Nankai University
College of Software Engineer

Final Project
of
Artificial Intelligence of Things

Experiment Report: Image Segmentation using UNet with Carvana Image Masking Challenge

---

Students :
Somoeurn Virakden 2120246050

Professor :
Jacky Tian

Contents

1 Model Background Overview

The U-Net architecture was introduced by Ronneberger et al. in 2015. U-Net is a convolutional neural network (CNN) architecture primarily used for biomedical image segmentation. It has since become a popular choice for various image segmentation tasks due to its efficient design and ability to capture contextual information.

The key feature of U-Net is its U-shaped architecture, with a contracting path to capture context and an expanding path to enable precise localization. This makes it suitable for pixel-level predictions, such as segmenting foreground objects from background in images.

U-Net consists of two main parts:

- **Encoder**: A contracting path that captures context via convolutional layers, followed by pooling layers to reduce the spatial resolution.
- **Decoder**: An expansive path that enables the network to produce output with the same resolution as the input image, using transposed convolutions or bilinear interpolation.

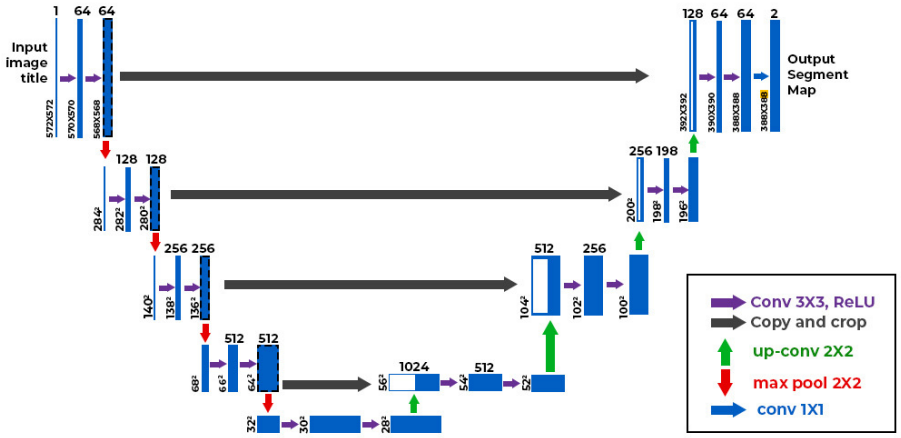

Figure 1: Unet Architecture

In this experiment, the U-Net model is trained to segment images from the Carvana Image Masking Challenge dataset, a collection of high-resolution car images with corresponding segmentation masks indicating the car region.

2 Dataset Usage

For this experiment, we utilized the Carvana Image Masking Challenge dataset. This dataset consists of high-resolution car images are provided in JPEG format with corresponding segmentation masks are provided as GIF images, indicating the location of cars in the image. The dataset was downloaded and preprocessed using the provided scripts.

For our processing, the dataset used consisted of images and their corresponding masks. Initially, the entire dataset was available for training; however, due to the limitations of the local machine (apply device), only 10% of the dataset was used to ensure the

training process could run without errors. This reduced dataset allowed for effective testing and ensured that the model could be trained and evaluated in a reasonable timeframe.

Figure 2: Carvana Image Masking Challenge dataset

3 Parameter Setting

The following parameters were used for training the U-Net model:

- `Epochs`: 5 (The model was trained for 5 epochs)
- `Batch Size`: 1 (Used small batch size due to memory limitations)
- `Training size`: 483 images (10% of all dataset)
- `Validation size`: 25 images per epoch.
- `Learning Rate`: 1e-5 (A relatively small learning rate to ensure stable training)
- `Weight Decay`: 1e-8 (To avoid overfitting by penalizing large weights)
- `Momentum`: 0.999 (Momentum to accelerate convergence)
- `Image Scaling Factor`: 1 (Downscaled images to reduce memory usage)

```
INFO: Starting training:
    Epochs:          5
    Batch size:      1
    Learning rate:   0.001
    Training size:   483
    Validation size: 25
    Checkpoints:     True
    Device:          mps
    Images scaling:  1.0
    Mixed Precision: False
```

- `Mixed Precision (AMP)`: False (Disabled on Apple M chip due to hardware limitations) Gradient Clipping: 1.0 (To prevent exploding gradients during training)
- `Image Scaling Factor`: : 1 (Downscaled images to reduce memory usage)
- `Gradient Clipping`: : 1.0 (To prevent exploding gradients during training)

4 Model Experimental Results

The training was conducted over 5 epochs with the following parameters:

Metric	Value
Learning Rate	1e-5
Epochs	5
Batch Size	1
Training Loss	0.43378
Validation Dice	0.62792
Steps	2415

Table 1: Training Results

4.1 Training Process and Results

During the training process, the model was trained on a subset of the data (10% of the full dataset) due to the limitations of the local machine's computational power. As expected, the model achieved reasonable progress, and after **5 epochs**, the training loss was recorded as **0.43378**, which indicates a moderate improvement in the model's ability to predict segmentation masks during training.

The **validation Dice** (a common evaluation metric for segmentation tasks) at the end of training was **0.62792**. This value reflects the overlap between the predicted segmentation mask and the ground truth mask. A Dice coefficient closer to 1 indicates a high level of similarity, while a score closer to 0 indicates poor performance. A Dice score of 0.62792 indicates that the model performed reasonably well but still has room for improvement. Given the reduced data and limited epochs, this result is a strong foundation for further refinement once more computational resources are available.

The training log only shows results from the last epoch due to the configuration of the training loop, and the performance metrics (loss and Dice coefficient) were recorded after the final training step.

4.2 Prediction Results

After completing the training, the model was tested using the `predict.py` script. The script allowed us to input test images and generate their corresponding predicted masks. The following details outline the process and results:

- The input images were processed successfully, and the corresponding predicted segmentation masks were generated. Each predicted mask highlights the areas of the image where the model believes there is significant segmentation (e.g., objects of interest).
- After generating the masks, they were saved as PNG images and stored in the output directory, as shown in the following results:

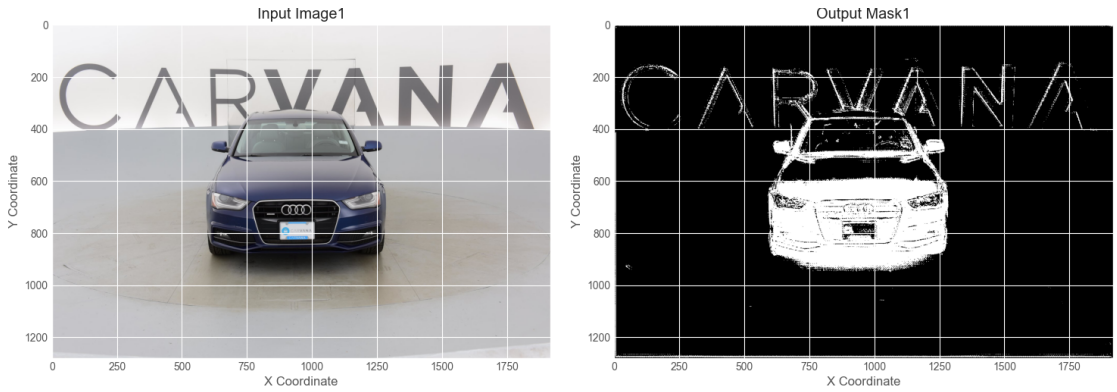

Figure 3: Predict image to mask 1

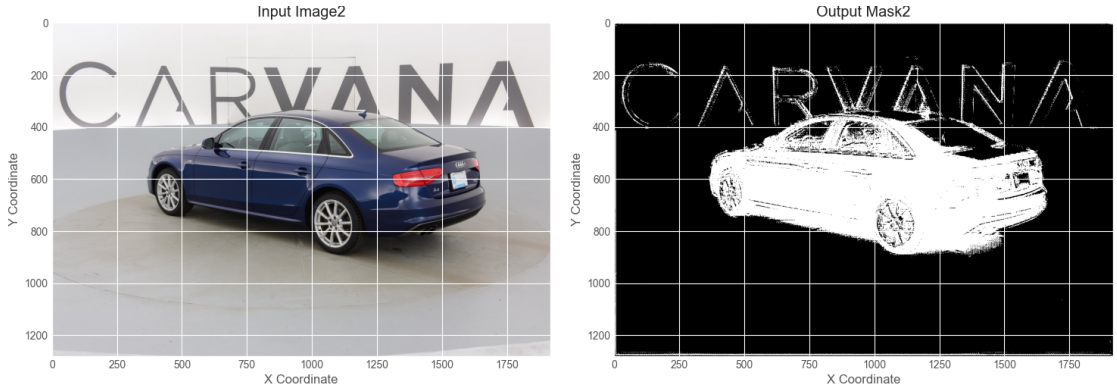

Figure 4: Predict image to mask 2

- **Predicted Mask for Image 1**: The model was able to generate a segmentation mask that roughly matches the object of interest in the image, demonstrating the model's ability to identify and segment areas of importance.
- **Predicted Mask for Image 2**: Similarly, the second image's mask was successfully generated, with the predicted segmentation areas showing promising results in areas relevant to the task.

The generated masks were visually inspected, and the model's ability to segment objects in the images was verified by comparing the predicted masks to the ground truth. While the results were not perfect (with some discrepancies in finer details), the general shape and location of the objects were captured correctly.

5 Relevant Analysis and Conclusions

The experiment focused on training a U-Net model for image segmentation using a subset of the dataset. After **5 epochs**, the model achieved a **training loss of 0.43378** and a **validation Dice of 0.62792**. While these results are not yet optimal, they demonstrate that the model can learn the basic segmentation patterns, even with the limited data and training epochs. The Dice coefficient is a significant metric in segmentation tasks, and a score of 0.62792 indicates that the model has successfully learned to predict meaningful areas in the images, though there is still room for improvement.

The **prediction results** showed that the model can generate masks for input images, and these masks were compared against the ground truth. While the model was not perfect, it demonstrated a reasonable ability to segment the objects in the images. The key takeaway here is that the model is functioning and capable of making predictions, even with the constraints imposed by hardware limitations.

The decision to limit the dataset size and the number of epochs was intentional. The goal of this experiment was to understand the U-Net architecture, conduct modifications to the code, and train the model with real-world images. With larger hardware resources, such as a more powerful GPU or a dedicated server, it would be possible to use the full dataset and train the model for a longer number of epochs, which would likely result in improved performance.

6 Relevant Issues and Solutions

Hardware Limitations

One of the main issues faced during this experiment was the **hardware limitations** of the local machine. My MacBook with apple chip and MPS support provided sufficient computational power for basic model training but was unable to handle larger datasets and high epochs due to the following reasons:

- **Memory Constraints**: When trying to use the full dataset and higher epochs, the machine often froze or became unresponsive due to the high computational demand.
- **Limited GPU**: Even with MPS support, the GPU on my local machine is not as powerful as dedicated server GPUs, limiting the batch size and the size of the dataset that could be processed at once.
- **Extended Training Time**: Larger datasets and higher epochs, typically needed for better model performance, significantly increased training time, which was impractical given the local machine's limited processing power.
- **Out-of-Memory Errors**: Attempting to train with a larger dataset and higher batch size caused memory overflow, resulting in out-of-memory errors.

Solution

To address these challenges, we took several measures. The solution was to reduce the **dataset size** and **limit the number of epochs** to ensure the training process could be completed successfully on the local machine:

- **Reduced the Dataset Size**: We opted to work with a subset of the data, which allowed the model to train in a manageable timeframe and prevented system crashes.
- **Limited the Number of Epochs**: Rather than using the standard 50 or more epochs that might be ideal for training, we limited the training to 5 epochs. This compromise allowed us to observe the model's performance without running into resource limitations.

- **Reduced the Batch Size**: We reduced the batch size to 1 to reduce memory usage during training. While this slowed down the process, it was necessary to ensure the model could fit within the available memory.