

## Homework 3: Medical Image Segmentation

Name: **Anay Abhijit Joshi**

### Step 1: Data

- 1) How many data samples are included in the dataset?

There are 100 Data Samples in the dataset.

```
import os
train_image_count = len(os.listdir('data/Data/train/image'))
test_image_count = len(os.listdir('data/Data/test/image'))
total_samples = train_image_count + test_image_count
print(f"Total data samples: {total_samples}")
```

Total data samples: 100

- 2) Which problem will this dataset try to address?

This dataset addresses the problem of retinal blood vessel segmentation to aid in the diagnosis and management of retinal vascular diseases like diabetic retinopathy and macular degeneration.

- 3) What is the dimension ranging in the dataset?

Initially, the given dataset has all images have dimensions as 512x512.

```
import os
from PIL import Image

# Paths to the train and test image folders
train_image_path = 'data/Data/train/image'
test_image_path = 'data/Data/test/image'

# Collect dimensions of all images in the train folder
train_dims = [Image.open(os.path.join(train_image_path, filename)).size for filename in os.listdir(train_image_path) if filename.endswith('.png')]
test_dims = [Image.open(os.path.join(test_image_path, filename)).size for filename in os.listdir(test_image_path) if filename.endswith('.png')]

# Print one line summaries
print(f"Train Image Dimensions: {set(train_dims)}")
print(f"Test Image Dimensions: {set(test_dims)}")
```

Train Image Dimensions: {(512, 512)}  
Test Image Dimensions: {(512, 512)}

- 4) Does the dataset have any missing information? E.g., missing features.

No, the dataset does not have any missing information, i.e. missing features.

```
train_images = 'data/Data/train/image'
train_masks = 'data/Data/train/mask'

missing_images = [img for img in os.listdir(train_images) if img not in os.listdir(train_masks)]
print(f"Missing masks for images: {missing_images}")
```

Missing masks for images: []

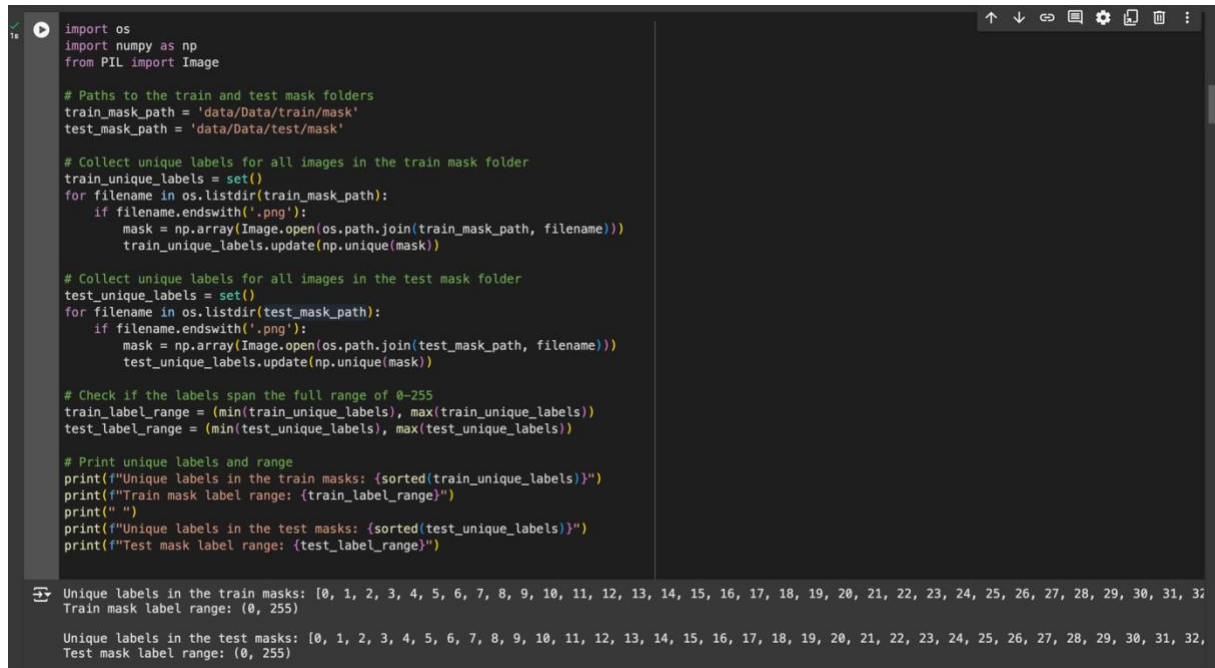
5) What is the label of this dataset?

The label of this dataset is a **binary mask** used for segmentation. Each pixel in the mask is labeled as:

1: Indicates a pixel that belongs to a blood vessel.

0: Indicates a background pixel.

This binary classification helps the model learn to differentiate between the blood vessels and the background in the retinal images for accurate segmentation.



```
import os
import numpy as np
from PIL import Image

# Paths to the train and test mask folders
train_mask_path = 'data/Data/train/mask'
test_mask_path = 'data/Data/test/mask'

# Collect unique labels for all images in the train mask folder
train_unique_labels = set()
for filename in os.listdir(train_mask_path):
    if filename.endswith('.png'):
        mask = np.array(Image.open(os.path.join(train_mask_path, filename)))
        train_unique_labels.update(np.unique(mask))

# Collect unique labels for all images in the test mask folder
test_unique_labels = set()
for filename in os.listdir(test_mask_path):
    if filename.endswith('.png'):
        mask = np.array(Image.open(os.path.join(test_mask_path, filename)))
        test_unique_labels.update(np.unique(mask))

# Check if the labels span the full range of 0-255
train_label_range = (min(train_unique_labels), max(train_unique_labels))
test_label_range = (min(test_unique_labels), max(test_unique_labels))

# Print unique labels and range
print(f"Unique labels in the train masks: {sorted(train_unique_labels)}")
print(f"Train mask label range: {train_label_range}")
print("")
print(f"Unique labels in the test masks: {sorted(test_unique_labels)}")
print(f"Test mask label range: {test_label_range}")
```

Unique labels in the train masks: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32]  
Train mask label range: (0, 255)

Unique labels in the test masks: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32]  
Test mask label range: (0, 255)

6) How many percent of data will you use for training, validation and testing?

Typically, I would use 80% for training, 10% for validation, and 10% for testing. This split ensures a balanced approach for model training and evaluation.

7) What kind of data pre-processing will you use for your training dataset?

For my training dataset, I resize the images and masks to a common size of 256x256 and ensure they are properly loaded and converted to arrays. I also normalize the pixel values by dividing them by 255.0 to scale them between 0 and 1 for consistent model training (only for “with normalization” case). Additionally, I apply data augmentation techniques, such as rotations and flips, to introduce variety and improve generalization. I am planning to use the Adam optimizer for stable and efficient training and the binary cross-entropy loss for accurate segmentation learning.

## Step 2: Model

Model	IoU	Dice
U-Net 2 layers	0.6888622045516968	0.8153834342956543
U-Net 3 layers	0.6654919981956482	0.7985665202140808
U-Net 4 layers (EXTRA)	0.6799156665802002	0.8089066743850708

## Step 3: Objective

### Binary Cross-Entropy

I used binary cross-entropy loss as it is standard for segmentation tasks with binary masks. This loss function measures the performance by calculating the difference between the true label and the predicted probability, making it suitable for normalized data (scaled between 0 and 1). Using binary cross-entropy helped me to ensure that the model effectively distinguished between the background and blood vessels, leading to stable and efficient learning.

## Step 4: Optimization

### Adam Optimizer

For Homework 3 as well, I selected the Adam optimizer to train my models, i.e., the U-Net variations. The reason for choosing Adam lies in its adaptive learning rate capabilities and momentum, which provide a balance of speed and stability during training. This optimizer is particularly well-suited for complex architectures, such as U-Net, where balancing the learning process is crucial.

Adam optimizer's ability to fine-tune learning rates individually for each parameter ensures efficient convergence and helps models achieve high accuracy without extensive manual tuning. This was especially helpful given the architectural variations in my/given U-Net models, as it allowed for robust training and consistent performance across different layers. The ease of use and reliability of Adam make it ideal for tackling medical image segmentation tasks, where precision and balanced training are key factors.

## Step 5: Model selection

Yes, I have used different learning rates, ranging from 0.000001 – 0.00000001. I preferred lower learning rates to prevent model from converging. This is because higher learning rate could lead to underfitting of the data.

Model	Dice and IoU with normalization	Dice and IoU without normalization
U-Net 2 layers	Dice: 0.8153834342956543 ; IoU: 0.6888622045516968	Dice: 0.29464489221572876 ; IoU: 0.1729494333267212
U-Net 3 layers	Dice: 0.7985665202140808 ; IoU: 0.6654919981956482	Dice: 0.30799880623817444 ; IoU: 0.18222160637378693
U-Net 4 layers	Dice: 0.8089066743850708 ; IoU: 0.6799156665802002	Dice: 0.27717503905296326 ; IoU: 0.16103629767894745

### **Best Model:**

#### With Normalization -

Based on the above table, I can state that the best model is **U-Net with 2 Layers**, and this model has secured a **Dice Score of “0.8153834342956543”**, and an **IoU score of “0.6888622045516968”**.

With normalization, I scale my input data to a 0–1 range, which helps the model converge faster and learn more effectively, as it avoids large value disparities. This leads to better gradient updates, resulting in a higher Dice Score and IoU Score, as seen with my **U-Net 2 Layers** achieving a **Dice Score of 0.8153834342956543** and **IoU Score of 0.6888622045516968**, as discussed earlier.

#### Without Normalization –

Based on the above table, I can state that the best model is **U-Net with 3 Layers**, and this model has secured a **Dice Score of “0.30799880623817444”**, and an **IoU score of “0.18222160637378693”**.

Without normalization, my input data retains its original range (0–255), leading to larger gradients and potential numerical instability. This can cause inefficient learning and poorer model performance. Hence, my best model, **U-Net with 3 Layers** (for without normalization case), only achieved a lower **Dice Score of 0.30799880623817444**, and an IoU Score of 0.18222160637378693. I expected these results because normalized inputs help models generalize better and optimize more effectively, in my opinion..

### How do you avoid overfit your model and underfit your model?

To avoid overfitting my model, I:

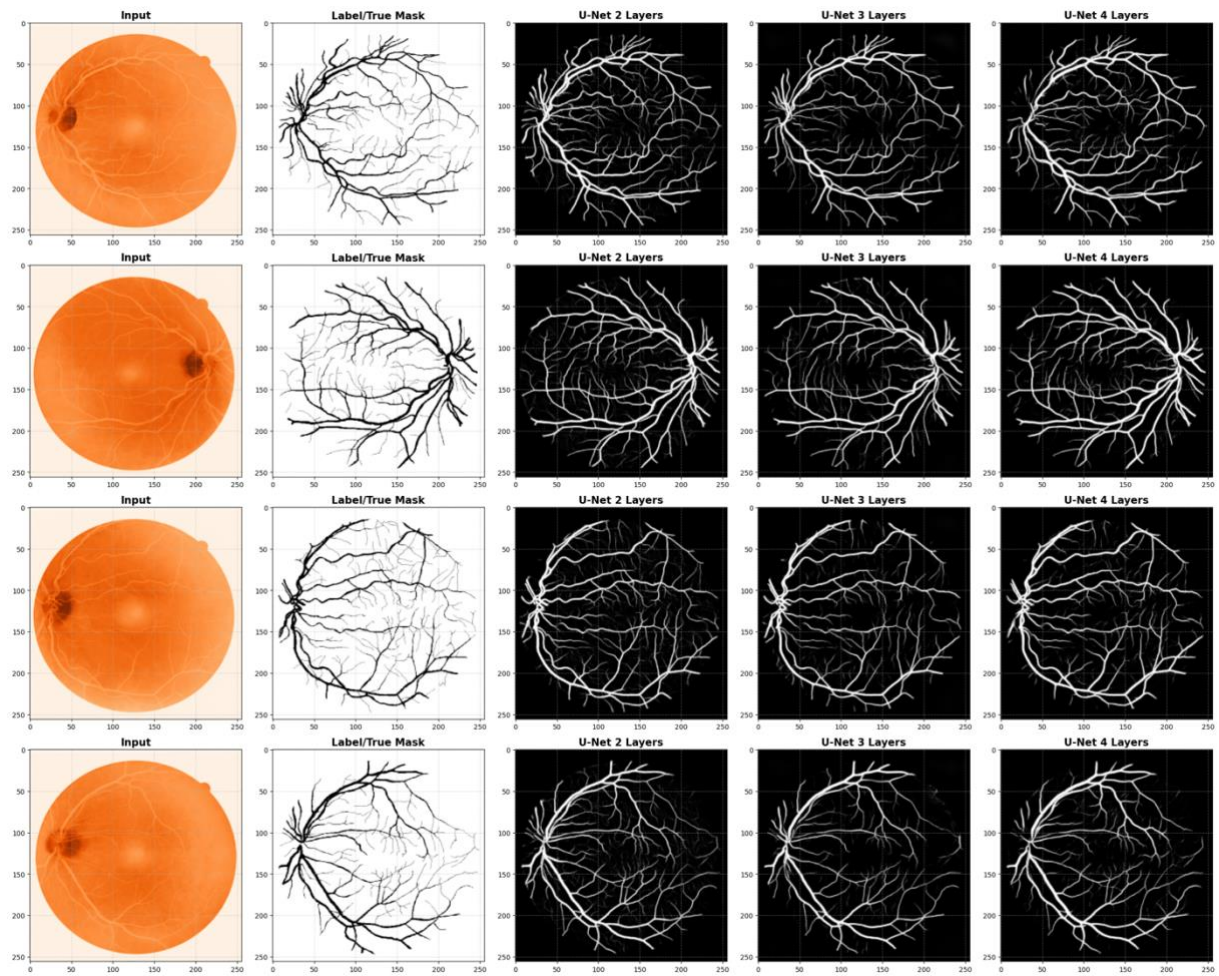
1. **Use Dropout layers:** In my U-Net code, I applied `Dropout(0.5)` to randomly drop units during training.
2. **Apply Data Augmentation:** I used `ImageDataGenerator` to rotate, flip, or shift my training data to introduce variety.
3. **Set Early Stopping:** I included an `EarlyStopping` callback to stop training when the validation loss stops improving.
4. **Add Regularization:** I added L2 regularization to my `Conv2D` layers to prevent large weights.
5. **Simplify the Model:** If needed, I reduce the number of layers or filters in `retina_unet_model()`.

To avoid underfitting my model, I:

1. **Increase Model Complexity:** I add more layers or filters in `retina_unet_model()` to capture more complex patterns and improve learning.
2. **Train for More Epochs:** I increase the `epochs` parameter (i.e., I test with different values) to ensure my model trains long enough to learn the data adequately.
3. **Reduce Regularization:** I was aiming to lower the strength of L2 regularization if it is/was too restrictive and prevents the model from learning effectively.
4. **Ensure Quality Data:** I double-checked my function: `load_and_prepare_retina_data()`, to confirm that my input data is correctly processed and loaded.
5. **Adjust the Learning Rate:** I fine-tune the learning rate in the `Adam` optimizer to avoid convergence issues and allow for more thorough training. I tested different learning rates, mostly, for the case of having “No/Without Normalization”.

## Step 6: Model demo

### With Normalization





## Without Normalization

