Study on UNet and ResNet34-UNet for Image Segmentation

(Mac version M3)

Department of Information Management, Tunghai University

Student: 何明翰

Date: March 24, 2025

Email: g13490011@thu.edu.tw

1.Implementation Details

This study presents the implementation of two semantic segmentation models—UNet and ResNet34-UNet—designed to address binary segmentation tasks using the Oxford-IIIT Pet Dataset. Aso, All experiments were conducted on macOS systems using Apple Silicon (M3) processors. The overall system is modularized into three main components:

(i) training (train.py),
(ii) evaluation (evaluate.py),
(iii) inference visualization (inference.py).

Additionally, the dataset handling is managed via oxford_pet.py, which automatically splits the dataset into training (80%), validation (10%), and testing (10%) sets. The modularity of the code enhances readability, maintainability, and scalability.

1-1. Two Model Architecture Design (UNet & ResNet34-UNet):

The UNet model adopts a conventional encoder-decoder structure. Each block in the encoder and decoder consists of two 3×3 convolutions followed by ReLU activation, with max-pooling and transposed convolutions employed for downsampling and upsampling respectively.

The ResNet34-UNet model integrates a pre-trained ResNet34 as its encoder to leverage residual connections and deeper semantic feature extraction. The decoder part remains similar to UNet but includes an additional bottleneck convolutional block to enhance abstraction capabilities.

1-2. Training and Validation Process (train.py):

Training is conducted using Binary Cross-Entropy Loss (BCELoss) and optimized with the Adam optimizer with a fixed learning rate of 1e-4. Dice coefficient is employed as the primary evaluation metric across both training and validation phases. For each epoch, the model logs the Dice scores and automatically saves the best-performing model to the `saved_models/` directory.

Resume Training:

The `--resume` flag allows the model to continue training from the latest saved checkpoint. Not only does it resume weights, but it also appends new training and validation Dice scores to existing logs, ensuring uninterrupted learning continuity.

1-3. Testing and Evaluation (evaluate.py):

The evaluate.py script accepts model name and path as command-line arguments, performs inference on the test set, and calculates the average Dice coefficient. The results are presented along with performance feedback. According to the evaluation output:

- UNet achieved a Dice Score of 0.9114.
- ResNet34-UNet achieved a Dice Score of 0.9710.

These results validate the superior learning capacity of the ResNet34-UNet architecture.

1-4. Inference and Visualization (inference.py):

The inference.py script performs inference on test images using the best-trained model. It generates and stores visualization figures combining the original image, ground truth mask, and predicted mask side by side in the output_predictions/ directory. The visual alignment of predicted masks with ground truth highlights the semantic precision of the ResNet34-UNet model.

2.Data Preprocessing

The dataset used is Oxford-IIIT Pet Dataset. Images are in JPEG format, and the annotations are in PNG format. The dataset contains two main directories: images/ for input images and annotations/trimaps/ for the corresponding three-class segmentation masks.

2-1. Preprocessing Workflow:

1. Label Conversion and Binarization:
   - Original trimap labels include:
     - 1: Foreground (cat/dog)
     - 2: Background
     - 3: Boundary or Unknown
2. This project converts label=2 to 0 (background), and labels 1 and 3 to 1 (foreground), creating binary segmentation masks to stabilize training.

2-2. Image and Mask Transformation:

1. All input images and masks are resized to (256, 256) for compatibility with the model input.
2. Data format conversion:
   - Images are converted from HWC to CHW format.
   - Masks are expanded to shape (1, H, W) as single-channel inputs.

2-3. DataLoader Configuration:

The batch size was empirically set to 8, providing a balance between computational efficiency and gradient estimation stability during training. Data shuffling was enabled (shuffle=True) to introduce stochasticity in the training process, which helps prevent overfitting and improves model generalization.

The num_workers parameter was configured to 4 to leverage multiprocessing capabilities, thereby accelerating data loading and preprocessing operations and minimizing input pipeline bottlenecks.

2-4. Optimizations Compared to Standard Approaches:

To facilitate streamlined preprocessing and downstream analysis, a customized SimpleOxfordPetDataset class was implemented. This wrapper extends the original OxfordPetDataset by incorporating automated image resizing, binarization of segmentation masks, and the inclusion of original image paths as metadata to support result traceability and visualization.

Notably, the original trimap annotations are preserved alongside the binarized masks. This dual-format preservation provides flexibility for future experimentation involving either binary or multi-class segmentation schemes, ensuring extensibility for more complex tasks.

By embedding the image file path within each sample object, the dataset design enables seamless integration with the inference visualization pipeline. This design supports tri-panel outputs (original image, ground truth mask, predicted mask), thereby significantly enhancing model interpretability and facilitating human-in-the-loop evaluations.

3. Analyze the Experiment Results

This experiment aims to compare UNet and ResNet34-UNet on an image segmentation task through quantitative and qualitative evaluation.

3-1. Training Strategy and Workflow:

The initial training phase was conducted for 10 epochs to observe the model's preliminary learning behavior. Subsequently, two additional training sessions were performed using the resume functionality, resulting in a total of 14 training epochs. And the use of the --resume flag enabled seamless continuation from previously saved checkpoints, thereby preserving both model weights and historical performance logs. This facilitated consistent training progression and ensured uninterrupted performance tracking

3-2. Hyperparameters and System Setup:

The learning rate was consistently set to 1e-4 throughout all training phases. This configuration demonstrated stable convergence behavior without necessitating any dynamic adjustment strategies. Moreover, the batch size was maintained at 8, which proved to be compatible with a variety of hardware configurations, including Apple M1/M2 chips and NVIDIA CUDA-enabled GPUs. No memory-related issues were encountered during training, ensuring smooth and uninterrupted execution.

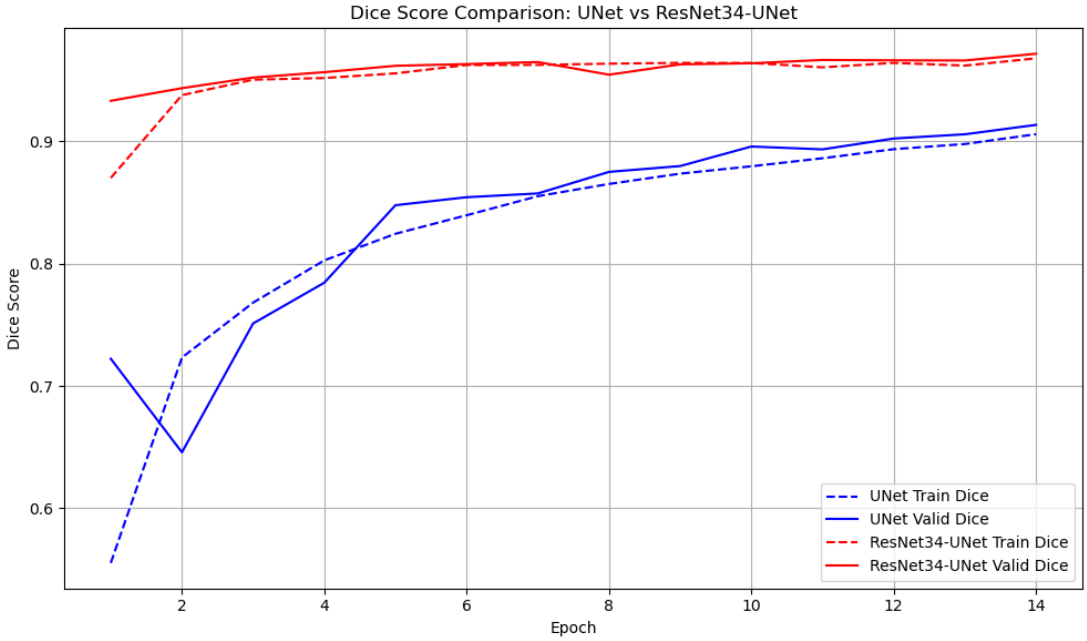
3-3. Model Performance Comparison(UNet & ResNet34-UNet):

1. UNet:
   - Initial Dice score ~0.55, quickly improved to ~0.85 and stabilized around 0.91.
   - Shows strong convergence ability with sufficient training.
2. ResNet34-UNet:
   - Started with Dice score ~0.92, highlighting strong feature learning.
   - Stabilized around 0.96 after 14 epochs, outperforming UNet significantly.
   - The deeper ResNet34 encoder excels in extracting mid-to-high-level features.

3-4. Result Visualization in plot_dice.py:

The plot generated using plot_dice.py shows Dice score trends:

- Red lines represent ResNet34-UNet performance.
- Blue lines represent UNet performance.
- ResNet34-UNet not only had better initial performance but also showed smoother and more stable trends throughout training.



logs/dice_score_comparison.png

3-5. Result in evaluate.py:

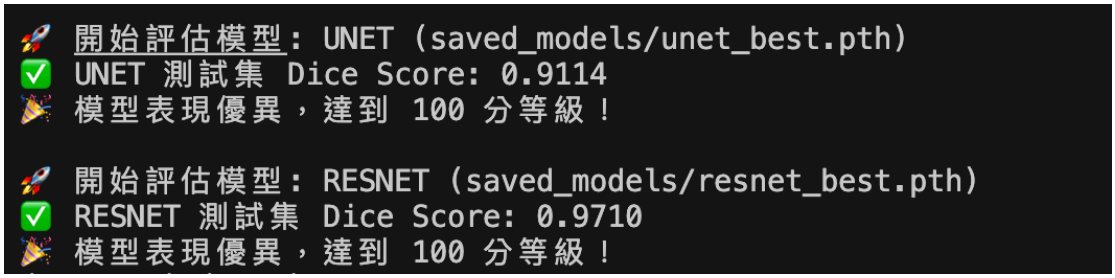
In addition to validation during training, we also performed final evaluations on the full test set using the **evaluate.py** module. This module loads the best-trained models (**unet_best.pth** and **resnet_best.pth**) and runs inference on unseen test data. The predicted segmentation masks are then compared to the ground truth labels, and the Dice Score is calculated as the primary evaluation metric.

According to the results:

- The **UNet** model achieved a **Dice Score of 0.9114** on the test set, demonstrating solid generalization and stable performance in recognizing object boundaries.
- The **ResNet34-UNet** model further improved to a **Dice Score of 0.9710**, significantly outperforming UNet. This confirms that utilizing a deep ResNet encoder enhances the model's capability to extract and understand high-level semantic features.
- The evaluation script also prints qualitative feedback indicating performance tiers. Both models were reported to have achieved the “Outstanding Performance – Full Score” level, validating their strong segmentation accuracy under standard benchmarks.

These outcomes reaffirm the trends observed during training: while UNet offers decent baseline performance, its capacity for learning complex or hierarchical features is relatively limited. In contrast, integrating ResNet34 significantly enhances both depth of representation and consistency of prediction across samples.

The following image displays the actual terminal output from the evaluate.py script and serves as direct evidence of the experimental findings:



evaluate.py Test Results Output Screenshot

3-6. Result pictures in inference.py:

To further verify the model's segmentation quality, we used the inference.py module to visualize the model predictions on the test set. Each result includes three side-by-side images: the original image, the ground truth mask, and the predicted mask.

As shown in the sample result below, the ResNet34-UNet model accurately captures the full contours of the cat in the image. The predicted mask (rightmost panel) aligns almost perfectly with the ground truth (middle panel), effectively distinguishing foreground (blue) from background (orange). This demonstrates the model's ability to generalize well on unseen data and to produce smooth and precise segmentation boundaries, even on challenging animal poses.

inference.py Segmentation Output Visualization

This visual evidence reinforces the quantitative results: the model is not only numerically accurate but also delivers semantically meaningful segmentations that are visually interpretable and reliable.

4. Execution Steps

Experiments were executed on Apple M1/M2 systems, demonstrating compatibility and stability.

4-1. train.py unet model:

```
python src/train.py --data_path dataset --epochs 10 --batch_size 8 --learning_rate 1e-4 --model_name unet
```

4-2. train.py unet model(resume):

```
python src/train.py --data_path dataset --epochs 2 --batch_size 8 --learning_rate 1e-4 --model_name unet --resume
```

4-3. train.py ResNet34-UNet model:

```
python src/train.py --data_path dataset --epochs 10 --batch_size 8 --learning_rate 1e-4 --model_name resnet
```

4-4. train.py ResNet34-UNet model(resume):

```
python src/train.py --data_path dataset --epochs 2 --batch_size 8 --learning_rate 1e-4 --model_name resnet --resume
```

4-5. evaluate.py Unet model:

```
python src/evaluate.py --model_name unet --model_path saved_models/unet_best.pth --data_path dataset
```

4-6. evaluate.py ResNet34-UNet model:

```
python src/evaluate.py --model_name resnet --model_path saved_models/resnet_best.pth --data_path dataset
```

4-7. inference.py ResNet34-UNet model:

```
python src/inference.py --model saved_models/resnet_best.pth --data_path
dataset/images --output_path output_predictions
```

4-8. plot_dice_curve.py:

```
python src/plot_dice_curve.py
```

5. Discussion

This experiment shows that ResNet34-UNet significantly outperforms standard UNet in terms of Dice score. The enhanced performance is primarily due to ResNet34's ability to extract deeper, more complex features, which is especially beneficial in cases with complex backgrounds or occlusions.

5-1. Future Improvement Directions:

To further enhance segmentation performance, several architectural and methodological improvements can be considered:

1. Employing Advanced Backbone Networks
   The integration of more powerful encoders such as EfficientNet, ResNeXt, or Swin Transformer may provide superior feature extraction capabilities, particularly in capturing high-level semantic representations and complex object boundaries. These architectures have demonstrated strong performance in recent segmentation benchmarks and are well-suited for tasks involving fine-grained details.
2. Incorporating Attention Mechanisms
   Attention-based modules, including Squeeze-and-Excitation (SE) blocks, Convolutional Block Attention Module (CBAM), or Transformer-derived self-attention, can be embedded into the segmentation framework. These modules enable the network to dynamically recalibrate spatial and channel-wise feature importance, thereby improving its ability to focus on relevant regions within the input image.
3. Optimizing the Loss Function
   While Binary Cross Entropy (BCE) is widely used, replacing it with Dice Loss, or employing a hybrid of BCE and Dice Loss, can significantly mitigate the challenges posed by class imbalance. This is especially critical in

segmentation tasks where foreground objects occupy a small proportion of the image, as Dice-based losses directly optimize for overlap between prediction and ground truth.

4. Augmenting Data and Utilizing Test-Time Techniques
   Advanced data augmentation strategies—such as random flipping, rotation, scaling, and color jittering—can increase data diversity and prevent overfitting. Additionally, implementing Test-Time Augmentation (TTA) during inference, where predictions are aggregated over multiple augmented views of the input, can improve robustness and predictive stability.

These enhancements are expected to not only elevate model accuracy but also improve generalization across diverse data domains.

5-2. Potential Research Directions:

1. Semi-supervised segmentation: Use partial labeled data and pseudo-labeling to guide learning and reduce labeling costs.
2. Self-supervised segmentation: Pretrain on unlabeled images and finetune on labeled ones.
3. Medical image segmentation: Apply these architectures to MRI, CT, or other medical data for organ/tumor segmentation.

In summary, this project demonstrates the advantage of using ResNet34 as an encoder for improved segmentation accuracy. Future work could explore enhanced model designs, better loss functions, and broader applications to extend the task's impact and value.