CISC/CMPE 472 – Assignment 3 (25%)

Segmentation & Deep learning

Due: March 24th, 2023

For question 1 of this assignment you will be using a subset of data from the PROMISE12 Grand challenge for prostate segmentation. You can read more about the challenge here: https://promise12.grand-challenge.org/Home/. The use of image processing libraries such as OpenCV, SimpleITK, matplotlib, etc. are **permitted for all questions** in this assignment. Any code from previous assignments may be updated to use image processing libraries. Any libraries that you choose to use must be easily installed into an anaconda environment using either pip install or conda install. You should include in your pdf a list of all libraries that required installation. For the deep learning component of the assignment, you should use the **Keras** library within Tensorflow. Your Tensorflow version must be >= 2.5.0. Your code must be compatible with Python 3 (preferably a recent version). Code that does not run will not be graded.

You have been provided with 2 python files: **Train_unet.py** and **UnetSequence.py.** For Questions 2-9 the code provided can be run using CPU alone relatively quickly.

Your submission should be uploaded to OnQ as a zip file containing one python file containing your code for question 1 named **CISC472_A3_Q1.py,** your modified versions of **Train_unet.py** and **UnetSequence.py** and one PDF file containing your written responses. Also include in your zip folder any files generated in specific questions including your saved model weights for questions 5,6, and 7. Ensure that your name and student number are in the header of all your code and written files.

Dataset:

Before training any kind of AI it is best to get a clear understanding of the data that you will be working with. For most of this assignment we will be working with a simulated dataset that isn't very interesting, so to give you a sense of more real-world data I have provided you with a sample dataset to analyze. The original dataset has been adapted for you to a format that we have seen in class. The dataset consists of anonymized MRI scans of the prostate for 50 different patients. Provided with each scan is a ground truth label map of the prostate in the volume.

1. **Data Exploration** *[5 marks]*
   Select a single scan from the dataset. Using your reslice function from Assignment 2, display the center slice from each imaging plane (axial, sagittal, coronal). Include the image of each of the 3 slices in your PDF. Repeat this process with the ground truth segmentations, including the resulting slices in your PDF. Note any observations that you make about the images. Visualize your scan in 3D Slicer as well in both the 3D view and the slice panels, as you did with your simulated tumor in Assignment 2. Note any further observations you make.
   Examine the dataset as a whole - what is the maximum depth of the volumes? Does each scan contain the same number of slices? What are the dimensions of the volumes? Are these values the same for all volumes in the dataset?

Simulated dataset

Since the dataset above is too large to train a model with on most laptops, we will be creating our own sample dataset to train our model.

2. **Dataset creation** *[5 marks]*
   Use your simulate image function from assignment 1 to generate 100 sample images of a white ellipse on a black background with a bit-depth of 8 (max value 255) with shape (128,128,1). To obtain your ground truth segmentations rescale the images to a bit-depth of 1 (binary image) and save your files as "segmentation_#.png" where # is the index of the order they were generated in (e.g. segmentation_0.png will be the first image generated). To obtain your set of images to use for training your model **randomly** apply the smoothing filter 1-100 times for each image, this will introduce variation into your data to make it more challenging for the model to learn. Save your images as "image_#.png", ensuring that your images are numbered in the same order as your segmentations. Save your images and segmentations in a separate directory named CISC_472_dataset. Store this directory with your assignment code.

Training preparation

Before we can train our model, we must ensure that our data is consistently formatted and reduced to minimize our hardware requirements.

3. **Data splitting** *[5 marks]*
   Divide your data randomly into 3 distinct sets for training, validation and testing. Ensure that there is no overlap between sets. The test set should be composed of a single volume, but you may decide how to divide the data between training and validation. Include a table showing which scans are in each of your 3 sets in your PDF.
4. **Initial preprocessing** *[10 marks]*
   Complete the functions readImage and readSegmentation in the python file entitled UnetSequence. These functions will be responsible for reading in your images and segmentations and providing them to the model. At this point you may also wish to enhance your images using the enhancement techniques covered in assignment 1 (filters, contrast enhancement, etc.). Document your process and explain why you chose the enhancement techniques that you did, or why you chose not to use any enhancement techniques.

   Since the model produces a fuzzy segmentation, we need to reformat our ground truth segmentations to match. We do this by a process called "one-hot encoding". This process converts categorical integer values to a binary representation for each class. For example a pixel value of 1 will be represented by the array [0 1] in a one-hot encoded labelmap, while a pixel with the value 0 would be represented by the array [1 0]. The index location of the 1 in the array indicates the class to which the pixel belongs (0 = background, 1=object). Once one-hot encoding is complete your resulting ground truth segmentations should have the shape (128,128,2).

Training the model

The following questions will require you to compile and train the weights of your model. The model implementation along with the implementation of the IoU metric and loss function have been provided for you in the python files provided. When compiling you may use any optimizer you wish and do not need to provide justification for this. The most commonly used is Adam. Select a relatively small learning rate (e.g. 1e-5). Experiment with the learning rate and make note of what this does to the performance of your model. The metrics used to evaluate the network should include: accuracy, IoU, Hausdorff Distance and Dice Coefficient. Accuracy is already included in Keras, and a custom implementation of the IoU metric has been provided for you with the model implementation, but you are responsible for implementing the Dice Coefficient and Hausdorff distance metrics.

5. **Cross-entropy loss** *[10 marks]*
   Compile your model using the loss function "categorical_crossentropy". Train your model for a set number of epochs. Use the ModelCheckPoint callback to monitor the validation accuracy to ensure that you only save the best weights. Your model should be saved as "unet_ce_loss_accuracy.h5". How many epochs did it take for the model to stop improving? Keras' model.fit() command will also return the training history. Plot the progression of the loss and metrics over the number of epochs for both the training and validation data. Include the graphs and any observations you make in your PDF. Experiment with different parameters to improve your model. Document everything you try and provide your justification in your PDF.
6. **IoU loss** *[5 marks]*
   Repeat all steps question 5, but this time using the custom IoU loss provided. In this case your model should be saved as "unet_iou_loss_accuracy.h5". Include all graphs and observations in your PDF.

7. **Additional metrics** *[5 marks]*
   Create additional model checkpoints that will save the best weights to maximize the validation IOU, Dice coefficient and Hausdorff distance. Repeat question 5 with these checkpoints. Your weights files should be saved as "unet_ce_<*metric_name*>.h5". Modify the number of epochs and/or value of hyperparameters to ensure that the final save point for each metric is not the same (e.g. models should produce slightly different segmentations).

Evaluate the model

Now that we have a trained model, we must evaluate how well it performs on our test dataset. In the Keras library, model.predict() will generate the output of the model, while model.evaluate() will compute the metrics that your network was compiled with on the test set. For each of the following questions use your models from questions 5-7 to generate predictions for the test set.

8. **Qualitative evaluation** *[5 marks]*
   Generate the predictions using each of your saved models (remember to load the best weights before predicting). Visually inspect the predicted images (remember to convert them back to ordinal values from one_hot_encoding). What do you notice about how the different metrics and loss functions affected the predicted segmentations? Provide a possible explanation why they differ. Include all images and explanations in your PDF. You are not required to submit your predicted image files.

9. **Quantitative evaluation** *[5 marks]*

   In addition to computing the metrics that the model was compiled with, also compute the following metrics for each model on the test data: average precision and recall (overall and for each class). Include a table in your PDF comparing the metrics for each of the models. Note any observations that you make about the results. Were some metrics higher than others? What do the metrics indicate about the similarity between the prediction and the ground truth labels?

Mark breakdown

The mark breakdown for this assignment is shown in the table below. Please note that in addition to marks associated with each question, there are 10 marks dedicated for proper coding style. Code that is illegible or difficult to interpret or does not meet the style guidelines outlined below will be docked marks.

Question	**Marks associated**
1.	5
2.	5
3.	5
4.	10
5.	10
6.	5
7.	5
8.	5
9.	5
Coding style	10
Total	**65**

Style guidelines

Code for Question 1 should be implemented in a **CISC472_A3_Q1.py.** You must also include in your submission the modified versions of **Train_unet.py** and **UnetSequence.py.** Your code (all files) must contain an initial header that includes your name and student number. All import statements should be properly organized at the top of the file. All code must be properly structured within functions and all functions should be documented with initial comments describing the behavior of the function as well as the parameters and returns. The use of inline comments should be minimized and used only where necessary. Function and variables should have meaningful names that indicate their purpose (e.g. no single letter variable names). For this assignment the use of image processing libraries such as OpenCV, SimpleITK, matplotlib, etc are permitted everywhere. For the deep learning component of the assignment, you should use the **Keras** library within Tensorflow. Your Tensorflow version must be >= 2.5.0. Any libraries that you use must be easily installed in an anaconda environment using a simple pip install or conda install command. Document which libraries you used in your pdf.

Reminder to make sure that your code runs using Python 3. If a function is causing errors that you are unable to fix before the deadline, please comment out that section of code. You may describe what you were trying to do in your PDF file to receive partial marks. Code that does not run will not be graded.