# CS 6073 Homework 3: Medical Image Segmentation

Luginbuhl, Dale
luginbdr@mail.uc.edu

October 30th 2023

## 1 Data

### 1.1 How many data samples are included in the dataset?

The dataset is pre-split into a train dataset and a test dataset. The train dataset contains 80 samples. The test dataset contains 20 samples.

### 1.2 Which problem will this dataset try to address?

The goal is to predict the segmentation of retina blood vessels from high-resolution retinal fundus images. "Accurate segmentation of blood vessels is a critical task in ophthalmology as it aids in the early detection and management of various retinal pathologies"

### 1.3 What is the dimension ranging in the dataset?

Each data sample has an input image of size 3x512x512 (CHW) pixels, for a total of 786,432 feature dimensions. Each feature (pixel) takes on integer values in the range $[0, 255]$.

Each data sample has a label image of size 1x512x512 (CHW) pixels, for a total of 262,144 feature dimensions. Each feature (pixel) takes on integer values in the range $[0, 1]$. Values of 0 represent background pixels, while values of 1 represent blood vessel pixels.

### 1.4 Does the dataset have any missing information? E.g., missing features.

The dataset does not have any missing information.

### 1.5 What is the label of this dataset?

The label of the dataset is a mask image representing the segmentation of retina blood vessels. The mask images are of size 1x512x512 (CHW) pixels, and the pixels take on integer values in the range $[0, 1]$. Values of 0 represent background pixels, while values of 1 represent blood vessel pixels.

### 1.6 How many percent of data will you use for training, validation and testing?

The dataset is pre-split into 80 training samples and 20 testing samples. We will further split the training set into a training set and a validation set. The validation set will consists of 20 samples and the remaining training set will consist of 60 samples.

The resulting percentage splits will then be:

Training : 60%

Validation : 20%

Test : 20%

### 1.7 What kind of data pre-processing will you use for your training dataset?

Besides converting the image tensors to floating point values, no data pre-processing was performed. With more time we would've explored data augmentation, as the data set was limited in size, and even the 2-layer unet models were likely complex enough to overfit.

## 2 Model

| Model | Dice | IoU |
|---|---|---|
| U-Net 2 layers | 0.0010 | 22.3292 |
| U-Net 3 layers | 0.0010 | 22.3292 |
| U-Net 4 layers | 0.0010 | 22.3292 |

Table 1: Comparison of model architectures

## 3 Objective

We chose dice loss as the loss function to train the model. We chose this instead of cross-entropy because it is more robust to the class imbalance in

the segmentation masks. There are many more background pixels than blood vessel pixels.

# 4 Optimization

We chose the Adam optimizer to take advantage of momentum in our optimization. Adam allows us to converge to an optimum value more quickly, and it allows provides the potential to escape from local minima.

# 5 Model Selection

| Model | Dice & IoU w/ norm | Dice & IoU w/o norm |
|---|---|---|
| U-Net 2 layers | dice 0.0010, IoU 22.3292 | dice 0.0010, IoU 22.3292 |
| U-Net 3 layers | dice 0.0010, IoU 22.3292 | dice 0.0010, IoU 22.3292 |
| U-Net 4 layers | dice 0.0010, IoU 22.3292 | dice 0.0010, IoU 22.3292 |

Table 2: Model selection

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

We could apply L1 or L2 regularization, dropout, or perform data augmentation.

# 6 Model Performance

In Figures 1 through 6 you can see the loss, dice, and IoU plots for the training and validation data sets for each model. Or well really, you probably can't see anything because the plots are too small. Don't worry though, you aren't missing any meaningful information. The data in the Figures, as well as in Tables 1 and 2 came from real training runs, but they don't tell us anything other than that the training was not being performed correctly. Unfortunately, running short on both time and compute, I was unable to debug any further. The "4 input image with label and models prediction" was not included as the images were useless given the invalid models. The code to generate the image *was* written though and can be found in *generate_prediction_image.py*.

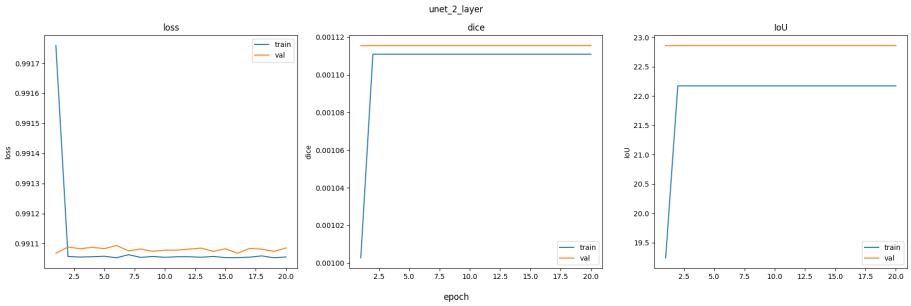


Figure 1: U-Net 2 layers training and validation Dice and IoU

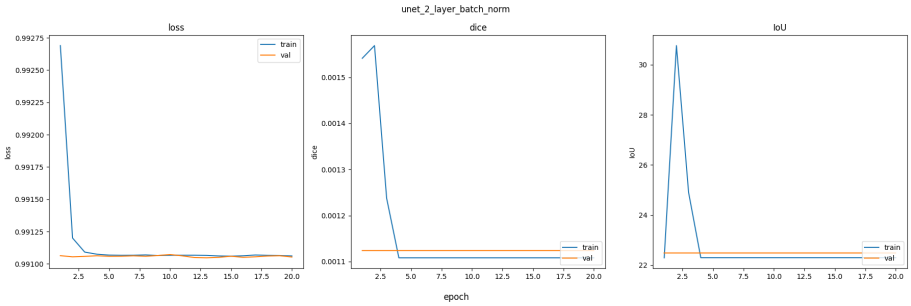


Figure 2: U-Net 2 layers with batch norm training and validation Dice and IoU

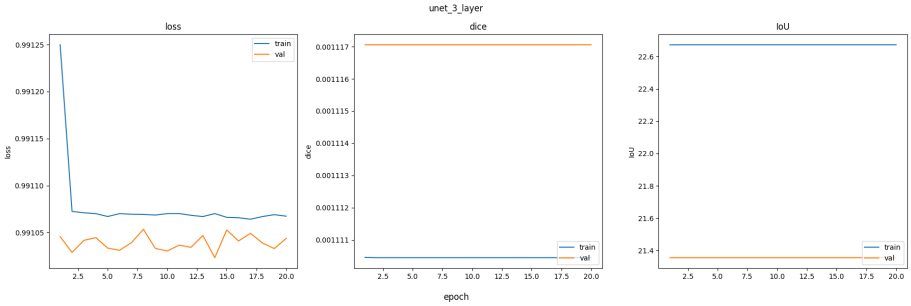


Figure 3: U-Net 3 layers training and validation Dice and IoU

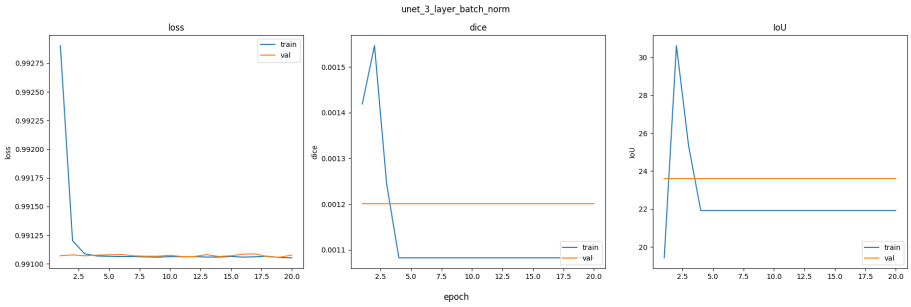


Figure 4: U-Net 3 layers with batch norm training and validation Dice and IoU

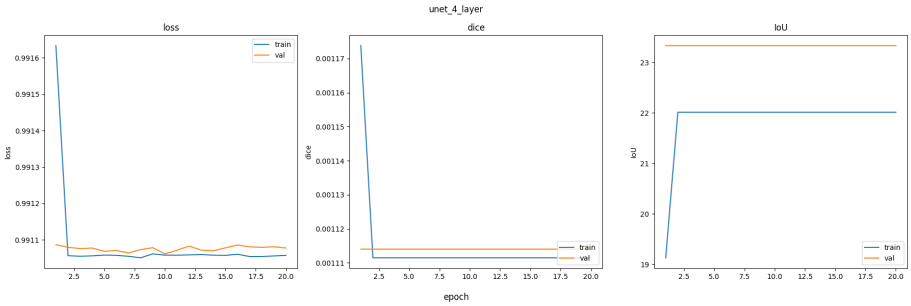


Figure 5: U-Net 4 layers training and validation Dice and IoU

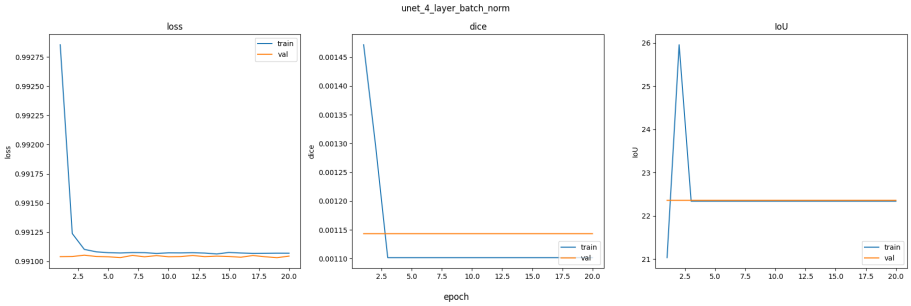


Figure 6: U-Net 4 layers with batch norm training and validation Dice and IoU