SIMON FRASER UNIVERSITY
ENGAGING THE WORLD

School of Engineering Science

ENSC 813 – DEEP LEARNING SYSTEMS

PROJECT REPORT

Submitted By (From): Nimitha Gopinath Student ID: 301459904	***Submitted Date:*** *11th April 2022*	***Document No:*** 2022-ENSC-813
To: Prof. Ivan Bajic	***Document Title:*** Project Report	

CHAPTER 1

INTRODUCTION

A polyp is a tissue that grows on the colon and rectum that projects into the intestines.There are two types of polyps; cancerous and noncancerous. But, noncancerous polyps can eventually become cancerous and harmful. It is considered as one of the leading causes of cancer-related deaths worldwide [1] . So the best thing we can do is it identify such polyps and treat it before it becomes cancerous. Colonoscopy is the standard procedure for the identification, localization, and removal of colorectal polyps. Due to variability in shape, size, and surrounding tissue similarity, colorectal polyps are often missed by the clinicians during colonoscopy. Over past years, computer aided diagnosis systems have been developed to reduce the miss rate. One such method is by semantic segmentation. The goal of the semantic segmentation is to label each pixel of an image with a corresponding class. There are various types of architectures available in deep learning for this purpose. Here, in this project a simple UNet architecture is developed and compare with a modified UNet architecture with pretrained ResNet50 as encoder.

CHAPTER 2

LITERATURE REVIEW

Over the past decades, researchers have made several efforts at developing prototypes for automated polyp segmentation. Fig 1 shows the various methods available for segmentation. It shows classical segmentation using image processing techniques, machine learning as well as deep learning technique. Out of which, our area of interest is deep learning techniques.

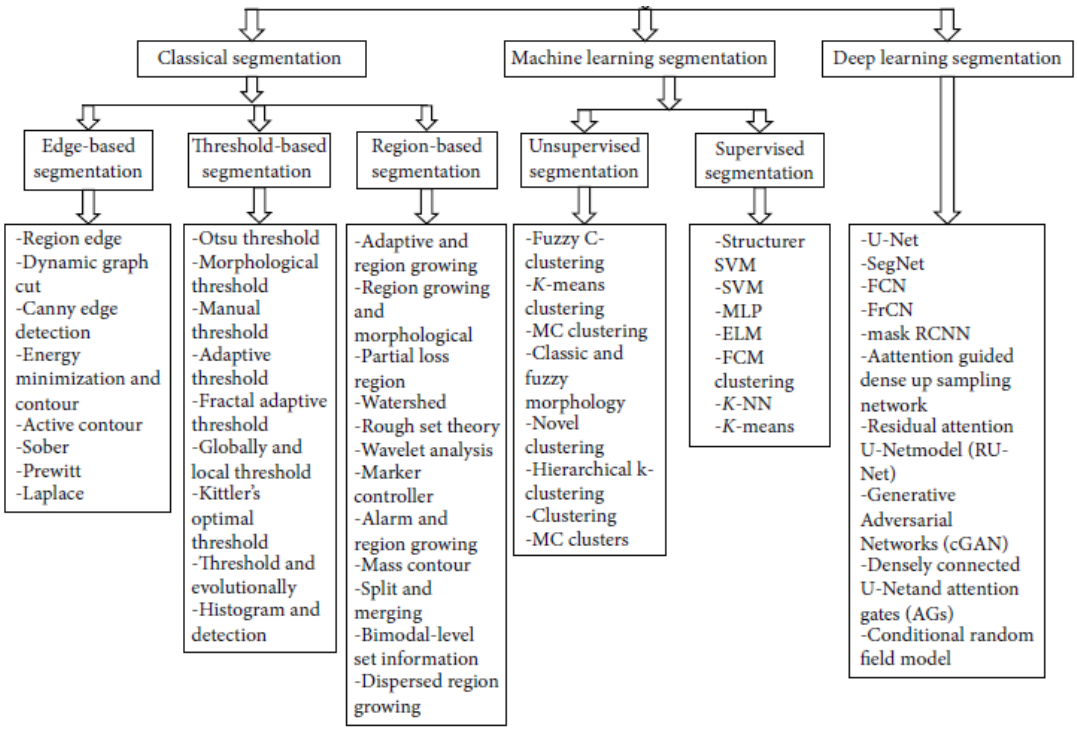

Fig 2.1 Types of semantic segmentation

There were are various research going on and more recent approaches used Convolutional Neural Network (CNN) and pre-trained networks.

Yu et al. [2] adopted a novel offline and online threedimensional deep learning integration framework to automatically detect polyps from colonoscopy videos by leveraging 3D- FCN's. Offline 3D-FCN is first developed and exploited for learning Spatio-temporal features from the training samples extracted from colonoscopy videos. Here explore 3D-CNN to learn spatiotemporal features from colonoscopy videos for automated polyp detection. This is the first approach that employs 3D-CNN for endoscopic video analysis. 3D FCN needs only to feed the whole video clip into the 3D-FCN and get the probability map of the whole video clip within a forward propagation directly. So, these methods can reduce the redundant computations and accelerate detection compared to the traditional sliding window approach.

Zhang et al. [3] presented another novel hybrid classification method for automatic polyp segmentation in colonoscopy video frames. The segmentation method is composed of two main

stages that are, the region proposal stage using the FCN and region refinement stage using texton-based patch representation followed by a random forest (RF) classifier. FCN provides initial polyp candidates and texton based patch representation which further discriminate polyp from non-polyp regions. Datadriven and hand-designed features are taken for segmentation. However, some false positives may present due to the lack of spatial regularization for FCN. Here FCN-8's was trained with two classes for polyp and non-polyps(background) given by the ground truth images. FCN8's are trained using MatConvNet which is commonly used in a deep learning framework.

Bardi et.al. [4] address colon polyp detection using a convolutional neural network (CNN) combined with autoencoder, but there is no image processing applied here. The tensor flow library is used for training the convolutional encoder-decoder model. In the encoder part, here used three similar modules, each consisting of a convolution layer with stride 2 and a non-overlapping max pool with kernel 2. In the decoder section, each layer in the encoder contains its counter-part. The network dimension is equal to the input dimension.

Xiao et al. [5] attempted to use the existing deep neural network called Deep Lab-V3 to detect polyps in colonoscopy images and for the semantic segmentation of polyps and to transmit it effectively, a long short-term memory is combined with Deep Lab-V3 to augment the signal of the location of the polyp. DeepLab_V3 is used to learn and extract features of polyps.

Urban et al. [6] designed and trained a deep CNN to detect polyps using a diverse and representative set of handlabeled images from screening colonoscopies collected from more than 2000 patients. They trained different CNN architectures in this study. All trained CNN consists of the same fundamental building blocks, including convolutional layer, fully connected layer, maximum or average pooling, nonlinear activation function, batch normalization operations, and skip connections. Here each layer is followed by a rectified linear (ReLu) activation function. The last hidden layer is connected to the output unit and optimized the loss with linear output units for localization problems. Softmax output units and optimized kull back-Leibler divergence are used for classification. Localization is implemented by predicting the size and location of a bounding box that tightly enclosed any identified polyps. This allowed building CNNs that could operate in real-time.

Shin et al. [7] applied a region-based object detection scheme for polyp detection. Here adopted the region proposal network (RPN) which was introduced in a faster R-CNN method [24] to obtain a polyp candidate region in polyp frames. Then applied a proper augmentation strategy such as rotating, scaling, shearing, blurring, and brightening. Then apply two post-learning schemes: false-positive learning and offline learning. In the FP learning scheme, post-training the detector system with automatically selected negative detection outputs (FP's) which are detected from normal colonoscopy videos. This is effective to reduce many of the polyp-like false positives.

Kang et al. [8] employed a Mask R-CNN network to identify and segment polyps. Mask R-CNN in this model consists of different backbone structures that are ResNet50 and ResNet101. Then use an ensemble method to combine the output of two Mask R-CNN networks. The bitwise combination is used as the ensemble method.

Zheng et al [9] proposed an algorithm for automatic polyp detection and localization in colonoscopy video. An efficient on-the-fly trained CNN has been deployed. To overcome tracking failure caused by motion effects, here also use object detection or segmentation network such as U-Net. It utilizes optical flow to track polyps and fuse temporal information. A CNN model is first trained to detect and segment polyp in each video frame. Once a polyp is detected, the center of the polyp is computed and traced through the following frames until stopping criteria are met. During tracing, optical flow is utilized to trace easier cases and CNN is used to process harder ones. If a frame doesn't contain any polyp center seed, the frame will be regarded as a negative frame. If there are multiple polyp seeds in a frame, a spatial voting algorithm is run and the most confident center is kept as the detection while others are eliminated.

Tashk et al. [10] proposed a network, which has a novel UNet architecture. This paper adopted a novel approach for fully automatic polyp detection. This includes three main steps: first, a preprocessing step is applied to the dataset images. The preprocess comprises 3 distinct color transformations known as La*b*, CMYK, and gray-level. In the second step, the U- Net is proposed for segmentation and the final step is post-processing for improving the pixel-wise classification outcomes.

Sun et al. [11] design a U-Net with dilation convolution, which is a novel end to end deep learning framework for the colorectal polyp segmentation. The model consists of an encoder to extract multi-scale semantic features of polyps and a decoder to expand the feature map to a polyp segmentation map. The dilated convolution is added to the encoder part of the network to learn high-level semantic features without resolution reduction which improves feature representation ability. The architecture of the model consists of an end-toend convolutional neural network which includes a construction part on the left and an expensive part on the right. The model takes a single colonoscopy image as the input and outputs a binary mask segmentation of polyps that has the same size as the input image on the last layer. To improve display effectiveness during colonoscopy, several post-processing operations are applied, such as smoothening, drop small objects, and combine nearby objects.

Feng et al. [12] develop a stair-shape network (SSN) for real-time polyp segmentation in colonoscopy images. The SSN can well balance the inference time and segmentation accuracy. The lightweight backbone with four specific residual blocks and simplified upsampled operation allows fast inference time. For the backbone network, designed an FCN to extract diverse features on different levels. Besides, some intestinal folds in colonoscopy images are likely to be taken mistakenly as polyps. To address these issues, a specific dual attention module is applied to refine the output feature of each residual block in the backbone. Then designed a multiscale fusion module (MFM) for fully fusing features of different layers.

Jia et al. [13] introduced a two-stage approach called ‘’polyp for automated pixel-accurate polyp recognition in colonoscopy images” (PLP-Net) for automated polyp recognition in colonoscopy images, using deep convolutional neural network. The PLP-Net improves the performance of polyp segmentation by using a two-stage learning strategy. The PLP- Net comprises two stages, that are the polyp proposal stage and the polyp segmentation stage. The learning process would be complicated by the complex colonic wall if pixel-wise training is performed directly on the CNN model. Therefore,

a two-stage framework is proposed, where the polyp proposal stage is constructed as a region-level polyp detector, aiming to accurately segment the area of the polyp that occupies in the image.

Tan et al. [14] proposed a three-dimensional GLCM based CNN for 3D polyp classification. This proposed model contains three steps. The first step is to convert the original Hounsfield unit CT value of the 3D polyp into gray-level value based on CT value. Here performs a gray level scaling on the original CT image pixel values to an appropriate value range.

One of the challenges in training the polyp segmentation model is the insufficient number of data for training. Obtaining a large number of polyp images with the corresponding ground truth of a polyp mask is generally quite difficult because access data is limited due to privacy concerns. Moving camera control and other color-setting are available but, they are not consistent. So, the appearance of available endoscopy images that we get from different laboratories will be different. The data augmentation steps bring endoscopy images into an extended space that can cover all their variances. Moreover, by augmenting the training data, can reduce the problem of overfitting. Table 1 shows a summary of the discussed approaches.

Table 1. Summary of discussed models

Authors	Methods	Datasets	Quantitative measures (%)	Limits
Yu et al [2]	Novel offline and online 3D deep learning integration framework	ASU-Mayo	Prec=78.7; Rec=53.8; F1=63.9; F2= 98.7	High processing time
Zhang et al. [3]	FCN with novel hybrid classification	CVC-ColonDB	Acc=97.54; sens=75.66; spec=98.81	High processing time
Bardi et al. [4]	Convolutional encoder-decoder model	CVC-ColonDB, CVC-ClinicDB, ETIS-LARIB	Acc=96.7	No promising result
Xia et al [5]	Deep lab and LSTM network	CVC-ClinicDB	mIoU=93.21	High training time and prone to overfitting
Urban et al [6]	Deep CNN with Resnet50	proprietary	Acc=99.0; sens=96.8	Missed polyps
Shin et al [7]	Region-based deep CNN and post-learning	CVC-ClinicDB, ETIS-LARIB	Prec=91.4; Rec=71.2; F1=80; F2= 74.5	High processing time
Kang et al [8]	Instance segmentation using Mask R-CNN	CVC-ColonDB, CVC-ClinicDB, ETIS-LARIB	Prec=73.84; Rec=74.37	Limited segmentation
Zheng et al. [9]	Optical flow with an on-the-fly trained CNN	CVC-videoClinicDB	Prec=84.58; Rec=97.29; F1=90.49; F2=94.45	No balance between FP and FN
Tashk et al [10]	U-Net and morphological post-process	CVC-ColonDB, CVC-ClinicDB, ETIS-LARIB	Acc=99.02; Prec=70.2; Rec=82.7; F1=76.0;	High training time
Sun et al [11]	U-Net with dilation convolution	CVC-ColonDB, CVC-ClinicDB, ETIS-LARIB	Prec=96.71; Rec=95.51; F1=96.11	High training time
Feng et al [12]	Novel stair shape network (SSN)	CVC-ColonDB, CVC-ClinicDB, Endo Scene	Prec=92.85; Rec=94.83	Not completely reduce miss polyp rates
Jia et al [13]	PLP-Net with two-stage pyramidal feature prediction	CVC-612 test set, ETIS-LARIB	Prec=85.9; Rec=59.4	High computational cost
Tan et al [14]	3D GLCM based CNN model	proprietary	Acc=91; sens=90; spec=71	Model is complex

Here, I provided a comprehensive review of some recent works for the detection and segmentation of colon polyps. Even now, the polyp detection and segmentation from colonoscopy images is a challenging task in the medical field. Among the various techniques available, deep learning has shown an efficient performance. From the table, we can infer that each model discussed above has its merits and demerits. All of the models have achieved impressive performance in various image segmentation tasks also. From this we can understand that U-Net is mostly frequently used based on deep learning models, because the method was developed specifically for medical image data and does not require many annotated images.

CHAPTER 3

THEORY, DESIGN AND IMPLEMENTATION

Here, I am modelling two architectures :

1) Simple UNet Architecture
2) Modified UNet Architecture

1) Simple UNet Architecture

The **UNet** is a fully convolutional neural network that was developed by **Olaf Ronneberger** at the Computer Science Department of the University of Freiburg, Germany. In normal fully convolution network shown in Fig 3.1, the output of the fully convolution layer is upsampled to the same resolution of the image and tried to use this as a segmentation mask. But, we compressed the image so much and undergone so much processing also, maxpool has thrown away some information. So just upsampling will not give us fine resolution that we want. It will be like converting a 20x20 image to 512x512 image. Hence, we need a different approach to get the output accurate.

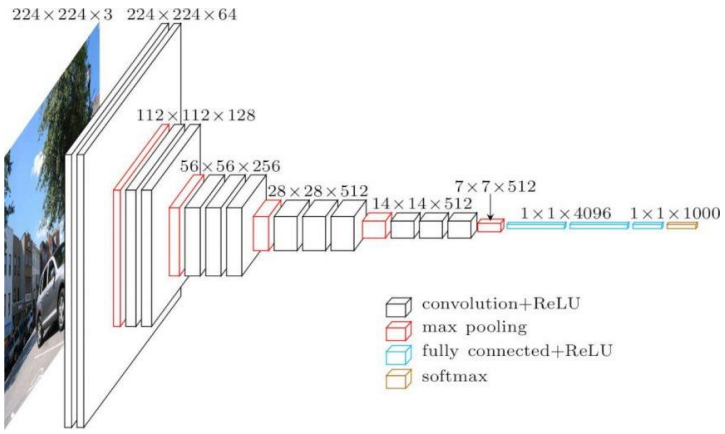

Fig 3.1 Fully Convolution network [15]

While, UNet is a convolutional neural network architecture that expanded with few changes in the CNN architecture. It was invented to deal with biomedical images where the target is not only to classify whether there is an infection or not but also to identify the area of infection. The architecture of UNet is illustrated in Fig 3.2:

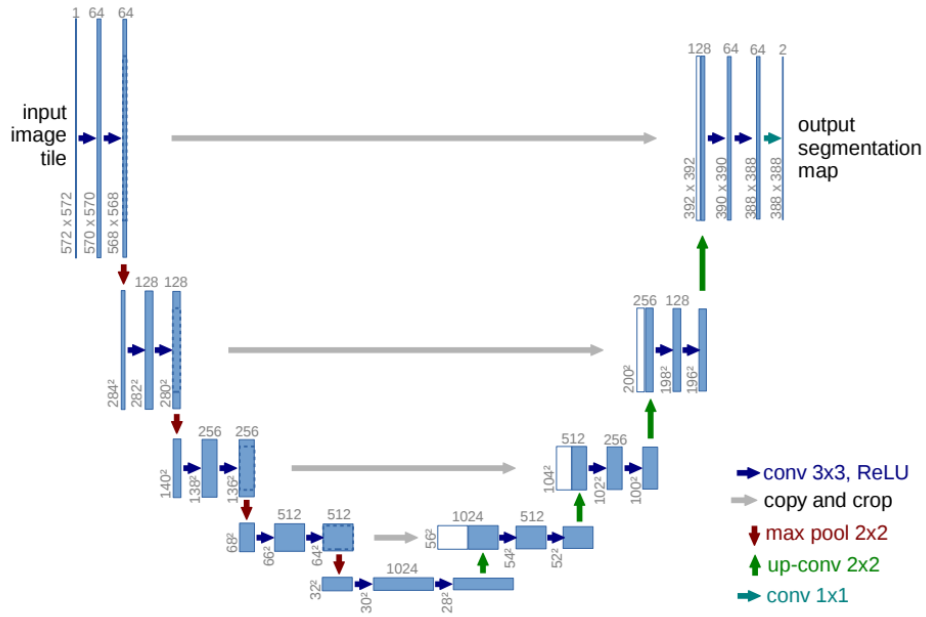

Fig 3.2 UNet Architecture [17]

It consists of an Encoder and a decoder.

1) Encoder

It is usually referred as contracting path and it follows the typical architecture of a convolutional network [18]. It consists of two 3x3 convolutions (unpadded convolutions), each followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation with stride 2 for downsampling.

This path is used to get more features of the image by doubling the number of filters in each stage. While moving into next stage we will do 2x2 max-pooling to get the maximum pixel value, thus loosing some features, but retaining the maximum pixel value. So, at the last layer of Down-sampling we are getting the lower level features of an image. Inshort, from the initial layers we get the small edges and small differences. But less features. If you go on and on deeper into the layers, more features will be there and we would get an approximate sense of where the polyp is. But inorder to exactly distinguish between the boundaries, it need the semantic information it captured earlier. If we fuse these two together, we are able to look at small semantic information of the polyps in the context of these global features. Then, we can determine the polyp in a better way. This is done by the decoder.

2) Decoder

This path consists of an upsampling of the feature map followed by a 2x2 convolution. This will make the number of feature channels half. Next, a concatenation is performed with the correspondingly cropped feature map from the contracting path. Decoder also consist of two 3x3 convolutions, each followed by a ReLU. At the final layer a 1x1 convolution is used followed by a sigmoid function thus, generating the desired mask.

The purpose of this expanding path is to enable precise localization combined with contextual information from the contracting path (encoder) [18]. Now in down-sampling we have got the pixel feature values for all the classes. Here, we concatenate the upsampled image with the output of each stage of downsampling. In up-sampling we are getting back the full image by copying the feature map

of a level having same filters of down-sampling to the level same filters of Up-sampling thus retaining the features. Thus, we get back the full image and can localize where the defect present is in the image for each class. This is known as Transpose convolution. Then again, we are learning the full-size image by applying convolution. So, in up-sampling the basically every feature layer of down-sampling side is added to the corresponding feature layer in up-sampling side so as to get the full resolution image, thus locating the class.

2) Modified UNet Architecture

Here, one thing to be noticed is that the encoder used in the UNet architecture is similar to that of fully CNN. So, transfer learning can be used and the encoder can be replaced by a pre-trained network such as VGG16 , VGG19 etc. These pre-trained networks are already trained on the ImageNet dataset and have the necessary feature extraction capabilities. ResNet50 is one of the commonly used architecture for any transfer learning task. It consist of two 3 × 3 convolutional layers as shown in Fig 3.3.

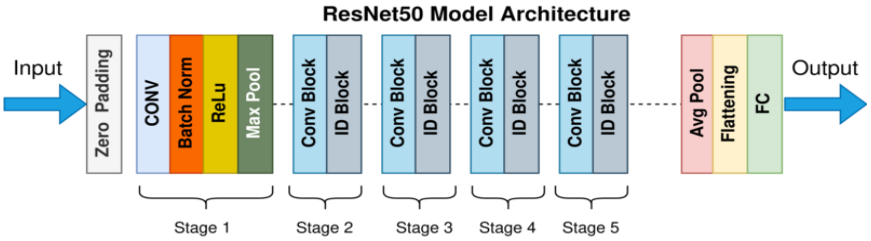

Fig 3.3 ResNet50 Architecture [19]

The modified UNet architecture obtained when we replace encoder of UNet with pretrained ReseNet50 is illustrated in Fig 3.4. The only difference in this model from the previous model is in the left side of the image.

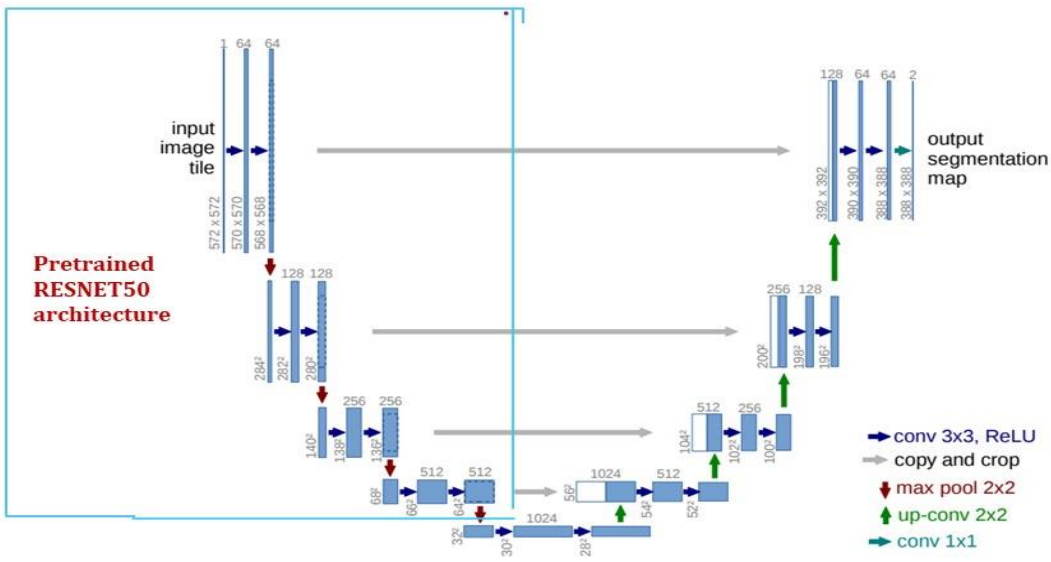

Fig 3.4 Modified UNet with ResNet 50 as encoder

ResNet50 used in the project is trained on ImageNet dataset. The use of a pre-trained encoder helps the model to converge easily. The input image is fed into the pre-trained ResNet50 encoder, consisting of a series of residual blocks as their main component. These residual blocks help the encoder extract the important features from the input image, which are then passed to the decoder. The decoder starts a transpose convolution that upscales the incoming feature maps into the desired shape. Next, these upscaled feature maps are concatenated with the specific shape feature maps from the pre-trained encoder via skip connections. These skip connections help the model to get all the low-level semantic information from the encoder, which allows the decoder to generate the desired feature maps. After that, it is followed by the two 3×3 convolution layer, where each layer is followed by a batch normalization layer and a ReLU non-linearity. The last decoder block's output is passed to a 1×1 convolution layer, which is further passed to a sigmoid activation function, finally generating the desired binary mask.

CHAPTER 4

MODEL IMPLEMENTATION AND RESULTS

With the use of an automatic, accurate, and fast polyp segmentation method during the colonoscopy, many colorectal polyps can be easily detected and removed. The "Medico automatic polyp segmentation challenge" provides an opportunity to study polyp segmentation and build an efficient and accurate segmentation algorithm. I am going to get the dataset from **CVC-ClinicDB**. It is the official database to be used in the training stages of MICCAI 2015 Sub-Challenge on Automatic Polyp Detection Challenge in Colonoscopy Videos . Link for getting the dataset is as follows:

https://polyp.grand-challenge.org/CVCClinicDB/

It contains two folders inside it namely: 'Original' and 'Ground truth'

1) Images from folder 'Original' are property of Hospital Clinic, Barcelona, Spain. It contains the original images of different types of polyps.
2) Images from folder 'Ground Truth' are propery of Computer Vision Center, Barcelona, Spain. It contains the masks of each of the original images.

Dataset information is given in Table 2.

Table 2 Dataset Information

	Images	**Masks**
Total number	612	612
Color	RGB	Greyscale
Size	256x256	256x256
Test set	61	61
Validation set	61	61
Training set	490	490

Two methods are implemented here:

1) Simple U-Net architecture
2) U-Net with pre-trained ResNet50 as the encoder

1) Simple U-Net architecture

The steps using which this model is built in showed in Fig 4.1. It consist of 4 steps. First, the dataset is loaded. Then, the model is developed. Here, the developed model is simple UNet. Next, the model is trained. Here, I selected a batch of 8 and learning rate of 10e-4. The model is trained for 20 epoch and Adam optimiser is selected. Last, the developed model is tested on unseed data to evaluate the accuracy of the model.

Step 1: Data	1)Load dataset, split into training, validation, test set 2) Read the image and the mask 3) Setting up tf.data pipeline for the training
Step 2: Model developing	1) Building encoder - Filter size is taken as 16, 32, 48, 64 Bridge Decoder Output layer
Step 3: Training of the model	1) Loading the training and validation dataset 2) Setting up tf.data pipeline for training and validation dataset 3) Setting up hyperparameters Batch =8 Learning rate = 10^{-4} Epochs = 20 4) Building U-Net model 5) Defining loss, optimiser, and metrics Loss: binary crossentropy Optimizer: Adam 6) Defining callbacks 7) Start the training
Step 4: Testing of the model	1) Loading the testing dataset 2) Setting up the tf.data pipeline for testing dataset 3) Loading the training U-Net model 4) Make the prediction and save the result

Fig 4.1 Steps of model implementation

The model parameters obtained after Step 2 by building the model is shown in Fig 4.2 and Fig 4.3.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 256, 256, 3)]	0
conv2d (Conv2D)	(None, 256, 256, 16)	448
batch_normalization (BatchNormalization)	(None, 256, 256, 16)	64
activation (Activation)	(None, 256, 256, 16)	0
conv2d_1 (Conv2D)	(None, 256, 256, 16)	2320
batch_normalization_1 (BatchNormalization)	(None, 256, 256, 16)	64
activation_1 (Activation)	(None, 256, 256, 16)	0
max_pooling2d (MaxPooling2D)	(None, 128, 128, 16)	0
conv2d_2 (Conv2D)	(None, 128, 128, 32)	4640
batch_normalization_2 (BatchNormalization)	(None, 128, 128, 32)	128
activation_2 (Activation)	(None, 128, 128, 32)	0
conv2d_3 (Conv2D)	(None, 128, 128, 32)	9248
batch_normalization_3 (BatchNormalization)	(None, 128, 128, 32)	128
activation_3 (Activation)	(None, 128, 128, 32)	0

Fig 4.2 Layers and parameters

```
otal params: 414,401
rainable params: 412,865
on-trainable params: 1,536
```

Fig 4.3 Total number of parameters

From these figure, we can observe that, out of the total 414,401 parameters, 412.865 trainable parameters and 1,536 non trainable parameters are present.

Output obtained after training the model in Step 3 is illustrated in Fig 4.4 (Step 3):

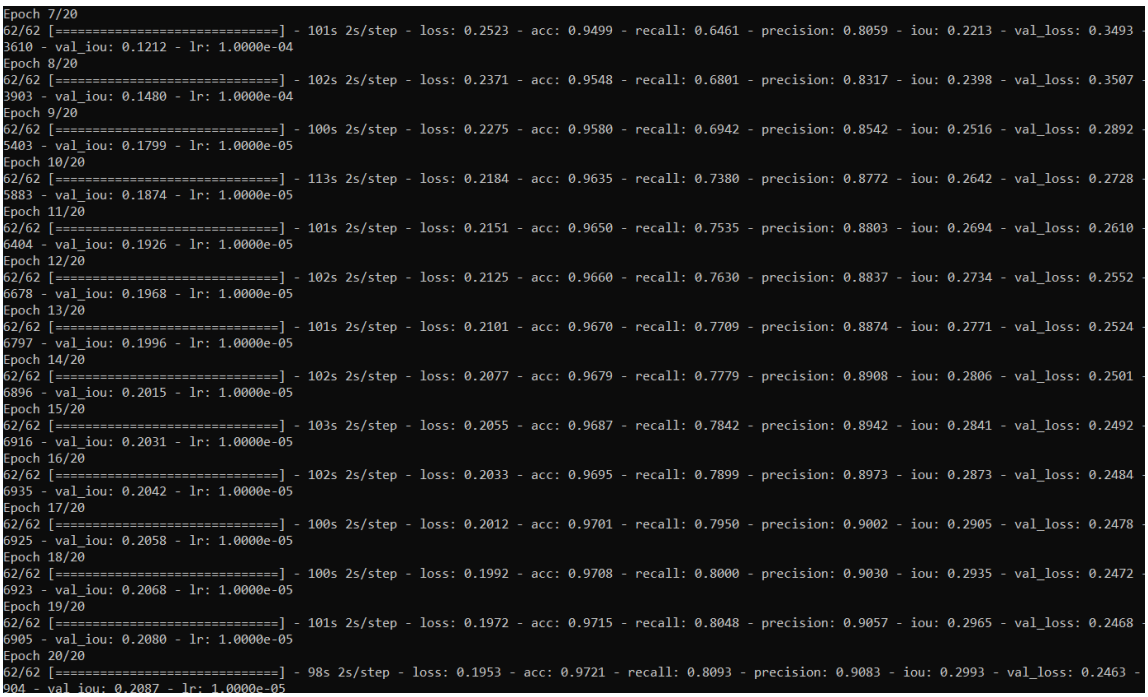

Fig 4.4 Output of Training stage

IoU matrics is used to measure the results. After 20 epochs, accuracy is obtained as 0.9721 and validation loss is 0.2463. Accuracy has improved from 0.9548 in first epoch to 0.9721. Validation loss decreased from 0.3507 to 0.2463.

The output obtained by testing the model on unseen data in Step 4 is shown in Fig 4.5 (Step 4):

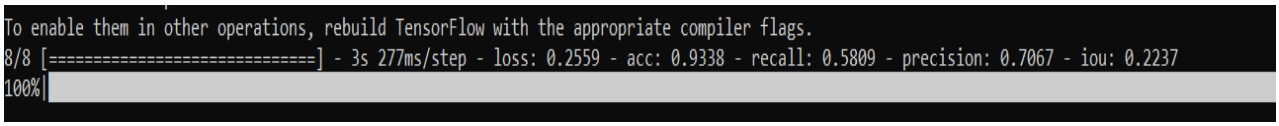

Fig 4.5 Output of Testing stage

The testing dataset is showing accuracy of 0.9338 which is less that that of the training data. The output images get automatically saved in the folder called results. Few samples of output images obtained are shown in Fig 4.6. Here, polyp image, given mask and the generated mask are displayed in the mentioned order. By observing these images, we can infer that few masks obtained are similar to that of the given ground truth while few are not. Here, I have selected the images from the results folder in such a way that both accurate and non accurate mask are visible. For example, from the shown set of images, for 3 sets of images the generates mask is similar to that of the given ground truth. While for other three the generated masks are not so accurate. The result of all the generated mask are available in the results folder.

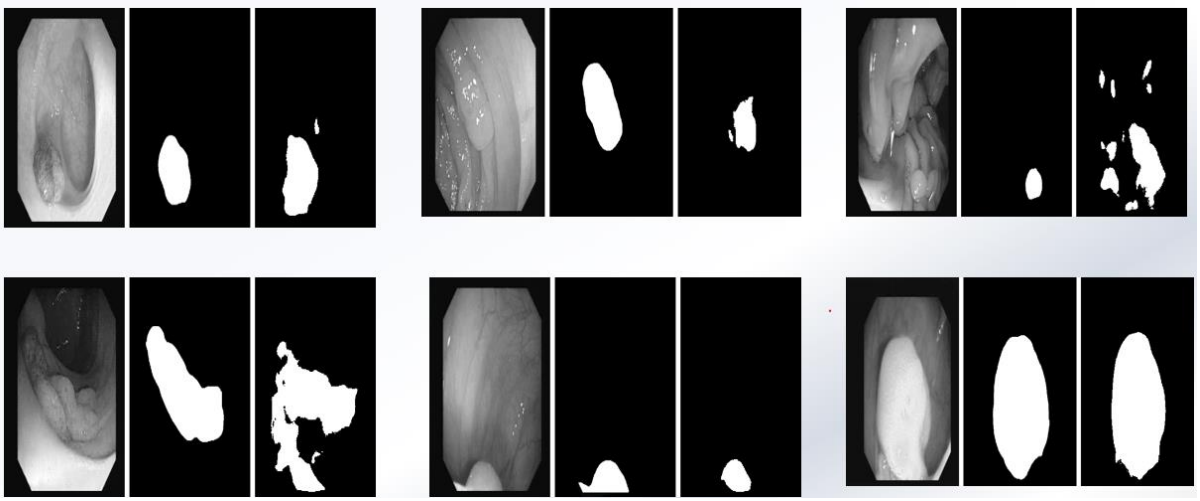

Fig 4.6 Output of Simple UNet architecture

2) U-Net with pre-trained ResNet50 as the encoder

The steps using which this model is built in showed in Fig 4.7. It consist of 4 steps. First, the dataset is loaded. Then, the model is developed. Here, the developed model is simple UNet. Next, the model is trained. Here also,, I selected a batch of 8 and learning rate of 10e-4. The model is trained for 20 epoch and Adam optimiser is selected. Last, the developed model is tested on unseed data to evaluate the accuracy of the model. The only difference in this model is in step 2. All other steps are same as that of the previous model.

Step 1: Data	**1)Load dataset, split into training, validation, test set** **2) Read the image and the mask** **3) Setting up tf.data pipeline for the training**
Step 2: Model developing	**1) Called the Predefined ResNet50 function and build encoder using that built Bridge** **2)Built Decoder** **3) Output layer**
Step 3: Training of the model	**1) Loading the training and validation dataset** **2) Setting up tf.data pipeline for training and validation dataset** **3) Setting up hyperparameters** **Batch =8** **Learning rate = 10^{-4}** **Epochs = 20** **4) Building U-Net model** **5) Defining loss, optimiser, and metrics** **Loss: binary crossentropy** **Optimizer: Adam** **1) Defining callbacks** **2) Start the training**
Step 4: Testing of the model	**1) Loading the testing dataset, Setting up the tf.data pipeline for testing dataset** **2) Loading the training U-Net model, Make the prediction and save the result**

Fig 4.7 Steps of model implementation

The model parameters obtained by building the model after Step 2 is shown in Fig 4.8 and Fig 4.9.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 256, 256, 3)]	0
conv1_pad (ZeroPadding2D)	(None, 262, 262, 3)	0
conv1_conv (Conv2D)	(None, 128, 128, 64)	9472
conv1_bn (BatchNormalization)	(None, 128, 128, 64)	256
conv1_relu (Activation)	(None, 128, 128, 64)	0
pool1_pad (ZeroPadding2D)	(None, 130, 130, 64)	0
pool1_pool (MaxPooling2D)	(None, 64, 64, 64)	0
conv2_block1_1_conv (Conv2D)	(None, 64, 64, 64)	4160
conv2_block1_1_bn (BatchNormalization)	(None, 64, 64, 64)	256
conv2_block1_1_relu (Activation)	(None, 64, 64, 64)	0
conv2_block1_2_conv (Conv2D)	(None, 64, 64, 64)	36928
conv2_block1_2_bn (BatchNormalization)	(None, 64, 64, 64)	256

Fig 4.8 Layers and parameters

```
Total params: 12,894,113
Trainable params: 12,861,601
Non-trainable params: 32,512
```

Fig 4.9 Total number of parameters

Output obtained after training the model in Step 3 illustrated in Fig 4.10 (Step 3):

```
38/62 [=================>............] - ETA: 1:32 - loss: 0.0600 - acc: 0.9907 - recall: 0.9450 - precision: 0.9821 - i
39/62 [=================>............] - ETA: 1:28 - loss: 0.0600 - acc: 0.9907 - recall: 0.9449 - precision: 0.9823 - i
40/62 [==================>...........] - ETA: 1:24 - loss: 0.0600 - acc: 0.9907 - recall: 0.9444 - precision: 0.9828 - i
41/62 [==================>...........] - ETA: 1:20 - loss: 0.0600 - acc: 0.9907 - recall: 0.9437 - precision: 0.9830 - i
42/62 [==================>...........] - ETA: 1:16 - loss: 0.0599 - acc: 0.9908 - recall: 0.9437 - precision: 0.9835 - i
43/62 [==================>...........] - ETA: 1:13 - loss: 0.0599 - acc: 0.9909 - recall: 0.9435 - precision: 0.9836 - i
44/62 [====================>.........] - ETA: 1:09 - loss: 0.0599 - acc: 0.9909 - recall: 0.9435 - precision: 0.9838 - i
45/62 [====================>.........] - ETA: 1:05 - loss: 0.0598 - acc: 0.9907 - recall: 0.9419 - precision: 0.9842 - i
46/62 [====================>.........] - ETA: 1:01 - loss: 0.0599 - acc: 0.9908 - recall: 0.9421 - precision: 0.9841 - i
47/62 [====================>.........] - ETA: 57s - loss: 0.0599 - acc: 0.9908 - recall: 0.9424 - precision: 0.9839 - io
48/62 [=====================>........] - ETA: 53s - loss: 0.0600 - acc: 0.9909 - recall: 0.9427 - precision: 0.9836 - io
49/62 [=====================>........] - ETA: 49s - loss: 0.0598 - acc: 0.9909 - recall: 0.9436 - precision: 0.9839 - io
50/62 [======================>.......] - ETA: 46s - loss: 0.0597 - acc: 0.9909 - recall: 0.9443 - precision: 0.9837 - io
51/62 [======================>.......] - ETA: 42s - loss: 0.0597 - acc: 0.9909 - recall: 0.9444 - precision: 0.9835 - io
52/62 [=======================>......] - ETA: 38s - loss: 0.0597 - acc: 0.9910 - recall: 0.9450 - precision: 0.9834 - io
53/62 [=======================>......] - ETA: 34s - loss: 0.0596 - acc: 0.9909 - recall: 0.9454 - precision: 0.9834 - io
54/62 [========================>.....] - ETA: 30s - loss: 0.0596 - acc: 0.9909 - recall: 0.9453 - precision: 0.9833 - io
55/62 [========================>.....] - ETA: 26s - loss: 0.0594 - acc: 0.9909 - recall: 0.9462 - precision: 0.9834 - io
56/62 [=========================>....] - ETA: 23s - loss: 0.0593 - acc: 0.9909 - recall: 0.9464 - precision: 0.9834 - io
57/62 [=========================>....] - ETA: 19s - loss: 0.0593 - acc: 0.9910 - recall: 0.9467 - precision: 0.9834 - io
58/62 [==========================>...] - ETA: 15s - loss: 0.0594 - acc: 0.9910 - recall: 0.9469 - precision: 0.9832 - io
59/62 [==========================>...] - ETA: 11s - loss: 0.0593 - acc: 0.9910 - recall: 0.9469 - precision: 0.9833 - io
60/62 [===========================>..] - ETA: 7s - loss: 0.0593 - acc: 0.9911 - recall: 0.9470 - precision: 0.9834 - iou
61/62 [===========================>..] - ETA: 3s - loss: 0.0593 - acc: 0.9911 - recall: 0.9470 - precision: 0.9835 - iou
62/62 [============================] - ETA: 0s - loss: 0.0593 - acc: 0.9912 - recall: 0.9470 - precision: 0.9835 - iou
62/62 [============================] - 243s 4s/step - loss: 0.0593 - acc: 0.9912 - recall: 0.9470 - precision: 0.9835
- iou: 0.6038 - val_loss: 0.1015 - val_acc: 0.9777 - val_recall: 0.8054 - val_precision: 0.9508 - val_iou: 0.4756 - lr:
1.0000e-04
```

Fig 4.10 Output of Training stage

IoU matrics is used to measure the results. After 20 epoch, Accuracy is 0.9912 and validation loss is 0.2629. Accuracy has improved from 0.8179 in first epoch to 0.9912. Validation loss decreased to 0.1015.

The output obtained by testing the model on unseen data after Step 4 is shown in Fig 4.11 (Step 4):

```
1/8 [==>............................] - ETA: 12s - loss: 0.1186 - acc: 0.9688 - recall: 0.7097 - precision: 0.9101 - iou:
2/8 [======>.......................] - ETA: 5s - loss: 0.1585 - acc: 0.9498 - recall: 0.6383 - precision: 0.9497 - iou:
3/8 [==========>...................] - ETA: 4s - loss: 0.1428 - acc: 0.9582 - recall: 0.7080 - precision: 0.9334 - iou:
4/8 [==============>...............] - ETA: 3s - loss: 0.1289 - acc: 0.9647 - recall: 0.7181 - precision: 0.9387 - iou:
5/8 [==================>...........] - ETA: 2s - loss: 0.1185 - acc: 0.9693 - recall: 0.7357 - precision: 0.9424 - iou:
6/8 [======================>.......] - ETA: 1s - loss: 0.1277 - acc: 0.9665 - recall: 0.7248 - precision: 0.9450 - iou:
7/8 [==========================>....] - ETA: 0s - loss: 0.1202 - acc: 0.9694 - recall: 0.7484 - precision: 0.9516 - iou:
8/8 [==============================] - ETA: 0s - loss: 0.1180 - acc: 0.9705 - recall: 0.7537 - precision: 0.9504 - iou:
8/8 [==============================] - 8s 836ms/step - loss: 0.1180 - acc: 0.9705 - recall: 0.7537 - precision: 0.9504 -
 iou: 0.4822
100%|                                                                                | 61/61 [00:14<00:00,  4.26it/s]
```

Fig 4.11 Output of Testing stage

The testing dataset is showing accuracy of 0.9705 which is less that that of the training data. The output images get automatically saved in the folder called results. Few samples of output images obtained are shown in Fig 4.12. Here, polyp image, given mask and the generated mask are displayed in the mentioned order. By observing these images, we can infer that few masks obtained are similar to that of the given ground truth while few are not. For example, in the last set of image the generated mask is similar to the given ground truth but, there are some unwated regions got displayes along with it. The result of all the generated mask are available in the results folder.

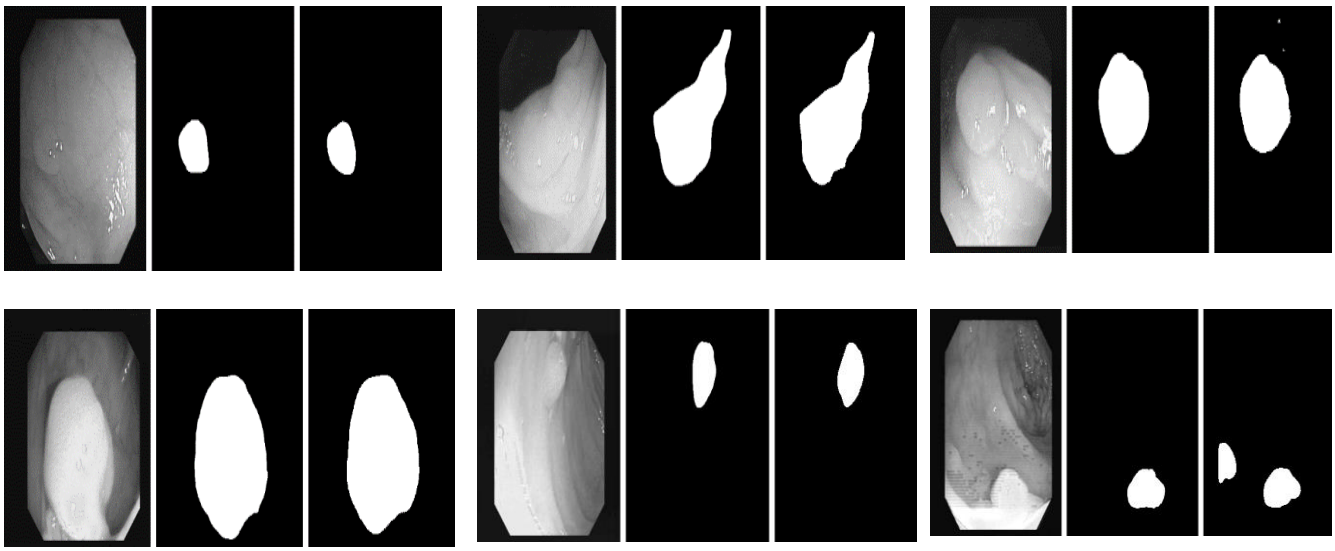

Fig 4.12 Output of Modified UNet architecture

COMPARISON OF OUTPUT OF BOTH THE METHODS

Comparison of the output by both the models are shown in Fig 4.13.

ACCURACY COMPARISON – Training data

Epoch	U-Net	Resnet50- UNet
10	0.9542	0.9893
20	0.9721	0.9912

ACCURACY COMPARISON – Testing data

Epoch	U-Net	Resnet50- UNet
10	0.9162	0.9078
20	0.9338	0.9705

Fig 4.13 Accuracy comparison

From the accuracy obtained by IoU matrics, we can infer that, when the number of epoch is increased the accuracy is getting better.In the second model where we use pretrained ResNet50 as encoder the accuracy is more and it reached upto 97 percent for the test data and 99 for the training data. While, the accuracy of UNet is 93 percent for unseendata.This improvement of the accuracy in ResNet-UNet architecture is due to the transfer learning adopted.

CHAPTER 5

CONCLUSION

Polyp detection and segmentation from colonoscopy images is a challenging task in the medical field. Among the various techniques available, deep learning has shown an efficient performance. In this project, a model is developed using simple UNet archichitecture and Modified UNet architecture. The accuracy obtained from these models on unseen data are 0.9338 and 0.9705 respectively, which quite good. Moreover, through visual analysis, it is clear that the proposed methods are successful in segmenting the cancer region of the selected dataset. This method is simple and fast because it doesn't comprise of any complex structures. The resulting images gathered from this process can be helpful for physician in cancer diagnosis and monitor the treatment process. In the future, we would like to increase the dataset size to obtain more accurate and robust results.

PYTHON Files Descriptions

1) Simple UNet architecture

polypsegmentation.py

This is the file for entire polypsegmentation using simple UNet. First, I made different files for each of the steps (given below) and then merged them together as a single file.

data.py

This is the seperate file for loading the data

model.py

This is the separate file where model is developed

train.py

This is the seperate file for training the dataset

predict.py

This is the seperate file for testing the data

2) Modified UNet architecture

data2.py

This is the seperate file for loading the data

model2.py

This is the separate file where model is developed

train2.py

This is the seperate file for training the dataset

predict2.py

This is the seperate file for testing the data

REFERENCES

[1] F. Bray, A. Jemal, R.A. Smith et al, ‘’Global cancer transactions according to the human development index (2008-2030): a population-based study’’, Lancet Oncol, Vol.13, pp.790—801, 2012

[2] Yu L, H. Chen, Q. Dou, J. Qin, and P.A. Heng,” Integrating online and offline three-dimensional deep learning for automated polyp detection in colonoscopy videos," IEEE Journal of Biomedical and Health Informatics, vol.21, pp.65–75, January 2017.

[3] L. Zhang, S. Dolwani, and X. Ye, "Automated Polyp Segmentation in Colonoscopy Frames Using Fully Convolutional Neural Network and Textons,” Springer international publishing, pp.707–717,2017.

[4] O. Bardi, D.S.Sosa, B.G.Zapirain and A.Elmaghrby, "Automatic colon polyp detection using convolutional encoder-decoder model, "2017 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT),pp.445–448, 2017

[5] W. T.Xiao, L.J. Chang, and W.M. Liu, "Semantic segmentation of colorectal polyps with deep lab and LSTM networks, "2018 IEEE International Conference on Consumer Electronics-Taiwan (ICCETW),pp.1–2, 2018

[6] G. Urban, P. Tripathi, T. Alkayali, M. Mittal, F. Jalali, W. Karnes, and P. Baldi, "Deep learning localizes and identifies polyps in realtime with 96% accuracy in screening colonoscopy,” Gastroenterology, pp.1069– 1078,2018

[7] Y. Shin, H. A. Qadir, L. Aabakken, J. Bergsland, and I. Balasingham, "Automatic colon polyp detection using region-based deep CNN and post-learning approaches," IEEE Access, vol. 6, pp. 40950–40962, 20185

[8] J. Kang and J. Gwak, "Ensemble of instance segmentation models for polyp segmentation in colonoscopy images" IEEE Access, vol.7, pp.26440–26447, February 2019.

[9] H. Zheng, H. Chen, J. Huang, X. Li, X. Han and J. Yao “Polyp tracking in video coloscopy using optical flow with an on-the-fly trained CNN,” IEEE international symposium on biomedical imaging, pp.79–82, April 2019

[10] A. Tashk, J. Herp, and E. Nadimi, "Fully automatic polyp detection based on a novel U-Net architecture and morphological postprocess,” 2019 International Conference on Control, Artificial Intelligence, Robotics and Optimization (ICCAIRO), pp.37– 41,2019.

[11] X. Sun, P. Zhang, D. wang, Y. Cao, and B. Liu, "Colorectal polyp segmentation by U-Net with dilation convolution, "2019 18th IEEE International conference on machine learning and applications, pp.851–858, 2019

[12] R. Feng, B. Lei, W. Wang, T. Chen, J. Chen, D.Z. Chen and J. Wu, "SSN: A stair-shape network for real-time polyp segmentation in colonoscopy images, "IEEE 17th international symposium on biomedical imaging, pp.225–229, April 2020.

[13] X. Jia, X. mai, Y. Cui, Y. Yuan, X. Xing, H. Seo, L. Xing, and M.Q.H. Meng, "Automatic polyp recognition in colonoscopy images using deep learning and two-stage pyramidal feature prediction," IEEE Transactions on automation science and engineering, pp.1–15,2020.

[14] J. Tan, Y. Gao, Z. Liang, W. Cao, M. Pomeroy, Y. Huo, L. Li, M.A. Barish, A.F. Abbasi, and P. J. Pickhardt, "3D-GLCM CNN: A 3- dimensional gray- level co-occurrence matrix-based CNN model for polyp classification via CT colonography," Transaction on medical imaging, vol. 39, pp. 2013– 2024, June 2020.

[15] Slides by Prof Ivan Bajic

[16]https://www.analyticsvidhya.com/blog/2021/08/all-you-need-to-know-about-skip-connections/

[17] https://arxiv.org/pdf/1505.04597.pdf

[18] https://medium.com/analytics-vidhya/deep-learning-image-segmentation-and-localization-u-net-architecture-ea4cff5595d9

[19] https://www.researchgate.net/figure/The-architecture-of-ResNet-50-model_fig4_349717475

[20] Saruar Alam1 , Nikhil Kumar Tomar2 , Aarati Thakur3 , Debesh Jha2,4, Ashish Rauniyar5,6, "Automatic Polyp Segmentation using U-Net-ResNet50", MediaEval'20, December 14-15 2020, Online

[21] Zhengxin Zhang† , Qingjie Liu†∗ and Yunhong Wang, Senior, "Road Extraction by Deep Residual U-Net", IEEE GEOSCIENCE AND REMOTE SENSING LETTERS

[22] https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/

[23] https://developers.arcgis.com/python/guide/how-unet-works/