Computer Vision - Spring 25

Assignment 4

Deadline : 09 April 2025 11:55 P.M.

Instructors: Prof Ravi Kiran Sarvadevabhatla
and Prof Makarand Tapaswi

March 21, 2025

1 General Instructions

- Your assignment must be implemented in Python.
- While you're allowed to use LLM services for assistance, you must explicitly declare in comments the prompts you used and indicate which parts of the code were generated with the help of LLM services. **If you haven't specified the prompts, then 50% of your marks will be deducted.**
- Plagiarism will only be taken into consideration for code that is not generated by LLM services. Any code generated with the assistance of LLM services should be considered as a resource, similar to using a textbook or online tutorial.
- The difficulty of your viva or assessment will be determined by the percentage of code in your assignment that is not attributed to LLM services. If during the viva you are unable to explain any part of the code, that code will be considered as plagiarized.
- Clearly label and organize your code, including comments that explain the purpose of each section and key steps in your implementation.
- Properly document your code and include explanations for any non-trivial algorithms or techniques you employ.
- Ensure that your files are well-structured, with headings, subheadings, and explanations as necessary. Your assignment will be evaluated not only based on correctness but also on the quality of code, the clarity of explanations, and the extent to which you've understood and applied the concepts covered in the course.
- Make sure to test your code thoroughly before submission to avoid any runtime errors or unexpected behavior.

- **Report all the analysis, comparison and any metrics in the notebook or a separate report that is part of the submission itself. No external links to cloud storage files, wandb logs or any other alternate will be accepted as part of your submission. Only the values and visualizations as part of your commits will be graded.**

2 Fully Convolutional Networks for Semantic Segmentation [35 Marks]

Use the given dataset for training and evaluation. The images and segmentation masks in the dataset are of size 224x224. Use validation split to evaluate the model's performance during training.
For each of the FCN variants and each task, plot the **loss** and **mIoU** (Mean Intersection over Union) curves over the training process. After training the model, evaluate its performance by reporting the **mIoU on the test set**. Additionally, **visualize** few sample predictions from the test set for each task. Display the **predicted segmentation mask** alongside the ground truth image and its corresponding mask side-by-side for comparison.
Note: For evaluating Mean Intersection over Union (mIoU), you can use the `torchmetrics.segmentation.MeanIoU` metric from `torchmetrics`

2.1 Dataset Visualization [5 Marks]

The dataset consists of the following 13 classes:

1. *Unlabeled*
2. *Building*
3. *Fence*
4. *Other*
5. *Pedestrian*
6. *Pole*
7. *Roadline*
8. *Road*
9. *Sidewalk*
10. *Vegetation*
11. *Car*
12. *Wall*
13. *Traffic sign*

Use the given dataset of images and corresponding segmentation masks to create class-specific visualizations. The dataset contains 13 different classes, as shown above, and each pixel in the first channel of the mask image is assigned a value corresponding to its class ID (0-12).
Write a function that reads a segmentation mask image, extracts the first channel, isolates each of the 13 classes individually by creating binary masks, and visualizes each binary mask clearly with appropriate titles indicating the respective class names.

2.2 FCN Variants [30 Marks]

1. Implement and train three variants of the FCN model for semantic segmentation as shown in the image, using either a VGG16 or VGG19 backbone pretrained on ImageNet. In this part, freeze the backbone weights during training. Train and evaluate three variants of the FCN architecture as

outlined in the paper: `FCN-32s`, `FCN-16s`, and `FCN-8s`, on the provided dataset.

2. Now repeat the previous task, this time fine-tuning all layers of the chosen VGG16 or VGG19 backbone, allowing updates to all network weights. Again implement, train, and evaluate the three variants (`FCN-32s`, `FCN-16s`, `FCN-8s`) on the provided dataset.

3. Explain the differences between FCN-32s, FCN-16s, and FCN-8s versions. Discuss the segmentation performance of all three versions. Additionally, compare the performance and segmentation quality with that obtained when the backbone is frozen.

   Reference Paper: https://arxiv.org/abs/1411.4038

3 Semantic Segmentation using U-Net [65 Marks]

Use the given dataset for training and evaluation. The images and segmentation masks in the dataset are of size 256×256. Use validation split to evaluate the model's performance during training.
For each of the following tasks, plot the **loss** and **mIoU** (Mean Intersection over Union) curves over the training process. Train each model for atleast **50 Epochs** or until convergence. After training the model, evaluate its performance by reporting the **mIoU on the test set**. Additionally, **visualize** few sample predictions from the test set for each task. Display the **predicted segmentation mask** alongside the ground truth image and its corresponding mask side-by-side for comparison.

3.1 Vanilla U-Net [25 Marks]

1. Implement a U-Net architecture as shown in the image. The network consists of three main components: an encoder, a bottleneck, and a decoder. It features multiple resolution levels, with each level containing a fixed number of convolutional blocks. Use four resolution levels, where feature resolution is progressively reduced using max-pooling or strided convolution and restored using transposed convolution. Each convolutional block consists of two convolutional layers.

   Reference Paper: https://arxiv.org/abs/1505.04597

3.2 U-Net without skip connections [10]

1. Skip connections between encoder and decoder are one of the key innovations in the U-Net architecture. In this task, implement and train the U-Net architecture without the skip connections. To ensure a fair comparison, train the model for the same number of epochs as in the previous task.

2. What differences do you observe in the visualized results compared to the standard U-Net results? Discuss.

3. Discuss the importance of skip connections in U-Net. Explain their role in U-Net's performance.

3.3 Residual U-Net [5]

1. Use the same U-Net architecture with skip connections from the first task (Vanilla U-Net). Replace the standard convolutional block with a **residual convolutional block**.

   A residual convolutional block consists of two convolutional layers with a skip connection that adds the input of the block to its output. However, when the number of input and output channels differs, a 1×1 convolution is applied in the skip connection to match dimensions.

3.4 Gated Attention U-Net [25 Marks]

1. Implement the additive attention gate mechanism as shown in the image and proposed in the paper. You can either use the vanilla U-Net or the residual U-Net as the base U-Net from previous tasks. Integrate the gated mechanism into the **skip connections** of the U-Net. An illustration from the paper is shown in the image. Keep the value of coefficient α as 1.

2. What are some advantages of using Attention gates as per the paper? How does gating signal at skip connections help in improved performance?

3. What differences in results do you observe as compared to the standard U-Net results? Discuss.

   Reference Paper: https://arxiv.org/abs/1804.03999

Good luck with the assignment!