Problem 2: Teeth Segmentation

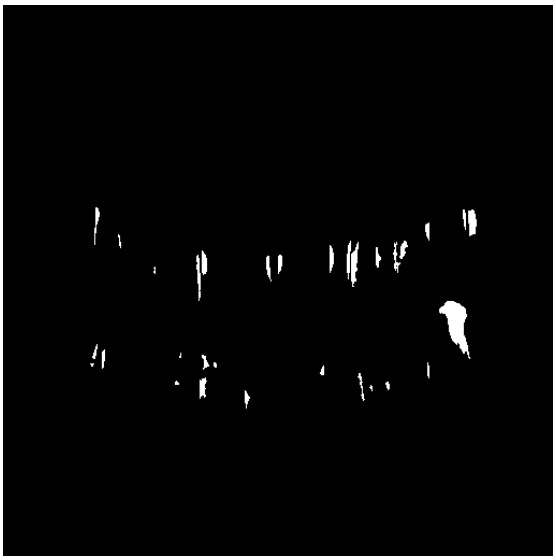
Introduction:

Image segmentation is a task in computer vision where the goal is to classify each pixel of an image into specific categories, enabling a detailed understanding of the image's structure. Unlike image classification, which assigns a single label to the whole image, segmentation provides pixel-level labels, making it essential for applications like medical imaging, autonomous driving, and object detection.

U-Net is a popular neural network architecture specifically designed for segmentation tasks. Its unique structure consists of a contracting path (encoder) to capture context and an expanding path (decoder) to enable precise localization. Originally developed for biomedical image segmentation, U-Net has become widely used in various fields due to its ability to produce high-resolution, accurate results in pixel-level predictions.

Dataset:

We will be using a dataset consisting of teeth images and their corresponding caries segmentation masks. The dataset contains approximately 700 samples, each of which is a 512x512 grayscale image paired with its mask which has the same size.



Problem Statement:

Your goal is to train a segmentation model that achieves optimal results in detecting tooth caries. The main challenge lies in optimizing the model's performance while adhering to size limitations. Using effective data augmentation techniques and selecting the right loss function are also key factors in improving the model's segmentation accuracy.
There also might be some issues within the dataset similar to data poisoning. It is in your best interest to resolve them before training your model.

Evaluation

The performance of your model will be measured using the Dice score, a widely recognized metric for segmentation tasks. This metric has been chosen for its ability to quantify the overlap between predicted and actual segmentations. To help you monitor your model's progress, the Dice score calculation has also been implemented and provided in the notebook available to you.

Submission

You are required to submit both the model weights and the code that defines the model architecture. Zip both files into a single archive. The Python file containing the model architecture should be named **model.py**, and you must include any necessary library imports in your code.

```
import zipfile
torch.save(model.state_dict(),'model.pth')
with zipfile.ZipFile('submission.zip', 'w') as zipf:
    zipf.write('model.pth')
    zipf.write('model.py')
```

Limitations:

- The model must be implemented using **torch** and **torchvision** only (no other deep learning libraries are allowed for the model architecture).
- The main class for the model must be named **Model**, and participants **must not change this name**.
- The model size must not exceed 70 MB.
- The input of the model has the following dimensions: (Batch_size x 1 x 512 x 512)
- The output has a shape of (Batch_size x 1 x 512 x 512)

Version: 1.1