Deep Learning - Spring 2023 Assignment 4
Image Segmentation

Due Date: 11:59 PM on Monday, 29th April 2023 Total Marks: 200

Submission:

Submit all of your codes and results in a single zip file with the name FirstName_RollNumber_04.zip

- Submit a single zip file containing.
  - (a) codes (b) report (c) Jupyter Notebooks (d) Readme.txt
- There should be **Report.pdf** detailing your experience and highlighting any interesting results. Kindly don't explain your code in the report, just explain the results. Your report should include your comments on the results of all the steps, with images, for example, what happened when you changed the learning rate, etc.
- Readme.txt should explain how to run your code, preferably it should accept the command line arguments e.g dataset path used for training the model.
- The assignment folder must contain one .py file and one .ipynb file. In Jupyter Notebook important outputs must be shown and will be checked during evaluation.
- In the root directory, **there should be 1 python files (contain all the codes (tak1, task2 … taskN), N jupyter files of each task separated for visualization purpose, and a report**.
- You can save weight on your system, will be asked to show it in evaluation.
- Root directory should be named **as FirstName_RollNumber_04**
- Your code script files should be named as '**rollNumber_04_task1.py**'
- You have to submit a .py code file (containing all the codes). Follow all the naming conventions.
- For each convention, there is a 3% penalty if you don't follow it.
- Email the instructor or TA if there are any questions. You cannot look at others' code or use others' code, however, you can discuss it with each other. **Plagiarism will lead to a straight zero with additional consequences as well.**
- 5% (of obtained marks) deduction per day for a late submission.
- Late submissions will only be accepted till 4th May 11:59PM. After that no assignment will be accepted.
- **DON'T RESUBMIT THE DATASETS PROVIDED IN YOUR SUBMISSION.**
- Use the provided dataset for this assignment, do not use any other dataset.

Note: For this assignment (and for others in general), **it is strictly prohibited to search online for any kind of implementation (except the link provided and do not copy code from provided links)**.

You are not allowed to share code or view others' code. You should only possess your own implementation related to the assignment.

⚠ **Important:** With your submission, you also need to submit one .py file named **'rollNumber_04_allCode.py'** in your zip folder. This file should contain the code of all 'code files' in your project. NO SUBMISSION WILL BE ACCEPTED WITHOUT IT.

Task 1: Implementation of UNet Architecture 100 Marks

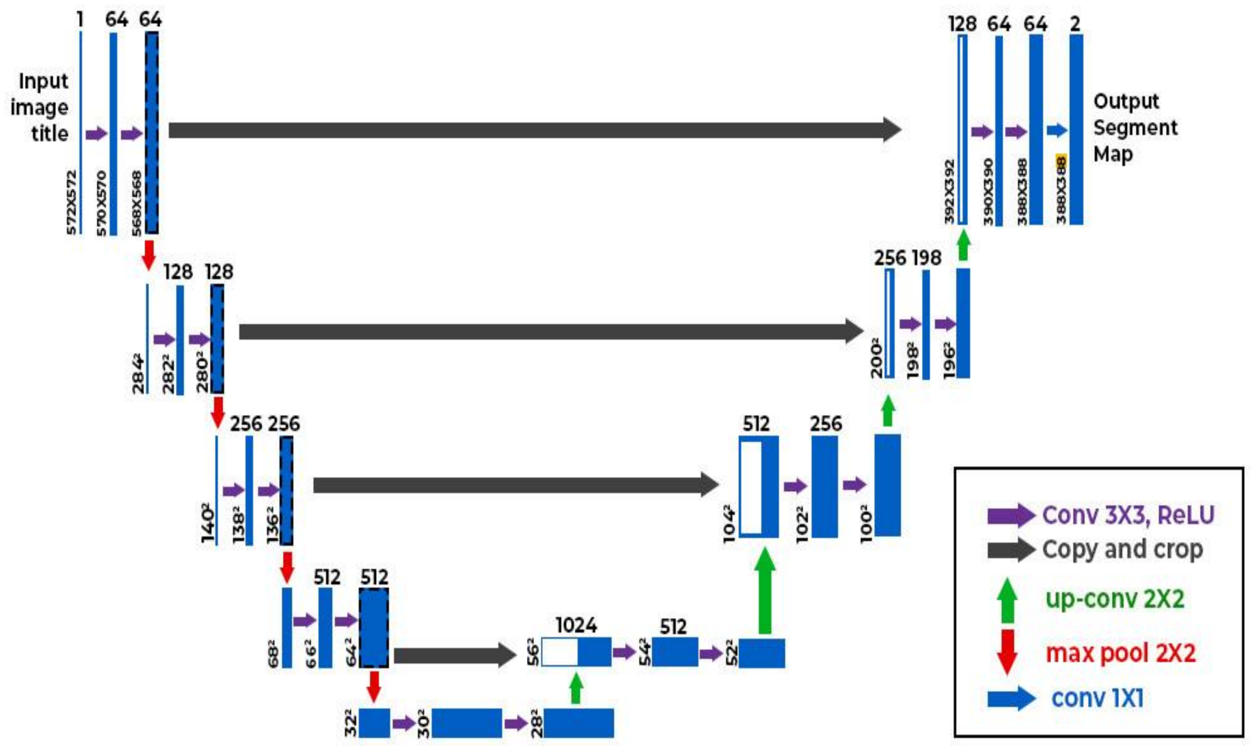
Dataset:

For this task you have to use Breast Cancer dataset. The images are in PNG format. The ground truth images are presented with original images. The images are categorized into three classes, which are normal, benign, and malignant. You have to load the dataset using dataset and data loader class.

Steps:

1. Load all images, all file names and split the mask name and image names from these filenames.
2. Split the filenames into training, validation, and testing sets.
3. In your dataset class, you must implement the code which will return image, mask and label (0,1, or 2).
4. Make sure your code will not load all the images and masks at once in memory. Because Dataset and dataloader classes use these things to save memory at runtime. If you did this way, your code would run for this dataset however in real life scenario you have to handle GBs of data, and your approached will consume memory. It is the best practice to write effective code.
5. Don't forget to normalize your images and masks.
6. If you loaded all the images at once, half marks from obtained of this section will be deducted.

Model Architecture:



Task 1 (Part 1): UNet with Skip Connections

Implement UNet with skip connection as shown in the above diagram. You can use any input shape like (128, 128, 3), (128, 128, 1) or (256, 256, 1) more or less. However, make sure that your input image and mask size should be the same because your model will output the image with same size along with one channel only. Your dataloader must handle these things. For in-depth reading please follow the paper link.

Task 1 (Part 2): UNet without Skip Connections

Now you have to modify your model architecture and remove skip-connections from your networks. Observe the model output with and without skip connections and write your findings and results in the report.

You have to implement two parts for Taks-1 and for each part you have to implement the following methods.

Important Functions to implement for each part:

1. **Training:**

   Now create a function to train the initialized network. You also have to keep track of loss and accuracy on training and validation data for each epoch to display loss and accuracy curves.

   The train method of your model should be like as below.

   model, loss_array, accuracy_array = train (net, train_set_x, train_set_y, valid_set_x, valid_set_y, learning_rate, training_epochs, loss_func, optimizer, batch_size)

   ----

2. **Update weights**.

   You can also use the built-in optimizer and loss functions of PyTorch.

3. **Early Stopping, save best weights and Learning rate decay**:

   Early stopping: In the training function, you have to keep track of training and validation loss. Early stopping is done when validation loss starts increasing while training loss is decreasing or constant.

   Best Weights: Keep record of dice score, save the weights where you got best validation dice score. Use these weights to test your model.

   Learning rate decay: If training loss is not decreasing up to several epochs, then we can decrease the learning rate (learning rate decay). If training loss does not decrease for many epochs, then we can also apply early stopping.

4. **Loss & Optimizer**:

   Use Dice Loss as your loss function. Write your own function for Dice Loss. Built-in functions aren't allowed. Optimizer and hyper-parameters are up to you.

5. **Save Network**:

   Write a function that saves your network so that it can be used later without training.

6. **Load Network**:

   Write a function that loads your saved network so that it can be used without training. The function should return the loaded model.

7. **Testing Step**:

   Now create a function that uses a trained network to make predictions on a test set. The function should return predictions of the model.

   pred = test (net, model, test_set_x, test_set_y)

8. **Runtime Testing**:

   Write a function that takes input an image from the user at run time, preprocesses it, and predicts its mask.

9. **Visualize Step**:

Write a function that plots loss and accuracy curves and sample images and predictions made by the model on them. Show your loss and dice score curves on training and validation sets. For a testing set, you should report only Dice score.

10. **Visualize 5 accurate and wrong predictions**.

Accurate Predictions are those whose dice score is greater than or equal to 60% or define your own threshold, and those less than the threshold will be considered as wrong predictions.

11. **TSNE data representation**.

Now you have to get the encoder last layer and pass it to the TSNE function. Your dataloader labels will be used here to show the representation of your encoder output with these labels.

Note: Make sure to write each function for Task-1 both parts.

Note: Any changes in deliverable will be updated to you.

🤓 Good Luck 🤓