Computer Vision Mini-Project

February 2025

Abstract

This describes the Computer Vision assessed practical. The main goal is to explore image segmentation techniques on a given dataset. You will experiment with traditional image processers and modern deep learning approaches and investigate how different pre-training and feature extraction strategies affect segmentation performance. There are 4 stages to this assignment, and you must do all 4 for full marks. You must do this project with a partner.

Task Background

Image segmentation is the process of partitioning an image into multiple segments (sets of pixels). The goal of segmentation is to simplify or change the representation of an image into something more meaningful and easier to analyze. In many computer vision systems, segmentation constitutes a crucial step towards object recognition, scene understanding, medical analysis, and more. You will be given a dataset, consisting of a set of images. The dataset is already split into training, validation, and testing folders for you. You will need to implement a few segmentation models to segment these images. Try to make each model as small as possible while keeping accuracy as high as possible. There are no expected evaluation scores and mostly your approach to the problem will be assessed.

The assignment is to:

1. **Dataset preprocessing and augmentation**
   The dataset contains labeled images and corresponding ground-truth masks that indicate which pixels belong to which object or category. Make sure you familiarize yourself with the dataset structure (train/validation/test splits, image dimensions, mask formats, label classes, etc.).

   a) Initially, the images in this dataset might have different dimensions, you are required to resize the image to make them consistent for training.
   b) Data augmentation: please implement data augmentation techniques as you see fit to increase your segmentation performance. These techniques include but are not limited to flipping, rotation, translation, elastic transformations, color jitter, random cropping.

After finishing task one, you will have a unified dataset suitable for training and testing all your following networks. Note, you must report your evaluation metrics on the same testing dataset.

2. **Segmentation network design and implementation**

   Given the dataset with **3** classes of objects, you can simply consider image segmentation as a pixel-wise classification task. The input to your segmentation model should be an image with the size of (H, W, 3), and the output will be a multi-class image mask with the size of (H, W).

   a) UNet-based end-to-end segmentation neural network

   UNet is a popular architecture for image segmentation. Its encoder-decoder structure, with skip connections between the encoder and decoder, helps preserve spatial information lost during

downsampling and improves segmentation accuracy. Design your own UNet, and train it on the dataset. The input to the UNet model should be the image, and the output will be the image mask.

b) Autoencoder pre-training for segmentation

An autoencoder learns to compress and reconstruct images. For this task, you should first train an autoencoder on the raw images (without labels) to reconstruct the image itself. You can then make the autoencoder's encoder fixed, use its features to train another decoder for the image segmentation task. The hypothesis is that pre-training helps the network learn relevant low-level and mid-level features, accelerating subsequent segmentation training and potentially improving accuracy.

c) CLIP features for segmentation

CLIP (Contrastive Language-Image Pre-training) is a large-scale model trained to learn a joint embedding space for images and text. While CLIP is often used for image classification or zero-shot tasks, its pre-trained image encoder can also provide powerful features for segmentation. Try CLIP features for this image segmentation task. As similar in b), the CLIP features are fixed during the segmentation network training.

d) Prompt-based segmentation

Prompt-based segmentation allows a user to provide minimal guidance (e.g., a point, bounding box, or scribble) to "prompt" the model on what region to segment. This approach has gained popularity with methods like SAM (Segment Anything Model) or other interactive segmentation frameworks. You are required to train a *point-based segmentation model*.

A possible approach is:

1) Based on the given dataset, generate new data by sampling points on images; you will have data like (image, point, ground truth mask for the prompt point).
2) After having the new dataset, pick your best performance architecture from a, b and c.
3) Modify the architecture to take an extra input of the prompt point, e.g., turning the prompt point into a heat map of the same size as the image, encode the heat map and concatenate the feature with image features. You should decide how to input the extra prompt point to the network.
4) Train the network.

For task d itself, you don't need to build any UI.

Bonus: you are encouraged to: a) implement a simple user interface to link with your trained point-based segmentation model to develop into a small segmentation application. For example, after loading an image, users can click on an object, and the network will predict the object mask, which will be sent back to the interface for result visualization. b) support other forms of prompts, e.g., scribble, bounding box, or text.

3. Evaluation and comparison

a) Quantitatively and qualitatively evaluate the segmentation performance on the testing set
   - Use metrics such as Intersection over Union (IoU), Dice coefficient (F1 score for segmentation), or Pixel Accuracy, as appropriate for the dataset.
   - The baseline's mean IoU across the three categories is 0.33, and your best-performing method is supposed to outperform the baseline.

b) Compare the performance of your first three segmentation models:

- Which architecture or approach achieves the best metrics?
- How does pre-training (with autoencoders, CLIP, etc.) influence the results?

c) For the prompt-based segmentation model, report both the visual and statistical results. If you have implemented the UI application, you must include screenshots in your report and short videos/GIFs in your supplemental material to illustrate the application.

4. **Robustness exploration**

   Compare the robustness of your best-performing network as progressively stronger perturbations are applied to the test image. Report the results of the evaluation of robustness.

Dataset

You will be provided with a processed version of The Oxford-IIIT Pet Dataset,the data and documentation can be found at here, the test set is provided and **must not be used** for model training.

Robustness Exploration Details

Choose your best-performing segmentation model for the robustness evaluation experiments. The central idea is that a variety of perturbations will be applied to the images, and then the segmentation accuracy (Dice score) will be re-calculated. You will produce a plot of segmentation accuracy *versus* amount of perturbation for each of the perturbations. Don't retrain the model, just evaluate its performance on the perturbed data.

The evaluation metric is mean Dice accuracy on the test set. Evaluation will use increasing levels of perturbation applied to the test set. The plots will show the change in accuracy with respect to the amount of perturbation. In total, there are 8 perturbations resulting in 8 plots. Show examples of each perturbation in your report.

a) Gaussian pixel noise

To each pixel, add a Gaussian distributed random number with 10 increasing standard deviations from {0, 2, 4, 6, 8, 10, 12, 14, 16, 18 }. Make sure that the pixel values are integers in the range 0..255 (e.g. replace negative numbers by 0, values > 255 by 255).

b) Gaussian blurring

Create test images by blurring the original image by 3x3 mask:

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Repeatedly convolve the image with the mask 0 times, 1 time, 2 times, ... 9 times. This approximates Gaussian blurring with increasingly larger standard deviations.

c) Image Contrast Increase

Create test images by multiplying each pixel by { 1.0, 1.01, 1.02, 1.03, 1.04, 1.05, 1.1, 1.15, 1.20, 1.25 }. Make sure that the pixel values are integers in the range 0..255 (e.g. replace > 255 values by 255).

d) Image Contrast Decrease
Create test images by multiplying each pixel by { 1.0, 0.95, 0.90, 0.85, 0.80, 0.60, 0.40, 0.30, 0.20, 0.10}.

e) Image Brightness Increase
Create test images by adding to each pixel: { 0, 5, 10, 15, 20, 25, 30, 35, 40, 45 }. Make sure that the pixel values are integers in the range 0..255 (e.g. replace > 255 values by 255).

f) Image Brightness Decrease
Create test images by subtracting from each pixel: { 0, 5, 10, 15, 20, 25, 30, 35, 40, 45 }. Make sure that the pixel values are integers in the range 0..255 (e.g. replace < 0 values by 0).

g) Occlusion of the Image Increase
In each test image, replace a randomly placed square region of the image by black pixels with square edge length of { 0, 5, 10, 15, 20, 25, 30, 35, 40, 45 }.

h) Salt and Pepper Noise
To each test image, add salt and pepper noise of increasing strength. Essentially replace the amount in skimage.util.random_noise(...) with {0.00, 0.02, 0.04, 0.06, 0.08, 0.10, 0.12, 0.14, 0.16, 0.18}

Your Report

Each team writes and submits a single report (10 pages long excluding reference, plus code in an appendix) and necessary supplemental files (e.g., the application demo video/GIFs) that describes:

- The augmentations or preprocessing steps, describe the algorithm and implementation details
- The network design and implementation
- How you trained each of the network on the dataset.
- Provide relevant metrics (IoU, Dice, pixel accuracy, etc.) on the testing sets, summarize results in a table or chart.
- Show example segmentation outputs (especially interesting failures or successes).
- Analyze key challenges: e.g., class imbalance, small objects, etc.
- Compare the performance of different approaches, try to analyze the reason.
- The algorithms that you used to perturb the test images in each exploration.
- Plots of the segmentation performance as a function of increasing perturbation.
- If you developed a UI, include your findings/screenshots.
- As an appendix, add the code that your team wrote. Do not include the code that was downloaded from other web sites, but include a statement about what code was used and where it came from.
- *In the appendix, please provide a detailed description of each member's responsibilities. If a particular task is carried out by both members, please specify which portions each person completed.*

Other comments

The assignment is estimated to take 40 hours coding/test and 10 hours report writing per person, resulting in a 10-page report plus the code appendix.

You must do this assignment in teams of 2.

A single, joint, report is to be submitted. Split the work so that each partner can do most work independently (i.e. share the work rather than duplicate it).

Asssignment Submission

The deadline for submission is 12:00am, 03/04/2025 . Please make your submission anonymous (ie. no identifying information in the PDF). Name the submitted PDF file as: <student-number-1><studentnumber-2>.pdf. Submit your report in PDF online using Learn. Details will be circulated later.

The proportion of marks are explained in the table (and the Learn page):

Issue	Percentage
1. Clear description of algorithms used	30%
2. Clear illustration of network design and implementation	20%
3. Performance on the dataset	10%
4. Clear MATLAB or Python code	15%
5. Discussion of results	25%

Publication of Solutions

We will not publish a solution set of code. You may make public your solution but only 2 weeks after the submission date. Making the solutions public before then will create suspicions about why you made them public.

Good Scholarly Practice:

- Please remember the University requirement as regards all assessed work. Details about this can be found at: http://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct
- Furthermore, you are required to take reasonable measures to protect your assessed work from unauthorised access. For example, if you put any such work on a public repository then you must set access permissions appropriately (generally permitting access only to yourself, or your group in the case of group practicals).