# BME646 and ECE60146: Homework 7

**Spring 2024**
**Due Date: 11:59pm, March 17, 2024**
**TA: Akshita Kamsali (akamsali@purdue.edu)**

Turn in typed solutions via BrightSpace. Additional instructions can be found at BrightSpace. Late submissions will be accepted with penalty: **-10 points per-late-day, up to 5 days**.

## 1 Introduction

This homework introduces you to semantic segmentation of images with neural networks. Semantic segmentation is of great importance in biomedical imaging where the anotomical features and the tissues affected by pathology can have highly irregular shapes. Most neural network architectures for semantic segmentation are based the Encoder-Decoder design as first proposed in [1]. That network is famously known as the U-Net.

In this homework, you will learn about the basics of the Encoder-Decoder networks for semantic segmentation through DLStudio's `mUnet` class. That class is a part of the larger SemanticSegmention class in DLStudio.

The semantic segmentation code in DLStudio is based on `nn.MSELoss`. As you can imagine, such a loss function is not likely to be sensitive to the boundaries of the pixel blobs that you want your neural network to identify. To remedy this shortcoming of the code in DLStudio, this homework will also ask you to code up what is known as the Dice loss, which is also known as the Sørensen-Dice coefficient. It's a popular choice in image segmentation, as it quantifies the overlap between the predicted and the target segmentation masks. Importantly, it provides a smooth and differentiable measure of segmentation accuracy. Additionally, the Dice loss is known to be particularly effective when you have imbalanced datasets.

## 2 Getting Ready for This Homework

1. First of all review the slides 63-83 from the Week 8 lecture. Understand the structure of the mUnet and how it performs semantic segmentation.

2. Download the latest version (2.3.6) of DLStudio that has improved code for Semantic Segmentation from the website or the link below:

   `https://engineering.purdue.edu/kak/distDLS/DLStudio-2.3.6.tar.gz`

   It is highly likely that the latest version of DLStudio in your computer is 2.3.5. What you need for this homework is 2.3.6.

   Perform the appropriate installation in your environment based on the instruction found at the link:

   `https://engineering.purdue.edu/kak/distDLS/#113`

3. Locate the file named `semantic_segmentation.py` in the main `Example` subdirectory in your installation of DLStudio.

   Make yourself as familiar as you can with the script `semantic_segmentation.py`. This is the only script you will be running for this homework.

4. Download the image dataset for DLStudio main module from the website or from the below link:

   `https://engineering.purdue.edu/kak/distDLS/datasets_for_DLStudio.tar.gz`

5. To extract the tar.gz dataset file, use the `tar zxvf` command as provided below:

   ```
   tar zxvf datasets_for_DLStudio.tar.gz
   ```

   You do NOT need to extract the internal **PurdueShapes5MultiObject-10000-train.gz** and **PurdueShapes5MultiObject-1000-test.gz**. You only need to provide the pathname for the folder on your machine containing all the datasets. If done correctly, rest should be handled.

## 3 Programming Tasks

The following are the programming tasks you must do for this homework:

1. Execute the `semantic_segmentation.py` script and evaluate both the training loss and the test results. Provide a brief write-up of your understanding of mUnet and how it carries out semantic segmentation of an image. By "evaluate" we mean just record the running losses during training. One of the most commonly used tools for evaluating a semantic segmentation network is through the IoU loss. If you wish, you can write that code yourself. But that is not required for this homework.

2. The `run_code_for_training_for_semantic_segmentation` function of the SemanticSegmentation class in DLStudio uses just the MSE loss. MSE loss may not adequately capture the subtleties of segmentation boundaries. To this end, we will implement our own Dice loss and augment it with MSE loss and compare it against vanilla MSE.

3. What follows is a code snippet to help you create your own implemenation for Dice Loss. Make sure you set `required_grad=True` wherever necessary to ensure backpropagation, therefore, enabling model learning.

```python

def dice_loss(preds: torch.Tensor, ground_truth: torch.
                                        Tensor, epsilon=1e-6):
    """

    inputs:
         preds: predicted mask
         ground_truth: ground truth mask
         epsilon (float): prevents division by zero

    returns:
        dice_loss
    """

    # implement your logic for dice  loss


    # Step1: Compute Dice Coefficient.
    # For the numerator, multiply your prediction with
    # ground truth and compute the sum of elements(in H
                                and W dimensions).
    # For the denominator, multiply prediction with
    # itself and sum the elements(in H and W dimensions)
                                and multiply ground
    # truth by itself and sum the elements(in H and W
                                dimensions).

    # Step2: dice_coeffecient = 2*numerator / (denominator
                                 + epsilon)

    # Step 3: Compute dice_loss = 1 - dice_coeffecient.

      return dice_loss
```

4. Plot the best- and the worst-case training-loss vs. iterations using just the MSE loss, just the Dice Loss and a combination of the two . Provide insights into potential factors contributing to the observed variations in performance.

5. State your *qualitative* observations on the model test results for MSE loss vs. Dice+MSE loss.

## 4 Extra Credit

For extra credit of 25 points, you repeat the segmentation task in 3 with COCO dataset classes `['cake', 'dog', 'motorcycle']` from HW6.

- Pick images with ONLY single object instance of atleast $200 \times 200$ bounding box. Extract the segmentation as a mask using the `annToMask` . This converts the segmentation in an annotation to binary mask.

- You will need to extract the binary masks instead of the bounding boxes for this task.

- Resize the images to $256 \times 256$ before storing them to the disk. You should also resize the masks accordingly.

- You may continue using the same network from DLStudio. However, you may need to adjust the network parameters to account for $256 \times 256$ resized images from COCO dataset as opposed to $64 \times 64$ images from PurdueShapesMultiObject dataset.

## 5 Submission Instructions

Include a typed report explaining how did you solve the given programming tasks. For HW7, you need to submit the following:

1. Your pdf must include a description of
   - The figures and descriptions as mentioned in Sec. 3, and 4 if you choose to do the extra credit.
   - Your source code. Make sure that your source code files are adequately commented and cleaned up.
2. Turn in a pdf file a typed self-contained report with source code and results. Rename your .pdf file as hw7_<First Name><Last Name>.pdf
3. Turn in a zipped file, it should include all source code files (only .py files are accepted). Rename your .zip file as hw7_<First Name><Last Name>.zip .
4. **Do NOT submit your network weights.**
5. **Do NOT submit your dataset.**
6. For all homeworks, you are encouraged to use `.ipynb` for development and the report. If you use `.ipynb`, please convert it to `.py` and submit that as source code.
7. You can resubmit a homework assignment as many times as you want up to the deadline. Each submission will overwrite any previous submission. **If you are submitting late, do it only once on BrightSpace.** Otherwise, we cannot guarantee that your latest submission will be pulled for grading and will not accept related regrade requests.
8. The sample solutions from previous years are for reference only. **Your code and final report must be your own work.**
9. To help better provide feedback to you, make sure to **number your figures and tables**.

## References

[1] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.