HistoNet

YAMAHA Hackathon 3.0

Group No.: P17-A

Problem Statement

- HistologyNet aims to revolutionize biomedical image segmentation by harnessing the power of self-supervised learning.

- **Why Self Supervised?!**
  - Machines learn from data without explicit labels, tapping into inherent patterns for training.

Our Solution:

For Self-Supervised Learning task:

1). state-of-the-art ViT(Vision Transformer) encoder - used for generating image embeddings.

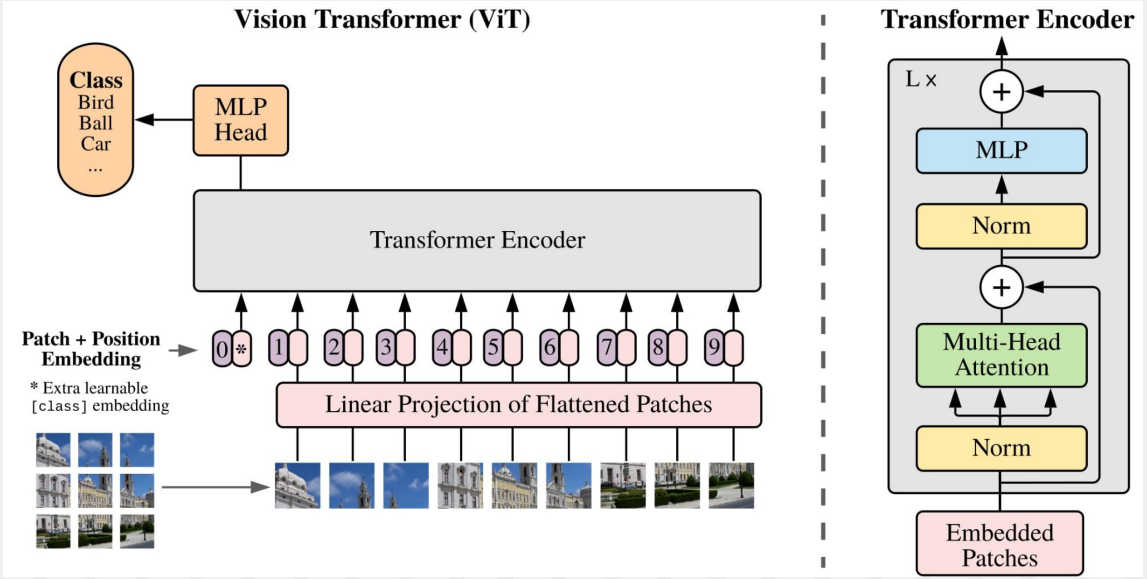
2). Decoder used for generating original image from encoder's embedding.

3). ViT encoder and Decoder from SAM is used for further fine tuning on labelled data.

4). Data preprocessing steps for further improvements.

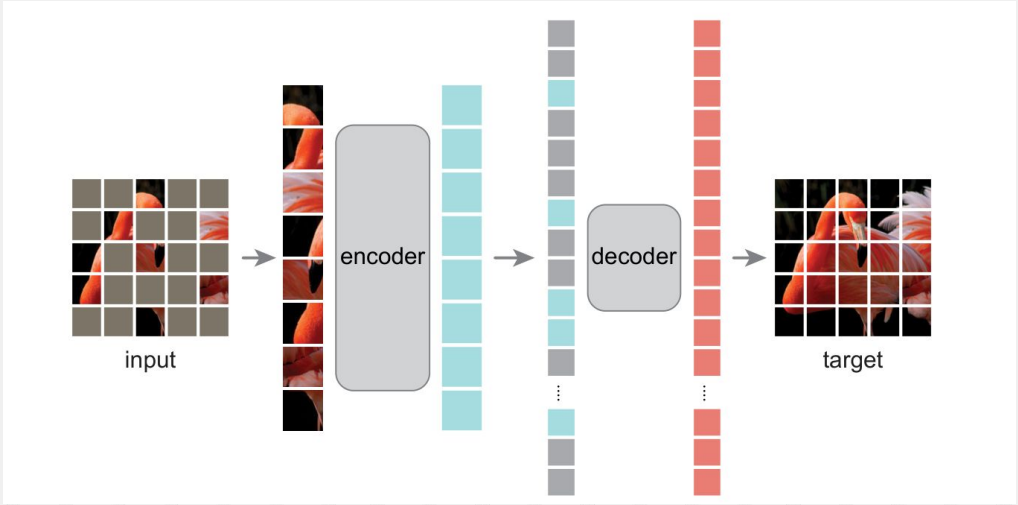
1. ViT Based Encoder

An image is patchified and then these patches are fed into transformer encoder as their Linear projections. Then, after normalizing them, they are fed into FFNN as shown.

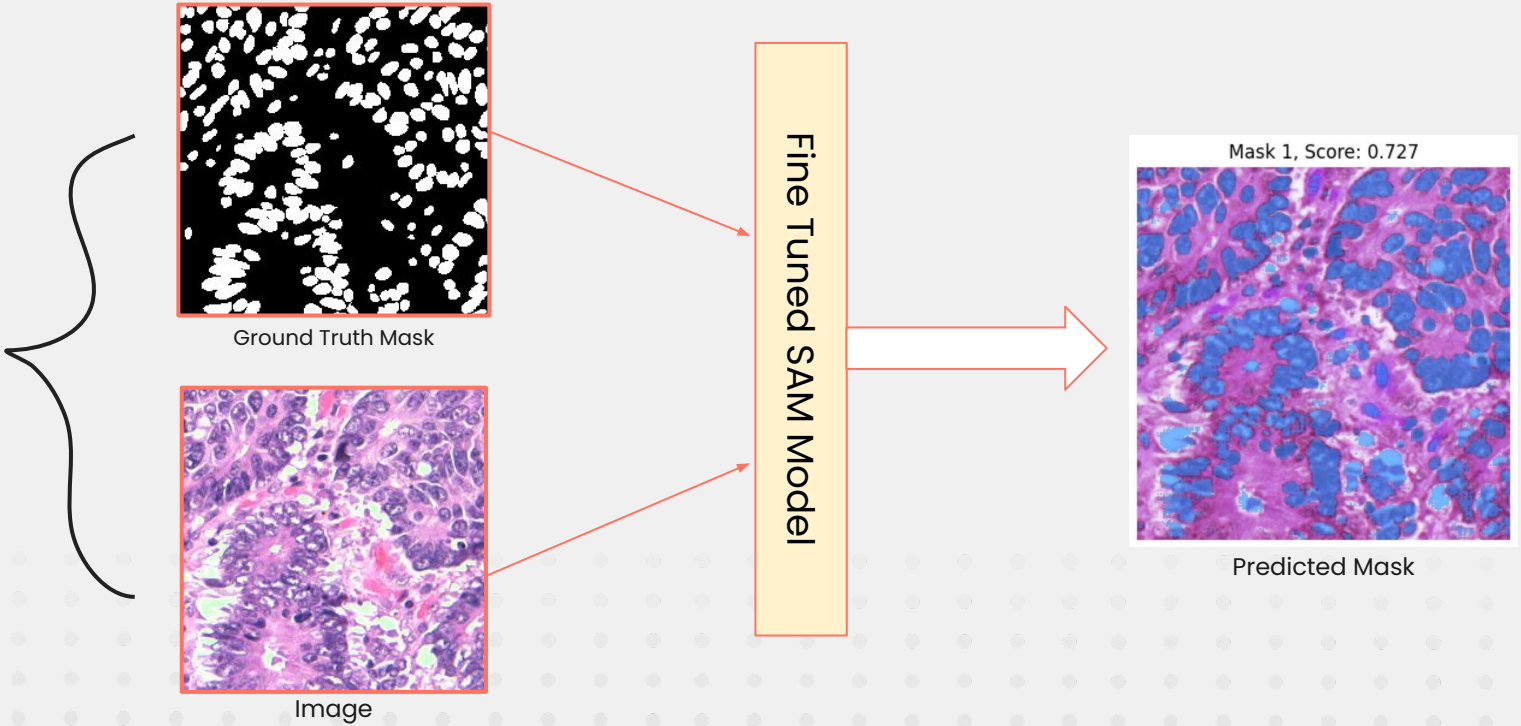


2. Decoder used for regenerating original image

- During pre-training, a large random subset of image patches is masked out (say 75%).
- The encoder is then applied to remaining small subset of unmasked/visible patches.
- Decoder is then trained to predict the true image
- After pre-training, decoder is discarded as Encoder learns the required spatial features.



3. Fine-Tuning on labelled images



4. Data Preprocessing

Stain Normalization and Augmentation:

Stain normalization reduces the variations among stains, while augmentation increases the variety of stain styles the program encounters during training. RandStainNA unifies these approaches for better overall performance

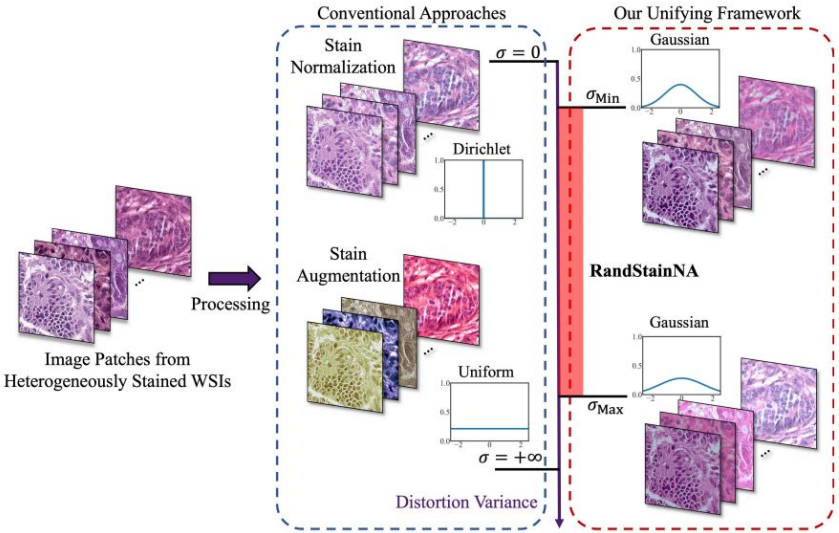


Fig. 1. The overall framework of the proposed RandStainNA.

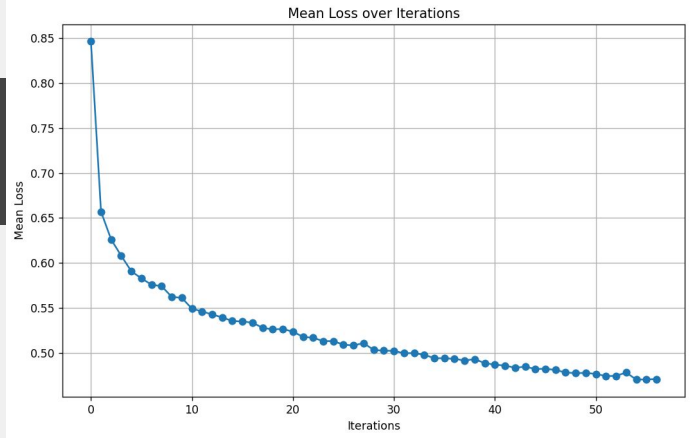
Training

4 different models trained over ParamHimalaya on 600 Labels:

1). Non-binary mask -100Ep.
2). Binary Mask - 200Ep.
3). RandStainNA Augmentation - 200Ep.
4). Stain Normalization - 100Ep.

With combined training time of 28Hrs.

Mean Loss values Vs. Epochs



Using ***"ADAM"*** optimizer with learning rate of 1e-5, and trying out multiple Loss functions such as ***Dice Focal*** Loss, ***Dice Cross Entropy*** Loss, ***Dice*** Loss, ***Focal*** Loss for best results. All these loss functions are implemented by us and no other open library (such as MonAI) were used.

```
from torch.optim import Adam
# Initialize the optimizer and the loss function
optimizer = Adam(model.mask_decoder.parameters(), lr=1e-5, weight_decay=0)
#Try DiceFocalLoss, FocalLoss, DiceCELoss
class DiceCELoss(nn.Module):
    def __init__(self, smooth=1e-5, squared_pred=True, reduction='mean'):
        super(DiceCELoss, self).__init__()
        self.smooth = smooth
        self.squared_pred = squared_pred
        self.reduction = reduction
```

Fig.: Example of implementation of custom DiceCELoss.

Training Dataset

Dataset used for current training tasks, consisted a total of 640 labelled images and their corresponding groundtruths (masks) as .png images. Total of 3842 unlabelled samples were used as given for the task.

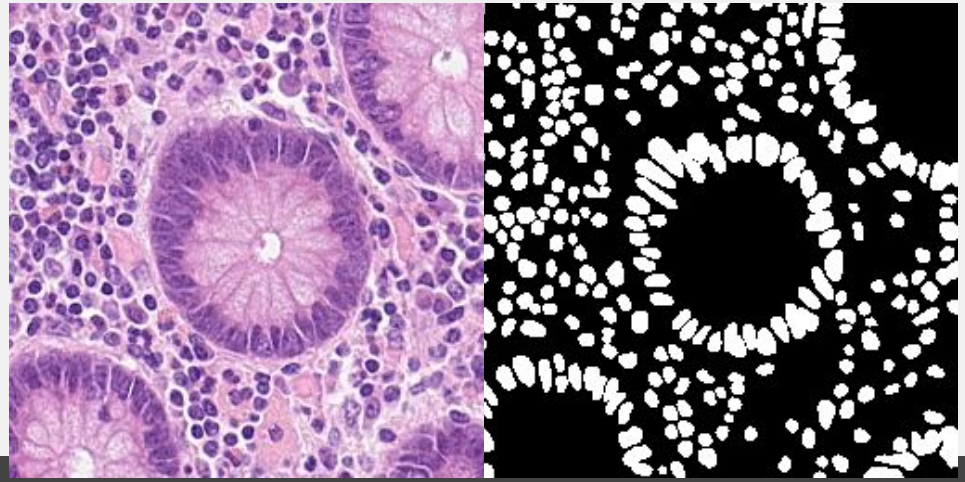


Fig.: Example Image from dataset used and its corresponding mask

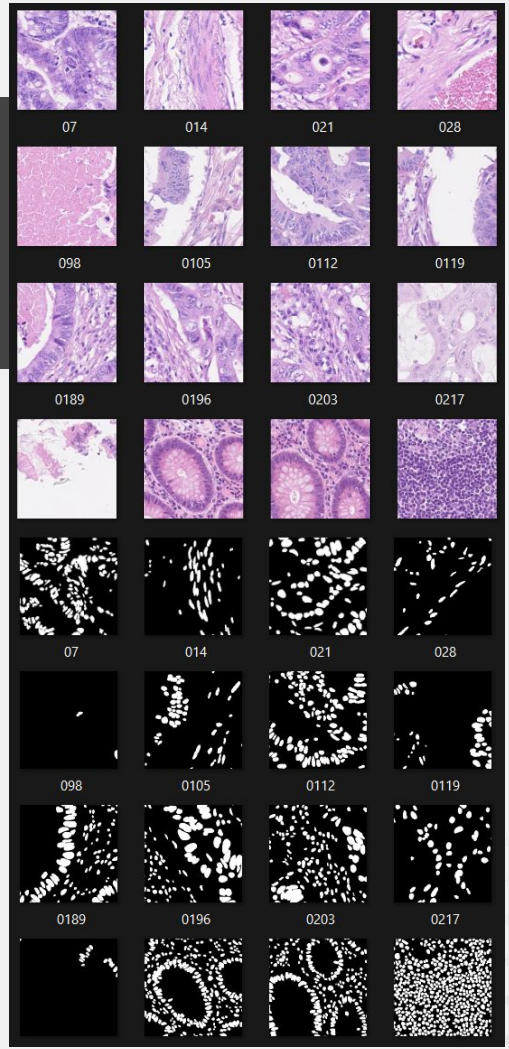
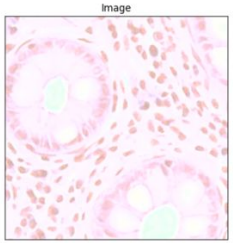
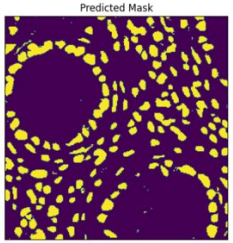


Fig. Snapshot of some of the samples(image(above) and masks(below)) from the dataset used.

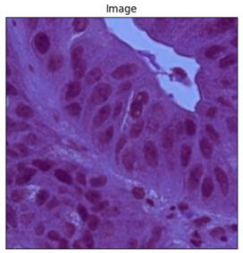
Results

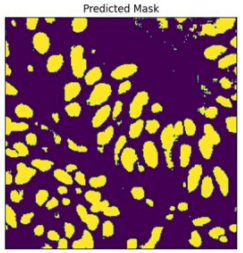




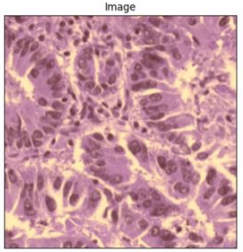

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                  powershell

2024-04-28 00:16:17.212207: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results due to floating-poi
nt round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
Model loaded
Pixel Accuracy: 0.9288787841796875

IOU: 0.6726595968818035

Dice coefficient: 0.8042994499727085
```

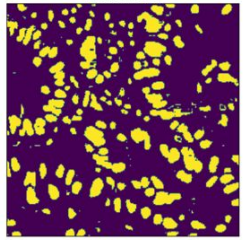
Results on model trained with augmentation






```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                  python

nt round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
Model loaded
Pixel Accuracy: 0.9150390625

IOU: 0.7196233445792839

Dice coefficient: 0.836954612058565
```

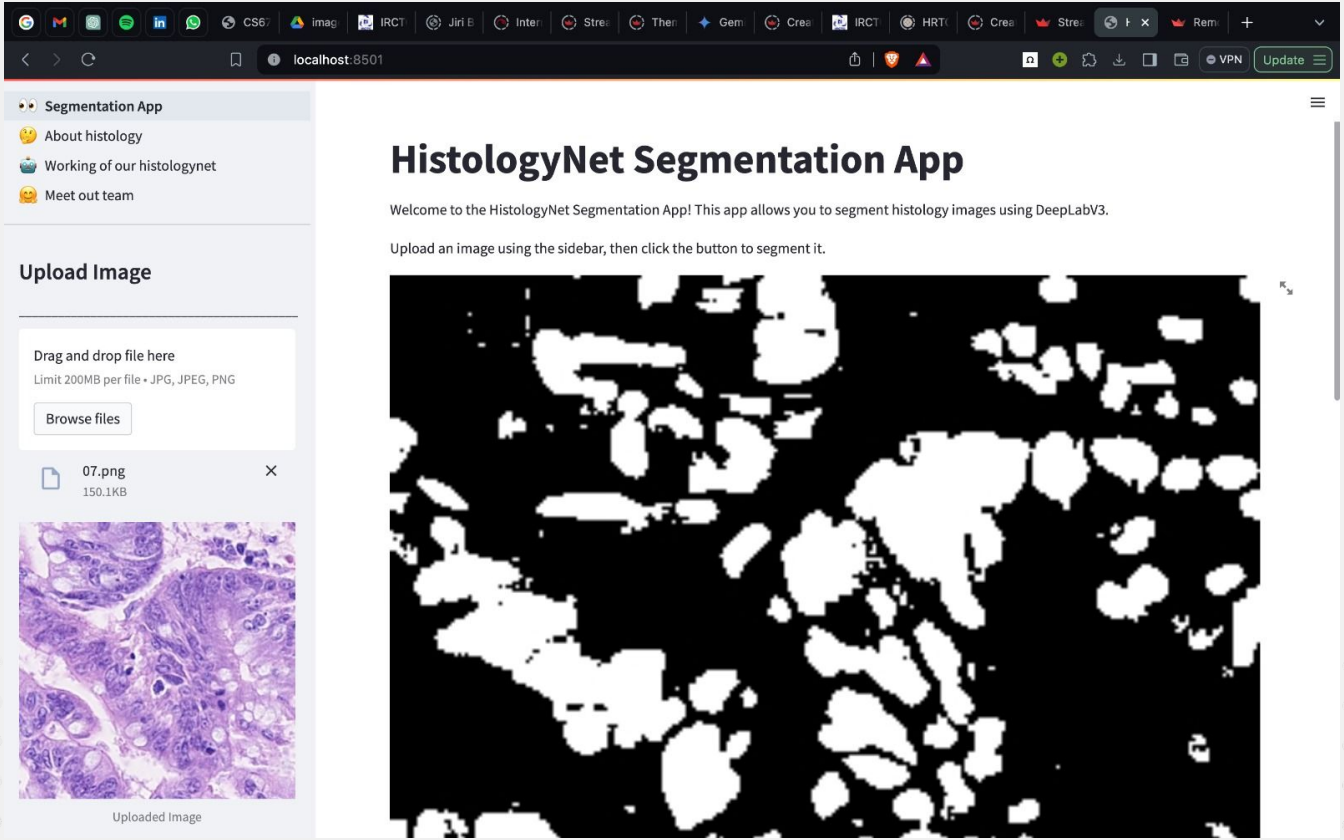





```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                  powershell

2024-04-28 00:12:02.413123: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results due to floating-poi
nt round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
Model loaded
Pixel Accuracy: 0.918304443359375

IOU: 0.6675979387843795

Dice coefficient: 0.8006701414743113
```

Current evaluated results on test data (on model without augmentation):
DiceCoeff: 0.661
IoU: 0.59

WebApp:


localhost:8501
Update
VPN
Segmentation App
About histology
Working of our histologynet
Meet out team
Upload Image
Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG
Browse files
07.png
150.1KB
Uploaded Image
HistologyNet Segmentation App
Welcome to the HistologyNet Segmentation App! This app allows you to segment histology images using DeepLabV3.
Upload an image using the sidebar, then click the button to segment it.

Thank You!

Harsh Kumar - B20045
Nikhil Ujjwal - B20056
Ujjawal - B20070
Ankit Kumar - B20148