Configuration Manual

MSc Research Project
Data Analytics

Priyanka Bundela
Student ID: x22247734

School of Computing
National College of Ireland

Supervisor: Bharat Agarwal

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Priyanka Bundela
Student ID:	x22247734
Programme:	Data Analytics
Year:	2024
Module:	MSc Research Project
Supervisor:	Bharat Agarwal
Submission Due Date:	12/12/2024
Project Title:	Configuration Manual
Word Count:	XXX
Page Count:	8

Signature:	
Date:	11th December 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Priyanka Bundela
x22247734

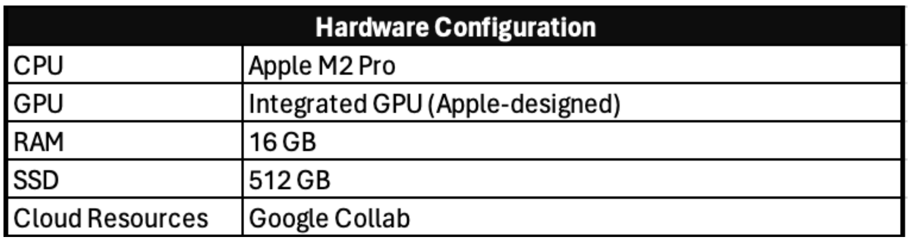
1 Introduction

The hardware and software employed in this study are detailed in the configuration manual , along with the breakdown of methodology in the paper ”Carotid Artery Plaque Analysis Using Deep Neural Networks for Improved Detection and Classification ”.

2 System Configuration

2.1 Hardware Configuration

This section provides an overview of the hardware and cloud resources used for the project, ensuring compatibility and optimal performance for the tasks.

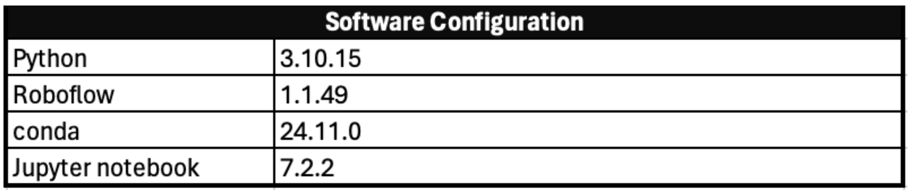


Hardware Configuration	
CPU	Apple M2 Pro
GPU	Integrated GPU (Apple-designed)
RAM	16 GB
SSD	512 GB
Cloud Resources	Google Collab

Figure 1: Hardware Configuration

2.2 Software Configuration

This section outlines the software requirements and installation steps to set up the development environment for the project.

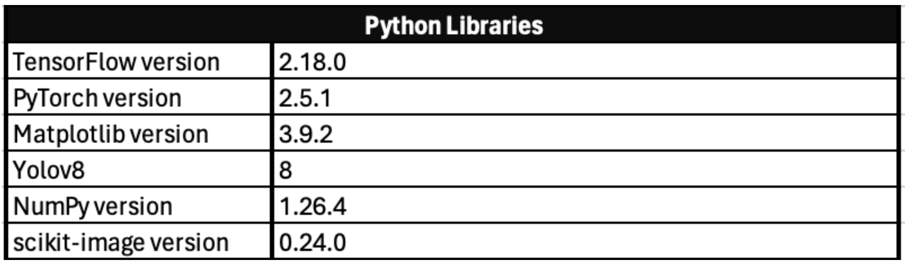


Software Configuration	
Python	3.10.15
Roboflow	1.1.49
conda	24.11.0
Jupyter notebook	7.2.2

Figure 2: Software Configuration

Python Libraries :

Python is the primary language used to build and run the models and for much of the data manipulations and library interfacing. It too helps a few libraries which are part of this examination. A description and version of those libraries can be described below



Python Libraries	
TensorFlow version	2.18.0
PyTorch version	2.5.1
Matplotlib version	3.9.2
Yolov8	8
NumPy version	1.26.4
scikit-image version	0.24.0

Figure 3: Python Libraries

Anaconda And Jupyter Notebook:
Anaconda provides the package management and deployment solutions that take care of all the dependencies for your Python projects and the Jupyter Notebook which is an interactive tool for Data Science, used for prototyping and Data Visualization.

RoboFlow:
Roboflow is a tool that helps you manage, annotate and augment your datasets for deep learning applications (including medical imaging). Enables seamless integration with frameworks such as TensorFlow and PyTorch for efficient model training and deployment

1. https://www.python.org/
2. https://www.anaconda.com/ download
3. https://roboflow.com/

3 Environment Setup

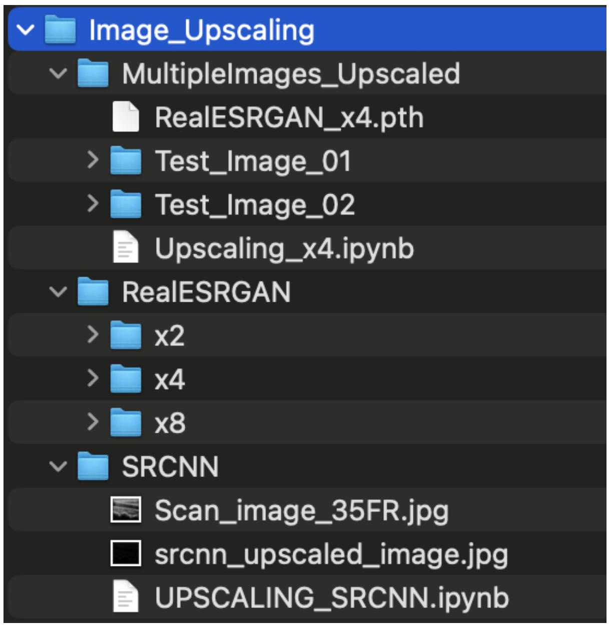
3.1 Installing Required Packages and creating file structure

This section explain about the set up, run, and manage the three primary tasks of the project: Image up scaling, segmentation of the features and measurements of the features. Each folder has a modular folder structure to resolve various tasks, enabling maintenance and scalability.

Project File Structure

1. Image UpScaling:

Purpose: Enhance the resolution of carotid artery images for better feature visibility.



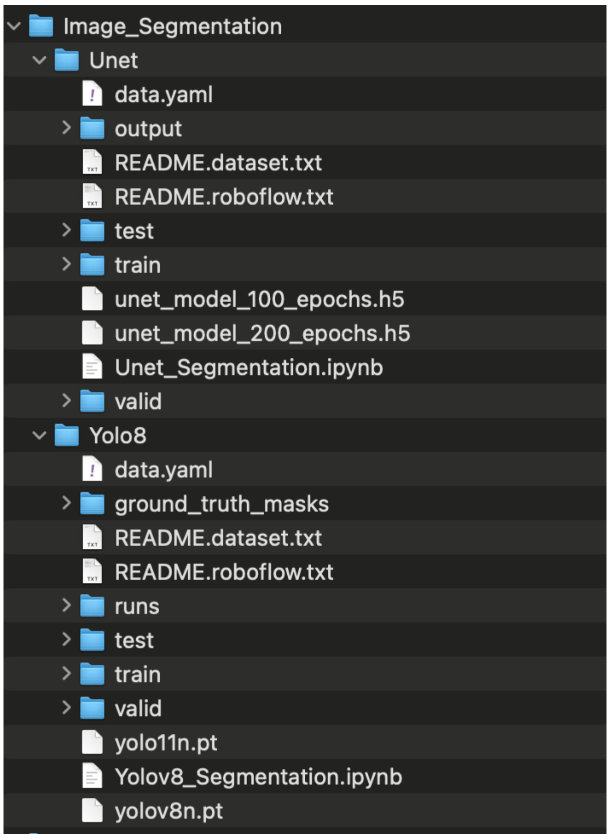
```
Image_Upscaling
    MultipleImages_Upscaled
        RealESRGAN_x4.pth
        Test_Image_01
        Test_Image_02
        Upscaling_x4.ipynb
    RealESRGAN
        x2
        x4
        x8
    SRCNN
        Scan_image_35FR.jpg
        srcnn_upscaled_image.jpg
        UPSCALING_SRCNN.ipynb
```

Figure 4: Image UpScaling

2. Image Segmentation:

Purpose: Segment carotid artery features using advanced architectures.

Folder Structure:

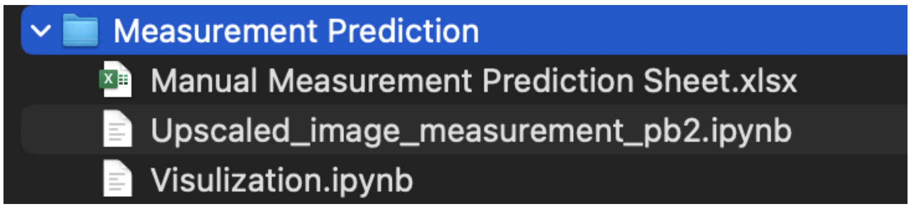


```
Image_Segmentation
    Unet
        data.yaml
        output
        README.dataset.txt
        README.roboflow.txt
        test
        train
        unet_model_100_epochs.h5
        unet_model_200_epochs.h5
        Unet_Segmentation.ipynb
        valid
    Yolo8
        data.yaml
        ground_truth_masks
        README.dataset.txt
        README.roboflow.txt
        runs
        test
        train
        valid
        yolo11n.pt
        Yolov8_Segmentation.ipynb
        yolov8n.pt
```

Figure 5: Image Segmentation File Structure

3. Measurement Prediction:

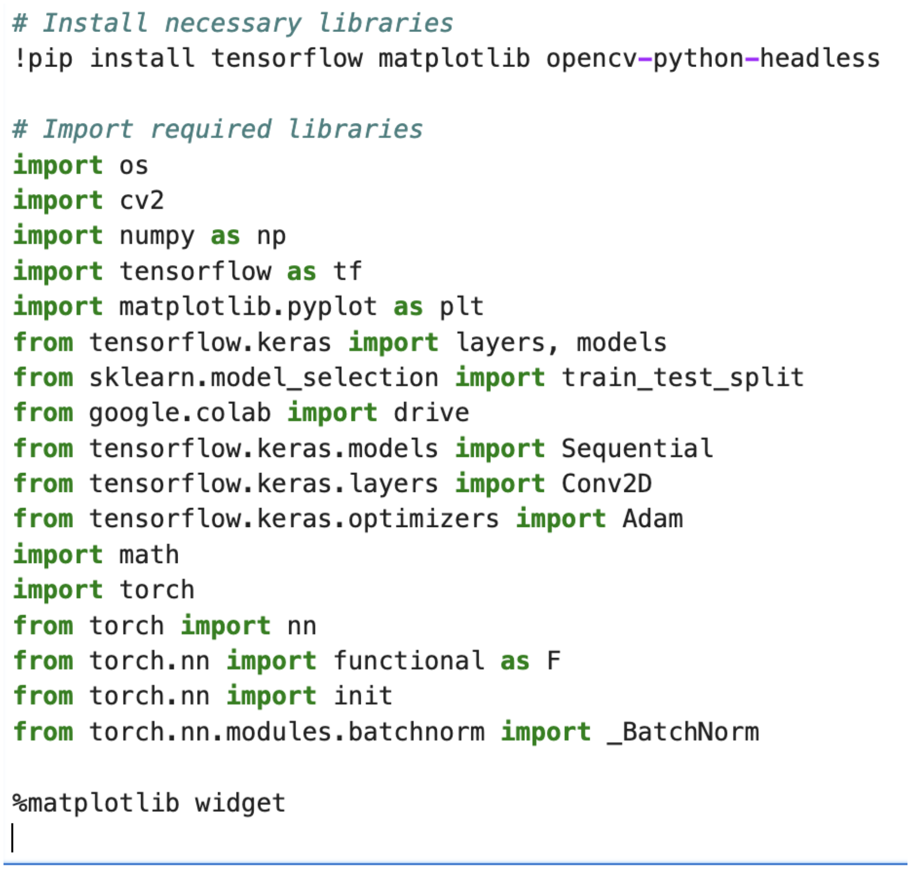
Purpose: Segment carotid artery features using advanced architectures.



Measurement Prediction
Manual Measurement Prediction Sheet.xlsx
Upscaled_image_measurement_pb2.ipynb
Visulization.ipynb

Figure 6: Measurement Prediction

Importing Necessary Libraries The libraries are needed for various part of the project including image up-scaling,image segmentation,measurement prediction and evaluation



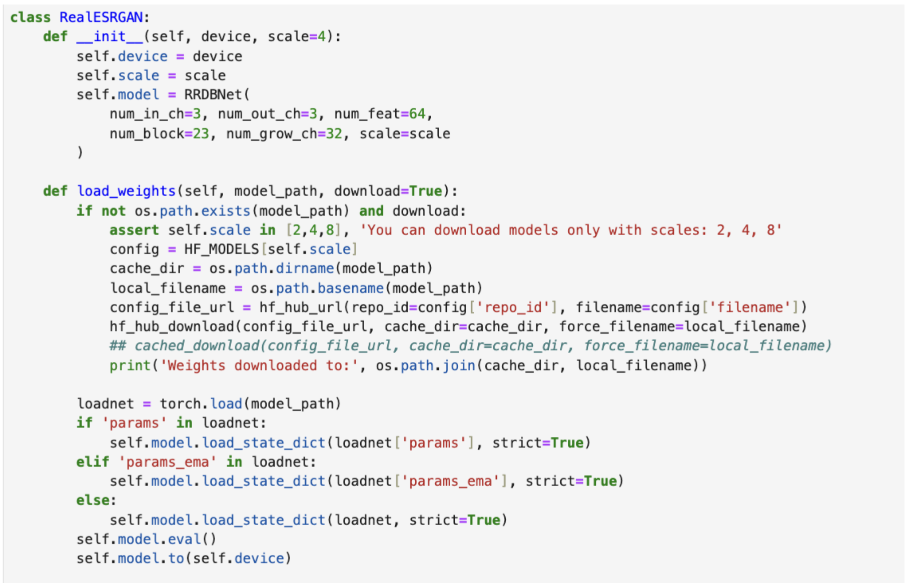
```
# Install necessary libraries
!pip install tensorflow matplotlib opencv-python-headless

# Import required libraries
import os
import cv2
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras import layers, models
from sklearn.model_selection import train_test_split
from google.colab import drive
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.optimizers import Adam
import math
import torch
from torch import nn
from torch.nn import functional as F
from torch.nn import init
from torch.nn.modules.batchnorm import _BatchNorm

%matplotlib widget
```

Figure 7: Libraries Details

4 Model Training and Evaluation

4.1 Data Preparation

The RealESRGAN class as shown in Figure 8 is intended for image enhancement through the RRDBNet architecture. It initializes a super-resolution model for 2x, 4x, or 8x scaling, loads pre-trained weights (either locally or through download), and configures the model for inference on a designated device (GPU or CPU). The class is utilized to upgrade low-resolution photos to high-resolution outputs, appropriate for activities such as photo restoration or meticulous analysis.



```python
class RealESRGAN:
    def __init__(self, device, scale=4):
        self.device = device
        self.scale = scale
        self.model = RRDBNet(
            num_in_ch=3, num_out_ch=3, num_feat=64,
            num_block=23, num_grow_ch=32, scale=scale
        )

    def load_weights(self, model_path, download=True):
        if not os.path.exists(model_path) and download:
            assert self.scale in [2,4,8], 'You can download models only with scales: 2, 4, 8'
            config = HF_MODELS[self.scale]
            cache_dir = os.path.dirname(model_path)
            local_filename = os.path.basename(model_path)
            config_file_url = hf_hub_url(repo_id=config['repo_id'], filename=config['filename'])
            hf_hub_download(config_file_url, cache_dir=cache_dir, force_filename=local_filename)
            ## cached_download(config_file_url, cache_dir=cache_dir, force_filename=local_filename)
            print('Weights downloaded to:', os.path.join(cache_dir, local_filename))

        loadnet = torch.load(model_path)
        if 'params' in loadnet:
            self.model.load_state_dict(loadnet['params'], strict=True)
        elif 'params_ema' in loadnet:
            self.model.load_state_dict(loadnet['params_ema'], strict=True)
        else:
            self.model.load_state_dict(loadnet, strict=True)
        self.model.eval()
        self.model.to(self.device)
```

Figure 8: Image upscaling data

Roboflow is utilized for data labeling, generating bounding box annotations for YOLO object recognition and segmentation masks for UNet. It enables accurate labeling and segmentation of the carotid artery for model training. The software streamlines dataset organizing and produces annotated data in compatible formats. This guarantees effective integration with AI models for medical imaging applications.

The data is split into 70

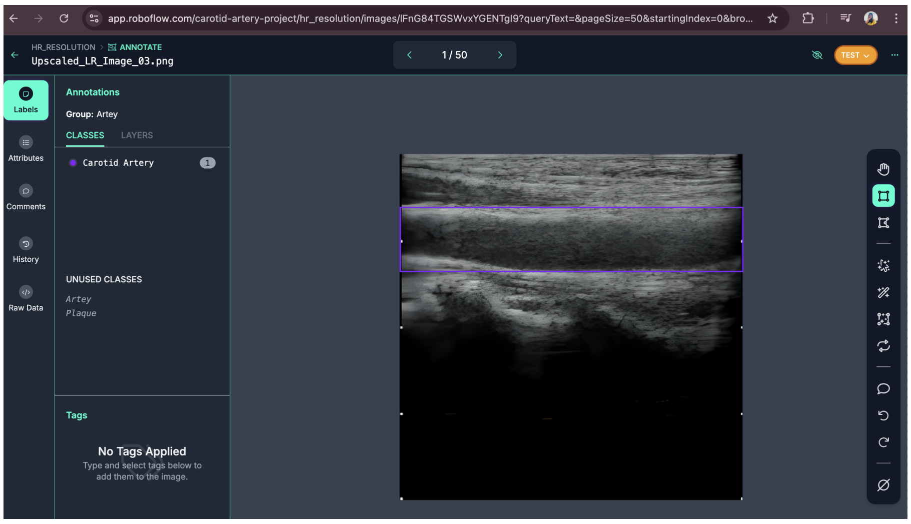
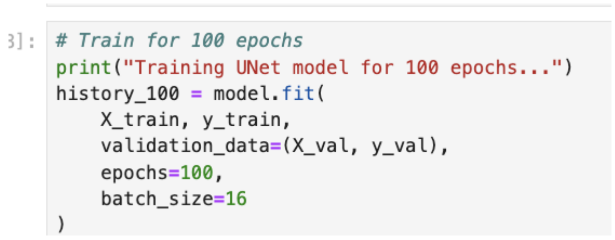


Figure 9: Data Preparation

4.2 Model Training

We used training for 100 and 200 epochs on both of our models, UNet and YOLOv8, to get the best results.

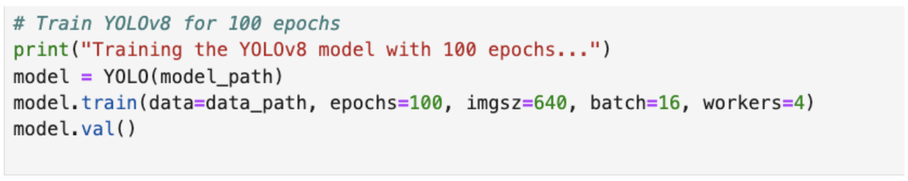
This section illustrates the training and evaluation of a UNet model for segmentation tasks. The model is trained for 100 epochs with a batch size of 16, utilizing the training $(X_{\text{train}}, y_{\text{train}})$ and validation $(X_{\text{val}}, y_{\text{val}})$ datasets in the initial section. The training history has been saved for analysis.



```
3]: # Train for 100 epochs
    print("Training UNet model for 100 epochs...")
    history_100 = model.fit(
        X_train, y_train,
        validation_data=(X_val, y_val),
        epochs=100,
        batch_size=16
    )
```

Figure 10: Training Model For Unet

A dataset with files named "train," "valid," and "test" is used by the code to train a YOLOv8 model for 100 events. Roboflow was used to name the dataset, and it was saved in the format needed for YOLO training. The training setup sets the image size to 640 pixels, the batch size to 16, and the number of workers to use for loading data to 4. Once the model has been trained, its validation performance is checked to see how accurate it is and how well it can find things.



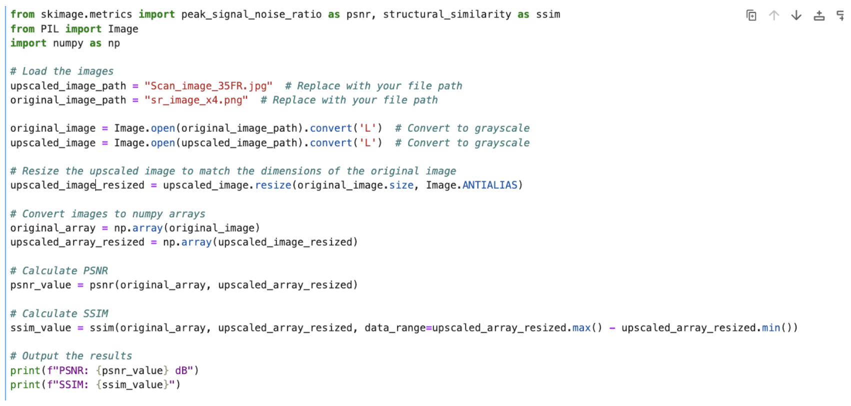
```
# Train YOLOv8 for 100 epochs
print("Training the YOLOv8 model with 100 epochs...")
model = YOLO(model_path)
model.train(data=data_path, epochs=100, imgsz=640, batch=16, workers=4)
model.val()
```

Figure 11: Training Model For Yolov8

4.3 Evaluation Metrics

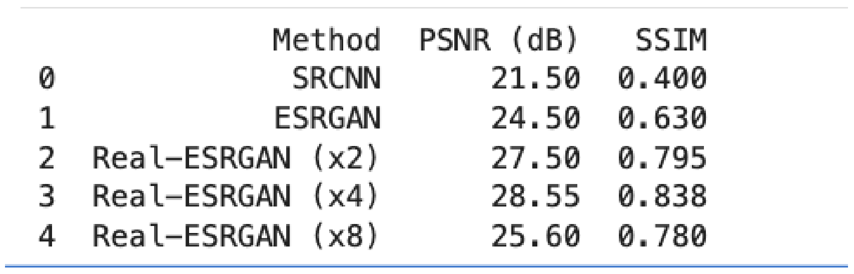
PSNR and SSIM Calculation

This code 12 computes image quality measures, PSNR and SSIM, to evaluate the quality of an upscaled image against its original counterpart. The process initially loads the photos, transforms them to grayscale for consistency, and standardizes their proportions by scaling the upscaled image. PSNR assesses the overall quality of the upscaled image concerning noise levels, whereas SSIM quantifies the structural similarity between the two images, indicating the extent to which the upscaled image preserves features and structural information. The computed numbers are thereafter printed for analysis.



```python
from skimage.metrics import peak_signal_noise_ratio as psnr, structural_similarity as ssim
from PIL import Image
import numpy as np

# Load the images
upscaled_image_path = "Scan_image_35FR.jpg"  # Replace with your file path
original_image_path = "sr_image_x4.png"  # Replace with your file path

original_image = Image.open(original_image_path).convert('L')  # Convert to grayscale
upscaled_image = Image.open(upscaled_image_path).convert('L')  # Convert to grayscale

# Resize the upscaled image to match the dimensions of the original image
upscaled_image_resized = upscaled_image.resize(original_image.size, Image.ANTIALIAS)

# Convert images to numpy arrays
original_array = np.array(original_image)
upscaled_array_resized = np.array(upscaled_image_resized)

# Calculate PSNR
psnr_value = psnr(original_array, upscaled_array_resized)

# Calculate SSIM
ssim_value = ssim(original_array, upscaled_array_resized, data_range=upscaled_array_resized.max() - upscaled_array_resized.min())

# Output the results
print(f"PSNR: {psnr_value} dB")
print(f"SSIM: {ssim_value}")
```

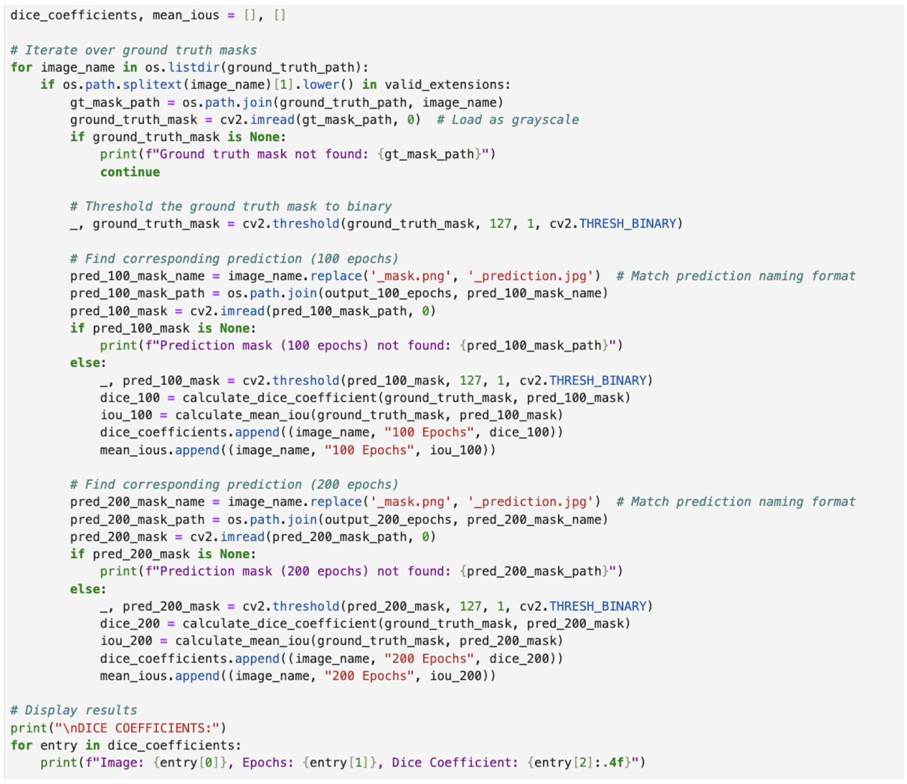
Figure 12: PSNR and SSIM Calculation



```
              Method  PSNR (dB)   SSIM
0              SRCNN      21.50  0.400
1             ESRGAN      24.50  0.630
2  Real-ESRGAN (x2)      27.50  0.795
3  Real-ESRGAN (x4)      28.55  0.838
4  Real-ESRGAN (x8)      25.60  0.780
```

Figure 13: PSNR and SSIM

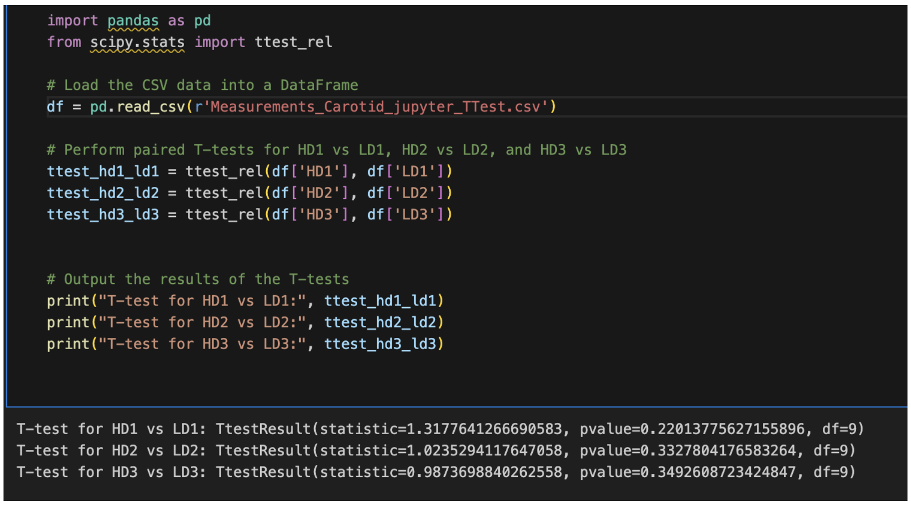
Calculation of Dice coefficient This section explains 14, the Dice Coefficient and Intersection over Union (IoU) are used to rate the success of segmentation. It goes through ground truth masks and compares them to projected masks that are made 100 and 200 times. Before the metrics are calculated, the ground truth and forecasts are turned into binary masks. The Dice Coefficient and IoU values are found for every mask and saved for forecasts made every 100 and 200 times. At the end, the results are printed, along with the Dice Coefficients for each picture.



```python
dice_coefficients, mean_ious = [], []

# Iterate over ground truth masks
for image_name in os.listdir(ground_truth_path):
    if os.path.splitext(image_name)[1].lower() in valid_extensions:
        gt_mask_path = os.path.join(ground_truth_path, image_name)
        ground_truth_mask = cv2.imread(gt_mask_path, 0)  # Load as grayscale
        if ground_truth_mask is None:
            print(f"Ground truth mask not found: {gt_mask_path}")
            continue

        # Threshold the ground truth mask to binary
        _, ground_truth_mask = cv2.threshold(ground_truth_mask, 127, 1, cv2.THRESH_BINARY)

        # Find corresponding prediction (100 epochs)
        pred_100_mask_name = image_name.replace('_mask.png', '_prediction.jpg')  # Match prediction naming format
        pred_100_mask_path = os.path.join(output_100_epochs, pred_100_mask_name)
        pred_100_mask = cv2.imread(pred_100_mask_path, 0)
        if pred_100_mask is None:
            print(f"Prediction mask (100 epochs) not found: {pred_100_mask_path}")
        else:
            _, pred_100_mask = cv2.threshold(pred_100_mask, 127, 1, cv2.THRESH_BINARY)
            dice_100 = calculate_dice_coefficient(ground_truth_mask, pred_100_mask)
            iou_100 = calculate_mean_iou(ground_truth_mask, pred_100_mask)
            dice_coefficients.append((image_name, "100 Epochs", dice_100))
            mean_ious.append((image_name, "100 Epochs", iou_100))

        # Find corresponding prediction (200 epochs)
        pred_200_mask_name = image_name.replace('_mask.png', '_prediction.jpg')  # Match prediction naming format
        pred_200_mask_path = os.path.join(output_200_epochs, pred_200_mask_name)
        pred_200_mask = cv2.imread(pred_200_mask_path, 0)
        if pred_200_mask is None:
            print(f"Prediction mask (200 epochs) not found: {pred_200_mask_path}")
        else:
            _, pred_200_mask = cv2.threshold(pred_200_mask, 127, 1, cv2.THRESH_BINARY)
            dice_200 = calculate_dice_coefficient(ground_truth_mask, pred_200_mask)
            iou_200 = calculate_mean_iou(ground_truth_mask, pred_200_mask)
            dice_coefficients.append((image_name, "200 Epochs", dice_200))
            mean_ious.append((image_name, "200 Epochs", iou_200))

# Display results
print("\nDICE COEFFICIENTS:")
for entry in dice_coefficients:
    print(f"Image: {entry[0]}, Epochs: {entry[1]}, Dice Coefficient: {entry[2]:.4f}")
```

Figure 14: Calculation of Dice coefficient

Paired T-Test Analysis for Carotid Measurements There are three sets of high-definition (HD) and low-definition (LD) carotid artery data that this code compares using paired t-tests. The sets are HD1 vs LD1, HD2 vs LD2, and HD3 vs LD3. The information is retrieved from a CSV file and stored in a pandas DataFrame. Then, SciPy's `ttest_rel` function is used to perform the paired t-tests for each set. After completing the tests, the t-statistic, p-value, and degrees of freedom are recorded for each comparison. These t-tests are used to determine if there is a statistically significant difference between the two readings.



```python
import pandas as pd
from scipy.stats import ttest_rel

# Load the CSV data into a DataFrame
df = pd.read_csv(r'Measurements_Carotid_jupyter_TTest.csv')

# Perform paired T-tests for HD1 vs LD1, HD2 vs LD2, and HD3 vs LD3
ttest_hd1_ld1 = ttest_rel(df['HD1'], df['LD1'])
ttest_hd2_ld2 = ttest_rel(df['HD2'], df['LD2'])
ttest_hd3_ld3 = ttest_rel(df['HD3'], df['LD3'])


# Output the results of the T-tests
print("T-test for HD1 vs LD1:", ttest_hd1_ld1)
print("T-test for HD2 vs LD2:", ttest_hd2_ld2)
print("T-test for HD3 vs LD3:", ttest_hd3_ld3)
```

```
T-test for HD1 vs LD1: TtestResult(statistic=1.3177641266690583, pvalue=0.22013775627155896, df=9)
T-test for HD2 vs LD2: TtestResult(statistic=1.0235294117647058, pvalue=0.3327804176583264, df=9)
T-test for HD3 vs LD3: TtestResult(statistic=0.9873698840262558, pvalue=0.3492608723424847, df=9)
```

Figure 15: Paired T-Test Analysis for Carotid Measurements