Link for the python notebooks mentioned here:

<https://drive.google.com/file/d/1rpQgrA996Oo0obOG69Awqtn63JwUagUT/view?usp=share_link>

**Model Train:** In the **Img\_seg\_Unet.ipynb** notebook, I constructed a U-Net model architecture, utilizing the Adam optimizer with binary cross-entropy loss function. Subsequently, I conducted model training over 10 epochs with a batch size of 8, employing the training data while validating against separate validation data.

**Model performance result:** I proceeded to assess the model's performance on the validation dataset. The training loss exhibited a decline from 0.5190 to 0.1615, maintaining a consistent training accuracy of approximately 89.06%. Similarly, the validation loss decreased from 0.2134 to 0.1536, with the validation accuracy also remaining stable at around 88.63%.

**Model Predict:** I initially attempted to visualize the original validation images, their corresponding ground truth labels, and the model predictions (check **predict1.ipynb** and **predict2.ipynb**). However, the predictions yielded black screens, prompting a refinement in the visualization approach. Consequently, I decided to visualize the original training images alongside their ground truth labels and model predictions(**predict3.ipynb**)

Despite this adjustment, the model still failed to produce meaningful segmentation predictions, displaying black screens for both training and validation images. This outcome suggests potential deficiencies in the model's learning process, attributable to several factors:

* Insufficient Data: The model might not have encountered an adequate variety of training examples or the training data might not sufficiently represent the target distribution.
* Complexity of the Task: The current model architecture may lack the expressiveness necessary to capture essential features in the data.
* Incorrect Model Parameters: The model's hyperparameters, such as learning rate, batch size, or choice of optimizer, might not be optimally configured for this specific task and dataset.
* Preprocessing Issues: Errors in data preprocessing, such as incorrect normalization or resizing procedures, could affect the model's ability to learn effectively.

**Proposed Solutions:**

To address these challenges, I made several adjustments to the model and training process:

**Model Architecture Adjustment:** I modified the model architecture to include **batch normalization layers** after each convolutional layer and switched the loss function from **binary cross-entropy** to **Dice loss**. This change can improve the model's stability during training and enhance its ability to capture intricate patterns in the data.

**Data Augmentation:** I augmented the training data by applying random transformations such as rotation, shifting, and flipping. This augmentation technique increases the diversity of the training dataset, allowing the model to learn more robust features and improve its generalization performance.

**Loss Function Change:** I switched from binary cross-entropy to Dice loss. Dice loss is well-suited for imbalanced segmentation tasks, as it penalizes false positives and false negatives differently, leading to better performance in scenarios where the classes are imbalanced or where there are small objects to detect.

**Hyperparameter Tuning:** I adjusted hyperparameters such as learning rate, batch size, and training epochs to find the optimal configuration for the model. Fine-tuning these parameters can significantly impact the model's convergence and overall performance.