Lab2: 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.

Important Date:

Code & report submission deadline: 7/22 (Tue.) 23:59

Submissions made after the deadline but before **8/28 (Thu.) 23:59** will still be accepted, but the final score will be multiplied by 0.7.

Submissions after **8/28 23:59** will **not** be accepted and will receive a grade of 0.

Submission Instructions

You must submit the following **two separate files**:

1. **Experiment Report (.pdf)**
   - Filename: DL_Lab2_YourStudentID_name.pdf
   - Example: DL_Lab2_311605004_劉子齊.pdf

2. **Source Code (.zip)**
   - Compress all your source code into a single zip file.
   - Filename: DL_Lab2_YourStudentID_name.zip
   - Example: DL_Lab2_311605004_劉子齊.zip

Important Notes:

- You must submit both files; otherwise, your lab will be graded as **0 points**.
- If the filename does not follow the specified naming rule, there will be a **10-point penalty**.
- If the TA is unable to unzip your code or open your PDF, it will be considered a missing submission and graded as **0 points**.

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.

The bounding box annotations provided in the dataset are for localization purposes only and should **not** be included as part of the foreground region in your segmentation task. Please strictly use the pixel-level segmentation mask (trimap) to define the foreground (pet region), and treat all other pixels as background.

Image

GT mask

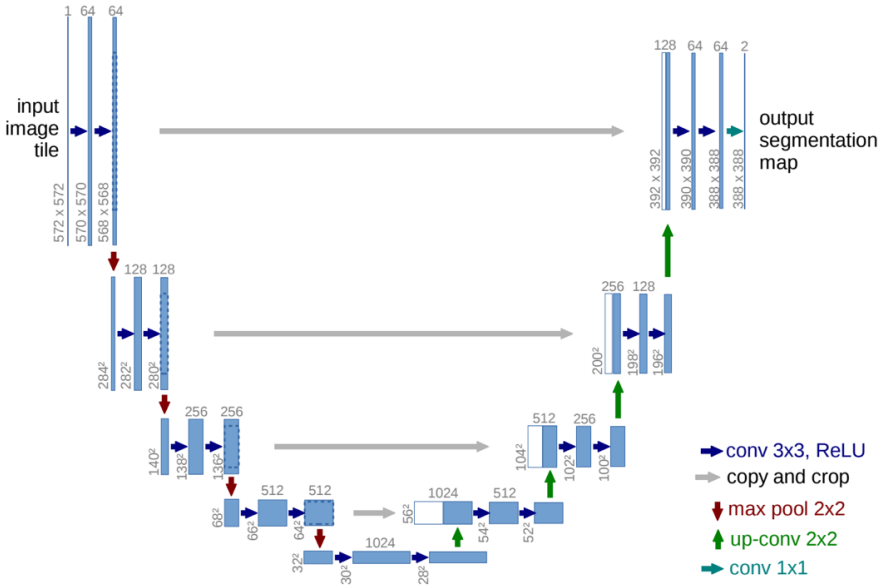
Download Link: https://www.robots.ox.ac.uk/~vgg/data/pets/

Dataset Usage Guidelines

- It is essential to keep the training, validation, and testing datasets completely separate.
- The **training and validation** sets should only be used for hyperparameter tuning and model evaluation during the training process.
- The **testing** set is reserved solely for final inference and must not be utilized at any point during training or validation.
- Please only train & validate your model on the training & validating dataset split by the provided class “SimpleOxfordPetDataset” in the sample code. TAs will retrain your models and review your code. If any use of the testing set is found during the training or validation phases, you will receive **0 points** for this lab.

Model Architecture:

UNet



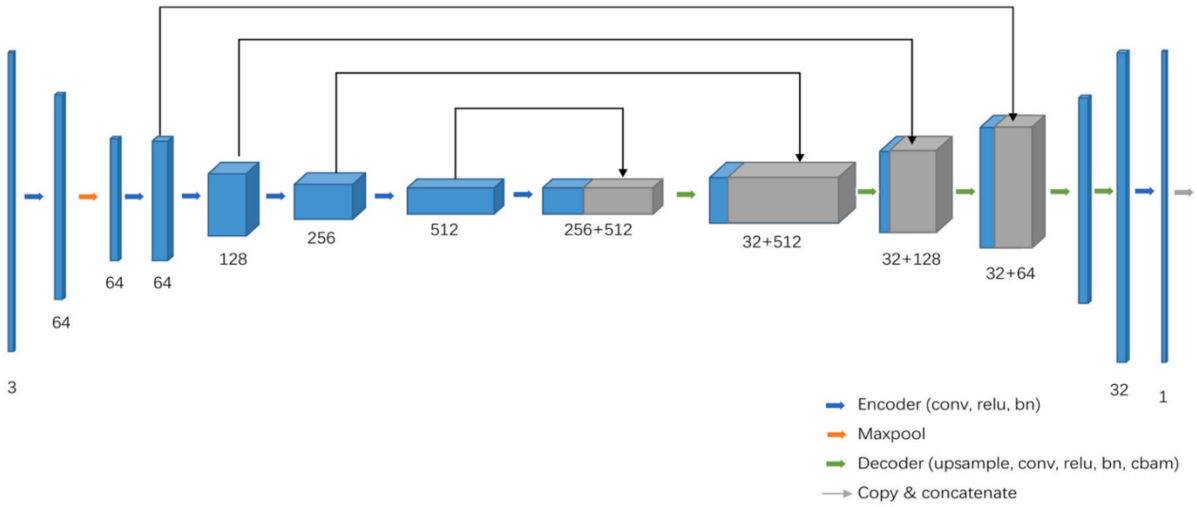
Reference: https://arxiv.org/abs/1505.04597v1

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 2 in the following file structure, you would get 0 points in Lab 2.

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

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.

requirements.txt
Please include a requirements.txt file in your submission. This file should list all the dependencies required to rebuild the training environment for retraining your code. This ensures that the Teaching Assistants can successfully retrain your model using the provided code.

Inference Code Organization Guidelines

- All testing (inference) logic must be placed in the inference.py file.

- Do not include any test-time inference code in train.py or evaluate.py.
- The TAs will run your code starting from inference.py for testing purposes; if your inference logic is missing or scattered in other files, it will be considered incomplete, and you will receive 0 points for the testing phase.
- This requirement is in place to ensure that your code is well organized and reproducible.

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 + ground 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

Dice Score Calculation Guidelines

- The Dice score for each model must be evaluated using the entire **testing dataset**.
- You are required to run inference on every image in the test set and then compute the average Dice score across all images.
- Do not report the Dice score from only the best-performing sample as the model's overall Dice score.
- The **average Dice score for the entire testing set** will serve as the final metric for grading your experimental results.

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. Implementation Details (30%)
Please describe the details of your training, evaluation, and inference code. Explain how the different components of your implementation work together and highlight any important design choices. Additionally, provide a detailed explanation of your models, specifically UNet and ResNet34_UNet, including architectural modifications, training settings, and optimization strategies. If there are any other relevant details that you find important, you are encouraged to include them in this section.

2. Data Preprocessing (25%)
Please explain how you preprocessed your data, including any specific techniques applied to clean, augment, or transform the data before feeding it into the model. Discuss what makes your preprocessing method unique compared to standard approaches. If there are additional insights or methods you wish to mention, please include them in this section.

3. Analyze the experiment results (25%)
In this section, please discuss your exploration during the training process, including different hyperparameters, model configurations, and training strategies you experimented with. Highlight any significant observations you made regarding the characteristics of the dataset and how those findings influenced your approach. If there are any other relevant insights from your analysis, please mention them here.

Additionally, you must explicitly report the **average Dice score** on the testing dataset for both of your models (UNet and ResNet34_UNet). Please include clear screenshots showing these inference results as evidence.

4. Execution steps (required, 0%)
Please provide the exact commands and parameters you used for training your models. Additionally, include the commands and parameters required to perform inference using your trained models. This section does not contribute to the final score, but it ensures the reproducibility of your results.

To ensure the TAs can reproduce your results accurately, the TAs will be retraining your models. Please provide the seed value you used to achieve the best results. This will help maintain consistency and reproducibility of the results.

5. Discussion (20%)
Please discuss what alternative architectures could potentially yield better results for this task. Provide reasoning for your choices based on your understanding of the model and dataset. Additionally, identify potential research directions related to this task and explain why these topics are worth exploring. Any other reflections or discussions that you find relevant can also be included in this section.

6. Demo Video Link (required, 0%)

Please provide a working link to your demo video (e.g., Google Drive, YouTube, or any other accessible platform). Make sure the permissions are set so that the TAs can view the video without needing to request access. If you do not provide a valid link, you will receive 0 points for this lab.

Scoring

Final score = Experimental results (30%) + Report (70%)

Please note that I will randomly select students to perform a demo. This demo may result in an additional score adjustment of up to 30 points, either added or deducted.

We will notify selected students via email, so please check your inbox regularly. If a selected student does not respond within 3 days or fails to participate in the demo as scheduled, Lab 2 will be graded as zero.

Experimental results (30%)

For each network architecture (15%)

- 100 pts: Dice score ≥ 0.925
- 80 pts: $0.925 >$ Dice score ≥ 0.875
- 60 pts: $0.875 >$ Dice score ≥ 0.85
- 0 pts: $0.85 >$ Dice score

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

If you're randomly picked to attend the demo session, you need to load the trained model and show the results of the two models in the demo phase. If you fail to do so, you will get 0 points on the model that you couldn't successfully demonstrate.

Please ensure the TA can successfully retrain your model using the uploaded code. If the TA is unable to do so, you will receive 0 points for this lab.

If you do not upload a requirements.txt file, or if the TA cannot successfully rebuild the environment using the provided file, you will receive 0 points for this lab.

If you're found faking your model predicting results, you'll get 0 points 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 points 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 emailing. 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.

Academic Honesty and Use of AI Tools

- If you choose to use any AI language models (LLMs), such as ChatGPT, Copilot, or other generative tools to assist with writing your report or code, you must clearly disclose this in your submission. Failure to disclose the use of these tools will result in a score of 0 points for the lab.

- Additionally, even if you do disclose your use of AI assistance, if your submission is determined to be primarily generated by AI (i.e., showing little to no original contribution or understanding), you will still receive 0 points for the lab.

- All students are responsible for upholding academic honesty and must ensure that their work reflects their own understanding and effort.