Lab3: Binary Semantic Segmentation

TA 劉子齊 Jonathan

In this lab, we will explore the fascinating field of **semantic segmentation** using deep learning techniques. Specifically, we will focus on **binary semantic segmentation**, where the goal is to classify each image pixel as belonging to the foreground (object of interest) or the background.

Dataset

The **Oxford-IIIT Pet Dataset** consists of images of cats and dogs, annotated with pixel-level labels for the pet regions. We will work with this dataset to train and evaluate their segmentation models.

Tasks

1. **Data Preparation**:
    - Load and preprocess the Oxford-IIIT Pet Dataset.
    - Split the data into training, validation, and test sets.
    - Preprocess the datasets to obtain better results.
2. **Model Architecture**:
    - Design 2 encoder-decoder networks.
    - The following two architectures are required:
        1. UNet
        2. ResNet34 + UNet (ResNet34 as the encoder and Unet as the decoder)
3. **Training and Evaluation**:
    - Train the model using appropriate loss functions (e.g., cross-entropy).
    - Monitor training progress and validate on the validation set.
    - Evaluate the model's performance using **Dice Score**.
4. **Inference and Visualization**:
    - Apply the trained model to unseen images from the test set.
    - Calculate the average dice score throughout the testing dataset.
    - Visualize the segmentation masks overlaid on the original images.

Learning Outcomes

By completing this lab, we aim to:

- Gain hands-on experience with semantic segmentation tasks.
- Understand the challenges of pixel-wise classification.
- Learn to interpret segmentation results and assess model performance.

Important Date:

1. Model weight upload deadline: 4/10 (Wed.) 11:55 p.m.
2. Code & experiment report submission deadline: 4/10 (Wed.) 11:55 p.m.
3. Demo date: 4/11 (Thu.)

Model Weight Upload Form

UNet: https://forms.gle/9unhKaY2DWJM72oG8

ResNet34_UNet: https://forms.gle/16E8nkQi5oX4hdVEA

Notice: Please name your model weight as "DLP_Lab3_Net_YourStudentID_Name.pth"
e.g. "DL_Lab3_UNet_311605004_劉子齊.pth"
"DL_Lab3_ ResNet34_UNet _311605004_劉子齊.pth"

Turn in Experiment Report (.pdf) & Source code (.py)

Notice: zip all files in one file and name it "DLP_Lab3_YourStudentID_name.zip"
e.g. "DL_Lab3_311605004_劉子齊.zip"

Requirements:

1. Design your own data preprocessing method.
2. Implement the UNet & ResNet34_UNet architecture independently; **do not call the model from any library**.
3. Train your model from scratch; **do not load parameters** from any pre-trained model.
4. Compare and visualize the accuracy trend between the **two architectures**; you need to **plot each epoch's training phase and validating phase**.
5. Implement your custom training, evaluating, and inferencing code.

Dataset - Oxford-IIIT Pet Dataset:

The **Oxford-IIIT Pet Dataset** is a widely used benchmark in computer vision research, particularly for semantic segmentation and object recognition tasks. The dataset was created for fine-grained image analysis, specifically focusing on pet images (cats and dogs). It is a valuable resource for developing and evaluating models that can understand and segment objects within images.

The dataset contains 7,349 images of pets (primarily cats and dogs). Each image is annotated with pixel-level labels, indicating the boundaries of the pet regions. Annotations include binary masks, where pixels belonging to the pet are labeled as foreground, and the rest are labeled as background.

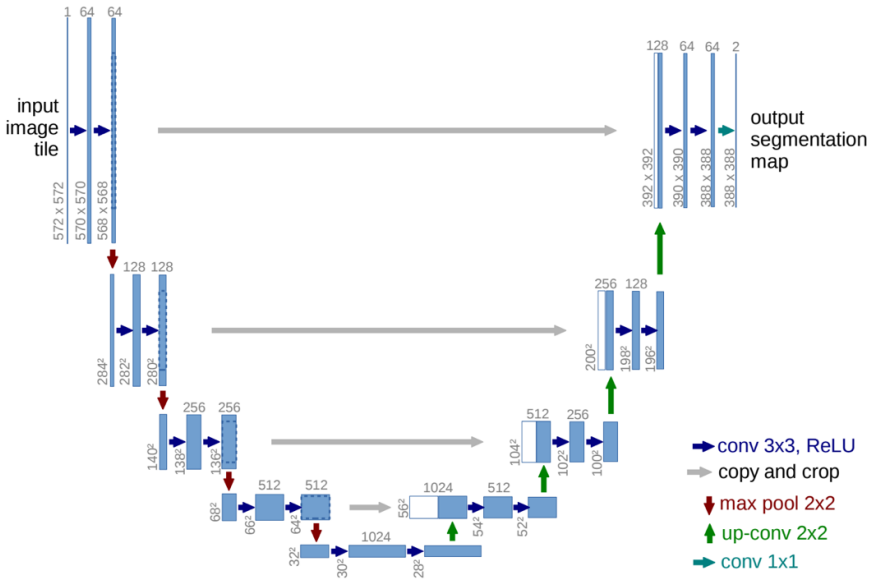
Image

GT mask

Download Link: https://www.robots.ox.ac.uk/~vgg/data/pets/

Model Architecture:

UNet



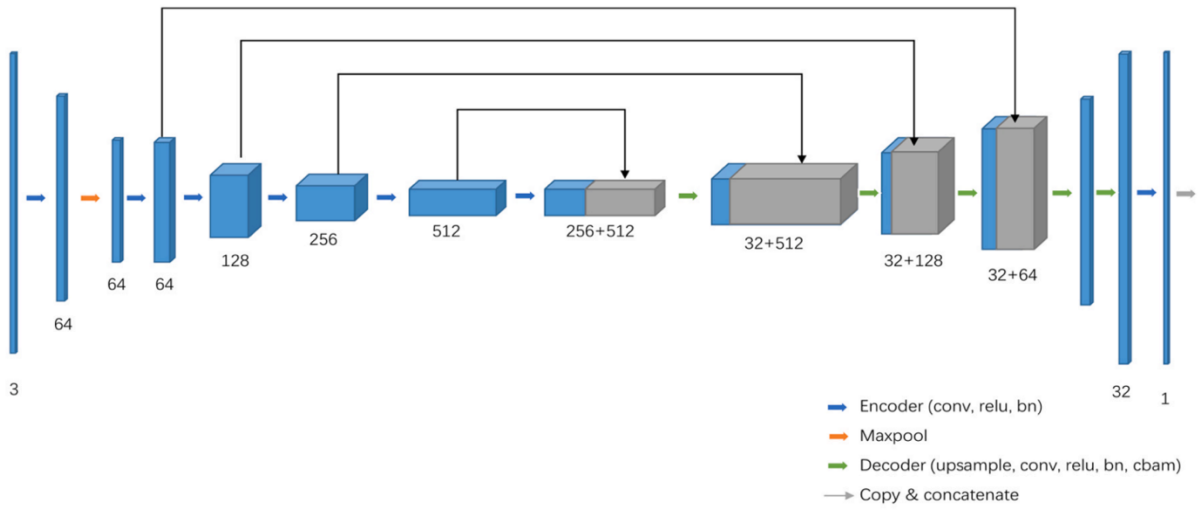
Reference: https://arxiv.org/abs/1505.04597v1

ResNet34

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Reference: https://arxiv.org/abs/1512.03385

ResNet34 (Encoder) + UNet (Decoder)



Reference: https://www.researchgate.net/publication/359463249_Deep_learning-based_pelvic_levator_hiatus_segmentation_from_ultrasound_images

Implementation Details:

File Structure

Please organize your project files using the following directory structure that separates data, code, and other resources. Note that if you didn't implement lab 3 in the following file structure, you would get 0 points in lab 3.

```
├── dataset/
│   └── oxford-iiit-pet/
├── src/
│   ├── models/
│   │   ├── unet.py
│   │   └── resnet34_unet.py
│   ├── oxford_pet.py
│   ├── utils.py
│   ├── train.py
│   ├── evaluate.py
│   └── inference.py
└── saved_models/
```

dataset/
This directory contains data related to the **Oxford-IIIT Pet Dataset**. Inside dataset/, there is a subdirectory named oxford-iiit-pet/. The dataset files (images, annotations, etc.) are likely stored within this subdirectory. We access and manipulate the dataset from here.

src/
The src/ directory holds the source code for your project. It contains various Python files responsible for different functionalities. Let's explore the contents of src/:

- **models/**: This subdirectory likely contains implementations of different neural network architectures. Specifically, there are two files:
  - unet.py: Contains the UNet architecture implementation.
  - resnet34_unet.py: Combines the ResNet-34 backbone with the UNet architecture.

- **oxford_pet.py**: This file contains code for handling the Oxford-IIIT Pet Dataset. It includes data loading, preprocessing, and other dataset-related functions.

- **utils.py**: The utils.py file typically contains utility functions used across the project. These functions might include visualization tools, metrics computation, or other common tasks.

- **train.py**: This file contains code for training the neural network models. It includes functions related to model training, optimization, and backpropagation.

- **evaluate.py**: This file probably handles model evaluation. It includes functions to assess model performance on validation.

- **inference.py**: This file deals with model inference (making predictions) on unseen data. It includes functions to apply the trained model to new images.

saved_models/
The saved_models/ directory is where you would store trained model checkpoints. After training, the best-performing model weights are saved here for later use. Please save the trained model weights in the ".pth" files.

Dice Score

The Dice score, also called the Dice Similarity Coefficient, serves as a metric for assessing the similarity between two sets of data, often represented as binary arrays. The Dice score quantifies how closely a predicted segmentation mask aligns with the ground truth segmentation mask in image segmentation. Its range spans from 0 (indicating no overlap) to 1 (representing perfect alignment).

Mathematically, the Dice score is computed as:

Dice score = 2 * (number of common pixels) / (predicted img size + groud truth img size)

In simpler terms, the Dice score corresponds to twice the size of the intersection divided by the sum of the sizes of the two sets. It measures the proportion of overlap normalized by the overall set sizes.

Reference: https://oecd.ai/en/catalogue/metrics/dice-score

Report Spec.

The report should strictly follow the structure of the topics, or there will be a penalty (-10 pts) on the final score. However, you can freely modify the subtopics.

1. Overview of your lab 3 (10%)

2. Implementation Details (30%)
    A. Details of your training, evaluating, inferencing code
    B. Details of your model (UNet & ResNet34_UNet)
    C. Anything more you want to mention

3. Data Preprocessing (20%)
    A. How you preprocessed your data?

   B. What makes your method unique?
   C. Anything more you want to mention

4. Analyze on the experiment results (20%)
   A. What did you explore during the training process?
   B. Found any characteristics of the data?
   C. Anything more you want to mention

5. Execution command (0%)
   A. The command and parameters for the training process
   B. The command and parameters for the inference process

6. Discussion (20%)
   A. What architecture may bring better results?
   B. What are the potential research topics in this task?
   C. Anything more you want to mention

Scoring

Final score = Experimental results (30%) + Report (30%) + Demo score (40%)

Experimental results (30%)

For each network architecture (15%)

- 100 pts: Dice score ≥ 0.87
- 80 pts: $0.87 >$ Dice score ≥ 0.8
- 60 pts: $0.8 >$ Dice score ≥ 0.7
- 0 pts: $0.7 >$ Dice score

If the zip filename or the report has a format error, it will be a penalty (-10 pts).

In the demo phase, you need to load the trained model and show the results of the two models. If you fail to do so, you will get 0 pts on the model that you couldn't successfully demonstrate.

Please only train & validate your model on the training & validating dataset split by the provided class "SimpleOxfordPetDataset" in the sample code. If the TA found your model trained on the testing dataset through retraining your model, you'll get 0 pts in this lab.

Please ensure the TA can execute the code and the model weight you uploaded. If the TA fails to do so, you'll get 0 pts in this lab.

If you're found faking your model predicting results, you'll get 0 pts in this lab. Also, we will take disciplinary action under school regulations.

Plagiarism is strictly prohibited. If you're found guilty of plagiarizing another's code, you and the person/people you plagiarized will get 0 pts in this lab. Also, we will take disciplinary action under school regulations.

Always cite the resources you applied in this lab in your report. Please cite properly to avoid your lab results being considered plagiarism.

If any issues are identified in your submitted files for this lab, the TA will reach out to you via email. Please monitor your inbox closely. Suppose the TA has not received your response within three days of sending the email. In that case, it will be considered that you have forfeited the opportunity to provide further clarification and have accepted the corresponding penalty or consequences.