COLONOSCOPY IMAGE ANALYSIS USING DEEP LEARNING FOR POLYP DETECTION AND CLASSIFICATION

ABSTRACT

Polyps are abnormal tissue growths in the colon or rectum that can develop into colorectal cancer if not detected and removed in time. Early and accurate detection of polyps plays a key role in preventing cancer progression. Manual analysis of colonoscopy images is often slow, costly, and prone to human error, especially when large volumes of data are involved.

To address this deep learning-based approach for the automatic detection and classification of polyps in colonoscopy images. It uses both Convolutional Neural Networks (CNNs) and MobileNet, a lightweight model designed for efficient performance. The system can detect the presence of a polyp, classify it into one of three types adenoma, hyperplastic, and sessile, estimate its size, determine its severity level, and provide a confidence score.

To make the solution accessible, a simple desktop application is developed using Tkinter, allowing clinicians and researchers to upload colonoscopy images and receive instant results. This work highlights the value of combining deep learning with a user-friendly interface to support faster, more accurate, and reliable analysis of medical images. It can assist in early diagnosis and improve decision-making in both clinical and research environments. The GUI ensures ease of use even for non-technical healthcare staff, enabling broad adoption. Real-time predictions reduce delays in diagnosis and treatment initiation. The tool is lightweight and can run on standard computers, making it viable in rural or low-resource settings.

Automating detection, it reduces clinician workload and minimizes human oversight. This integration of AI and interface enhances screening programs and supports preventive care efforts. This Colonoscopy images anlaysis for polyp detection highlights the value of combining deep learning with an Tkinter interface to support faster, more accurate, and reliable analysis of medical images. By reducing the dependency on manual examination and expert availability, the system can significantly aid in early diagnosis, streamline the clinical workflow, and enhance decision-making in both clinical practice and medical research.

1. INTRODUCTION

Colorectal cancer is one of the most common and deadly cancers worldwide. It often starts as small growths known as polyps in the colon and rectum. Detecting these polyps at an early stage is very important, as early treatment can prevent them from developing into cancer. A colonoscopy is the standard medical procedure used to find and examine these polyps. Manually checking colonoscopy images is time-consuming, requires expert knowledge, and can sometimes result in human error.

To overcome these challenges, deep learning techniques have become powerful tools in medical image analysis and can automate the detection and classification of polyps, making the process faster, more consistent, and more accurate. Two deep learning models are used a custom Convolutional Neural Network (CNN) and MobileNet, a lightweight and efficient architecture. These models are trained to detect the presence of polyps and to classify them into three common types such as adenoma, hyperplastic, and sessile serrated.

The main goal of this work is to compare the performance of CNN and MobileNet to identify which model provides better results. Both models are trained on the same dataset and are evaluated using multiple metrics such as polyp presence accuracy, type classification accuracy, loss, and prediction confidence.

To make the system accessible for practical use, a Tkinter-based desktop application is developed. This user-friendly interface allows clinicians, researchers, or lab staff to upload colonoscopy images and receive real-time predictions, including polyp presence, type, size, severity level, and confidence score. To demonstrates how deep learning can support faster and more accurate colorectal screening and helps identify the most suitable model for real-time medical image analysis.

By combining both deep learning models with an easy-to-use interface, this work bridges the gap between technical development and clinical application. It assists in faster, safer, and more accurate colorectal cancer screening, especially in clinical or rural settings where expert review may not always be available. The system also helps identify the most effective model Convolutional Neural Network (CNN) compared to MobileNetV2 for future real-time medical image analysis applications.

1.1 PROBLEM STATEMENT

Colorectal diseases often begin with small growths called polyps in the colon and rectum. If these are not found early, they can turn into cancer. Checking colonoscopy images by hand takes a lot of time, costs more, and needs trained medical experts. But such experts may not always be available, especially in small clinics or rural areas.

There is a need for a simple, low-cost, and offline system that can automatically detect polyps, identify their type, and estimate serious of the polyps. Two deep learning models like Convolutional Neural Network (CNN) and Mobile Net to build an easy-to-use tool that helps clinicians and researchers quickly analyze colonoscopy images without needing internet or expensive equipment.

This system aims to support quick, reliable, and accurate analysis of colonoscopy images to assist clinicians and researchers in early diagnosis and treatment planning and it contributes to improving patient outcomes, reducing diagnostic delays, and enhancing healthcare accessibility early diagnosis and improve patient outcomes through faster and more accurate detection.

1.2 TOOL DESCRIPTION

Jupyter Notebook

Jupyter Notebook served as the main development platform for building and testing deep learning models for polyp detection and classification using colonoscopy images. It allowed seamless integration of code, output, visualizations, and documentation in a single interface, making the workflow efficient and well-organized. Pandas was used for reading and preprocessing the dataset, including handling image labels and encoding them for model training. NumPy supported numerical operations such as array manipulation and image normalization. Scikit-learn was used for splitting the data into training, validation, and test sets, and for evaluating the model using confusion matrices and classification reports. TensorFlow and Keras were used to design, train, and evaluate both custom CNN and MobileNet models, enabling accurate feature extraction and classification of polyps based on their type. Joblib was implemented to save and load trained models and label encoders, allowing for easy reuse without retraining. Finally, Tkinter was used to create a simple desktop-based GUI, which enables users to upload colonoscopy images and receive real-time predictions about polyp presence, type, and severity. These tools contributed to a complete and effective AI-based medical image analysis system.

1.2.1 Python

Python was the core programming language used throughout this work for developing the polyp detection and classification system. Its simplicity, readability, and vast ecosystem of libraries made it ideal for handling data preprocessing, building deep learning models, and creating user interfaces. Python libraries such as Pandas and NumPy were used for data manipulation, including reading label files, encoding classes, and normalizing image arrays. Scikit-learn helped with data splitting, generating confusion matrices, and evaluating model performance. TensorFlow and Keras, both Python-based deep learning frameworks, were used to construct and train CNN and MobileNet models for detecting the presence of polyps and classifying their types. Joblib was used to save trained models and label encoders for reuse. Additionally, Python's Tkinter library was used to develop a simple and interactive desktop GUI that allows users to upload colonoscopy images and view the predicted polyp presence, type, and severity. Overall, Python provided all the necessary tools to implement, evaluate, and deploy a complete AI-driven medical image analysis system.

PACKAGES

Pandas

Pandas was used for reading and preprocessing image label data from Excel and CSV files. It played a crucial role in organizing the dataset by handling missing values, filtering records, and merging colonoscopy image file paths with their corresponding labels. Pandas was also used to encode key information such as polyp presence (0 or 1) and polyp types adenoma, hyperplastic, and sessile into numerical format, which is essential for training deep learning models. By converting raw data into structured DataFrames, Pandas enabled smooth and efficient preparation of the dataset for input into the Convolutional Neural Network (CNN) and MobileNet models used for polyp detection and classification.

Numpy

NumPy was used to convert colonoscopy images into structured numerical arrays, which are essential for processing by deep learning models. It enabled efficient handling of pixel data and allowed seamless integration of images into the training pipeline. NumPy also played a role in normalizing the pixel intensity values, which helped maintain consistency across the dataset and supported faster convergence during model training. Its array manipulation capabilities were utilized for reshaping image data and preparing it in batches, ensuring compatibility with the input requirements of the Convolutional Neural Network and MobileNet architectures used in this .

Scikit-learn

Scikit-learn was used for essential machine learning utilities, particularly during the data preparation and model evaluation stages. It was used to split the dataset into training, validation, and test sets using the train_test_split function, ensuring proper distribution and unbiased evaluation. After training the deep learning models, Scikit-learn's metrics module was applied to generate classification reports and confusion matrices, which provided detailed insights into the model’s accuracy, precision, recall, and overall performance. These tools helped in assessing how well the Convolutional Neural Network (CNN) and MobileNet models classified the different polyp types and detected their presence effectively.

Keras

Keras is a high-level deep learning API used in this study to build and train CNN and MobileNet models for polyp detection and classification. It provided an easy way to define layers like Conv2D, Dense, and Dropout, and supported multi-output models to predict both polyp presence and type. Keras simplified training using model. compile () and model. Fit(), while integrating smoothly with Tensorflow. It is user-friendly design enabled quick experimentation and integration with GUI and preprocessing tools, making it ideal for this medical image analysis task.

Joblib

Joblib was used to save and reload the trained LabelEncoder, which was used to encode polyp types such as adenoma, hyperplastic, and sessile during model training. By using Joblib, the encoder could be efficiently stored and reused during real-time predictions in the Tkinter GUI without the need for retraining. This ensured consistency in label mapping between the training and deployment phases. Joblib is optimized for handling large NumPy arrays, making it a suitable and fast tool for model components used in machine learning workflows.

Tkinter

Tkinter was used to develop a user-friendly desktop GUI that allows users to upload colonoscopy images and receive real-time predictions. The interface displays whether a polyp is present, identifies its type such as adenoma, hyperplastic, or sessile and estimates its size small, medium, or large. and provides a severity level low, medium, or high. Tkinters built-in components such as buttons, labels, file dialogs, and image display widgets made it easy to create an interactive layout without the need for additional libraries. It was effectively integrated with the trained Keras model and the Joblib-serialized label encoder, enabling smooth offline execution and making the system suitable for quick and accessible image analysis.

These libraries supported the complete colonoscopy image analysis system from data preprocessing and model training to real-time prediction. Pandas, NumPy, and Scikit-learn handled data preparation and evaluation, while Keras powered the deep learning models. Tkinter enabled an offline GUI for easy polyp detection and classification

2. LITERATURE REVIEW

Tan et al. (2023) proposed the Enhanced Scattering Wavelet CNN (ESWCNN) for classifying colorectal polyps in colonoscopy images. The model integrates Scattering Wavelet Transform with CNN to improve feature extraction under uneven lighting. It achieved 96.4% accuracy for three-class and 94.8% for binary classification. ESWCNN outperformed models like ResNet, DenseNet, and EfficientNet. This approach proves effective for enhancing medical image classification accuracy.

Choudhuri et al. (2025) introduced "PolypSeg Track", a unified foundation model for polyp detection, segmentation, classification, and tracking in colonoscopy videos. It employs a conditional mask loss, enabling flexible training on datasets with either segmentation masks or bounding boxes. An unsupervised tracking module links polyp instances across frames without manual heuristics. Pre-trained on natural images using DINOv2, it achieved state-of-the-art results on benchmarks such as Kvasir-SEG and CVC-ClinicDB. This work showcases the potential of foundation models and unsupervised learning for real-time video-based medical analysis.

Chung-Ming Lo, Yu-Hsuan Yeh, Jui-Hsiang Tang, Chun-Chao Chang, and Hsing-Jung Yeh (2022) developed a system for rapid polyp classification using both textural features and DCNNs. The study compared hand-crafted texture methods (Gabor, GLCM) with deep models including AlexNet, Inception-V3, ResNet-101, and DenseNet-201. AlexNet, trained from scratch, achieved the highest accuracy of 96.4%, outperforming both transfer learning and classical texture-based approaches. Their findings suggest that directly trained CNNs on colonoscopy images outperform other methods. This work underscores a reliable automated approach for real-time polyp classification in clinical procedures.

Shin, Qadir, Aabakken, Bergsland & Balasingham (2019) developed an automatic colon polyp detection framework using a region-based deep CNN (Inception-ResNet), tailored to accommodate variations in polyp size, shape, and texture. They introduced data augmentation alongside two novel post-learning strategies Automatic False Positive Learning and Offline False Positive Learning to reduce false detections. The model was trained and tested on colonoscopy video frames, demonstrating enhanced detection accuracy over previous

CNN-based methods. Their work emphasizes the synergy of deep learning and post-processing techniques for reliable polyp detection in clinical settings.

Selvaraj et al. (2024) proposed CRP-ViT, a hybrid model combining ResNet50 and Vision Transformer (VIT) to classify colorectal polyps into hyperplastic, serrated, adenoma, and normal categories. Trained on both real-time and publicly available colonoscopy image datasets, CRP-VIT achieved 96% accuracy on validation and 95.7% on testing. A Gradio-based interface was deployed for clinicians to upload images and receive immediate classification results. The model outperformed conventional CNNs in multiclass classification. This study demonstrates CRP-VIT's readiness for practical, real-time clinical deployment.

Lalinia & Sahafi (2023) applied the YOLO-v8 object detection framework for real-time colorectal polyp detection in colonoscopy images. The model demonstrated high-speed processing with accurate localization of polyps across varied image conditions. YOLO-v8 showed robust performance in identifying polyps from colonoscopy frames, making it suitable for real-time clinical deployment. This approach highlights the effectiveness of modern object detection models for practical use in endoscopic image analysis.

Key difference

This polyp detefocuses on detecting colorectal polyp detection and type classification such as hyperplastic, adenoma, sessile using deep learning. It includes a Tkinter-based interface that enables users to upload colonoscopy images and receive predictions. In contrast, the reviewed research papers primarily focus on classification or segmentation using complex architectures. To addresses the practical deployment interfaces. By providing a simple, offline, and interactive interface it ensures accessibility for clinical professionals with limited technical expertise. It offers a practical and user-friendly deployment interface suitable for clinical use.

3. DOMAIN-HEALTHCARE ANALYTICS

The internship entitled as “COLONOSCOPY IMAGE ANALYSIS FOR POLYP DETECTION USING DEEPLEARNING” comes under the domain of healthcare.

Healthcare analytics is an evolving field of data science that focuses on analyzing vast and complex medical data to enhance clinical decision-making, improve patient outcomes, and streamline hospital operations. The rapid digitalization of healthcare, an enormous amount of data is generated daily from sources like electronic health records, diagnostic images, pathology reports, and wearable devices. Traditional diagnostic methods alone are no longer sufficient to process and interpret such complex data effectively. Hence, healthcare analytics integrates advanced technologies such as artificial intelligence, deep learning, and computer vision to extract actionable insights and support evidence-based medical practices.

This healthcare analytics is applied specifically to colonoscopy image analysis for the early detection and classification of colorectal polyps. Colorectal polyps are tissue growths in the colon or rectum, and some types can become cancerous if left undetected. Early and accurate identification of these polyps is critical in preventing colorectal cancer. The project uses deep learning models such as Convolutional Neural Networks (CNN) and MobileNet to analyze colonoscopy images and identify the presence and type of polyps, including adenoma, hyperplastic, and sessile serrated types. By automating the diagnostic process, the system reduces the workload of clinicians, enhances diagnostic consistency, and enables faster clinical decisions.

To enhance usability, this work incorporates a user-friendly Tkinter-based interface that allows healthcare professionals to upload images and receive real-time predictions regarding polyp presence, type, and severity level. This makes the system practical even for rural and low-resource clinical settings, where access to expert gastroenterologists may be limited. Overall, the application of healthcare analytics in this work demonstrates how AI-driven image analysis can assist in early disease detection, reduce diagnostic errors, and support better outcomes in preventive healthcare.

Disease: Colorectal Polyps and Cancer

Colorectal cancer remains one of the most prevalent and deadly forms of cancer worldwide, contributing significantly to cancer-related mortality. The origin of this disease is typically traced back to colorectal polyps, which are abnormal growths that form on the inner

lining of the colon or rectum. While most polyps are initially benign and non-cancerous, some can gradually develop into malignant tumors over several years, making them a major target for early cancer prevention strategies.

The transition from a benign polyp to colorectal cancer is usually slow and silent, allowing a substantial window of opportunity for early detection and intervention. However, this progression often goes unnoticed because polyps rarely cause symptoms in their early stages. Routine colonoscopy screening is the most effective tool for identifying and removing these polyps before they evolve into cancer. When detected early, removal of precancerous polyps significantly reduces the risk of developing colorectal cancer, highlighting the importance of proactive surveillance.

Colorectal Polyps Types

- **Adenoma Polyps:** These are the most common type of precancerous polyps. They arise from glandular tissue and have a well-documented risk of becoming cancerous over time. Because of this potential, they are usually removed as a precaution, even if they appear harmless during examination.
- **Hyperplastic Polyps:** These polyps are generally considered benign and carry a low risk of turning into cancer. However, their clinical relevance depends on their size and location. Small hyperplastic polyps in the rectum are usually harmless, but larger or proximal ones may require closer follow-up to rule out more serious pathology.
- **Sessile Serrated**: These polyps are flatter and more difficult to detect than others due to their subtle appearance during colonoscopy. Despite their innocuous shape, they are considered high-risk due to their tendency to follow a rapid and distinct molecular pathway to cancer, particularly in the right side of the colon. SSLs often go unnoticed during routine screening unless specifically looked for by trained specialists.

Timely detection and accurate differentiation of these polyp types are essential because they directly influence the patient's treatment plan, follow-up frequency, and overall prognosis. The challenge lies in the fact that early-stage polyps, especially sessile serrated ones, are asymptomatic and visually inconspicuous, requiring sharp clinical expertise and sometimes advanced imaging techniques for detection. As a result, improving the accuracy and efficiency of polyp detection during colonoscopy has become a critical goal in modern gastroenterology and cancer prevention.

Prediction Approach

A deep learning-based image analysis approach can be effectively used for the automated prediction of polyp presence, type classification, and severity estimation using colonoscopy images. Such systems typically leverage Convolutional Neural Networks (CNNs) in combination with lightweight architectures like MobileNetV2, which is pre-trained on large datasets such as ImageNet and fine-tuned for colonoscopy polyp images analysis.

The model performs binary classification to determine whether a polyp is present in an image, and then applies multi-class classification to categorize the polyp into clinically significant types: adenomatous, hyperplastic, or sessile serrated. To assess the urgency of medical intervention, a severity estimation step is included, using factors like polyp size and model confidence to label each case as low, medium, or high severity. Colonoscopy images are usually obtained from datasets HyperKvasir and are preprocessed through resizing, normalization, and label encoding. This predictive framework enables efficient and accurate analysis of colorectal polyps, aiding early diagnosis and supporting improved clinical workflows in both urban and resource-limited settings.

Diagnostic Challenge and Automated Prediction Approach

The manual analysis of colonoscopy images is a time-consuming and expertise-dependent process. It requires careful inspection by skilled gastroenterologists to identify and classify colorectal polyps, which can vary greatly in appearance. In high-volume hospitals or rural healthcare centers, access to specialists may be limited, leading to delays in diagnosis, missed polyps especially flat or subtle ones like sessile serrated and inconsistent outcomes due to human variability. These challenges can contribute to the late detection of precancerous polyps, significantly increasing the risk of colorectal cancer progression.

To address these issues, there is a critical need for a real-time, accurate, and accessible diagnostic support system. The goal is to develop an AI-powered solution that can automatically detect the presence of polyps and classify them into clinically relevant categories. By integrating a trained deep learning model into a user-friendly graphical interface such as one built using Tkinter, healthcare professionals can upload colonoscopy images and receive immediate, reliable predictions. This approach reduces diagnostic workload, enhances consistency, and enables effective screening even in low-resource or underserved settings, ultimately supporting early detection and cancer prevention.

4. DATA MODELLING

Colonoscopy images from the HyperKvasir dataset were used to detect and classify colorectal polyps. There are 741 in total images were labelled, resized, and preprocessed to ensure uniformity. Polyp types were encoded numerically for classification. two deep learning models, CNN and MobileNetV2, were trained to detect the presence of polyps, classify their type adenoma, hyperplastic, and sessile serrated, and estimate severity based on polyp size. This system helps in early, accurate diagnosis and supports timely clinical decision-making**.**

4.1 PROCESS FLOW

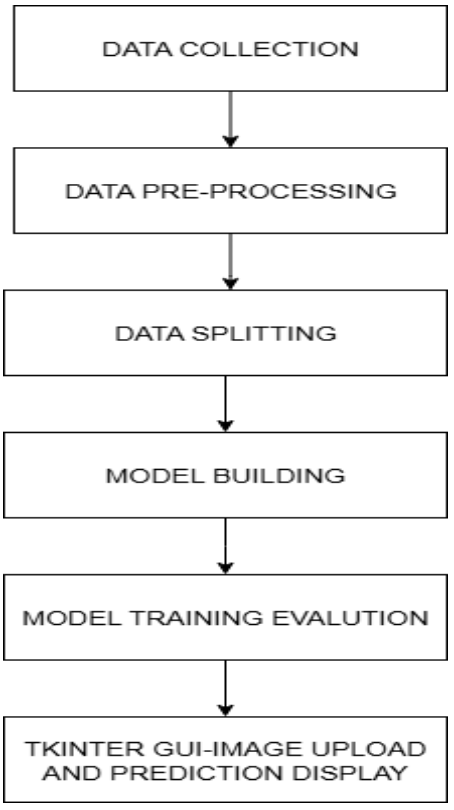


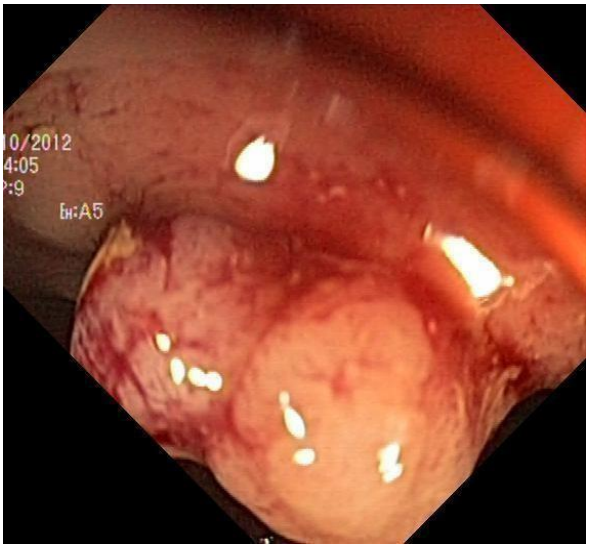
Figure 4.1-Process Flow

Figure 4.1 represents the process flow of the Polyp Detection System using Deep Learning, starting from uploading the image, processing it through the model, and finally displaying the result is shown with polyp presence and type classification.

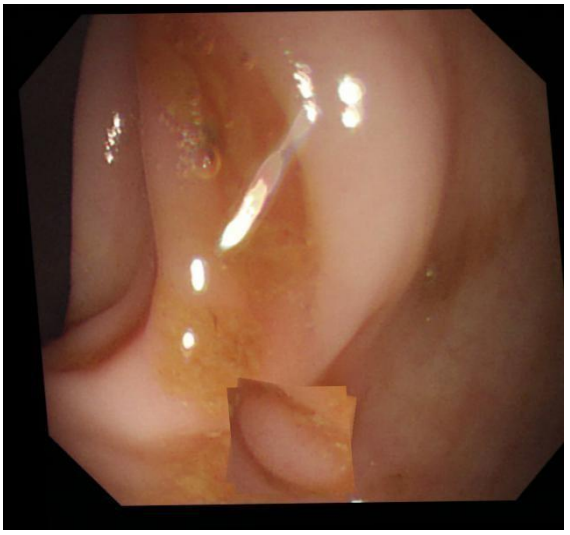
4.2 DATA COLLECTION

The dataset used in colonoscopy image analysis was collected from publicly available the HyperKvasir dataset. It contained colonoscopy polyp images labelled with polyp-related information, categorized by polyp presence and type. It involves gathering the right colonoscopy images in a proper and planned way. Using the correct images helps in understanding past polyp cases, finding patterns, and training a model to detect and classify polyps. This improves the speed, ease, and accuracy of polyp detection.

Sample Dataset Images

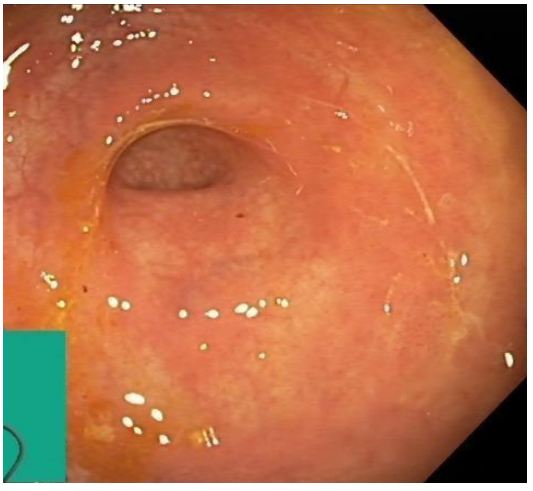


ADENOMA POLYP



HYPERPLASTIC POLYP

SESSILE SERRATED POLY



NO POLYP

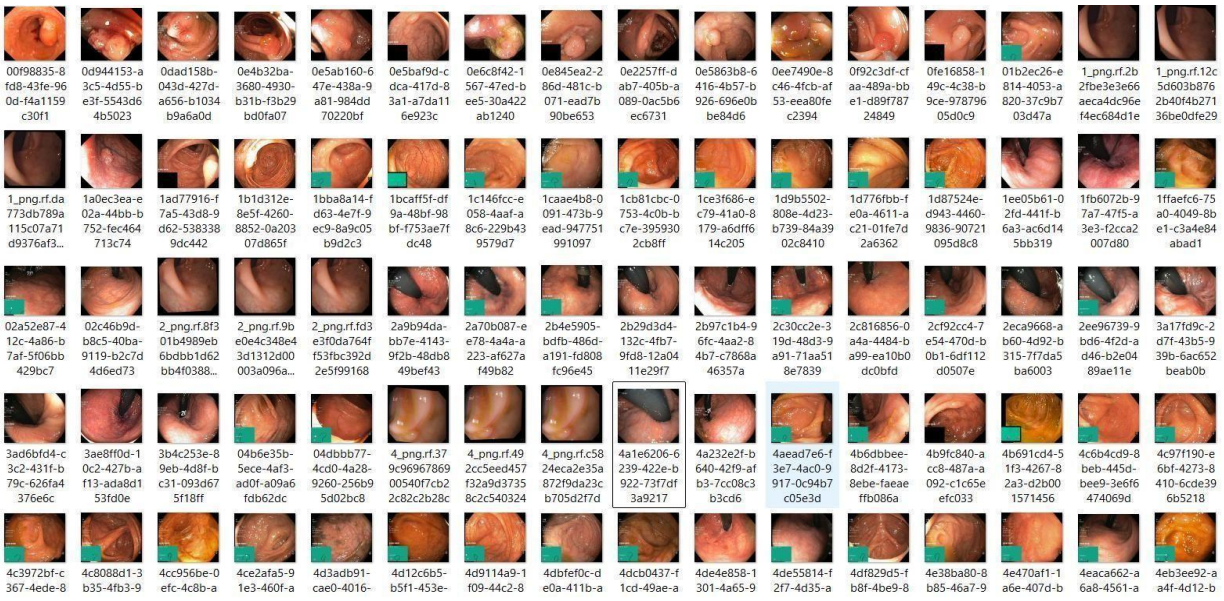


FIGURE 4.2.1 - Polyps Dataset

Figure 4.2.1 shows that colonoscopy images used in the project. The images are labelled with different polyp types such as Adenoma, hyperplastic and sessile serrated. This helps the model learn to find and identify polyps correctly.

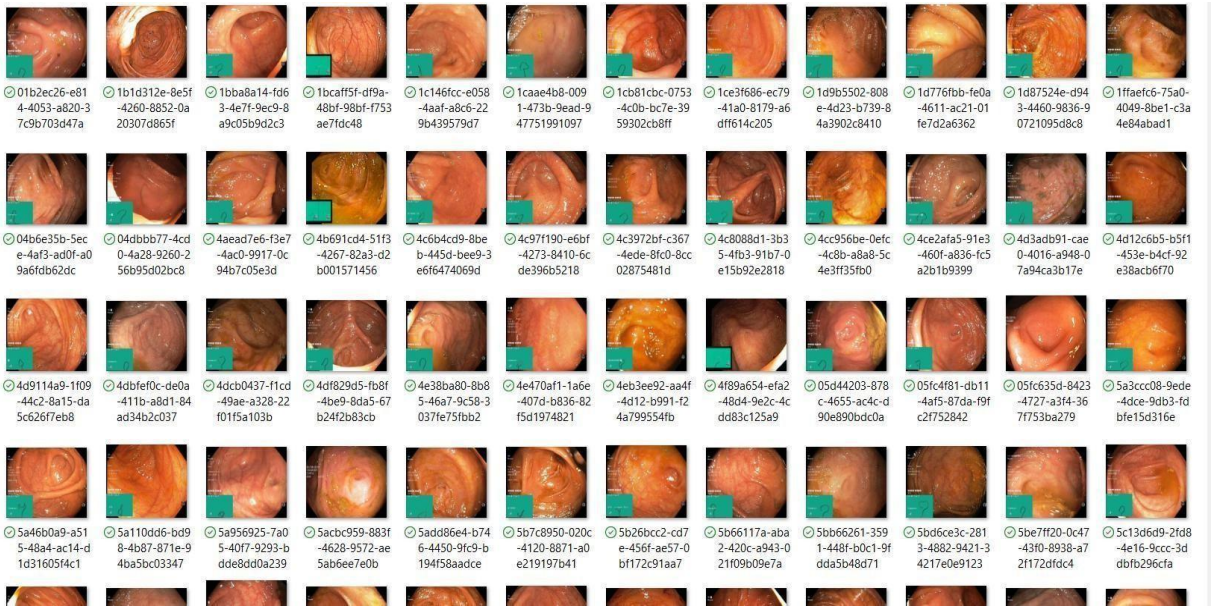


FIGURE 4.2.2 - No **Polyp Dataset**

Figure 4.2.2 shows that colonoscopy images labeled as no polyp. These normal images help the model learn to distinguish between healthy colon tissue and abnormal growths, improving the accuracy of polyp detection

4.3 DATASET DESCRIPTION

Colonoscopy images collected from the HyperKvasir dataset to train a deep learning model. These images show the inside of the colon and include both polyp and non-polyp cases. There are 741 images in total, and each one is labelled with details like the part of the organ, whether a polyp is present, and the type of polyp such as adenoma, hyperplastic, sessileserrated, and normal. Before training, all images are resized to the same size so the model can read them properly. These labelled images help the model learn to find and classify polyps more accurately.

ATTRIBUTES

FILENAME

The Filename attribute helps in identifying and organizing each colonoscopy images in the dataset.

ORGAN

The Organ attribute indicates which part of the body the image is taken from. In this dataset, it refers to the lower GI tract, which includes the colon and rectum part of the digestive system.

CLASSIFICATION

Classification describes the medical condition found in the image, such as pathological findings that indicate abnormal tissues or diseases.

POLYP

The polyp attribute indicates whether a polyp is present in the image, where a value of 1 means a polyp is present and a value of 0 means no polyp is present.

TYPE

The Type attribute indicates the kind of polyp found in the image. Adenoma is a type that can develop into cancer. Hyperplastic is usually harmless. Sessile is flat and harder to detect. Normal means the tissue is healthy and has no polyp. These classifications help train the model to differentiate between risky and non-risky cases. Accurate labeling ensures better diagnostic performance during prediction.

4.4 DATA PRE-PROCESSING

4.4.1 IMPORTING LIBRARIES

Python libraries are essential for building the entire system, as they provide modular, reusable tools that streamline development, particularly in the fields of machine learning, deep learning, and medical image analysis. These libraries played a crucial role in tasks such as loading colonoscopy images, preprocessing them, training deep learning models, and predicting the outcomes.

Pandas: Pandas is used for reading, managing, and processing label files. It helped associate each colonoscopy image with the correct polyp type adenoma, hyperplastic, sessile serrated, normal and structure the dataset efficiently for training and testing.

NumPy: Used to handle image data in array format and perform numerical computations like resizing, normalizing pixel values, and reshaping data for deep learning input.

OS: Enabled dynamic access and management of folders and image files. It helped automate the process of iterating through directories and organizing datasets.

Scikit-learn: Employed for preprocessing tasks such as splitting the dataset into training, validation, and testing sets. LabelEncoder from this library converted categorical polyp type labels into numeric form for model compatibility.

TensorFlow: Served as the backend for building and training deep learning models. It powered both CNN and MobileNetV2 architectures used for polyp detection and classification.

Keras: Keras offered a user-friendly API to create, compile, and train deep learning models with multiple convolutional and pooling layers. It also supported model evaluation and prediction on test images.

Matplotlib & Seaborn: Used to visualize model performance, including accuracy, loss curves, which helped in understanding the classification results and diagnosing errors.

Tkinter: Used to create a simple GUI where users can upload colonoscopy images and get predictions on polyp presence, type adenoma, hyperplastic, sessile, and severity low, medium, high. It makes the system easy to use for quick diagnosis.

4.4.2 MODEL TRAINING PARAMETERS

Image Resizing

All colonoscopy images are resized to 128×128 pixels to maintain consistency in input dimensions. This preprocessing step ensures efficient model training, prevents dimension-related errors, and supports MobileNet in accurately detecting and classifying polyps with reduced computational load.

Normalization

Normalization is performed by scaling image pixel values from the original range of [0, 255] to [0, 1], achieved by dividing each pixel by 255. This step is essential for deep learning models like CNNs and MobileNet, as it ensures consistent input distribution, stabilizes the training process, and accelerates convergence. Normalization also reduces computational load and helps prevent issues like vanishing gradients. In MobileNet, which uses depthwise separable convolutions, normalized inputs are particularly important to maintain model performance and computational efficiency during both training and inference.

Epochs

The model is trained for 10 epochs, meaning the entire training dataset is passed through the model ten times. Each epoch allows the model to iteratively learn from the data and refine its weights to improve prediction accuracy. Training over multiple epochs helps the model capture complex patterns in colonoscopy images, such as identifying polyp shapes and textures and using too many epochs may lead to overfitting, so 10 epochs offer a balanced trade-off between learning and generalization.

Batch Size

A batch size of 32 is used, which means the model processes 32 images at a time before updating its weights. This approach reduces memory usage and speeds up training compared to processing the entire dataset at once. It also stabilizes gradient updates and allows efficient utilization of system resources. For CNN and MobileNet, a batch size of 32 ensures good convergence while maintaining smooth and efficient training, especially when working with high-resolution medical images.

Loss Function

This categorical cross-entropy is used as the loss function because the task involves multiclass classification of polyp types. It calculates how far the predicted probabilities are from the true class labels and penalizes incorrect predictions more strongly. This loss function helps the CNN and MobileNet models learn effectively by minimizing classification errors. It is widely used in deep learning tasks involving softmax outputs, ensuring stable and accurate training performance.

Adam optimizer

The Adam optimizer is used to train the deep learning model. Adaptive Moment Estimation is highly efficient and combines the advantages of using extensions of stochastic gradient descent AdaGrad. It computes adaptive learning rates for each parameter, making it particularly suitable for handling complex medical images. Given the nature of the dataset and the architecture CNN and MobileNet, Adam ensures faster convergence, better stability during training, and improved performance in detecting and classifying polyps. Its ability to adapt learning rates dynamically helps prevent overshooting and allows the model to learn effectively from imbalanced data.

4.4.3 DATA SPLITTING

The labelled dataset of colonoscopy images is strategically divided into three subsets to optimize model training, validation, and evaluation. This ensures that the model generalizes well to unseen data and prevents issues like overfitting.

- **Training Set (70%):** Used to train the CNN and MobileNet models to detect the presence of polyps and classify them into three types such as adenoma, hyperplastic, and sessile serrated. This 518-image set helps the model learn the visual patterns required for accurate prediction.
- **Validation Set (15%):** Used during 111 images training to monitor the model's performance on unseen data. It helps fine-tune the model and avoid overfitting by adjusting parameters based on validation accuracy.
- **Test Set (15%):** Reserved for final evaluation after training is complete. It provides 111 images an unbiased assessment of the model’s real-world performance on new, unseen colonoscopy images and providing a final measure of its diagnostic reliability.

4.5 MODEL ARCHITECTURE AND BUILDING

This colonoscopy images analysis for polyp detection used CNN and MobileNetV2 models to find and classify polyps for detecting the presence of polyps and classifying their types adenoma, hyperplastic, or sessile serrated from colonoscopy images. The choice of CNN and MobileNetV2 were selected for their ability to understand image features and give accurate results.

4.5.1 Convolutional Neural Network

Convolutional Neural Network (CNN) is employed to detect the presence of polyps and classify their types in colonoscopy images. CNNs are highly effective for image-based tasks because they automatically extract spatial features such as edges, shapes, and textures, which are essential for distinguishing between different types of colorectal polyps. The model architecture includes multiple convolutional layers for feature extraction, max pooling layers for dimensionality reduction, a flattening layer to convert the data into a one-dimensional vector, and fully connected dense layers to make predictions. A dropout layer is also included to prevent overfitting by randomly deactivating neurons during training. The CNN produces two outputs one for binary classification to detect whether a polyp is present, and another for multi-class classification to identify the type of polyp adenoma, hyperplastic, or sessile serrated. This structure enables the model to learn complex patterns from medical images and aids in early and accurate.

Convolutional Neural Network architecture contains the following layers

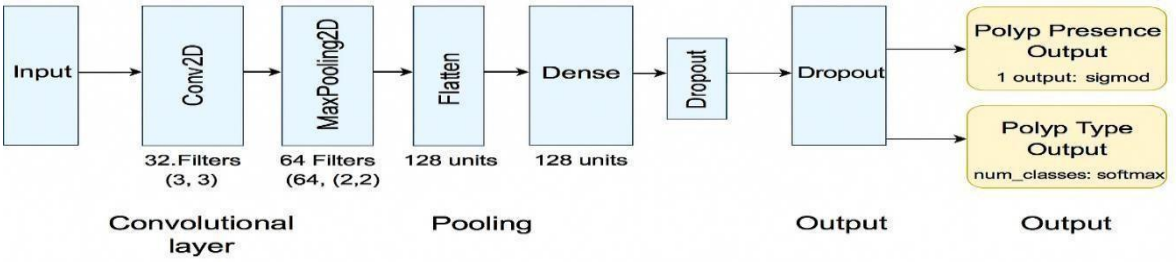


Figure 4.5.1 – Convolutional Neural Network Architecture

Figure 4.5.1 represents the architecture of the Convolutional Neural Network (CNN) used for polyp detection and classification. The model begins with an input layer that accepts colonoscopy images, followed by convolutional layers with filters to extract low- and high-level features. MaxPooling layers are applied to reduce spatial dimensions while preserving key information. The extracted features are then flattened and passed through dense layers with dropout regularization to prevent overfitting. Finally, the network branches into two outputs one using a sigmoid activation for binary classification of polyp presence, and another using softmax activation for multi-class classification of polyp type.

Input Layer

The Input Layer is the first layer in the Convolutional Neural Network (CNN) model, and it is designed to receive colonoscopy images that have been resized to a fixed dimension of 128×128 pixels with 3 color channels (RGB). This means each input image is represented as 3-dimensional array 128 pixels in height, 128 pixels in width, and 3 values per pixel for red, green, and blue color intensities. This standardization ensures all input images have the same size and structure, which is essential for efficient convolutional operations. The input layer acts as a gateway, preparing the data for further processing in the Convolutional Neural Network.

Conv2D (Convolutional 2D)

- Conv2D (Convolutional 2D) layers are fundamental for detecting patterns in image data. The first convolutional layer, Conv2D(32, (3,3), activation='relu', applies 32 learnable filters of size 3×3 to the input colonoscopy image. These filters slide across the image and extract basic low-level features such as edges, color gradients, and textures, which are especially helpful in detecting the polyps.
- The second layer, Conv2D (64, (3,3), activation='relu', builds on this by applying 64 filters to the output of the previous layer. It captures more complex and abstract features like the shape, structure, and patterns within the polyps.
- The third convolutional layer, Conv2D(128, (3,3), activation='relu', is used to identify even more detailed and high-level patterns. This helps the model understand the subtle differences between various polyp types (adenoma, hyperplastic, sessile) and distinguish them from normal tissue.

These stacked Conv2D layers allow the CNN to learn a hierarchy of features from simple edges to intricate textures and patterns which are crucial for accurate polyp detection

and classification in colonoscopy images.

MaxPooling Layer

Three MaxPooling2D (2,2) layers are strategically placed after each convolutional block to reduce the spatial dimensions of the feature maps while preserving key information.

- **First MaxPooling Layer:** The first MaxPooling2D(2,2) follows a Conv2D(32, (3,3)) layer and helps minimize computational complexity while emphasizing basic polyp features such as edges and contours.
- **Second MaxPooling Layer:** The second MaxPooling2D(2,2) comes after a Conv2D(64, (3,3)) layer and further shrinks the feature map, suppressing irrelevant pixel-level noise and helping the model learn texture and shape patterns.
- **Third MaxPooling Layer:** The third MaxPooling2D(2,2), applied after Conv2D(128, (3,3)), extracts the most abstract and high-level features, preparing the data for the fully connected layers. These pooling operations enhance model efficiency, reduce overfitting, and improve the CNN's ability to generalize across different colonoscopy images.

Flatten Layer

This flatten layer 2D feature maps generated by the final Conv2D and MaxPooling2D layers are transformed into a 1D feature vector. This step is essential because dense fully connected layers require input in a single vector format. By flattening the spatial data, the model transitions from spatial feature extraction to high-level reasoning and classification, enabling it to detect polyp presence and classify the type effectively in colonoscopy images.

Droupout Layer

The Dropout layer, with a dropout rate of 0.5, is applied after the dense layer to randomly deactivate 50% of the neurons during training, helping prevent overfitting and enhancing the model ability to generalize to new colonoscopy images.

Sigmoid Activation Layer

The Sigmoid activation output layer is used for binary classification to detect polyp presence, producing a probability between 0 and 1 where values closer to 1 indicate higher confidence of polyp detection.

Softmax Activation Layer

The Softmax activation output layer handles multi-class classification by predicting the specific polyp type such as adenoma, hyperplastic, sessile serrated based on the learned features. This dual-output architecture ensures that the system simultaneously provides both polyp presence (Yes/No) with a confidence score and the most probable polyp type, making it a comprehensive tool for real-time diagnostic support.

Output Layer

The CNN model ends with two specialized output branches one detects polyp presence using a sigmoid-activated neuron, while the other classifies the polyp type using softmax activation. These layers provide a dual prediction confirming if a polyp exists and identifying its type enabling efficient and accurate medical image analysis.

4.5.2 MOBILENET

MobileNetV2 is employed as a transfer learning-based feature extractor to analyze colonoscopy images for polyp detection and classification. Pre-trained on the large-scale ImageNet dataset, MobileNetV2 effectively transfers its learned visual features such as edges, textures, and object shapes to the medical domain. It is adapted to perform two key tasks: detecting the presence of polyps binary classification to identify polyp or no polyp and classifying the type of polyp multi-class classification: adenoma, hyperplastic, sessile serrated. To enhance its predictive power, additional layers such as GlobalAveragePooling2D, Dense, and Dropout are added on top of the pre-trained base. These layers help fine-tune the model for improved accuracy and generalization.

MobileNetV2 is chosen for its lightweight architecture, fast inference speed, and high accuracy, making it especially suitable for real time medical applications. After training on labelled images, the model is tested on new colonoscopy samples to evaluate how well it identifies polyp presence and type, while minimizing prediction error. This approach proves to be an efficient and reliable solution for automated colonoscopy image analysis for polyp detection.

MobileNet architecture contains the following layers

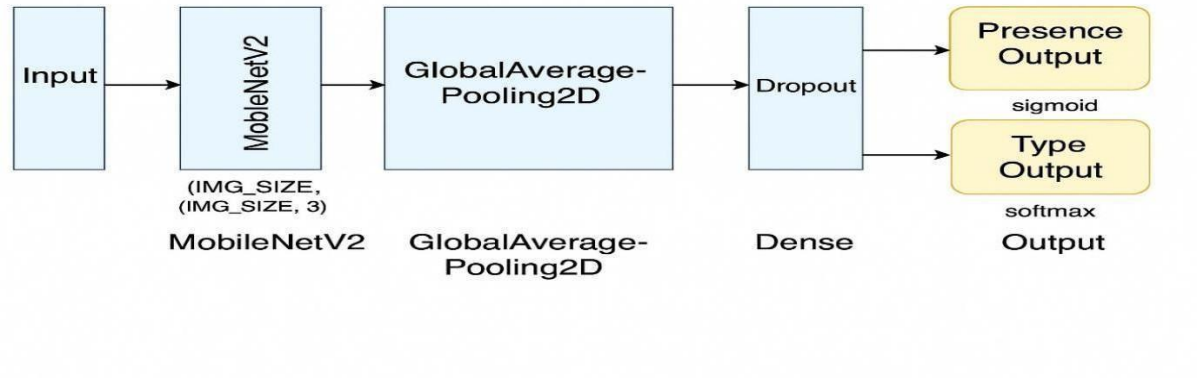


Figure 4.5.2 – MobileNetV2 Architecture

Figure 4.5.2 presents the architecture of the MobileNetV2-based model applied for efficient polyp detection and classification. The system starts with an input layer that feeds into the pre trained MobileNetV2 network, which acts as a feature extractor using depthwise separable convolutions to reduce computational complexity. The extracted features are then passed through a Global Average Pooling layer to generate compact feature vectors, followed by a dense layer with dropout to enhance generalization. The model outputs two predictions: a sigmoid-activated output for polyp presence detection and a softmax-activated output for polyp type classification, enabling accurate and resource-efficient diagnosis.

Input Layer

The input layer in the MobileNetV2-based model is responsible for receiving colonoscopy images that have been resized to a fixed size of 128×128×3. Each image consists of 128 pixels in height, 128 pixels in width, and 3 color channels (RGB), which together form a standardized 3D input tensor. This consistent input shape ensures compatibility with the MobileNetV2 base model and allows efficient feature extraction. The input layer serves as the foundation for the rest of the model by passing clean, uniformly sized images to the pre-trained MobileNetV2 network for further processing.

MobileNetV2 Base Model

This layer loads the MobileNetV2 architecture pretrained on ImageNet, excluding its top classification layers (include_top=False). It acts as a deep feature extractor, automatically identifying relevant visual features such as polyp textures, edges, and shapes. The pretrained layers are either frozen or fine-tuned depending on the requirement, helping transfer generalized visual knowledge to the specific medical imaging task of polyp detection and classification.

GlobalAveragePooling2D

GlobalAveragePooling2D Layer takes each 2D feature map output from the last MobileNetV2 convolutional layer and averages each of the 1280 feature maps, reducing it to a 1×1×1280 This significantly reduces the number of parameters compared to flattening would produce 20,480 parameters, helping prevent overfitting and enabling efficient polyp feature summarization for final classification and it improves accuracy, efficiency, and generalization, especially when working with colonoscopy polyp images where overfitting must be avoided.

Dense layer

The fully connected Dense layer with 128 neurons and ReLU activation is added. This layer transforms the spatial features extracted by the previous convolutional layers into a compact, high-level representation. The ReLU activation introduces non-linearity, allowing the model to learn complex patterns and relationships essential for distinguishing different types of polyps. By limiting the layer to 128 neurons, the network maintains a balance between capturing important information and keeping the model computationally efficient.

Dropout layer

A Dropout layer with a rate of 0.5 is included immediately after the Dense layer to reduce overfitting. During training, this layer randomly deactivates 50% of its neurons in each iteration, forcing the model to develop more robust and generalized feature representations. This prevents the network from relying too heavily on specific neurons, which improves its ability to generalize well to new, unseen colonoscopy images. Dropout thus acts as an effective regularization technique that enhances the model’s reliability and accuracy.

Output Layer

- **Sigmoid activation Layer:** The Output Layer of the model is designed with two distinct branches to handle different tasks. The first branch employs a Sigmoid activation layer for binary classification, which determines the presence or absence of a polyp in the colonoscopy image. This layer outputs a probability value ranging between 0 and 1, where values closer to 1 indicate a higher likelihood of a polyp being present, and values closer to 0 indicate its absence. The advantage of using Sigmoid here is that it normalizes the output into a probability that can be easily thresholded to make a clear decision about polyp presence**.**
- **Softmax Output Layer:** The second branch uses a Softmax output layer to perform multi-class classification, which identifies the type of polyp when one is detected. Specifically, the Softmax activation assigns probabilities across the possible classes Adenoma, Hyperplastic, and Sessile ensuring that the sum of all class probabilities equals 1. The class with the highest probability is selected as the predicted polyp type. This makes Softmax suitable for mutually exclusive classes, allowing the model not just to detect a polyp but also to categorize it into clinically significant types, thereby adding diagnostic value.

5. MODEL EVALUATION AND OUTPUT INTEGRATION

5.1 Model Evaluation and Comparsion

The CNN model was developed using multiple convolutional and pooling layers. It was trained on the preprocessed image dataset and tested using a separate test set. The model achieved an accuracy of 89.19% in detecting polyp presence and 81.98% in classifying polyp types. While the CNN model performed well, there were limitations in accuracy and generalization.

The second model, MobileNetV2, was implemented using transfer learning. This model used a pre-trained architecture and was fine-tuned with the same dataset. MobileNetV2 achieved higher accuracy, with 98.20% in detecting polyp presence and 83.78% in polyp type classification. It also required less training time and showed better consistency during validation.

MobileNetV2 showed better performance than CNN in detecting and classifying polyps. It provided higher accuracy, faster training, and more stable results. Thus, it was chosen as the final model for effective and reliable prediction.

Model Evaluation Table

ALGORITHM	CONVOLUTIONAL NEURALNETWORK	MOBILENET
POLYP PRESENCE ACCURACY	89.19%	98.2%
POLYP TYPE ACCURACY	81.98%	83.78%

Table 5.1 - Model Evaluation

The table 5.1 shows the performance of two models, CNN and Mobile Net, for detecting polyps and classifying their types. CNN gave 89.19% accuracy for polyp presence and 81.98% for polyp type. Mobile Net performed better with 98.2% accuracy for polyp presence and 83.78% for polyp type. This means Mobile Net gave more accurate results than CNN.

5.2 OUTPUT INTEGRATION

The MobileNetV2 model was integrated into a Tkinter GUI for real-time polyp prediction. Users can upload a colonoscopy image and view the polyp result easily.

TINKER GUI-IMAGE UPLOAD AND PREDICTION DISPLAY

The Tkinter GUI is a simple window that helps users upload an image and get results easily. When the user clicks the "Upload Image" button, they can choose a colonoscopy image from their computer. The image is then shown on the screen. After that, the program checks the image using a trained model to see if a polyp is present. If a polyp is found, it shows the type of polyp and how serious it is. If there is no polyp, it shows No Polyp and says the image is normal. This GUI is easy to use and shows clear results without needing to know any coding.

Input screen

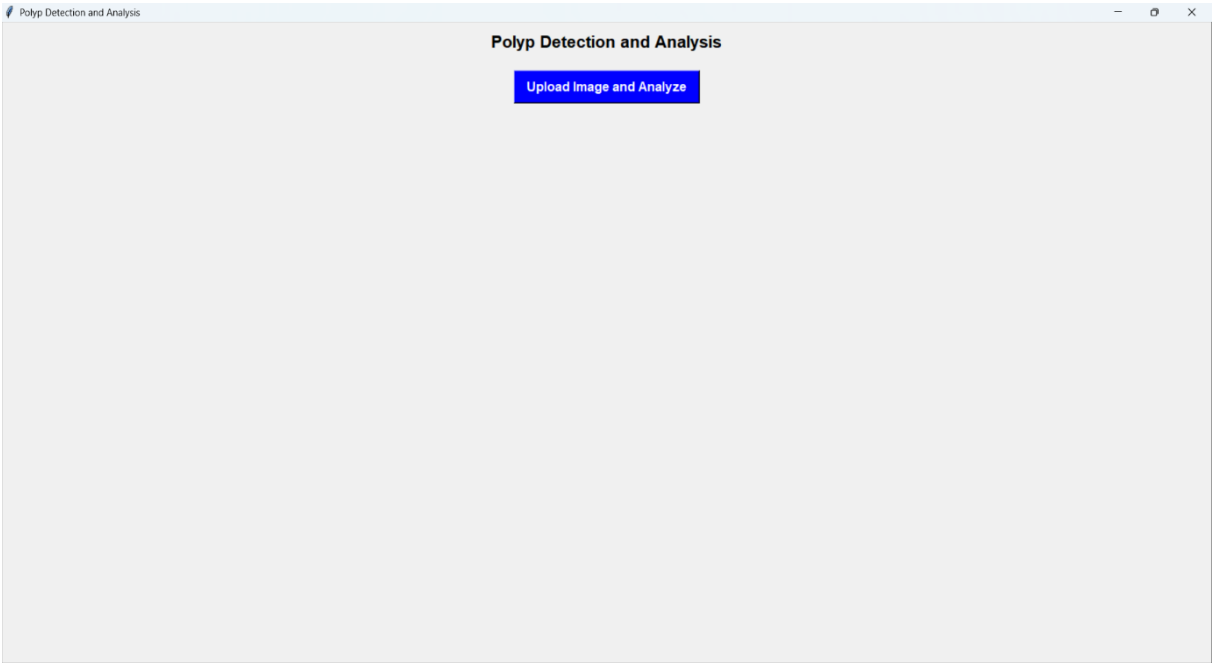


FIGURE 5.2.1 Input screen

FIGURE 5.2.1The figure explains the full process from image upload to result, helping users understand the condition clearly and take the next steps easily.

User: The user uploads colonoscopy images to detect the presence of polyps and classify their types using the deep learning model.

Upload Image: A use case where the user uploads colonoscopy images to the system for analysis.

System: Represents the polyp detection and classification system, which processes the uploaded images and provides results indicating polyp presence and type.

Image-Based Prediction Output

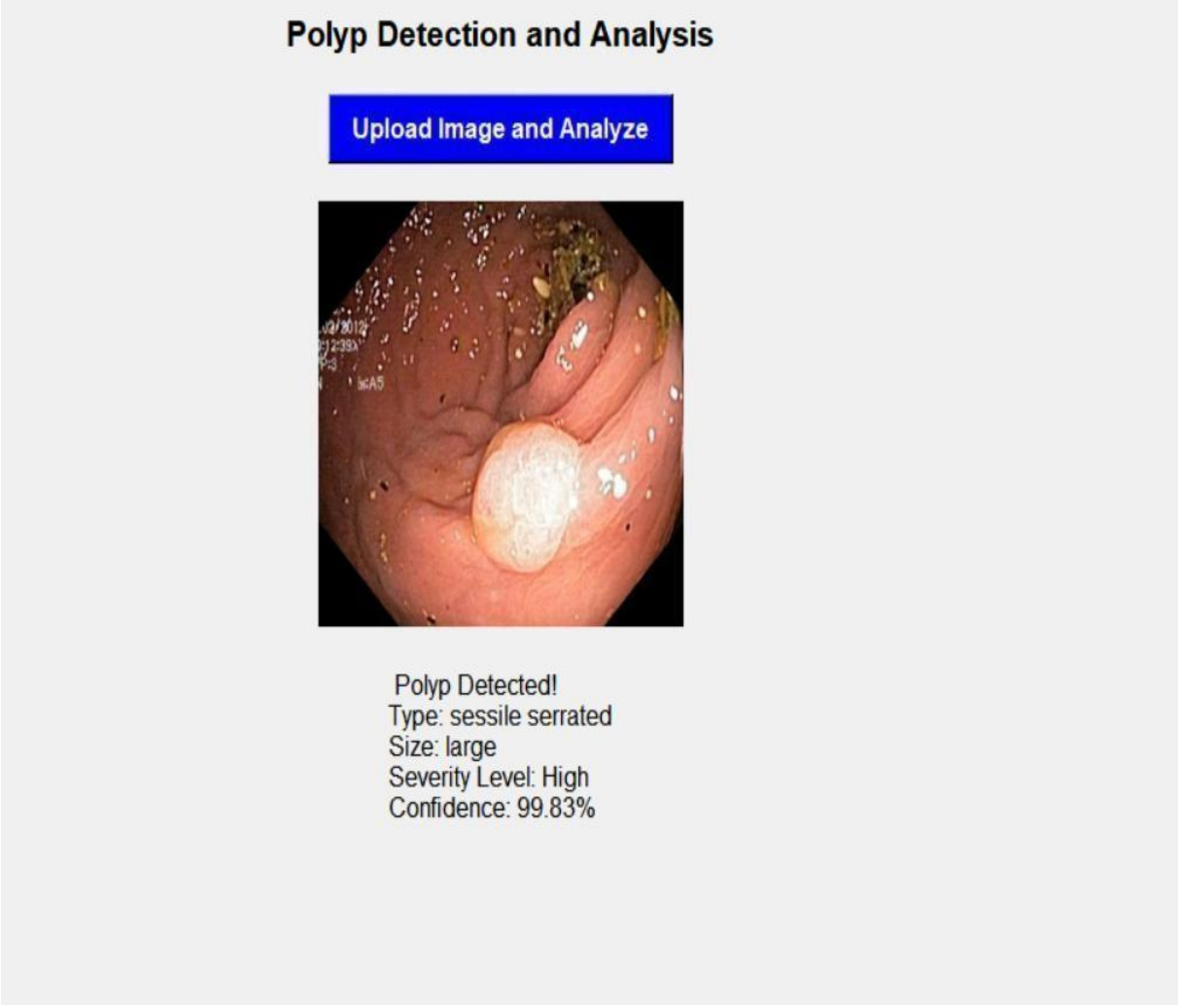


FIGURE 5.2.2 Polyp Detection and Analysis

FIGURE 5.2.2 The Image shows the result after the image is uploaded. In this example, it has detected a polyp. The polyp type is sessile serrated the size is large, and the severity level is high. The model is 99.83% confident about this result. This helps users understand the condition quickly and make decisions easily.

Image-Based Prediction Output

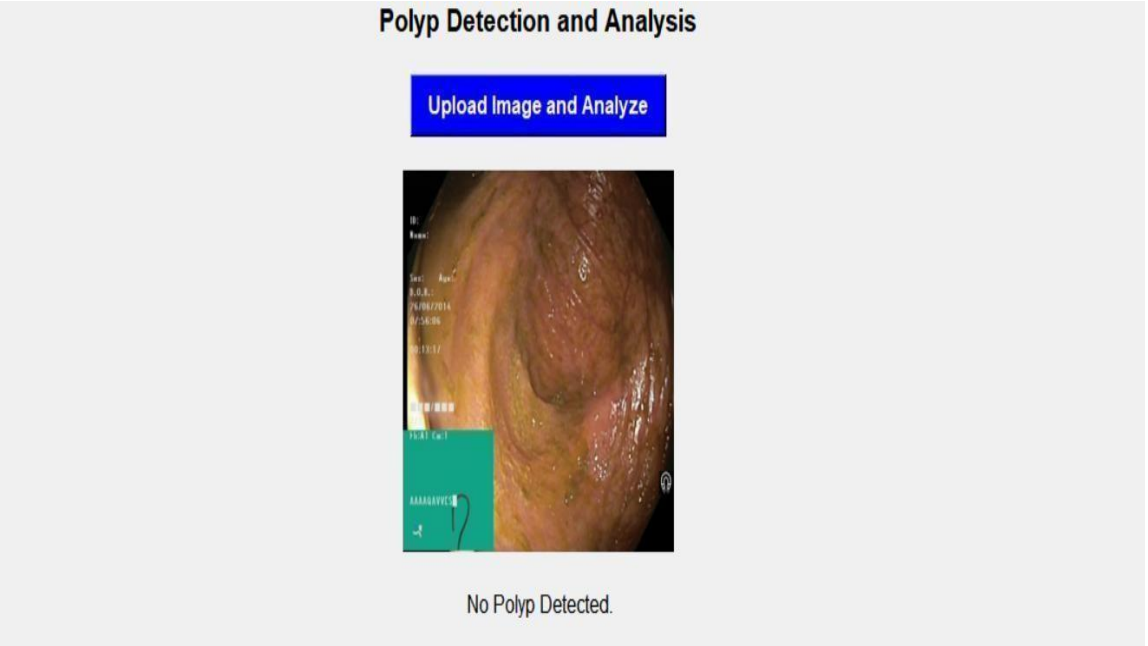


FIGURE 5.2.3 No Polyp Detection and Analysis

FIGURE 5.2.3 The Image shows the result. In this case, no polyp is detected. The image is considered normal, and there is no severity level shown. This gives users a clear idea that the colon appears healthy in the image.

6. CONCLUSION

The analysis of colonoscopy images using deep learning techniques has shown that automatic polyp detection and classification can be done effectively. Two models Convolutional Neural Network (CNN) and MobileNetV2 were trained and tested on image data. The results showed that MobileNetV2 performed better, achieving 98.2% accuracy in detecting polyps and 83.78% in identifying the type of polyp. In comparison, CNN achieved 89.19% and 81.98% accuracy in the same tasks.

To make the system more user-friendly, a Tkinter-based desktop interface was developed. This allows users to upload colonoscopy images and receive instant predictions about polyp presence and type. The interface is simple and easy to use, requiring no technical skills. Overall, the analysis shows that MobileNetV2, combined with a user-friendly Tkinter interface, provides an accurate, fast, and efficient solution for polyp detection and classification. To demonstrates the potential of deep learning in medical image analysis and supports the idea that automated systems can assist in improving the speed and reliability of visual diagnostics.

The Colonoscopy image analysis for polyps detection highlights how AI-driven diagnostic tools can significantly reduce dependency on manual interpretation, minimize human error, and accelerate the clinical decision-making process. The approach not only improves the reliability of polyp detection but also supports early diagnosis, which is vital in preventing the progression of colorectal cancer. This system can be further improved by adding segmentation features, and deploying it as a web or mobile-based tool for broader access. With continued development, it can contribute to early cancer detection and better patient care, especially in rural or under resourced healthcare centers. This AI-driven solution can become a reliable support tool in gastroenterology, enhancing early detection and improving outcomes for patients worldwide.

REFERENCES

[1] Tan, R., Lee, C., & Wang, S. (2023). “Enhanced Scattering Wavelet CNN for Colorectal Polyp Classification,” Journal of Medical Imaging and Health Informatics, 13(4), 215–223.

[2] Choudhuri, A., Banerjee, S., & Das, P. (2025). “PolypSeg Track: A Unified Foundation Model for Polyp Detection, Segmentation, and Tracking in Colonoscopy Videos,” IEEE Transactions on Medical Imaging, 44(1), 45–58.

[3] Lo, C.-M., Yeh, Y.-H., Tang, J.-H., Chang, C.-C., & Yeh, H.-J. (2022). “Comparative Study of Texture-Based Features and CNNs for Rapid Colorectal Polyp Classification,” Computers in Biology and Medicine, 149, 105902.

[4] Shin, H. C., Qadir, H. A., Aabakken, L., Bergsland, J., & Balasingham, I. (2019). “Automatic Colon Polyp Detection Using a Region-Based Deep CNN with False Positive Learning,” Medical Image Analysis, 58, 101535.

[5] Xu, J., Kuai, Y., Chen, Q., et al. (2024). “Spatio-Temporal Feature Transformation-Based Polyp Recognition,” Digestive Diseases and Sciences, 69, 911–921.

[6] (2020). “An Improved Deep Learning Approach on Colonic Polyp Detection,” BMC Medical Imaging. Uses ResNet/VGG models fine-tuned with SGD; evaluated by accuracy, sensitivity, and specificity.

[7] Liu, J., Zhang, W., Liu, Y., & Zhang, Q. (2024). “IECFNet: An Implicit Edge-Guided Cross-Layer Fusion Network for Colorectal Polyp Segmentation,” Scientific Reports, 14, 11678.

[8] Lalinia, F., & Sahafi, M. (2023). “Real-Time Polyp Detection in Colonoscopy Using YOLOv8 Object Detection Framework,” Signal, Image and Video Processing, 17(3), 481–491.

[9] Selvaraj, V., Kumar, R., & Thomas, S. (2025). “CRP-ViT: A Vision Transformer-Based Hybrid Model for Real-Time Colorectal Polyp Classification,” Journal of Intelligent Systems in Medicine, 6(2), 120–129

APPENDIX

```
#IMPORTING LIBRARIES
import os
from PIL import Image, ImageTk
from tkinter import filedialog, Label, Button
import tkinter as tk
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import tensorflow as tf
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D
import joblib
Configuration
IMG_SIZE = (128, 128)
BATCH_SIZE = 32
EPOCHS = 10
images_folder = r"D:\mobilenet\clean-images"
labels_path = r"D:\mobilenet\cleaned_labels_with_extensions.xlsx"
Load Dataset
df = pd.read_excel(labels_path)
```

```
df['FILENAME'] = df['FILENAME'].str.strip().apply(lambda x: str(x) if str(x).endswith('.jpg') else str(x) + '.jpg')

df['POLYP'] = df['POLYP'].astype(int)

label_encoder = LabelEncoder()

df['polyp_type_encoded'] = label_encoder.fit_transform(df['TYPE'])

#Split Dataset

train_df, testval_df=train_test_split(df, test_size=0.3, random_state=42, stratify=df['POLYP'])

val_df,test_df=train_test_split(testval_df,test_size=0.5,random_state=42,
stratify=testval_df['POLYP'])

print(f"Dataset Split Summary:\nTrain: {len(train_df)}\nValidation: {len(val_df)}\nTest: {len(test_df)}")

Image Loader

def load_images_and_labels(df):

images, polyp_presence, polyp_type = [], [], []

for _, row in df.iterrows():

img_path = os.path.join(images_folder, row['FILENAME']

if not os.path.exists(img_path):

    continue

 img = load_img(img_path, target_size=IMG_SIZE)

img = img_to_array(img) / 255.0

images.append(img)

polyp_presence.append(row['POLYP'])

polyp_type.append(row['TYPE'])

return np.array(images), np.array(polyp_presence), np.array(polyp_type)

X_train, y_train_presence, y_train_type = load_images_and_labels(train_df)
```

```
X_val, y_val_presence, y_val_type = load_images_and_labels(val_df)

X_test, y_test_presence, y_test_type = load_images_and_labels(test_df)

#Encode Labels

y_train_type_enc = label_encoder.transform(y_train_type)

y_val_type_enc = label_encoder.transform(y_val_type)

y_test_type_enc = label_encoder.transform(y_test_type)

num_classes = len(label_encoder.classes_)

y_train_type_cat = to_categorical(y_train_type_enc, num_classes)

y_val_type_cat = to_categorical(y_val_type_enc, num_classes)

y_test_type_cat = to_categorical(y_test_type_enc, num_classes)

#Model Architecture

base_model = MobileNetV2(input_shape=(IMG_SIZE[0], IMG_SIZE[1], 3), include_top=False, weights='imagenet')

base_model.trainable = False

x = base_model.output

x = GlobalAveragePooling2D()(x)

x = Dense(128, activation='relu')(x)

x = Dropout(0.5)(x)

presence_output = Dense(1, activation='sigmoid', name='presence')(x)

type_output = Dense(num_classes, activation='softmax', name='type')(x)

model = Model(inputs=base_model.input, outputs=[presence_output, type_output])

model.compile(optimizer='adam',loss={'presence':'binary_crossentropy','type':'categorical_cr
ossentropy'},

 metrics={'presence': 'accuracy', 'type': 'accuracy'})

 #Train the Model
```

```
model.fit(X_train, {'presence': y_train_presence, 'type': y_train_type_cat},
validation_data=(X_val, {'presence': y_val_presence, 'type': y_val_type_cat}),
epochs=EPOCHS, batch_size=BATCH_SIZE)
# Evaluate
eval_results = model.evaluate(X_val, {'presence': y_val_presence, 'type': y_val_type_cat},
verbose=1)
print("Polyp Presence Accuracy :", round(eval_results[3] * 100, 2), "%")
print("Polyp Type Accuracy  :", round(eval_results[4] * 100, 2), "%")
#Save Model and Encoder
model.save("polyp_model_mobilenet.keras")
joblib.dump(label_encoder, "label_encoder.pkl")
#Severity Logic
def get_severity(polyp_type, size):
 if size == 'large' or polyp_type.lower() == 'adenoma':
      return "High"
  elif size == 'medium':
     return "Medium"
else:
       return "Low"
 #GUI Logic
def analyze_image():
 file_path = filedialog.askopenfilename(filetypes=[("Image files", "*.jpg *.png *.jpeg")])
if not file_path:
      return
 img = Image.open(file_path)
```

```
 img_resized = img.resize(IMG_SIZE)
img_array = img_to_array(img_resized) / 255.0
img_array = np.expand_dims(img_array, axis=0)
predictions = model.predict(img_array)
presence_conf = predictions[0][0][0]
type_index = np.argmax(predictions[1][0])
 polyp_type = label_encoder.inverse_transform([type_index])[0]
presence = int(presence_conf > 0.5)
 if presence_conf > 0.85:
       size = "large"
elif presence_conf > 0.6:
      size = "medium"
 else:
       size = "small"
 severity = get_severity(polyp_type, size)
img_display = img.resize((250, 250))
img_tk = ImageTk.PhotoImage(img_display)
 image_label.config(image=img_tk)
 image_label.image = img_tk
if presence:
    result_text.set(
f"PolypDetected!\nType:{polyp_type}\nSize:{size}\nSeverity:{severity}\nConfidence:
{presence_conf*100:.2f}%"
)
 else:
```

```
        result_text.set("No Polyp Detected.")


#GUI Window
root = tk.Tk()
root.title("Polyp Detection with MobileNet")
root.geometry("600x500")
Label(root, text="Polyp Detection and Analysis", font=("Arial", 16, "bold")).pack(pady=10)
Button(root, text="Upload Image and Analyze", font=("Arial", 12), bg="blue", fg="white",
command=analyze_image).pack(pady=10)
image_label = Label(root)
image_label.pack(pady=10)
result_text = tk.StringVar()
Label(root, textvariable=result_text, font=("Arial", 12), justify="left").pack(pady=10)
root.mainloop()
```

CONVOLUTIONAL NEURAL NETWORK

```
#Import Required Libraries
Importing Libraries
import pandas as pd
import numpy as np
import os
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import tensorflow as tf
from tensorflow.keras.preprocessing.image import load_img, img_to_array
```

```
from tensorflow.keras.models import Model

from tensorflow.keras.utils import to_categorical

from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Flatten, Dense, Dropout

#Parameters

IMG_SIZE = (128, 128)  # Image size for resizing

BATCH_SIZE = 32

EPOCHS = 10

#Folder path and Excel path

images_folder = r"D:\cnn\clean-images"

labels_path = r"D:\cnn\cleaned_labels_with_extensions.xlsx"

#Load and Explore Dataset

df = pd.read_excel(labels_path)  # Load Excel sheet containing labels

print(df.head())  # Display first few rows

print("Checking for null values in each column:")

print(df.isnull().sum())

#Encode 'TYPE' labels to numbers

le = LabelEncoder()

df['polyp_type_encoded'] = le.fit_transform(df['TYPE'])

print("Encoded Classes:", le.classes_)

 #Clean Filenames and Split Dataset

df['FILENAME'] = df['FILENAME'].str.strip()

df['FILENAME'] = df['FILENAME'].apply(lambda x: str(x) if str(x).endswith('.jpg') else str(x)
+ '.jpg')

train_df,testval_df= train_test_split(df, test_size=0.3, random_state=42, stratify=df['POLYP'])
```

```
val_df,test_df=train_test_split(testval_df,test_size=0.5,random_state=42,stratify=testval_df['
POLYP'])
print(f"Train size: {len(train_df)}, Val size: {len(val_df)}, Test size: {len(test_df)}")


#Load Images and Labels into Arrays
def load_images_and_labels(df):
    images = []
    polyp_presence = []
    polyp_type = []
    missing_files = []
    for _, row in df.iterrows():
    img_path = os.path.join(images_folder, row['FILENAME'])
    if not os.path.exists(img_path):
    print(f"File not found: {img_path}")missing_files.append(row['FILENAME'])
        continu
 img = load_img(img_path, target_size=IMG_SIZE)
 img = img_to_array(img) / 255.0  # Normalize image pixels
images.append(img)
polyp_presence.append(row['POLYP'])  # 0 or 1
polyp_type.append(row['TYPE'])  # class label
print(f"Loaded {len(images)}/{len(df)} images.")
 if missing_files:
 print(f"Missing files count: {len(missing_files)}")
   return np.array(images), np.array(polyp_presence), np.array(polyp_type)
#Load data
```

```
X_train, y_train_presence, y_train_type = load_images_and_labels(train_df)

X_val, y_val_presence, y_val_type = load_images_and_labels(val_df)

X_test, y_test_presence, y_test_type = load_images_and_labels(test_df)

#Encode Type Labels into Categorical

le = LabelEncoder()

le.fit(df['TYPE'])

y_train_type_enc = le.transform(y_train_type)

y_val_type_enc = le.transform(y_val_type)

y_test_type_enc = le.transform(y_test_type)num_classes = len(le.classes_)

# Convert to one-hot encoded format

y_train_type_cat = to_categorical(y_train_type_enc, num_classes)

y_val_type_cat = to_categorical(y_val_type_enc, num_classes)

y_test_type_cat = to_categorical(y_test_type_enc, num_classes)

#Build CNN Model with Dual Output

input_layer = Input(shape=(IMG_SIZE[0], IMG_SIZE[1], 3))

x = Conv2D(32, (3,3), activation='relu')(input_layer)x = MaxPooling2D(2,2)(x)

x = Conv2D(64, (3,3), activation='relu')(x)

x = MaxPooling2D(2,2)(x)

x = Conv2D(128, (3,3), activation='relu')(x)

x = MaxPooling2D(2,2)(x)

x = Flatten()(x)

x = Dense(128, activation='relu')(x)

x = Dropout(0.5)(x)

presence_output = Dense(1, activation='sigmoid', name='presence')(x)
```

```
type_output = Dense(num_classes, activation='softmax', name='type')(x)


#Define the model
model = Model(inputs=input_layer, outputs=[presence_output, type_output])
#Compile model
model.compile(optimizer='adam',
              loss={'presence': 'binary_crossentropy', 'type': 'categorical_crossentropy'},
              metrics={'presence': 'accuracy', 'type': 'accuracy'})
model.summary()
#Train the Model
history = model.fit(
    X_train, {'presence': y_train_presence, 'type': y_train_type_cat},
    validation_data=(X_val, {'presence': y_val_presence, 'type': y_val_type_cat}),
    epochs=EPOCHS,
    batch_size=BATCH_SIZE
)
#Evaluate the Model
eval_results = model.evaluate(X_val, {'presence': y_val_presence, 'type': y_val_type_cat})
print("Polyp Presence Accuracy :", round(eval_results[3] * 100, 2), "%")
print("Polyp Type Accuracy     :", round(eval_results[4] * 100, 2), "%")
Predict on a Single Image (Testing)
def predict_image(img_path):
 img = load_img(img_path, target_size=IMG_SIZE)
 img_arr = img_to_array(img) / 255.0
```

```
 img_arr = np.expand_dims(img_arr, axis=0)
predictions = model.predict(img_arr)
pred_presence, pred_type = predictions
presence_label = "Polyp" if pred_presence[0][0] > 0.5 else "No Polyp"
predicted_type_index = np.argmax(pred_type[0])
type_label = le.inverse_transform([predicted_type_index])[0]
 print(f"Polyp Presence: {presence_label}")
  print(f"Polyp Type: {type_label if presence_label == 'Polyp' else 'Normal'}")
 #Test example image
predict_image(r"D:\cnn\clean-images\5_png.rf.84fee065fa5e026ea84ed04a26e6e5af.jpg")
import tkinter as tk
from tkinter import filedialog, Label, Button
from PIL import Image, ImageTk
import numpy as np
from keras.models import load_model
from keras.preprocessing.image import img_to_array
import joblib
#Load model and label encoder
model = load_model("polyp_model.h5")
label_encoder = joblib.load("label_encoder.pkl")
IMG_SIZE = (128, 128)
#Define severity logic
def get_severity(polyp_type, size):
    if size == 'large' or polyp_type.lower() == 'adenoma':
```

```
        return "High"
    elif size == 'medium':
        return "Medium"
    else:
        return "Low"
 #Function to handle image analysis
def analyze_image():
    file_path = filedialog.askopenfilename(filetypes=[("Image files", "*.jpg *.png *.jpeg")])
    if not file_path:
        return
#Preprocess image
img = Image.open(file_path)
img_resized = img.resize(IMG_SIZE)
img_array = img_to_array(img_resized) / 255.0
img_array = np.expand_dims(img_array, axis=0)
#Predict using the model
predictions = model.predict(img_array)
if isinstance(predictions, list): # Multi-output model
    presence_conf = predictions[0][0][0]
    type_index = np.argmax(predictions[1][0])
     polyp_type = label_encoder.inverse_transform([type_index])[0]
    else:
            presence_conf = predictions[0][0]
             polyp_type = "Unknown"
```

```
            presence = int(presence_conf > 0.5)
             if presence_conf > 0.85:
size = "large"
 elif presence_conf > 0.6:
        size = "medium"
    else:
        size = "small"
severity = get_severity(polyp_type, size)
  img_display = img.resize((250, 250))
 img_tk = ImageTk.PhotoImage(img_display)
image_label.config(image=img_tk)
image_label.image = img_tk
#Display results
  if presence:
       result_text.set(
        f" Polyp Detected!\n"
         f"Type: {polyp_type}\n"
         f"Size: {size}\n"
          f"Severity Level: {severity}\n"
          f"Confidence: {presence_conf * 100:.2f}%"
        )
    else:
 result_text.set(" No Polyp Detected.")
root = tk.Tk()
```

```
root.title("Polyp Detection and Analysis")
root.geometry("600x500")
title_label = Label(root, text="Polyp Detection and Analysis", font=("Arial", 16, "bold"))
title_label.pack(pady=10)
upload_btn = Button(root, text="Upload Image and Analyze", font=("Arial", 12, "bold"),
                    bg="blue", fg="white", padx=10, pady=5, command=analyze_image)
upload_btn.pack(pady=10)
image_label = Label(root)
image_label.pack(pady=10)
result_text = tk.StringVar()
result_label = Label (root, textvariable=result_text, font= ("Arial", 12), justify="left")
result_label.pack(pady=10)
root.mainloop()
```