Received 16 November 2024, accepted 1 December 2024, date of publication 9 December 2024, date of current version 17 December 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3512664

RESEARCH ARTICLE

Medical Image Segmentation for Anomaly Detection Using Deep Learning Techniques

BIBAT THOKAR[1], BINOD SAPKOTA[1], BABU R. DAWADI[2], AND SHASHIDHAR R. JOSHI[2]
[1]Department of Electronics and Computer Engineering, Thapathali Campus, Tribhuvan University, Kathmandu 46000, Nepal
[2]Department of Electronics and Computer Engineering, Pulchowk Campus, Tribhuvan University, Lalitpur 44700, Nepal

Corresponding author: Binod Sapkota (replybinod@gmail.com)

This work was supported by University Grants Commission (UGC), Nepal, under Grant ID: CRG-078/79-Engg-01.

ABSTRACT Medical images are the standard approach for the analysis and diagnosis of critical issues of diseases. To minimize the time-consuming inspection and evaluation process of the medical images from physicians in diagnosis, an automatic segmentation mechanism of abnormal features in medical images is required. To address the limited availability of medical image data, deep learning frameworks for multi-class image segmentation have been implemented. Moreover, the existing deep learning frameworks are static. Hence to make dynamic for the image segmentation purpose, the existing deep learning-based frameworks for medical image segmentation have been updated by integrating advanced architectures to observe performance enhancements. The existing UNet model has been integrated with a vision transformer to capture the structural properties of the medical image. Similarly, the ResNet50 architecture has been integrated with DeepLabv3plus for better extraction of features. The CVC-ClinicDB dataset, ISIC dataset, Brain Tumor dataset and HyperKVasir dataset have been collected from multiple sources. The collected datasets have been processed with image augmentation, Contrast Limited Adaptive Histogram equalization (CLAHE), and normalization. The preprocessed image dataset has been categorized into training, validation, and testing parts and used accordingly. The training image data has been used to train the multiple deep-learning models. The adaptive moment (Adam) optimizer has been used for the optimization process. The loss measurement has been carried out using categorical cross-entropy. The trained models have been evaluated using a testing dataset. The performance of the implemented deep learning framework has been measured by Accuracy, Precision, Recall, F1-Score, Dice Coefficient, and Validation Dice Coefficient metrics.

INDEX TERMS DeepLabv3plus, medical image, segmentation, UNet, vision transformer.

I. INTRODUCTION

Medical image semantic segmentation (MISS) is a crucial task in medical imaging, presenting unique challenges compared to natural image segmentation. MISS involves classifying image pixels based on specific anatomical labels across different imaging techniques. Benchmark datasets for MISS often exhibit organ distortions due to diverse image acquisition methods, and the limited availability of detailed pixel level annotations adds another layer of complexity. For effective segmentation, models need to capture both local semantic details, reflecting intricate organ structures, and global dependencies, which show the relationships between various organs. Convolutional networks have revolutionized computer vision due to their strong feature representation capabilities, with encoder-decoder architectures now advancing position-sensitive tasks like semantic segmentation. These models enhance receptive fields through down-sampling and accumulate local responses from multiple convolution layers to achieve global context. However, two inherent limitations persist: convolutions cannot capture long-range dependencies directly, only extracting information from nearby pixels, and fixed convolution kernel sizes restrict adaptability to varied inputs. Automated medical image segmentation, a key step in computer-aided diagnosis, aims to identify anatomical structures and regions of interest

The associate editor coordinating the review of this manuscript and approving it for publication was Zhan-Li Sun.

(ROIs). To produce high-quality segmentation masks, models need to balance contextual awareness and high spatial resolution since CNNs face challenges in such areas, as they often reduce feature resolution to capture more context.

Medical images differ significantly from natural images, and they are not as readily available. Although deep learning frameworks have shown promising results in anomaly detection [2], implementing these methods remains challenging due to limited access to medical image data. Furthermore, once deep learning models are trained for a specific purpose, they are generally static and cannot be easily updated. To overcome this dataset limitation, an approach has been developed to train and test models using the available data. For more effective analysis and adaptive model performance, a new deep learning framework has been proposed. These frameworks are crucial in extracting important features from images, which can enhance model accuracy and efficiency. By serving as the foundation, this proposed framework facilitates the precise detection and classification of unusual regions within the medical image.

A. CONTRIBUTIONS

The deep learning-based approach is an efficient framework for the detection of an anomalous part in the image because before detecting the anomalous part, significant features are extracted and highlighted which improves the model performance for detection. To address the limitations of previous deep learning frameworks [3], enhance their performance, and limit the availability of medical image data, the existing deep learning architectures have been updated as follows:

- The existing deep learning architectures have been integrated with advanced frameworks. The split strategy of the deep learning frameworks makes the succeeding framework detect multiple instances of different objects present in an image.
- Rather than training the model to recognize specific objects in the training/testing dataset, the model is trained to know the similarities and differences between objects in an image.
- The implemented strategy reduces data collection effort and computational effort.
- The framework helps to detect and classify abnormal parts in the medical images obtained from different bio-medical equipment more efficiently and accurately.
- Medical images having abnormal parts have been segmented and detected by implementing deep learning-based segmentation architectures with additional computational architecture.

B. RELATED WORKS

UTNet is a U-shaped hybrid Transformer Network that combines the strengths of self-attention and convolution techniques for medical image segmentation [4]. The main objective is to use self-attention to acquire long-range associative information while applying convolution layers to extract local intensity features to avoid large-scale transformer pretraining. The self-attention mechanism reduces the overall complexity from $O(n^2)$ to approximate $O(n)$ in both time and space. The multi-label, multi-vendor cardiac magnetic resonance imaging (MRI) challenge data has been used to thoroughly test the UTNet, encompassing left ventricle (LV), right ventricle (RV), and left ventricular myocardium (MYO) segmentation. The segmentation performance in terms of *dice scores* for UTNet is found to be 93.1, 83.5, and 88.2 for the LV dataset, MYO dataset, and RV dataset respectively, and on average 88.3.

A hierarchical U-shaped transformer called MISSFormer [5] has been used which is position-free and designed for medical image segmentation. Enhanced Mix-FFN has been redesigned which is a potent feed-forward network, with improved feature discrimination, long-range dependencies, and local context. To create a robust feature representation, it has been expanding to obtain an Enhanced Transformer Block (ETB). The ETB serves as the foundation for the Enhanced Transformer Context Bridge, which is designed to capture both the global and local correlations of hierarchical multi-scale characteristics. The efficacy, superiority, and robustness of the used MISSFormer are demonstrated by the excellent experimental results on medical image segmentation datasets. Automated cardiac diagnostic challenge dataset (ACDC) and Synapse multi-organ segmentation dataset (Synapse) have been used. 30 abdominal CT scans totaling 3779 axial abdominal clinical CT images make up the Synapse dataset. The dataset is randomly split into 12 scans for testing and 18 scans for training. The *dice score coefficient* for the ACDC dataset is found to be 90.86.

Two 2D medical images, called polyp segmentation in colonoscopy images and optic disc/cup segmentation in fundus images from the REFUGE 20 challenge have been used for segmentation tasks using a transformer-based Segtran model [6]. In addition, the model has been assessed on a 3D image segmentation task: the BraTS'19 challenge's brain tumor segmentation in MRI data. Compared to U-Net and its variations (UNet++, UNet3+, PraNet, and nnU-Net), as well as DeepLabV3+, Segtran has continuously demonstrated superior performance. The model has a *dice score* of 0.817 on average.

To segment colonoscopic images, an enhanced ResUNet architecture called ResUNet++ [7] has been employed. With a *dice coefficient* of 81.33%, a mean intersection over union *mIoU* of 79.27% for the Kvasir-SEG dataset, a *dice coefficient* of 79.55%, and a *mIoU* of 79.62% for the CVC-612 dataset, it has received good evaluation ratings. The ResUNet++ architecture leverages squeezing and excitation blocks, residual blocks, attention blocks, and Atrous Spatial Pyramidal Pooling (ASPP). When comparing ResUNet++ to other cutting-edge techniques, the segmentation findings for the colorectal polyps were noticeably better. The suggested architecture functions best when there are fewer photos. An experienced gastroenterologist has assisted in the annotation of the Kvasir dataset, resulting in the creation of the Kvasir-SEG dataset.

The diagnostic performance of a Transformer-based Residual network (TransNetR) [8] has been assessed and utilized for colon polyp segmentation. The suggested architecture is an encoder-decoder network with three decoder blocks, an upsampling layer at the network's end, and an encoder that is a pre-trained ResNet50. On the Kvasir-SEG dataset, TransNetR achieves a high *dice coefficient* of 0.8706, a mean intersection over the union *mIoU* of 0.8016, and a real-time processing speed of 54.60. Aside from this, the work's primary addition is investigating the TransNetR's generalizability by evaluating the suggested approach on an out-of-distribution dataset (one in which the test distribution differs from the training distribution).

II. METHOD

The proposed system diagram of the deep learning-based segmentation technique is as shown in figure 1. The block diagram shows the overall working mechanism from data preprocessing, model training, testing, and evaluation process.

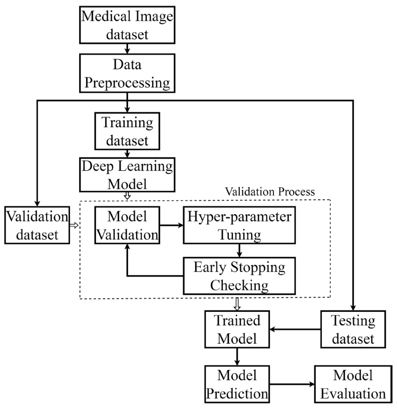

FIGURE 1. Proposed system block diagram.

A. SEMANTIC SEGMENTATION

The technique of classifying each pixel in an image into particular groups is known as semantic segmentation. By assigning specifics to different sections according to their shared semantic significance, the objective is to understand the scene's general context [9]. It assigns each pixel in an image with an appropriate class label without taking any further context or information into account. The aim is to densely label the image by giving a label to each pixel in the image as a whole. Using an image as input, the algorithm creates a segmentation map in which the image's pixels with values (0,1,...255) are converted to class labels $(0, 1, \ldots, n)$. It is helpful in situations where it's crucial to distinguish between various classes of roadside things. A segmentation mask is made up of all the pixels connected to the same class group. These models can precisely create boundaries for localization and classify objects at a granular level. An input image is fed via a sophisticated neural network architecture by a semantic model.

TABLE 1. Data summary table.

Source	Images	Masks
Brain Tumor	3064	3064
CVC ColonDB	4284	4284
HyperKVasir	1000	1000
ISIC	4048	4048

B. DATA COLLECTION

The medical image dataset has been collected from multiple sources as explained in the dataset source explanation section below. The International Skin Imaging Collaboration (ISIC) [10] has compiled a sizable dataset of dermoscopy images that is available to the public. The collection contains over 20,000 photos sourced from top clinical centers using a range of technologies employed by each institution. The aim was to supply a stable dataset snapshot to facilitate the creation of automated algorithms for melanoma diagnosis in three different lesion analysis tasks: segmentation, dermoscopic feature identification, and classification. The HyperKvasir dataset [11] contains 1,000 images from the polyp class together with the original image, a segmentation mask, and a bounding box. The region of interest, or polyp tissue, is represented by the foreground (white mask) in the mask, whereas the background (black) is devoid of polyp pixels. The boundaries of the detected polyp are the pixels that make up the bounding box. Each of the 1,000 JPEG-compressed photos in the dataset's images and matching masks is included. A database of frames taken from colonoscopy videos is called CVC-ClinicDB. The collection includes many polyp frame samples together with the ground truth that corresponds to them. A mask that matches the area of the image that the polyp is covering makes up the Ground Truth images. This brain tumor dataset includes 3064 T1-weighted, contrast-enhanced images from 233 patients who had meningioma (708 slices), glioma (1426 slices), and pituitary tumor (930 slices) as their three types of brain tumors. Owing to the repository's file size constraint, we divided the entire dataset into four subgroups, each of which produced four.zip files with 766 slices. The overall summary of data amount that has been used is shown in the table 1. The table shows the total amount of image data along with corresponding masks in the training, validation, and testing process.

C. DATA PREPROCESSING

At the very beginning, the collected sample medical image data was labeled with the help of a medical expert [12]. The labeled image dataset has been further pre-processed so that data becomes more efficient and effective for both training and testing processes. At the end of the data preprocessing phase, the entire dataset has been divided into training, testing, and validation sections. The training dataset has been used for training the framework.

1) IMAGE AUGMENTATION

When deep learning is used for medical image analysis, image augmentation is a data preparation method that boosts the variety and variability of the data. The original images from the training dataset are subjected to numerous image modifications, including translation, rotation, scaling, flipping, intensity transformation, deformation, and noise addition. By strengthening deep learning models' capacity to handle real-world medical images with a variety of properties, these strategies can aid in improving performance and generalization. To enhance the model performance, geometric augmentation of flipping horizontally and Gaussian noise addition has been done. The horizontal flipping of the image can be expressed by equation 1.

$$x' = width - x \tag{1}$$

where *width* and *height* are the relative width and height of the image. The model learns mirror-invariant characteristics and becomes more flexible at handling pictures of various orientations by flipping images. In the preparation of medical imaging data for the model, the Gaussian blur is applied as an image augmentation technique. It entails putting a Gaussian filter on the original image to blur it and eliminate high-frequency noise. Let the input image be $I(x, y)$ of size $W \times H$, then the mathematical formulations of the Gaussian blur operation is represented by equation2.

$$G(x, y) = \frac{1}{2\pi\sigma^2} \times \exp\left(\frac{-(x^2 + y^2)}{(2 \times \sigma^2)}\right) \tag{2}$$

where x and y are the pixel coordinates of the image. σ is the standard deviation of the Gaussian distribution to control the amount of blur applied to the image. Hence, a higher value of sigma generates a stronger blur effect. The Gaussian blur operation is implemented by convolving the input image $I(x, y)$ with the Gaussian kernel $G(x, y)$ using a convolution operation. The resulting blurred image $\hat{I}(x, y)$ is obtained as represented by equation 3,

$$\hat{I}(x, y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} [I(x + i, y + j) * G(i, j)] \tag{3}$$

where i and j are the indices of the Gaussian kernel $G(x, y)$ ranging from $-k$ to k and k is an integer that determines the size of the Gaussian kernel. The larger value of k generates a stronger blur effect. $I(x + i, y + j)$ is the pixel value of the input image at the coordinates $(x + i, y + j)$. The convolution operation has been implemented using a sliding window approach having the Gaussian kernel centered at each pixel of the input image, and the pixel values are multiplied with the corresponding values in the Gaussian kernel and then summed up to obtain the blurred pixel value as an output image.

2) INTENSITY NORMALIZATION

Normalization involves scaling the pixel intensities of the images to a standardized range. Normalization can help improve the model's performance by reducing the effects of differences in image brightness and contrast, which can be caused by variations in illumination or imaging conditions. Let I be an image of resolution $(H \times W \times C)$. Here H is the height, W is the width, and C is the channel number of the image. The pixel intensities of I are represented as a tensor of shape (H, W, C). To normalize the pixel intensities of I, the mean and standard deviation of the pixel intensities are first computed across all pixels and channels of the image. If μ and σ are the mean and standard deviation respectively, these values can be computed as represented by equation 4 and 5.

$$\mu = \left(\frac{1}{N}\right) \times \sum I \tag{4}$$

The above equation computes the mean across all pixels and channels

$$\sigma = \sqrt{\left(\left(\frac{1}{N}\right) \times \sum((I - \mu)^2)\right)} \tag{5}$$

Here σ is the standard deviation across all pixels and channels where $N = H \times W \times C$ is the total number of pixels in the image. The image can be normalized by subtracting the mean and dividing by the standard deviation following the computation of the mean and variance of the pixel intensities as represented by equation 6.

$$I_{Normalize} = \frac{(I - \mu)}{\sigma} \tag{6}$$

The resulting image $I_{Normalize}$ has pixel intensities centered around zero and unit variance. In some cases, the normalized pixel intensities are scaled to a specific range such as [0,1] or [−1,1]. This can be done by applying a linear transformation to the normalized pixel intensities as represented by equation 7.

$$I_{NS} = \frac{(I_{Normalize} - a)}{(b - a)} \tag{7}$$

where $a = min(I_{Normalize}())$ is minimum pixel intensity and $b = max(I_{Normalize}())$ is maximum pixel intensity. The resulting image I_{NS} has pixel intensities that are scaled to the range [0,1] based on the minimum and maximum pixel intensities in the normalized image.

3) CONTRAST LIMITED ADAPTIVE HISTOGRAM EQUALIZATION (CLAHE)

Contrast Limited Adaptive Histogram Equalization (CLAHE) [13] is used to enhance image contrast. It is a variation of Adaptive Histogram Equalization (AHE) that is more sophisticated. CLAHE was created to enhance the accuracy of complex structure imaging in medicine. Medical imaging local contrast and usefulness are enhanced by CLAHE. Low-contrast images have been effectively improved with CLAHE. The CLAHE method separates the images into contextual "tiles," which are then each given a histogram that is then used to approximate the output to a specified histogram distribution parameter [14]. Let $W \times H$ be the pixel size of an image and each tile size in the image be $w \times h$. The number of tiles is obtained from the equation 8.

$$T = \frac{W \times H}{w \times h} \tag{8}$$

The histograms of the obtained tiles are generated using the clip limit C_L of the image from the equation 9.

$$C_L = N_{CL} \times N_{avg} \tag{9}$$

where N_{CL} is Normalized Contrast Limit and N_{avg} is average count of pixels. N_{avg} is obtained using the equation 10.

$$N_{avg} = \frac{N_x \times N_y}{N_g} \tag{10}$$

where N_x, N_y, and N_g are the number of gray scales, x and y dimensions of the pixels, respectively. The relationship produces a mean of clip pixel values from equation 11.

$$N_{cp} = \frac{N \sum C_L}{N_g} \tag{11}$$

where $N \sum C_L$ is overall number of C_L. The remaining pixels are redistributed using the equation 12.

$$R = \frac{N_g}{N_r} \tag{12}$$

The bi-linear interpolation is implemented to minimize false borders in the image and to combine neighboring tiles. The CLAHE methodology focuses on enhancing local contrast to overcome the limitations of global approaches. Important hyper-parameters for this approach include the tile size and clip limit. The use of hyper-parameters improperly may result in an inappropriate impact on image quality. The best options for clip limit, clip size (10, 10), and other parameters are selected after a thorough analysis. A picture histogram created from the raw photos and the CLEHE shows the intensity levels of the image pixels. Using photographs with an intensity range of pixels 0 to 255, the study uses a statistical representation of the data. When compared to the original raw image, a CLAHE image's contrast is found to be noticeably better.

4) RESIZING

Resizing involves transforming the original image I with dimensions (H, W, C) into a new image Y with dimensions (H', W', C'), where H' and W' are the new height and width, and C' is the number of channels. The resizing process can be performed using various interpolation techniques, such as nearest-neighbor interpolation, bilinear interpolation, or cubic interpolation. Let $F(X, Y)$ be the interpolation function used to compute the pixel values of the new image Y based on the pixel values of the original image X. Then, the resizing process can be mathematically expressed with an equation 13.

$$\begin{aligned} Y(i,j,k) = F(X, i \times (H-1)/(H'-1), \\ j \times (W-1)/(W'-1), k) \end{aligned} \tag{13}$$

where $Y(i,j,k)$ is the pixel value of the new image Y at position (i,j) in channel k, and i and j are the indices of the pixel positions in the new image. The values $(H-1)/(H'-1)$ and $(W-1)/(W'-1)$ represent the scaling factors used to map the pixel positions in the new image to the corresponding pixel positions in the original image. The interpolation function F is applied to the pixel values of the original image X at the corresponding pixel positions $(i \times (H-1)/(H'-1), j \times (W-1)/(W'-1))$ to compute the pixel value of the new image Y at position (i,j,k). Resizing can help to ensure that the input images are of a consistent size and can be efficiently processed by the model.

D. DEEP LEARNING MODELS

Among the multiple deep learning frameworks [15], the existing deep learning architectures have been integrated with other deep learning-based architectures for a better segmentation process [16].

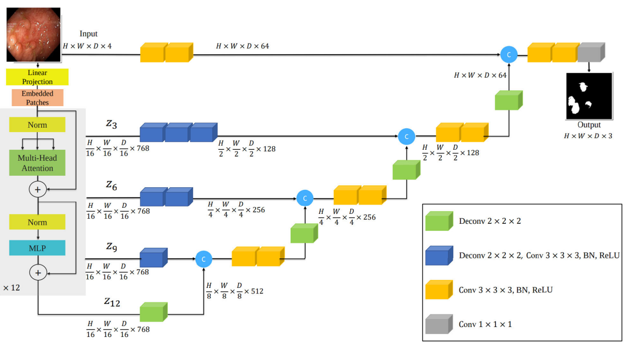
Input
$H \times W \times D \times 4$
$H \times W \times D \times 64$
Linear Projection
Embedded Patches
Norm
Multi-Head Attention
Norm
MLP
× 12
z_3
z_6
z_9
z_{12}
$\frac{H}{16} \times \frac{W}{16} \times \frac{D}{16} \times 768$
$\frac{H}{2} \times \frac{W}{2} \times \frac{D}{2} \times 128$
$\frac{H}{4} \times \frac{W}{4} \times \frac{D}{4} \times 256$
$\frac{H}{8} \times \frac{W}{8} \times \frac{D}{8} \times 512$
C
Output
$H \times W \times D \times 3$
Deconv 2 × 2 × 2
Deconv 2 × 2 × 2, Conv 3 × 3 × 3, BN, ReLU
Conv 3 × 3 × 3, BN, ReLU
Conv 1 × 1 × 1

FIGURE 2. Combination of UNet model with vision transformer.

1) EXTENSION TO EXISTING DEEP LEARNING MODELS

The existing UNet architecture has been integrated with Vision Transformer [17] to observe the better performance of the segmentation process as shown in the figure 2. The UNet model uses Vision Transformer as a backbone architecture. Initially, local features are extracted from ViT which helps better visualization of structural features extraction from the UNet model. The updated model uses a contracting-expanding pattern with an encoder that is a stack of transformers connected to a decoder by skip connections. The $1D$ input embedding sequences are used by the transformers to operate. The supplied image is split up into uniformly flattened, non-overlapping patches.

A linear layer that projects the patches into an embedding space that is K dimensional and stays that way across all transformer levels. The projected patch embedding has a 1D learnable positional embedding added to it to retain the spatial information of the extracted patches. Because the transformer backbone is made for semantic segmentation, the learnable [class] token is not added to the embedding sequence. Following the embedding layer, a stack of transformer blocks with sublayers for multilayer perceptrons (MLP) and multi-head self-attention (MSA) has been used. Two linear layers with GELU activation functions make up the MLP. There are n parallel self-attention (SA) heads in an MSA sublayer. The SA block, in particular, is a parameterized function that discovers the correspondence between a query (Q) and the associated value (V) and key (K) representations.

Sequence representations are taken out of the transformer and molded into a tensor, much like U-Net, which combines features from various encoder resolutions with the decoder. The reshaped tensors from the embedding space are projected into the input space at each resolution by the use of successive $3 \times 3 \times 3$ convolutional layers, which are then followed by layers of normalization. The modified feature map is given a deconvolutional layer to boost its resolution by a factor of two. The enlarged feature map is fed into successive $3 \times 3 \times 3$ convolutional layers, which upsample the output using a deconvolutional layer, after being concatenated with the feature map of the preceding transformer output. To create voxel-wise semantic predictions, the final output is fed into a $1 \times 1 \times 1$ convolutional layer with a softmax activation function. The process is repeated for all the other following layers up to the original input resolution.

A contracting path (left side) and an expanding path (right side) make up the U-Net architecture. The convolutional network is followed by the contracting path. Two 3×3 convolutions (unpadded convolutions) are applied repeatedly, and each is then followed by a rectified linear unit (ReLU) and a 2×2 max pooling operation with stride 2 for downsampling. The number of feature channels is increased by two at every round of downsampling [18]. The expanding path consists of upsampling the feature map, concatenating it with the correspondingly cropped feature map from the contracting path, reducing by half to the number of feature channels with a 2×2 convolution, and then performing two 3×3 convolutions, each followed by a ReLU.

Every convolution results in the loss of boundary pixels, which makes cropping inevitable. A 1×1 convolution is employed at the last layer to transfer every 64-component feature vector to the required number of classes. The framework contains 23 convolutional layers in total [19]. It is crucial to choose the input tile size so that all 2×2 max-pooling operations are applied to a layer with an even x-size and y-size to provide a continuous tiling of the output segmentation map. The cross-entropy penalizes at each position the deviation of $p_l(x)$ as equation 14.

$$E = \sum_{x} \in \omega x(x) \log(p_l(x)) \tag{14}$$

where $l : \omega \rightrightarrows (1, \ldots, K)$ is true class label and R is weight map. The separation border is computed using morphological operations where is weight map is calculated as,

$$w(x) = w_c(x) + w_0 \times \exp \frac{-((d_1(x) + d_2(x))^2}{2\sigma^2} \tag{15}$$

where w_c is the weight map to balance the class frequencies, d_1 is the distance to the border of the nearest cell, and d_2 distance to the border of the second nearest cell.

The existing deelabv3+ model has been integrated with the ResNet50 model as shown in the figure 3. The architecture is a combination of ResNet50 and deeplabv3plus [20]. The ResNet50 works as a backbone architecture that helps to extract the primary features of the input image which can be further processed for segmentation purposes.

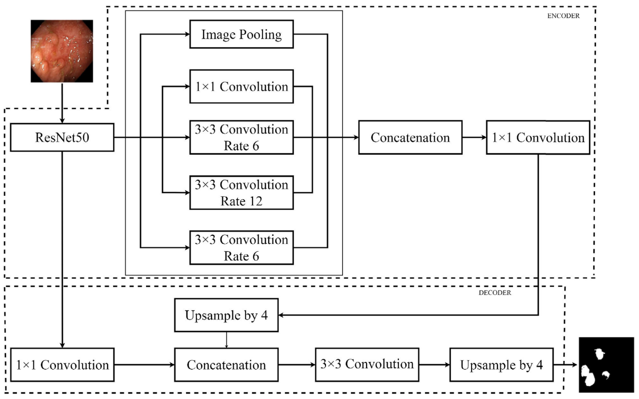

FIGURE 3. Combination of DeepLabv3+ model with ResNet50.

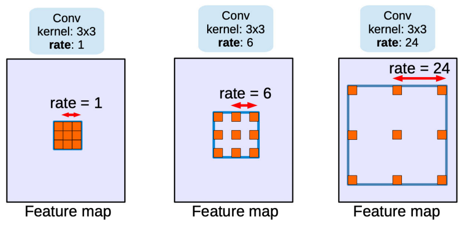

FIGURE 4. Atrous mechanism.

2) ATROUS CONVOLUTIONS AND ASPP IN DEEPLABV3+

In conventional convolutions, a filter is applied with a set stride across the input picture or feature map. Dot products between the filter and the corresponding input pixel are used to calculate each pixel in the final feature map. Dilated convolutions, on the other hand, create "holes" or gaps in the filter that let it collect additional spatial context without adding more parameters or lengthening the computation time. Such dilated (or atrous) convolutions apply the filter with a fixed stride, but with gaps between the filter elements, to the input picture or feature map. The amount of spatial context that the filter can capture depends on the gap size or dilation rate [21]. A dilation rate of 2, for instance, indicates that there is a single space between every filter element. The filter can better preserve the spatial resolution of the input image or feature map and capture more spatial context by raising the dilation rate as shown in figure 4. The Atrous Convolution [22] modifies the convolution's effective field-of-view. It modifies field-of-view using a parameter called atrous/dilation rate. Expanding the area of view of filters without affecting calculation or parameter count is a straightforward yet effective method. Atrous convolutions can aid the network in better capturing the global context and identifying objects of various sizes, which can result in more accurate segmentation results. This can be particularly helpful in semantic segmentation tasks.

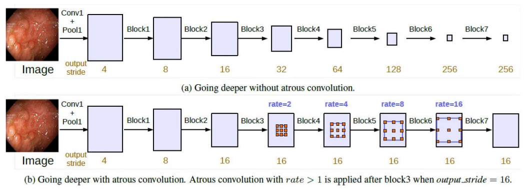

FIGURE 5. ASPP mechanism.

3) VISION TRANSFORMER

The Transformer architecture, which was initially created for natural language processing (NLP), is extended to the computer vision domain by a vision transformer (ViT). A deep learning model called a "Vision Transformer" has an attention-based transformer architecture that is appropriate for pattern identification in images [23]. The vision transformer has an encoder-only architecture as opposed to the original transformer's encoder and decoder. The schematic diagram of the Vision transformer is shown in figure 6. The Vision Transformer [23] architecture is the encoder version of the existing Transformer based deep learning architecture for addressing the computer vision, and image processing related application.

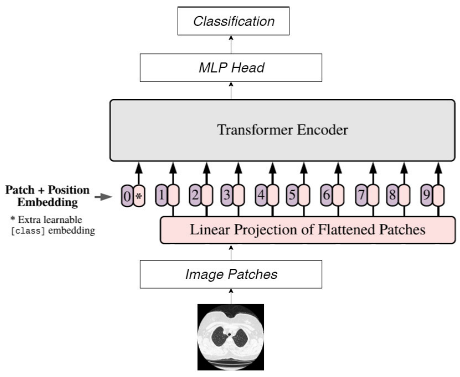
Classification
MLP Head
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
0*
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches
Image Patches

FIGURE 6. Vision transformer architecture.

- **Patch Generation**: The two-dimensional ($2D$) image I_{Input} from the training dataset having width W, height H, and channel C such that $I_{Input} \in \mathbb{R}^{(H \times W \times C)}$ is broken down into a sequence of N2-D patches. The split patch P such that $P \in \mathbb{R}^{(N \times S \times S \times C)}$ is passed into the transformer encoder. Here, (S, S) is the patch resolution, and N is the number of patches. The number of patches is calculated by using the equation 16.

$$N = \frac{(H \times W)}{S^2} \tag{16}$$

- **Positional Encoding**: The patches are embedded into a D-dimensional embedding space via a linear layer. To maintain the positional information, positional embeddings are appended to the patch embeddings. The transformer encoder layer maps input patches are mapped to the embedding space and are augmented with positional information using equation 17.

$$X = [P_{class}; P_1.E; P_2.E; \ldots P_N E] + E_{Position} \tag{17}$$

where $E \in \mathbb{R}^{(S^2.C) \times D}$ and $E_{Position} \in \mathbb{R}^{(N+1) \times D}$. P_{class} is a class label and $P_1, P_2, \ldots, P_N$ are the image patches.
- **Self Attention Mechanism**: The input sequence is represented as a sequence of vectors, where each vector represents a token in the sequence. Scaled Dot-Product Attention computes the dot product between the query vector and each of the key vectors in the input sequence. The resulting scores are then scaled by the square root of the dimensionality of the key vectors and passed through a softmax function to obtain a set of weights, which are used to compute a weighted average of the value vectors in the input sequence. Let Q, K, and V denote the query, key, and value matrices, respectively, and d_k denote the dimensionality of the key vectors. The scaled dot-product attention is computed using the equation 18.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}}) \tag{18}$$

The mentioned mathematical relation computes the attention weights, and V is the matrix of value vectors. The result of this computation is a weighted sum of the value vectors, where the weights are determined by the similarity between the query vector and each key vector. This mechanism allows the model to focus on the most relevant parts of the input sequence when making predictions.
- **Multi Head Attention Mechanism**: The Multi-Head Attention mechanism uses multiple heads, each processing the input sequence independently using the standard self-attention mechanism, and then concatenates the output vectors from each head to form the final output sequence. For a sequence of N patches, denoted as $X = x_1, x_2, \ldots, x_N$, the Multi-Head Attention mechanism aims to compute a new sequence of vectors $Y = y_1, y_2, \ldots, y_N$ that captures multiple relationships between the different patches in the image. To achieve this, the Multi-Head Attention mechanism first splits the input sequence X into h different sub-sequences or "heads", denoted as $X_1, X_2, \ldots, X_h$. Each head is then processed independently using the standard self-attention mechanism described in the equation. For each head i, attention scores $Attention(i, j)$ are computed

between each patch h_i and all other patches h_j in the same head using a scoring function $Attention(x_i, x_j)$ that is specific to that head. The attention scores are then used to compute a weighted average of the other patches in the same head, where the weights are determined by the attention scores. For each patch i in head k, the output vector $Y_{i,k}$ is computed using equation 19.

$$y_{i,k} = \sum_{j=1}^{N} softmax(Attention(i,j)^k)h_j \tag{19}$$

where the sum is taken over all patches j in the same head k, and the softmax function ensures that the weights sum to 1. The output vectors from each head are then concatenated to form the final output sequence Y. For each patch i, the final output vector y_i is computed using the equation 20.

$$y_i = concat(y_{i,1}, y_{i,2}, \ldots, y_{i,h}) \tag{20}$$

where *concat* denotes the concatenation operation. The resulting sequence of vectors Z can then be fed into a feed-forward neural network to obtain the final output of the model which is typically a set of class probabilities. The soft-max function provides the probability distribution for the classification of the corresponding class as described by equation 21.

$$\sigma(\vec{x_i}) = \frac{exp(x_i)}{\sum_{j=1}^{k} \exp(x_i)} \tag{21}$$

where x_i is the input vector to the soft-max function k is the number of classes in the multi-class classifier.

E. IMPLEMENTATION DETAILS

The overall experiments for UNet with Vision Transformer model and DeepLabv3plus with ResNet50 have been conducted by Kaggle and Google Colab. We used the TensorFlow environment for deep learning computation. The data preprocessing tasks have been carried out in $i5$, 8^{th} generation device using Jupyter Notebook. The 8-core NVIDIA P100 has been accessed from Kaggle and Google Colab. Initially, the learning rate for UNet with Vision Transformer has been set to be 10^{-4} which is gradually updated depending upon need of updating for a better learning process during the training process. The batch size to be set to be 10 and the number of layers is set to be 12. Similarly, the number of Layers is set to 12, the hidden dimension set to be 128, MLP Dimension is set to be 32, number of Heads is set to be 6, and the dropout rate is set to 0.1 for DeepLabv3plus with the ResNet50 model. The initial learning rate is started from 0.1 which is updated automatically based on the need for optimization. The input image size to taken to be 256 $\times$ 256 $\times$ 3 and the patch size is taken to be 16 $\times$ 16 for the Vision Transformer backbone model [24].

III. EXPERIMENTS AND RESULTS

A. TRAINING RESULTS OF DeepLabv3plus+ResNet50 MODEL USING BRAIN TUMOR DATA

1) DICE COEFFICIENT AND VALIDATION DICE COEFFICIENT RESULT

The figure 7 shows the *dice coefficient* of a model at each epoch during the training process. As shown in the figure, the *dice coefficient* and the *validation dice coefficient* both show an upward trend, indicating improved model performance. The *dice coefficient*, which measures the model's accuracy on the training set, increases significantly from 0.4001 to 0.9676, showing a continuous improvement in the model's ability to correctly identify segments. The *validation dice coefficient* measures the model's accuracy on unseen validation data, which has been improved overall but with more fluctuations compared to the training *dice coefficient*. It starts very low at 0.0198, rapidly increases to 0.8229 by the 5^{th} epoch, and then continues to improve more gradually, peaking at 0.8507 around the 30^{th} epoch and stabilizing around 0.8505 towards the end.

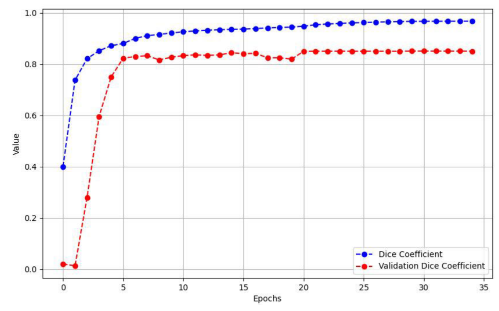

FIGURE 7. Dice coefficient and validation dice coefficient figure of DeepLabv3plus+ResNet50 model for brain tumor data.

2) TRAINING ACCURACY AND TRAINING LOSS RESULT

The figure 8 represents the relationship between epoch and loss of a model during the training phase. During 35 epochs, the *training accuracy* of the model steadily increases from 0.9426 to 0.9984, indicating that the model is progressively learning. Simultaneously, the *training loss* decreases from 0.5992 to 0.0324, reflecting a reduction in the error between the predicted outputs and the actual targets. The rapid improvement in the early epochs slows down as the model approaches its optimal performance, demonstrating that most learning happens during the initial training phase, and the rate of improvement diminishes as the model fine-tunes its parameters to achieve higher accuracy and lower loss.

3) VALIDATION ACCURACY AND VALIDATION LOSS RESULT

The figure 9 represents the relationship between the *validation accuracy* and *validation loss* with the epoch. During 35 epochs, the *validation accuracy* starts high at 0.9824 and fluctuates slightly but shows an overall increasing trend, reaching around 0.9946 by the end of the training by improving or maintaining its performance. Meanwhile,

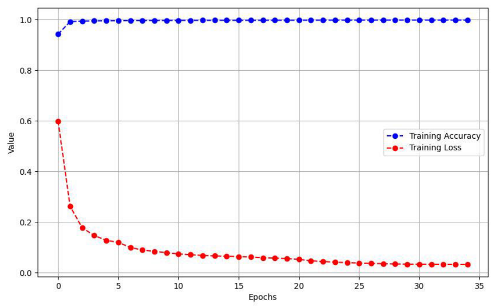

FIGURE 8. Training accuracy and training loss of DeepLabv3plus+ResNet50 for brain tumor data.

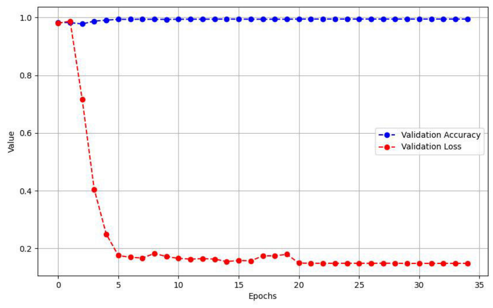

FIGURE 9. Validation accuracy and validation loss of DeepLabv3+ResNet50 for brain tumor data.

TABLE 2. Performance metrics of DeepLabv3plus with ResNet50 model on brain tumor image dataset.

Metrics	Results
Accuracy	0.9949
Precision	0.8904
Recall	0.8153
F1-Score	0.8512

the *validation loss* decreases significantly from 0.9802 to approximately 0.1483, especially in the early epochs, suggesting that the model is becoming better at minimizing errors on the validation set. However, the *validation loss* exhibits some fluctuations, especially after epoch 16, indicating some variability in model performance.

B. TESTING RESULTS OF DeepLabv3plus+ResNet50 MODEL FOR BRAIN TUMOR DATA

1) CONFUSION MATRIX

On analyzing these results for the brain tumor dataset, the *accuracy*, *precision*, *recall*, and $F1 - score$ are found to be as shown in the table 2 respectively. The table shows the evaluation metrics during testing using the Brain Tumor image dataset.

2) DeepLabv3plus WITH ResNet50 ANALYSIS ON BRAIN TUMOR DATA

The *learning rate* has been updated automatically as 10^{-4} up to epoch 19, 10^{-5} up to epoch 28 and 10^{-6} up to epoch 35. The image 10 shows the segmentation performance of the DeepLabv3plus+ResNet50 model for the Brain Tumor image. The leftmost image is the input image, the middle image is the ground truth mask of the input image and the rightmost image is the predicted segmentation mask from the trained model while testing the process on the brain tumor image dataset.

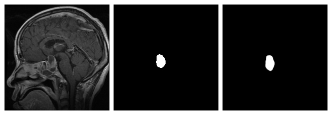

FIGURE 10. Segmentation result of DeepLabv3plus+ResNet50 on brain image.

3) ROC AND PR CURVES

ROC curves are used to evaluate binary classification models. The *ROC* curve for each class represents the relationship between the true positive rate (*TPR*) and the false positive rate (*FPR*). The *TPR* is the percentage of correctly classified positive instances and the *FPR* is the percentage of incorrectly classified negative instances. The *PR* and *ROC* Curves of DeepLabv3plus+ResNet50 Model for Brain Tumor testing data have been generated as shown in figure 11 with *AUC* of 0.91 and 12 with *AUC* of 0.85 respectively.

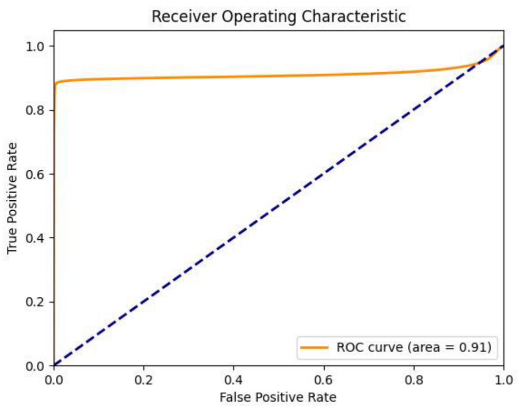

FIGURE 11. ROC curve for DeepLabv3plus+ResNet50 on brain tumor data.

C. TRAINING RESULTS OF DEEPLABV3PLUS+RESNET50 MODEL ON CVC-ClinicDB DATA

1) DICE COEFFICIENT AND VALIDATION DICE COEFFICIENT RESULT

The figure 13 shows the *dice coefficient* of a model at each epoch during the training process. As shown in the figure, the *dice coefficient* consistently increases from 0.575 to 0.881, indicating that the model's performance on the training data improves significantly over time. The *validation dice coefficient* also shows an overall increasing trend but with more variability compared to the training *dice coefficient*. It starts at 0.023 and rises

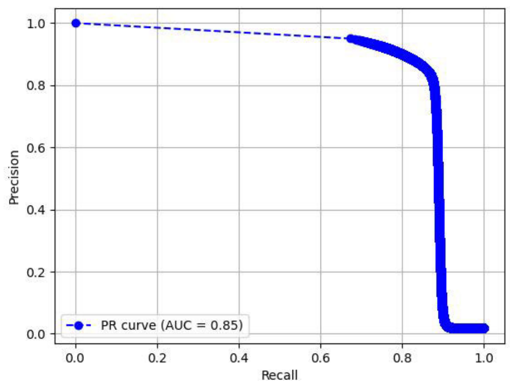

FIGURE 12. PR curve for DeepLabv3plus+ResNet50 on brain tumor data.

to around 0.729, with notable fluctuations, particularly in the early and mid-epochs. The initial rise in the *validation dice coefficient* is steep, suggesting rapid improvement in the model's ability to generalize to unseen data, but subsequent fluctuations indicate periods of over-fitting and adjustments.

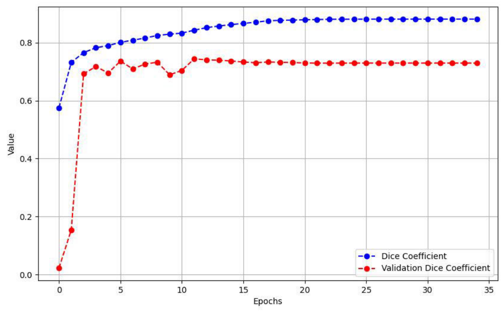

FIGURE 13. Dice coefficient and validation dice coefficient of DeepLabv3plus+ResNet50 for CVC-ColonDB data.

2) TRAINING ACCURACY AND TRAINING LOSS RESULT

The figure 14 represents the relationship between epoch and loss of a model during the training phase. Initially, the *training accuracy* increases rapidly, starting at 0.7451 and reaching 0.7960 by epoch 9.

Correspondingly, the *training loss* decreases significantly from 0.4247 to 0.1711 during the same period, indicating that the model is effectively learning from the data. As the epochs progress beyond this point, both the *training accuracy* and the *training loss* continue to improve, albeit at a slower rate. The *training accuracy* eventually stabilizes around 0.8027 by epoch 32, with minimal increases thereafter, while the training loss plateaus around 0.1190.

3) VALIDATION ACCURACY AND VALIDATION LOSS RESULT

The figure 15 represents the *validation loss* and its relationship with the epoch. Initially, *validation accuracy* increases sharply from 0.7563 to approximately 0.7998 by epoch 11, while validation loss drops significantly from 0.9769 to around 0.2538 in the same period. This indicates that the model is effectively learning and generalizing well initially. However, after epoch 11, *validation accuracy* fluctuates slightly, stabilizing around 0.7982 from epoch 22 onwards. Concurrently, *validation loss* shows minor fluctuations but generally stabilizes around 0.2686. The stabilization of both metrics suggests that the model's performance on the validation set has plateaued, indicating that additional training epochs yield diminishing improvements in model accuracy and loss.

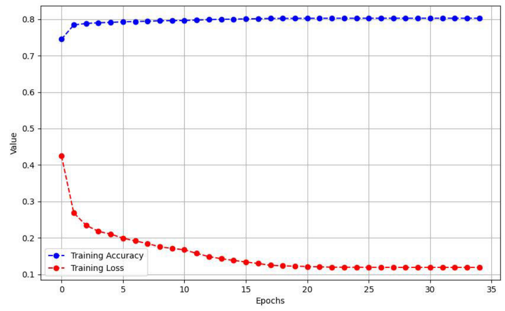

FIGURE 14. Training accuracy and training loss of DeepLabv3plus+ResNet50 for CVC-ColonDB data.

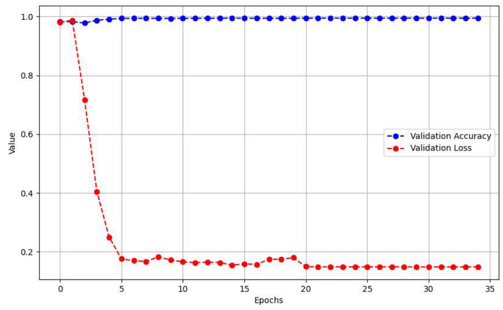

FIGURE 15. Validation accuracy and validation loss of DeepLabv3plus+ResNet50 for CVC-ClinicDB data.

TABLE 3. Performance metrics of DeepLabv3plus with ResNet50 model on CVC-ColonDB image dataset.

Metrics	Results
Accuracy	0.9579
Precision	0.7845
Recall	0.7481
F1-Score	0.7659

D. TESTING RESULTS OF DeepLabv3plus+ResNet50 MODEL FOR CVC-ClinicDB DATA

The performance of DeepLabv3plus+ResNet50 Model on CVC-ClinicDB data has been observed as shown in the table 3. The table shows the evaluation metrics during the testing process using the CVC-ColonDB image dataset.

The image 16 shows the segmentation performance of the DeepLabv3plus+ResNet50 model for the CVC-ColonDB image. The leftmost image is the input image, the middle image is the ground truth mask of the input image and the rightmost image is the predicted segmentation mask from the trained model while testing the process on the CVC-ColonDB image dataset.

The *learning rate* has been updated automatically as 10^{-4} up to epoch 11, 10^{-5} up to epoch 16, 10^{-6} up to epoch 21, 10^{-7} up to epoch 26, 10^{-8} up to epoch 31 and 10^{-9} up to epoch 35.

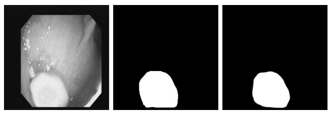
FIGURE 16. **Segmentation result of DeepLabv3plus+ResNet50 on CVC-ColonDB.**

1) ROC AND PR CURVES

The *ROC* and *PR* Curves of DeepLabv3+ with ResNet50 Model for CVC-ColonDB testing data have been generated as shown in figure 17 with *AUC* of 0.87 and 18 with *AUC* of 0.75 respectively.

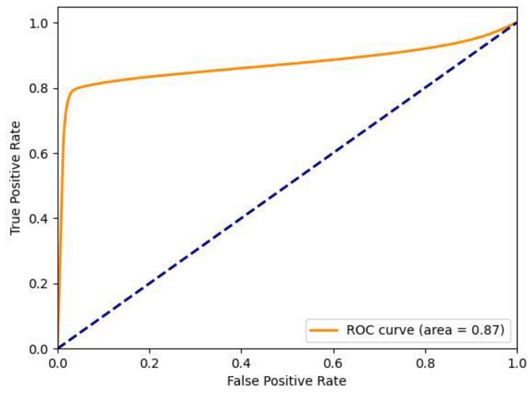

FIGURE 17. **ROC curve for DeepLabv3plus+ResNet50 on CVC-ColonDB data.**

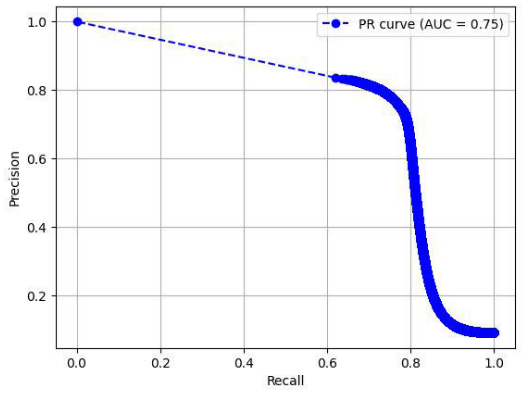

FIGURE 18. **PR curve for DeepLabv3plus+ResNet50 on CVC-ColonDB data.**

2) TRAINING RESULTS OF DeepLabv3plus+ResNet50 MODEL ON HyperKVasir DATA

3) DICE COEFFICIENT AND VALIDATION DICE COEFFICIENT RESULTS

The figure 19 shows the *dice coefficient* of a model at each epoch during the training process. The *dice coefficient*, representing training performance, shows a consistent increase from 0.6372 in the initial epoch to 0.9810 in the final epoch, indicating that the model is progressively improving its ability to segment the training data accurately. Conversely, the *validation dice coefficient* starts at a low value of 0.2409, decreases slightly, and then gradually increases to 0.8604, peaking around epoch 19 before stabilizing with minor fluctuations thereafter.

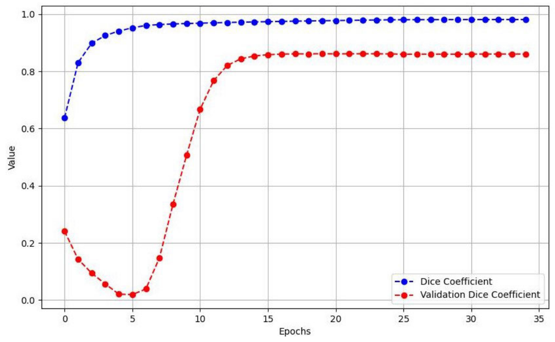

FIGURE 19. **Dice coefficient and validation dice coefficient of DeepLabv3+ResNet50 for HyperKVasir data.**

4) TRAINING ACCURACY AND TRAINING LOSS RESULTS

The figure 20 represents the relationship between epoch and loss of a model during the training phase. As the number of epochs increases, *training accuracy* shows a steady improvement from 0.8794 in the initial epoch to 0.9852 in the final epoch, indicating that the model is learning and fitting the training data progressively better. Simultaneously, *training loss* decreases consistently from 0.3628 to 0.0190, demonstrating that the model's predictions are becoming more accurate concerning the true labels. This inverse relationship between *training accuracy* and training loss is typical during model training, where higher accuracy corresponds with lower loss, reflecting improved model performance.

5) VALIDATION ACCURACY AND VALIDATION LOSS RESULTS

The figure 21 represents the validation Loss and its relationship with the epoch.

The *validation accuracy* starts at 0.6988 and increases steadily, reaching 0.9185 by epoch 10 and continuing to improve until it stabilizes around 0.9536 from epoch 22 onwards. Simultaneously, *validation loss* shows a different pattern. It begins at 0.7590, increases to 0.9806 by epoch 5, indicating some initial instability, and then starts to decrease steadily, reaching around 0.1391 by epoch 10 and stabilizing around this value for the remaining epochs.

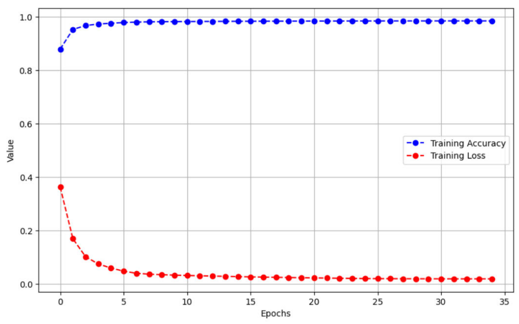

FIGURE 20. Training accuracy and validation loss of DeepLabv3plus+ResNet50 for HyperKVasir data.

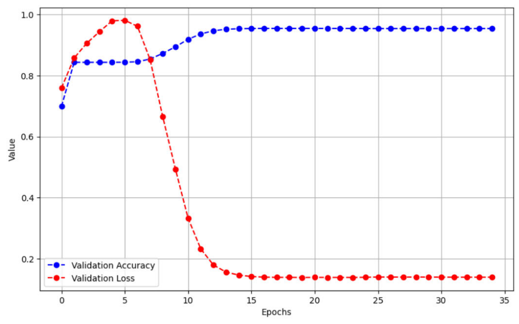

FIGURE 21. Validation accuracy and validation loss of DeepLabv3plus+ResNet50 for HyperKVasir data.

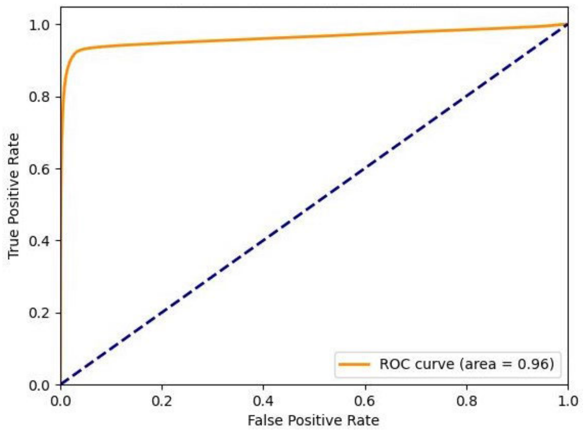

FIGURE 22. ROC curve for DeepLabv3plus+ResNet50 on HyperKVasir data.

E. TESTING RESULTS OF DeepLabv3plus+ResNet50 MODEL ON HYPERKVASIR DATA

1) ROC CURVES

The *ROC* and *PR* Curves of DeepLabv3+ with ResNet50 Model for HyperKVasir testing data have been generated as shown in figure 22 with *AUC* of 0.96 and 23 with AUC of 0.93 respectively.

The performance of DeepLabv3plus+ResNet50 Model on HyperKVasir data has been observed as shown in the table 4. The table shows evaluation metrics during the testing process using the HyperKVasir image dataset. The learning rate has been updated automatically as 10^{-4} up to epoch 5, 10^{-5} up to epoch 25, and 10^{-6} up to epoch 30, and 10^{-7} up to epoch 35. The image 24 shows the segmentation performance of the DeepLabv3plus+ResNet50 model for the HyperKVasir image. The leftmost image is the input image, the middle image is the ground truth mask of the input image and the rightmost image is the predicted segmentation mask from the trained model while testing the process on the HyperKVasir image dataset.

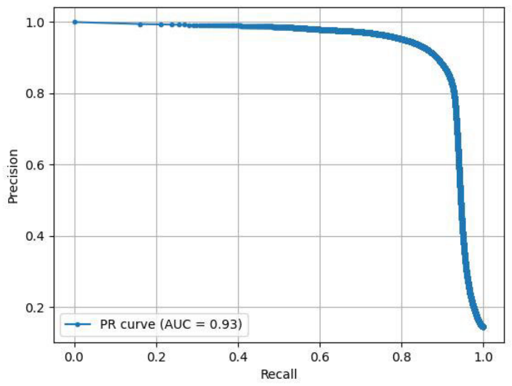

FIGURE 23. PR curve for DeepLabv3plus+ResNet50 on HyperKVasir data.

TABLE 4. Performance metrics of DeepLabv3plus with ResNet50 model on HyperKVasir image dataset.

Metrics	Results
Accuracy	0.9701
Precision	0.9168
Recall	0.8698
F1-Score	0.8927

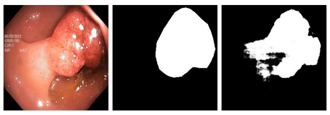

FIGURE 24. Segmentation result of DeepLabv3plus+ResNet50 on HyperKVasir data.

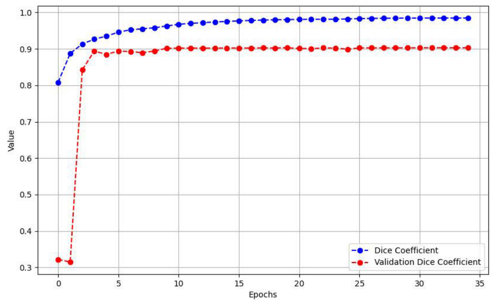

FIGURE 25. Dice coefficient and validation coefficient of DeepLabv3plus+ResNet50 for ISIC data.

F. TRAINING RESULTS OF DEEPLABV3PLUS+RESNET50 MODEL ON ISIC DATA

1) DICE COEFFICIENT AND VALIDATION DICE COEFFICIENT RESULTS

The figure 25 shows the *dice coefficient* and *validation dice coefficient* of a model at each epoch during the training process.

The *dice coefficient* shows a consistent increase, indicating that the model's performance on the training data is improving steadily. Conversely, the *validation dice coefficient* exhibits less fluctuation and eventually stabilizes around 0.902, suggesting that after a certain point, further training epochs do not significantly enhance the model's generalization on unseen data.

2) TRAINING ACCURACY AND TRAINING LOSS RESULTS

The figure 26 represents the relationship between epoch and loss of a model during the training phase. As training progresses, the model shows a substantial increase in accuracy and a significant decrease in loss with each epoch, reflecting rapid learning and model refinement. Over time, the rate of improvement in both accuracy and loss diminishes, indicating that the model is approaching its optimal performance. By epoch 34, the *training accuracy* has nearly plateaued at around 0.9927, while the *training loss* has similarly approached a stable low value of around 0.0152.

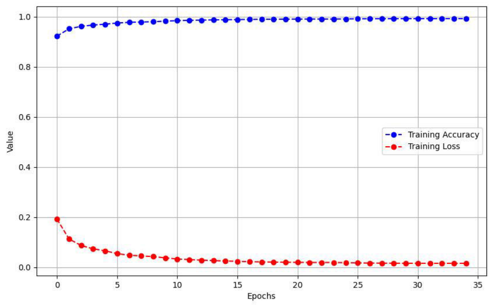

FIGURE 26. Training accuracy and training loss of DeepLabv3plus+ResNet50 for ISIC data.

3) VALIDATION ACCURACY AND VALIDATION LOSS RESULTS

The figure 27 represents the *validation loss* and its relationship with the epoch. As epochs progress, *validation accuracy* increases significantly while *validation loss* decreases, showing that the model is learning and improving its predictions. The *validation accuracy* stabilizes around 0.954, while the validation loss converges to a value slightly above 0.097.

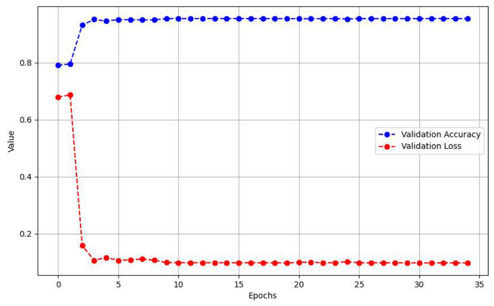

FIGURE 27. Validation accuracy and validation loss of DeepLabv3plus+ResNet50 for ISIC data.

G. TESTING RESULTS OF DeepLabv3plus+ResNet50 MODEL ON ISIC DATA

The performance of DeepLabv3plus+ResNet50 Model on ISIC Image has been observed as shown in the table 5. The table shows the evaluation metrics of the model during the testing process using the ISIC image dataset. The *learning rate* has been updated automatically as 10^{-4} up to epoch 8, 10^{-5} up to epoch 25, and 10^{-6} up to epoch 29, and 10^{-7} up to epoch 35. The image 28 shows the segmentation performance of the DeepLabv3plus with ResNet50 model for the ISIC image. The leftmost image is the input image, the middle image is the ground truth mask of the input image and the rightmost image is the predicted segmentation mask from the trained model while testing the process on the ISIC image dataset.

TABLE 5. Performance metrics of DeepLabv3plus with ResNet50 model on ISIC image dataset.

Metrics	Results
Accuracy	0.9603
Precision	0.9210
Recall	0.8923
F1-Score	0.9064

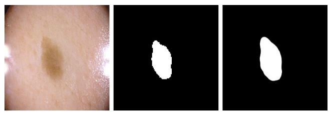

FIGURE 28. Segmentation result of DeepLabv3plus+ResNet50 on ISIC data.

1) ROC AND PR CURVES

The *ROC* Curve of DeepLabv3+ with ResNet50 Model for ISIC testing data have been generated as shown in figure 29 with *AUC* of 0.94 and 30 with *AUC* of 0.92 respectively.

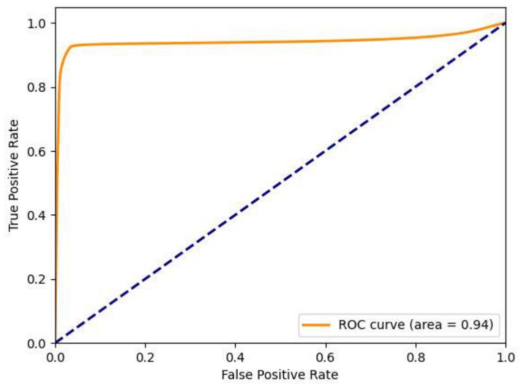

FIGURE 29. ROC curve for DeepLabv3plus+ResNet50 on ISIC data.

H. TRAINING RESULTS OF UNet-ViT MODEL FOR BRAIN TUMOR DATASET

1) DICE COEFFICIENT AND VALIDATION DICE COEFFICIENT RESULTS

The figure 31 shows the *dice coefficient* and *validation dice coefficient* of the model at each epoch during the training process.

The *dicecoefficient* rises from 0.0344 in the early epochs to 0.8014 by the end, suggesting the model progressively learns

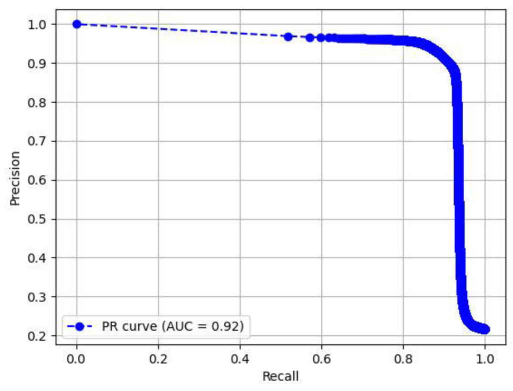

FIGURE 30. PR curve for DeepLabv3plus+ResNet50 on ISIC data.

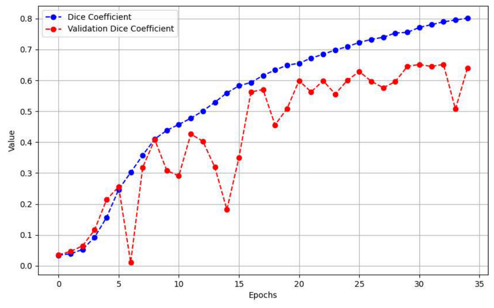

FIGURE 31. Dice coefficient and validation dice coefficient for brain tumor dataset.

and enhances its segmentation capabilities. Conversely, the *validation dice coefficient* measures performance on a separate validation set, which is crucial for assessing how well the model generalizes to unseen data. The validation metric exhibits more fluctuation compared to the training metric, peaking at around 0.6518 in the later epochs, but also showing some declines.

2) TRAINING ACCURACY AND TRAINING LOSS RESULTS

The figure 32 represents the relationship between epoch and *training accuracy* and *training loss* of a model during the training phase. As the number of epochs increases, the model typically shows improvement in both *training accuracy* and training loss. Initially, the *training accuracy* starts low and the *training loss* is high, but as training progresses through more epochs, accuracy steadily increases while loss decreases.

The trend indicates that the model is learning and improving its ability to make correct predictions. Specifically, from epoch 0 to epoch 34, the *training accuracy* rises from approximately 12.3% to around 99.3%, and the *training loss* decreases from about 0.97 to 0.20.

3) VALIDATION ACCURACY AND VALIDATION LOSS RESULTS

The figure 33 represents the *validation accuracy* and *validation loss* and its relationship with the epoch.

As the number of epochs increases, *validation accuracy* typically improves, the *validation accuracy* starts at 0.0287 and rises sharply to 0.9893 by the 34th epoch, indicating that the model learns effectively from the data over time. Conversely, *validation loss*, which measures how well the model's predictions match the actual values, generally decreases as training progresses. The loss decreases from 0.9651 to 0.3393 over the same period, showing that the model's predictions become more accurate.

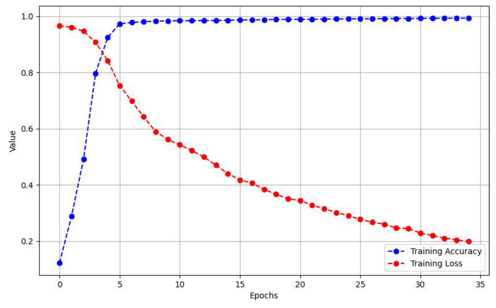

FIGURE 32. Training accuracy and training loss for brain tumor data.

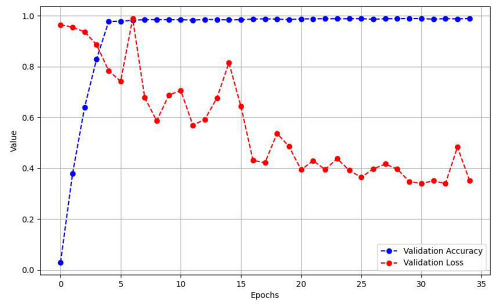

FIGURE 33. Validation accuracy and validation loss for brain tumor data.

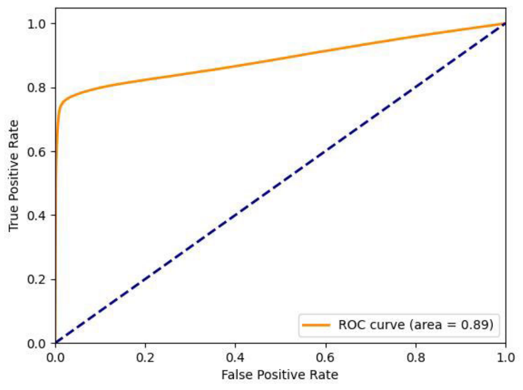

FIGURE 34. ROC curve for UNet-ViT on brain tumor data.

I. TESTING RESULTS OF UNET-VIT MODEL ON BRAIN TUMOR DATA

1) ROC CURVE AND PR CURVE

The *ROC* and *PR* Curves of the UNet-ViT Model for the Brain Tumor testing data have been generated as shown in figure 34 with *AUC* of 0.89 and 35 with *AUC* of 0.67 respectively.

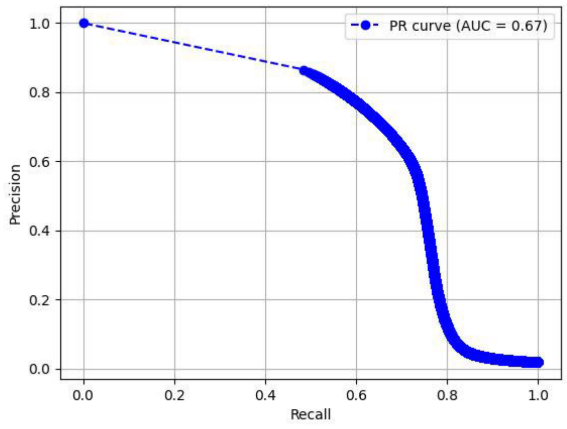

FIGURE 35. PR curve for UNet-ViT on brain tumor data.

On analyzing these results for the brain tumor dataset, the true positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN) are found to be as shown in the table 6. The table shows the evaluation metrics of the model during the testing process using the brain tumor image dataset.

TABLE 6. Performance metrics of UNet with ViT model on brain tumor image dataset.

Metrics	Results
Accuracy	0.9894
Precision	0.7444
Recall	0.6247
F1-Score	0.6794

The image 36 shows the segmentation performance of the UNet-ViT model for the Brain Tumor data. The leftmost image is the input image, the middle image is the ground truth mask of the input image and the rightmost image is the predicted segmentation mask from the trained model while testing the process on the brain tumor image dataset.

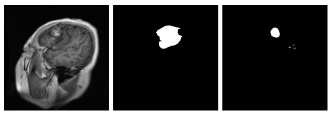

FIGURE 36. Segmentation result of UNet-ViT on brain tumor on brain tumor data.

J. TRAINING RESULTS OF UNet-ViT MODEL USING CVC-ColonDB DATA

1) DICE COEFFICIENT AND VALIDATION DICE RESULTS

The figure 37 shows the *dice coefficient* and *validation coefficient* of the model at each epoch during the training process.

The *dice coefficient* increases from 0.2707 at epoch 0 to 0.6758 at epoch 34, suggesting that the model's performance on the training data is improving over time. The

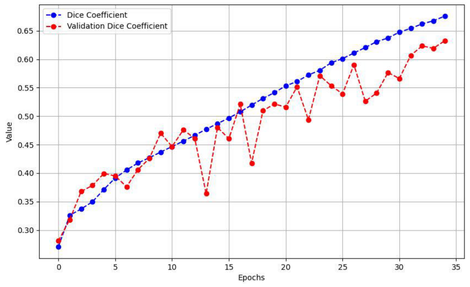

FIGURE 37. Dice coefficient and validation dice coefficient of UNet-ViT model on CVC ColonDB data.

validation dice coefficient rises from 0.2809 to 0.6325, there are periods of decline, such as between epochs 14 and 17.

2) TRAINING ACCURACY AND TRAINING LOSS RESULTS

The figure 38 represents the relationship between epoch, *training accuracy*, and *training loss* of the model during the training phase.

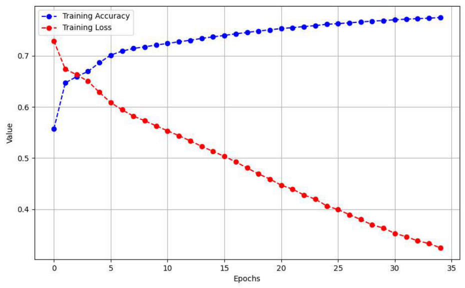

FIGURE 38. Training accuracy and training loss of UNet-ViT model on CVC-ColonDB data.

The accuracy gradually increases from 55.7% at epoch 0 to 77.5% at epoch 34, indicating that the model is learning and performing better. Simultaneously, the *training loss*, which starts at 0.729, decreases over time, reaching 0.324 by epoch 34.

3) VALIDATION ACCURACY AND VALIDATION LOSS RESULTS

The figure 39 represents the *validation accuracy* and *validation loss* and its relationship with the epoch. The *validation accuracy* rises consistently from 0.472 to a peak of 0.785, indicating a steady improvement in the model's performance. Conversely, *validation loss*, which quantifies the model's prediction error on the validation set, tends to decrease as epochs progress, suggesting that the model's predictions are becoming more accurate. The *validation loss* drops from 0.718 to 0.363, aligning with the increasing *validation accuracy* and showcasing the model's enhancement in minimizing prediction errors over time.

K. TESTING RESULTS OF UNET-VIT MODEL FOR CVC-ColonDB DATA

1) ROC AND PR CURVES

The *ROC* and *PR* Curves of the UNet-ViT Model for the CVC-ColonDB testing data have been generated as shown in figure 40 and 41 with *AUC* of 0.83 and 0.63 respectively.

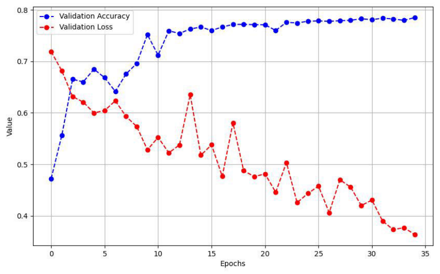

FIGURE 39. Validation accuracy and validation loss of UNet-ViT model for CVC-ColonDB data.

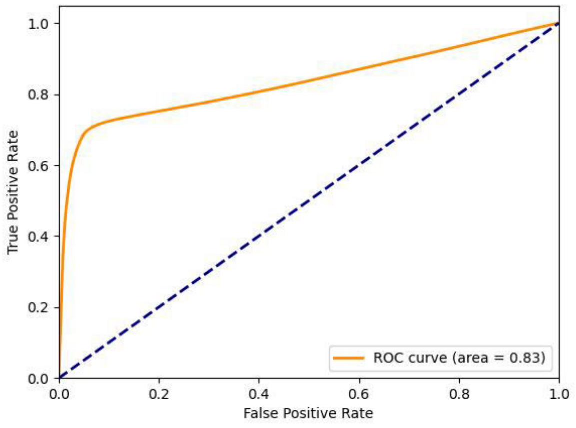

FIGURE 40. ROC curve for UNet-ViT on CVC-ColonDB data.

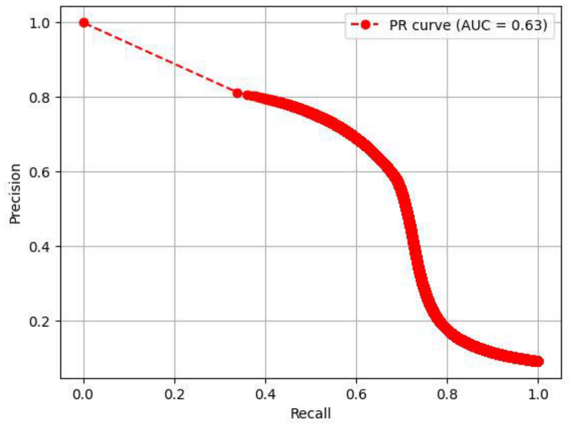

FIGURE 41. PR curve for UNet-ViT on CVC-ColonDB data.

On analyzing these results for the CVC-ColonDB dataset, the true positives (*TP*), False Positives (*FP*), True Negatives (*TN*), and False Negatives (*FN*) are found to be as shown in the table 7. The table shows the evaluation metrics of the model during the testing process using the CVC-ColonDB image dataset. The image 42 shows the segmentation performance of the UNet-ViT model for the CVC-ColonDB data. The leftmost image is the input image, the middle image is the ground truth mask of the input image and the rightmost image is the predicted segmentation mask from the trained model while testing the process on the CVC-ColonDB image dataset.

TABLE 7. Performance metrics performance metrics of UNet with ViT model on CVC-ColonDB image dataset.

Metrics	Results
Accuracy	0.9387
Precision	0.6964
Recall	0.5934
F1-Score	0.6408

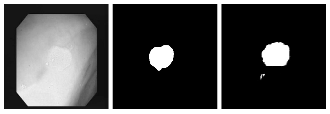

FIGURE 42. Segmentation result of UNet-ViT on CVC-ColonDB data.

L. TRAINING RESULTS OF UNet-ViT MODEL USING HyperKVasir DATA

1) DICE COEFFICIENT AND VALIDATION DICE COEFFICIENT RESULTS

The figure 43 shows the dice coefficient and validation dice coefficient of the model at each epoch during the training process.

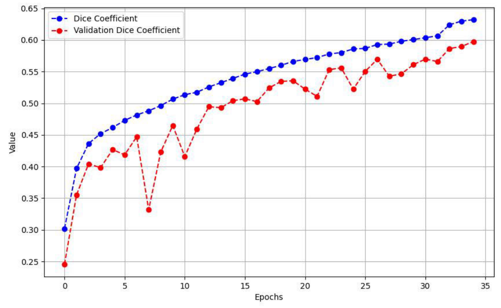

FIGURE 43. Dice coefficient and validation dice coefficient of UNet-ViT model for HyperKVasir data.

The *dice coefficient* started at 0.302 and steadily increased to 0.633. Similarly, the *validation dice coefficient* exhibited an upward trend, starting from 0.246 and reaching up to 0.598. The fluctuations in the *validation dice coefficient*, particularly around epochs 8, 9, and 22, could indicate some degree of over-fitting or data variance, but the overall trend demonstrates significant progress in the segmentation capability.

2) TRAINING ACCURACY AND TRAINING LOSS RESULTS

The figure 44 represents the relationship between epoch, *training accuracy*, and *training loss* of a model during the training phase. The *training accuracy* starts at 0.4862 and gradually increases to 0.8727 by the end of the epochs, showing a steady enhancement in the model's ability to correctly segment the images. Simultaneously, the *training loss* decreases from 0.6979 to 0.3678, demonstrating a reduction in errors made by the model.

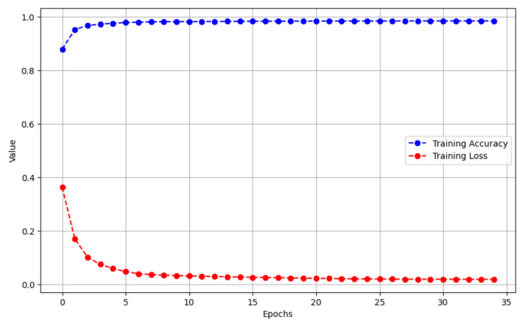

FIGURE 44. Training accuracy and loss of UNet-ViT model for HyperKVasir data.

3) VALIDATION ACCURACY AND VALIDATION LOSS RESULTS

The figure 45 represents the *validation accuracy* and *validation loss* and its relationship with the epoch. The *validation accuracy* showed a consistent upward trend, starting at 0.740 and reaching up to 0.876. Notably, the accuracy surpassed the 0.85 mark around the 17th epoch and continued to improve, demonstrating the model's increasing ability to correctly identify the segments in the images. Similarly, the validation loss steadily declined, beginning at 0.750 and dropping to 0.398 by the last epoch.

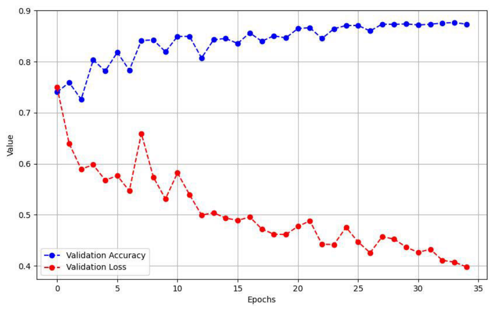

FIGURE 45. Validation accuracy and validation loss of UNet-ViT model for HyperKVasir data.

M. TESTING RESULTS OF UNET-VIT MODEL FOR HyperKVasir DATA

1) ROC AND PR CURVES

The *ROC* and *PR* Curves of the UNet-ViT Model for the HyperKVasir testing data have been generated as shown in figure 46 and 47 with *AUC* of 0.93 and 0.59 respectively.

On analyzing these results for the HyperKVasir dataset, the true positives (*TP*), False Positives (*FP*), True Negatives (*TN*), and False Negatives (*FN*) are found to be as shown in the table 8. The table shows the evaluation of the model during the testing process using the HyperKVasir image dataset. The *learning rate* has been updated automatically as 10^{-2} from epoch 33 up to epoch 35. The image 48 shows the segmentation performance of the UNet-ViT model for the HyperKVasir data. The leftmost image is the input image, the middle image is the ground truth mask of the input image and the rightmost image is the predicted segmentation mask from the trained model while testing the process on the HyperKVasir image dataset.

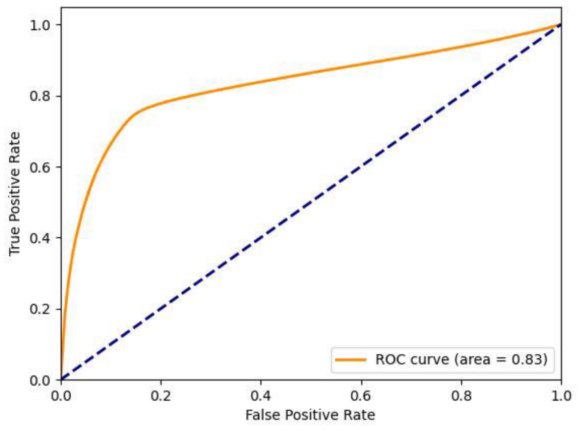

FIGURE 46. ROC curve for UNet-ViT on CVC-ColonDB data.

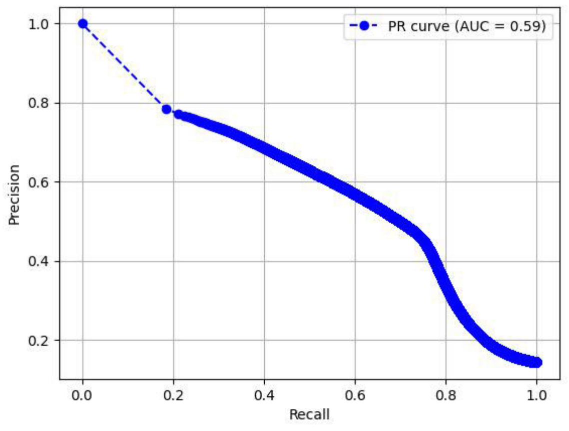

FIGURE 47. PR curve for UNet-ViT on HyperKVasir data.

TABLE 8. Performance metrics of UNet with ViT model on HyperKVasir image dataset.

Metrics	Results
Accuracy	0.8758
Precision	0.5606
Recall	0.6117
F1-Score	0.5851

N. TRAINING RESULTS OF UNet-ViT MODEL USING ISIC DATA

1) DICE COEFFICIENT AND VALIDATION DICE COEFFICIENT RESULTS

The figure 49 shows the *dice coefficient* and *validation dice coefficient* of the model at each epoch during the training process.

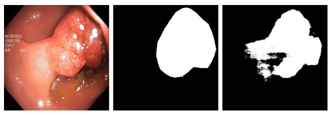

FIGURE 48. Segmentation result of UNet-ViT on HyperKVasir data.

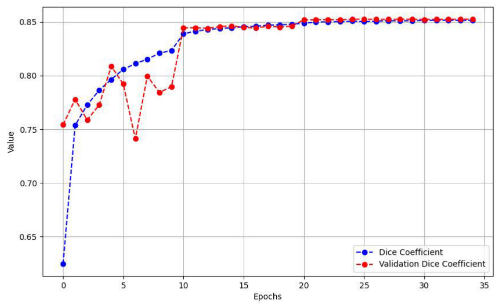

FIGURE 49. Dice coefficient and validation dice coefficient of UNet-ViT model for ISIC data.

The *dice coefficient* values for the training set start at 0.6247 and gradually improve, reaching 0.8514 by the final epoch. This steady rise reflects the model's learning and adaptation to the data. The *validation dice coefficient* follows a similar trend, starting at 0.7541 and increasing to 0.8525. Although there are minor fluctuations in both metrics, the overall upward trajectory signifies that the model is effectively generalizing and performing well on unseen data.

2) TRAINING ACCURACY AND TRAINING LOSS RESULTS

The figure 50 represents the relationship between epoch, *training accuracy*, and *training loss* of a model during the training phase. The *training accuracy* increased from 0.7966 to 0.9341, indicating that the model became progressively better at correctly classifying the images. Concurrently, the *training loss* decreased from 0.3753 to 0.1486, showing that the model's predictions became more accurate with reduced errors over time.

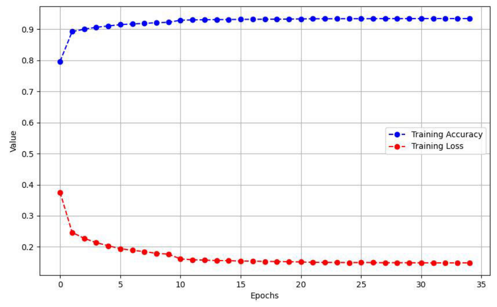

FIGURE 50. Training accuracy and training loss of UNet-ViT model for ISIC data.

3) VALIDATION ACCURACY AND VALIDATION LOSS RESULTS

The figure 51 represents the *validation accuracy* and *validation loss* and its relationship with the epoch. The *validation accuracy* started at 0.885647178 and consistently improved, reaching a peak of 0.933357298 by the final epoch. This indicates a steady enhancement in the model's ability to correctly identify and segment the images. Concurrently, the *validation loss* showed a decreasing trend, beginning at 0.247276068 and reducing to 0.148534998.

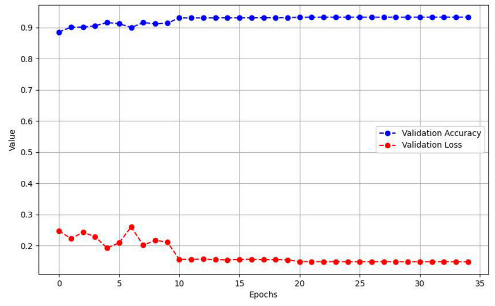

FIGURE 51. Validation accuracy and validation loss of UNet-ViT model for ISIC data.

O. TESTING RESULTS OF UNET-VIT MODEL FOR ISIC DATA

1) ROC AND PR CURVES

The *ROC* and *PR* Curve of the UNet-ViT Model for the ISIC testing data have been generated as shown in figure 52 and 53 with *AUC* of 0.96 and 0.91 respectively.

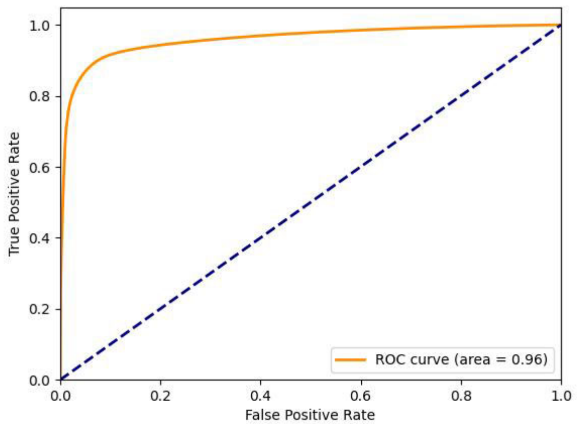

FIGURE 52. ROC curve for UNet-ViT on ISIC data.

On analyzing these results for the ISIC dataset, the *accuracy*, *precision*, *recall*, and $F1 - score$ of the model during the testing process using the ISIC image dataset. are found to be as shown in the table 9. The *learning rate* has been updated automatically as 10^{-2} from epoch 10 up to epoch 19, 10^{-3} from epoch 11 up to epoch 29 and 10^{-7} from epoch 30 up to epoch 35. The image 54 shows the segmentation performance of the UNet-ViT model for the ISIC data. The leftmost image is the input image, the middle image is the ground truth mask of the input image and the rightmost image is the predicted segmentation mask from the trained model while testing the process on the ISIC image dataset.

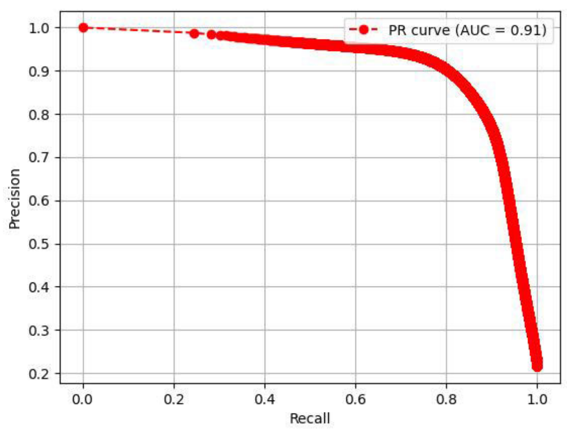

FIGURE 53. PR curve for UNet-ViT on ISIC data.

TABLE 9. Performance metrics of UNet with ViT model on ISIC image dataset.

Metrics	Results
Accuracy	0.9378
Precision	0.8713
Recall	0.8345
F1-Score	0.8525

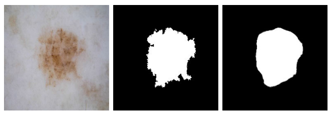

FIGURE 54. Segmentation result of UNet-ViT on ISIC data.

IV. DISCUSSION AND CONCLUSION

A deep learning model may have high values of False Positives (*FP*) and False Negatives (*FN*) for a variety of reasons about the data, model architecture, and training procedure. An unbalanced dataset with substantially more negative than positive occurrences can lead to high *FP* because the model is prone to predict positives to boost recall. The model may classify more cases as positive than it should if the decision threshold is set too low. Overfitting represents a further possible problem, whereby the model overfits to incoming data, producing false positive classifications, by learning spurious correlations and noise in the training set. Due to the possibility of the model becoming confused by irrelevant or noisy characteristics, misleading features in the dataset can potentially lead to high *FP*.

An imbalanced dataset may also be the cause of high *FN*, as the model may be biased toward predicting negatives to improve precision due to a lack of positive cases. Positive cases may be overlooked if the model is underfitted, which occurs when the model is too basic to adequately represent the complexity of the data. Positive cases may be missed by the model due to feature insufficiency, a situation in which the features lack sufficient information. High *FN* might also result from selecting a model architecture that is improper or unsuitable for the task. True patterns might be hidden by noisy or low-quality data, which makes it more difficult for the model to find positive examples.

DeepLabv3plus and UNet have been used to perform semantic segmentation of medical images. ResNet50 and Vision Transformers have been successfully implemented to enhance the performance of these existing architectures. The dataset was collected from multiple sources. The collected dataset had unequal distribution for each class. To address the inefficient and unequal distribution of data, multiple data prep-recessing techniques are employed. The preprocessed dataset has been divided into training, validation, and testing sets. The experimental observations require some dedicated software and hardware requirements and are fulfilled by Kaggle and Google Colaboratory. Following the end of the training procedure, the trained model was verified and tested using the validation and testing datasets. Some significant experimental discoveries have been obtained when implementing multi-class medical image segmentation. When used with efficient and massive datasets, the updated architectures deliver exceptional performance and produce better results.

Experimental results have shown that the overall *accuracy* of the DeepLabv3plus in combination with ResNet50 is 0.9949 with a *precision* of 0.8904, *sensitivity* of 0.8153, and $F1-Score$ 0.8512 for brain tumor data. The model has shown an *accuracy* of 0.9579 with a *precision* of 0.7845, a *sensitivity* of 0.7481, and $F1-Score$ 0.7659 for CVC-ColonDB data. The model has shown an *accuracy* of 0.9701 with a *precision* of 0.9168, a *sensitivity* of 0.8698, and $F1-Score$ 0.8927 for HyperKVasir data. The model has shown an *accuracy* of 0.9603 with a *precision* of 0.9210, a *sensitivity* of 0.8923, and an $F1-Score$ 0.9064 for ISIC data.

Similarly, the UNet model in combination with the vision transformer as the backbone model has shown an *accuracy* of 0.9894, *precision* of 0.7444, *sensitivity* of 0.6247, and $F1-Score$ 0.6794 for brain tumor data. The model has shown an *accuracy* of 0.9387, *precision* of 0.6964, *sensitivity* of 0.5934, and $F1-Score$ 0.6408 for CVC-ColonDB data. The model has shown an *accuracy* of 0.8758, *precision* of 0.5606, *sensitivity* of 0.6117, and $F1-Score$ 0.5851 for HyperKVasir data. The model has shown an *accuracy* of 0.9378, *precision* of 0.8713, *sensitivity* of 0.8345, and $F1-Score$ 0.8525 for ISIC data.

The ResNet50 and Vision transformers are found to be extremely scalable and effective at handling enormous datasets and processing images concurrently. With the thesis contributions, the medical personnel may diagnose patients more effectively using the model's decision-making process and informative insights. In the future, the existing deep learning models can be incorporated with any other hybrid deep learning models for better analysis and prediction purposes. The hybrid model could be crucial in better understanding data and its features.

The Deeplabv3+ with ResNet50 model has the Atrous convolutions mechanism which allows the model to capture multi-scale contextual information, which can enhance resilience to input variations such as lighting variations. Atrous convolution enables the model to focus on spatial structures and patterns rather than strictly relying on pixel intensities, which can be affected input variations [20]. Moreover, the ASPP module allows the model to capture features at different spatial resolutions, helping it be more robust to input variations by providing a multi-scale representation of features [21]. ResNet50 is well-suited for extracting local features like edges and textures, but input variations may affect these local features but the impacts can be reduced after supplying the achieved features to the Deeplabv3+ model. On the other hand, the UNet with Vision Transformer model has Vision Transformer (ViT) as the backbone network followed by UNet model [8]. The encoder-decoder structure with skip connections of UNet allows it to preserve fine details and spatial localization [12]. Skip connections from the encoder to the decoder ensure that the model retains local features, which are often input-invariant, from shallow layers. This helps in handling segmentation tasks where precise boundary delineation is crucial. ViT includes capturing global dependencies and understanding the overall structure. This global view is less sensitive to local input variations, as ViT can focus on anatomical structures and shape patterns rather than intensity variations caused by lighting. The self-attention mechanism in ViT allows the model to weigh different parts of the image independently, which can make it more robust to varying input variations. Even if lighting changes affect certain regions, the ViT can focus on essential parts of the image based on context rather than brightness alone [17]. ViTs rely less on local pixel values than convolutional layers. As a result, they can be less affected by input variations, which often change local pixel intensities but leave the overall structure intact.

V. FUTURE DIRECTIONS

A. INSTANCE SEGMENTATION

Instance segmentation is one of the image segmentation where all pixels corresponding to each object share a unique pixel value. For example in the image of a human cell, all pixels in nuclei share the same pixel value. This type of approach can be very useful if we are interested in object parameters for each object in the input image. When instances are to be differentiated between the instances of the same class, instance segmentation is used which is closely related to object detection. Image segmentation aims to find the boundaries of specific features. The instance segmentation includes identifying and segmenting every object in an image [25]. With the additional responsibility of segmenting the object's boundaries, it is comparable to object detection. The algorithm separates overlapping objects without specifying the region's class. Applications that require the identification and tracking of individual objects can benefit from instance segmentation. A comprehensive knowledge of all the instances that are present in the image is provided by instance segmentation, which assigns each pixel a unique label.

B. PANOPTIC SEGMENTATION

A comprehensive strategy that combines instance and semantic segmentation is called panoptic segmentation. It is designed to provide a thorough comprehension of the scene's general semantic composition as well as each object in it. The word ''panoptic'' refers to the ability to see everything in one vision. Panoptic segmentation is a sophisticated technique that classifies each pixel in an image based on its class label while identifying the particular instance of that class to which it belongs [26]. It is not merely a simple combination of its counterparts. Using panoptic segmentation, for example, multiple cars in an image would be recognized and differentiated, with a unique instance ID assigned to each car. Although instance segmentation recognizes unique objects without necessarily classifying every pixel and semantic segmentation assigns pixels to their corresponding classes without discriminating between specific instances, panoptic segmentation accomplishes both tasks. A label denoting the class of each pixel and an instance number corresponding to each pixel in an image that has been processed with panoptic segmentation. Pixels in ''stuff'' regions (such as the sky or pavement), which are more difficult to measure, may have an instance number that relates to that category, or none at all. In contrast, distinct instance IDs would be assigned to pixels that are part of objects. This sophisticated segmentation method has potential uses in digital image processing, self-driving cars, and medical imaging, among other domains.

ACKNOWLEDGMENT

This research work would not have been possible without the guidance and help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this study. First of all, the authors would like to express sincere gratitude to Asst. Prof. Dinesh Baniya Kshatri, M.Sc. Coordinator, for providing invaluable guidance, insightful comments, meticulous suggestions, and encouragement for the conduct of this research work. They are also thankful to Department of Electronics and Computer Engineering, Thapathali Campus, for providing a suitable environment for the research work. They are also grateful to their friends for offering them advice and moral support.

REFERENCES

[1] Q. Liu, C. Kaul, J. Wang, C. Anagnostopoulos, R. Murray-Smith, and F. Deligianni, ''Optimizing vision transformers for medical image segmentation,'' in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2023, pp. 1–5.

[2] R. Wang, T. Lei, R. Cui, B. Zhang, H. Meng, and A. K. Nandi, ''Medical image segmentation using deep learning: A survey,'' *IET Image Process.*, vol. 16, no. 5, pp. 1243–1267, Apr. 2022.

[3] S. Navya, P. Nishitha, and V. Hema, ''Medical image segmentation using deep learning,'' in *Proc. Int. Conf. Advancements Smart, Secure Intell. Comput. (ASSIC)*, Nov. 2022, pp. 1–4.

[4] Y. Gao, M. Zhou, and D. N. Metaxas, "UTNet: A hybrid transformer architecture for medical image segmentation," in *Proc. 24th Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.*, Strasbourg, France. Cham, Switzerland: Springer, Sep. 2021, p. 6171.

[5] X. Huang, Z. Deng, D. Li, and X. Yuan, "MISSFormer: An effective medical image segmentation transformer," 2021, *arXiv:2109.07162*.

[6] S. Li, X. Sui, X. Luo, X. Xu, Y. Liu, and R. Goh, "Medical image segmentation using squeeze-and-expansion transformers," 2021, *arXiv:2105.09511*.

[7] D. Jha, P. H. Smedsrud, M. A. Riegler, D. Johansen, T. D. Lange, P. Halvorsen, and H. D. Johansen, "ResUNet++: An advanced architecture for medical image segmentation," in *Proc. IEEE Int. Symp. Multimedia (ISM)*, Dec. 2019, pp. 225–2255.

[8] D. Jha, N. K. Tomar, V. Sharma, and U. Bagci, "TransNetR: Transformer-based residual network for polyp segmentation with multi-center out-of-distribution testing," in *Proc. Med. Imag. With Deep Learn.*, 2024, pp. 1372–1384.

[9] G.-P. Ji, G. Xiao, Y.-C. Chou, D.-P. Fan, K. Zhao, G. Chen, and L. Van Gool, "Video polyp segmentation: A deep learning perspective," *Mach. Intell. Res.*, vol. 19, no. 6, pp. 531–549, Dec. 2022.

[10] D. Gutman, N. C. F. Codella, E. Celebi, B. Helba, M. Marchetti, N. Mishra, and A. Halpern, "Skin lesion analysis toward melanoma detection: A challenge at the international symposium on biomedical imaging (ISBI) 2016, hosted by the international skin imaging collaboration (ISIC)," 2016, *arXiv:1605.01397*.

[11] K. Pogorelov, K. R. Randel, C. Griwodz, S. L. Eskeland, T. de Lange, D. Johansen, C. Spampinato, D.-T. Dang-Nguyen, M. Lux, P. T. Schmidt, M. Riegler, and P. Halvorsen, "KVASIR: A multi-class image dataset for computer aided gastrointestinal disease detection," in *Proc. 8th ACM Multimedia Syst. Conf. (MMSys)*, New York, NY, USA, 2017, pp. 164–169.

[12] P. Vasuki, J. Kanimozhi, and M. B. Devi, "A survey on image preprocessing techniques for diverse fields of medical imagery," in *Proc. IEEE Int. Conf. Electr., Instrum. Commun. Eng. (ICEICE)*, Apr. 2017, pp. 1–6.

[13] F. M. J. M. Shamrat, S. Azam, A. Karim, R. Islam, Z. Tasnim, P. Ghosh, and F. De Boer, "LungNet22: A fine-tuned model for multiclass classification and prediction of lung disease using X-ray images," *J. Personalized Med.*, vol. 12, no. 5, p. 680, Apr. 2022.

[14] A. M. Reza, "Realization of the contrast limited adaptive histogram equalization (CLAHE) for real-time image enhancement," *J. VLSI Signal Process.-Syst. Signal, Image, Video Technol.*, vol. 38, no. 1, pp. 35–44, Aug. 2004.

[15] M. Z. Khan, M. K. Gajendran, Y. Lee, and M. A. Khan, "Deep neural architectures for medical image semantic segmentation: Review," *IEEE Access*, vol. 9, pp. 83002–83024, 2021.

[16] M. Lai, "Deep learning for medical image segmentation," 2015, *arXiv:1505.02000*.

[17] A. Hatamizadeh, Y. Tang, V. Nath, D. Yang, A. Myronenko, B. Landman, H. R. Roth, and D. Xu, "UNETR: Transformers for 3D medical image segmentation," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2022, pp. 1748–1758.

[18] N. Siddique, S. Paheding, C. P. Elkin, and V. Devabhaktuni, "U-Net and its variants for medical image segmentation: A review of theory and applications," *IEEE Access*, vol. 9, pp. 82031–82057, 2021.

[19] H. Huang, L. Lin, R. Tong, H. Hu, Q. Zhang, Y. Iwamoto, X. Han, Y.-W. Chen, and J. Wu, "UNet 3+: A full-scale connected UNet for medical image segmentation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 1055–1059.

[20] D. Hasan and A. M. Abdulazeez, "Lung segmentation from chest X-ray images using Deeplabv3plus-based CNN model," *Indonesian J. Comput. Sci.*, vol. 13, no. 1, Feb. 2024.

[21] J. Wang and X. Liu, "Medical image recognition and segmentation of pathological slices of gastric cancer based on deeplab v3+ neural network," *Comput. Methods Programs Biomed.*, vol. 207, Aug. 2021, Art. no. 106210.

[22] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.

[23] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*.

[24] F. Jiang, A. Grigorev, S. Rho, Z. Tian, Y. Fu, W. Jifara, K. Adil, and S. Liu, "Medical image semantic segmentation based on deep learning," *Neural Comput. Appl.*, vol. 29, pp. 1257–1265, Mar. 2018.

[25] L. Chen, M. Strauch, and D. Merhof, "Instance segmentation of biomedical images with an object-aware embedding learned with local constraints," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.* Cham, Switzerland: Springer, 2019, pp. 451–459.

[26] O. Elharrouss, S. Al-Maadeed, N. Subramanian, N. Ottakath, N. Almaadeed, and Y. Himeur, "Panoptic segmentation: A review," 2021, *arXiv:2111.10250*.

BIBAT THOKAR received the B.E. degree in computer engineering and the M.Sc. degree in informatics and intelligent systems engineering from Thapathali Campus, Institute of Engineering, Tribhuvan University, Nepal. Since 2019, he has been a Lecturer with the Computer Engineering Department, Lalitpur Engineering College (affiliated to Tribhuvan University). Currently, he is the Head of Department of the Bachelor in Computer Engineering Program. He has been involved in teaching computer science and engineering-related subjects and supervising the projects of undergraduate students. His research interests include computer vision and deep learning.

BINOD SAPKOTA received the B.E. degree in electronics and communication engineering from the Institute of Engineering, Tribhuvan University, and the M.Sc. degree in information system engineering from Purbanchal University. Currently, he is pursuing the Ph.D. degree in computer engineering from Pulchowk Campus, Institute of Engineering, Tribhuvan University. Since 2014, he has been an Assistant Professor with the Department of Electronics and Computer Engineering, Thapathali Campus, Institute of Engineering, Tribhuvan University. He has been involved in teaching electronics and computer engineering related subjects and supervising projects/thesis to bachelor's and graduate students. His research interests include next-generation networking, software-defined networking, machine learning, and cybersecurity.

BABU R. DAWADI received the B.E. degree in computer engineering, in 2003, the M.Sc. degree in information and communication engineering, in 2008, the master's degree in publication administration degree from Tribhuvan University, Nepal, in 2013, and the Ph.D. degree in computer engineering from Institute of Engineering (IOE), Tribhuvan University, in 2021. He worked as the Assistant Director with Nepal Telecommunications Authority, an autonomous ICT regulatory body of Nepal, from December 2009 to November 2012. Since November 2012, he has been a full-time faculty with the Department of Electronics and Computer Engineering, IOE, Pulchowk Campus. He deserves to teach different subjects in computer science and engineering at the bachelor's, master's, and Ph.D. levels. He has ample experience in handling networking and internet server systems with IPv6 research networks at the Center for Information Technology (CIT), IOE, as a System/Network Engineer and a System In-Charge with CIT. He was a Consultant IT specialist at organizations, such as the World Bank and in different ICT policy formulations for the Nepal government. He was the research operator of the AI3/SOI-ASIA IPv6-only network research project, he completed his three months of research on the IPv6 network at Keio University, Japan, and attended several research meetings abroad as a research/training visit and conference presenter. He served the institute in different posts starting as a Network and System Engineer with CIT, the Deputy Head with the Department of Electronics and Computer Engineering, the Chief Technical Officer at the Computer-Based Entrance Exam of IOE, a Board Member of the Entrance and Admission Board of IOE, the Deputy Chief of the Examination Control Division. He was awarded the Best Ph.D. Fellow from Nepal Academy of Science and Technology (NAST). He is the Co-Chairman of the Laboratory for ICT Research and Development (LICT), and the Director of the Network, Cyber Security, and the Digital Forensic Wing of LICT, an ICT Research Laboratory at IOE, TU.

SHASHIDHAR R. JOSHI received the Bachelor of Electrical Engineering degree (Hons.) from Sardar Vallabhbhai Regional College of Engineering, South Gujarat University, Surat, India, in 1984, the Master of Science degree in electrical engineering from the University of Calgary, Canada, in 1992, and the Ph.D. degree from Tribhuvan University, Nepal, in 2007. He was a Research Fellow with Osaka Sangyo University, Japan, from 1998 to 1999. He joined the Institute of Engineering, in 1985. He is currently a Full Professor with the Department of Electronics and Computer Engineering, Institute of Engineering, Pulchowk Campus, and a Visiting Professor with South Asian University, New Delhi, India. He is the former Dean of the Institute of Engineering, Tribhuvan University. His research interests include image processing, networking, and artificial intelligence.