**Step 1: Install Tensorflow**

```
!pip install tensorflow
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.15.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.5.26)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorfl
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.9.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (16.0.6)
Requirement already satisfied: ml-dtypes~=0.2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.25.2)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/loca
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.9.0)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorf
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.60.1)
Requirement already satisfied: tensorboard<2.16,>=2.15 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.2)
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from tensorflo
Requirement already satisfied: keras<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensor
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.1
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.1
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->ter
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15-
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensor
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->ter
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorb
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthli
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboar
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tens
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tens
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboa
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->go
```

**Step 2:** Make folders for dataset

```
import os

os.makedirs("/content/sample_data/kaggle/")
os.chdir("/content/sample_data/kaggle/")
```

**Step 6: Connect to Google Drive**

Make connection to the Google Drive

```
# make connect to google drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

**Step 3:** Copy dataset from Google Drive to Collab

```
cp /content/drive/MyDrive/kaggle_dataset/breast-cancer-semantic-segmentation-bcss.zip /content/sample_data/kaggle/breast-cancer-
```

**Step 4:** Unzip dataset

```
!unzip -x breast-cancer-semantic-segmentation-bcss.zip
```

```
  inflating: BCSS_512/val_mask_512/TCGA-OL-A97C-DX1_xmin68058_ymin32495_MPP-0_512_0_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-OL-A97C-DX1_xmin68058_ymin32495_MPP-0_512_1024_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-OL-A97C-DX1_xmin68058_ymin32495_MPP-0_512_1536_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-OL-A97C-DX1_xmin68058_ymin32495_MPP-0_512_2048_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-OL-A97C-DX1_xmin68058_ymin32495_MPP-0_512_2560_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-OL-A97C-DX1_xmin68058_ymin32495_MPP-0_512_3072_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-OL-A97C-DX1_xmin68058_ymin32495_MPP-0_512_3584_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-OL-A97C-DX1_xmin68058_ymin32495_MPP-0_512_512_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_0_0_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_0_1024_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_0_1536_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_0_2048_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_0_2560_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_0_512_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_1024_0_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_1024_1024_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_1024_1536_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_1024_2048_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_1024_2560_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_1024_512_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_1536_0_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_1536_1024_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_1536_1536_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_1536_2048_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_1536_2560_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_1536_512_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_2048_0_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_2048_1024_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_2048_1536_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_2048_2048_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_2048_2560_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_2048_512_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_2560_0_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_2560_1024_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_2560_1536_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_2560_2048_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_2560_2560_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_2560_512_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_512_0_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_512_1024_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_512_1536_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_512_2048_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_512_2560_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA10-DX1_xmin43039_ymin23986_MPP-0_512_512_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA15-DX1_xmin55486_ymin28926_MPP-0_0_0_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA15-DX1_xmin55486_ymin28926_MPP-0_0_1024_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA15-DX1_xmin55486_ymin28926_MPP-0_0_512_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA15-DX1_xmin55486_ymin28926_MPP-0_1024_0_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA15-DX1_xmin55486_ymin28926_MPP-0_1024_1024_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA15-DX1_xmin55486_ymin28926_MPP-0_1024_512_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA15-DX1_xmin55486_ymin28926_MPP-0_1536_0_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA15-DX1_xmin55486_ymin28926_MPP-0_1536_1024_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA15-DX1_xmin55486_ymin28926_MPP-0_1536_512_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA15-DX1_xmin55486_ymin28926_MPP-0_512_0_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA15-DX1_xmin55486_ymin28926_MPP-0_512_1024_size512.png
  inflating: BCSS_512/val_mask_512/TCGA-S3-AA15-DX1_xmin55486_ymin28926_MPP-0_512_512_size512.png
  inflating: gtruth_codes_512.tsv
  inflating: make_patches_ori.py
```

**Step 5: Setup Environment**

First, set up your environment by installing necessary libraries and setting the working directory. If you've already set up your environment as per your previous code, you can skip this step.

```
import os
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import img_to_array, load_img
import glob
import gc  # Garbage Collector interface

# After predicting or loading large data, clear unused variables and collect garbage
gc.collect()  # Collect garbage to free memory
```

```
0
```

**Step 7: Load the Model**

Load the trained model from the file you saved earlier.

```
# Load the saved model
model_path = '/content/drive/MyDrive/SavingsOfTrainingModels/cancer_predict_trainigs.h5'
model = load_model(model_path)
```

**Step 8: Prepare the Test Data**

You need to prepare your test data similar to how you prepared your training data. I'll assume you have a separate folder for test images.

```
# Update these paths according to your test data location
test_image_paths = glob.glob('/content/sample_data/kaggle/BCSS/test/*.png')  # Update path as needed

# Function to load and convert test images
# Function to process and predict images in batches
def predict_in_batches(image_paths, image_size, model, batch_size=5):  # You can adjust the batch size
    num_images = len(image_paths)
    predictions = []

    for i in range(0, num_images, batch_size):
        batch_paths = image_paths[i:i+batch_size]
        batch_images = np.zeros((len(batch_paths), *image_size, 3), dtype=np.float32)

        # Load and preprocess images
        for j, path in enumerate(batch_paths):
            img = load_img(path, target_size=image_size, color_mode='rgb')
            img = img_to_array(img) / 255.0
            batch_images[j] = img

        # Predict current batch
        batch_predictions = model.predict(batch_images)
        predictions.extend(batch_predictions)

    return np.array(predictions)

# Define the image size
image_size = (512, 512)

# Load test images
test_images = predict_in_batches(test_image_paths, image_size, model)
```

```
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 1s 534ms/step
```

**Step 9: Make Predictions**

Use the loaded model to predict the test data.

```
def load_batch_images(image_paths, image_size, num=10):
    images = []
    for path in image_paths[:num]:  # Load only the first 'num' images
        img = load_img(path, target_size=image_size)
        images.append(img_to_array(img) / 255.0)
    return np.array(images)
```

**Step 10: Display the Results**

You may want to display the results to visually inspect how well your model is performing. You can plot the original test images alongside their predicted masks.

```
import matplotlib.pyplot as plt

def plot_predictions(original_image_paths, predictions, image_size, num=10):
    # Load a batch of original images
    original_images = load_batch_images(original_image_paths, image_size, num=num)

    # Plotting
    plt.figure(figsize=(20, 4))
    for i in range(num):
        # Display original
        ax = plt.subplot(2, num, i + 1)
        plt.imshow(original_images[i])
```

```
            plt.title("Original")
            plt.axis("off")

            # Display prediction
            ax = plt.subplot(2, num, i + 1 + num)
            plt.imshow(predictions[i].squeeze(), cmap='gray')  # Make sure predictions are the correct shape
            plt.title("Predicted")
            plt.axis("off")

        plt.show()


    # Plot some test images and their predicted masks
    plot_predictions(test_image_paths, test_images, image_size)
```