RBE 474X
Project 1

Colin Balfour
Worcester Polytechnic Institute
Worcester, Massachussetts 01609

Khang Luu
Worcester Polytechnic Institute
Worcester, Massachussetts 01609

Thinh Nguyen
Worcester Polytechnic Institute
Worcester, Massachussetts 01609

I. Part 1

A. Dataset generation

We used Blender's Python API to generate a dataset of 5000 RGB images with 640x360 pixels. Each RGB images also has a corresponding instance segmentation mask, which is a grayscale image with the same dimensions, and different grayscale value for different instances.

Windows are generated randomly, with a random number of windows placed in the image, and a random scale for each window. The windows are placed randomly in the image. The background is picked randomly from a dataset of 10000 background images, in different lighting, indoor or outdoor, etc. The instance segmentation mask is generated by assigning a unique grayscale value to each window. Obstruction is picked from 8 preset objects in blender, and placed randomly to obstruct the view of the windows.

Hyperparameters for the dataset generation: - Min and Max number of windows: 1 to 3 - Min and Max object scale: 0.25 to 3 - Min and Max object rotation: 0 to 3/2pi - Min and Max object translation: -3 to 3 meters - % of images with an object obstructing a window: 50% - Obstruction: 1 object, randomly picked from 8 preset objects in blender - rotation: 0 to 3/2pi - translation: -1.5 to 1.5 meters - scale: 1.5 to 2

Obstruction objects are placed in an "Obstruction" collection, out of the camera view, and then copied into the view. The original window is also done the same. Objects copied in (the obstruction and windows) are in a "window" collection, and are deleted and re-copied every image iteration.

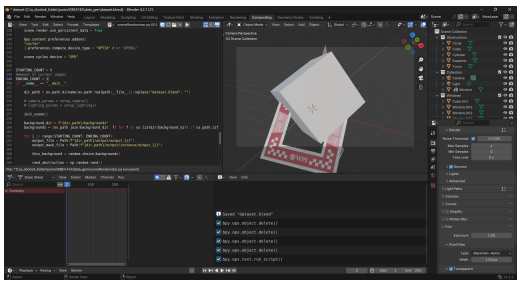

Fig. 1. Blender GUI for dataset generation

While the dataset generation is slow, and obstruction is not always placed in a realistic way and not textured, the dataset is diverse enough to train a model to generalize to detect windows in images.

Please note that the dataset is not included in the submission. To run the dataset generation script, please make sure output directories in the python file are set correctly.

B. Results

Below are some results of the generated dataset:

Fig. 2. Image 1, RGB

Fig. 3. Image 1, Segmentation mask

Fig. 4. Image 2, RGB

Fig. 5. Image 2, Segmentation mask

Before training, we generate 10 times this data. The dataset is augmented using random blur, noise, crop, etc. to generate around 50000 images.

II. Part 2

For this part, we are performing semantic segmentation to find the regions of the image/video that contain the window.

A. Network architecture

The architecture used to train this model is UNET. Below is the diagram for the architecture:

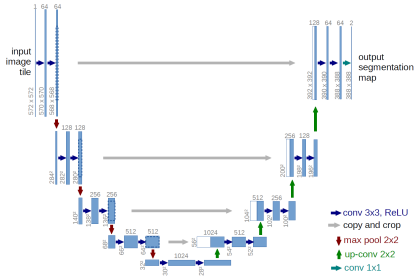

Fig. 6. UNET architecture

TABLE I
HYPERPARAMETERS FOR SEMANTIC SEGMENTATION

Optimizer	Learning rate	Batch size
Adam Optimizer	1e-4	32

B. Implementation

The model was trained on the dataset generated in Part 1. We are using the **cross-entropy** loss function. The hyperparameters used for training are shown in Table 1.

C. Results

Results for the model are in the vdeo "part2.mp4". Below are the loss curve per epoch plot:

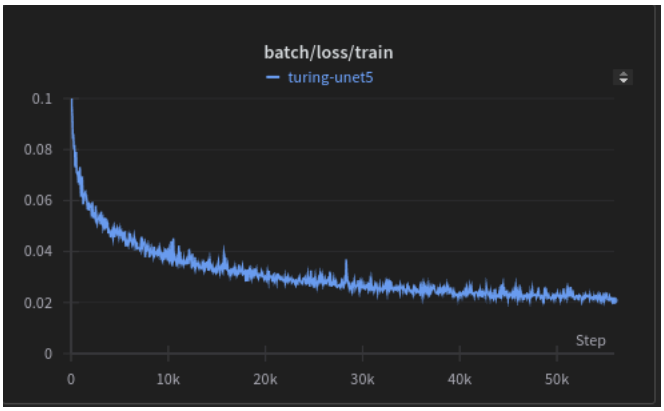

Fig. 7. Training loss curve, batch

III. PART 3

For this part, we are performing instance segmentation to find the regions of the image/video that contain the window.

A. Network architecture

The architecture used to train this model is UNET. The architecture diagram for this is depicted in figure 6.

B. Implementation

The model was trained on the dataset generated in Part 1. The hyperparameters used for training are the same as in Part 2, and is shown in Table 1. Loss function used is depicted below.

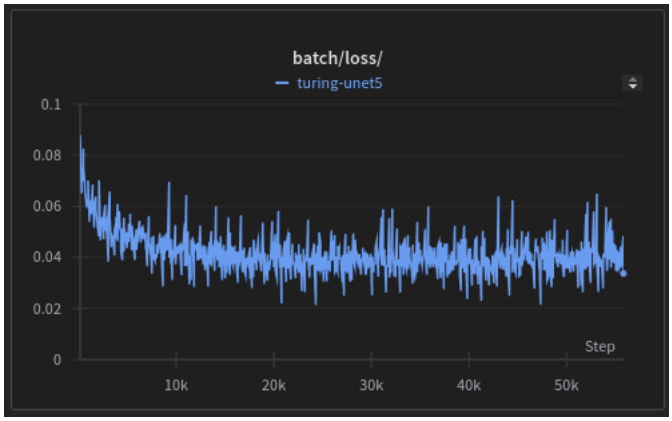

Fig. 8. Evaluate loss curve, batch

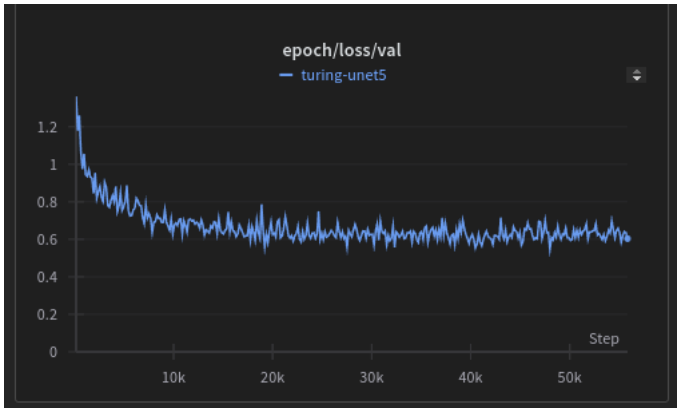

Fig. 9. Evaluate loss curve, epoch

1) Loss function: For instance segmentation, we trained our network on three different loss functions, each with different goals in mind. Initially, we tried Cross Entropy.

Cross Entropy is a common loss function for classification tasks, and is used to measure the difference between two probability distributions. In this case, it is used to measure the difference between the predicted mask and the ground truth mask.

However, this loss function does not take into account the spatial information of the image, and is not ideal for instance segmentation.

Next, we tried Dice Loss. Dice Loss is a loss function that is used to measure the similarity between two samples. It is used in this case to measure the similarity between the predicted mask and the ground truth mask.

Dice Loss is better suited for instance segmentation than Cross Entropy, as it is more robust to class imbalance, which in this case is due to more single-instance images. Finally, we tried a combination of Cross Entropy and Dice Loss. The combined loss function is a weighted sum of the two loss functions.

C. Results

Results for the model are in the vdeo ”part3.mp4”. Below are the loss curve per epoch plot:

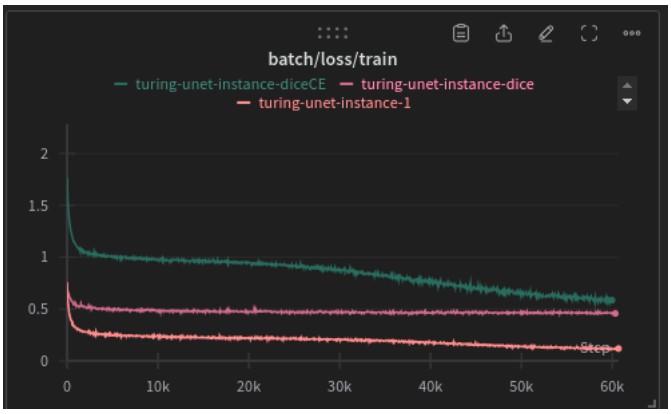

Fig. 10. Training loss curve, batch

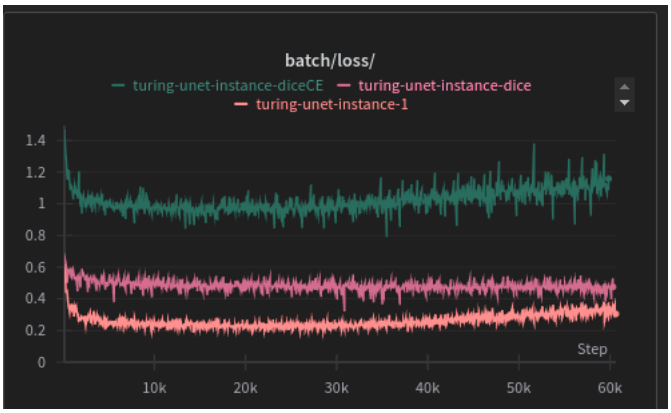

Fig. 11. Evaluate loss curve, batch

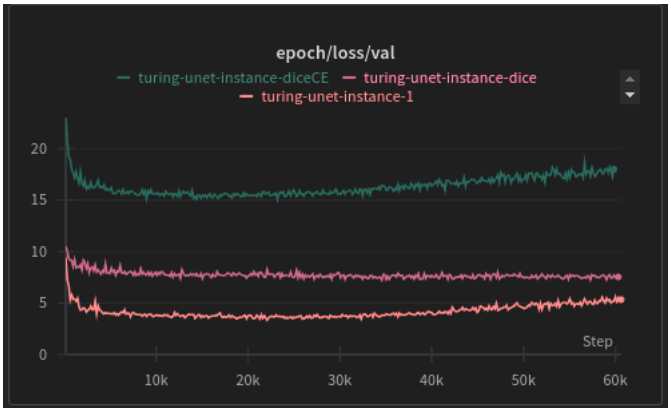

Fig. 12. Evaluate loss curve, epoch