BME646 and ECE60146: Homework 8

Spring 2025
Due Date: 11:59pm, March 24, 2025
TA: Akshita Kamsali (akamsali@purdue.edu)

Turn in typed solutions via Gradescope. Post questions to Piazza. Additional instructions can be found at the end. Late submissions will be accepted with penalty: -10 points per-late-day, up to 5 days.

1 Introduction

In HW7, you successfully detected multiple objects in an image — congratulations on detecting and localizing Dr. Eval! However, object detection alone is not sufficient for tasks requiring pixel-level delineation of the detected objects, as would be the case for detecting, say, tumors in biomedical imagery. This challenge brings us to Semantic Image Segmentation, often referred to simply as Semantic Segmentation.

This homework on Semantic Segmentation adds a new twist to a similar homework given in Spring 2024: This year you are asked to add ASPP at the output of the Encoder in an Encoder-Decoder based segmenter as represented by mUnet in DLStudio. That is, you will split the mUnet at the junction between the Encoder and the Decoder and insert your implementation of ASPP there.

To refresh your memory about ASPP, please review Slide 20 of your instructor's Week 8 lecture dealing with Semantic Segmentation. As shown in the network schematic on Slides 22 and in the Somrita network on Slide 26, the ASPP layer comes after the Encoder in the Encoder-Decoder architecture.

Programming up ASPP should be a piece of cake for you by this time. Let's assume you want to use the same ASPP as in the Somrita network, as mentioned on Slide 28, that network uses Atrous convos at three different rates: 2, 4, and 6. For these you are going to use the same nn.Conv2d operator as for regular convolution, except that you will specify a value for the `dilation`. These values would be 2, 4, and 6 for the three dilation rates.

Another much simpler twist we are adding this year is for you to upload selected images from the Shapes and COCO dataset to the SAM (Segment Anything Model) online interface. You will qualitatively compare the segmentation masks generated by your mUnet model (with and without ASPP) against those produced by the SAM model.

Also note that the current implementation of semantic segmentation in DLStudio utilizes `nn.MSELoss` as the loss function. As for last year, we are asking you to also implement the Dice loss function for a more accurate delineation of the boundaries of the estimated segmentation masks. The Dice loss is widely used in image segmentation as it quantifies the overlap between predicted and target segmentation masks while providing a smooth and differentiable measure of segmentation accuracy.

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. 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.

3. 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

4. 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 with the modified model ASPP at the bottleneck 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. When using Dice+MSE loss, do you think there should be a scaling factor to scale the Dice Loss? Why or Why not?

   Hint: Consider the fact that Dice loss is typically bounded within a specific range, whereas MSE loss is unbounded and can have significantly larger magnitudes. How might this influence optimization, and what adjustments could be made to address it?

5. 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 with two different scale factors for Dice. Provide insights into potential factors contributing to the observed variations in performance.

6. State your *qualitative* observations on the model test results for MSE loss vs Dice loss vs. Dice+MSE loss and show atleast 3 images per case.

4 Repeat with MSCOCO

Now, you repeat the segmentation task in 3 with COCO dataset classes from HW7.

- Pick images with instances of atleast 200×200 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 256×256 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 256×256 resized images from COCO dataset as opposed to 64×64 images from PurdueShapesMultiObject dataset.

5 Bonus: Compare with SAM

In this section, you will compare the segmentation results of your trained mUnet model with the segmentation masks generated by the Segment Anything Model (SAM), a state-of-the-art segmentation model by Meta AI.

5.1 Steps to Work with SAM

Follow these steps to generate segmentation masks using SAM and compare them with your models output:

1. Choose 5 images from the Shapes dataset and 5 images from the COCO dataset that you previously worked with.

2. Open the online Segment Anything Model (SAM) demo available at: `https://segment-anything.com/demo`

3. Upload each of your selected images to the SAM interface.

4. Click on different objects in the image to let SAM generate segmentation masks. Please feel free to play with clicks, evrything and other options. Take screenshots for your report.

5.2 Compare with mUnet Results

Place the segmentation masks from SAM side by side with the masks generated by your mUnet model. Visually compare the **quality** of segmentation, paying attention to:

1. Show images side by side of SAM and mUNet. Making it 2 sets 5x2 images, 1 set for Shapes and the other set for COCO.
2. Edge accuracy: How well does the model outline the object?
3. Completeness: Does it capture all parts of the object?
4. False positives/negatives: Does it include/exclude parts that shouldnt be segmented?

Summarize your qualitative observations comparing SAM vs. mUnet for both datasets. Discuss any differences in performance and how SAMs generalization compares to your model.

6 Submission Instructions

Include a typed report explaining how you solved the given programming tasks. You may refer to the homework solutions posted at the class website for the previous years for examples of how to structure your report

1. **Turn in a PDF file and mark all pages on gradescope**.
2. Submit your code files(s) as zip file.
3. **Code and Output Placement:** Include the output directly next to the corresponding code block in your submission. Avoid placing the code and output in separate sections as this can make it difficult to follow.
4. **Output Requirement:** Ensure that all your code produces outputs and that these outputs are included in the submitted PDF. Submissions without outputs may not receive full credit, even if the code appears correct.

5. For this homework, you are encouraged to use `.ipynb` for development and the report. If you use `.ipynb`, please convert code to `.py` and submit that as source code. **Do NOT submit .ipynb notebooks.**

6. You can resubmit a homework homework 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.** Otherwise, we cannot guarantee that your latest submission will be pulled for grading and will not accept related regrade requests.

7. The sample solutions from previous years are for reference only. **Your code and final report must be your own work.**