Polyp segmentation using Unet++

Business Objective

Machine learning and deep learning technologies are increasing at a fast pace with respect to the domain of healthcare and medical sciences. These technologies sometimes even out perform medical doctors by producing results that might not be easily notable to a human eye. Polyp recognition and segmentation is one such technology which helps doctors identify polyps from colonoscopic images.

Data Overview

CVC-Clinic database consists of frames extracted from colonoscopy videos. The dataset contains several examples of polyp frames & corresponding ground truth for them.
The Ground Truth image consists of a mask corresponding to the region covered by the polyp in the image. The data is available in both .png and .tiff formats

Data Src : https://www.kaggle.com/balraj98/cvcclinicdb

Aim
To segment the polyps from colonoscopy images

Tech Stack

Language used : Python
Deep learning library used : Pytorch
Computer vision library used : OpenCV
Other python libraries : Scikit-learn, pandas, numpy, albumentations etc.

Approach

1. Data Understanding :
   Understanding the essence of the dataset
2. Understanding evaluation metrics:
   Understanding the metrics that are going to be used for evaluating the predictions
3. Unet Architecture :
   Understanding Unet architecture and why is it preferred widely in building deep learning models with respect to medical sciences.
4. Unet ++ :
   Understanding Unet++ and how is it different from Unet
5. Environment Setup :

Setting up a working environment for the project

6. Data Augmentation :

    Creating new data by making modifications on the existing data

7. Model building :

    Building Unet ++ model using pytorch

8. Model Training

    Training the model. (A GPU might be required since model training takes a really long time in CPUs)

9. Model Prediction

Modular code overview

```
input
  |__PNG
      |__Ground Truth
      |__Original
  |__TIF
      |__Ground Truth
      |__Original

src
  |__engine.py
  |__config.yaml
  |__ML_pipeline
              |__averagemeter.py
              |__dataset.py
              |__iou.py
              |__network.py
              |__predict.py
              |__train.py
              |__validate.py
              |__vggblock.py

lib
  |__Unet++.ipynb
output
  |__models
         |__logs
         |__model.pth
```

Once you unzip the modular_code.zip file you can find the following folders within it.

1. input
2. src
3. output
4. lib
5. requirements.txt

1. The input folder contains all the data that we have for analysis. In our case, it will contain two folders namely.

    a. PNG
    b. TIF

` The PNG folder contains the images used for training the model in .png format while the TIF folder contains the same set of images in .tiff format

2. The src folder is the heart of the project. This folder contains all the modularized code for all the above steps in a modularized manner. It further contains the following.
    a. ML_pipeline
    b. engine.py
    c. config.yaml

The ML_pipeline is a folder that contains all the functions put into different python files which are appropriately named. These python functions are then called inside the engine.py file. The config.yaml file contains various project constants that can be altered according to users needs.

3. The output folder contains the model that we trained for this data saved as a reusable file. It also contains the predicted images
4. The lib folder is a reference folder. It contains the original ipython notebook that was shown in the videos
5. Finally, the requirements.txt consists of all the packages and libraries used in the project.

Project Takeaways

1. Understanding Polyp Segmentation Problem
2. Understanding IOU
3. Understanding Data augmentation
4. Data augmentation using pytorch
5. Understanding Computer vision and its applications in medical field
6. Understanding and implementing CNN models
7. OpenCV for computer vision
8. Understanding VGG architecture
9. Understanding Unet architecture
10. Understanding Unet++ architecture
11. Building VGG block using Pytorch
12. Building Unet++ network using Pytorch
13. Training and predicting Unet++ models