***ĐOÀN NGỌC CƯỜNG 20210141***

Link to my github: <https://github.com/DoanNgocCuongBKEGNH/submit_Hw3DL_DNC210141>

Model summited: <https://www.kaggle.com/datasets/cngonngc/submit-model-pth>

***1. The transformation techniques I have used:***

```python

# Update transform thêm Augmentation: https://albumentations.ai/docs/getting\_started/mask\_augmentation/

# Nhiều hơn: https://albumentations.ai/docs/api\_reference/full\_reference/

train\_transform = A.Compose(

    [

    A.Resize(256, 256, interpolation=cv2.INTER\_LINEAR),       # kiểu numpy array,

    A.VerticalFlip(),

    A.HorizontalFlip(),  #  mặc định p=0.5

    A.OneOf([

    A.Blur(blur\_limit=3, p=0.3),   # làm mờ, độ mờ ngẫu nhiên từ 0-3

    A.GaussianBlur(blur\_limit=3, p=0.3),

    A.MedianBlur(blur\_limit=3, p=0.3),

    A.RandomBrightnessContrast()  # độ tương phản hình ảnh, mặc định p=0.2

    ], p=0.3),

    A.Rotate(limit=30, p=0.3),

    A.RGBShift (r\_shift\_limit=20, g\_shift\_limit=20, b\_shift\_limit=20, always\_apply=False, p=0.5),

    # ngẫu nhiên dịch chuyển kênh màu từ 0-20

    A.Cutout (num\_holes=5, max\_h\_size=8, max\_w\_size=8, fill\_value=0, always\_apply=False),    # mặc định p=0.5

    # cắt đi 1 khoảng model vẫn học được

    A.GaussNoise(p=0.3),

    A.Normalize (mean=(0.485, 0.456, 0.406), std=(0.229, 0.224, 0.225), max\_pixel\_value=255.0, always\_apply=False, p=1.0),

    # Mean and std of ImageNet (tập data rất lớn) là như trên phổ biến cho các model,

    ToTensorV2(),

]

)

- I used a series of augmentation, transformation techniques to create diversity in the training data and enhance the generalization ability of the model.

1. **\*\*Resize\*\*:** Resize the image to 256x256 pixels using linear interpolation method.

2. \*\***VerticalFlip\*\*:** Flip the image vertically with default probability of 0.5.

3. \*\***HorizontalFlip**\*\*: Flip the image horizontally with default probability of 0.5.

4. \*\***OneOf (Blur, GaussianBlur, MedianBlur, RandomBrightnessContrast)\*\*:** Randomly choose one of the following transformations:

- \*\*Blur\*\*: Blur the image with random blur from 0-3.

- \*\*GaussianBlur\*\*: Blurs the image using the Gaussian method with random blur from 0-3.

- \*\*MedianBlur\*\*: Blur the image using the median method with random blur from 0-3.

- \*\*RandomBrightnessContrast\*\*: Increase or decrease contrast and brightness randomly.

5. \*\***Rotate**\*\*: Rotate the image randomly between -30 and 30 degrees with probability 0.3.

6. \***\*RGBShift**\*\*: Randomly shifts color channels (red, green, blue) from 0-20 units with probability 0.5.

7. **\*\*Cutout**\*\*: Cut some holes into the image with the number of holes being 5, the maximum hole size being 8x8 pixels, and the fill value being 0, with a default probability of 0.5.

8. \*\***GaussNoise\*\*:** Adds Gaussian noise to the image with probability 0.3.

9. **\*\*Normalize\*\*:** Normalize the image according to the mean value and standard deviation of ImageNet with probability 1.0.

10. **\*\*ToTensorV2\*\*:** Convert image into tensor format.

***### 2.What I have changed in the model architecture:I have tested many Resnet models***

- Resnet-34:

- U-Net++ with ResNet-34: 83.3M parameters. Will be faster in training and inference due to lighter encoder architecture. However, you have to train for about 300 epochs to reduce the loss to 0.1, acc > 0.7 - U-Net++ with ResNet-50: 97.8M parameters

- U-Net++ with ResNet-101: 170M para. Using ResNet-101 as an encoder provides a balance between accuracy and speed compared to ResNet-152.

- U-Net++ with ResNet-152: 232M parameters. Train for about 100 epochs will reduce loss to 0.1, acc > 0.7

- This is a model with a very deep encoder, providing the ability to learn complex representations.

- Can achieve high accuracy but will be slower in training and inference.

- Requires more GPU memory resources and can be difficult to train without powerful enough hardware.

- U-Net++ with Attention ResNet-34: ...

=> Finally: I use the U-Net++ ResNet-101 và U-Net++ ResNet-152

3.

A screenshot of a computer

Description automatically generated

Plot quá trình train, val của tôi.

***3.1. Train Unet++101resnet***

In the first 30 epochs: best model at epoch 24 best\_val\_loss is 0.05103267043828964

- Epoch: 24, Train Loss: 0.03385688684673773, Train Dice: 0.8075790292686886, Train IoU: 0.7563386370076074, Train Accuracy: 0.989578111436632. at Epoch 24 Avg Validation Loss 0.05103267043828964

- Not until epoch 48 will there be the next best\_val\_loss: Epoch: 48, Train Loss: 0.01923132327902648, Train Dice: 0.8683773051367866, Train IoU: 0.8282751733726925, Train Accuracy: 0.9937682766384549. at Epoch 48 Avg Validation Loss 0.047675849981606004

- Next is Epoch: 53 best\_val\_loss is 0.046704819910228255

- Next is Epoch 82. Now Best Validation Loss, at epoch 82 best\_val\_loss is 0.04516859341412783

=> Predict: until epoch 200 still at Avg Validation Loss 0.05

***3.2. Train Unet++152resnet:***

- If Unet++101resnet takes about 5 epochs, Unet++152resnet only takes 1 epoch to have an equally good best\_val\_loss.

- However, in the following epochs, Unet++152resnet is not much better than Unet++101resnet. Forever and ever, epoch 100/300 still gives Avg Validation Loss 0.05

- Epoch 100/200: 100%|██████████| 225/225 [02:06<00:00, 1.77it/s]

Epoch: 100, Train Loss: 0.01883689166771041, Train Dice: 0.8711983256869846, Train IoU: 0.829300952355067, Train Accuracy: 0.9934579637315538

at Epoch 100 Avg Validation Loss 0.05606572521850467

- Epoch 101/200: 100%|██████████| 225/225 [02:06<00:00, 1.77it/s]

Epoch: 101, Train Loss: 0.019911806943515936, Train Dice: 0.8612699868943956, Train IoU: 0.8194689223501417, Train Accuracy: 0.9932804701063368

at Epoch 101 Avg Validation Loss 0.05327896352857351

- According to output.log, in 200 epochs:

The best epoch is: 67 best\_val\_loss is 0.04574570441618562 and up output reaches 0.74.

Link to download best model: https://wandb.ai/doanngoccuong\_nh/PolypSegment/artifacts/model/best\_model/v1/files

(Unet++152resnet\_68best/200e link: https://wandb.ai/doanngoccuong\_nh/PolypSegment/runs/0dh23g1s?workspace=user-doanngoccuong) - weighs 1 GB, the downloaded file is .pth, when on wandb it is .pardam

3.3 Img - wandb: train loss, Train dice, train accuracy, Avg Validation Loss, Best Validation Loss

and Details when training the model

=> Report at Wandb: <https://api.wandb.ai/links/doanngoccuong_nh/yx32b0wt>

A graph of a graph

Description automatically generatedA graph with green and red lines

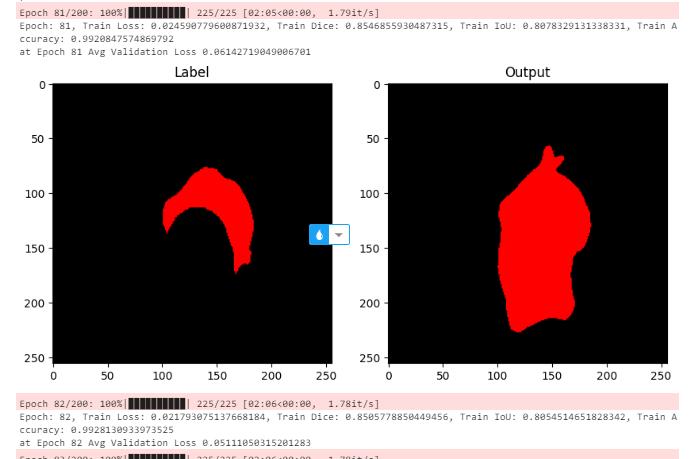
Description automatically generatedA graph of a train accuracy

Description automatically generatedA graph of a graph

Description automatically generated with medium confidenceA graph of a train loss

Description automatically generated

3.4 Chi tiết lúc train model



A screenshot of a computer screen

Description automatically generated

4. Link to my github: <https://github.com/DoanNgocCuongBKEGNH/submit_Hw3DL_DNC210141>

Model summited: <https://www.kaggle.com/datasets/cngonngc/submit-model-pth>