the target dataset using a small amount of labeled data. This process of fine-tuning enables the model to specialize and adapt its knowledge to the particular characteristics of the target dataset or domain. After the process of fine-tuning, an ensemble is carried out by combining the results of all the selected models. In the process of ensemble, different techniques such as weighted averaging, max voting, stacking, and blending are applied.

During ensemble in weighted averaging, each model's prediction is multiplied by a predefined weight before combining all the weights. In weighted averaging, the model weights are defined based on the model's performance on the validation dataset or through other heuristic measures. In the process of stacking, several estimators are merged to reduce biases, and the predictions from each estimator are combined. These combined predictions are fed into the final estimator, also known as the meta-model, as input, which computes the final prediction using a cross-validation technique. The process of blending in the ensemble is a variation of stacking that uses a holdout validation set instead of cross-validation, and in blending, the training data is split into two parts, where one part of the training data is used to train the base model and the other part is used to train the meta-model or final estimator. In blending, in order to train the meta model, the base model produces predictions on the validation set, and then these predictions are given as input into the meta model, which computes the final prediction by combining the predictions of the base model on the test data. Lastly, in the process of voting, each base model gives an output, and the final output is determined by calculating the majority votes. Ensemble helps to reduce the bias that arises by training a single model, and it also helps to reduce overfitting. By combining the predictions of different models, the ensemble also helps to improve the strength of a model.

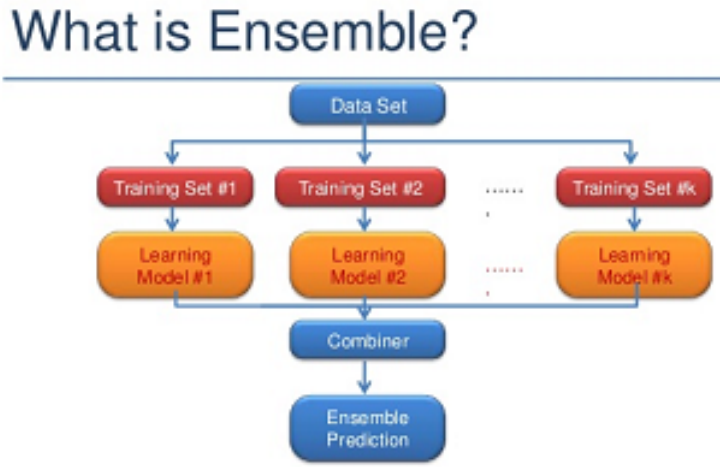



Figure 3.8: Ensemble

## 3.4 Description of the data

We have selected the Kvasir SEG dataset from Kaggle for our research purposes. The Kvasir SEG dataset is an openly accessible multiclass image dataset that includes

images of polyps along with images from 2 classes related to polyp removal and 3 anatomical features in the gastrointestinal (GI) tract.
This dataset has 8000 images in total, where the classes are classified into 8 classes. Each class of the dataset contains 1000 images. The Kvasir SEG dataset was created using medical images of the GI tract that were collected and analyzed by medical professionals at Vestre Viken Health Trust Hospital in Norway. In order to enhance medical practice and improve healthcare systems, this dataset has been created exclusively for computer-aided gastrointestinal disease detection by medical professionals. The original dataset can be downloaded from

https://www.kaggle.com/datasets/tanmaydebnath/kvasir-seg-dataset [17]

Out of all the publicly available datasets on the internet, we have selected this dataset because it contains medical images that have been gathered and validated by medical professionals, as opposed to the images of another publicly available dataset that have not been validated by medical professionals. In contrast to other datasets, the size of the Kvasir SEG dataset is significantly larger, and this dataset's images have been separated into numerous classes. As a result, it makes it easier to distinguish between polyps and non-polyp structures. Additionally, it will enable us to train our model more accurately.

## 3.5 Preliminary analysis

For our research purposes, we have selected the dataset from Kaggle [17]. For the following dataset, we used the EDA technique. It is a method for examining data sets to highlight their key properties. The selected dataset is a multiclass dataset that contains a total of 8000 images. In order to analyze the dataset, we have at first counted the number of classes in the dataset. Our dataset contains 8 different classes which are:

1. Dyed-Lifted-Polyps
2. Dyed-Resection-Margins
3. Esophagitis
4. Normal-Cecum
5. Normal-pylorus
6. Normal-z-line
7. Polyps
8. Ulcerative-colitis

The names of the classes are considered as the labels of the images inside the classes. The labels for this dataset are "inferred" (labels are constructed using the directory structure). The label mode is classified as 'categorical' which denotes the labels' categorical vector encoding. The class names are also considered valid since the labels are inferred. Here, each class contained 1000 images. No missing values were

identified in any of the classes. We also checked for duplicate images in the dataset, but no duplicate image was found. We also did not find any outliers in the dataset. Each class contained images of different sizes, such as 224x224, 720x576, 1920x1072, etc. In order to use this dataset, we resized all the images into 128x128 and added three channels (Red, Green, and Blue) as the color mode for the input dataset. As the dataset contained classes with different numbers of images, we selected a batch size of 128. Then we divided our dataset into training and testing sets with an 8:2 ratio, and we also normalized the pixel values of the images by dividing the train set and test set by 255.0. In order to get a broader dataset, we augmented the images in each divided set using OpenCV, where OpenCV has augmented the images by rotating the images. The image augmentation enriched our dataset and helped us to get better accuracy.

## 3.6 Data preprocessing

The data preprocessing procedures that we used on our original dataset were to partition the dataset into training and testing sets in 8:2 ratio. We used a Python code for the segmentation of the dataset based on the specified ratio. Using the training dataset, we have trained our model, and using the testing dataset, we have tested our model to find the accuracy and validation loss of the model.

## 3.7 Image preprocessing

Inorder to preprocess the images of our dataset, the image preprocessing technique that we have applied is image augmentation, which is the process of creating new, modified versions of the original images [13]. In our dataset, we have used OpenCV to augment the images. OpenCV reads each of the images from the dataset, and using a loop, it rotates the images one at a time using the cv2.getRotationMatrix2D function [20]. The function centers, angles, and scales the image while rotating the image. Image augmentation has diversified our dataset, which will help our model perform better and yield better results.

## 3.8 Transfer Learning Model Selection

In this research for transfer learning, we have used CNN-based transfer learning model and attention-based transfer learning model. We have also ensembled these different types of models together to determine whether the combination of all these models can help us to get a better accuracy while detecting polyps using our dataset. Inception-V3, Res2Net, ResNet-152, Resnet-50, and MobileNet-V2 are the CNN-based transfer learning models that we utilized in our study, while the Swin Transformer is the attention-based classification model that we used in our research. After applying these different types of transfer learning models and ensembling these models, we have compared the accuracy of these models to determine which transfer learning model is more suitable for detecting polyps from our dataset.

# Chapter 4

# Implementation

## 4.1 Environment Set Up

We have used Google Colab for our virtual environment setup. Then, we created a Python version 3.9-based virtual environment and installed the necessary libraries and software in Google Colab; these libraries include:

- Keras
- Pandas
- Tensorflow
- Tensorflow.gpu
- Matplotlib.pyplot
- Numpy
- Inception V3
- Imported Input, Lamda, Dense, Flatten, Conv2D, Batch normalization, Activation, Maxpooling 2D, Global average pooling 2D from tensorflow.keras.layers
- Imported model from tensorflow.keras.model
- os
- Seaborn
- Sklearn
- Scikit-learn
- open cv
- ResNet50
- ResNet152V2
- MobileNetV2

- Imported ModelCheckpoint from keras.callbacks

As we are working with an image dataset, we opted to use tensorflow-gpu for importing our deep learning libraries. For that reason, we have used the NVIDIA SMI libraries along with the TensorFlow installation. We have also used the Seaborn library to graphically represent the results found in our research model and used Matplotlib.pyplot to graphically plot our model accuracy and model loss graphs.

Here is the system configuration of our test bench:

- CPU: Core i5 9th Gen
- GPU: NVIDIA GTX 1650 Super
- OS: Windows 10 Pro
- RAM: 16 GB

## 4.2 Implementation of Transfer Learning Models

Before starting the implementation of all the transfer learning models, we started by preprocessing our dataset and dividing it into train and test sets in the ratio of 8:2. After dividing our dataset into the train, and test set we have implemented transfer learning models such as Inception V3, ResNet-152, MobileNet-V2, Res2Net, ResNet-50, and Swin Transformer using our preprocessed dataset. Each of the transfer learning models was saved in an .hd5 file after training. Lastly, we have combined the predictions of all the transfer learning models through ensemble to get better accuracy for our model. Now we will discuss the implementation of each of the transfer learning models in detail.

### 4.2.1 Implementation of Inception V3

In this instance, the suggested model makes use of the Inception-V3 weights that were previously learned using ImageNet and takes into account the resized images of 128 x 128 for all classes. We started by preprocessing our dataset and dividing it into train and test sets in an 8:2 ratio. Then we built a reference to the inception model, where we provided our input shape (dimension), three channels (Red, Green, and Blue), and picture weights. We also deactivated the out-of-the-box, last layer of the completely linked layer of the inceptionV3 network since we want to add more layers of our new model to it. Then we declared the layers as not trainable, as we are not aiming to retrain a pre-trained model; instead, we want to train our new model on the basis of this pre-trained model. The learning rate for the model was set to 1e-5, the batch size of the model was set to 128, and the validation split and label smoothing were set to 0.1. Additionally, we added two new layers to the model, which are Flatten Layer and Dense Layer. The flattened layer will flatten the neurons and add a fresh layer to the inception model. Then we added our dense layer which consists of the class numbers and an activation function of Softmax, with the flattened layer. After adding the layers, we started creating our model. The model is based on the inception input and the dense layer that we just added. The model was then compiled by defining the loss with a categorical function and

using an Adam optimizer with such a computed value of 4e-5 to achieve greater performance and accuracy as metrics. The model is then trained for 30 epochs and the progress is saved in a log file. The epoch index, however, is also defined to be an input for a scheduler learning rate. Then, in graphs, we plotted the train-validation accuracy graph and the train-validation data loss graph. Finally, we saved the model into a .hd5 file format, validated it with the test set, created the confusion matrix, and computed the classification metrics.

### 4.2.2 Implementation of ResNet-152

During the implementation of ResNet-152 on our dataset we at first imported the necessary libraries such as from tensorflow. keras.layers we imported Input, Lambda, Dense, and Flatten layers and we also imported models from Keras. models. After importing the necessary libraries we resized our images into 128 x 128 pixels, set the batch size to 128, set the learning rate of the model to $1e^{-5}$, set the validation split to 0.1 and set the value of the label smoothing to 0.1. Then we created the ResNet-152 base model using the weights of the imagenet model. After creating the ResNet-152 base model, we added a layer in the model by applying GlobalAveragePooling in the layers of ResNet-152 and we added a Dense layer, where the number of classes of the Dense layer was equal to the number of classes in our dataset and the activation function of the Dense layer was set to softmax. After adding the dense layer we compiled the model and trained the model for 30 epochs. Lastly, after training the model we saved the model in a .h5 file format, plotted the train accuracy vs. validation accuracy and the train loss vs validation data loss graphs, and generated the confusion matrix based on our obtained results. We also calculated the F1 score, precision, and recall score.

### 4.2.3 Implementation of MobileNet-V2

During the implementation of MobileNet-V2 we at first imported and installed the necessary libraries such as we imported tensorflow, installed nvidia-smi, and installed scikit-learn along with other libraries. After installing the necessary libraries we set the batch size of the MobileNet-V2 model to 128 and resized the images of our dataset to 128 x 128 pixels. After resizing the images, we created the MobileNet-V2 base model without the top classification layer using the weights of imagenet. Then we added a classification layer on top of the pre-trained model by first applying GlobalAveragePooling in the MobileNet-V2 model and after that, we added the Dense layer in the model, where the number of classes of the Dense layer was equal to the number of classes in our dataset and the activation function of the Dense layer was set to softmax. The learning rate was set to $1e^{-5}$, the batch size of the model was set to 128, and the validation split and label smoothing was set to 0.1. Following the addition of the Dense layer in the model we froze the layers of the base model, compiled the model, and trained the model for 30 epochs using our preprocessed dataset. Finally, we saved the model in a .h5 file format, plotted the train-validation accuracy and the train-validation data loss graphs, and generated the confusion matrix based on our obtained results. Additionally, we computed the classification metrics.

### 4.2.4 Implementation of Res2Net

Firstly while implementing Res2Net we imported all the libraries i.e. numpy, pandas, tensorflow, and keras. After that, we converted the images to 128x128 and made the batch size 128. Afterward, we imported the functions named Input, Lambda, concatenate, Conv2D, BatchNormalization, Activation, MaxPooling2D, GlobalAveragePooling2D, Dense, and Concatenate. Then we created a Res2Net block and defined the convolutional layer. The learning rate is $1e^{-5}$ for this model, the validation split and label smoothing was set to 0.1. We did the batch normalization of the convolutional layer and split the output tensor along the channel dimension into "scale"; smaller tensors also apply 3x3 convolution with the same stride to each split tensor and concatenate the split tensors along the channel dimension. In the next layer we did the max pooling and created the Res2Net blocks and the size of each block is 2. Finally did the average pooling and compiled the model. After that, we trained the model with 30 epochs. We saved the model. in h5 format also we plotted the accuracy and loss graph, generated the confusion matrix, and finally calculated the F1 score, precision, and recall score.

### 4.2.5 Implementation of ResNet-50

Firstly while implementing the ResNet50 we imported the libraries that include TensorFlow, tesorflow.keras, tensorflow._dataset, Matplot_pyplot, numpy. We also imported the imagedataGenerator and from Keras applications we imported ResNet50. After that, we converted the images to 128x128 and made the batch size 128. Next, we load the pre-trained ResNet50 model without the top layer using the image and freeze the pre-trained model layers, and create a new model. The learning rate is $1e^{-5}$ for this model, the validation split and label smoothing was set to 0.1. We created the new model using the sequential function while activating the softmax and compiled the model. Finally, we trained the model for 30 epochs. After compiling save the model with a .h5 foment also we plotted the accuracy and loss graph, generated the confusion matrix, and finally calculated the F1 score, precision, and recall score.

### 4.2.6 Implementation of Swin Transformer

Firstly while implementing the Swin Transformer we imported the libraries that include tensorflow, tensorflow.keras, tensorflow._dataset, Matplot_pyplot, numpy. After that we resize the images to (32,32) since the input_shape is (32,32,3) and appended in a list. We also normalized the pixel value and Convert class vectors to binary class matrices, the number of classes in the dataset is 8. We declared the patch size 2-by-2. We fixed the dropout rate as 0.03, attention rate as 8, Embedding dimension as 64, and MLP layer as 256. Converted embedded patches to query, key, and values with a learnable additive value by setting the value of qkv_bias as True, here shift size and window size are respectively 2 and 1. Here the image dimension is 128. We also fixed a learning rate and The learning rate is $1e^{-3}$ for this model, batch size is 128. After all this, we trained the model for 30 epochs. Before training the model we did the window partitioning and then did the window reversing and dropped some of the layers. Here we introduced a new function named window

partitioning did the window reversing and dropped some of the layers. Here we introduced a new function windowAttention and built a new window. After this, we introduced the main Swin Transformer model, in the model we declared the number of input dimensions, number of embedded patches, number of attention heads, size of the window, size of window shift, and number of MLP nodes. Here we did the array masking in the windows as well. Then again we PatchExtract, and PatchEmbedding, and we merged the path by PatchMerging. Lastly, after compiling we saved the model with a .h5 format and also plotted the accuracy and loss graph, generated the confusion matrix, and finally calculated the F1 score, recall score, and precision.

### 4.2.7 Implementation of Ensemble

During the ensemble, we imported and installed the necessary libraries which are needed to ensemble the models, we loaded our preprocessed dataset and the saved models through the .h5 files, which we had previously saved while training all five models ( ResNet-152, ResNet-50, MobileNet-V2, Res2Net, and Inception V3 ) individually. In order to compute the overall performance of our models through an ensemble, we have formed different combinations using the 5 CNN-based transfer learning models. The combinations of the models have been divided into the combination of all 5 models, the combination of only 2 models, the combination of only 3 models, and the combination of only 4 models.

The combinations which we have formed are:

1. Combination of all 5 models ( ResNet-152, ResNet-50, MobileNet-V2, Res2Net and Inception V3 model )
2. Combination of ResNet-152 and ResNet-50
3. Combination of ResNet - 152 and MobileNet-V2
4. Combination of ResNet - 152 and Res2Net
5. Combination of ResNet - 152 and Inception V3
6. Combination of ResNet-50 and MobileNet-V2
7. Combination of ResNet - 50 and Res2Net
8. Combination of ResNet - 50 and Inception V3
9. Combination of MobileNet-V2 and Res2Net
10. Combination of MobileNet-V2 and Inception V3
11. Combination of Res2Net and Inception V3
12. Combination of ResNet-152, ResNet-50 and MobileNet-V2
13. Combination of ResNet-152, ResNet-50 and Res2Net
14. Combination of ResNet-152, ResNet-50 and Inception V3

15. Combination of ResNet-152, MobileNet-V2 and Res2Net
16. Combination of ResNet-152, MobileNet-V2 and Inception V3
17. Combination of ResNet-152, Res2Net and Inception V3
18. Combination of ResNet-50, MobileNet-V2 and Res2Net
19. Combination of ResNet-50, MobileNet-V2 and Inception V3
20. Combination of ResNet-50, Res2Net and Inception V3
21. Combination of MobileNet-V2, Res2Net and Inception V3
22. Combination of ResNet-152, ResNet-50, MobileNet-V2 and Res2Net
23. Combination of ResNet-152, ResNet-50, MobileNet-V2 and Inception V3
24. Combination of ResNet-152, ResNet-50, Res2Net and Inception V3
25. Combination of ResNet-152, MobileNet-V2, Res2Net and Inception V3
26. Combination of ResNet-50, MobileNet-V2, Res2Net and Inception V3

In all of the above combinations of the model, we have applied three different types of ensemble techniques which are blending, weighted averaging and stacking. Blending is an ensemble technique where we have to train multiple models on the same dataset and combine their predictions. Here, we used two different layers of machine learning algorithms (base models and meta-models) following the holdout approach. In this process, we have divided our training set of the dataset into a base model train set and a holdout (validation) set in the ratio of 8:2. Then we trained our transfer learning models on the base model train set and fed the holdout sets to the models depending on which they started their predictions. These predictions were then considered as the new training data for the metamodel. For our meta-model, we used a Random Forest Classifier with the n-estimator value of 100 and the random state value of 42. The output of this meta-model was then picked as the result of the ensembled models. Finally, we used our default test set for the blended model evaluation. Lastly, we obtained the accuracy, mean F1 score, mean recall, and mean precision of the blending model. For all 26 combinations the same process of blending was followed and the accuracy, mean F1 score, mean recall score, and mean precision score of the combinations was determined.

Weighted averaging is an ensemble technique where different weights are assigned to each model's predictions according to the model's previous performance or based on our previous knowledge regarding the model. While applying the weighted averaging technique on our model, we have set the weights of the models according to each of the model's base performances on the test set and finally computed the average of the models' predictions. We also calculated the accuracy, mean F1 score, mean recall score, and mean precision score for each of the combinations of the models.

Stacking is a technique similar to blending where we combine the base models through a meta-model while using k-fold cross-validation technique. In this case, we split the dataset into k=5 equal folds. We trained each of our base models on k-1

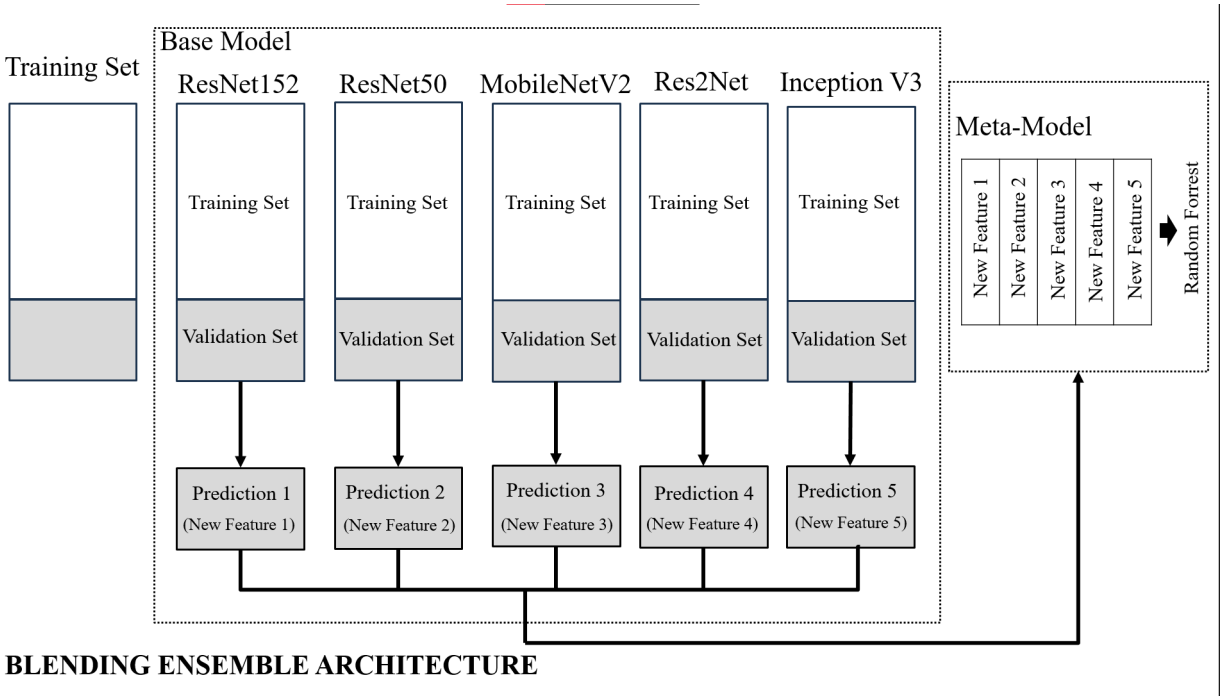



Figure 4.1: Blending Architecture

folds. This means that for each fold in the cross-validation loop, we had a separate set of base models trained on different data. After training, we used these base models to make predictions on the data in the remaining 1 fold. Each base model generates predictions for this fold. Then we collected these predictions from all the base models and stacked them horizontally (side by side) for each data point in the validation fold. This created a new dataset where each row corresponded to a data point, and each column contained the prediction from one of the base models. After completing the k-fold loop, we got a new dataset of predictions, where each data point has k predictions from the base models. This newly formed dataset is then used to train our meta-model. We used a logistic regression classifier for our meta-model. It took the stacked predictions as input features and learned to make the final prediction. Finally, it is evaluated on the default test set. We also computed the F1 score, recall, and precision. For each of the combinations mentioned above, we applied these ensemble techniques of blending, weighted averaging, and stacking in the same way and computed the accuracies of these techniques. We implemented this ensemble technique to evaluate which combination of the transfer learning models provides us with the maximum accuracy for predicting polyps from our dataset.

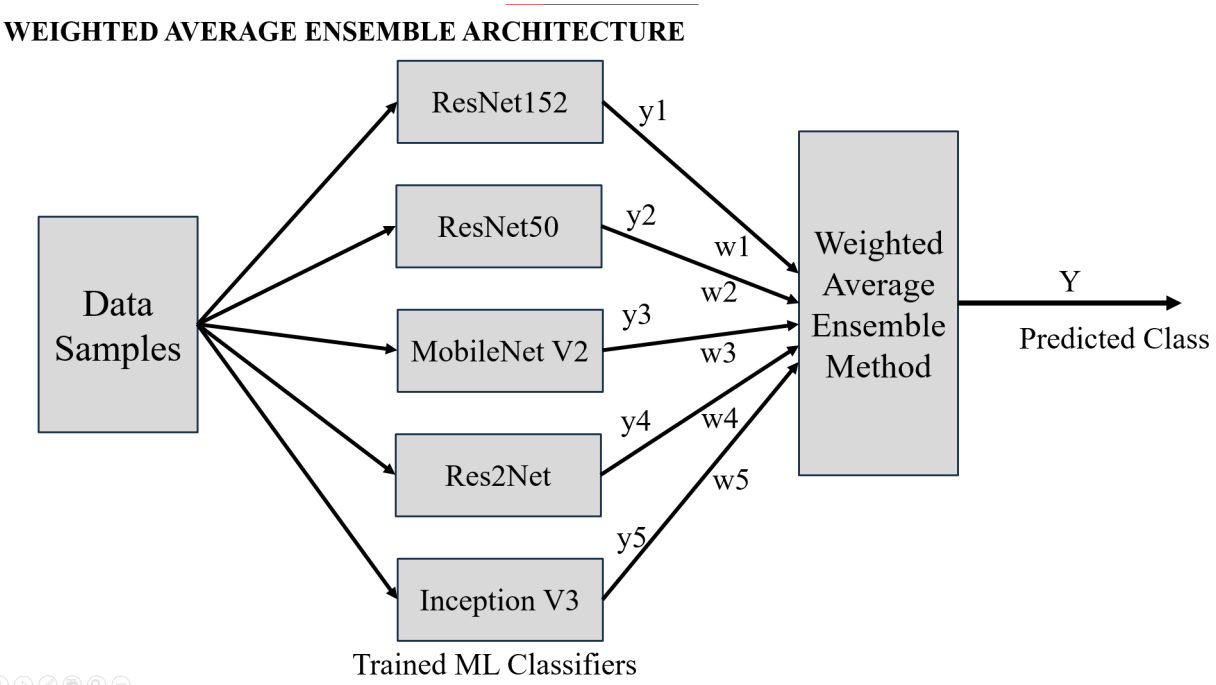



Figure 4.2: Weighted Averaging Architecture

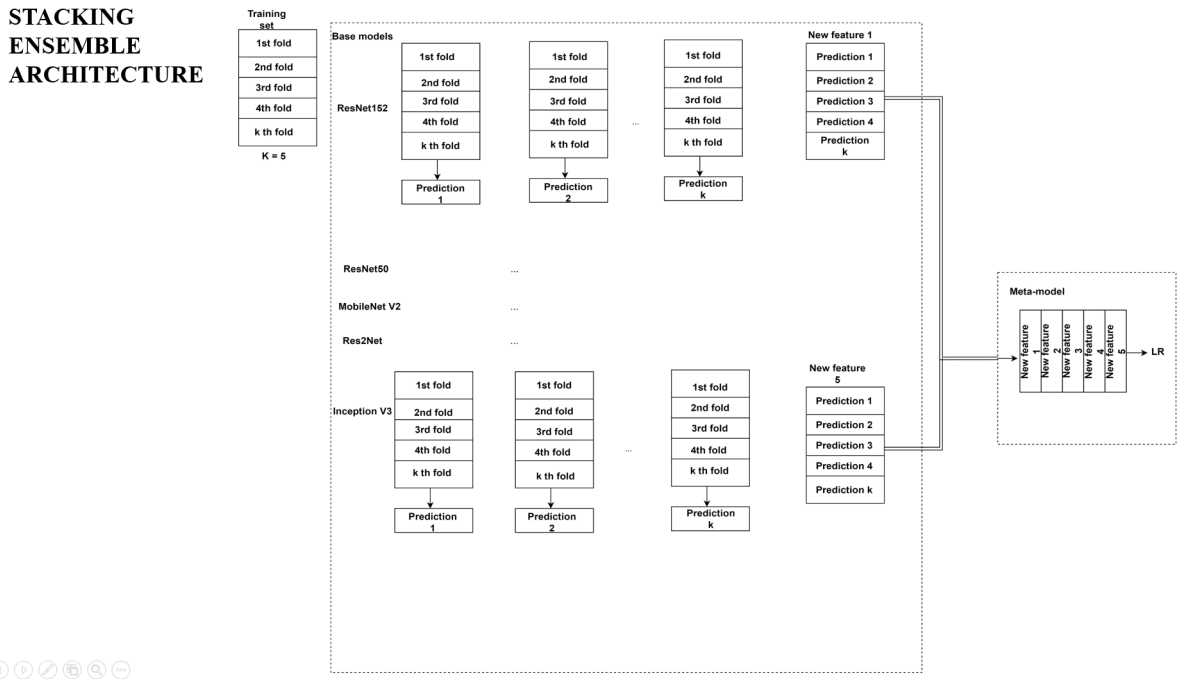



Figure 4.3: Stacking Architecture

# Chapter 5

# Results and Analysis

## 5.1 Results

The performance of the model was assessed with parameters like accuracy, recall, f1 score, precision, macro avg and weighted avg. The formulas for some of the metrics are given as:

$$precision = \frac{TP}{TP + FP} \tag{5.1}$$

$$recall = \frac{TP}{TP + FN} \tag{5.2}$$

$$F1 = \frac{2 \times precision \times recall}{precisio + recall} \tag{5.3}$$

$$accuracy = \frac{TP + TN}{TP + FN + TN + FP} \tag{5.4}$$

- Precision indicates the fraction of identifications that were correct for a particular class. If all images in a class were identified as true positives, then the precision value would be 1.0. In the event of any false positives, the value will be less than 1.0.
- Recall denotes the fraction of correctly identified true positives. When there are no false negatives, recall will give a value of 1.0 for a particular class. If less than 1.0, it is an indicator of true positives that were mislabeled.
- The F1 score combines both precision and recall to give a new evaluation metric. Achieving an F1 score of 1.0 would mean there were no misclassifications.
- Accuracy is the accuracy percentage of the model. If the model is accurately predicting all test cases at all times, its accuracy will be 1.0. Accuracy graphs have also been shown as a performance indicator, labeling the particular class and saying that it is giving a perfect result.
- Macro avg is the average precision, recall, and F1 score between the classes. It is an indicator of which class disparities are present.

- Weighted avg is the weighted average precision, recall, and F1 score between the classes. This is calculated with respect to the number of samples in each class, and hence, class imbalance is a factor in differences between the macro and weighted average.

## 5.2 Performance Study of Inception V3

We trained the Inception v3 model for 30 epochs. The mean training accuracy is 89.04%, the mean test accuracy is 77.27%, the mean F1 score is 0.7710, the mean recall score is 0.7732, and the mean precision is 0.7756.

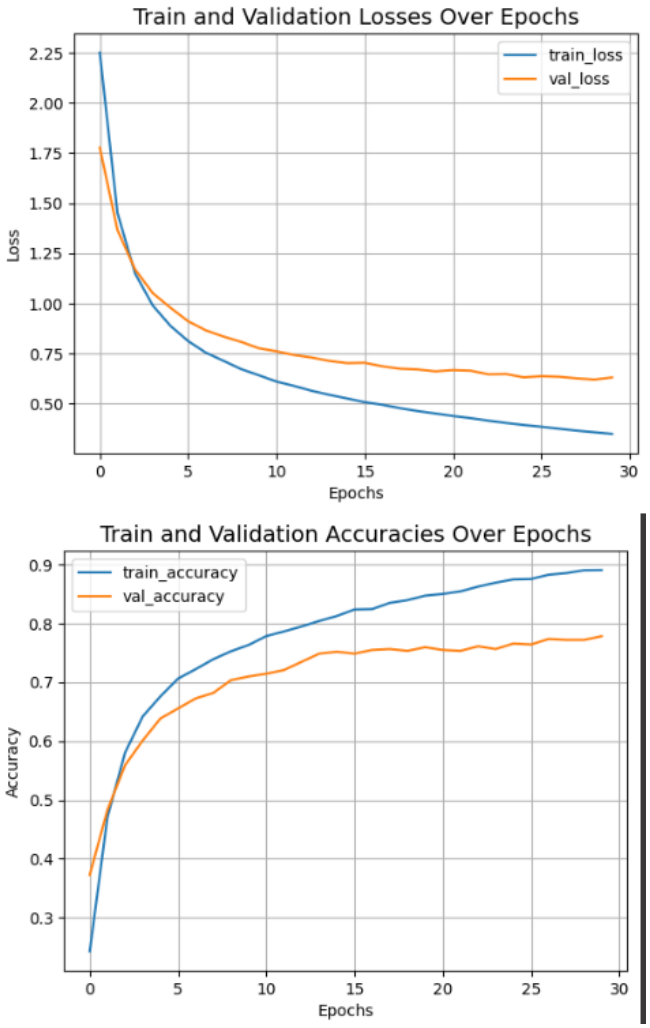



Figure 5.1: Train and Validation Loss Accuracy graph

The train loss of Inception V3 is 0.35, and the test loss of Inception V3 is 0.59. Train validation loss over the epoch graph shows that both the train and validation loss decrease, but compared to the train loss, the validation loss decreases slowly. According to the training validation accuracy graph, the training accuracy is greater than the validation accuracy.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.87 | 0.86 | 0.86 | 202 |
| 1 | 0.77 | 0.71 | 0.74 | 216 |
| 2 | 0.70 | 0.70 | 0.70 | 189 |
| 3 | 0.68 | 0.82 | 0.74 | 197 |
| 4 | 0.72 | 0.79 | 0.75 | 210 |
| 5 | 0.79 | 0.57 | 0.66 | 201 |
| 6 | 0.89 | 0.92 | 0.90 | 195 |
| 7 | 0.80 | 0.81 | 0.81 | 200 |
| accuracy | | | 0.77 | 1610 |
| macro avg | 0.78 | 0.77 | 0.77 | 1610 |
| weighted avg | 0.78 | 0.77 | 0.77 | 1610 |

Table 5.1: Classification report of Inception V3

In the classification report of InceptionV3, the numbers 0–7 represent the different classes of dyed lifted polyps, dyed resection margins, esophagitis, normal cecum, normal pylorus, normal z line, polyps, and ulcerative colitis, respectively. From the classification report, it can be seen that the precision macro avg and weighted avg are 0.78 and the recall and F1 score macro avg and weighted avg are 0.77.

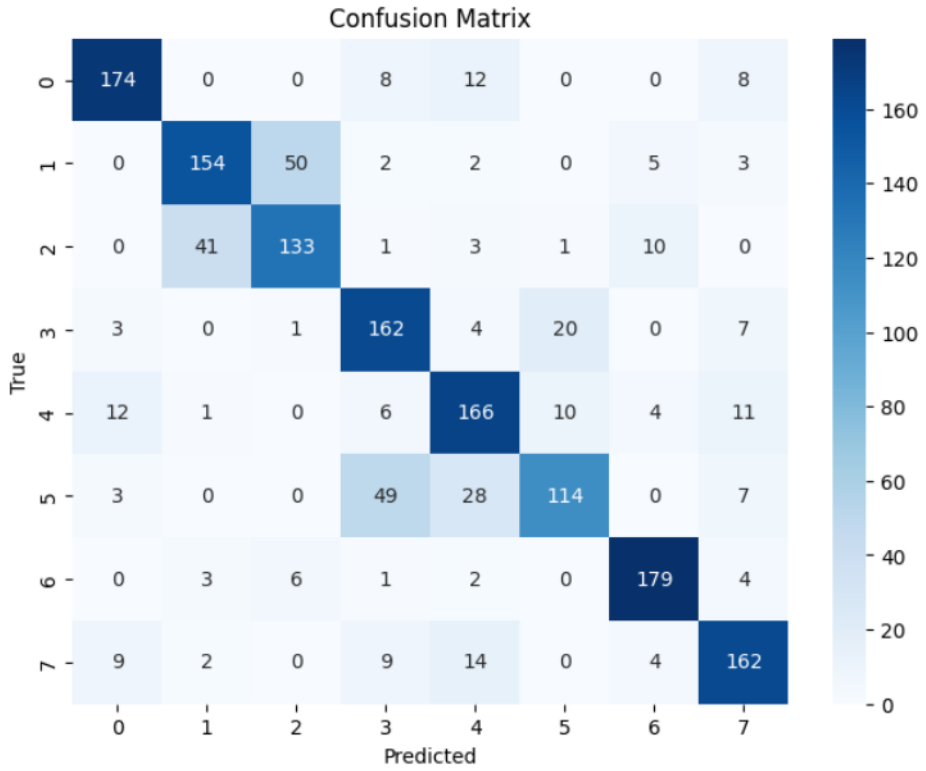



Figure 5.2: Confusion Matrix of inception V3

The row of the confusion matrix represents the true labels, while the column represents the predicted labels. For instance, the number 174 in the first row and first column represents that there were 174 images of dyed lifted polyps that were correctly classified as dyed lifted polyps. The number 0 in the second row and the first

column indicates that there were 0 images of dyed resection margins that were incorrectly classified as dyed lifted polyps. In Inception V3, we can see that 3 images are misclassified as dyed resection margins, 6 images are misclassified as esophagitis, 1 image is misclassified as normal cecum, 2 images are misclassified as normal pylorus, 4 images are misclassified as ulcerative colitis, and finally 179 images are classified as polyps.

## 5.3 Performance Study of ResNet-152

We have trained the ResNet-152 model for 30 epochs using our preprocessed dataset, which has been earlier divided into test and train sets in the ratio of 8:2. After training the model for 30 epochs, we generated a train accuracy vs. validation accuracy graph, a train loss vs. validation loss graph, a confusion matrix, and a classification report. The model has a train accuracy of 89.54% and a test accuracy of 81.86%. The graph below is the train accuracy vs. validation accuracy graph of the ResNet-152 model, where the X axis contains the epoch numbers and the Y axis of the graph contains the accuracy.

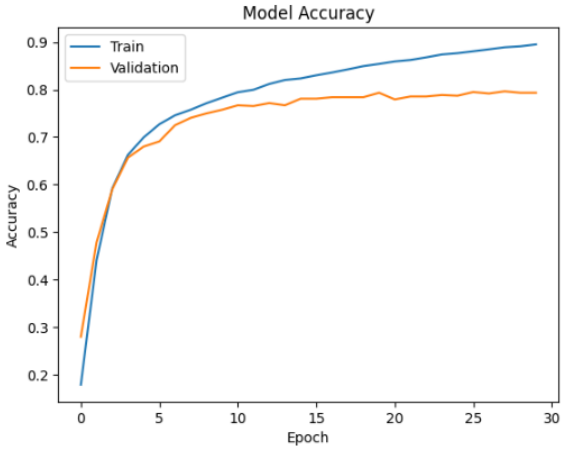



Figure 5.3: Train and Validation Accuracy graph

From the graph, it can be seen that the training accuracy of the model increases gradually during the training, whereas the validation accuracy of the model increases during the first 10 epochs, and after 10 epochs, the validation accuracy graph slowly reaches a plateau. From this graph, it can be inferred that the model learned the training data well and could classify the training images accurately. However, when the model was given the validation images, it was unable to classify the validation images as accurately as the training images, yet the model classified most of the validation images accurately. Hence, the model has a higher train accuracy compared to the validation accuracy.

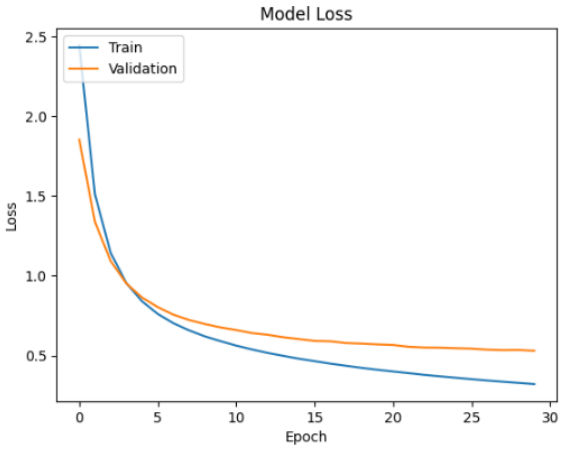



Figure 5.4: Train and Validation Loss graph

Now, comparing the train loss and validation loss graphs of the ResNet 152 model, it can be seen that both the train loss and validation loss of the model are decreasing. The train loss of the model is 0.32, and the validation loss of the model is 0.51.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.88 | 0.88 | 0.88 | 202 |
| 1 | 0.79 | 0.71 | 0.75 | 216 |
| 2 | 0.69 | 0.79 | 0.74 | 189 |
| 3 | 0.79 | 0.83 | 0.81 | 197 |
| 4 | 0.84 | 0.84 | 0.84 | 210 |
| 5 | 0.79 | 0.73 | 0.76 | 201 |
| 6 | 0.94 | 0.93 | 0.96 | 195 |
| 7 | 0.84 | 0.85 | 0.85 | 200 |
| accuracy | | | 0.82 | 1610 |
| macro avg | 0.82 | 0.82 | 0.82 | 1610 |
| weighted avg | 0.82 | 0.82 | 0.82 | 1610 |

Table 5.2: Classification report of ResNet-152

The above table 5.2 shows the classification report of the ResNet 152 Model. In the table below, 0 represents the class dyed lifted polyps, 1 represents the class dyed resection margins, 2 represents the class esophagitis, 3 represents the class normal cecum, 4 represents the class normal pylorus, 5 represents the class normal z line, 6 represents the class polyps, and 7 represents the class ulcerative colitis. The classification report shows that the ResNet-152 model has a macroaverage and weighted average of 0.82. From the classification report, we also got a mean precision score of 0.8197, a mean recall score of 0.8197 and a mean F1 score of 0.8188.

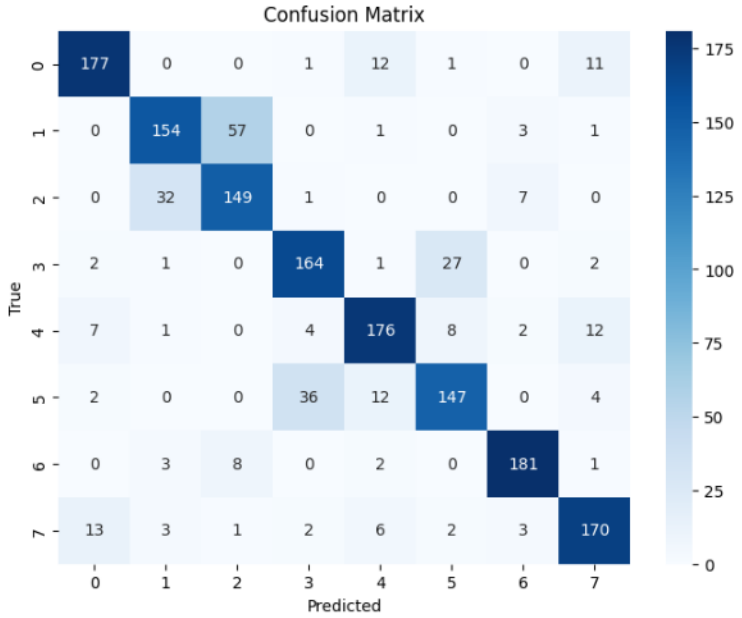



Figure 5.5: Confusion Matrix of ResNet-152

Here, from the above image we can see the confusion matrix of the ResNet - 152 Model. Using the confusion matrix we can see the number of classes the model classified correctly and which classes the model mislabeled as True positive class. From the confusion matrix it can be seen that out of 195 polyp images, the model was able to correctly classify 181 polyp images accurately, however the model mislabeled 3 of the polyp images as dyed resection margins, 8 of the polyp images as esophagitis, 2 of the polyp images as normal pylorus and 1 polyp image as ulcerative colitis. The confusion matrix also shows that the ResNet 152 model has been able to correctly classify 177 images of the dyed lifted polyps out of 202 images of dyed lifted polyps and it has misclassified 1 image of dyed lifted polyp as normal cecum, 12 image of dyed lifted polyp as normal pylorus, 1 image of dyed lifted polyp as normal z line, 11 image of dyed lifted polyp as ulcerative colitis. Hence, from the confusion matrix it can be interpreted that most of the polyp images and dyed lifted polyp images were accurately classified by the model, even though it mislabeled few of the polyp and dyed lifted polyp images. Using the confusion matrix, it can also be figured out that the ResNet-152 model misclassified the ulcerative colitis class as images of the ulcerative colitis class were misclassified as images of all the other classes.

## 5.4 Performance Study of MobileNet-V2

We trained the MobileNetV2 model for 30 epochs.The train accuracy of the model is 86.72%, the test accuracy of the model was 80.99%,where the mean F1 score is 0.8102, the mean Recall score is 0.8101, and the mean precision score is 0.8107. This model seemed to perform well in most of the classes. It has some difficulty distinguishing between dyed lifted polyps and polyps and between esophagitis and normal cecum.

The two graphs in this image show the relationship between training accuracy and validation accuracy for a machine-learning model. The training accuracy shows how

the model is performing in the training data while the validation accuracy shows how the model is performing in the new data. We want the training accuracy to be as close as possible to the validation accuracy. To sum it up, the graph shows how well a machine learning model is performing on training data that is training loss line and the new data (validation loss curve). It shows that the model is learning and improving over time but it started to overfit the training data as the training loss line is getting higher than the validation loss curve. To avoid the complexity of the model we have to increase the amount of the training data.

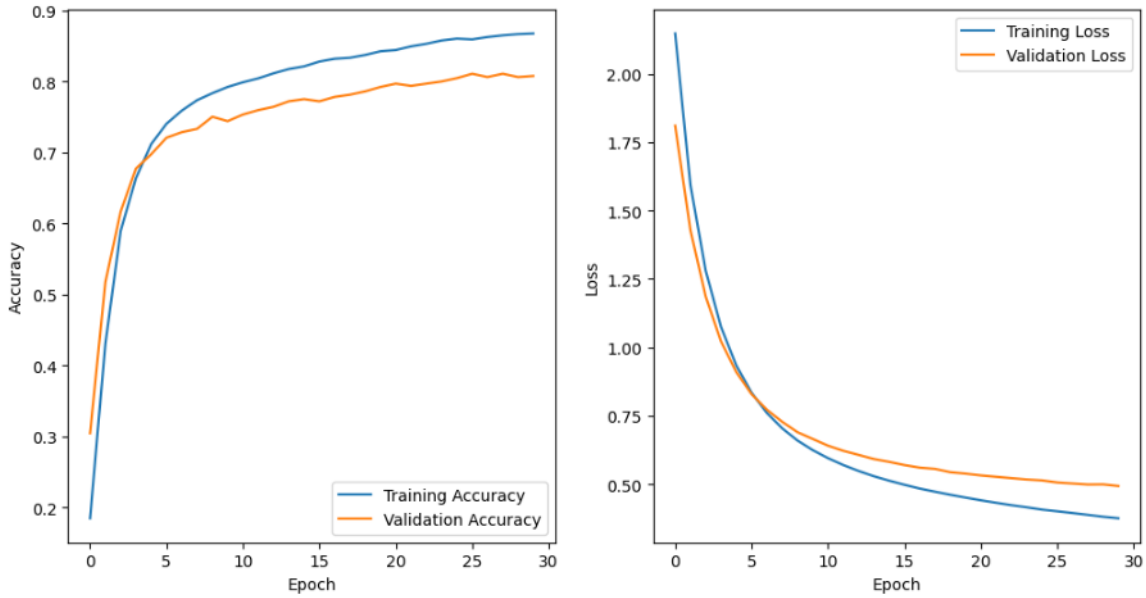



Figure 5.6: Train Validation Accuracy Loss Graphs

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.84 | 0.87 | 0.85 | 202 |
| 1 | 0.74 | 0.74 | 0.74 | 216 |
| 2 | 0.68 | 0.70 | 0.69 | 189 |
| 3 | 0.82 | 0.76 | 0.79 | 197 |
| 4 | 0.82 | 0.81 | 0.81 | 210 |
| 5 | 0.79 | 0.81 | 0.80 | 201 |
| 6 | 0.99 | 0.94 | 0.94 | 195 |
| 7 | 0.86 | 0.86 | 0.86 | 200 |
| accuracy | | | 0.81 | 1610 |
| macro avg | 0.81 | 0.81 | 0.81 | 1610 |
| weighted avg | 0.81 | 0.81 | 0.81 | 1610 |

Table 5.3: Classification Report of MobileNet-V2

In the classification report of MobileNet-V2, the numbers represents different classes such as 0 - dyed lifted polyps, 1- dyed resection margins, 2 - esophagitis, 3 - normal cecum, 4 - Normal pylorus, 5- normal z line, 6 - polyps, 7- ulcerative colitis. From the classification report, it can be seen that the precision macro avg and weighted avg is 0.81 and the recall and f1 score macro average and weighted average is 0.81.

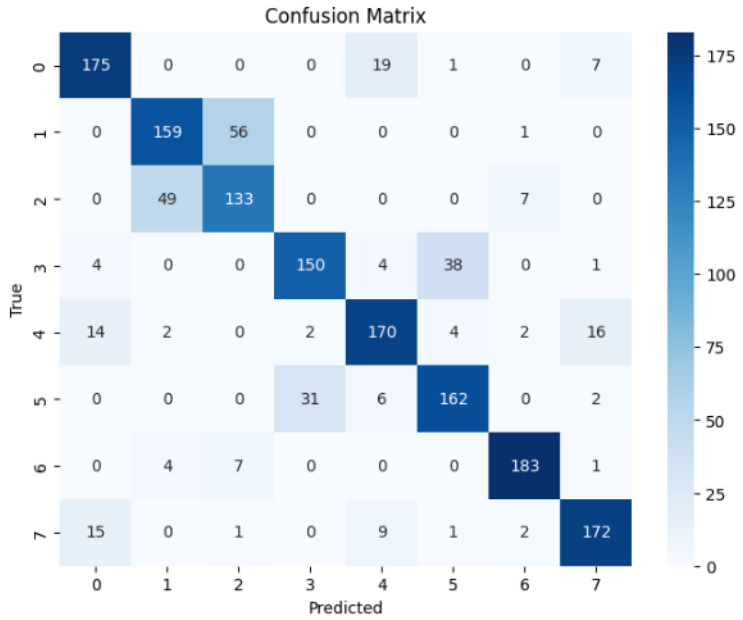



Figure 5.7: Confusion Matrix of MobileNet-V2

From the confusion matrix it can be seen that the number 175 in the first row and first column represents that there were 175 images of dyed lifted polyps that were correctly classified as dyed lifted polyps. The 0 in the second row and first column indicates that there were 0 images of dyed resection margins that were incorrectly classified as dyed lifted polyps. The class 6 was polyp. From the confusion matrix we can observe that it misclassified 4 image as dyed resection margins, 7 image as esophagitis and 1 image as ulcerative colitis. Finally, it classified the 183 image as polyps. It is to mention that the classes are 0 - dyed lifted polyps, 1- dyed resection margins, 2 - esophagitis, 3 - normal cecum, 4 - Normal pylorus, 5- normal z line, 6 - polyps, 7- ulcerative colitis.

## 5.5 Performance Study of Res2Net

The Res2Net transfer learning model has been trained for 30 epochs using our preprocessed dataset which has been earlier divided into test and train sets in the ratio of 8:2 . After training the model for 30 epochs we generated train accuracy vs validation accuracy graph, train loss vs validation loss graph, confusion matrix and the classification report. The model has a train accuracy of 86.33% and a test accuracy of 76.46 %. The graph below is the train accuracy vs validation accuracy graph of the ResNet-152 model where the X axis contains the number of the epochs and the Y axis of the graph contains the accuracy. The graph shows that the training accuracy curve rapidly grows during the first ten epochs and gently increases after that.

On the other hand, the validation curve increases rapidly during the first 14 epochs. However, compared to the train accuracy curve during this time period the validation accuracy curve increases slowly. After 14 epochs the validation accuracy curve started to increase slowly and reached a plateau but after reaching a plateau the curve started to increase slowly. This shows that the model is still learning the

training data but it is not generalizing well with the unseen data. Therefore, the Res2Net model has a higher train accuracy compared to the validation accuracy.

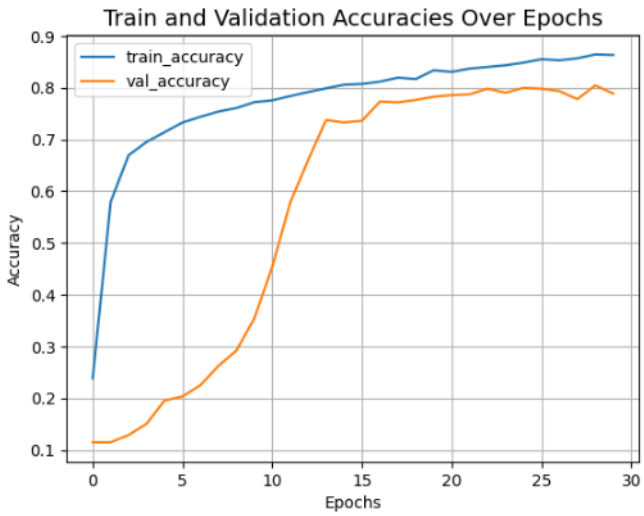



Figure 5.8: Train Validation Accuracy Graph

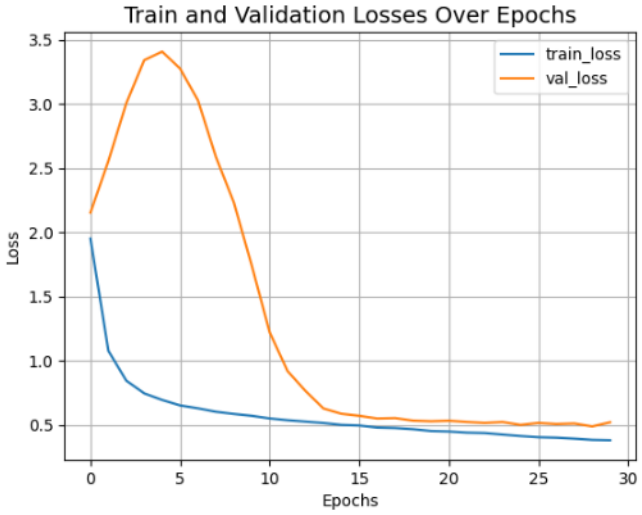



Figure 5.9: Train Validation Loss Graph

The second graph above shows the train loss and validation loss of the Res2Net model. The train loss of the Res2Net model is 0.38 and the test loss of the Res2Net model is 0.52. By looking into the train loss and validation loss graphs it can be seen that the train loss curve of the Res2Net model decreases gradually. However, from the graph, it can also be seen that during the first 4 epochs, the validation loss curve increases rapidly and after 4 epochs till 15 epoch it decreases rapidly. After 15 epochs the validation loss curve decreases slowly and eventually reaches a plateau. During the first 4 epochs, the behavior of the train loss and validation loss indicates overfitting has occurred as during the first 4 epochs from the graph it can be seen that the train loss was decreasing and during the same time period the validation loss was increasing. Therefore, from the training loss and the validation loss graph,

it can also be interpreted that due to overfitting the Res2Net model has a lower validation accuracy than the train accuracy.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.89 | 0.79 | 0.84 | 202 |
| 1 | 0.75 | 0.78 | 0.76 | 216 |
| 2 | 0.74 | 0.72 | 0.73 | 189 |
| 3 | 0.69 | 0.78 | 0.73 | 197 |
| 4 | 0.61 | 0.90 | 0.73 | 210 |
| 5 | 0.73 | 0.60 | 0.66 | 201 |
| 6 | 0.95 | 0.96 | 0.95 | 195 |
| 7 | 0.92 | 0.58 | 0.71 | 200 |
| accuracy | | | 0.76 | 1610 |
| macro avg | 0.78 | 0.76 | 0.76 | 1610 |
| weighted avg | 0.78 | 0.76 | 0.76 | 1610 |

Table 5.4: Classification Report of Res2Net

The table above shows the classification report of the Res2Net Model. In the table, 0 represents the class dyed lifted polyps, 1 represents the class dyed resection margins, 2 represents the class esophagitis, 3 represents the class normal cecum, 4 represents the class normal pylorus, 5 represents the class normal z line, 6 represents the class polyps and 7 represents the class ulcerative colitis. The classification report of the model shows that for precision the macro average and weighted average is 0.78 and for recall and f1 score the macro average and weighted average is 0.76. From the classification report, we also got a mean precision score of 0.7845, mean recall score of 0.7641 and mean F1 score of 0.7646 .

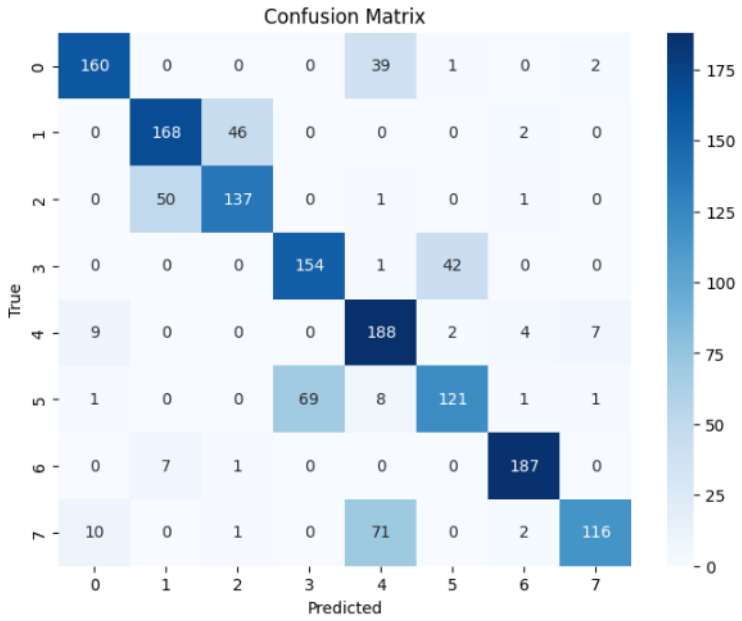



Figure 5.10: Confusion Matrix of Res2Net

In the above image, we can see the confusion matrix of the Res2Net model. Using the confusion matrix, we can see the number of classes the model classified correctly

and which classes the model mislabeled as true positive classes. From the confusion matrix, it can be seen that out of 202 dyed lifted polyp images, the Res2Net model classified 160 dyed lifted polyp images accurately; however, the model also mislabeled some of the classes, such as 39 dyed lifted polyp images as normal pylorus, 1 dyed lifted polyp image as normal z line, and 2 dyed lifted polyp images as ulcerative colitis. The confusion matrix also shows that out of 195 polyp images, Res2Net has been able to classify 187 images accurately; however, it has also mislabeled the labels, as it misclassified 7 of the polyp images as dyed resection margins and 1 polyp image as esophagitis.

## 5.6 Performance Study of ResNet50

We trained the ResNet50 model for 30 epochs. The training accuracy for the ResNet50 is 39.48%, the test accuracy is 38.43%, the mean f1 score is 0.3481, the mean recall score is 0.3832, and the mean precision score is 0.4297.

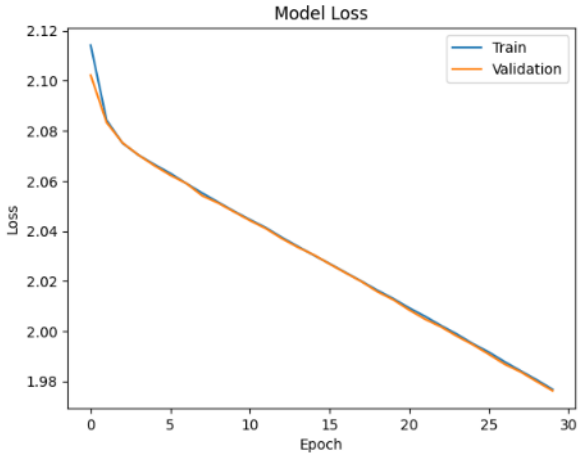



Figure 5.11: Train Validation Loss Graph

We can see in the loss graph that the train loss for ResNet50 is 1.976 and the validation loss is 1.98, which is almost the same. Also in the graph, the loss curve is decreasing slowly, and both the train and validation curves almost match in the end.

Moreover, the model accuracy for ResNet50 is 38.43%. In the graph, the train and validation accuracy are almost the same too. In the 30th epoch, the train and validation curves give almost the same number. From the train accuracy and validation accuracy graphs, it can be seen that there are some ups and downs in the curve too. The ups and downs in the train accuracy and validation accuracy indicate that the model is not learning the training data well, and due to this, the model has a lower training accuracy compared to the other transfer learning models such as ResNet-152, MobileNet V2, Inception V3, and Res2Net. As the model is unable to learn from the training data efficiently, it is also unable to generalize the data when presented with the validation data. As a result, the model also has lower test accuracy. The train accuracy and validation accuracy graphs of ResNet-50 also indicate that the model is overfitting the training data. In order to overcome this, the learning rate of the model can be decreased.

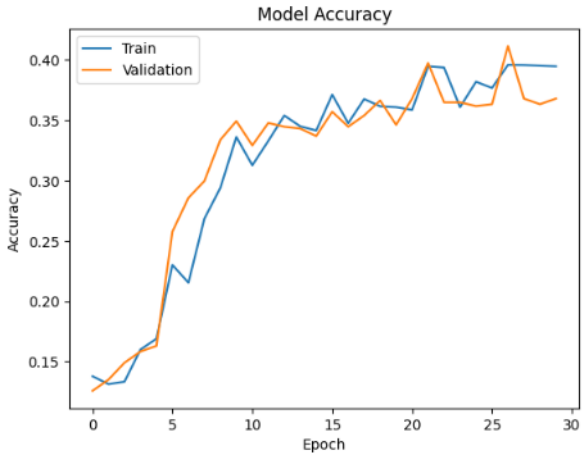



Figure 5.12: Train Validation Accuracy Graph

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.57 | 0.36 | 0.44 | 207 |
| 1 | 0.29 | 0.22 | 0.25 | 212 |
| 2 | 0.32 | 0.26 | 0.29 | 189 |
| 3 | 0.21 | 0.12 | 0.16 | 192 |
| 4 | 0.40 | 0.84 | 0.54 | 211 |
| 5 | 0.85 | 0.10 | 0.19 | 210 |
| 6 | 0.43 | 0.45 | 0.44 | 191 |
| 7 | 0.36 | 0.40 | 0.48 | 196 |
| accuracy | | | 0.38 | 1608 |
| macro avg | 0.43 | 0.38 | 0.35 | 1608 |
| weighted avg | 0.43 | 0.38 | 0.35 | 1608 |

Table 5.5: Classification report of ResNet50

In the classification report of ResNet50 the numbers represents different classes such as 0 - dyed lifted polyps, 1- dyed resection margins, 2 - esophagitis, 3 - normal cecum, 4 - Normal pylorus, 5- normal z line, 6 - polyps, 7- ulcerative colitis. From the classification report it can be seen that the precision macro average and weighted avg are 0.43. The recall and f1 score macro average and weighted average are 0.38 and 0.35 respectively.

For the confusion matrix the classes are 0 - dyed lifted polyps, 1- dyed resection margins, 2 - esophagitis, 3 - normal cecum, 4 - Normal pylorus, 5- normal z line, 6 - polyps, 7- ulcerative colitis. From the confusion matrix, it can be seen that the model correctly classified 75 images of dyed lifted polyps and it misclassified the other images of dyed lifted polyp with other classes such as it misclassified 2 images of dyed lifted polyp with dyed resection margins, 11 images of dyed lifted polyp with esophagitis, 38 images of dyed lifted polyp with normal cecum, 54 images of dyed lifted polyp with normal pylorus, 15 images of dyed lifted polyp with polyp and 12 images of dyed lifted polyp with ulcerative colitis. The confusion matrix also shows that the ResNet-50 model is misclassifying polyps and dyed lifted polyps.

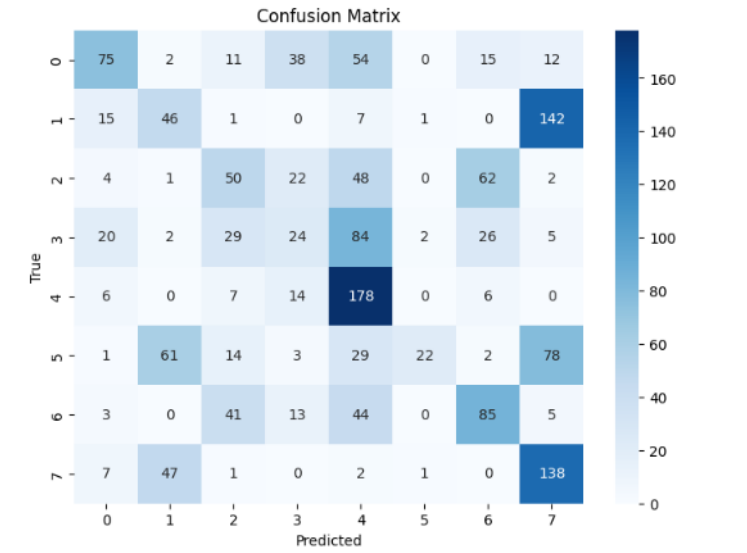



Figure 5.13: Confusion Matrix of ResNet50

## 5.7 Performance Study of Swin Transformer

We trained the Swin Transformer model for 30 epochs. The training accuracy for the Swin Transformer is 79.44%, the test accuracy is 76.71%, the mean f1 score is 0.7684, the mean recall score is 0.7736 and the mean precision score is 0.7773.

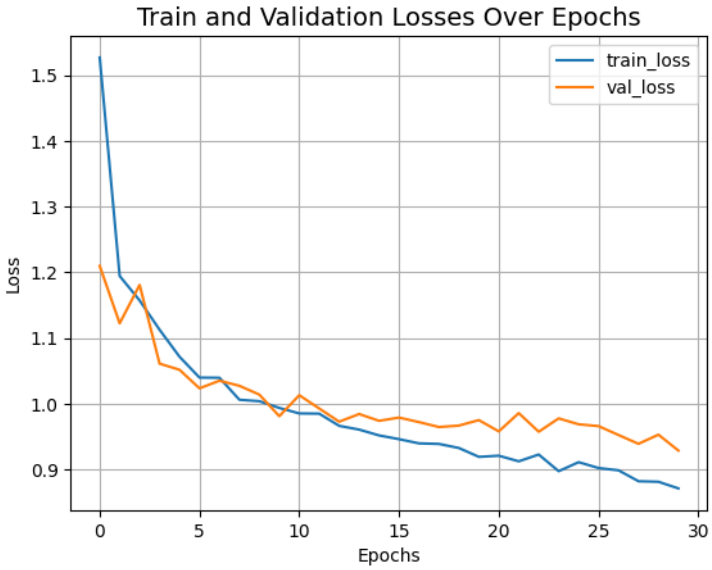



Figure 5.14: Train Validation Loss Graph

We can see in the loss graph, the train loss for the Swin Transformer is 0.8714 and the validation loss is 0.93. Also in the graph the loss curve is decreasing slowly for both train and the validation loss. But in the 30th epoch the train loss is lower than the validation loss. Moreover, the graph of the model accuracy for the Swin Transformer is showing good data. We can see the highest training accuracy is 79.44% and validation accuracy is 76.71%. We can see that the training of the

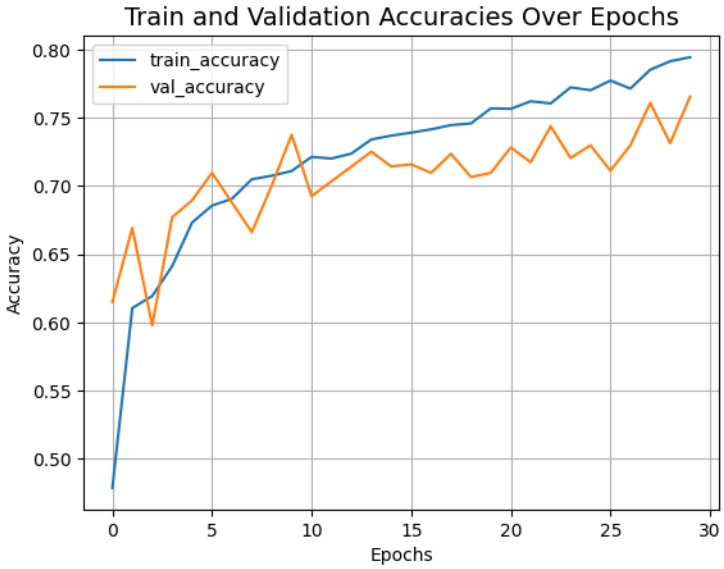



Figure 5.15: Train Validation Accuracy Graph

dataset is gradually going up till the 30th epoch while for the validation accuracy the graph has some ups and downs.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.76 | 0.93 | 0.84 | 202 |
| 1 | 0.86 | 0.64 | 0.73 | 216 |
| 2 | 0.67 | 0.89 | 0.77 | 189 |
| 3 | 0.69 | 0.80 | 0.74 | 197 |
| 4 | 0.75 | 0.65 | 0.69 | 210 |
| 5 | 0.76 | 0.62 | 0.68 | 201 |
| 6 | 0.94 | 0.95 | 0.95 | 195 |
| 7 | 0.79 | 0.71 | 0.75 | 200 |
| accuracy | | | 0.77 | 1610 |
| macro avg | 0.78 | 0.77 | 0.77 | 1610 |
| weighted avg | 0.78 | 0.77 | 0.77 | 1610 |

Table 5.6: Classification Report of Swin Transformer

In the classification report of Swin Transformer the numbers represents different classes such as 0 - dyed lifted polyps, 1- dyed resection margins, 2 - esophagitis, 3 - normal cecum, 4 - Normal pylorus, 5- normal z line, 6 - polyps, 7- ulcerative colitis. From the classification report it can be seen that the precision macro average and weighted avg are both 0.78. The recall and f1 score macro average and weighted average are both 0.77 which is the same.
The row of the confusion matrix represents the true labels, while the column represents the predicted labels. For instance, the number 175 in the first row and first column represents that there were 190 images of dyed lifted polyps that were correctly classified as dyed lifted polyps. The number 0 in the second row and the first column indicates that there were 0 images of dyed resection margins that were incorrectly classified as dyed lifted polyps. For the confusion matrix the classes are

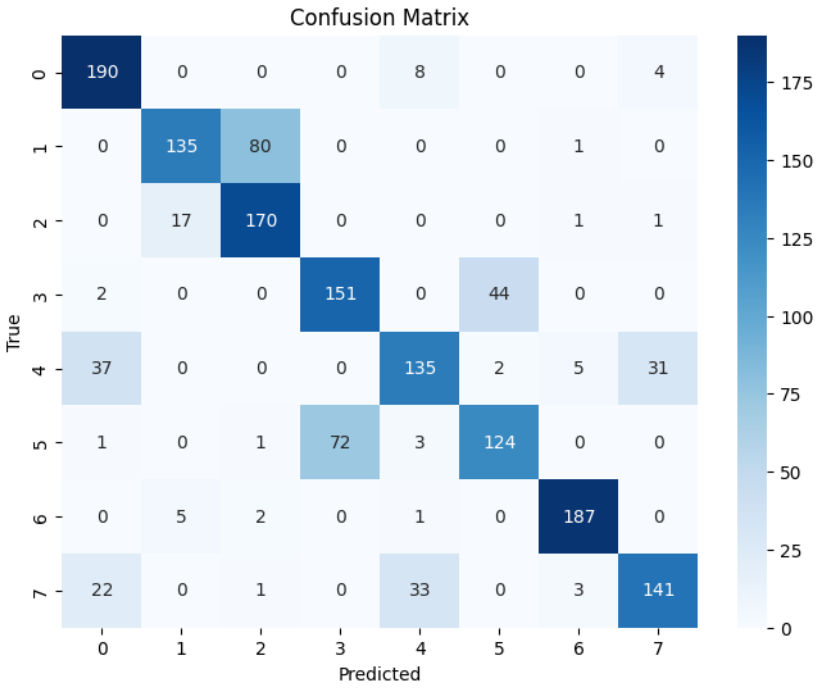



Figure 5.16: Confusion Matrix of Swin Transformer

0 - dyed lifted polyps, 1- dyed resection margins, 2 - esophagitis, 3 - normal cecum, 4 - Normal pylorus, 5- normal z line, 6 - polyps, 7- ulcerative colitis. In Swin Transformer we can see that 2 images are misclassified as dyed resection margins, 6 image are misclassified as esophagitis, 1 image is misclassified as normal cecum, 1 image are misclassified as Normal pylorus, 3 images are misclassified as ulcerative colitis and finally 187 images are classified as polyps.

## 5.8 Performance Study of Ensemble

In order to perform ensemble, we have at first created 26 different combinations using the 5 different transfer learning models ( ResNet-152, ResNet50, MobileNet V2, Res2Net and Inception V3 ). After we have created the different combinations of the transfer learning models we have applied the ensemble techniques blending, weighted averaging and stacking on the different combinations of the transfer learning models. After applying these ensemble techniques we computed the accuracy of these techniques along with the classification metrics such as mean precision, mean recall and mean F1 score. The tables below show the accuracy, mean precision,mean recall and mean F1 score of the blending, weighted average and stacking techniques.

| *Combination name* | *M. Precision* | *M. Recall* | *M. F1* | *Blending Acc* |
|---|---|---|---|---|
| **Combination of all 5 models** | **0.8909** | 0.8388 | **0.8614** | 0.8385 |
| **ResNet-152 and ResNet-50** | 0.8334 | 0.7955 | 0.8119 | 0.7950 |
| **ResNet - 152 and MobileNet-V2** | 0.8653 | 0.8063 | 0.8317 | **0.8662** |
| **ResNet - 152 and Res2Net** | 0.8727 | 0.8226 | 0.8443 | 0.8223 |
| **ResNet - 152 and Inception V3** | 0.8544 | 0.7930 | 0.8194 | 0.7925 |
| **ResNet-50 and MobileNet-V2** | 0.8200 | 0.7846 | 0.7997 | 0.7844 |
| **ResNet - 50 and Res2Net** | 0.8109 | 0.7830 | 0.7932 | 0.7826 |
| **ResNet - 50 and Inception V3** | 0.8031 | 0.7526 | 0.7742 | 0.7521 |
| **MobileNet-V2 and Res2Net** | 0.8619 | 0.8294 | 0.8435 | 0.8291 |
| **MobileNet-V2 and Inception V3** | 0.8588 | 0.7923 | 0.8205 | 0.7919 |
| **Res2Net and Inception V3** | 0.8638 | 0.8122 | 0.8337 | 0.8118 |
| **ResNet-152, ResNet-50 and MobileNet-V2** | 0.8688 | 0.8027 | 0.8311 | 0.8024 |
| **ResNet-152, ResNet-50 and Res2Net** | 0.8734 | 0.8191 | 0.8425 | 0.8186 |
| **ResNet-152, ResNet-50 and Inception V3** | 0.8561 | 0.7817 | 0.8133 | 0.7813 |
| **ResNet152,MobileNet-V2 and Res2Net** | 0.8817 | **0.8417** | 0.8590 | 0.8416 |
| **ResNet-152, MobileNetV2, InceptionV3** | 0.8682 | 0.8082 | 0.8342 | 0.8080 |
| **ResNet-152, Res2Net and Inception V3** | 0.8770 | 0.8303 | 0.8508 | 0.8304 |
| **ResNet-50, MobileNet-V2 and Res2Net** | 0.8730 | 0.8259 | 0.8462 | 0.8254 |
| **ResNet-50, MobileNet-V2 and Inception V3** | 0.8572 | 0.7789 | 0.8120 | 0.7782 |
| **ResNet-50, Res2Net and Inception V3** | 0.8722 | 0.8053 | 0.8333 | 0.8049 |
| **MobileNet-V2, Res2Net and Inception V3** | 0.8782 | 0.8304 | 0.8509 | 0.8298 |
| **ResNet-152, ResNet-50, MobileNet-V2 and Res2Net** | 0.8859 | 0.8370 | 0.8583 | 0.8366 |
| **ResNet-152, ResNet-50, MobileNet-V2 and Inception V3** | 0.8667 | 0.8050 | 0.8316 | 0.8049 |
| **ResNet-152, ResNet-50, Res2Net and Inception V3** | 0.8790 | 0.8293 | 0.8510 | 0.8291 |
| **ResNet-152, MobileNet-V2, Res2Net and Inception V3** | 0.8870 | 0.8381 | 0.8595 | 0.8278 |
| **ResNet-50, MobileNet-V2, Res2Net and Inception V3** | 0.8785 | 0.8209 | 0.8456 | 0.8204 |

Table 5.7: Results of ensemble blending technique

From the blending table it can be seen that the combination of all the 5 transfer learning model has the highest mean precision score of 0.8909 and the highest mean F1 score of 0.8614 and the combination of ResNet152, MobileNet-V2 and Res2Net has the highest mean recall score of 0.8417. It can also be seen that the combination of ResNet152 and MobileNet-V2 has the highest blending accuracy of 0.8662, however this combination has an F1 score of 0.8317. As the F1 score is the harmonic mean of precision and recall, hence we will be taking into account the value of the F1 score instead of the blending accuracy inorder to determine the better performing model. By analyzing the results of the blending table it can be seen that the combination of all the 5 transfer learning models has the highest mean F1 score, hence in this case the combination of all the 5 transfer learning models is the better performing model.

| Combination names | Mean Precision | Mean Recall | Mean F1 | Weighted average accuracy |
|---|---|---|---|---|
| Combination of all 5 models | **0.8985** | 0.7762 | 0.8312 | 0.7758 |
| ResNet-152 and ResNet-50 | 0.8423 | 0.7482 | 0.7911 | 0.7472 |
| ResNet - 152 and MobileNet-V2 | 0.8646 | 0.8014 | 0.8311 | 0.8006 |
| ResNet - 152 and Res2Net | 0.8672 | 0.7957 | 0.8284 | 0.7950 |
| ResNet - 152 and Inception V3 | 0.8476 | 0.7817 | 0.8119 | 0.7807 |
| ResNet-50 and MobileNet-V2 | 0.8375 | 0.7426 | 0.7866 | 0.7422 |
| ResNet - 50 and Res2Net | 0.8043 | 0.7171 | 0.7482 | 0.7174 |
| ResNet - 50 and Inception V3 | 0.8067 | 0.7099 | 0.7519 | 0.7093 |
| MobileNet-V2 and Res2Net | 0.8690 | 0.8058 | 0.8356 | 0.8056 |
| MobileNet-V2 and Inception V3 | 0.8523 | 0.7783 | 0.8132 | 0.7783 |
| Res2Net and Inception V3 | 0.8691 | 0.7716 | 0.8123 | 0.7714 |
| ResNet-152, ResNet-50 and MobileNet-V2 | 0.8712 | 0.7567 | 0.8088 | 0.7559 |
| ResNet-152, ResNet-50 and Res2Net | 0.8733 | 0.7641 | 0.8133 | 0.7634 |
| ResNet-152, ResNet-50 and Inception V3 | 0.8662 | 0.7565 | 0.8058 | 0.7553 |
| ResNet-152, MobileNet-V2 and Res2Net | 0.8785 | 0.8085 | 0.8415 | 0.8081 |
| ResNet-152, MobileNet-V2 and Inception V3 | 0.8716 | 0.7994 | 0.8332 | 0.7988 |
| ResNet-152, Res2Net and Inception V3 | 0.8792 | 0.7993 | 0.8353 | 0.7988 |
| ResNet-50, MobileNet-V2 and Res2Net | 0.8845 | 0.7719 | 0.8229 | 0.7720 |
| ResNet-50, MobileNet-V2 and Inception V3 | 0.8746 | 0.7491 | 0.8059 | 0.7491 |
| ResNet-50, Res2Net and Inception V3 | 0.8726 | 0.7231 | 0.7855 | 0.7229 |
| MobileNet-V2, Res2Net and Inception V3 | 0.8833 | 0.7969 | 0.8363 | 0.7969 |
| ResNet-152, ResNet-50, MobileNet-V2 and Res2Net | 0.8924 | 0.7705 | 0.8253 | 0.7702 |
| ResNet-152, ResNet-50, MobileNet-V2 and Inception V3 | 0.8851 | 0.7658 | 0.8198 | 0.7652 |
| ResNet-152, ResNet-50, Res2Net and Inception V3 | 0.8828 | 0.7641 | 0.8169 | 0.7634 |
| ResNet-152, MobileNet-V2, Res2Net and Inception V3 | 0.8885 | **0.8100** | **0.8459** | **0.8093** |
| ResNet-50, MobileNet-V2, Res2Net and Inception V3 | 0.8891 | 0.7657 | 0.8214 | 0.7658 |

Table 5.8: Results of ensemble weighted averaging technique

| Combination Name | Mean F1-Score | Mean Recall Score | Mean Precision | Stacking Accuracy | Stacking K-FOLD |
|---|---|---|---|---|---|
| Combination of all 5 models | 0.8672 | 0.8670 | 0.8683 | 0.8671 | **0.9431** |
| ResNet-152 and ResNet-50 | 0.8168 | 0.8167 | 0.8184 | 0.8168 | 0.8884 |
| ResNet - 152 and MobileNet-V2 | 0.8454 | 0.8453 | 0.8470 | 0.8453 | 0.9093 |
| ResNet - 152 and Res2Net | 0.8601 | 0.8602 | 0.8621 | 0.8602 | 0.9242 |
| ResNet -152 and Inception V3 | 0.8341 | 0.8342 | 0.8357 | 0.8342 | 0.9268 |
| ResNet - 50 and MobileNet-V2 | 0.8110 | 0.8106 | 0.8120 | 0.8106 | 0.8673 |
| ResNet - 50 and Inception V3 | 0.7745 | 0.7751 | 0.7757 | 0.7752 | 0.8877 |
| ResNet - 50 and Res2Net | 0.8003 | 0.8000 | 0.8045 | 0.8000 | 0.8721 |
| MobileNet-V2 and Res2Net | 0.8571 | 0.8571 | 0.8578 | 0.8571 | 0.9089 |
| MobileNet-V2 and Inception V3 | 0.8356 | 0.8354 | 0.8363 | 0.8354 | 0.9161 |
| Res2Net and Inception V3 | 0.8515 | 0.8515 | 0.8524 | 0.8516 | 0.9304 |
| ResNet-152, ResNet-50 and MobileNet-V2 | 0.8448 | 0.8447 | 0.8463 | 0.8447 | 0.9091 |
| ResNet-152, ResNet-50 and Res2Net | 0.8589 | 0.8590 | 0.8609 | 0.8590 | 0.9240 |
| ResNet-152, ResNet-50 and Inception V3 | 0.8341 | 0.8341 | 0.8357 | 0.8342 | 0.9268 |
| ResNet-152, MobileNet-V2 and Res2Net | 0.8684 | 0.8683 | 0.8700 | 0.8683 | 0.9295 |
| ResNet-152, Res2Net and Inception V3 | 0.8632 | 0.8633 | 0.8642 | 0.8634 | 0.9430 |
| ResNet-152, MobileNet-V2 and Inception V3 | 0.8479 | 0.8478 | 0.8495 | 0.8478 | 0.9310 |
| ResNet-50, MobileNet-V2 and Res2Net | 0.8570 | 0.8571 | 0.8578 | 0.8571 | 0.9089 |
| ResNet-50, MobileNet-V2 and Inception V3 | 0.8361 | 0.8360 | 0.8369 | 0.8360 | 0.9163 |
| ResNet-50, Res2Net and Inception V3 | 0.8522 | 0.8521 | 0.8530 | 0.8522 | 0.9305 |
| MobileNet-V2, Res2Net and Inception V3 | **0.8714** | **0.8714** | **0.8723** | **0.8714** | 0.9361 |
| ResNet-152, ResNet-50, MobileNet-V2 and Res2Net | 0.8678 | 0.8677 | 0.8693 | 0.8677 | 0.9293 |
| ResNet-152, ResNet-50, MobileNet-V2 and Inception V3 | 0.8473 | 0.8472 | 0.8489 | 0.8472 | 0.9309 |
| ResNet-152, ResNet-50, Res2Net and Inception V3 | 0.8638 | 0.8639 | 0.8648 | 0.8640 | **0.9431** |
| ResNet-152, MobileNet-V2, Res2Net and Inception V3 | 0.8678 | 0.8677 | 0.8688 | 0.8677 | **0.9431** |
| ResNet-50, MobileNet-V2, Res2Net and Inception V3 | 0.8708 | 0.8708 | 0.8717 | 0.8708 | 0.9368 |

Table 5.9: Results of ensemble stacking technique

Now, from the weighted average table it can be seen that the combination of all the 5 transfer learning models have a highest mean precision score of 0.8985 and the combination of ResNet152, MobileNet-V2, Res2Net and Inception V3 has a highest mean recall score of 0.8100. The combination of ResNet152, MobileNet-V2, Res2Net and InceptionV3 also has a highest mean F1 score of 0.8459 and at the same time this combination of ResNet152, MobileNet-V2, Res2Net and Inception V3 has a highest weighted average accuracy of 0.8093. By analyzing the results of the weighted average table it can be seen that in this case, the combination of ResNet152, MobileNet-V2, Res2Net and InceptionV3 is performing better compared to the other combination of the models as it has both the highest mean F1 score and the highest weighted accuracy.

Lastly, from the stacking ensemble table it can be seen that the combination of MobileNet-V2, Res2Net and Inception V3 has the highest mean F1 score and highest mean recall score of 0.8714, it has the highest mean precision score of 0.8723 and it also has the highest stacking accuracy of 0.8714. From the table it can be seen that 3 of the combinations among the 26 combinations has the highest stacking K fold accuracy of 0.9431 and those combinations are combination of all the 5 models, combination of ResNet152, MobileNet-V2, Res2Net and Inception V3, combination of ResNet-152, ResNet50, Res2Net and Inception V3. As the mean F1 score is the harmonic mean of precision and recall, hence we will be taking into account the value of the F1 score instead of the other metrics. Therefore in this case of stacking ensemble the combination of the model MobileNet-V2, Res2Net and Inception V3 is performing better than the other combinations of models.

Now if we only consider the mean F1 score of all the ensembling techniques of blending, weighted averaging and stacking it can be seen that the combination of all the 5 transfer learning model has an highest mean F1 score of 0.8614 during blending, the combination of ResNet152, MobileNet-V2, Res2Net and InceptionV3 has a highest mean F1 score of 0.8459 during weighted averaging and the combination of MobileNet-V2, Res2Net and Inception V3 has the highest mean F1 score of 0.8714 during stacking. Among all of these F1 scores, the combination of MobileNet-V2, Res2Net and Inception V3 has the highest mean F1 score overall during stacking ensemble. Hence, in general the stacking ensemble technique performed better compared to the blending ensemble technique and weighted average ensemble technique.

## 5.9 Comparative Study

### 5.9.1 Comparing the performance of the transfer learning models and attention-based model before performing ensemble

| *Model Name* | *M. Precision* | *M. Recall* | *M. F1* | *Train Acc* | *Test Acc* |
|---|---|---|---|---|---|
| **Inception V3** | 0.7757 | 0.7732 | 0.7710 | 0.8904 | 0.7727 |
| **Res2Net** | 0.7844 | 0.7641 | 0.7645 | 0.8633 | 0.7646 |
| **ResNet152** | **0.8197** | **0.8197** | **0.8188** | **0.8954** | **0.8186** |
| **ResNet50** | 0.4296 | 0.3832 | 0.3481 | 0.3948 | 0.3843 |
| **MobileNetV2** | 0.8107 | 0.8101 | 0.8102 | 0.8672 | 0.8099 |
| **Swin Transformer** | 0.7773 | 0.7736 | 0.7684 | 0.7944 | 0.7671 |

Table 5.10: Results of all the models before ensemble

In this table, we demonstrated the value of mean precision, mean recall, mean F1 score, training accuracy, and testing accuracy of every model we used on our dataset before performing the ensemble. Five of these six models (ResNet-152, ResNet-50, MobileNet-V2, Res2Net, and Inception V3 model) are transfer learning models and one (Swin Transformer) is attention-based. From the table, we can figure out that ResNet152 has the highest precision value (0.8197), highest recall value (0.8197), and highest F1 score (0.8188). ResNet152 also has the highest training accuracy (0.8954) and the highest testing accuracy (0.8186). Therefore, by analyzing the table and considering the F1 score, which is the harmonic mean of precision and recall, we can conclude that ResNet152 model is performing better on our dataset compared to other models.

### 5.9.2 Comparing the performance of the transfer learning models and attention-based model after performing ensemble

After the model run and evaluation, we found several differences in the scores. We can see that the scores are different. Firstly, in Inception V3, the F1 score is 0.7710, the training accuracy is 0.8940 and the testing accuracy is 0.7727. Secondly, the Res2Net F1 score is 0.7645, training accuracy is 0.8633 and testing accuracy is 0.7646. Thirdly, the ResNet152 F1 score is 0.8188, training accuracy is 0.8954 and testing accuracy is 0.8186. Moreover, our next model, ResNet50, has a 0.3481 F1 score, 0.3948 training accuracy and 0.3843 testing accuracy. Our next model is MobileNetV2. The F1 score is 0.8102, the training accuracy is 0.8672 and the testing accuracy is 0.8099. So, these results are the individual results of the transfer learning model. Our last model is an attention-based learning model called the Swin Transformer and the F1 score is 0.7684, training accuracy is 0.7944 and testing accuracy is 0.7671. So, these are the results we get from individual model training.

However, we also combined these transfer learning models, and after that, we applied the ensemble method. The three ensemble approaches that we applied were stacking, blending, and weighted averaging. With regard to the ensemble, we used

the five transfer learning models in a total of 26 possible combinations. The stacking ensemble's highest F1 score, 0.8714, is provided by MobileNet-V2, Res2Net, and Inception V3. The same combo also has the highest recall and precision, which are 0.8714 and 0.8723, respectively. However, the mean accuracy on validation sets (K-fold) is the highest for the combination of ResNet-152, ResNet-50, Res2Net, and Inception V3, ResNet-152, MobileNet-V2, Res2Net and Inception V3 and the combination of all five models. The value is 0.9431. As a result, the stacking ensemble with the highest accuracy is the combination of MobileNet V2, Res2Net, and Inception V3. So in the stacking ensemble, we can see that the results have improved a lot in terms of the F1 score. In the individual model running, no model scores as much as this combination in the stacking ensemble.

Additionally, we combined the models using the blending ensemble method. The five transfer learning models combined yielded the highest F1 score and precision in this study. The values are 0.8614 and 0.8909 respectively. The ResNet152, MobileNet-V2, and Res2Net Combination has the highest recall score (0.8417). Finally, the ResNet-152 and MobileNet V2 combo produced a blending accuracy score of 0.8662, which is the maximum possible. From the result, we can see that the highest blending F1 score of all the five models is higher than the individual F1 scores of the models but lower than the highest stacking F1 score. Additionally, the Swin Transformers accuracy provides less precision than the majority of models.

Furthermore, the combination of all five transfer learning models has the highest mean precision score of 0.8985, while the combination of ResNet152, MobileNet-V2, Res2Net, and Inception V3 has the highest mean recall score of 0.8100, according to the weighted average table. ResNet152, MobileNet-V2, Res2Net, and InceptionV3 also have the highest mean F1 score of 0.8459 and the highest weighted average accuracy of 0.8093. By studying the weighted average table results, it is evident that the combination of ResNet152, MobileNet-V2, Res2Net, and InceptionV3 beats the other model combinations in this case, since it has the highest mean F1 score. Meanwhile, we can conclude that the weighted average table highest F1 score is lower than the stacking ensemble but higher than Blending, individual model, and Swin transformer. Therefore, the Stacking Ensemble gives the best performance in terms of F1 scores.

# Chapter 6

# Conclusion

## 6.1 Future Work

In our future work, we aim to develop a robust and advanced model for the early detection of polyps, addressing several key aspects to enhance its effectiveness and applicability. We are going to work on six segments in the future. First of all, in the future, we would like to work on cancer detection using our model. Polyps can be precursors to colorectal cancer, so an integrated approach that can identify not only polyps but also assess their malignancy potential will significantly improve patient outcomes. Secondly, using image masking to enhance the model's ability to discern polyps accurately, we plan to incorporate image masking techniques. By segmenting the polyps within the images, our model can focus on the relevant regions, reducing false positives and detection accuracy.

Thirdly, we need variations when we are detecting the polyps. We acknowledge that there are many existing variants of polyps. To achieve a comprehensive solution, our model will be trained to identify different types of polyps, ensuring that it can adapt to the varying challenges presented by each type. The next step would be that we must utilize all the latest algorithms to stay up-to-date with the latest advancements in computer vision and medical imaging. We will continually explore different algorithms and techniques to enhance the performance of our polyp and cancer detection model.

Moreover, we need to train and evaluate on different datasets to ensure the model's generalizability and reliability. We plan to train and evaluate it on diverse datasets. Collecting data from multiple sources, including different hospitals in Bangladesh, will help us account for variations in imaging equipment and patient demographics. Finally, by expanding our current dataset, our future work involves expanding the current dataset. By collecting a more extensive dataset and a diverse set of polyp and cancer images, we can train a more robust model that can handle a wide range of scenarios and variations.

## 6.2 Conclusion

In essence, the purpose of our thesis study is to determine the type of the transfer learning model that can help to detect polyps accurately from our dataset. To sum up, in our thesis study, we first determined the individual performance of the transfer learning models such as Inception V3, ResNet-152, MobileNet V2, Res2Net, and ResNet50 using our dataset. We have also determined the individual performance of the attention-based model, Swin Transformer, using our dataset. Secondly, we combined all of the transfer learning models, such as Inception V3, ResNet-152, MobileNet V2, Res2Net, and ResNet50, and created different combinations using these models to figure out how the combinations of these models are performing on our dataset and how accurately these different combinations of transfer learning models are able to detect polyps from our dataset. Lastly, we compared the performance of individual transfer learning models with the Swin Transformer based on the architecture of these models and after the ensemble, we compared the performance of these transfer learning models with the Swin Transformer.

In general, in our thesis work, we have shown three different perspectives, where at first we have showed the differences between the performance of the Swin Transformer and the transfer learning models based on the architecture of the models, then we have showed the difference between the performance of the Swin Transformer and the transfer learning model after ensemble and finally we have showed that which combinations of the transfer learning models are giving better performance during ensemble learning.

# Bibliography


[1] Q. Li, W. Cai, X. Wang, Y. Zhou, D. D. Feng, and M. Chen, "Medical image classification with convolutional neural network," in *2014 13th International Conference on Control Automation Robotics  Vision (ICARCV)*, 2014, pp. 844–848. DOI: 10.1109/ICARCV.2014.7064414.

[2] N. Tajbakhsh, S. R. Gurudu, and J. Liang, "Automatic polyp detection in colonoscopy videos using an ensemble of convolutional neural networks," in *2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI)*, 2015, pp. 79–83. DOI: 10.1109/ISBI.2015.7163821.

[3] S. Y. Park and D. Sargent, *Colonoscopic polyp detection using convolutional neural networks*, G. D. Tourassi and S. G. Armato, Eds., Mar. 2016. DOI: 10.1117/12.2217148. [Online]. Available: http://dx.doi.org/10.1117/12.2217148.

[4] P. Sasmal, Y. Iwahori, M. K. Bhuyan, and K. Kasugai, "Classification of polyps in capsule endoscopic images using cnn," in *2018 IEEE Applied Signal Processing Conference (ASPCON)*, 2018, pp. 253–256. DOI: 10.1109/ASPCON.2018.8748527.

[5] Y. Shin, H. A. Qadir, L. Aabakken, J. Bergsland, and I. Balasingham, "Automatic colon polyp detection using region based deep cnn and post learning approaches," *IEEE Access*, vol. 6, pp. 40 950–40 962, 2018. DOI: 10.1109/ACCESS.2018.2856402.

[6] M. Anupama, V. Sowmya, and K. Soman, "Breast cancer classification using capsule network with preprocessed histology images," in *2019 International Conference on Communication and Signal Processing (ICCSP)*, 2019, pp. 0143–0147. DOI: 10.1109/ICCSP.2019.8698043.

[7] J. Kang and J. Gwak, "Ensemble of instance segmentation models for polyp segmentation in colonoscopy images," *IEEE Access*, vol. 7, pp. 26 440–26 447, 2019. DOI: 10.1109/ACCESS.2019.2900672.

[8] H. Zheng, H. Chen, J. Huang, X. Li, X. Han, and J. Yao, "Polyp tracking in video colonoscopy using optical flow with an on-the-fly trained cnn," in *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, 2019, pp. 79–82. DOI: 10.1109/ISBI.2019.8759180.

[9] V. C. Burkapalli and P. C. Patil, "Transfer learning: Inception-v3 based custom classification approach for food images," *ICTACT journal on image and video processing*, vol. 11, no. 01, 2020.

[10] X. Sun, D. Wang, C. Zhang, *et al.*, "Colorectal polyp detection in real-world scenario: Design and experiment study," in *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, 2020, pp. 706–713. DOI: 10.1109/ICTAI50040.2020.00113.

[11] J. Fan, J. Lee, and Y. Lee, "A transfer learning architecture based on a support vector machine for histopathology image classification," *Applied Sciences*, vol. 11, no. 14, p. 6380, Jul. 2021. DOI: 10.3390/app11146380. [Online]. Available: http://dx.doi.org/10.3390/app11146380.

[12] Z. Qian, Y. Lv, D. Lv, *et al.*, "A new approach to polyp detection by preprocessing of images and enhanced faster r-cnn," *IEEE Sensors Journal*, vol. 21, no. 10, pp. 11 374–11 381, 2021. DOI: 10.1109/JSEN.2020.3036005.

[13] *What is image data augmentation?* https://www.picsellia.com/post/image-data-augmentation#:~:text=Image%20data%20augmentation%20is%20the,to%20generate%20new%2C%20augmented%20images, Aug. 2022.

[14] P. Baheti, *A newbie-friendly guide to transfer learning.* https://www.v7labs.com/blog/transfer-learning-guide#h1, May 2023.

[15] *Cancer today — gco.iarc.fr*, https://gco.iarc.fr/today/online-analysis-table?v=2020&mode=cancer&mode_population=continents&population=900&populations=900&key=asr&sex=0&cancer=39&type=0&statistic=5&prevalence=0&population_group=0&ages_group%5B%5D=0&ages_group%5B%5D=17&group_cancer=1&include_nmsc=0&include_nmsc_other=1, [Accessed 11-Sep-2022].

[16] *Colorectal cancer statistics — WCRF International — wcrf.org*, https://www.wcrf.org/cancer-trends/colorectal-cancer-statistics/:%7E:text=Latest%20colorectal%20cancer%20data,of%20colorectal%20cancer%20in%202020, [Accessed 11-Sep-2022].

[17] *Kvasir SEG dataset — kaggle.com*, https://www.kaggle.com/datasets/tanmaydebnath/kvasir-seg-dataset, [Accessed 15-09-2023].

[18] mrgrhn, *ResNet with TensorFlow (Transfer Learning) — medium.com*, https://medium.com/swlh/resnet-with-tensorflow-transfer-learning-13ff0773cf0c, [Accessed 28-07-2023].

[19] D. Mwiti, *A Comprehensive Guide to Ensemble Learning: What Exactly Do You Need to Know — neptune.ai*, https://neptune.ai/blog/ensemble-learning-guide, [Accessed 28-07-2023].

[20] M. Yogi, *Data Augmentation Techniques using OpenCV — medium.com*, https://medium.com/analytics-vidhya/data-augmentation-techniques-using-opencv-657bcb9cc30b, [Accessed 17-09-2023].