**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**

**---🙠**🕮**🙢---**



**ASIGNMENT INTRO TO DEEP LEARNING REPORT**

***Student: Đoàn Anh Vũ***

***Student’s ID: 20225465***

***Class ID: 152474***

***Năm học 2023 - 2024***

1. Problem

The NeoPolyp challenge presents a dataset of *1200 polyp* images. This dataset is divided into a training set of 1000 images and a test set of 200 images. Each *polyp* image is classified as either *neoplastic* or *non-neoplastic*, visually indicated by *red* and *green* colors, respectively.

The goal of this challenge is to develop a model that can accurately classify imaged polyps, supporting the medical prediction process, especially the early detection of polyps that are likely to transform into cancer. Accurately classifying *polyps* is of great significance in providing appropriate treatment regimens, improving patient survival rates and reducing the burden on the health system.

To achieve this goal, we need to find a way to read and process images, then give expected results about the characteristics of each *polyp*.

[BKAI-IGH NeoPolyp Challenge](https://www.kaggle.com/competitions/bkai-igh-neopolyp)

1. Transformation

The BKAI-IGH NeoPolyp Challenge dataset is divided into three subsets: Training, Testing, and Training Ground Truth. The Training set comprises 1000 polyp images. The corresponding ground truth images, located in the Training Ground Truth folder, are binary masks highlighting neoplastic and non-neoplastic regions in green and red, respectively.

To enhance color-based image analysis, we convert the images from the BGR color space to the HSV color space. This transformation is motivated by the observation that the HSV color space often aligns more closely with human color perception.

Subsequently, red and green color masks are applied to the HSV images to isolate regions of interest. These masks generate binary images where pixels corresponding to red and green regions are assigned specific values, enabling further analysis and feature extraction.

1. Model architecture

For this task, we employed a pretrained UNet++ model. UNet++ was selected over the traditional UNet due to its superior ability to capture fine-grained details in medical images. Its dense skip connections enable the model to preserve essential features across different scales, mitigating information loss during downsampling and upsampling. This enhanced feature preservation leads to more accurate and robust segmentation results, especially when dealing with complex image patterns like those found in polyp images. Additionally, UNet++ has demonstrated superior generalization performance and a better ability to handle class imbalance, making it well-suited for medical image segmentation tasks.

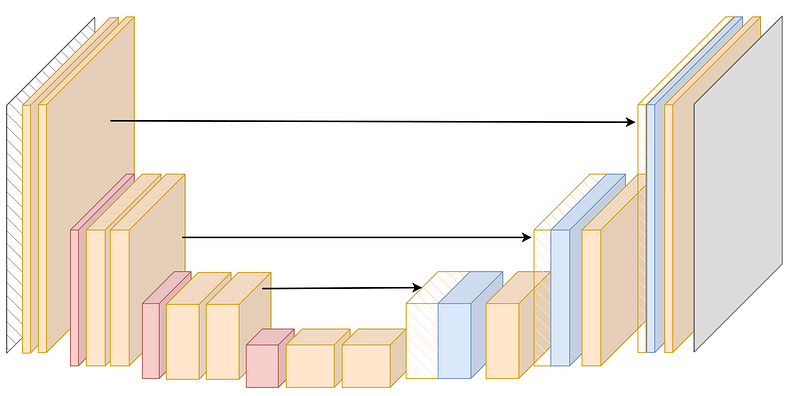


Figure 1:UNet Architecture

A diagram of a diagram

Description automatically generated

Figure 2: UNet++ Architecture

The model takes a batch of images as input, where each image is represented as a multi-dimensional tensor. This tensor typically consists of dimensions for the batch size, the number of color channels and the spatial dimensions (height and width). In this challenge, input images are medical endoscopic images, preprocessed and resized to a fixed size.

U-Net++ employs an advanced encoder-decoder architecture that leverages dense skip connections to achieve high accuracy. The encoder progressively reduces the spatial dimensions of the input image while extracting hierarchical features through convolutional layers and pooling operations. The decoder, in turn, upsamples these features and combines them with corresponding features from the encoder via dense skip connections. These connections are a significant enhancement over the original U-Net architecture, as they not only connect encoder and decoder layers but also link intermediate layers at the same resolution, reducing the semantic gap between feature maps. This enables the model to preserve fine-grained details and better localize objects in the image.

The model's output is a segmentation mask of the same spatial dimensions as the input image. For binary segmentation tasks like the Neo-Polyp Challenge, each pixel in the mask is assigned a probability value between 0 and 1, representing the likelihood of belonging to a polyp or the background. This is achieved through a sigmoid activation function in the final layer. U-Net++ excels in this challenge due to its ability to capture and propagate information across multiple scales, making it particularly effective at segmenting polyps with diverse sizes and shapes.

1. Evaluation

Initially, we started training with a constant learning rate and the *AdamOptimizer*. While the results were acceptable, they were not optimal. To refine the learning process, we experimented with different learning rates from the set {0.0001, 0.001, 0.01}. Among these, a learning rate of 0.0001 yielded the best performance.

However, upon observing the validation loss curve, we noticed periods where the best validation loss plateaued over consecutive epochs. To address this, we decided to adjust the learning rate dynamically during training.

Our first approach was to use *torch.optim.lr\_scheduler.StepLR*, which decreases the learning rate by a factor after a predefined number of steps. While this method provided some improvement, the learning rate adjustments felt somewhat arbitrary.

Subsequently, we adopted *torch.optim.lr\_scheduler.ReduceLROnPlateau*, which reduces the learning rate by a specified factor if the validation loss does not improve after a certain number of epochs. This approach proved to be the most effective among those tested, as it adaptively responded to the model's progress and significantly enhanced training stability and performance.

Three images below are plot of train loss validation loss from training process, thanks to *WandB*

A screen shot of a graph

Description automatically generated

Figure 3: Validation loss with constant learning rate 0.0001

A graph with blue lines

Description automatically generated

Figure 4: Train loss with constant learning rate 0.0001

A graph with red and white lines

Description automatically generated

Figure 5: Validation loss with learning rate change by step 4

A white paper with red text

Description automatically generated with medium confidence

Figure 6:Train loss with learning rate change by step 4

A white graph with purple lines

Description automatically generated

Figure 7: Validation loss with learning rate change plateau 5

A white graph with black text

Description automatically generated

Figure 8: Train loss with learning rate change plateau 5

1. Conclustion

After experimenting with various learning rates and different learning rate adjustment strategies, we observed that the U-Net++ model achieved the best performance when trained with an initial learning rate of 0.0001. Specifically, we found that reducing the learning rate by a factor of 10 after 5 consecutive epochs without improvement in the best validation loss yielded the most promising results.

While we aimed to explore and implement a broader range of methods for optimizing the learning rate schedule, time constraints limited the scope of our experiments. As such, we focused on the most practical and impactful approaches within the available timeframe. This iterative process allowed us to balance effectiveness and efficiency, resulting in a model training strategy that delivered competitive outcomes.

1. Github

The folder data contains initial files, which are ***train***,***train\_gh***, ***test***from[BKAI-IGH NeoPolyp/Data](https://www.kaggle.com/competitions/bkai-igh-neopolyp/data). You should download the folder on this link and *Extract* all the files.

In the Working Directory, you should put a folder name ***data*** if you want to modify and run the ***.ipynb*** type code. Inside ***data*** are three folders ***train***, ***train\_gh***, ***test***. Each folder contain images corresponding to the data of the challenge

[Best-model-checkpoint](https://drive.google.com/file/d/17_x_oimwTMf5nYlxmNfO3FyOJXV6D4kz/view?usp=sharing) is store here, you must download it and place it inside folder ***code.***

Then, you can run the file ***infer.py*** smoothly.

Here is the steps:

Open your git bash or terminal and do these successively:

* git clone <https://github.com/bluff-king/ASMNeoPolyp20241.git>
* cd ASMNeoPolyp20241
* download the [Best-model-checkpoint](https://drive.google.com/file/d/17_x_oimwTMf5nYlxmNfO3FyOJXV6D4kz/view?usp=sharing) and place it inside ***code***
* python3 infer.py --image\_path image.jpeg

1. References

d  
[1] An, Nguyen S, et al. “BlazeNeo: Blazing Fast Polyp Segmentation and Neoplasm Detection.” *IEEE Access*, vol. 10, 1 Jan. 2022, pp. 43669–43684, ieeexplore.ieee.org/document/9759450, https://doi.org/10.1109/access.2022.3168693. Accessed 23 Nov. 2024.

[2] Conor O'Sullivan. “U-Net Explained: Understanding Its Image Segmentation Architecture.” *Medium*, Towards Data Science, 8 Mar. 2023, medium.com/m/global-identity-2?redirectUrl=https%3A%2F%2Ftowardsdatascience.com%2Fu-net-explained-understanding-its-image-segmentation-architecture-56e4842e313a. Accessed 23 Nov. 2024.

[3] Duc, Nguyen Thanh, et al. “ColonFormer: An Efficient Transformer Based Method for Colon Polyp Segmentation.” *IEEE Access*, vol. 10, 2022, pp. 80575–80586, https://doi.org/10.1109/access.2022.3195241. Accessed 3 Nov. 2022.

[4] Hoang, Thuan Nguyen, et al. “RaBiT: An Efficient Transformer Using Bidirectional Feature Pyramid Network with Reverse Attention for Colon Polyp Segmentation.” *ArXiv.org*, 2023, arxiv.org/abs/2307.06420. Accessed 23 Nov. 2024.

[5] Lan, Phan Ngoc, et al. “NeoUNet : Towards Accurate Colon Polyp Segmentation and Neoplasm Detection.” Lecture Notes in Computer Science, 1 Jan. 2021, pp. 15–28, https://doi.org/10.1007/978-3-030-90436-4\_2. Accessed 23 Nov. 2024.

[6] Zhou, Zongwei, et al. *UNet++: A Nested U-Net Architecture for Medical Image Segmentation*.