Part I : Global includes

```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import os
import pathlib
from matplotlib import pyplot as plt
import torch
import numpy as np
import cv2
import tensorflow as tf

# Uncomment to disable GPU usage.
# This is required for some models like Pridnet which has too many traininable parameters
tf.config.set_visible_devices([], 'GPU')

from tqdm.notebook import tqdm
import random

import data_importer

from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession

config = ConfigProto()
config.gpu_options.allow_growth = True
session = InteractiveSession(config=config)
```

Part II : Loading test images

```
from dataloader_lodopab_ct import get_validation_dataloader
noisy_dataset = get_validation_dataloader("../../../../Dataset/LoDoPaB-CT/ground_truth_validation/")
```

```
number of image paths :  28
validation data image len : , 28
```

```
def reconstruct_image_from_patches(patches, num_patches_per_row):
    patch_size = patches.shape[1]  # Assuming square patches
    num_patches = patches.shape[0]

    # Calculate the number of rows
    num_patches_per_col = num_patches // num_patches_per_row

    # Initialize an empty image to store the reconstructed result
    reconstructed_image = np.zeros((num_patches_per_col * patch_size, num_patches_per_row * patch_size))
```

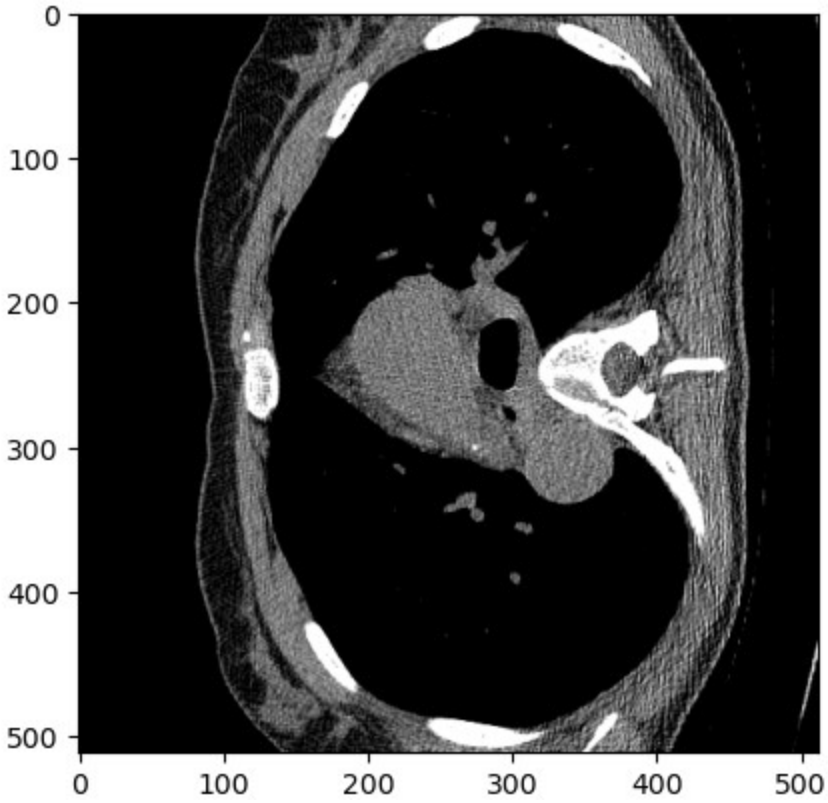
```
    # Reshape the patches into a 2D array
    patches_2d = patches.reshape((num_patches_per_col, num_patches_per_row, patch_size, patch_size))

    # Reconstruct the image by placing each patch in its corresponding position
    for i in range(num_patches_per_col):
        for j in range(num_patches_per_row):
            reconstructed_image[i * patch_size:(i + 1) * patch_size, j * patch_size:(j + 1) * patch_size

    return np.expand_dims(reconstructed_image, axis=-1)

noisy_array = [None] * 28
print(len(noisy_dataset))
for i, data in enumerate(noisy_dataset):
    noisy_array[i] = reconstruct_image_from_patches(torch.squeeze(data[i], axis=0), 8)
    if i == 28:
        break
noisy_array = np.array(noisy_array)
print(noisy_array)
```

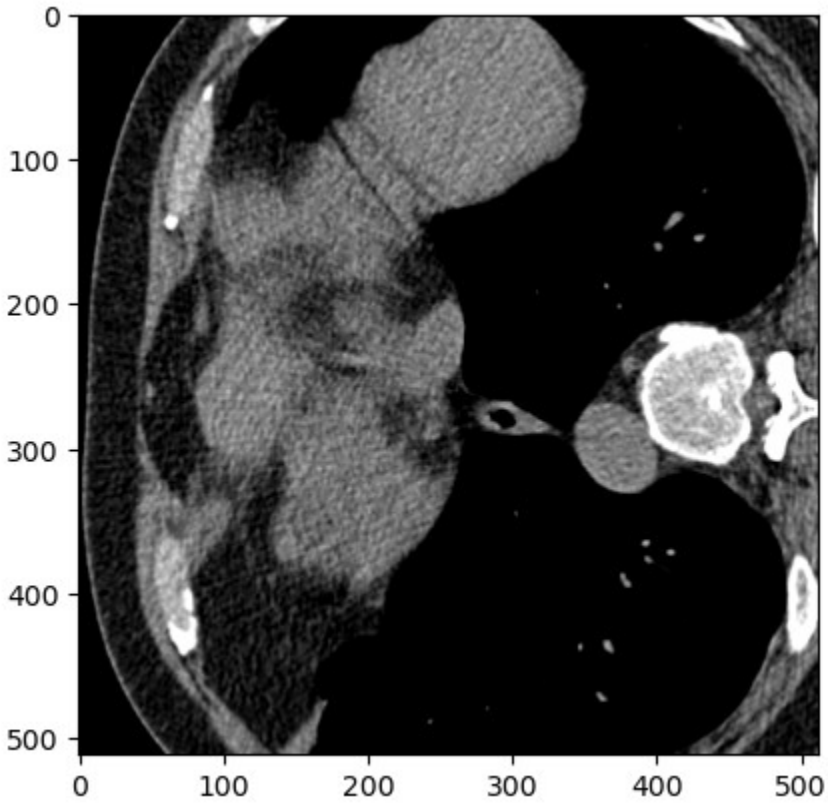
```
28
[[[[0.00000000e+00]
   [0.00000000e+00]
   [0.00000000e+00]
   ...
   [2.01545864e-01]
   [2.27233559e-01]
   [2.47729927e-01]]

  [[0.00000000e+00]
   [0.00000000e+00]
   [0.00000000e+00]
   ...
   [1.97884500e-01]
   [2.25247324e-01]
   [2.57035911e-01]]

  [[0.00000000e+00]
   [1.35118389e-04]
   [4.09101369e-04]
   ...
   [1.93417832e-01]
   [2.24700466e-01]
   [2.67172039e-01]]

  ...

  [[1.91558048e-03]
   [8.41807865e-04]
   [1.29352743e-03]
   ...
   [2.50156038e-02]
   [1.58237405e-02]
   [7.00143981e-04]]

  [[2.05797283e-03]
   [2.28885561e-03]
```

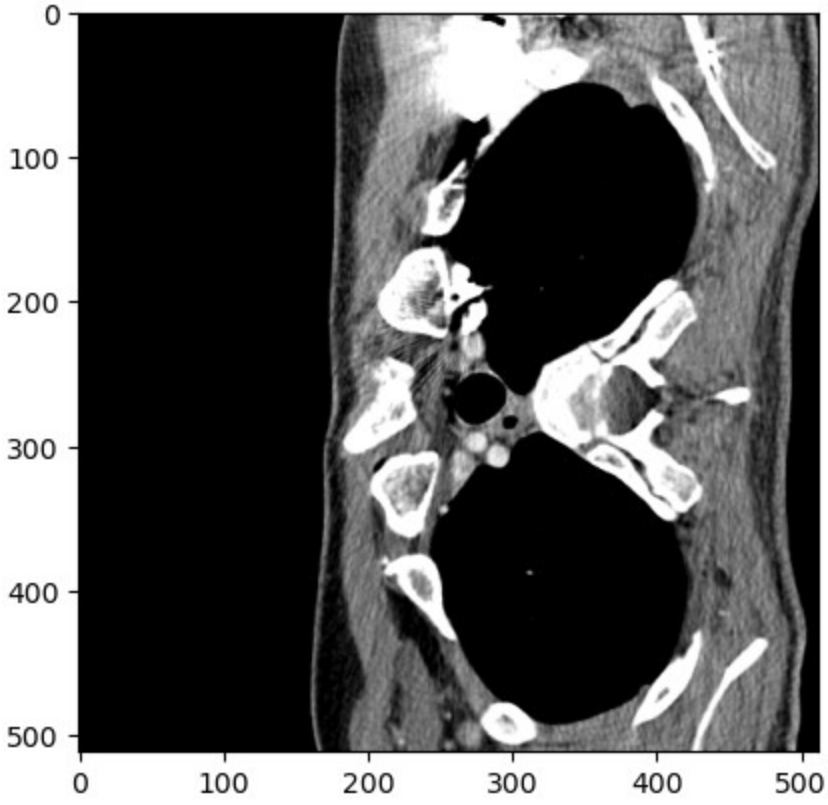
```
          [2.79895985e-03]
          ...
          [2.37569660e-02]
          [1.74138136e-02]
          [9.26138554e-03]]

         [[1.34693610e-03]
          [3.74236028e-03]
          [4.11643507e-03]
          ...
          [1.76892225e-02]
          [1.77762713e-02]
          [2.03140154e-02]]]


        [[[6.74816035e-03]
          [3.39281606e-03]
          [6.21013518e-04]
          ...
          [1.21287532e-01]
          [7.91603923e-02]
```

Visualization of the noisy / ground truth image pair

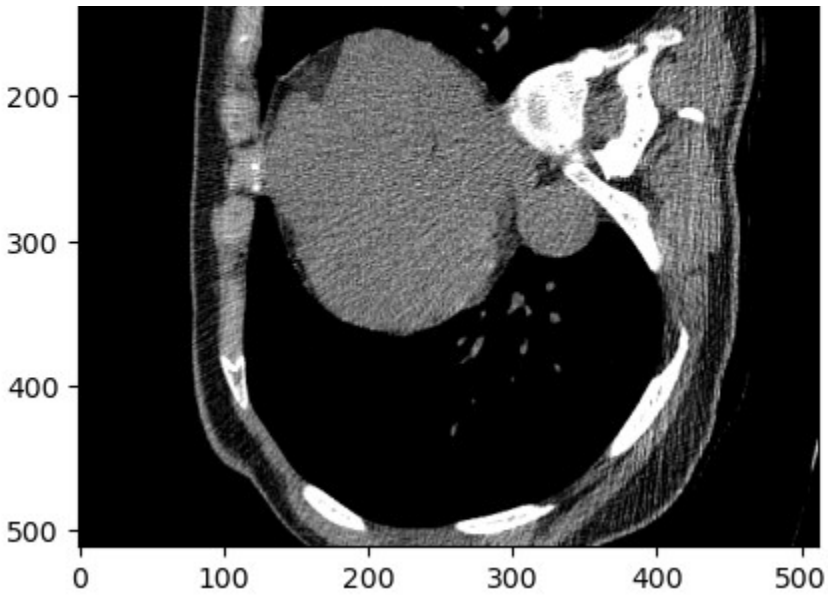
```
from data_importer import denormalize, trunc

with torch.no_grad():
    for i, data in enumerate(noisy_array):
        plt.imshow(trunc(denormalize(data)), vmin=-160.0, vmax=240.0, cmap='gray')
        plt.show()
```

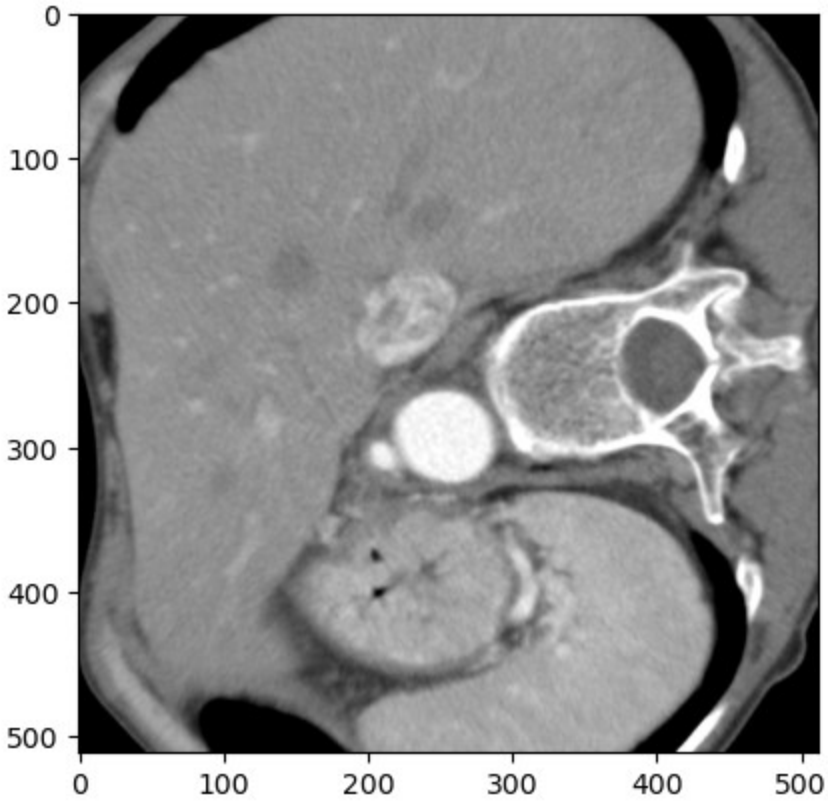


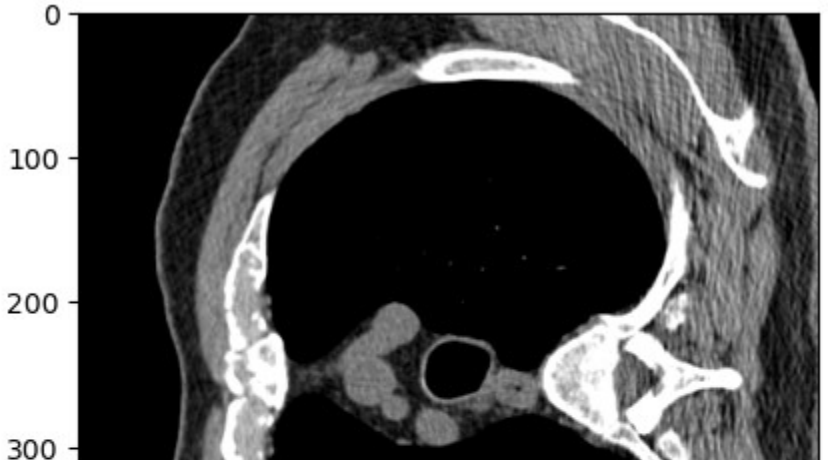

0
100
200
300
400
500
0
100
200
300
400
500

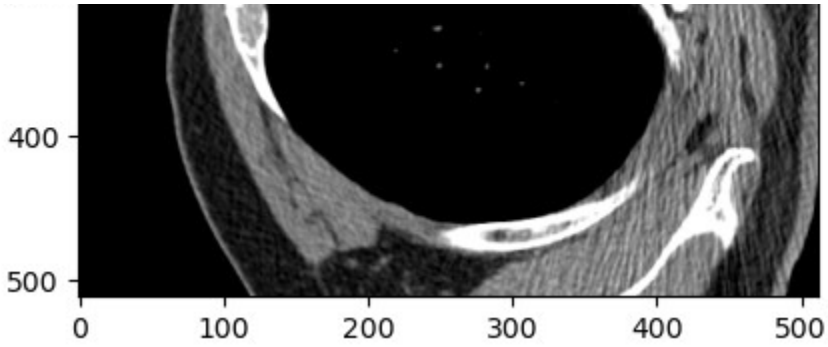

0
100
200
300
400
500
0
100
200
300
400
500

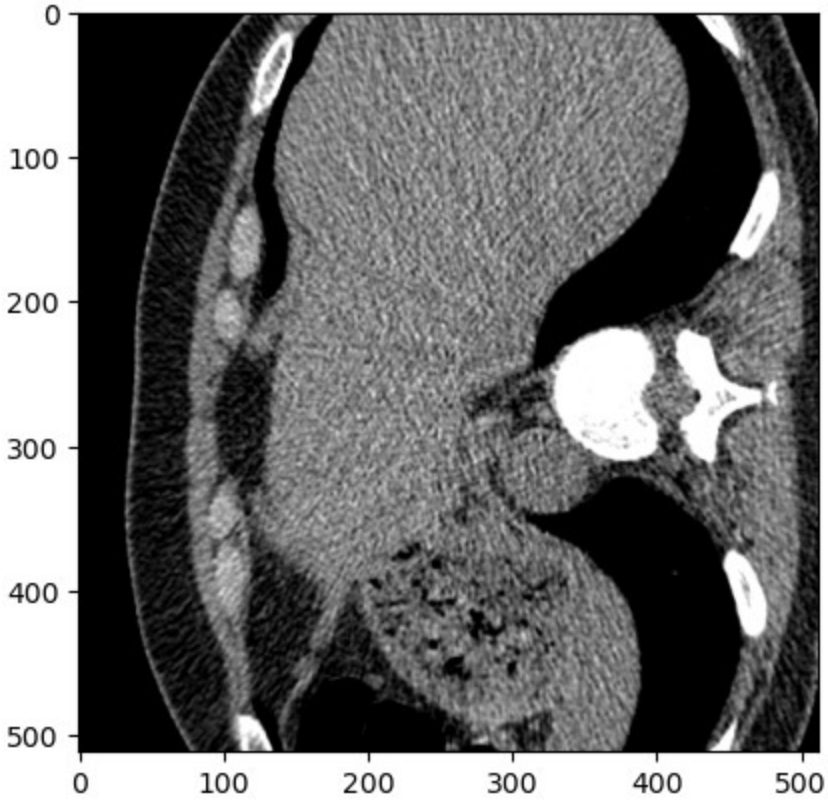

0
100


200
300
400
500
0
100
200
300
400
500

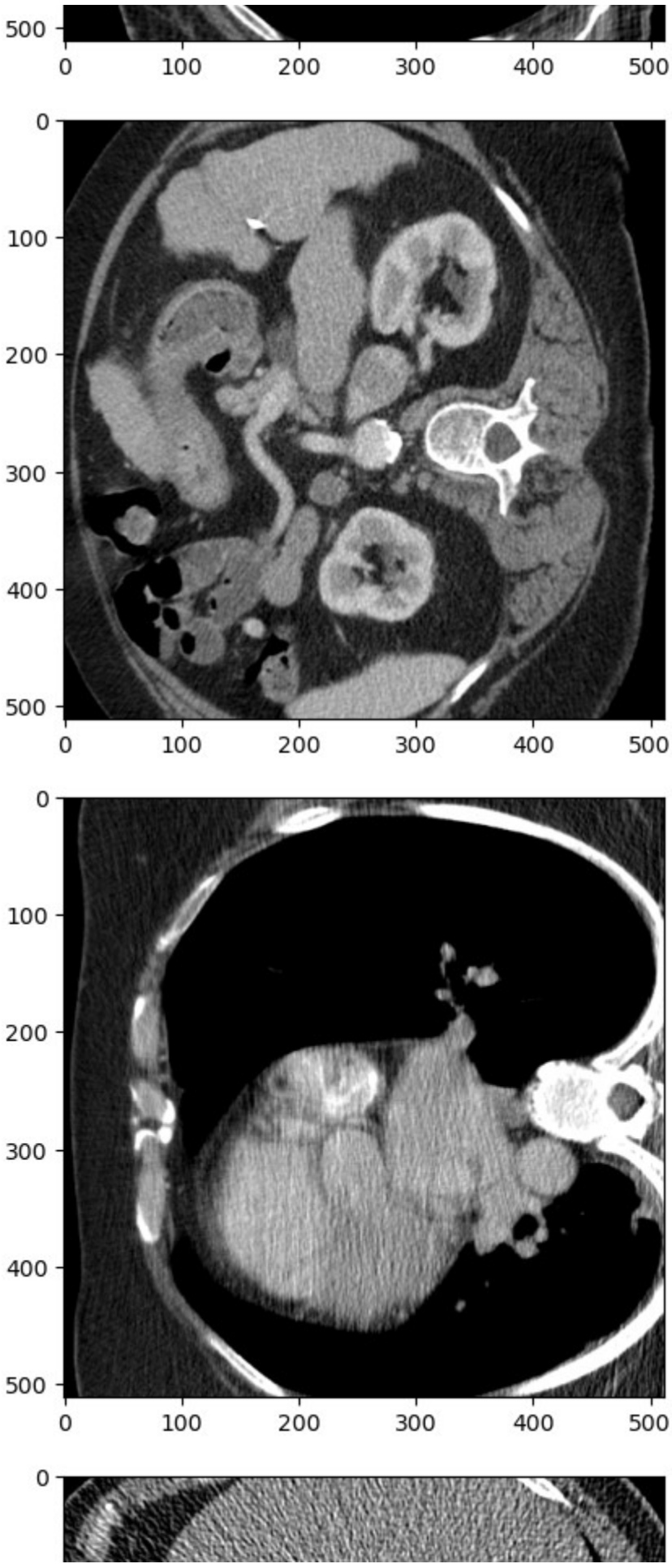

0
100
200
300
400
500
0
100
200
300
400
500

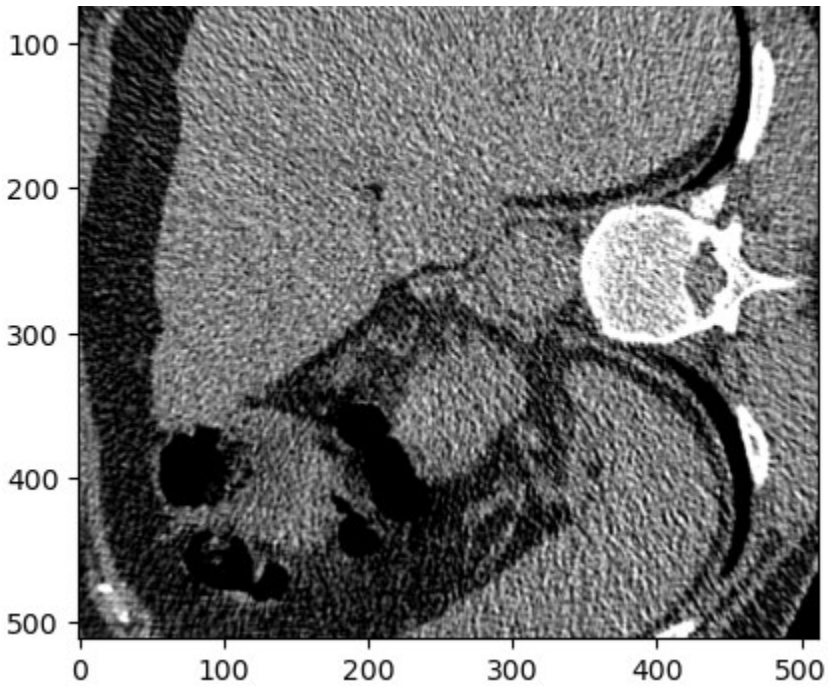

0
100
200
300

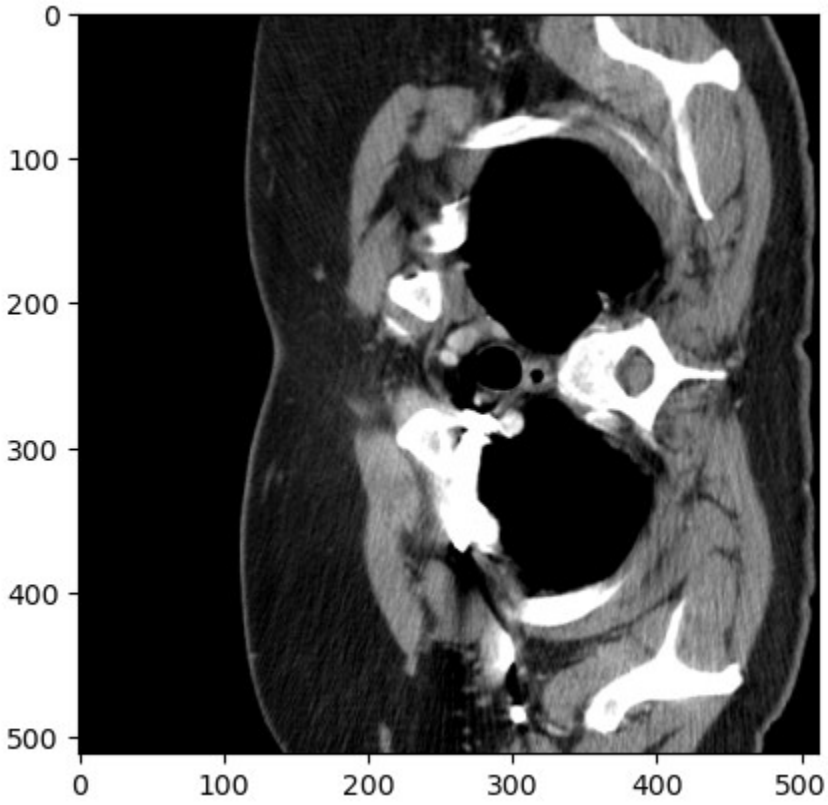

400
500
0
100
200
300
400
500

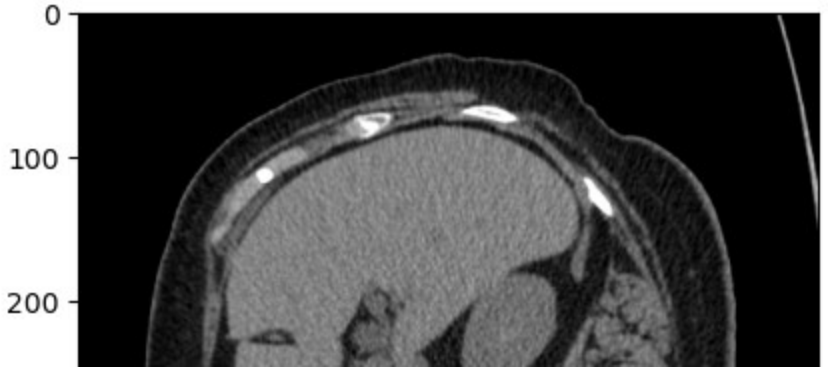

0
100
200
300
400
500
0
100
200
300
400
500

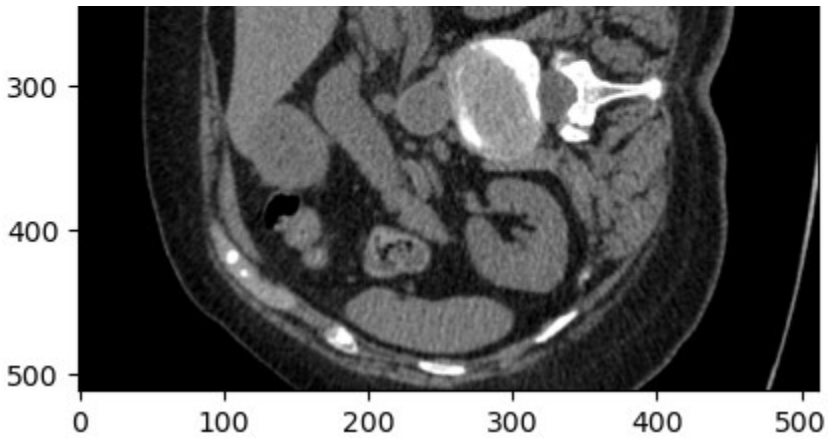

0
100
200
300
400

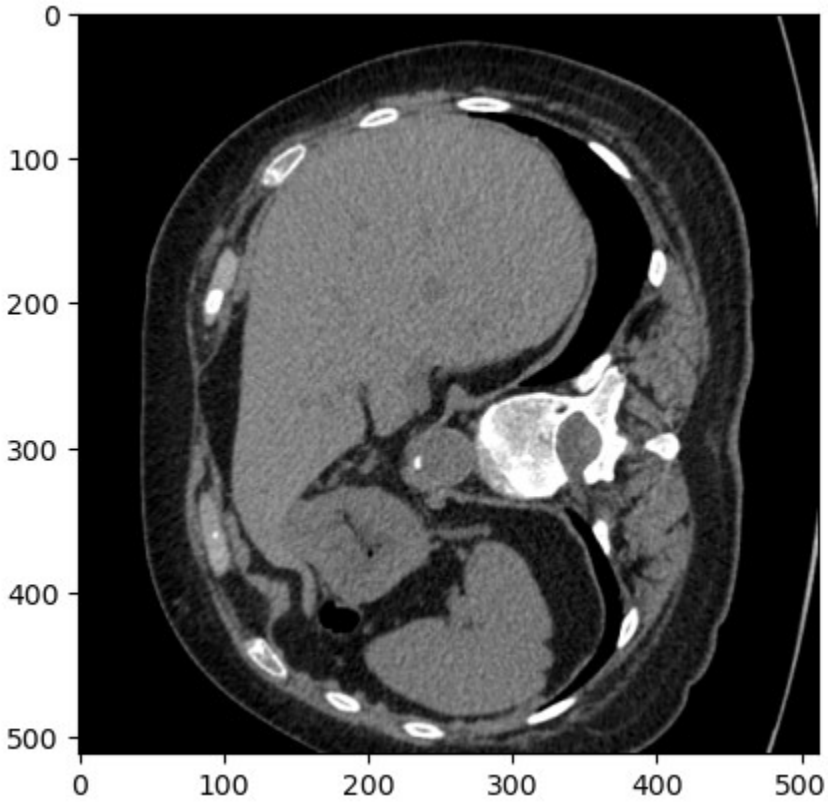

0
100
200
300
400
500


100
200
300
400
500
0
100
200
300
400
500

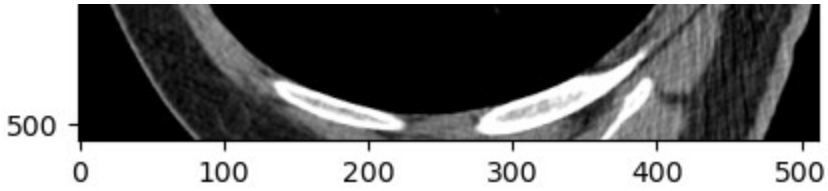

0
100
200
300
400
500
0
100
200
300
400
500

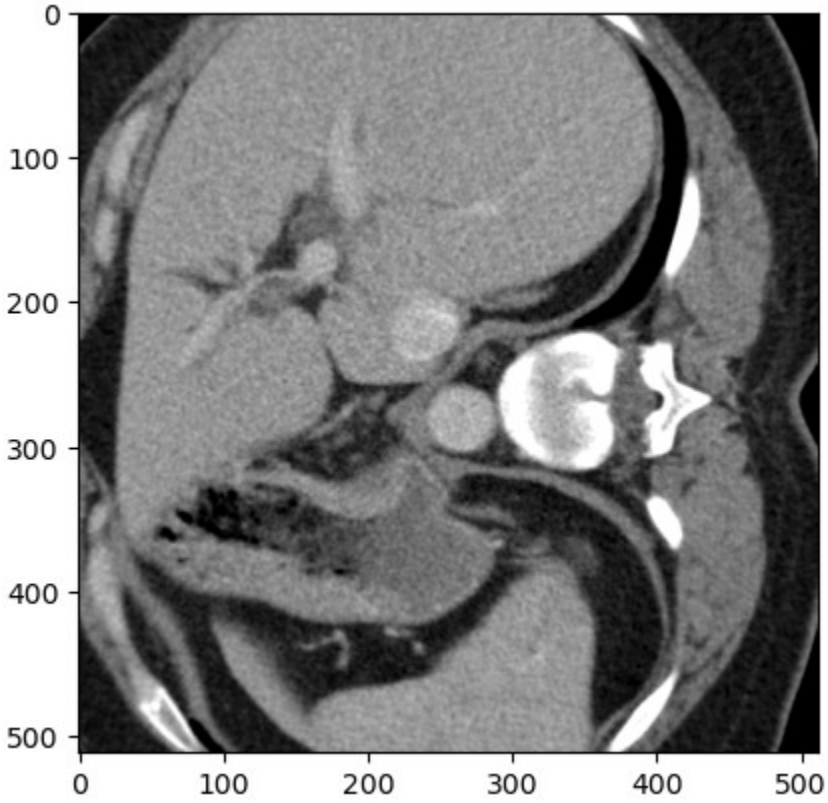

0
100
200

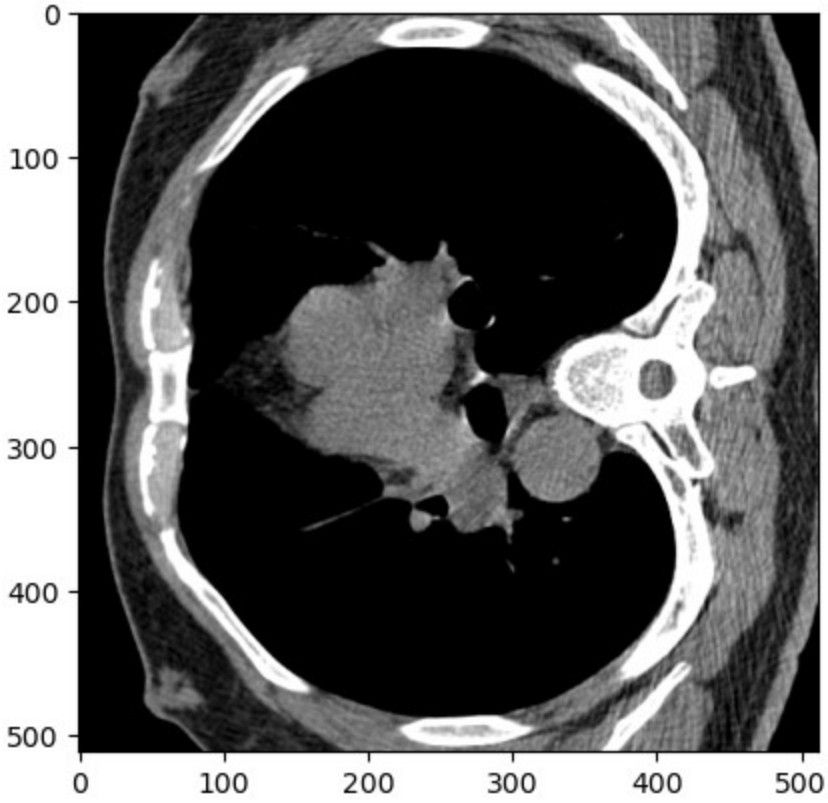

300
400
500
0
100
200
300
400
500


0
100
200
300
400
500
0
100
200
300
400
500

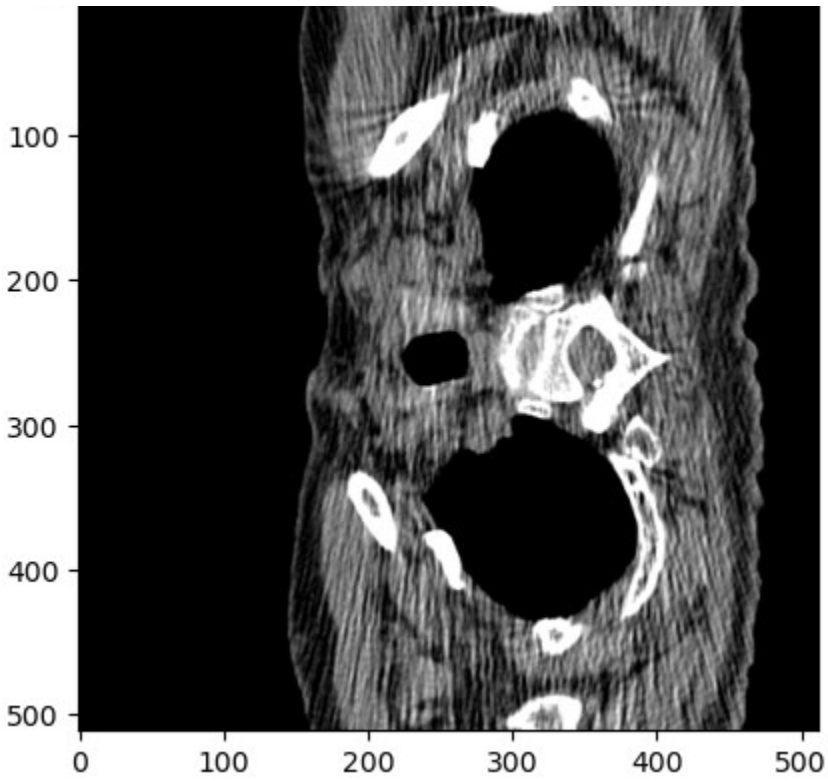

0
100
200
300
400

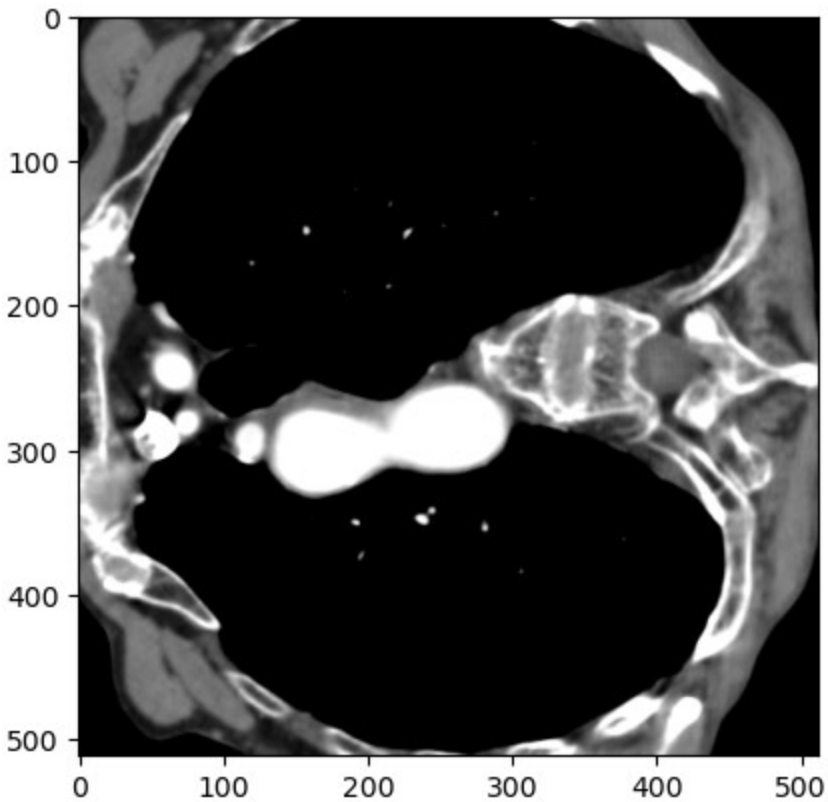

500
0
100
200
300
400
500

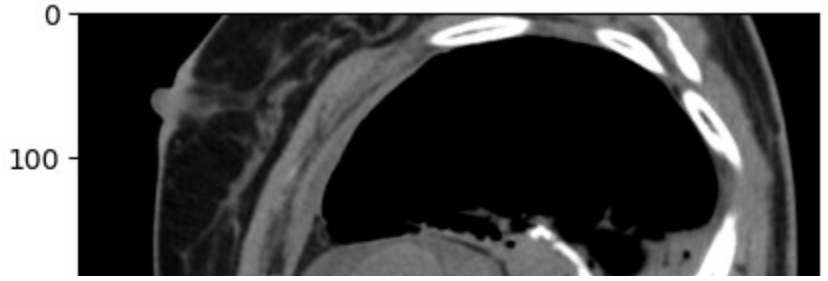

0
100
200
300
400
500
0
100
200
300
400
500

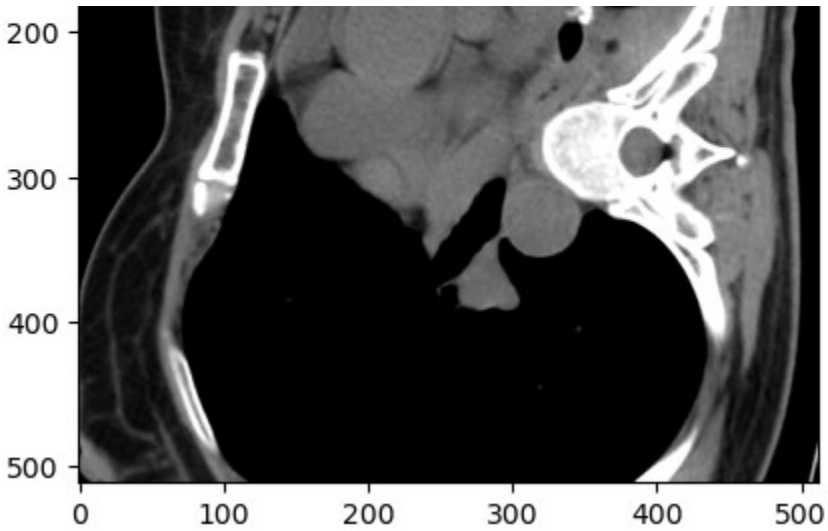

0
100
200
300
400
500
0
100
200
300
400
500

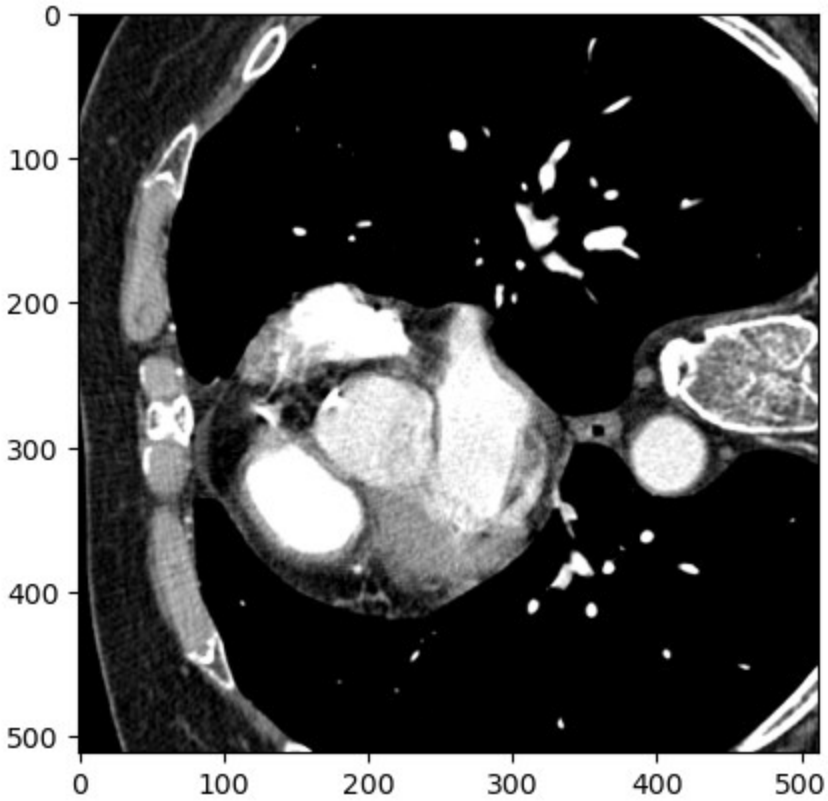

0

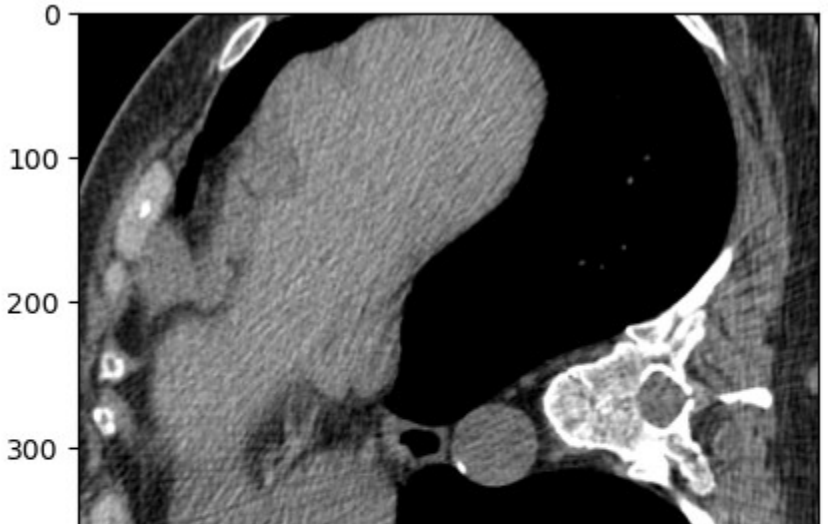

100
200
300
400
500
0
100
200
300
400
500

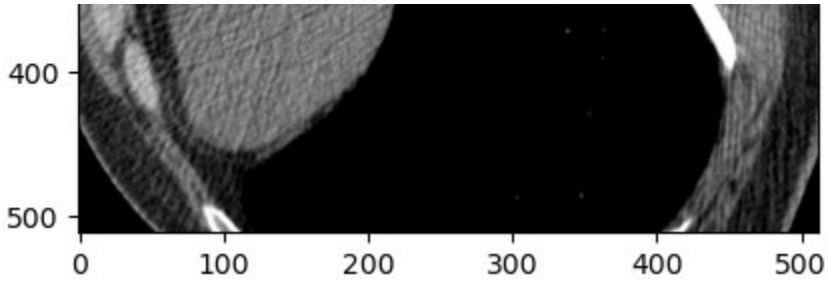

0
100
200
300
400
500
0
100
200
300
400
500

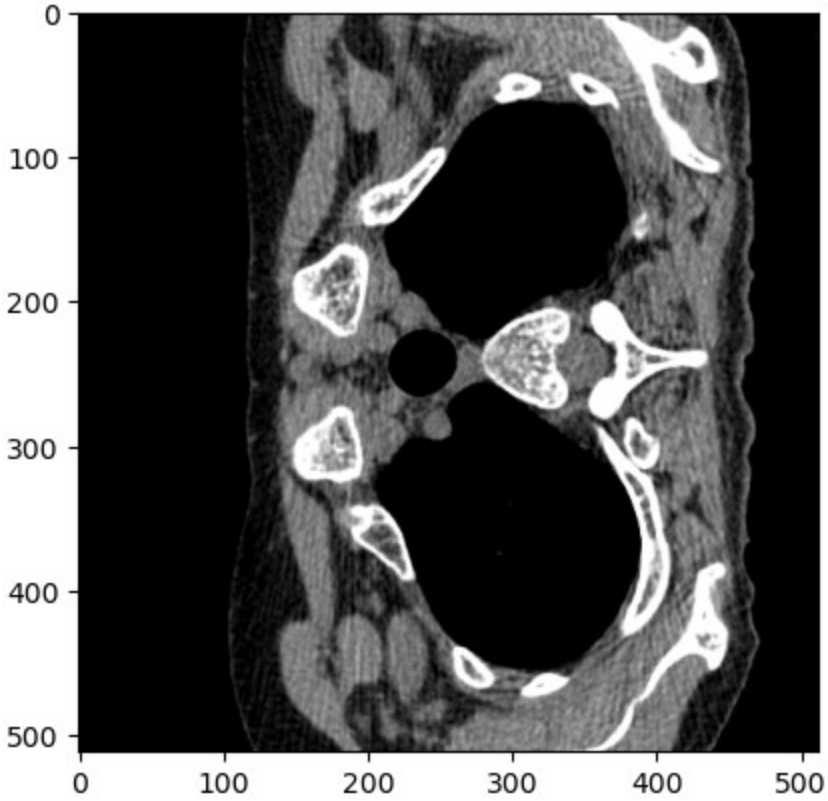

0
100

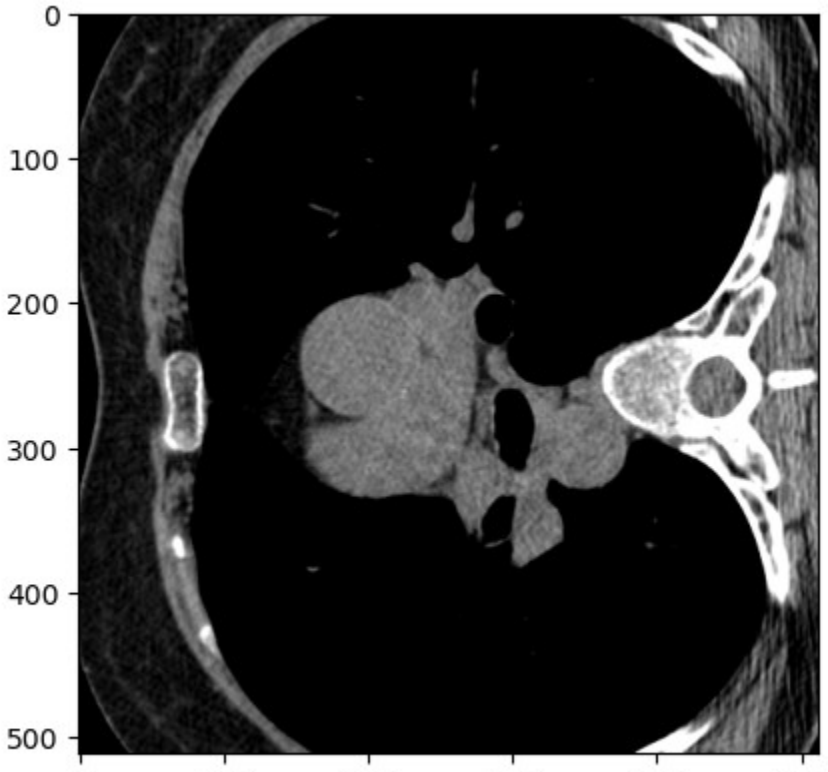

200
300
400
500
0
100
200
300
400
500

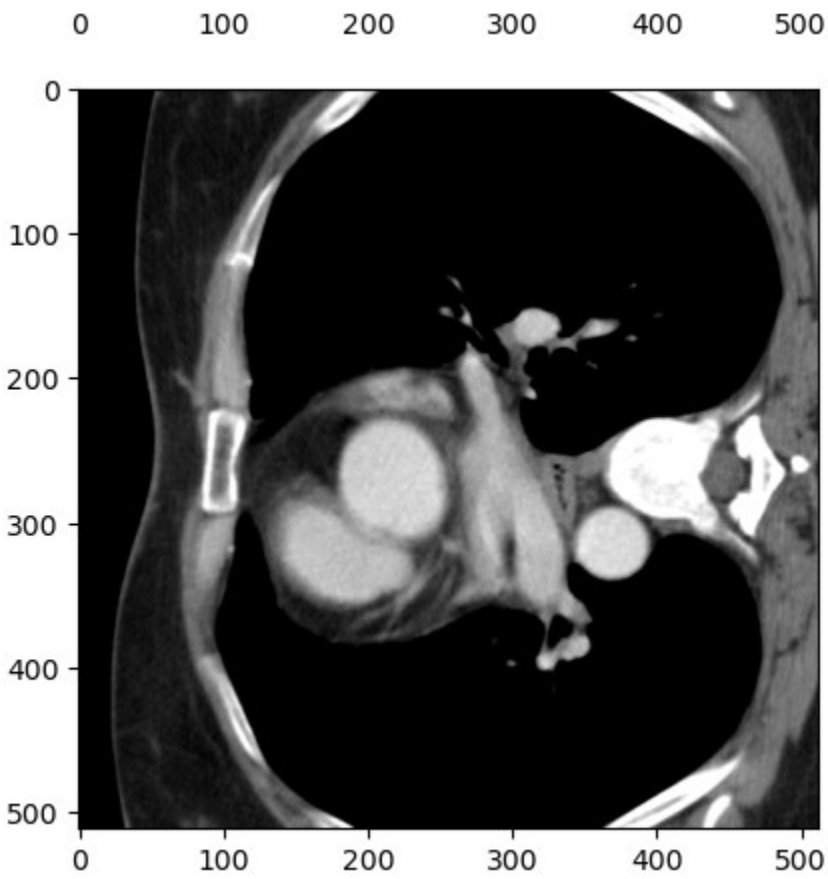

0
100
200
300
400
500
0
100
200
300
400
500

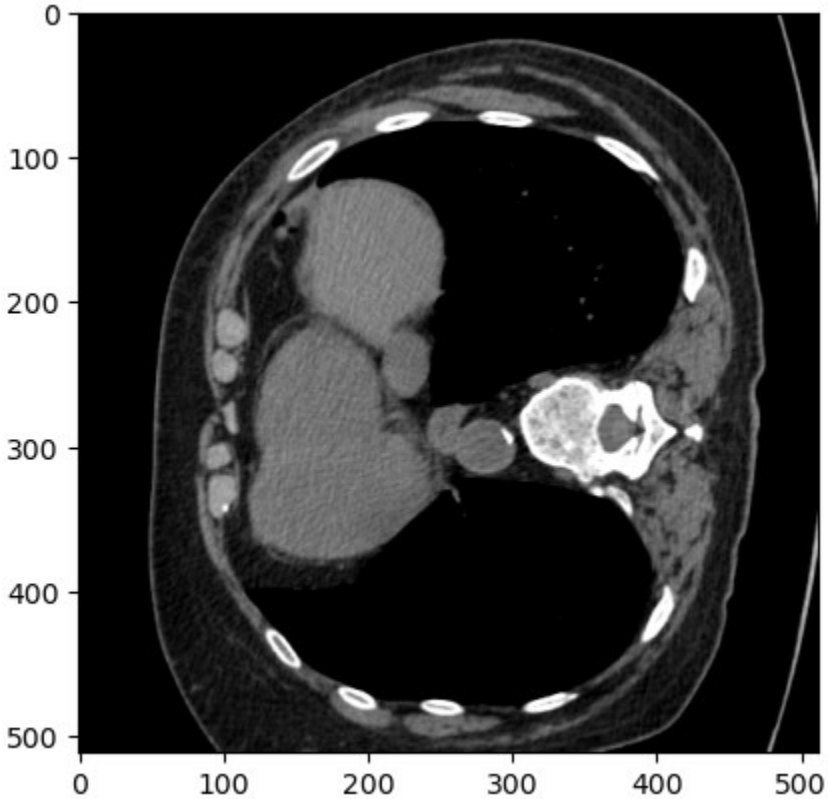

0
100
200
300

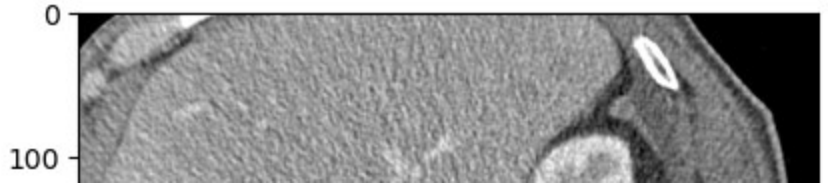

400
500
0
100
200
300
400
500

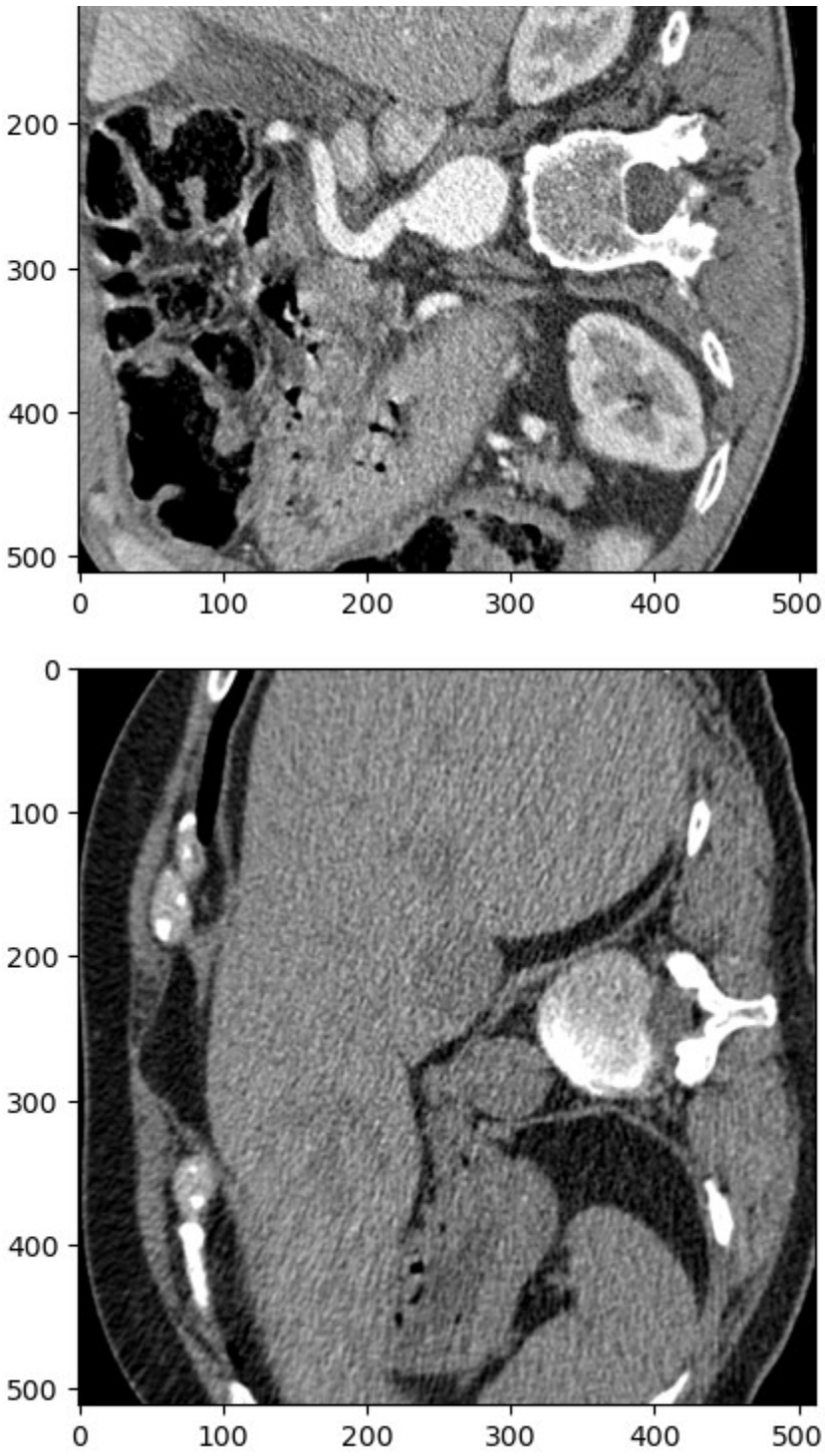

0
100
200
300
400
500
0
100
200
300
400
500


0
100
200
300
400
500


0
100
200
300
400
500


0
100
200
300
400
500


0
100



Part III : Setup for Inference

```
# Inference
def inference_single_image(model, noisy_image):
    input_image = np.expand_dims(noisy_image, axis=0)
    predicted_image = model.predict(input_image)
    a = np.abs(np.min(predicted_image))
    b = np.max(predicted_image)
```

```
    #predicted_image = predicted_image * (b - a) + a
    return predicted_image[0]

def inference_batch_images(model, noisy_images):
    input_image = noisy_images

    predicted_image = model.predict(input_image).astype(np.float64)
    return predicted_image


def rgb2gray(rgb):
    return np.expand_dims(np.dot(rgb[...,:3], [0.2989, 0.5870, 0.1140]), axis=-1)

from skimage.metrics import structural_similarity as ssim
from skimage.metrics import peak_signal_noise_ratio
import sys
sys.path.append('../')

from metrics import compute_SSIM, compute_PSNR
from skimage.metrics import mean_squared_error  as mse

def calculate_psnr(original_image, reconstructed_image,range=400):
    return peak_signal_noise_ratio(original_image, reconstructed_image,data_range=range)

    psnr_value = peak_signal_noise_ratio(original_image, reconstructed_image, data_range=240+160)
    return psnr_value

def calculate_ssim(original_image, reconstructed_image, range=400.0):
    ssim_value = ssim(original_image.astype(np.int16), reconstructed_image.astype(np.int16), win_size=11
    return ssim_value

def calculate_rmse(original_image, reconstructed_image):
    return mse(original_image, reconstructed_image)

def visualize_predictions(model, X_test,  n, predictions, model_name):
    random_numbers = list(range(n)) # not very random
    for i in random_numbers:
        gt_image= X_test[i].astype(np.float16)
        predicted_image = predictions[i].astype(np.float16)

        if predicted_image.shape[-1] == 3:
            predicted_image = rgb2gray(predicted_image)


        psnr_recon =  calculate_psnr(trunc(denormalize(gt_image)), trunc(denormalize(predicted_image)))
        ssim_recon = calculate_ssim(trunc(denormalize(gt_image)),  trunc(denormalize(predicted_image)))
        rmse_recon = calculate_rmse(trunc(denormalize(gt_image)),  trunc(denormalize(predicted_image)))

        psnr_recon = round(psnr_recon, 4)
        ssim_recon = round(ssim_recon, 4)
        rmse_recon = round(rmse_recon, 4)

        f, axarr = plt.subplots(1,2, figsize=(21,21))

        axarr[0].imshow(trunc(denormalize(gt_image)), cmap='gray', vmin=-160.0, vmax=240.0)
        axarr[0].set_title("QD Image")
        axarr[0].set_axis_off()
        axarr[1].imshow(trunc(denormalize(predicted_image)), cmap='gray', vmin=-160.0, vmax=240.0)
```

```
        axarr[1].imshow(trunc(denormalize(predicted_image)), cmap='gray', vmin=-160.0, vmax=240.0)
        axarr[1].set_title("{} Predicted Image : PSNR={}\nSSIM={}\nRMSE={}".format(model_name, psnr_recc
        axarr[1].set_axis_off()

        plt.show()

from skimage.metrics import peak_signal_noise_ratio

def get_average_metrics(predicted_images,  _noisy_array):
    psnr_original_mean = 0
    psnr_prediction_mean = 0

    ssim_original_mean = 0
    ssim_prediction_mean = 0

    mse_original_mean = 0
    mse_prediction_mean = 0

    i = 0
    for  gt_img, predicted_img in zip(noisy_array, predicted_images):
        predicted_img=  predicted_images[i]
        if predicted_img.shape[-1] == 3:
            predicted_img = rgb2gray(predicted_img)

        psnr_recon =  calculate_psnr(trunc(denormalize(gt_img)), trunc(denormalize(predicted_img)))
        ssim_recon = calculate_ssim(trunc(denormalize(gt_img)),  trunc(denormalize(predicted_img)))
        rmse_recon = calculate_rmse(trunc(denormalize(gt_img)),  trunc(denormalize(predicted_img)))

        psnr_prediction_mean += psnr_recon

        ssim_prediction_mean += ssim_recon

        mse_prediction_mean += rmse_recon

        i = i + 1

    psnr_prediction_mean/=noisy_array.shape[0]

    ssim_prediction_mean/=noisy_array.shape[0]

    mse_prediction_mean/=noisy_array.shape[0]

    print("Predicted average gt-predicted PSNR ->", psnr_prediction_mean)

    print("Predicted average gt-predicted SSIM ->", ssim_prediction_mean)

    print("Predicted average gt-predicted MSE->", mse_prediction_mean)

    return round(psnr_prediction_mean, 4), round(ssim_prediction_mean, 4), round(mse_prediction_mean, 4)
```

Part IV : Evaluation of each model

Model 1 : Hformer (for base reference)

```
sys.path.append('../denoising-models/hformer_vit/model/')
sys.path.append('../denoising-models/hformer_vit/')
from hformer_model_extended import get_hformer_model, PatchExtractor

hformer_model = get_hformer_model(num_channels_to_be_generated=64, name="hformer_model_extended")
hformer_model.build(input_shape=(None, 64, 64, 1))
hformer_model.load_weights('../denoising-models/hformer_vit/test/experiments/full_dataset/hformer_64_cha
print('Model summary : ')
print(hformer_model.summary())
```

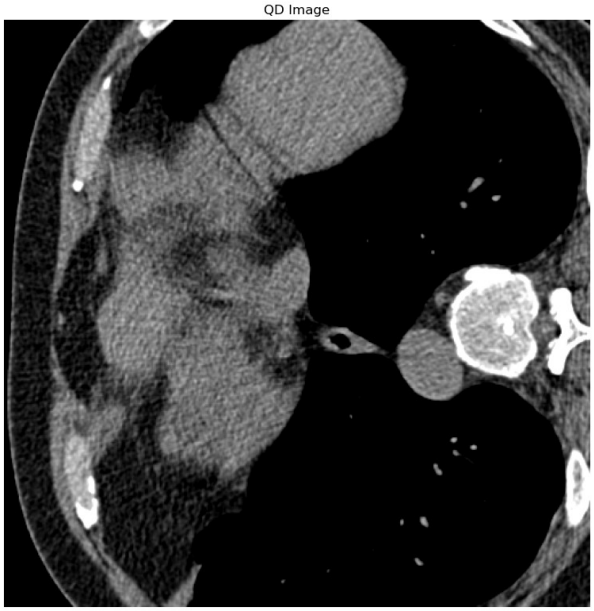
```
Model summary :
Model: "hformer_model_extended"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_projection_layer (Inp  multiple                 320
 utProjectionLayer)

 output_projection_layer (Ou  multiple                 257
 tputProjectionLayer)

 conv_net_block_1 (Convoluti  multiple                 36416
 onBlock)

 conv_net_block_2 (Convoluti  multiple                 36416
 onBlock)

 downsampling_layer_1 (Conv2  multiple                 32896
 D)

 downsampling_layer_2 (Conv2  multiple                 131328
 D)

 upsampling_layer_1 (Conv2DT  multiple                 131200
 ranspose)

 upsampling_layer_2 (Conv2DT  multiple                 32832
 ranspose)

 hformer_block_1 (HformerBlo  multiple                 187392
 ck)

 hformer_block_2 (HformerBlo  multiple                 735232
 ck)

 hformer_block_3 (HformerBlo  multiple                 187392
 ck)

=================================================================
Total params: 1,511,681
Trainable params: 1,511,681
Non-trainable params: 0
```

```
None
```

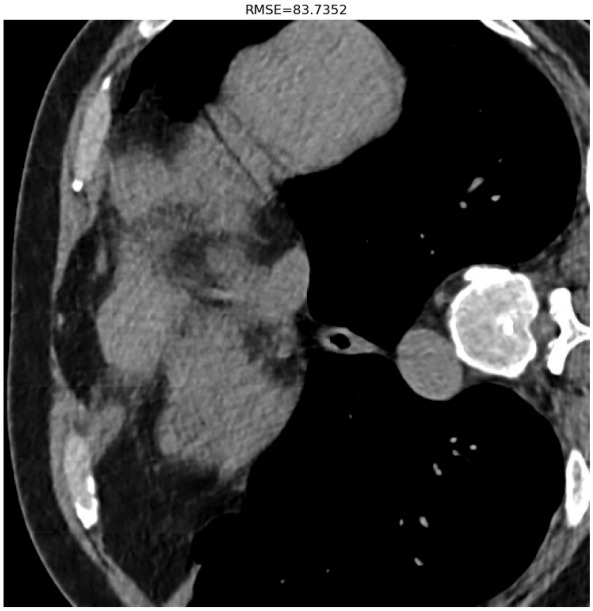
```
def reconstruct_image_from_patches(patches, num_patches_per_row):
    patch_size = patches.shape[1]  # Assuming square patches
    num_patches = patches.shape[0]

    # Calculate the number of rows
    num_patches_per_col = num_patches // num_patches_per_row

    # Initialize an empty image to store the reconstructed result
    reconstructed_image = np.zeros((num_patches_per_col * patch_size, num_patches_per_row * patch_size))

    # Reshape the patches into a 2D array
    patches_2d = patches.reshape((num_patches_per_col, num_patches_per_row, patch_size, patch_size))
    # Reconstruct the image by placing each patch in its corresponding position

    for i in range(num_patches_per_col):
        for j in range(num_patches_per_row):
            reconstructed_image[i * patch_size:(i + 1) * patch_size, j * patch_size:(j + 1) * patch_size

    return np.expand_dims(reconstructed_image, axis=-1)


# View the predictions
patch_extractor = PatchExtractor(patch_size=64, stride=64, name="patch_extractor")
noisy_image_patches_array = patch_extractor(noisy_array)

hformer_prediction_patches = hformer_model.predict(noisy_image_patches_array)
hformer_predictions = np.expand_dims(reconstruct_image_from_patches(hformer_prediction_patches[0:64], 8)

for i in range(1, int(hformer_prediction_patches.shape[0] / 64)):
    reconstructed_image = reconstruct_image_from_patches(hformer_prediction_patches[i * 64 : i * 64 + 64
    reconstructed_image = np.expand_dims(reconstructed_image, axis=0)

    hformer_predictions = np.append(hformer_predictions, reconstructed_image, axis=0)
visualize_predictions(hformer_predictions, noisy_array,  len(noisy_array), hformer_predictions, "hformer
```

```
56/56 [==============================] - 57s 980ms/step
```

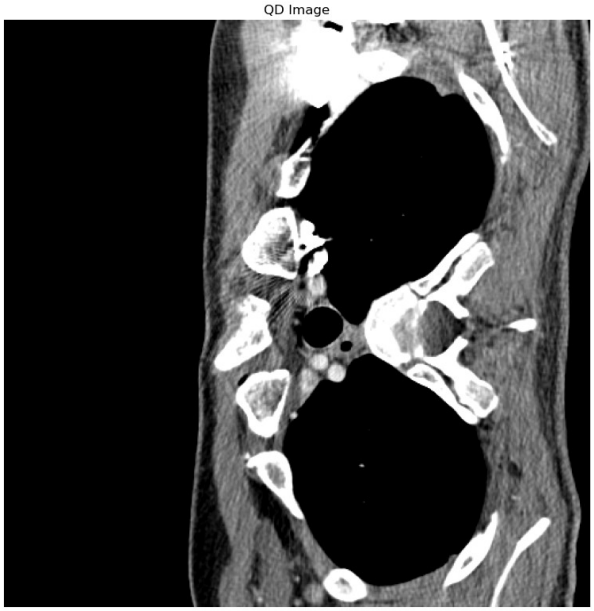
QD Image



hformer Predicted Image : PSNR=32.8121
SSIM=0.9265
RMSE=83.7352

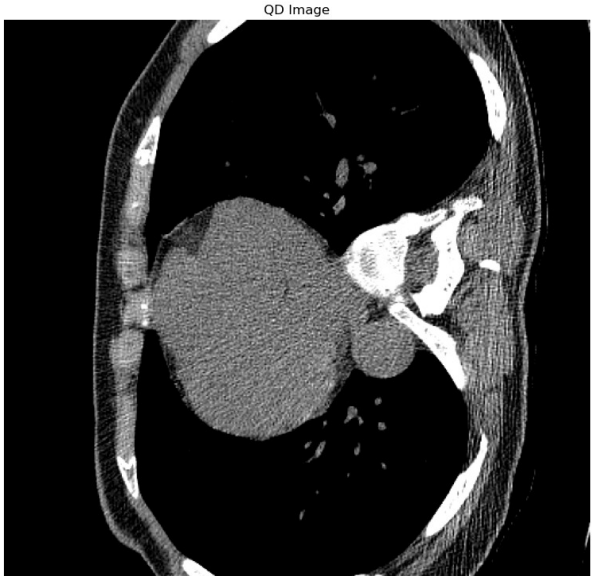


QD Image

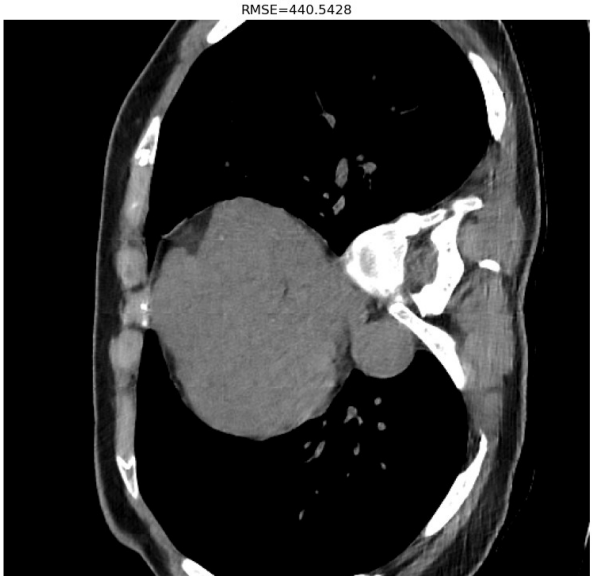


hformer Predicted Image : PSNR=28.3003
SSIM=0.9636
RMSE=236.6388

QD Image



hformer Predicted Image : PSNR=25.6013
SSIM=0.8788
RMSE=440.5428



QD Image

hformer Predicted Image : PSNR=41.7632
SSIM=0.9739
RMSE=10.6609

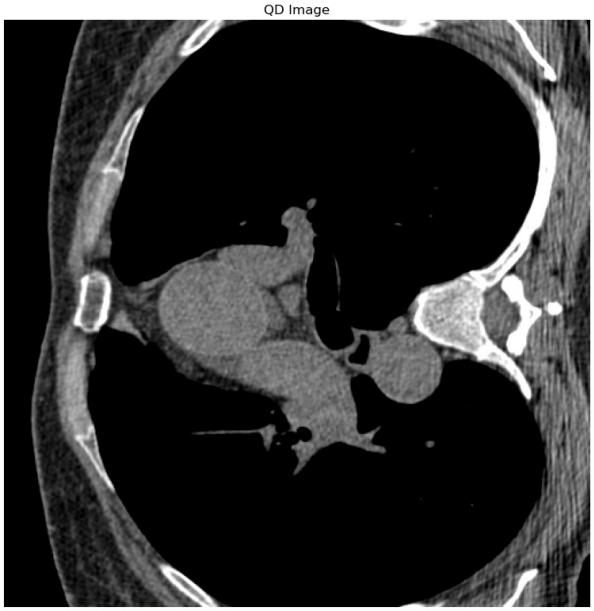
QD Image

hformer Predicted Image : PSNR=32.2629
SSIM=0.9453
RMSE=95.0229

QD Image

hformer Predicted Image : PSNR=26.4624
SSIM=0.8105
RMSE=361.3108

QD Image

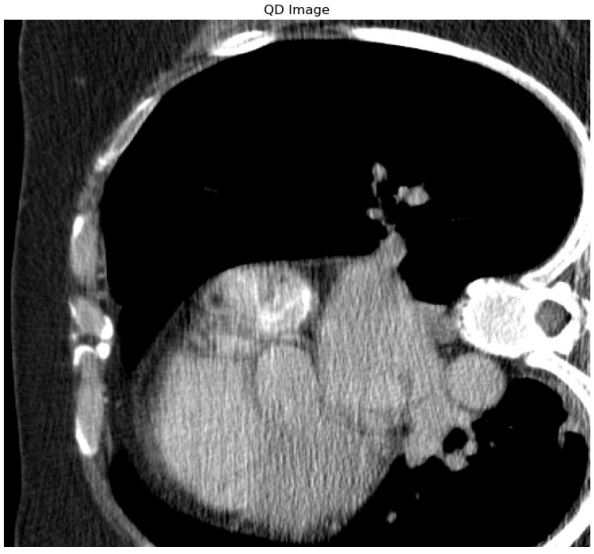


hformer Predicted Image : PSNR=35.6345
SSIM=0.9714
RMSE=43.7192

QD Image

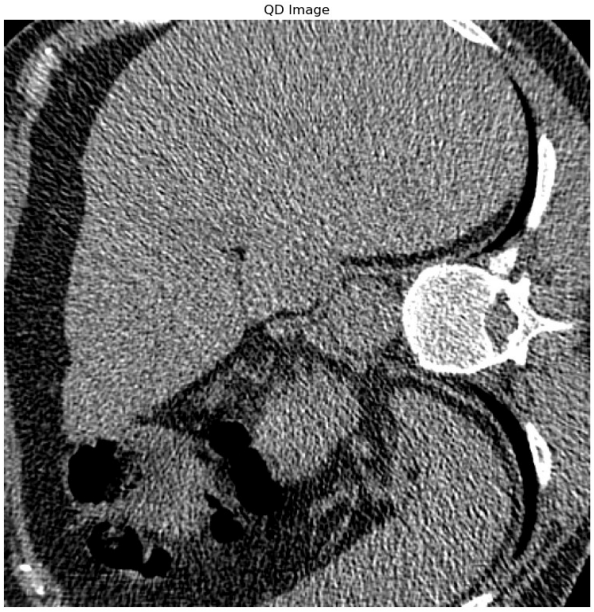
hformer Predicted Image : PSNR=32.3595
SSIM=0.8886
RMSE=92.9333

QD Image

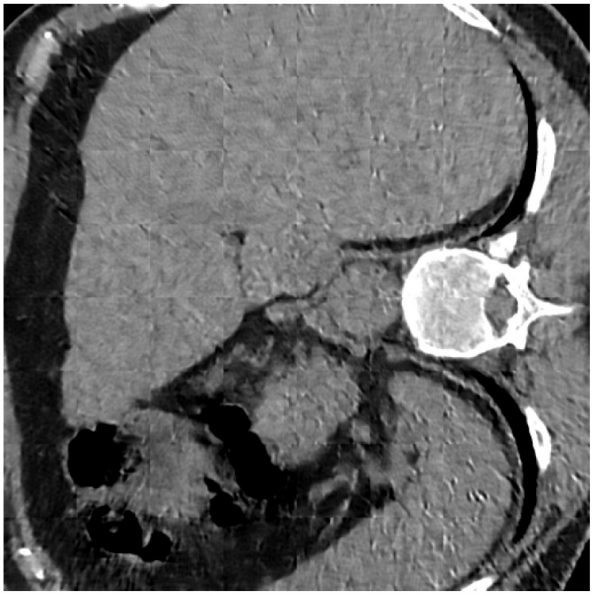


hformer Predicted Image : PSNR=31.1293
SSIM=0.902
RMSE=123.3631

QD Image

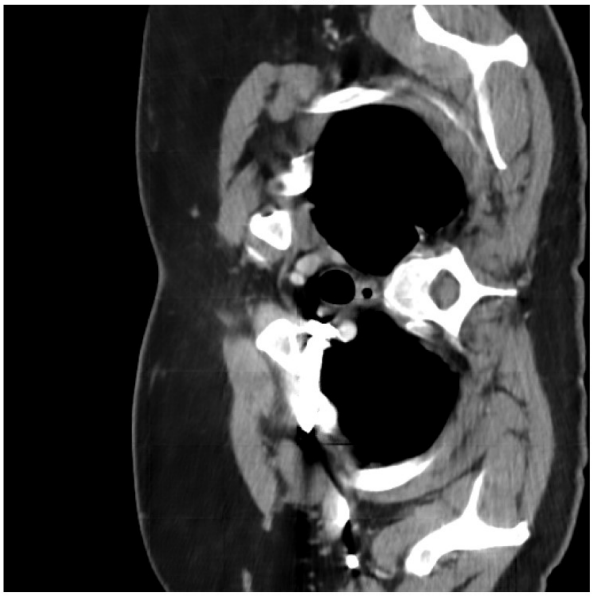


hformer Predicted Image : PSNR=19.1266
SSIM=0.5663
RMSE=1956.3892



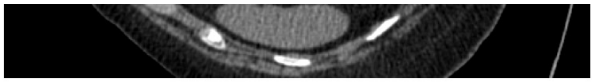
QD Image

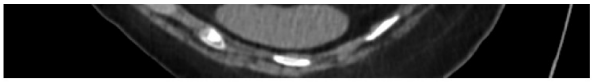
hformer Predicted Image : PSNR=34.4729
SSIM=0.9316
RMSE=57.126



QD Image

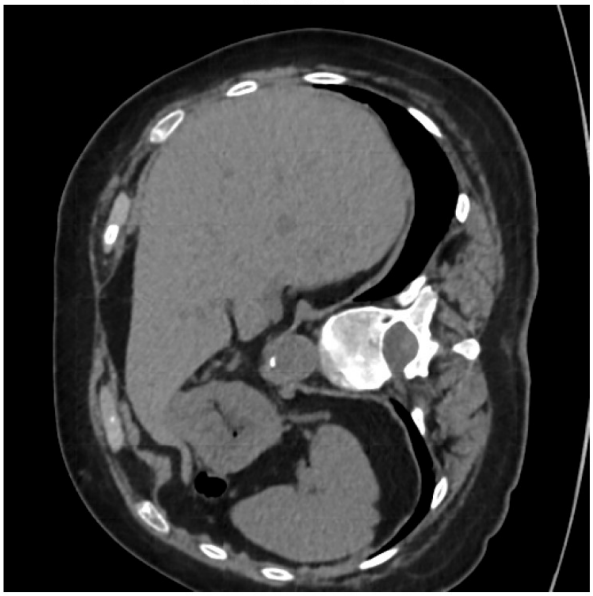
hformer Predicted Image : PSNR=34.1448
SSIM=0.9281
RMSE=61.6084



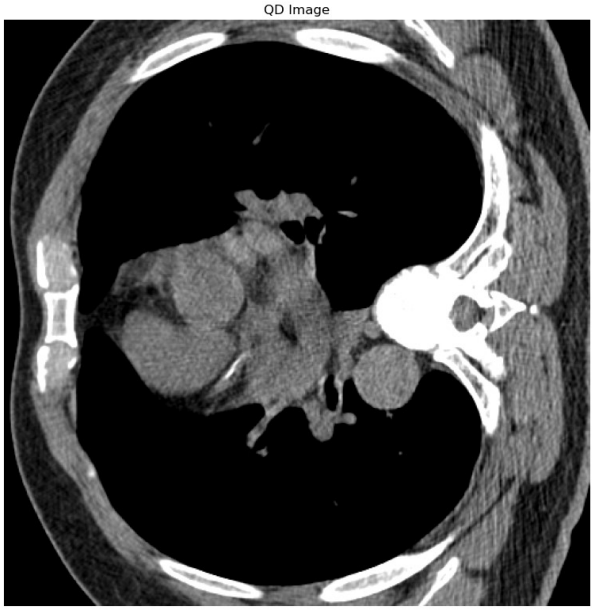


QD Image

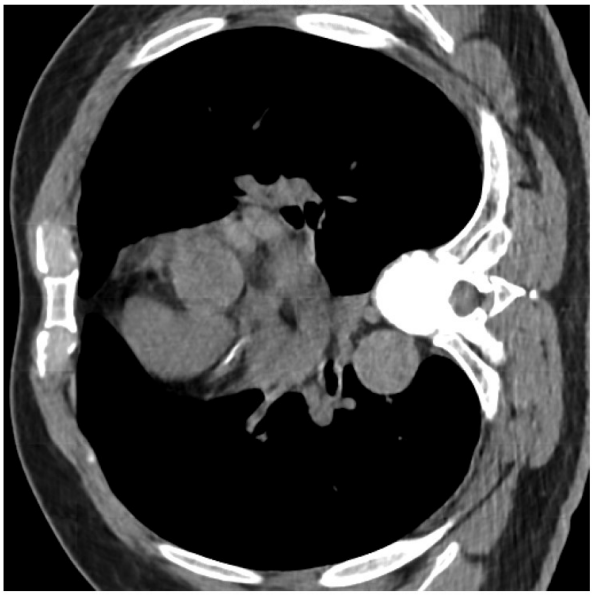
hformer Predicted Image : PSNR=34.0201
SSIM=0.9187
RMSE=63.4027



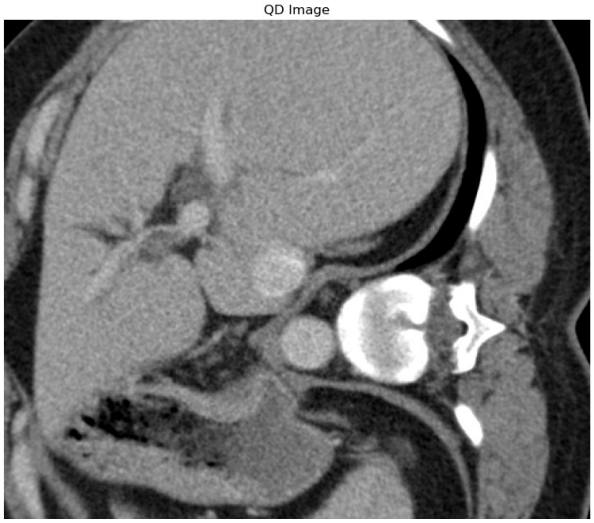
QD Image



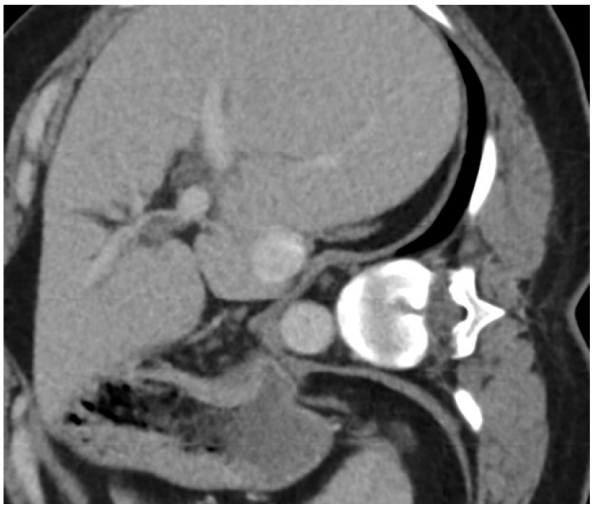
hformer Predicted Image : PSNR=34.6176
SSIM=0.9598
RMSE=55.2531



QD Image



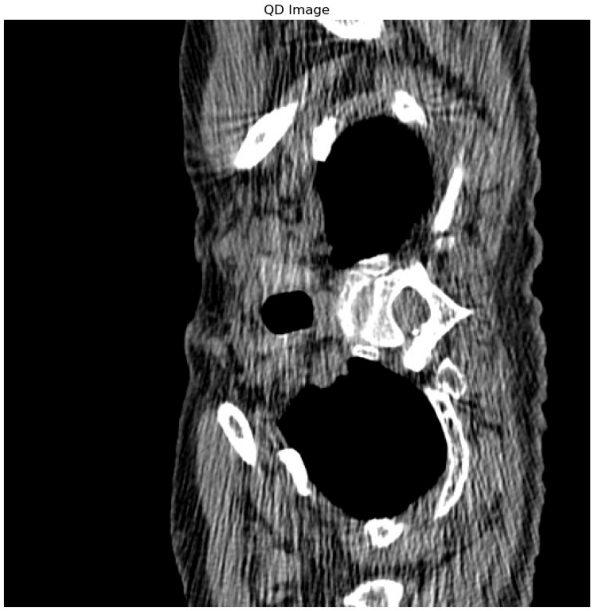
hformer Predicted Image : PSNR=35.9607
SSIM=0.9305
RMSE=40.5553



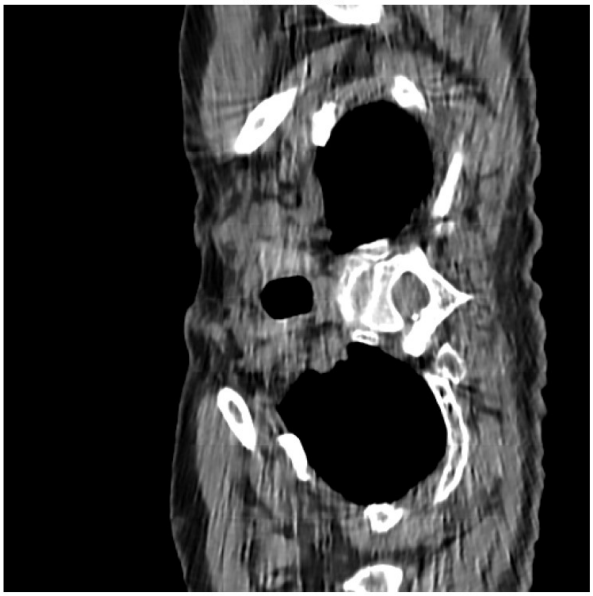
QD Image

hformer Predicted Image : PSNR=34.3789
SSIM=0.9581
RMSE=58.3756

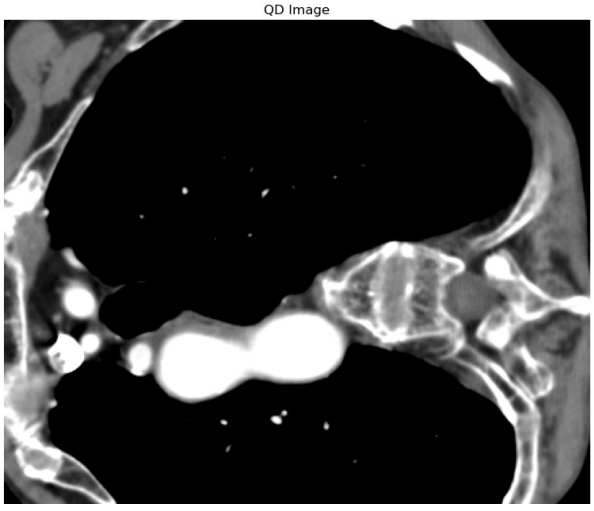
QD Image



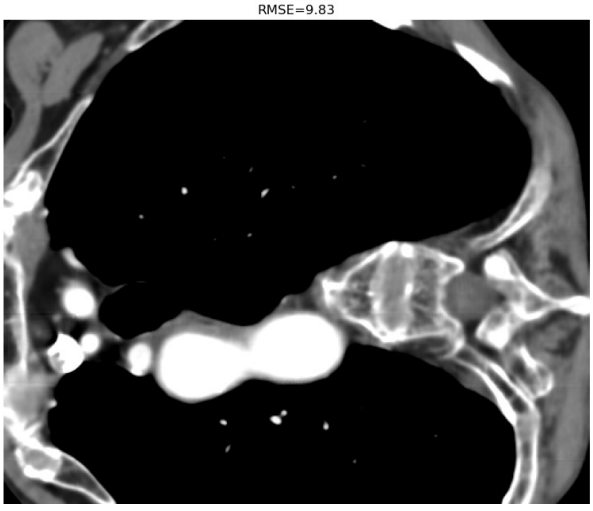
hformer Predicted Image : PSNR=28.553
SSIM=0.9306
RMSE=223.267



QD Image

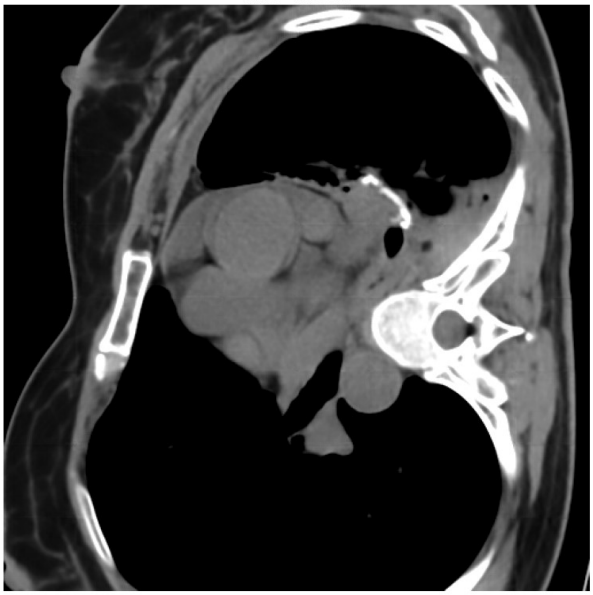


hformer Predicted Image : PSNR=42.1157
SSIM=0.9939
RMSE=9.83



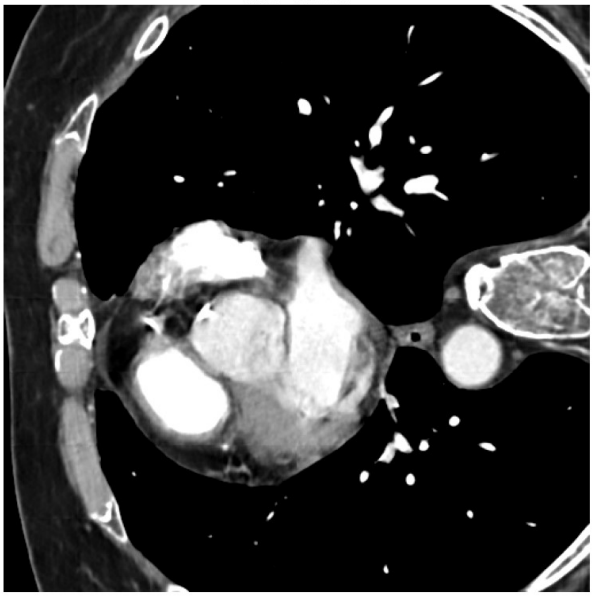
QD Image

hformer Predicted Image : PSNR=40.8744
SSIM=0.9838
RMSE=13.0821

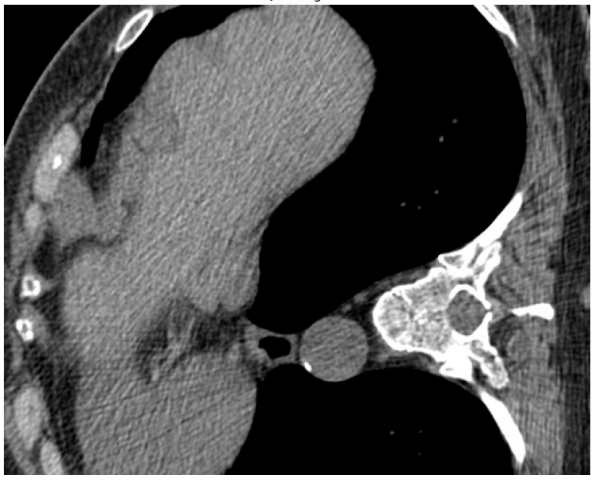


QD Image

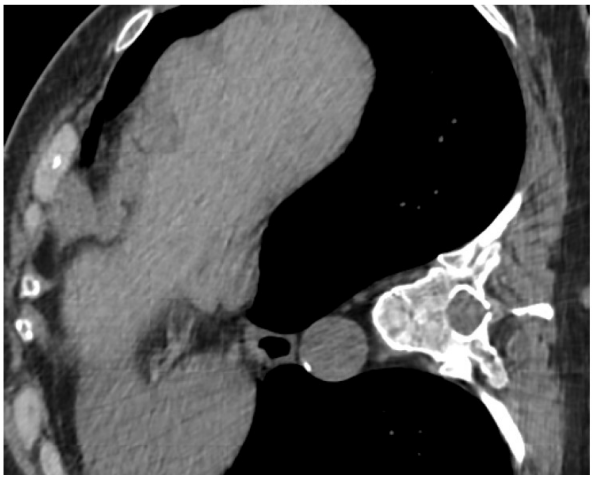
hformer Predicted Image : PSNR=34.5645
SSIM=0.9593
RMSE=55.9333



QD Image

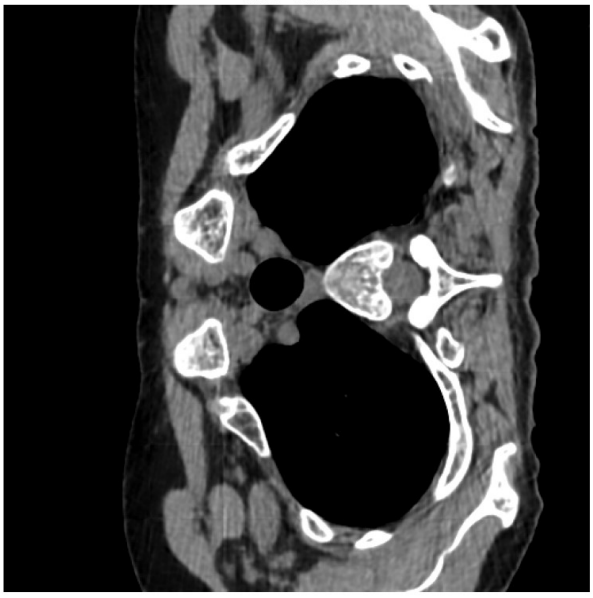


hformer Predicted Image : PSNR=32.5687
SSIM=0.9158
RMSE=88.5616

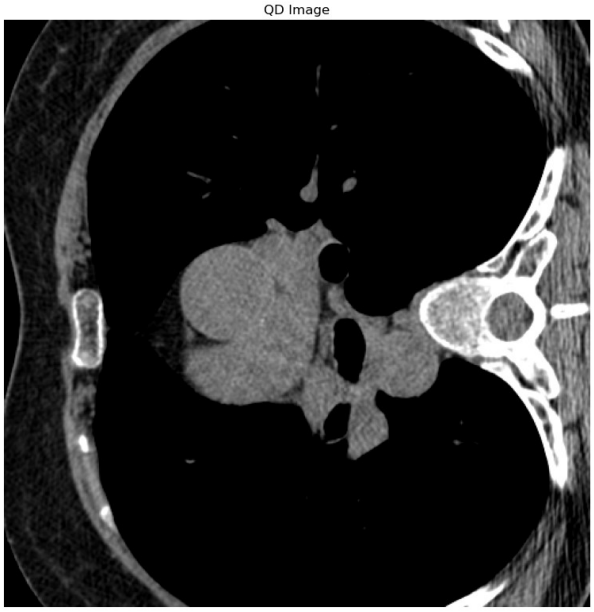


QD Image

hformer Predicted Image : PSNR=35.0377
SSIM=0.9668
RMSE=50.1589



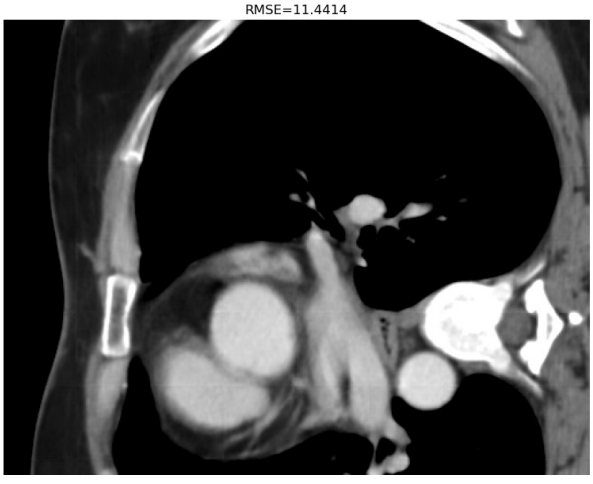
QD Image



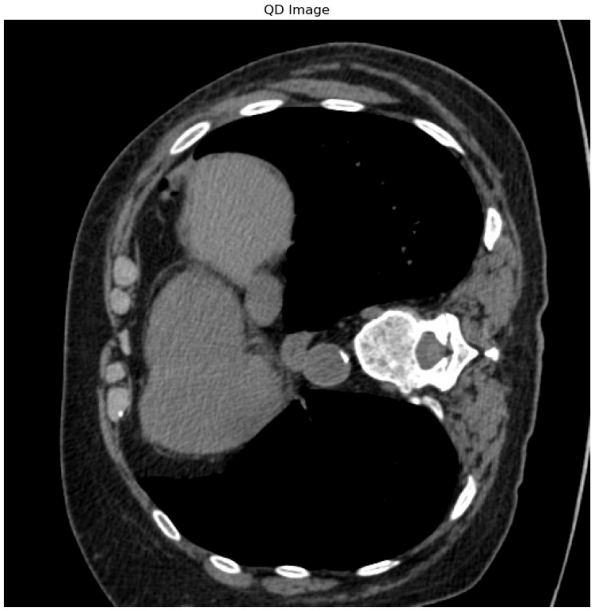
hformer Predicted Image : PSNR=35.7783
SSIM=0.9599
RMSE=42.2955

QD Image

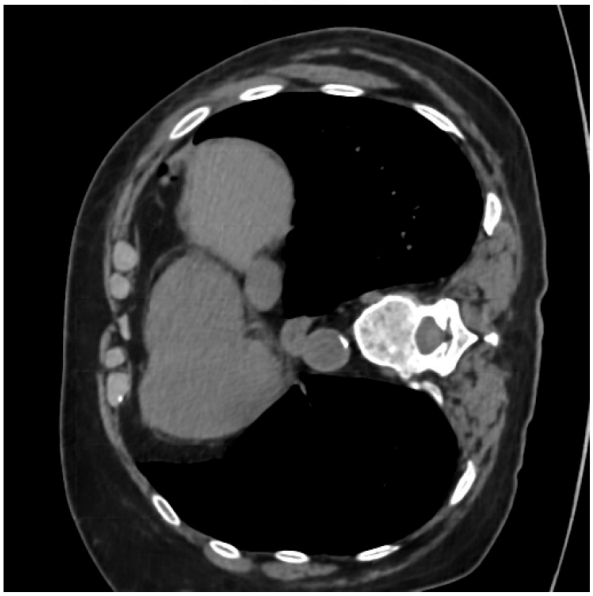
hformer Predicted Image : PSNR=41.4564
SSIM=0.9832
RMSE=11.4414



QD Image



hformer Predicted Image : PSNR=37.1773
SSIM=0.9624
RMSE=30.6474

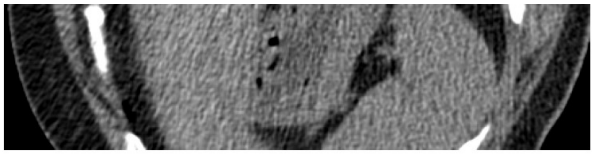


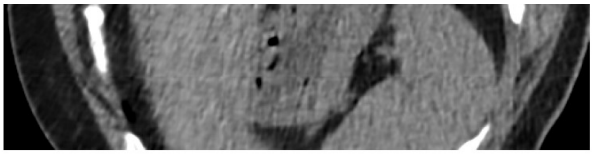
QD Image

hformer Predicted Image : PSNR=26.9225
SSIM=0.8118
RMSE=324.9903

QD Image

hformer Predicted Image : PSNR=29.2817
SSIM=0.8456
RMSE=188.7754





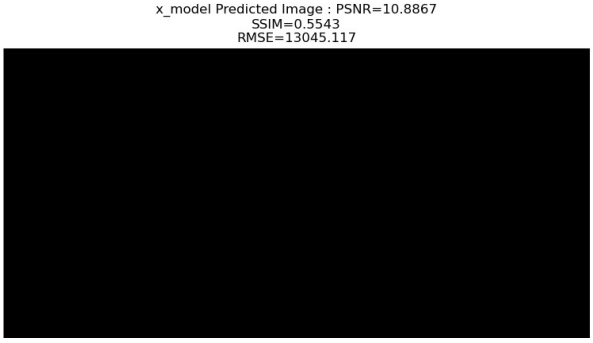
Model 2 : XModel

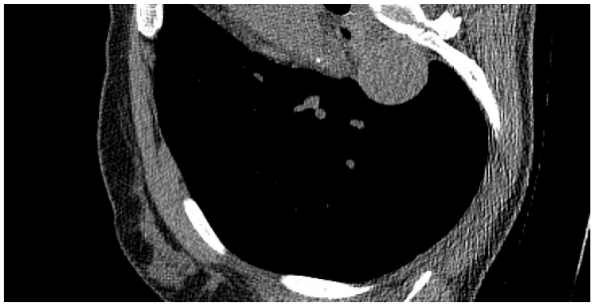
```
from torchinfo import summary
sys.path.append('../denoising-models/novel_model/')
from x_denoiser import XModel
x_model = XModel(num_channels=64)
x_model.load_state_dict(torch.load('../denoising-models/novel_model/weights/x_model/model_19.pth'))
x_model.eval()
print('Model summary : ')
print(summary(x_model))
```

```
Model summary :
==========================================================================
Layer (type:depth-idx)                                   Param #
==========================================================================
XModel                                                   --
├─ConvTranspose2d: 1-1                                   5
├─Conv2d: 1-2                                            16,448
├─InputProjectionLayer: 1-3                              --
│    └─Conv2d: 2-1                                       640
├─OutputProjectionLayer: 1-4                             --
│    └─ConvTranspose2d: 2-2                              577
├─ConvBlock: 1-5                                         --
│    └─ZeroPad2d: 2-3                                    --
│    └─DepthWiseSeparableConv2d: 2-4                     --
│    │    └─Conv2d: 3-1                                  3,136
│    │    └─Conv2d: 3-2                                  4,096
│    └─LayerNorm: 2-5                                    --
│    └─Conv2d: 2-6                                       803,072
│    └─GELU: 2-7                                         --
│    └─Conv2d: 2-8                                       16,448
├─ConvBlock: 1-6                                         --
│    └─ZeroPad2d: 2-9                                    --
│    └─DepthWiseSeparableConv2d: 2-10                    --
│    │    └─Conv2d: 3-3                                  3,136
│    │    └─Conv2d: 3-4                                  4,096
│    └─LayerNorm: 2-11                                   --
│    └─Conv2d: 2-12                                      803,072
│    └─GELU: 2-13                                        --
```

```
│    └─Conv2d: 2-14                       16,448
├─ConvBlock: 1-7                         --
│    └─ZeroPad2d: 2-15                    --
│    └─DepthWiseSeparableConv2d: 2-16     --
│    │    └─Conv2d: 3-5                   3,136
│    │    └─Conv2d: 3-6                   4,096
│    └─LayerNorm: 2-17                    --
│    └─Conv2d: 2-18                       803,072
│    └─GELU: 2-19                         --
│    └─Conv2d: 2-20                       16,448
├─ConvBlock: 1-8                         --
│    └─ZeroPad2d: 2-21                    --
│    └─DepthWiseSeparableConv2d: 2-22     --
│    │    └─Conv2d: 3-7                   3,136
│    │    └─Conv2d: 3-8                   4,096
│    └─LayerNorm: 2-23                    --
│    └─Conv2d: 2-24                       803,072
│    └─GELU: 2-25                         --
│    └─Conv2d: 2-26                       16,448
├─XBlock: 1-9                            --
│    └─Linear: 2-27                       4,160
│    └─Linear: 2-28                       4,160
│    └─Linear: 2-29                       4,160
│    └─Conv2d: 2-30                       4,160
│    └─LayerNorm: 2-31                    --
│    └─Conv2d: 2-32                       16,640
│    └─GELU: 2-33                         --
│    └─Conv2d: 2-34                       16,448
├─ConvBlock: 1-10                        --
│    └─ZeroPad2d: 2-35                    --
```

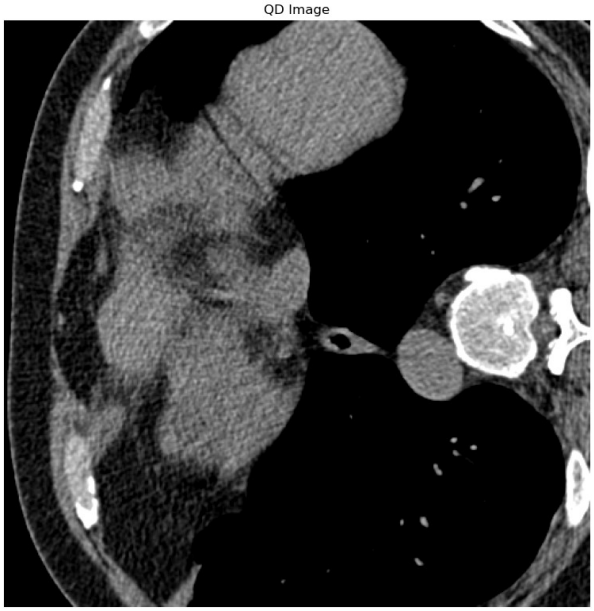
```
# Get prediction images.

x_model_prediction_patches = []

for img in noisy_array:
    img_tensor = torch.from_numpy(img).float()

    with torch.no_grad():
        output_tensor = x_model(img_tensor)
        output_tensor = output_tensor.cpu().numpy()

        x_model_prediction_patches.append(output_tensor)

visualize_predictions(x_model, noisy_array,  len(noisy_array), np.concatenate(x_model_prediction_patches
```





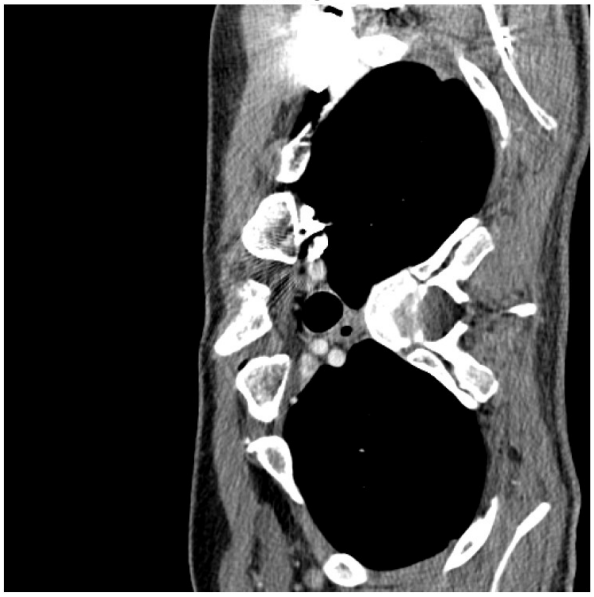
x_model Predicted Image : PSNR=10.1793
SSIM=0.3995
RMSE=15352.7121

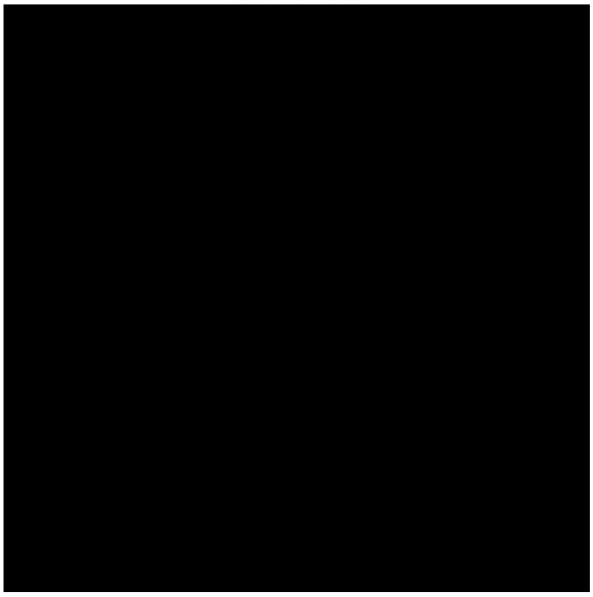
QD Image



x_model Predicted Image : PSNR=8.0612
SSIM=0.4824
RMSE=25003.3838

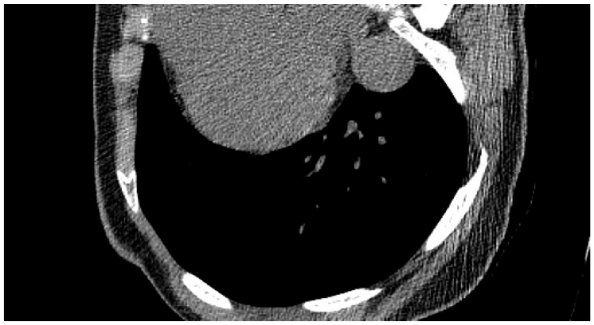
QD Image





x_model Predicted Image : PSNR=10.0961
SSIM=0.5335
RMSE=15649.8625

QD Image

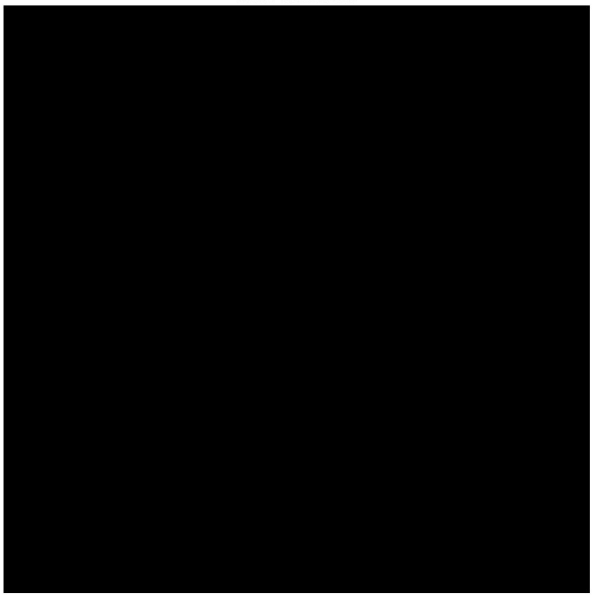


x_model Predicted Image : PSNR=5.2602
SSIM=-0.2802
RMSE=47654.2589

QD Image

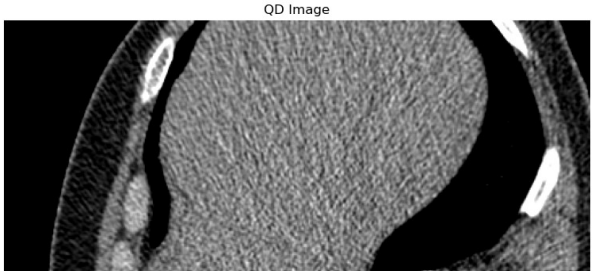
x_model Predicted Image : PSNR=8.6443
SSIM=0.4223
RMSE=21861.878

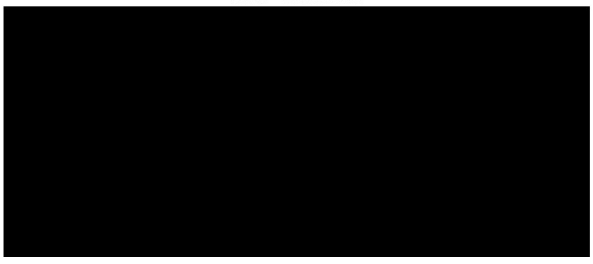
QD Image

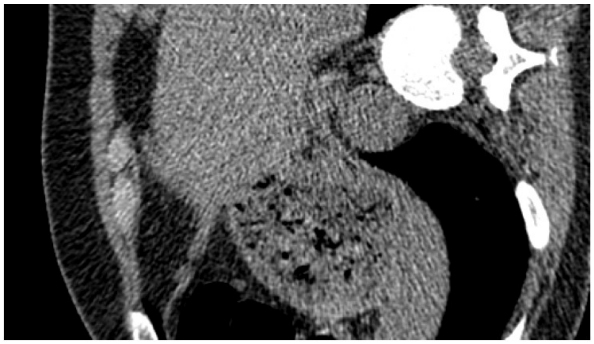


x_model Predicted Image : PSNR=7.6802
SSIM=0.1716
RMSE=27296.2699

QD Image

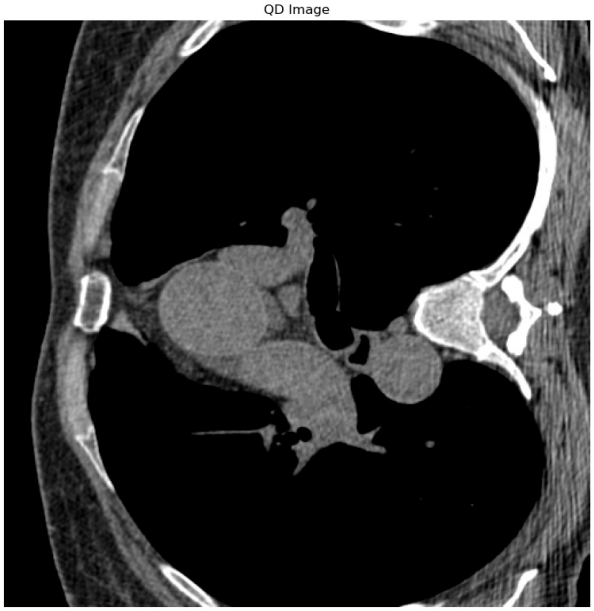






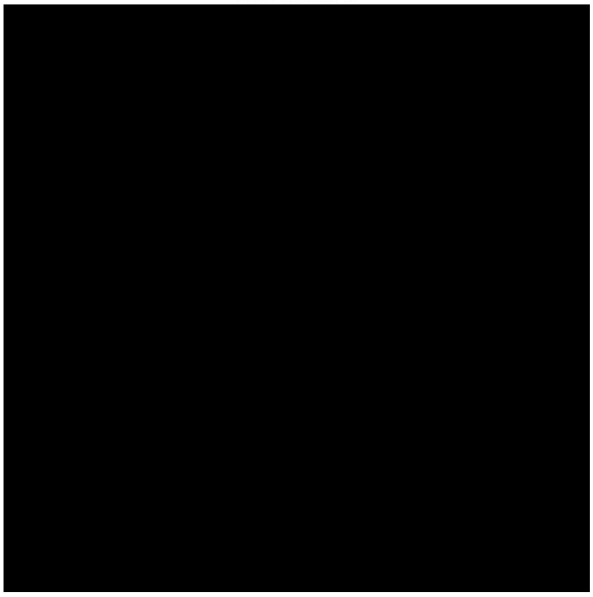
x_model Predicted Image : PSNR=10.8561
SSIM=0.5422
RMSE=13137.4863

QD Image



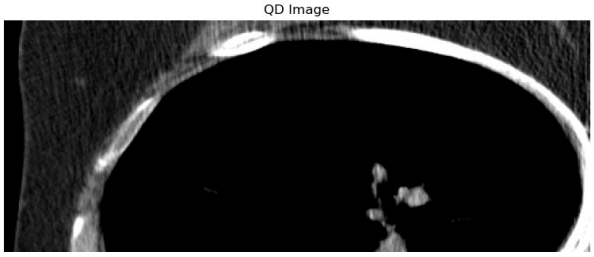
x_model Predicted Image : PSNR=7.9647
SSIM=0.0918
RMSE=25565.442

QD Image



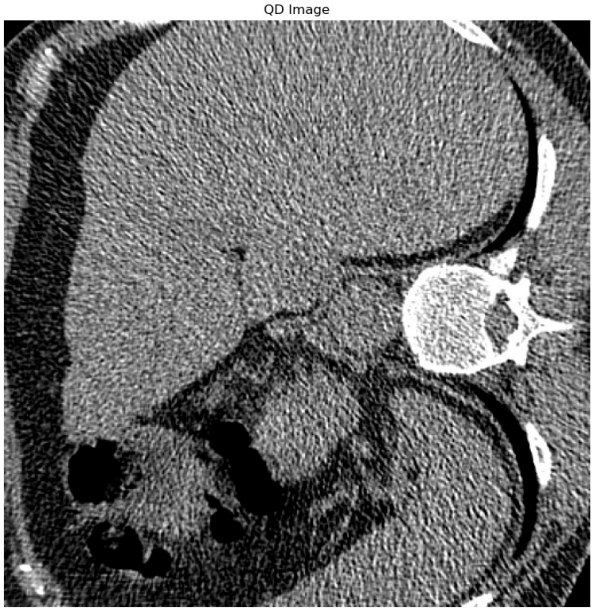
x_model Predicted Image : PSNR=8.3301
SSIM=0.4146
RMSE=23502.0094

QD Image



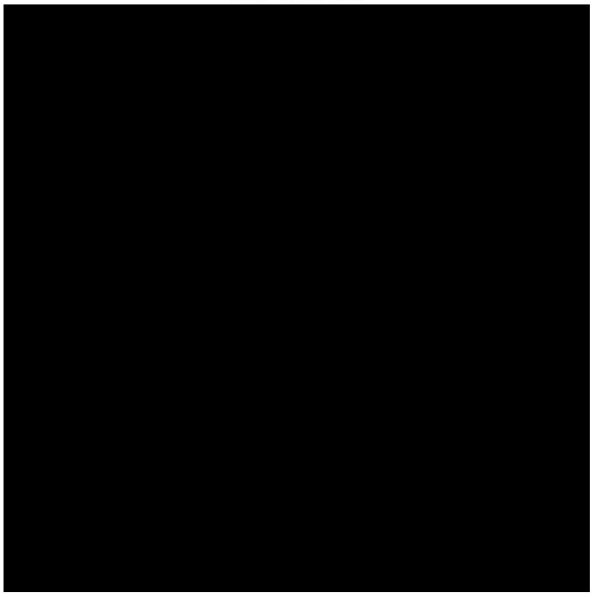
x_model Predicted Image : PSNR=6.5611
SSIM=0.0194
RMSE=35318.9118

QD Image



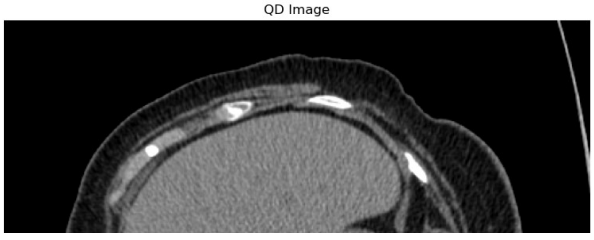
x_model Predicted Image : PSNR=8.859
SSIM=0.3801
RMSE=20807.4418

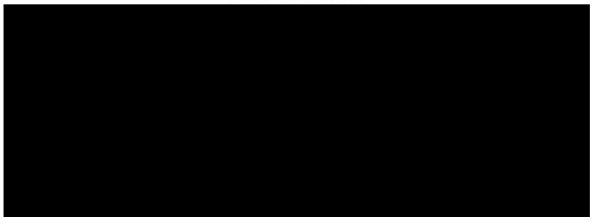
QD Image



x_model Predicted Image : PSNR=10.2639
SSIM=0.3265
RMSE=15056.7882

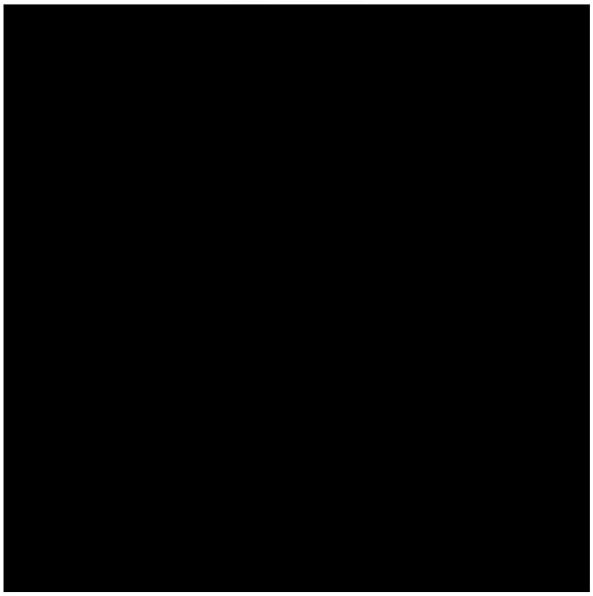
QD Image





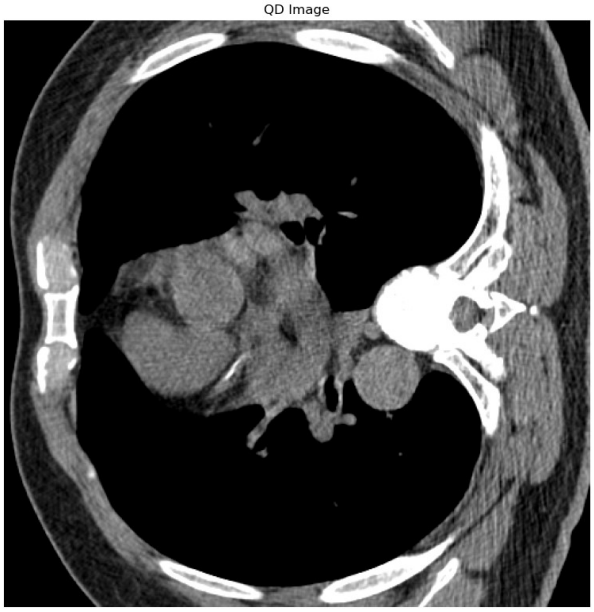
x_model Predicted Image : PSNR=9.5935
SSIM=0.2867
RMSE=17570.0392

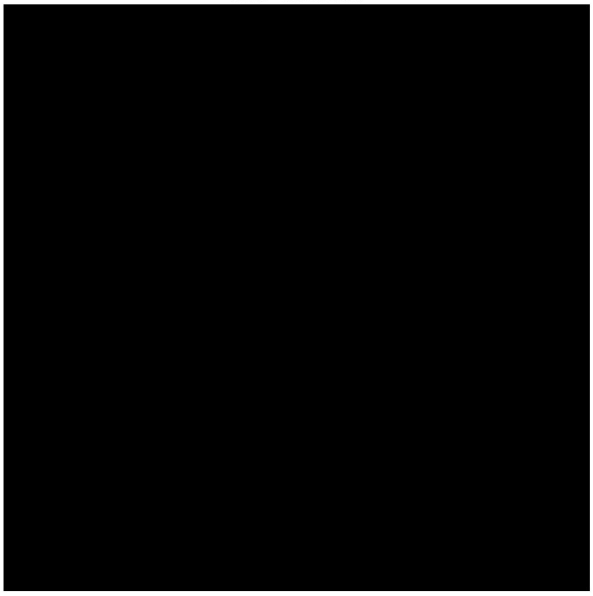
QD Image



x_model Predicted Image : PSNR=8.5387
SSIM=0.379
RMSE=22400.3476

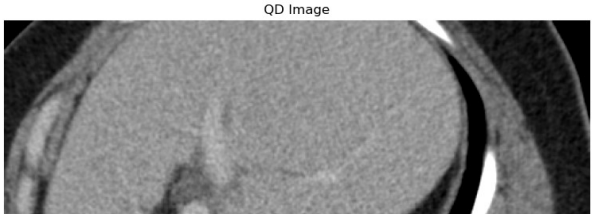
QD Image

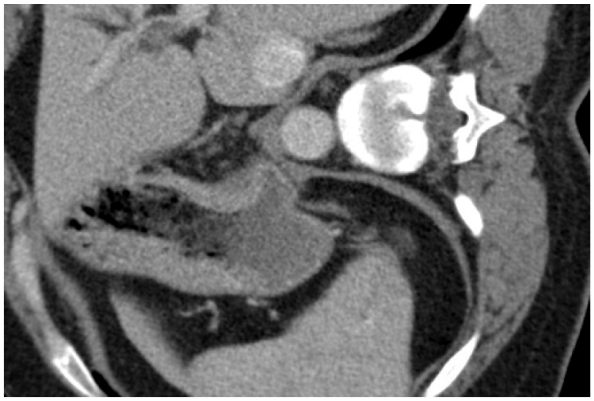




x_model Predicted Image : PSNR=6.4351
SSIM=-0.0604
RMSE=36358.9252

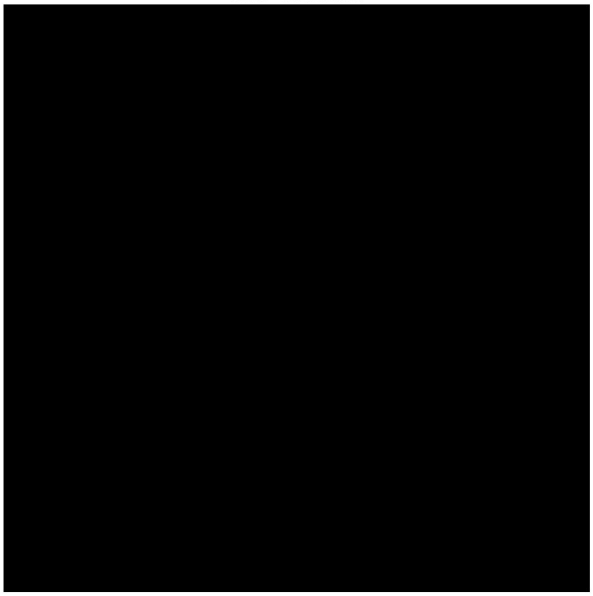
QD Image





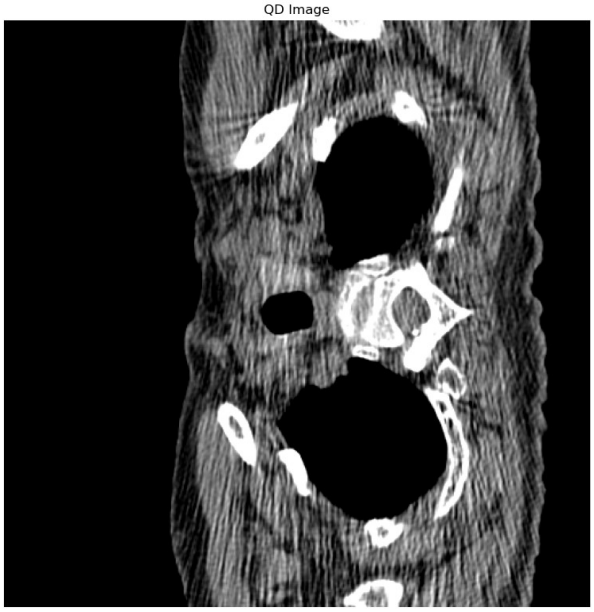
x_model Predicted Image : PSNR=8.0351
SSIM=0.3918
RMSE=25153.9269

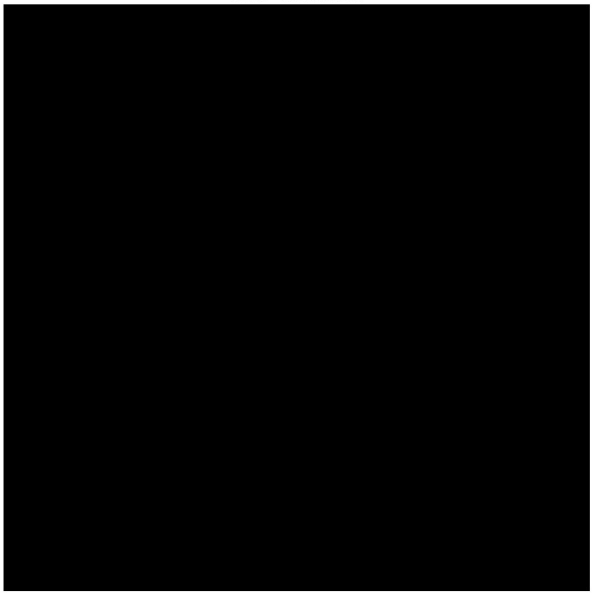
QD Image



x_model Predicted Image : PSNR=9.3606
SSIM=0.4486
RMSE=18537.7629

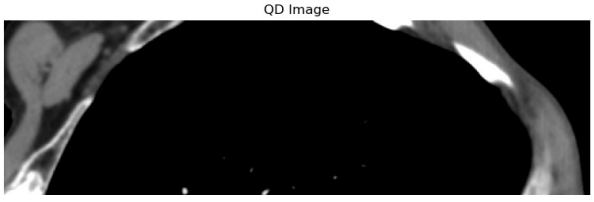
QD Image

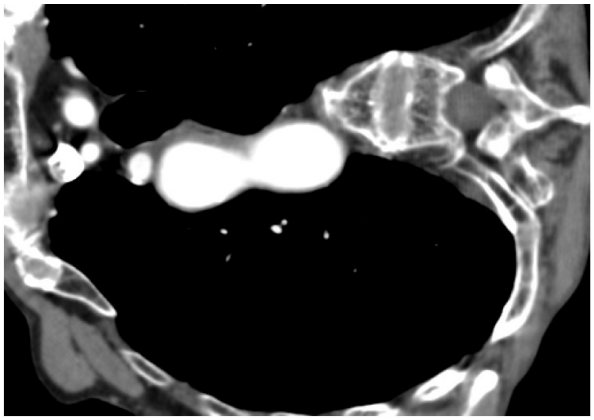




x_model Predicted Image : PSNR=8.8123
SSIM=0.5345
RMSE=21032.2221

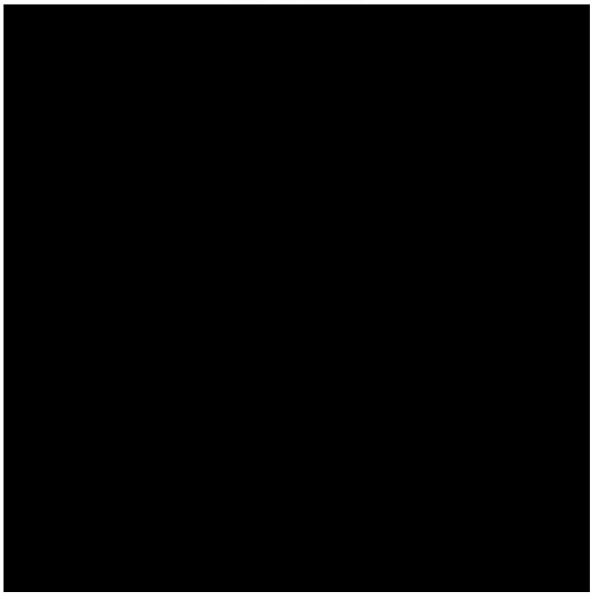
QD Image





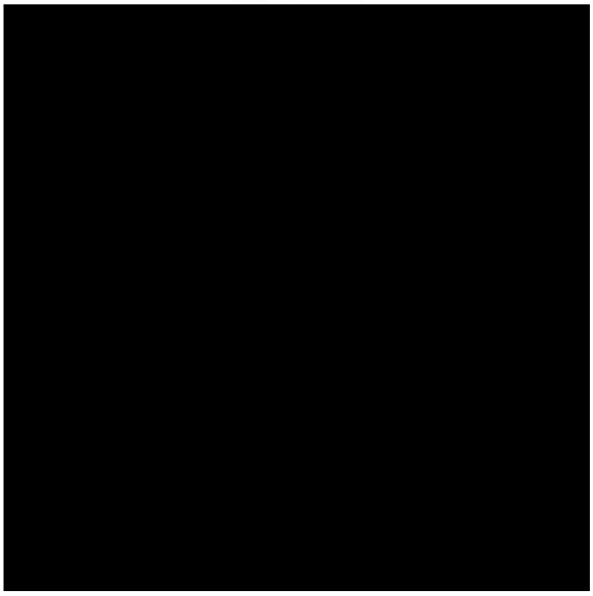
x_model Predicted Image : PSNR=9.2743
SSIM=0.4299
RMSE=18909.8739

QD Image



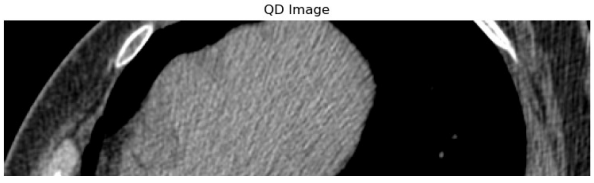
x_model Predicted Image : PSNR=8.3621
SSIM=0.4852
RMSE=23329.6099

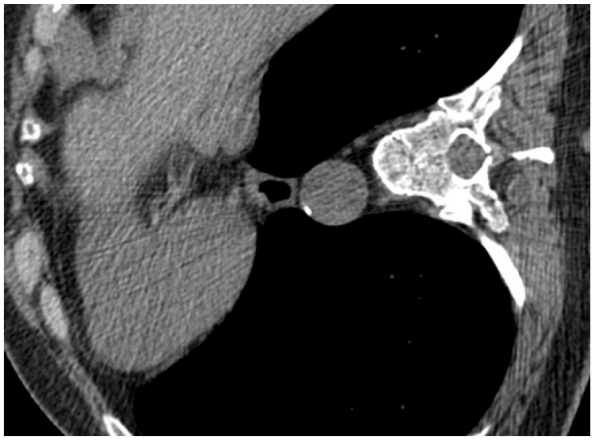
QD Image



x_model Predicted Image : PSNR=8.7448
SSIM=0.3217
RMSE=21361.9342

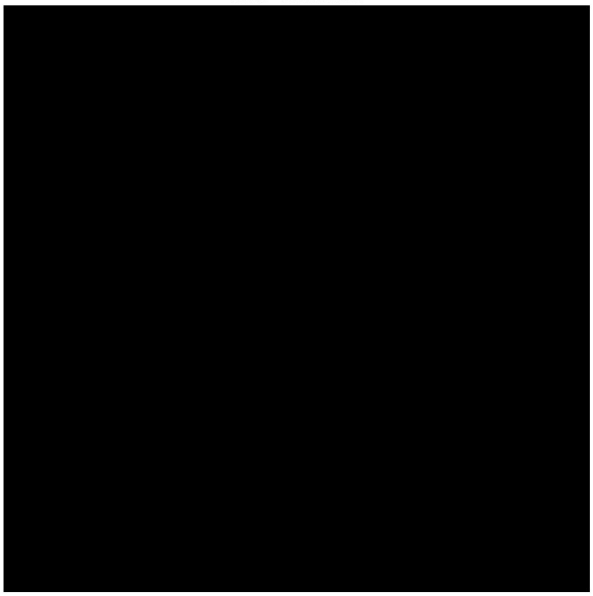
QD Image





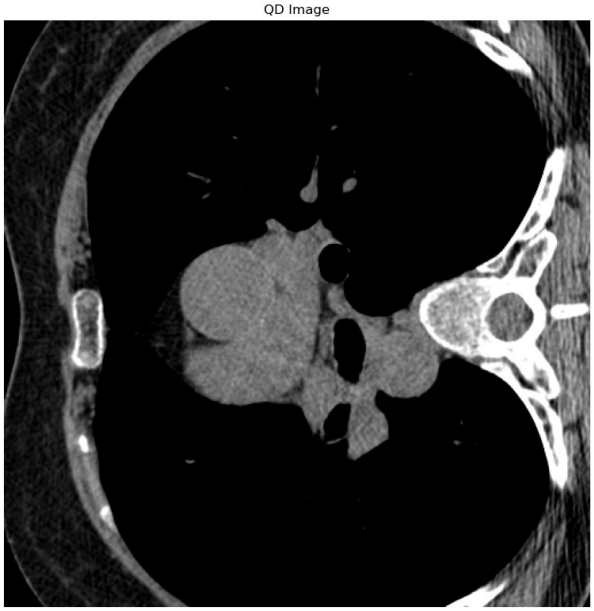
x_model Predicted Image : PSNR=9.119
SSIM=0.4541
RMSE=19598.4959

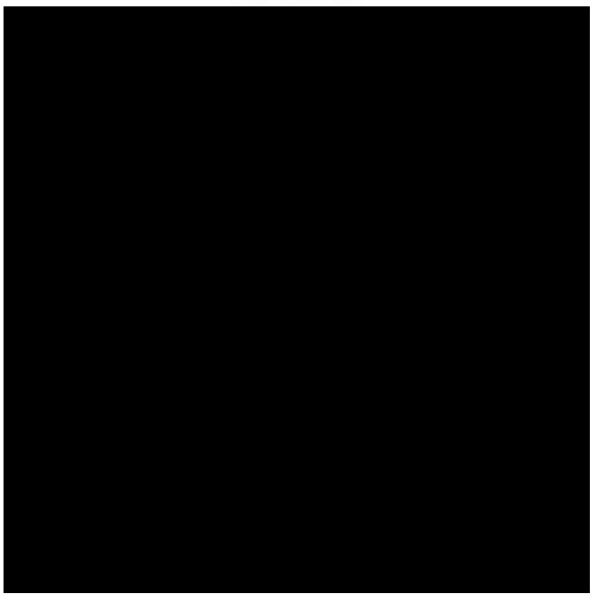
QD Image



x_model Predicted Image : PSNR=10.924
SSIM=0.5625
RMSE=12933.5925

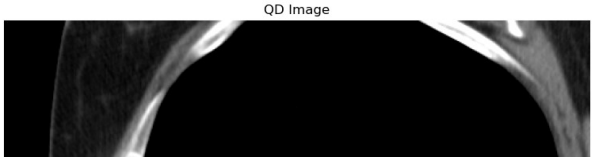
QD Image

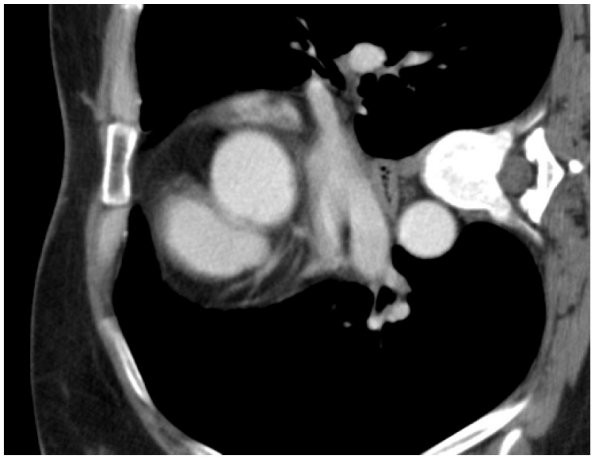




x_model Predicted Image : PSNR=9.4163
SSIM=0.5015
RMSE=18301.7504

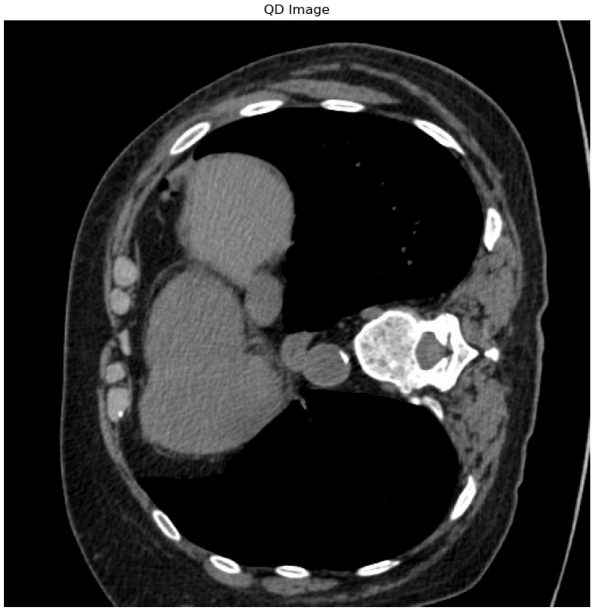
QD Image

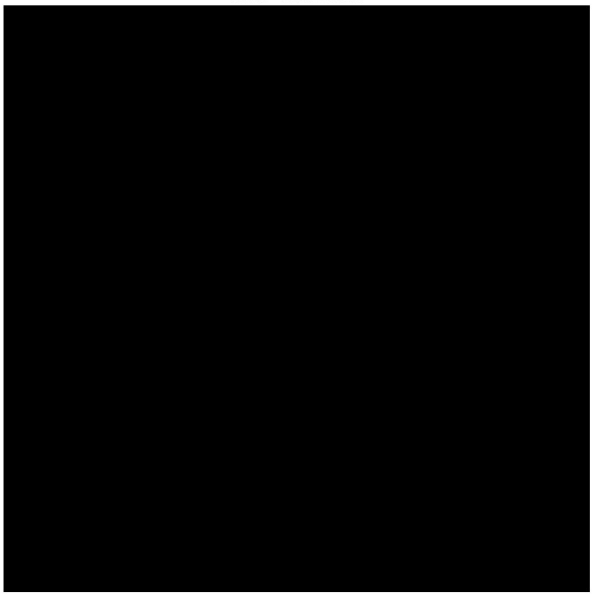




x_model Predicted Image : PSNR=11.6414
SSIM=0.4954
RMSE=10964.3882

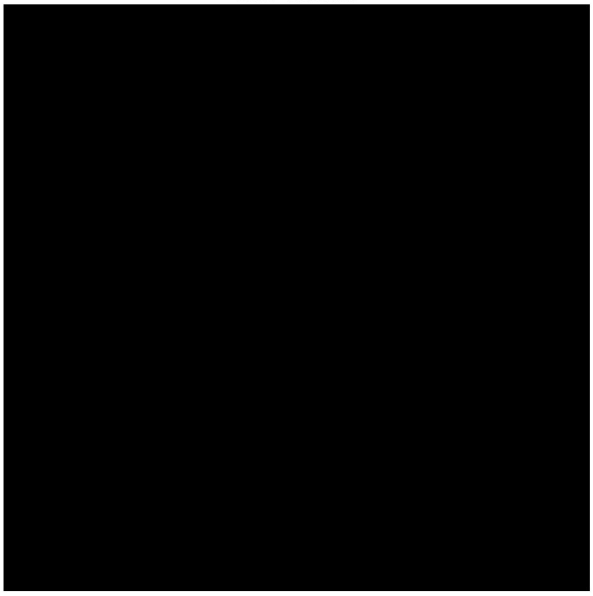
QD Image





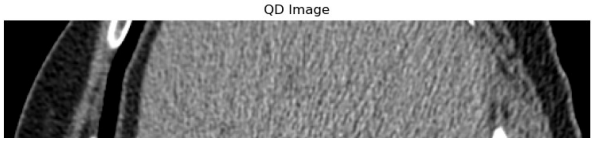
x_model Predicted Image : PSNR=5.5878
SSIM=0.0054
RMSE=44191.538

QD Image



x_model Predicted Image : PSNR=6.8974
SSIM=0.0068
RMSE=32687.452

QD Image



Model 3 : Y model

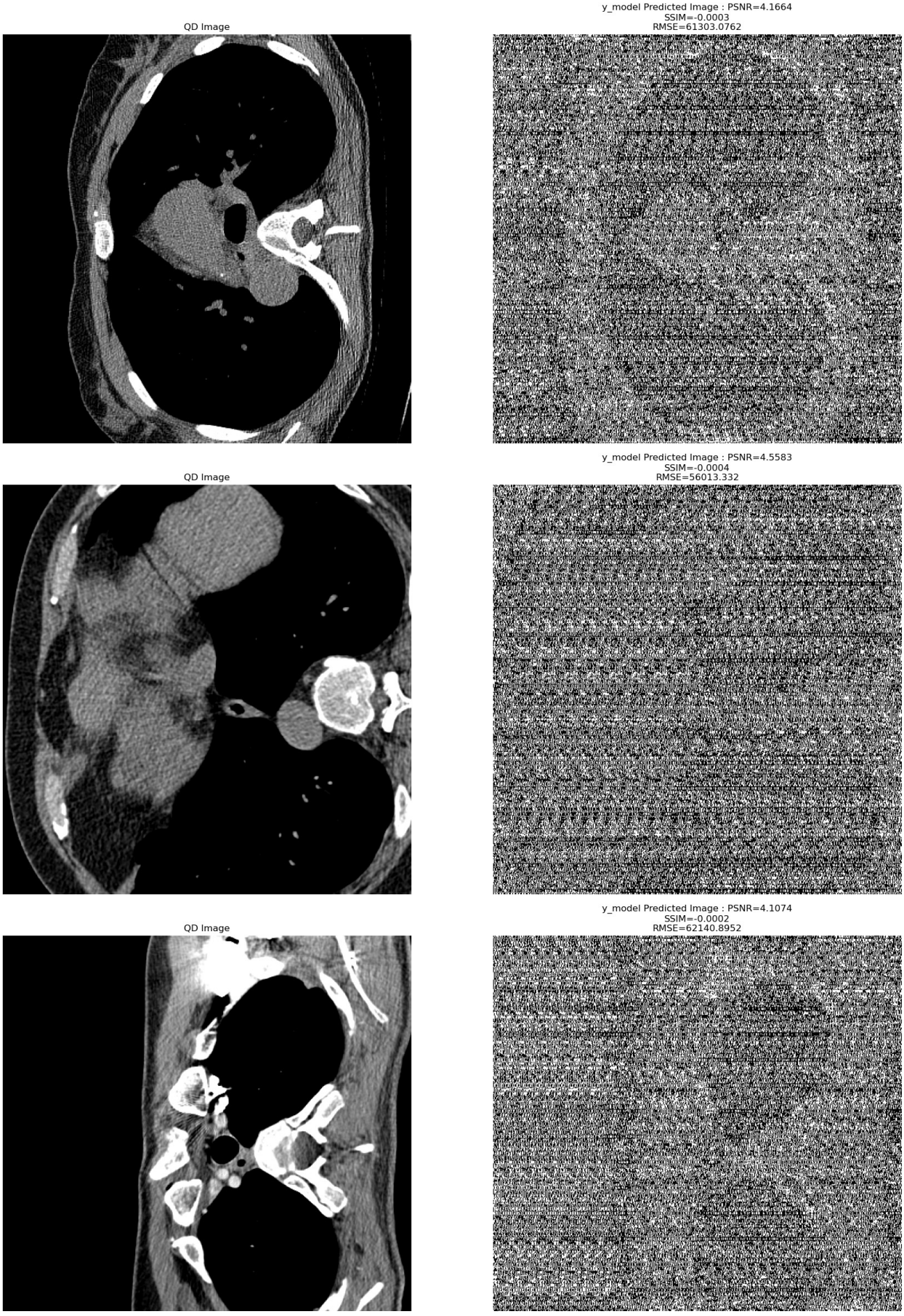
```
sys.path.append('../denoising-models/mwcnn/')
from y_denoiser import YModel
y_model = YModel(num_channels=56)
y_model.load_state_dict(torch.load('../denoising-models/novel_model/weights/y_model/model_24.pth'))
y_model.eval()
print('Model summary : ')
print(summary(y_model))
```

```
Model summary :
==========================================================================
Layer (type:depth-idx)                                   Param #
==========================================================================
YModel                                                   --
├─InputProjectionLayer: 1-1                              --
│    └─Conv2d: 2-1                                       560
├─OutputProjectionLayer: 1-2                             --
│    └─ConvTranspose2d: 2-2                              505
├─ConvBlock: 1-3                                         --
│    └─ZeroPad2d: 2-3                                    --
│    └─DepthWiseSeparableConv2d: 2-4                     --
│    │    └─Conv2d: 3-1                                  2,744
│    │    └─Conv2d: 3-2                                  3,136
│    └─LayerNorm: 2-5                                    --
│    └─Conv2d: 2-6                                       614,880
│    └─GELU: 2-7                                         --
│    └─Conv2d: 2-8                                       12,600
├─YBlock: 1-4                                            --
```

```
|    └─Linear: 2-9                        3,192
|    └─Linear: 2-10                       3,192
|    └─Linear: 2-11                       3,192
|    └─Conv2d: 2-12                       3,192
|    └─LayerNorm: 2-13                    --
|    └─Conv2d: 2-14                       12,768
|    └─GELU: 2-15                         --
|    └─Conv2d: 2-16                       12,600
├─Conv2d: 1-5                             25,200
├─ConvBlock: 1-6                          --
|    └─ZeroPad2d: 2-17                    --
|    └─DepthWiseSeparableConv2d: 2-18     --
|    |    └─Conv2d: 3-3                   5,488
|    |    └─Conv2d: 3-4                   12,544
|    └─LayerNorm: 2-19                    --
|    └─Conv2d: 2-20                       2,459,072
|    └─GELU: 2-21                         --
|    └─Conv2d: 2-22                       50,288
├─YBlock: 1-7                             --
|    └─Linear: 2-23                       12,656
|    └─Linear: 2-24                       12,656
|    └─Linear: 2-25                       12,656
|    └─Conv2d: 2-26                       12,656
|    └─LayerNorm: 2-27                    --
|    └─Conv2d: 2-28                       50,624
|    └─GELU: 2-29                         --
|    └─Conv2d: 2-30                       50,288
├─Conv2d: 1-8                             100,576
├─ConvBlock: 1-9                          --
|    └─ZeroPad2d: 2-31                    --
|    └─DepthWiseSeparableConv2d: 2-32     --
|    |    └─Conv2d: 3-5                   10,976
|    |    └─Conv2d: 3-6                   50,176
|    └─LayerNorm: 2-33                    --
|    └─Conv2d: 2-34                       9,835,392
|    └─GELU: 2-35                         --
|    └─Conv2d: 2-36                       200,928
├─YBlock: 1-10                            --
|    └─Linear: 2-37                       50,400
```

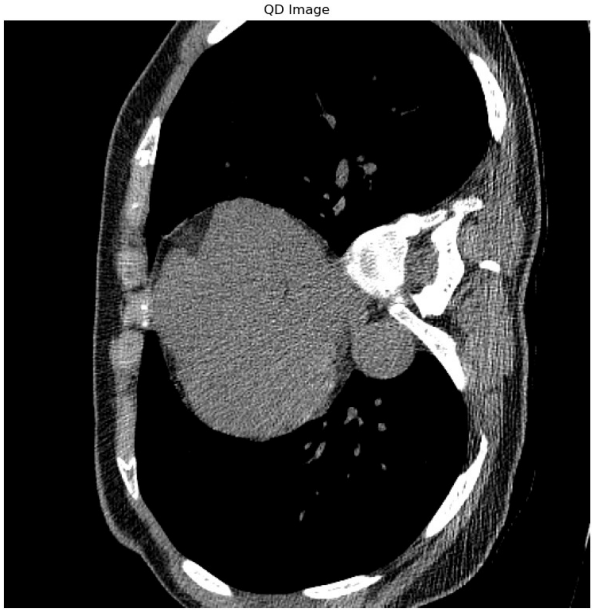
```
# Get prediction images.

y_model_prediction_patches = []

for img in noisy_array:
    img_tensor = torch.unsqueeze(torch.from_numpy(img).float(), dim=0)


    with torch.no_grad():
        output_tensor = y_model(img_tensor)
        output_tensor = output_tensor.cpu().numpy()

        y_model_prediction_patches.append(output_tensor)
```

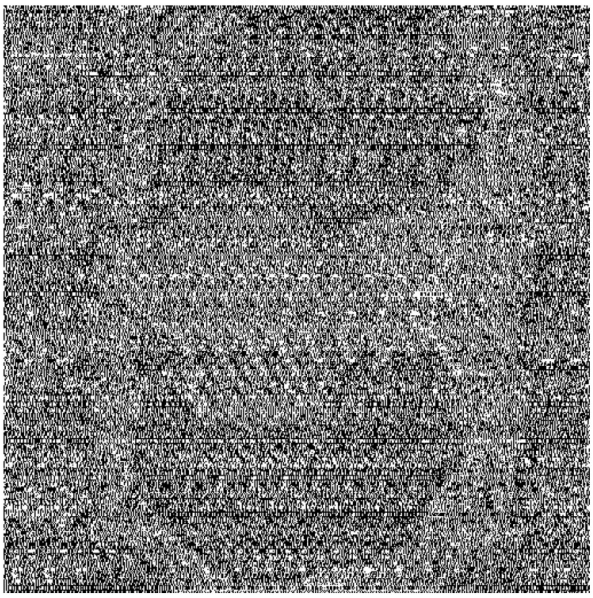
```
visualize_predictions(y_model, noisy_array, len(noisy_array), np.concatenate(y_model_prediction_patches
```



y_model Predicted Image : PSNR=4.1733
SSIM=-0.0004
RMSE=61205.3534

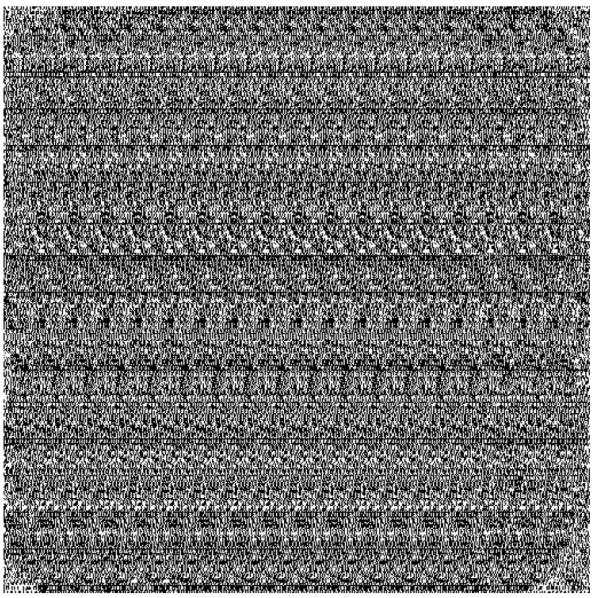
QD Image





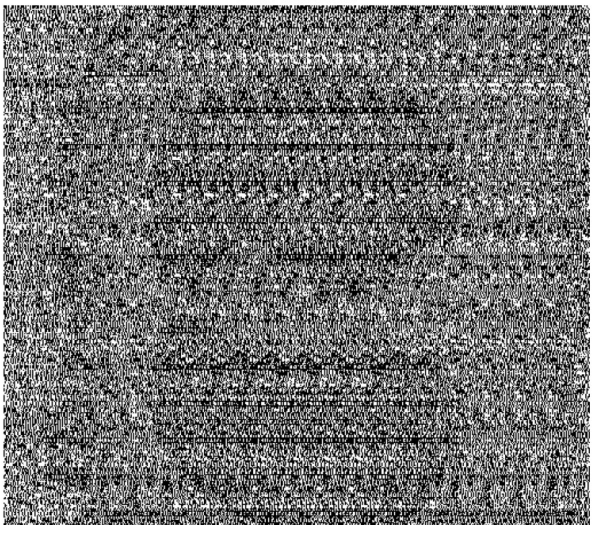
y_model Predicted Image : PSNR=5.4648
SSIM=0.0004
RMSE=45460.7049

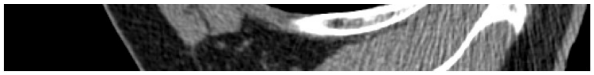
QD Image



y_model Predicted Image : PSNR=4.4328
SSIM=0.0005
RMSE=57656.033

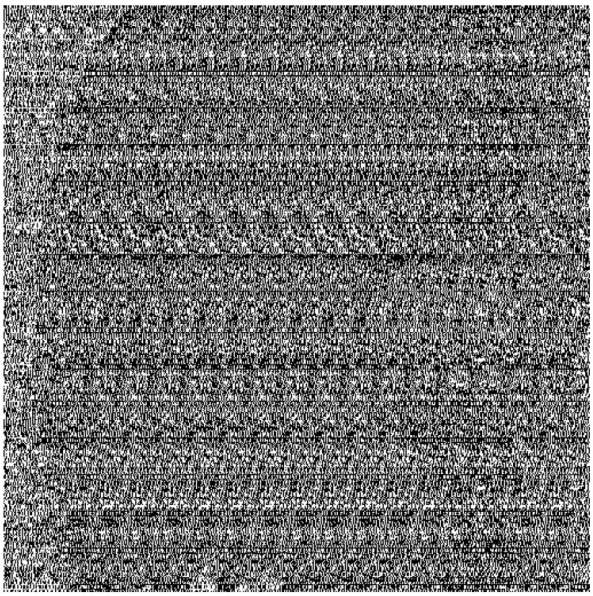
QD Image





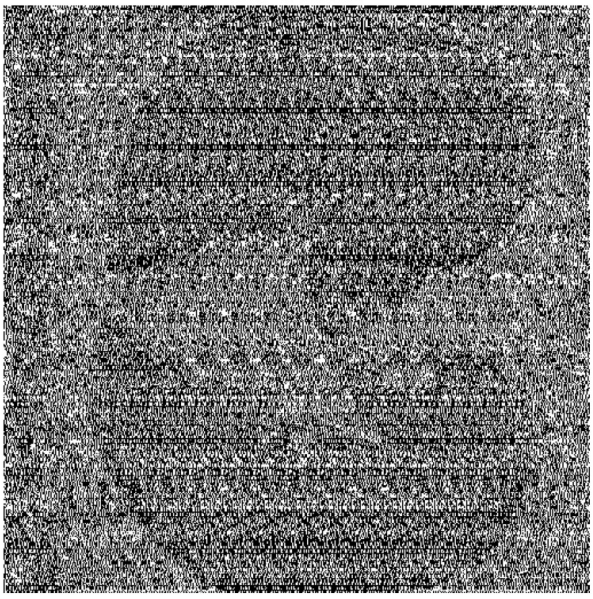
y_model Predicted Image : PSNR=4.8648
SSIM=-0.0
RMSE=52195.7871

QD Image



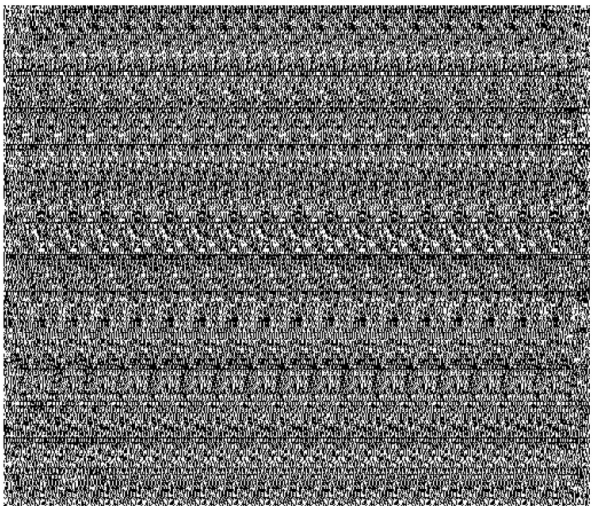
y_model Predicted Image : PSNR=4.4488
SSIM=-0.0
RMSE=57443.4173

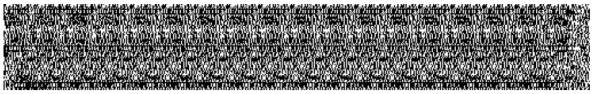
QD Image



y_model Predicted Image : PSNR=5.1998
SSIM=-0.0001
RMSE=48321.0675

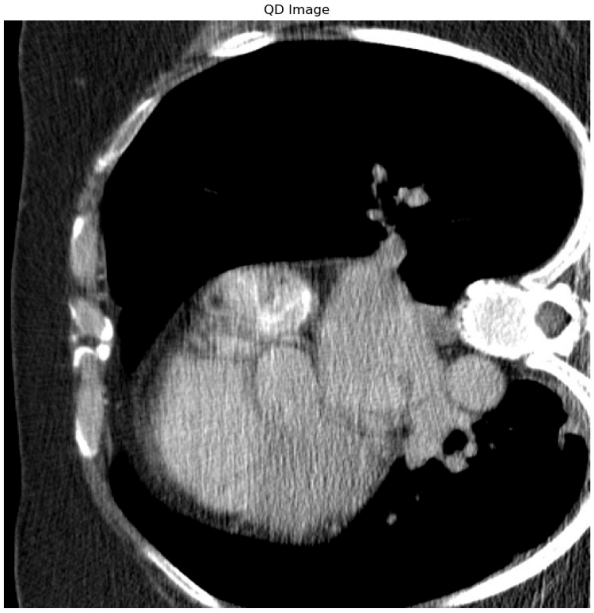
QD Image

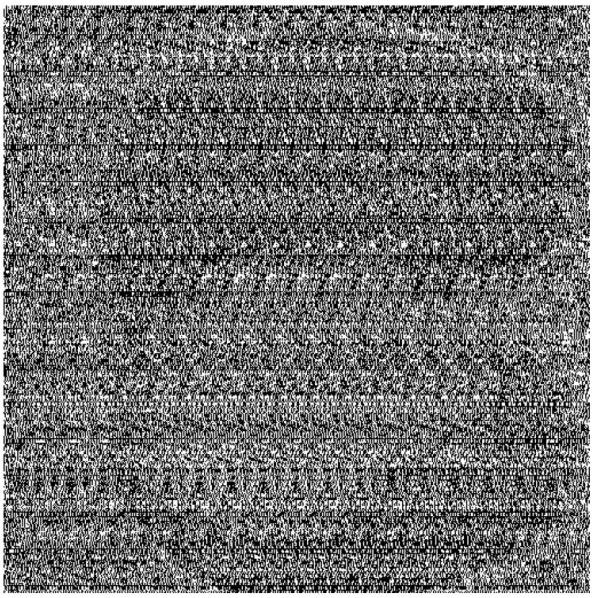




y_model Predicted Image : PSNR=4.5733
SSIM=0.0004
RMSE=55819.5905

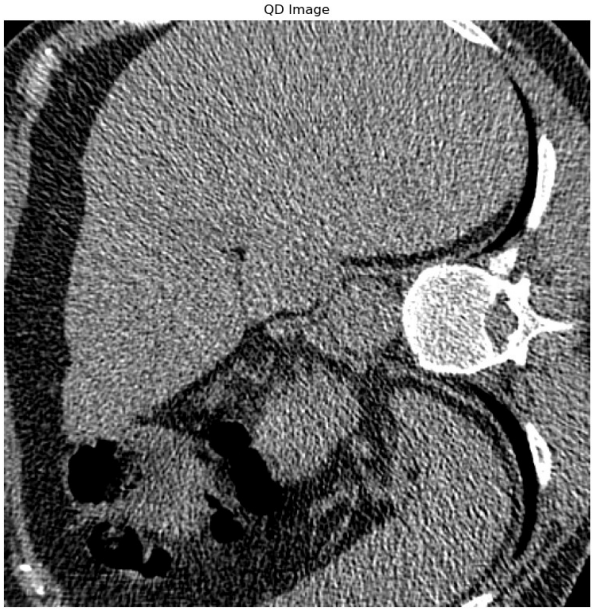
QD Image

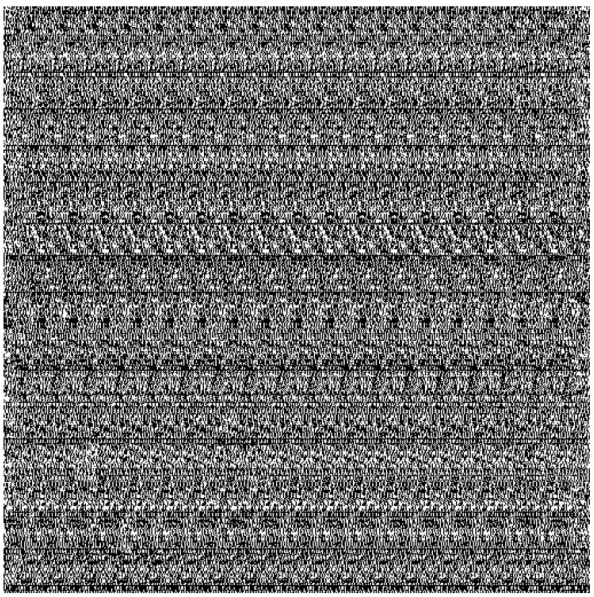




y_model Predicted Image : PSNR=5.1896
SSIM=0.0002
RMSE=48434.9263

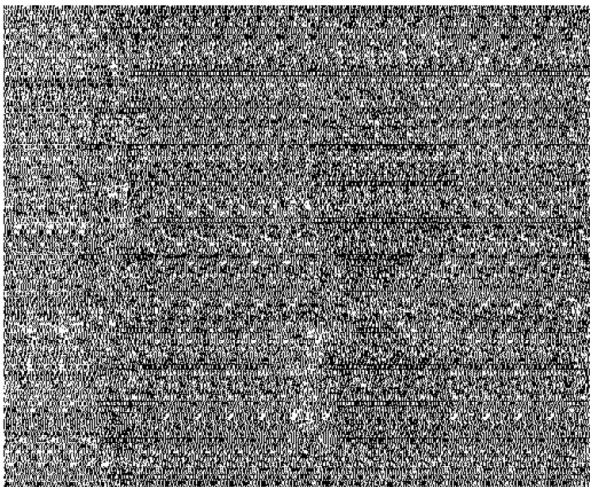
QD Image

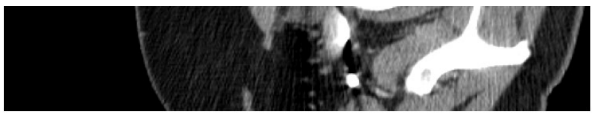


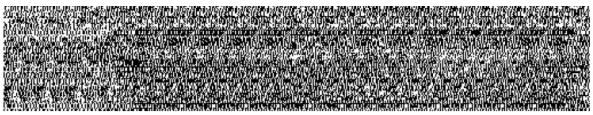


y_model Predicted Image : PSNR=4.3398
SSIM=-0.0003
RMSE=58903.4096

QD Image

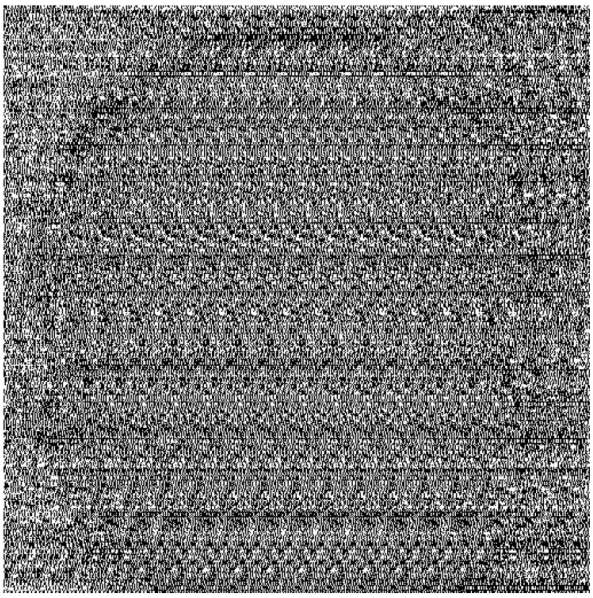






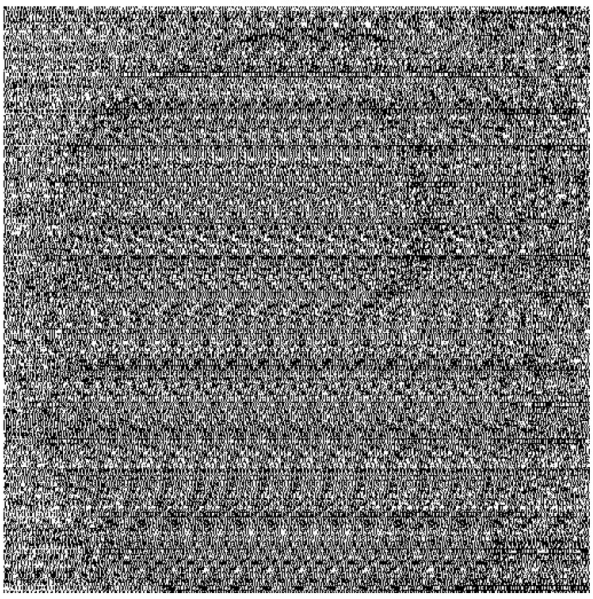
y_model Predicted Image : PSNR=4.6127
SSIM=-0.0001
RMSE=55316.1083

QD Image



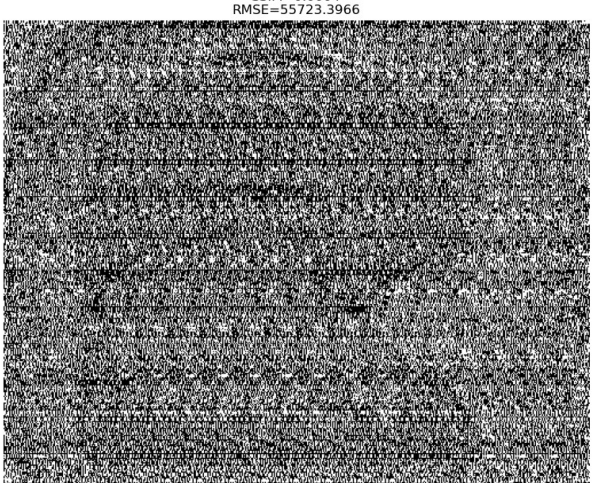
y_model Predicted Image : PSNR=4.6076
SSIM=0.0001
RMSE=55381.2584

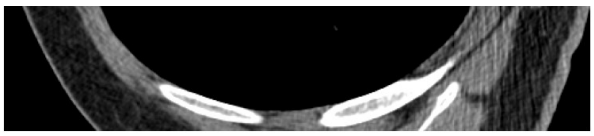
QD Image

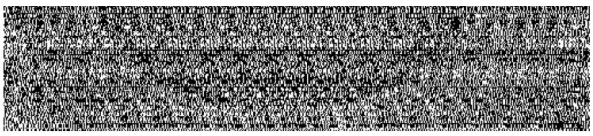


y_model Predicted Image : PSNR=4.5808
SSIM=0.0004
RMSE=55723.3966

QD Image

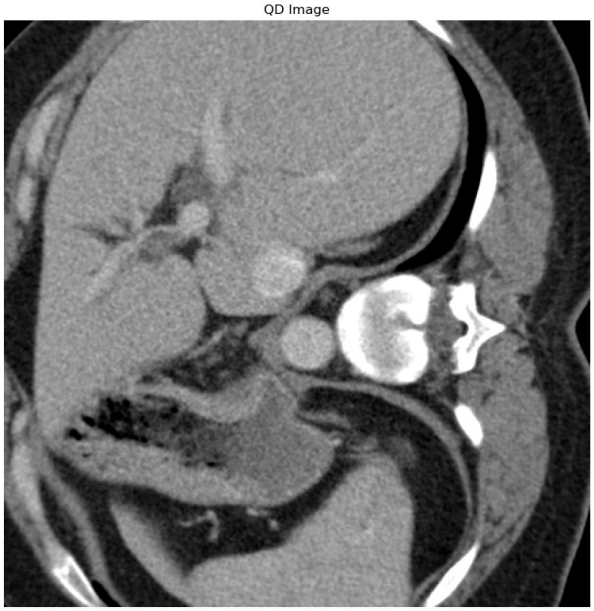


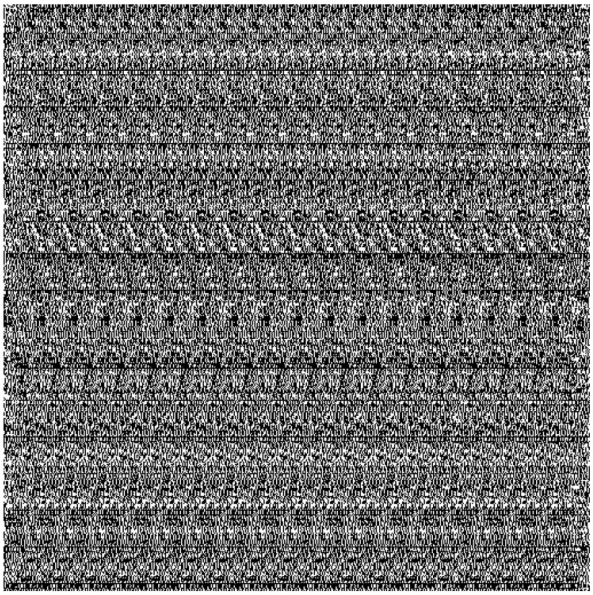




y_model Predicted Image : PSNR=5.4038
SSIM=0.0002
RMSE=46104.0151

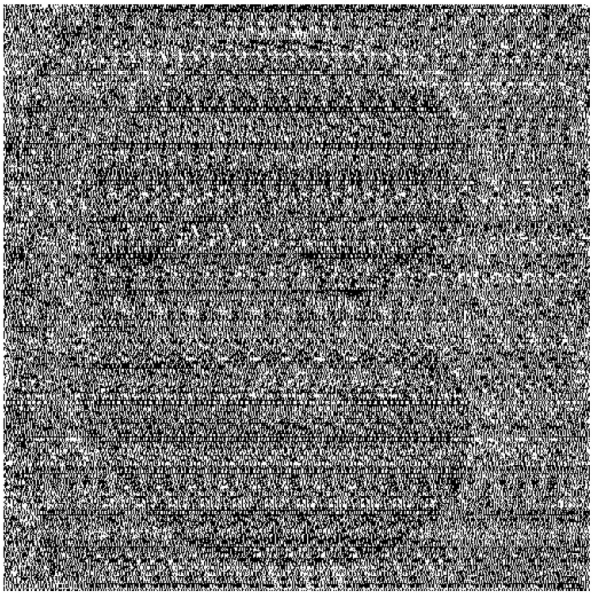
QD Image





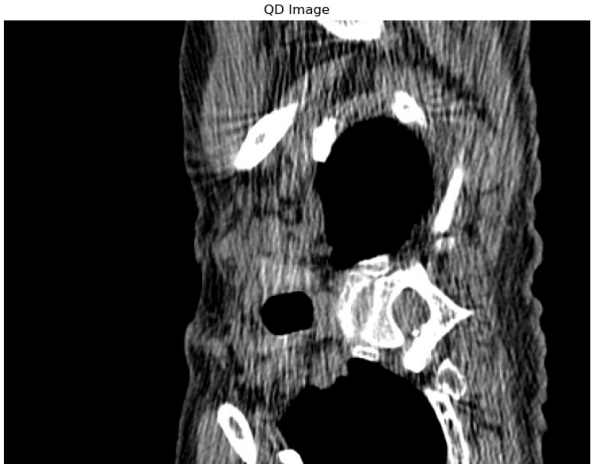
y_model Predicted Image : PSNR=4.5031
SSIM=0.0005
RMSE=56729.6448

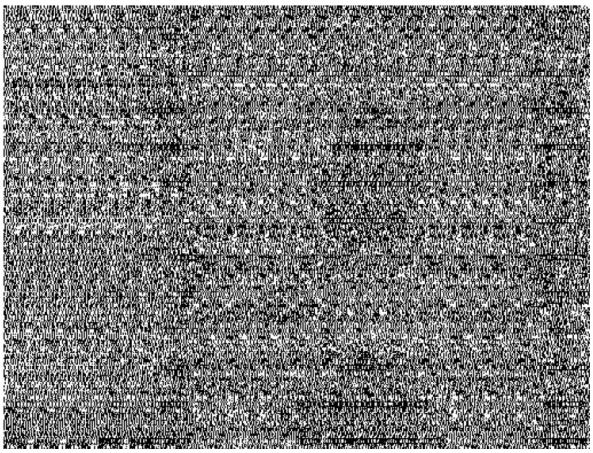
QD Image

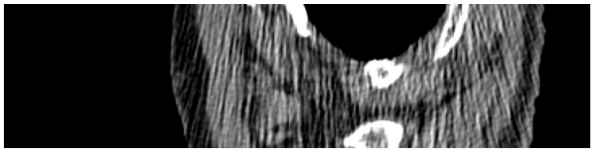


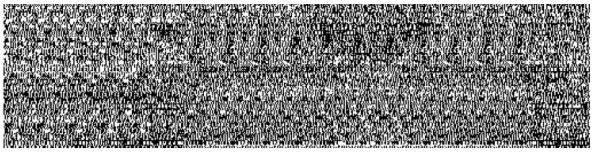
y_model Predicted Image : PSNR=4.2247
SSIM=-0.0003
RMSE=60484.8817

QD Image



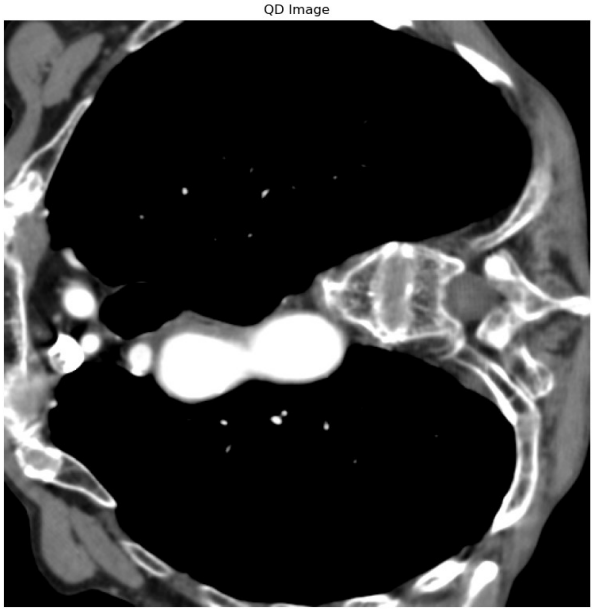


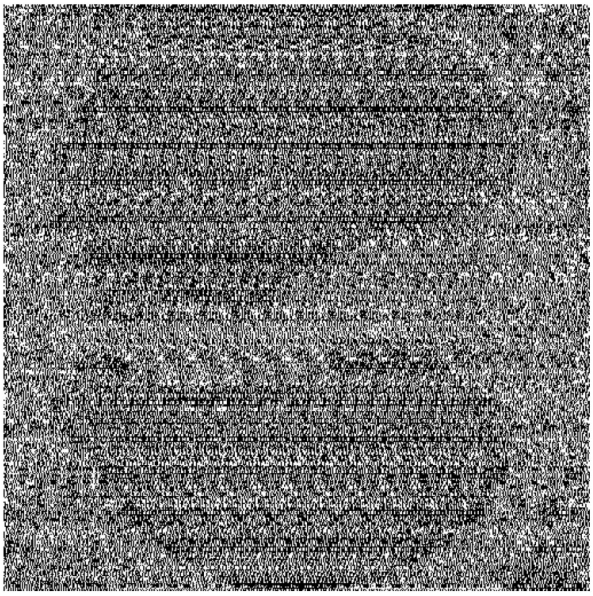




y_model Predicted Image : PSNR=4.3008
SSIM=-0.0002
RMSE=59434.827

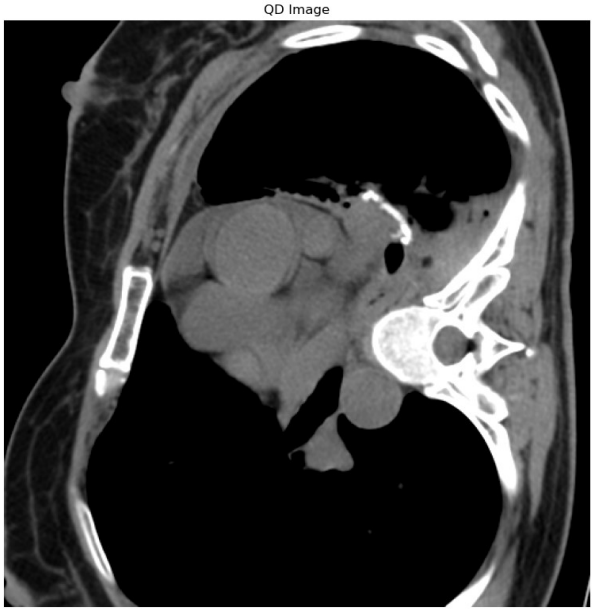
QD Image

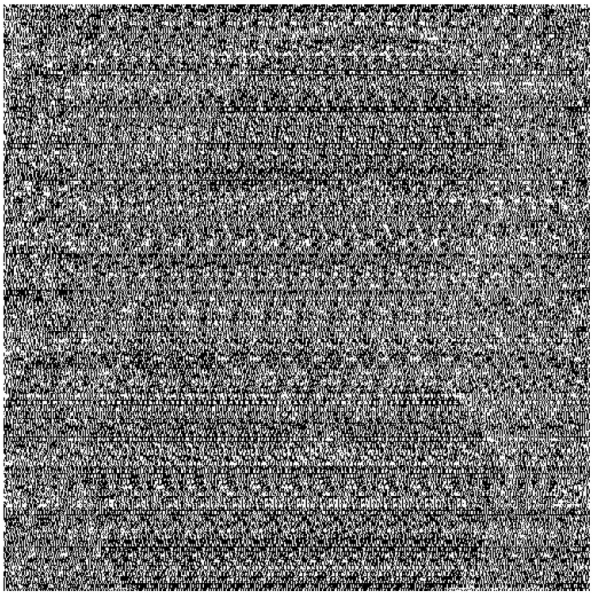




y_model Predicted Image : PSNR=4.5255
SSIM=0.0
RMSE=56438.4324

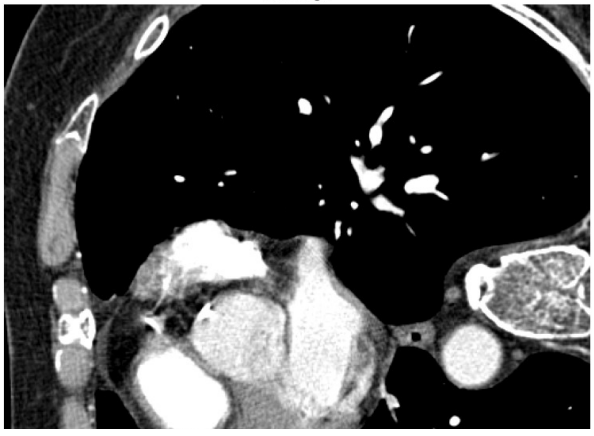
QD Image

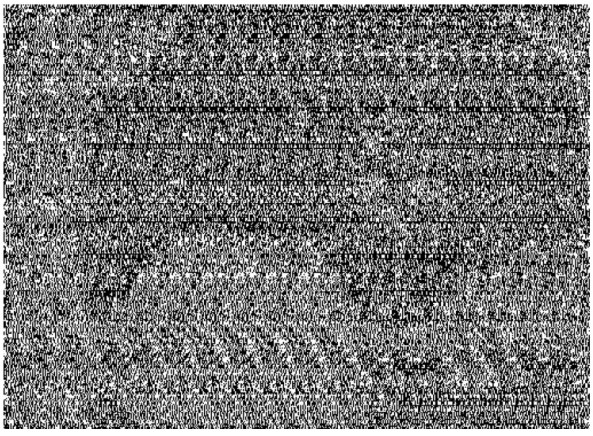


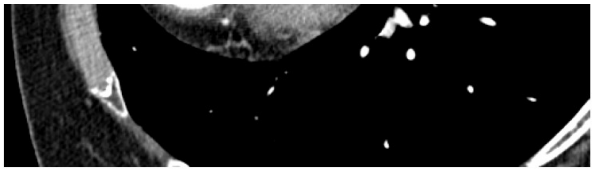


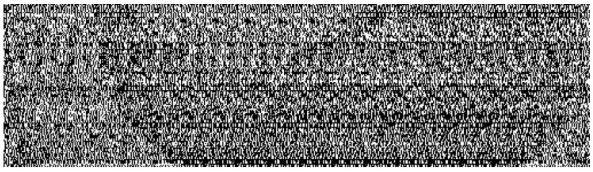
y_model Predicted Image : PSNR=4.196
SSIM=0.0003
RMSE=60886.8914

QD Image



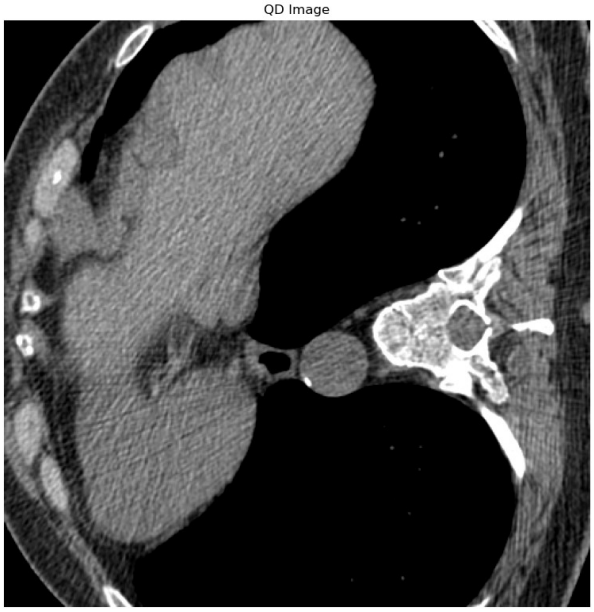


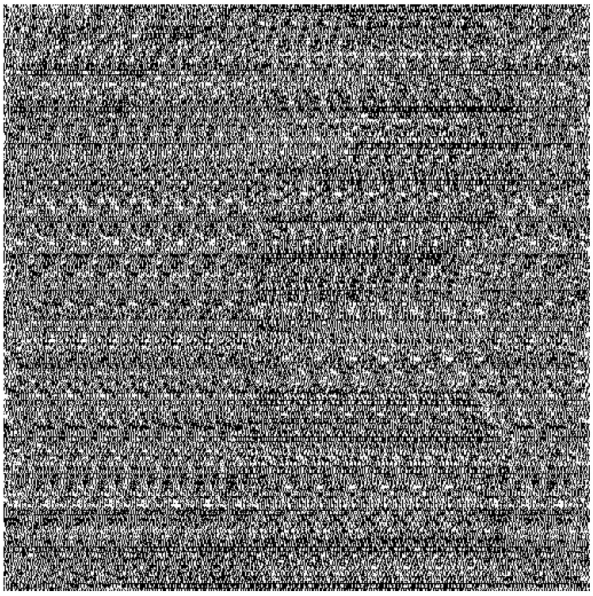




y_model Predicted Image : PSNR=4.8094
SSIM=0.0006
RMSE=52866.6788

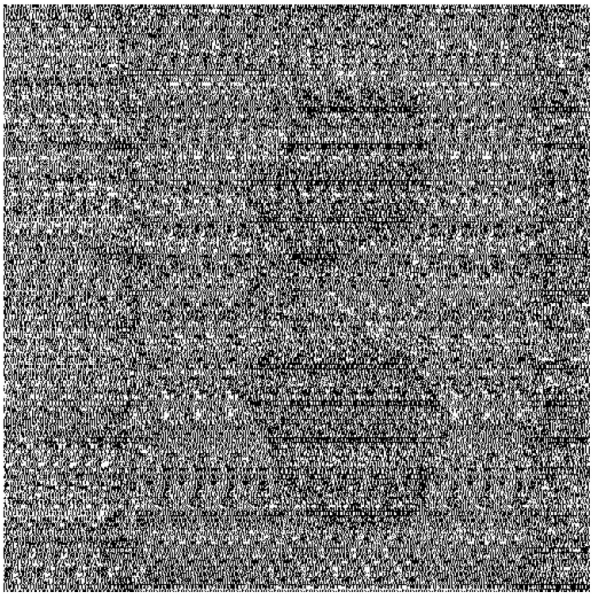
QD Image





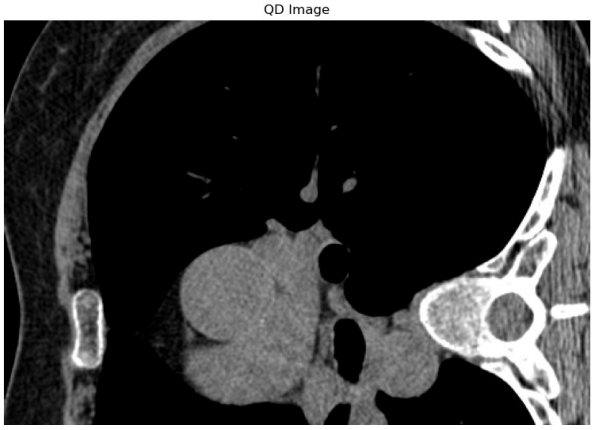
y_model Predicted Image : PSNR=4.2706
SSIM=0.0005
RMSE=59849.4279

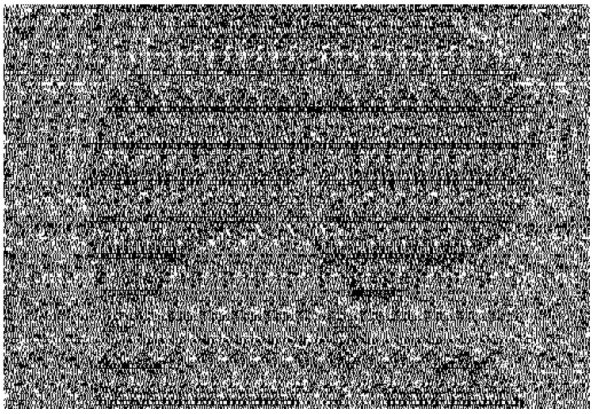
QD Image

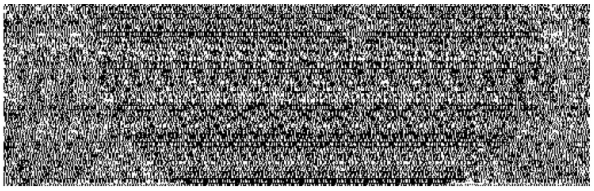


y_model Predicted Image : PSNR=4.3537
SSIM=0.0003
RMSE=58715.1042

QD Image

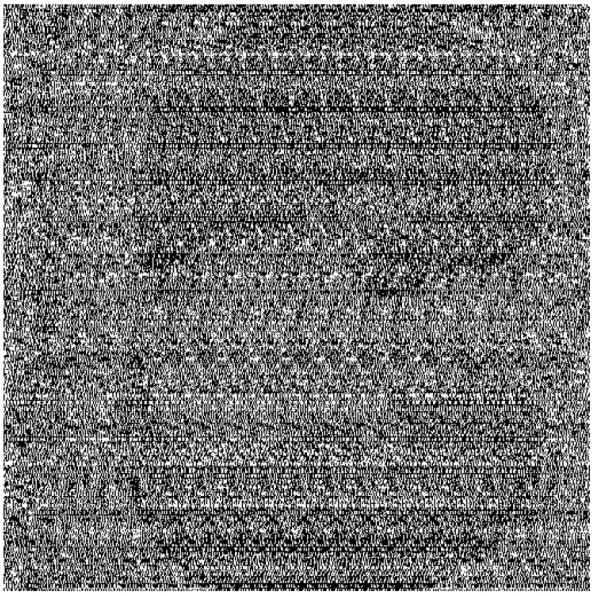




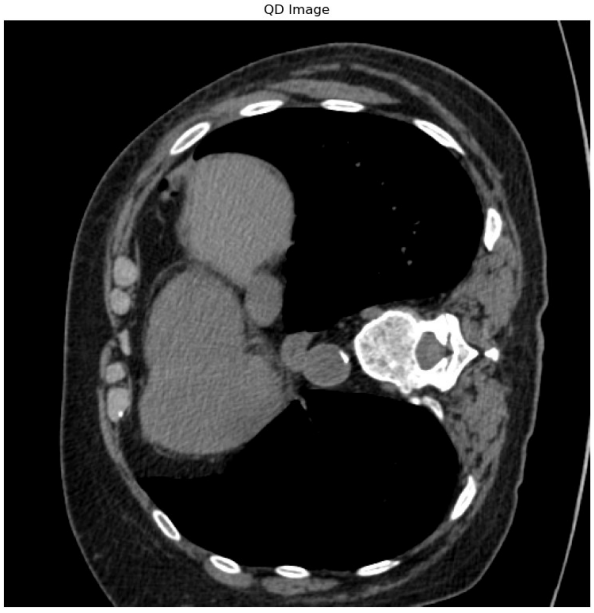


QD Image

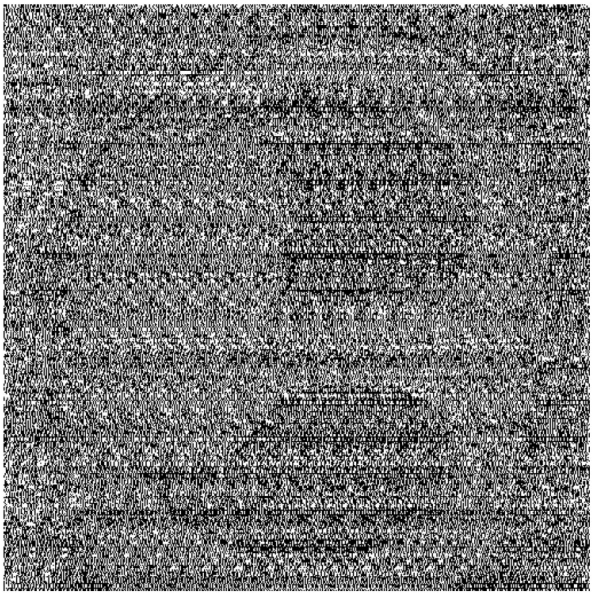
y_model Predicted Image : PSNR=4.4191
SSIM=0.0002
RMSE=57837.8401



QD Image

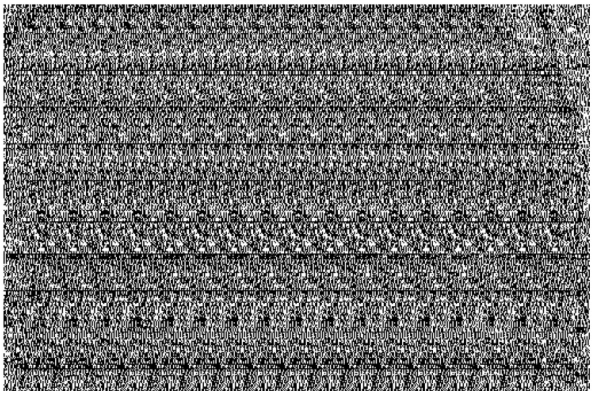


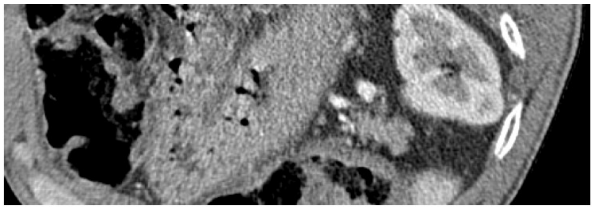
y_model Predicted Image : PSNR=4.3045
SSIM=-0.0006
RMSE=59383.5666

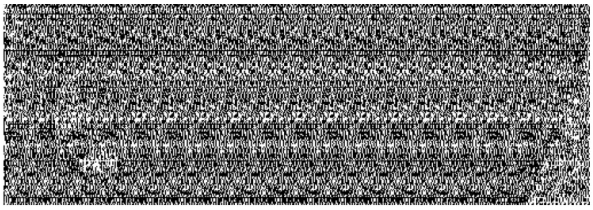


QD Image

y_model Predicted Image : PSNR=5.215
SSIM=0.0006
RMSE=48152.7325

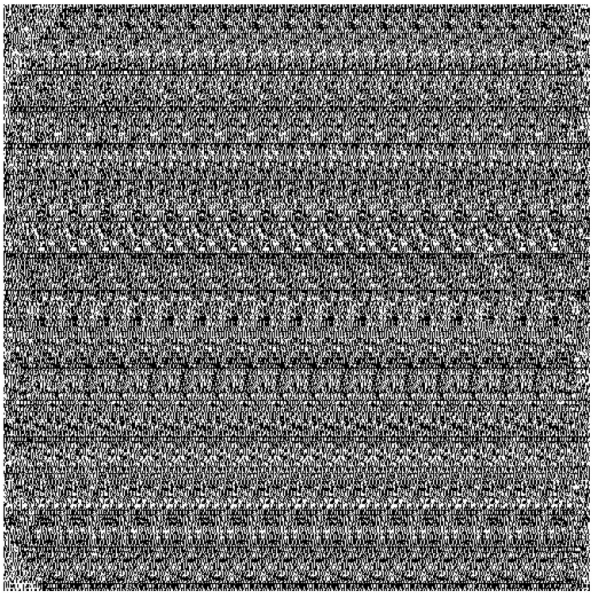






y_model Predicted Image : PSNR=5.4297
SSIM=0.0006
RMSE=45829.8564

QD Image



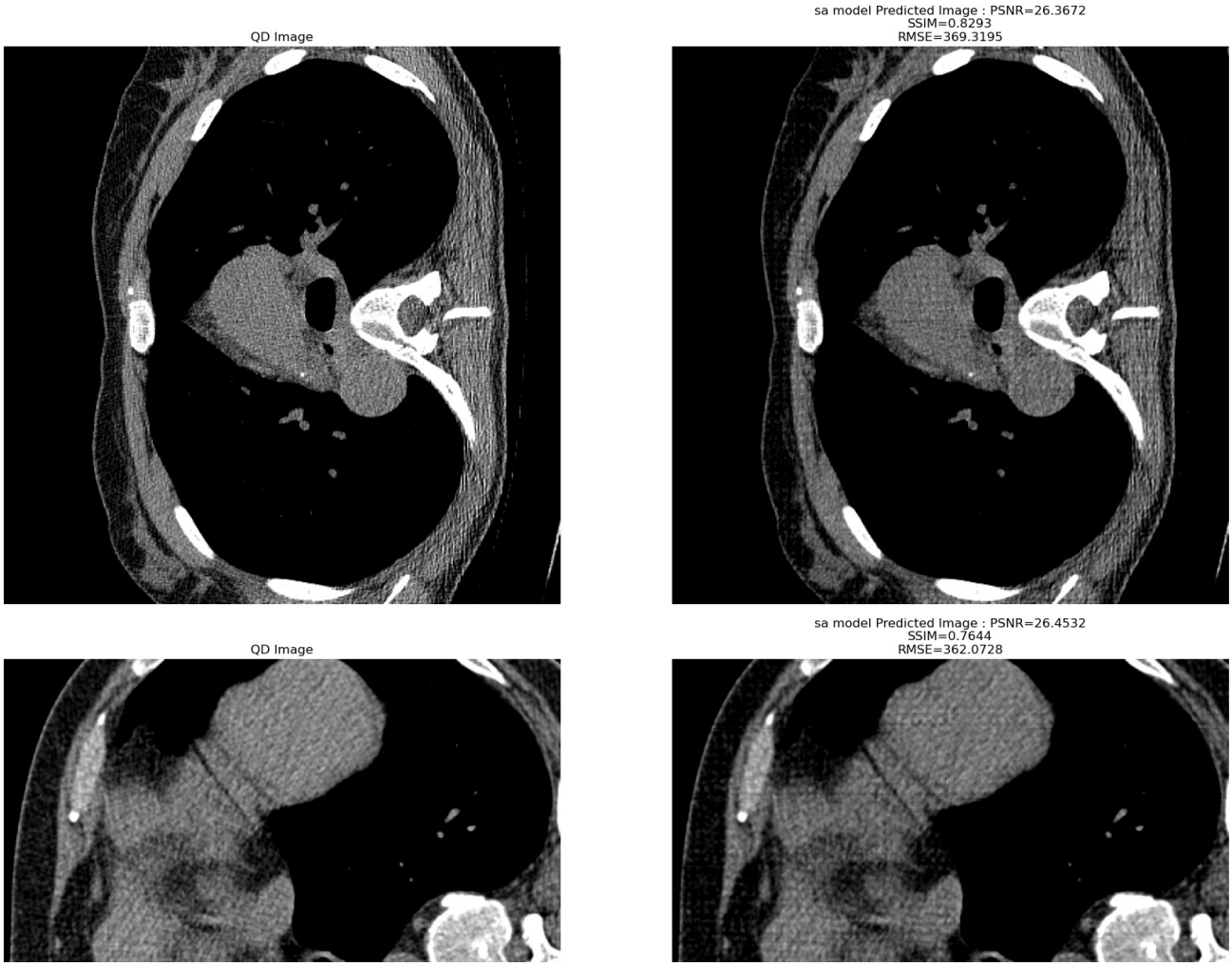
Model 3 : SA Model

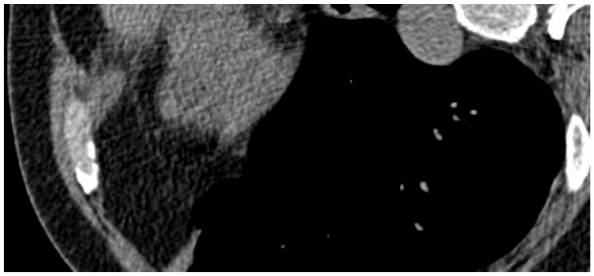
```
sys.path.append('../denoising-models/hformer_self_attention')
from torchinfo import summary


from hformer_sa_model import HformerSAModel

sa_model = HformerSAModel(num_channels=64).cuda()
sa_model.load_state_dict(torch.load('../denoising-models/hformer_self_attention/weights/model_75.pth'))
sa_model.eval()
print('model summary\n', summary(sa_model, input_size=(64, 64, 64, 1)))
```

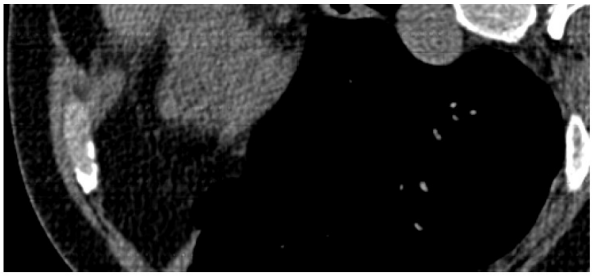
```
model summary
 ==========================================================================================
Layer (type:depth-idx)                             Output Shape              Param #
```

```
==========================================================================================
HformerSAModel                                [64, 64, 64, 1]           --
├─InputProjectionLayer: 1-1                   [64, 64, 64, 64]          --
│    └─Conv2d: 2-1                            [64, 64, 64, 64]          640
├─ConvBlock: 1-2                              [64, 64, 64, 64]          --
│    └─ZeroPad2d: 2-2                         [64, 64, 18, 18]          --
│    └─DepthWiseSeparableConv2d: 2-3          [64, 64, 18, 18]          --
│    │    └─Conv2d: 3-1                       [64, 64, 18, 18]          3,136
│    │    └─Conv2d: 3-2                       [64, 64, 18, 18]          4,096
│    └─Conv2d: 2-4                            [64, 256, 12, 12]         803,072
│    └─GELU: 2-5                              [64, 256, 12, 12]         --
│    └─Conv2d: 2-6                            [64, 64, 12, 12]          16,448
│    └─ZeroPad2d: 2-7                         [64, 64, 18, 18]          --
│    └─DepthWiseSeparableConv2d: 2-8          [64, 64, 18, 18]          --
│    │    └─Conv2d: 3-3                       [64, 64, 18, 18]          3,136
│    │    └─Conv2d: 3-4                       [64, 64, 18, 18]          4,096
│    └─Conv2d: 2-9                            [64, 256, 12, 12]         803,072
│    └─GELU: 2-10                             [64, 256, 12, 12]         --
│    └─Conv2d: 2-11                           [64, 64, 12, 12]          16,448
│    └─DWTInverse: 2-12                       [64, 64, 64, 64]          --
├─Conv2d: 1-3                                 [64, 128, 32, 32]         32,896
├─HformerBlock: 1-4                           [64, 128, 32, 32]         16,512
│    └─DepthWiseSeparableConv2d: 2-13         [64, 128, 32, 32]         --
│    │    └─Conv2d: 3-5                       [64, 128, 32, 32]         6,272
│    │    └─Conv2d: 3-6                       [64, 128, 32, 32]         16,384
│    └─MaxPool2d: 2-14                        [64, 128, 16, 16]         --
│    └─Linear: 2-15                           [64, 256, 128]            16,384
│    └─Linear: 2-16                           [64, 256, 128]            16,384
│    └─Linear: 2-17                           [64, 1024, 128]           16,384
│    └─LayerNorm: 2-18                        [64, 128, 32, 32]         --
│    └─Conv2d: 2-19                           [64, 512, 32, 32]         66,048
│    └─GELU: 2-20                             [64, 512, 32, 32]         --
│    └─Conv2d: 2-21                           [64, 128, 32, 32]         65,664
├─Conv2d: 1-5                                 [64, 256, 16, 16]         131,328
├─HformerBlock: 1-6                           [64, 256, 16, 16]         65,792
│    └─DepthWiseSeparableConv2d: 2-22         [64, 256, 16, 16]         --
│    │    └─Conv2d: 3-7                       [64, 256, 16, 16]         12,544
│    │    └─Conv2d: 3-8                       [64, 256, 16, 16]         65,536
│    └─MaxPool2d: 2-23                        [64, 256, 8, 8]           --
│    └─Linear: 2-24                           [64, 64, 256]             65,536
│    └─Linear: 2-25                           [64, 64, 256]             65,536
│    └─Linear: 2-26                           [64, 256, 256]            65,536
│    └─LayerNorm: 2-27                        [64, 256, 16, 16]         --
│    └─Conv2d: 2-28                           [64, 1024, 16, 16]        263,168
│    └─GELU: 2-29                             [64, 1024, 16, 16]        --
│    └─Conv2d: 2-30                           [64, 256, 16, 16]         262,400
├─ConvTranspose2d: 1-7                        [64, 128, 32, 32]         131,200
├─HformerBlock: 1-8                           [64, 128, 32, 32]         16,512
│    └─DepthWiseSeparableConv2d: 2-31         [64, 128, 32, 32]         --
│    │    └─Conv2d: 3-9                       [64, 128, 32, 32]         6,272
│    │    └─Conv2d: 3-10                      [64, 128, 32, 32]         16,384
│    └─MaxPool2d: 2-32                        [64, 128, 16, 16]         --
│    └─Linear: 2-33                           [64, 256, 128]            16,384
│    └─Linear: 2-34                           [64, 256, 128]            16,384
│    └─Linear: 2-35                           [64, 1024, 128]           16,384
```

```
sa_prediction_patches = []

with torch.no_grad():
    for i, data in enumerate(noisy_image_patches_array):
        noisy = data

        predictions = sa_model(torch.unsqueeze(torch.from_numpy(noisy.numpy()), dim=0).to('cuda')).cpu()

        sa_prediction_patches.append(predictions.detach().cpu())

sa_prediction_patches = np.concatenate(sa_prediction_patches, axis=0)
sa_predictions = np.expand_dims(reconstruct_image_from_patches(sa_prediction_patches[0:64], 8), axis=0)


for i in range(1, int(sa_prediction_patches.shape[0] / 64)):
    reconstructed_image = reconstruct_image_from_patches(sa_prediction_patches[i * 64 : i * 64 + 64], nu
    reconstructed_image = np.expand_dims(reconstructed_image, axis=0)

    sa_predictions= np.append(sa_predictions, reconstructed_image, axis=0)



visualize_predictions(sa_model, noisy_array,  len(noisy_array), sa_predictions, "sa model")
```

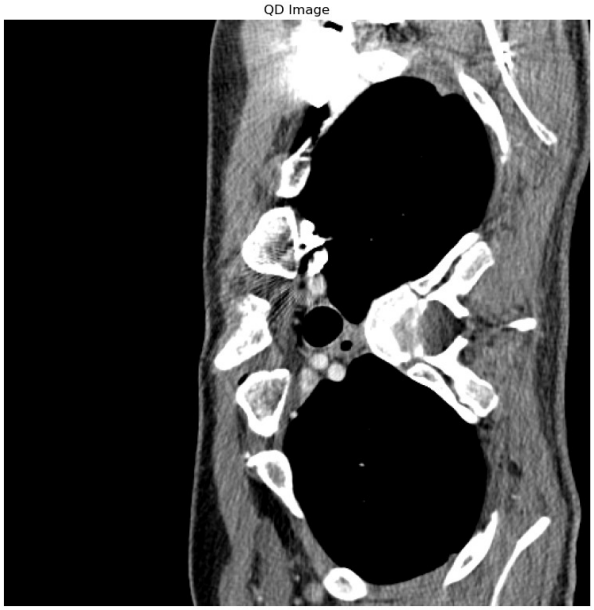


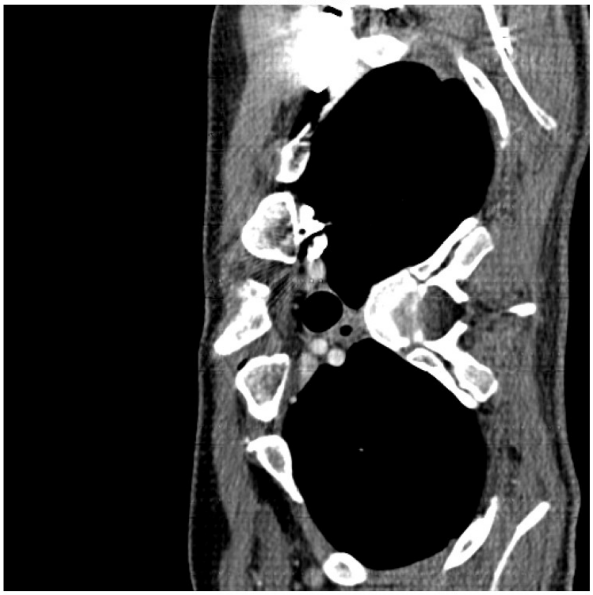




sa model Predicted Image : PSNR=27.4016
SSIM=0.8381
RMSE=291.0442

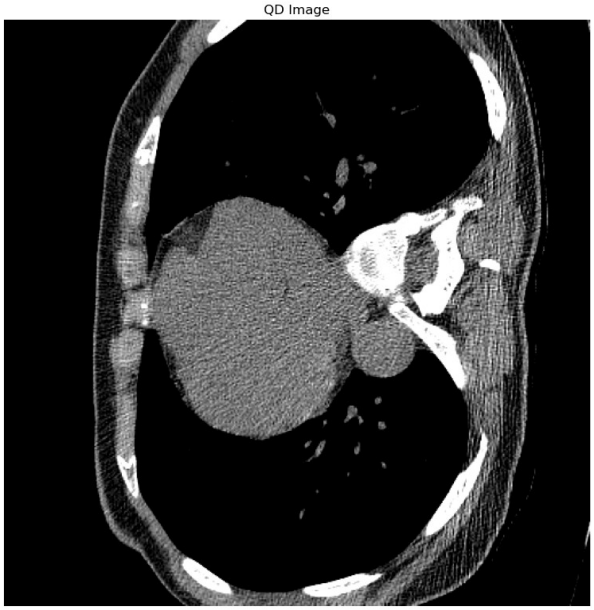
QD Image

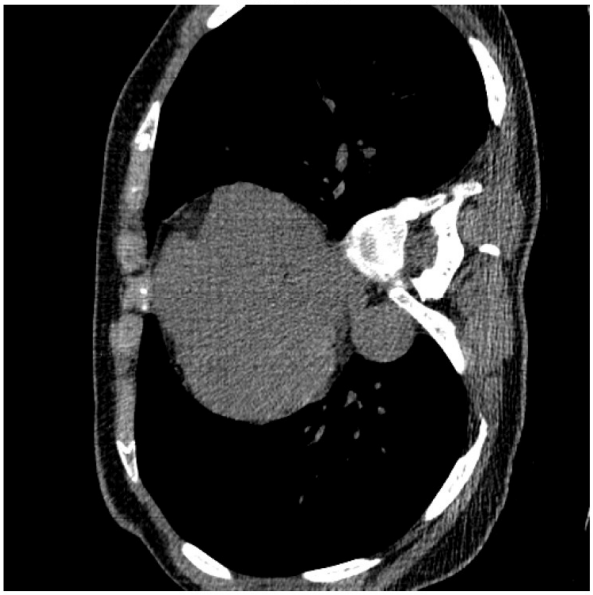




sa model Predicted Image : PSNR=25.9423
SSIM=0.8172
RMSE=407.2804

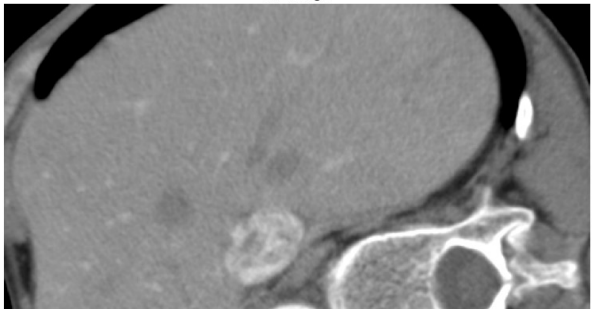
QD Image

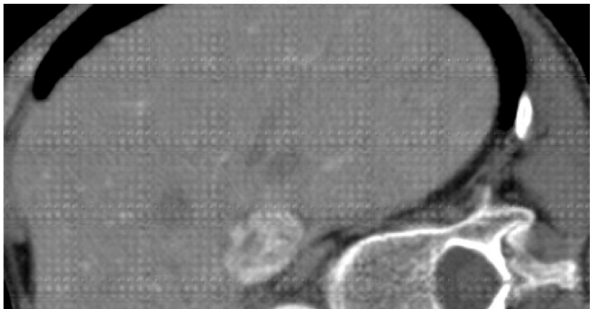


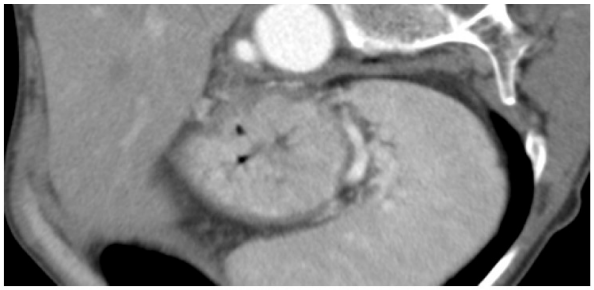


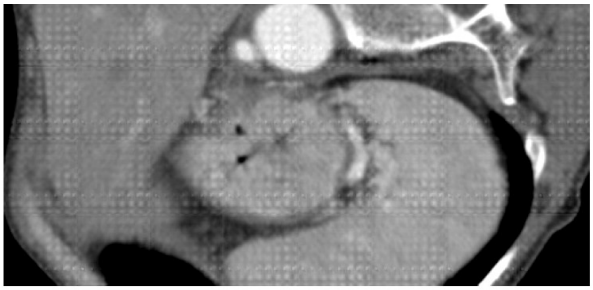
sa model Predicted Image : PSNR=24.2968
SSIM=0.6421
RMSE=594.8936

QD Image









sa model Predicted Image : PSNR=26.7021
SSIM=0.8124
RMSE=341.907

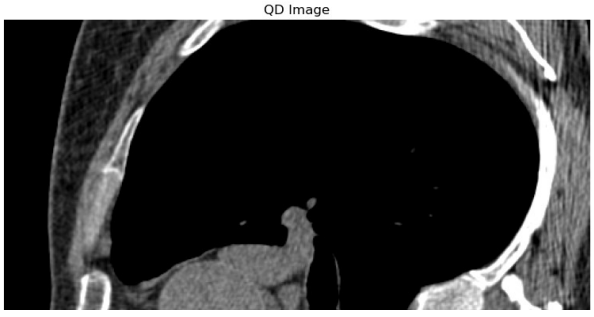
QD Image

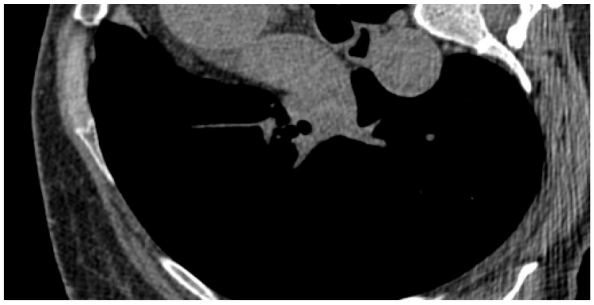
sa model Predicted Image : PSNR=24.9768
SSIM=0.7058
RMSE=508.6717

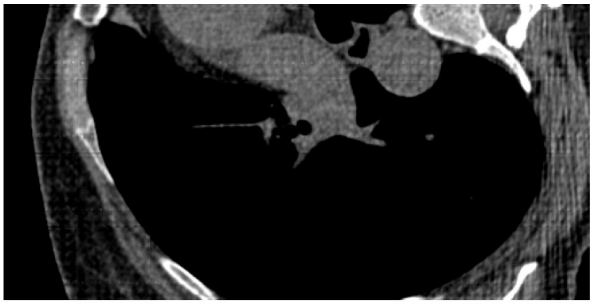
QD Image

sa model Predicted Image : PSNR=27.7264
SSIM=0.8256
RMSE=270.0703

QD Image

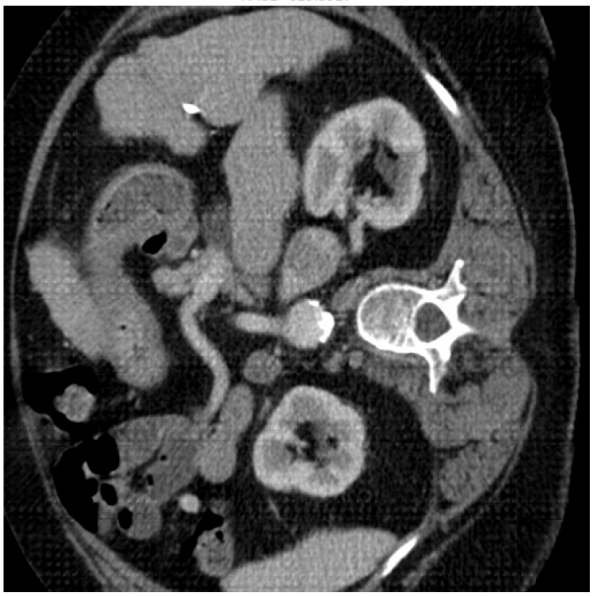






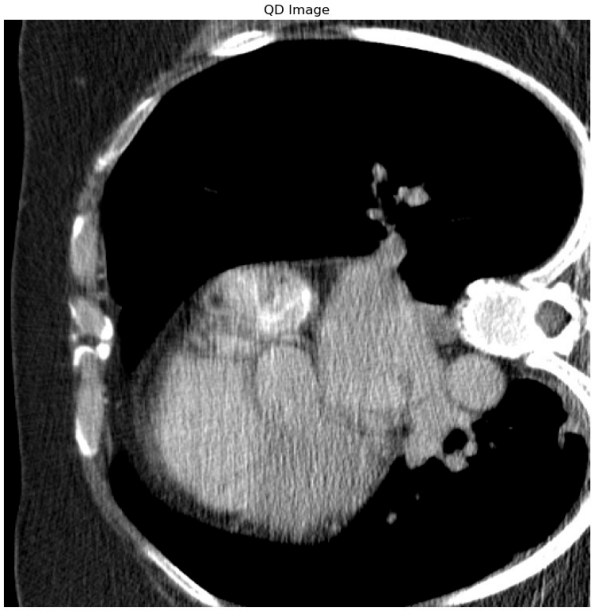
sa model Predicted Image : PSNR=24.0783
SSIM=0.7006
RMSE=625.5927

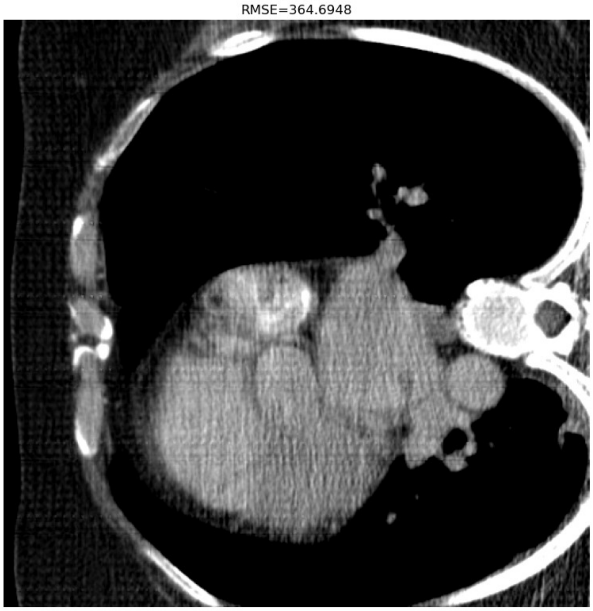
QD Image



sa model Predicted Image : PSNR=26.4219
SSIM=0.8758
RMSE=364.6948

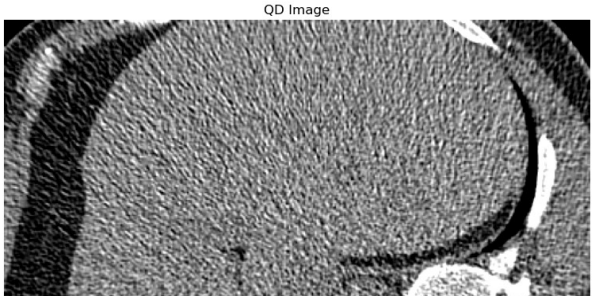
QD Image

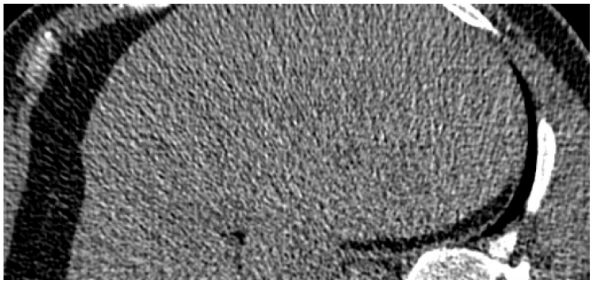


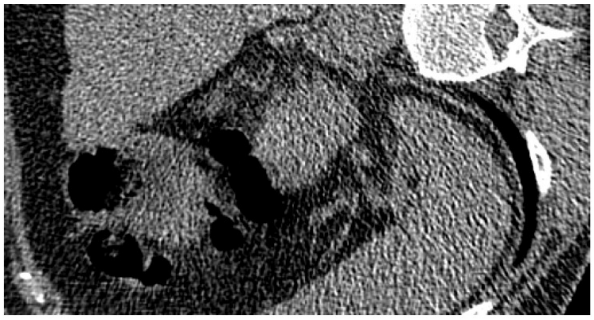


sa model Predicted Image : PSNR=22.743
SSIM=0.6763
RMSE=850.7809

QD Image





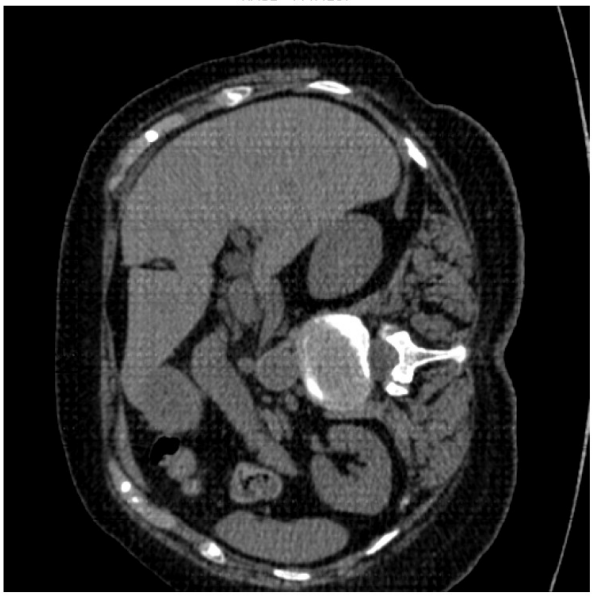


sa model Predicted Image : PSNR=25.9929
SSIM=0.7982
RMSE=402.5555

QD Image

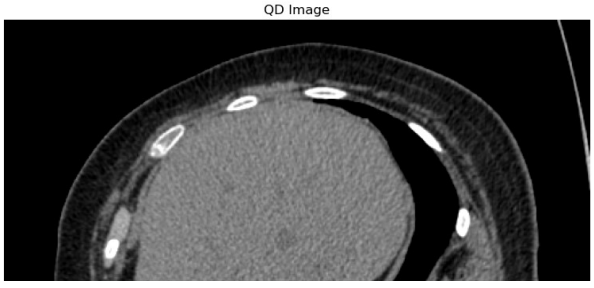
sa model Predicted Image : PSNR=25.5632
SSIM=0.7296
RMSE=444.4287

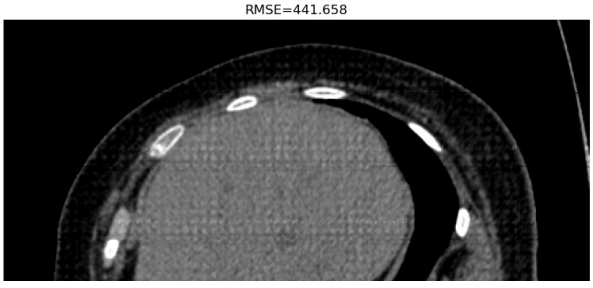
QD Image

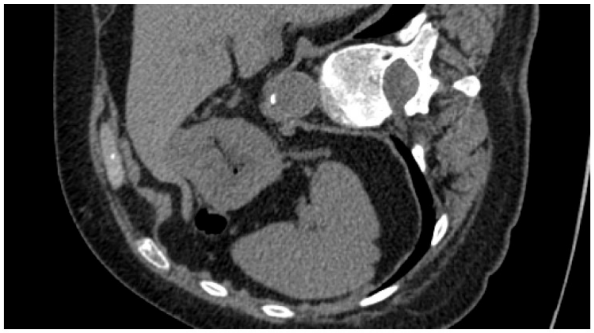


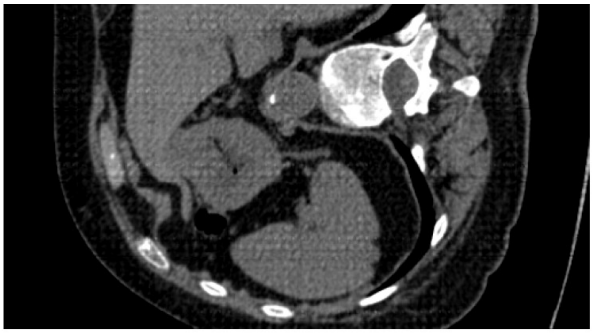
sa model Predicted Image : PSNR=25.5903
SSIM=0.737
RMSE=441.658

QD Image



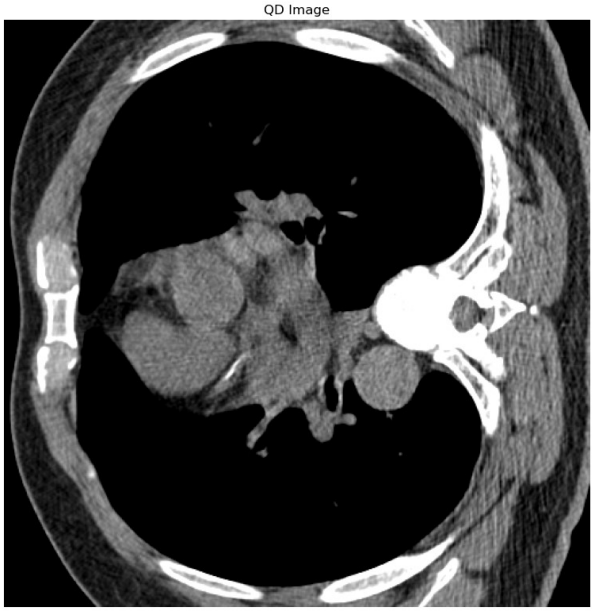


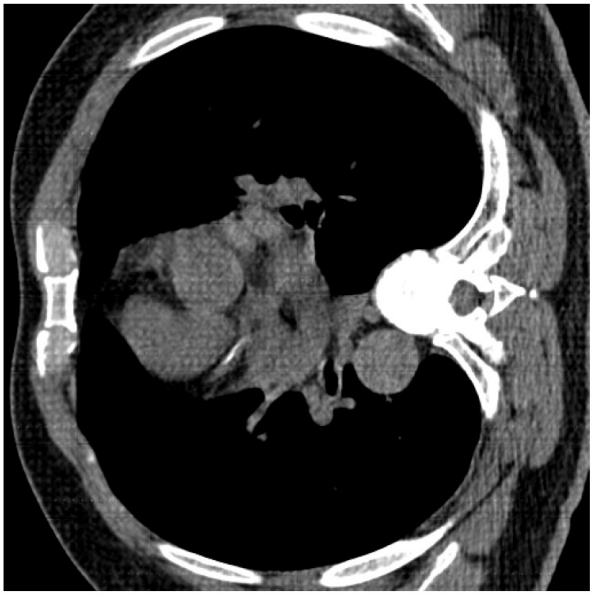




sa model Predicted Image : PSNR=26.6006
SSIM=0.7829
RMSE=349.9951

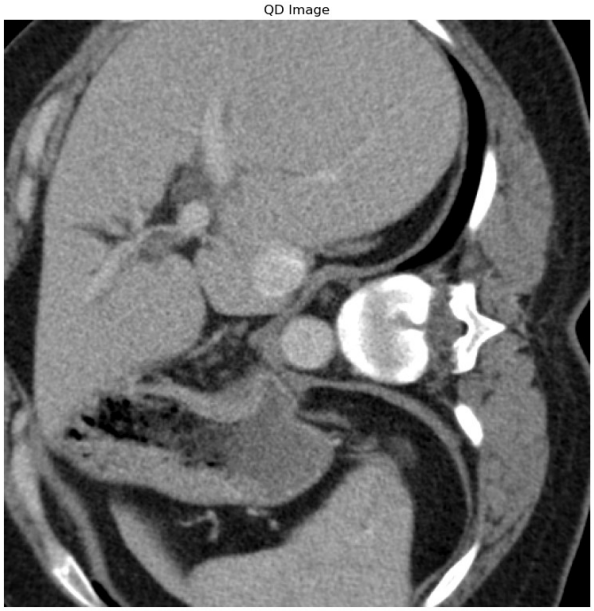
QD Image

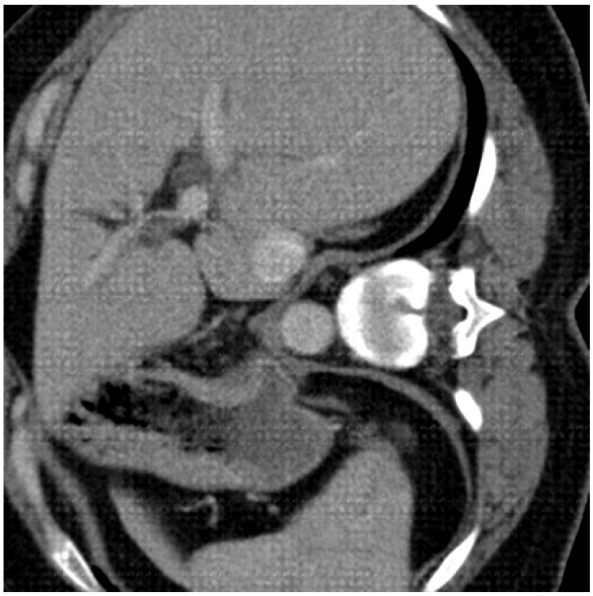




sa model Predicted Image : PSNR=23.985
SSIM=0.6617
RMSE=639.1757

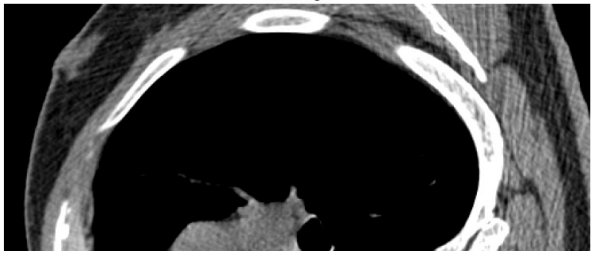
QD Image

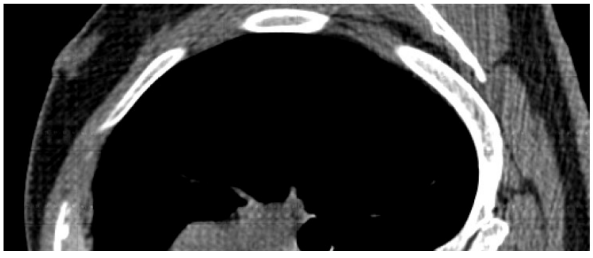




sa model Predicted Image : PSNR=26.8709
SSIM=0.8118
RMSE=328.8745

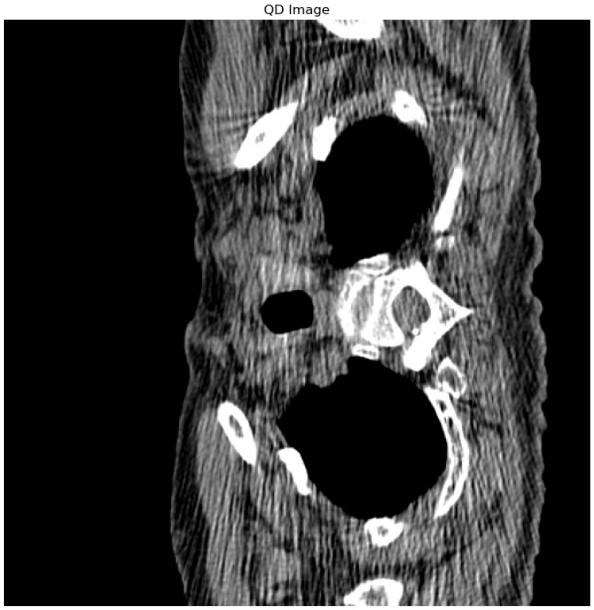
QD Image

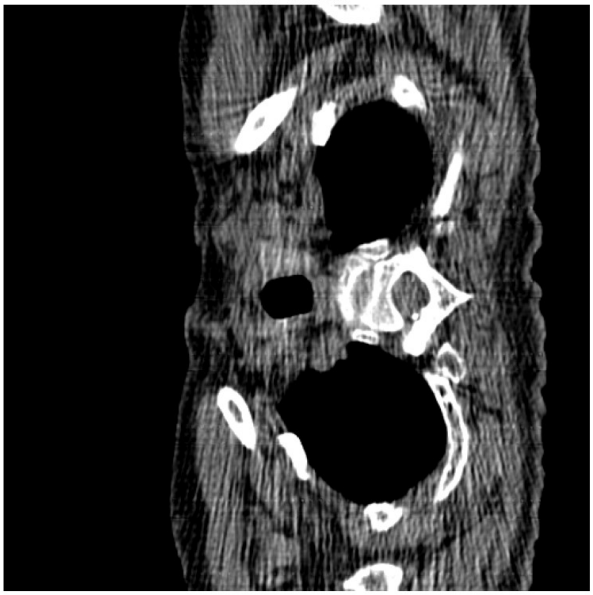




sa model Predicted Image : PSNR=26.6818
SSIM=0.7891
RMSE=343.5106

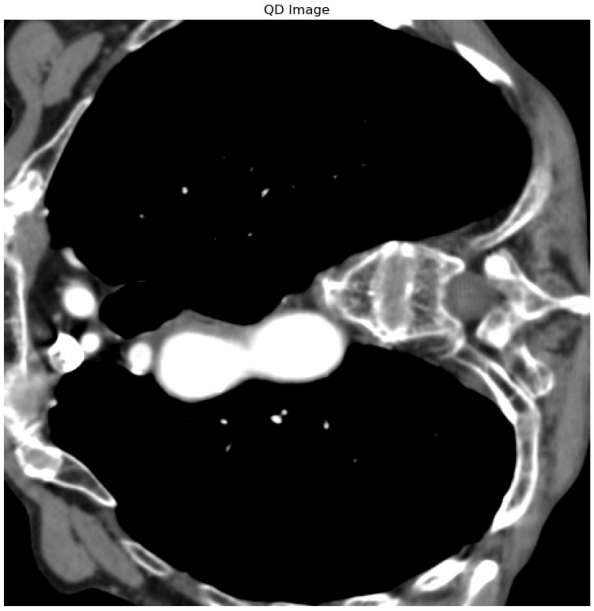
QD Image

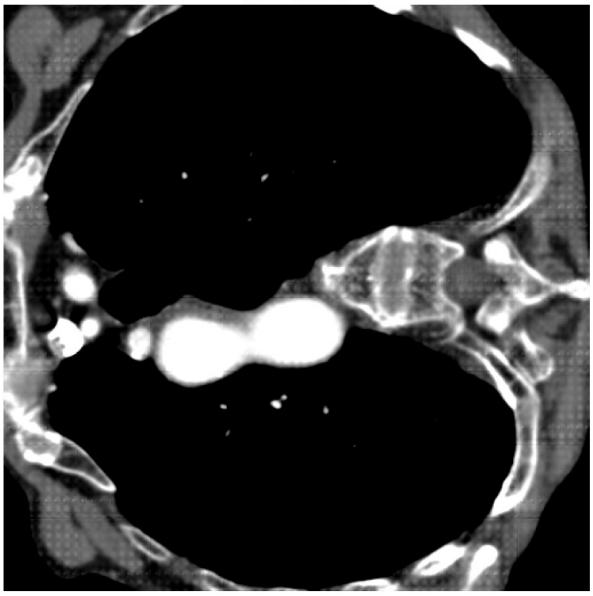




sa model Predicted Image : PSNR=28.5477
SSIM=0.8565
RMSE=223.538

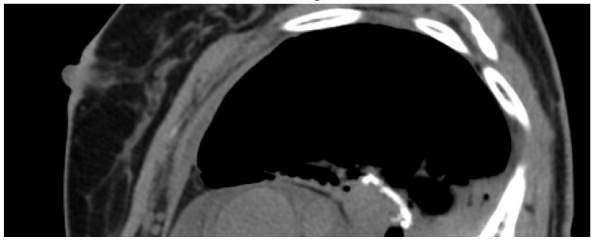
QD Image

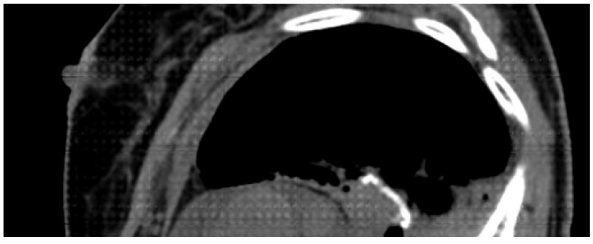


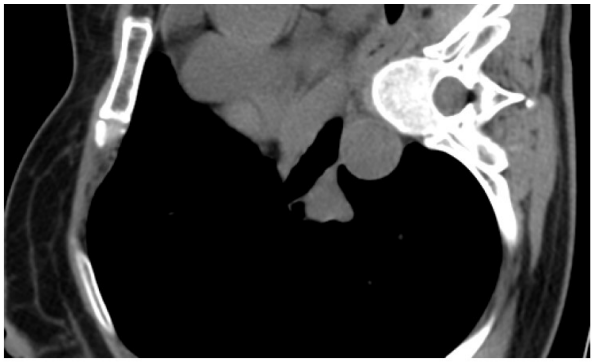


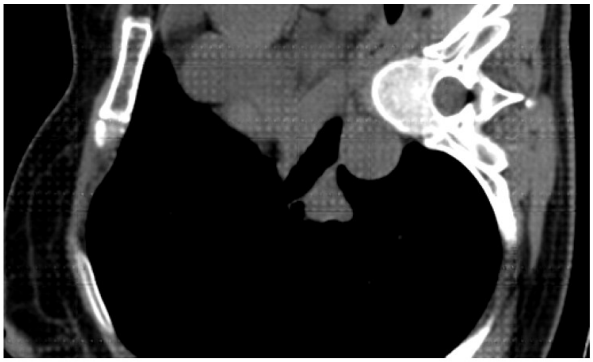
sa model Predicted Image : PSNR=26.9756
SSIM=0.772
RMSE=321.0383

QD Image



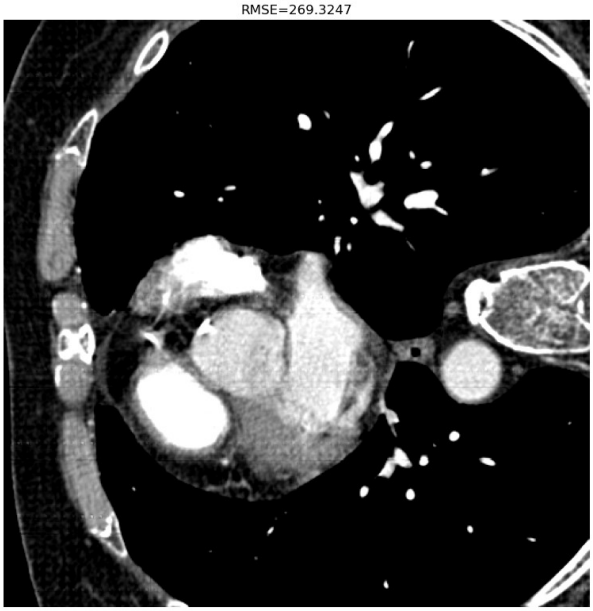






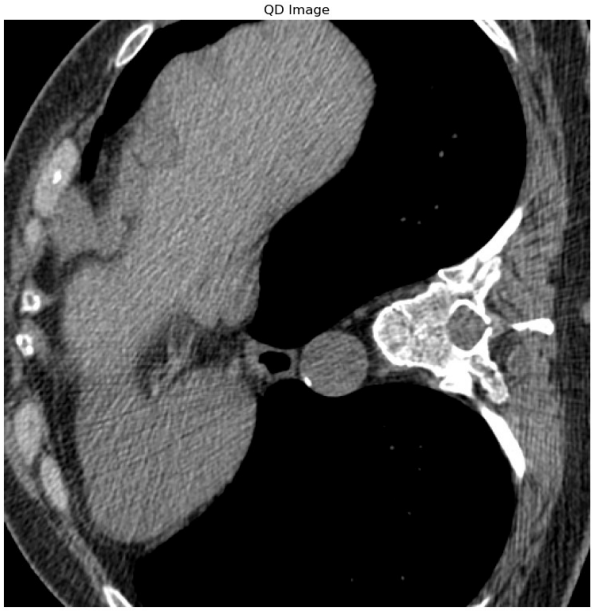
sa model Predicted Image : PSNR=27.7384
SSIM=0.9143
RMSE=269.3247

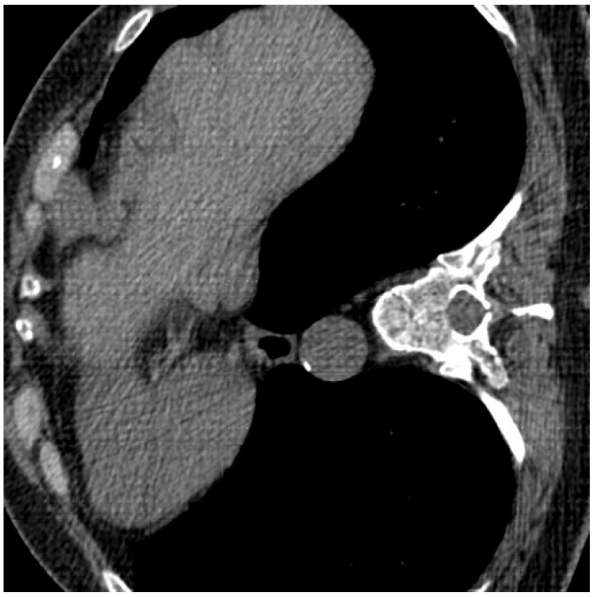
QD Image



sa model Predicted Image : PSNR=25.9151
SSIM=0.7501
RMSE=409.8322

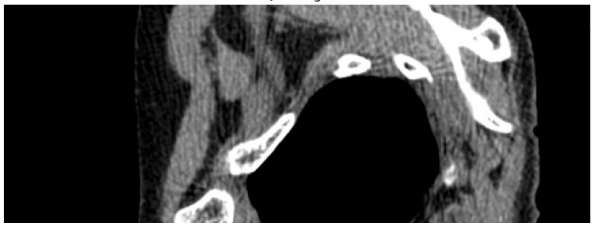
QD Image

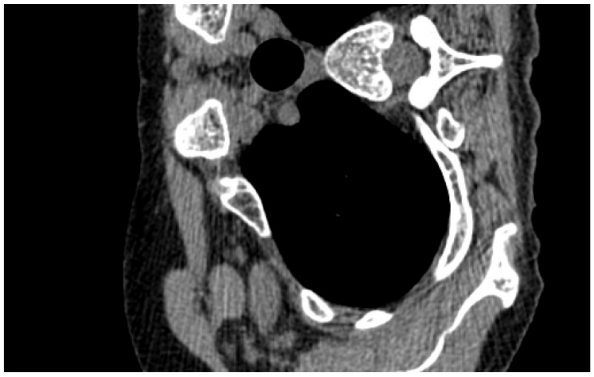


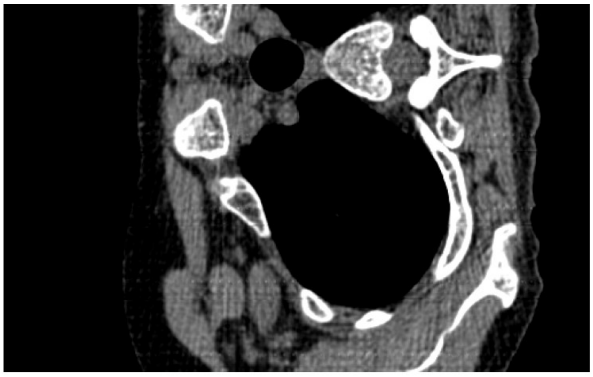


sa model Predicted Image : PSNR=27.2627
SSIM=0.812
RMSE=300.5005

QD Image

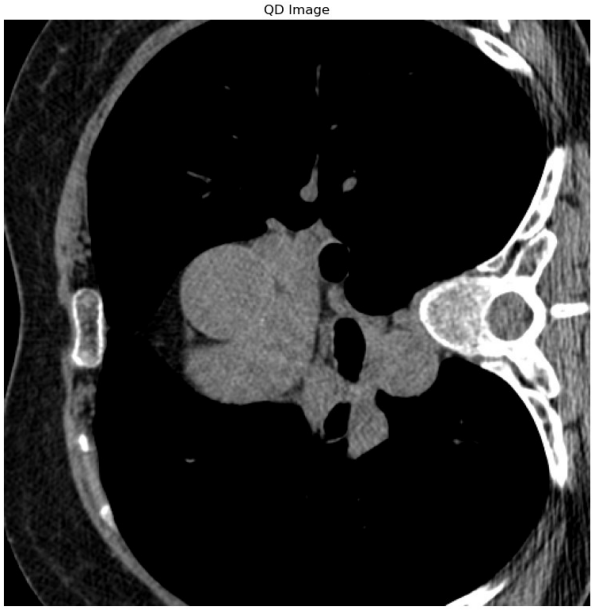






sa model Predicted Image : PSNR=28.0553
SSIM=0.8414
RMSE=250.3721

QD Image

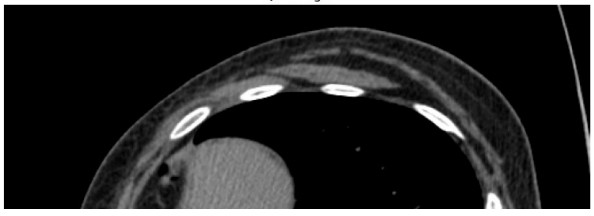


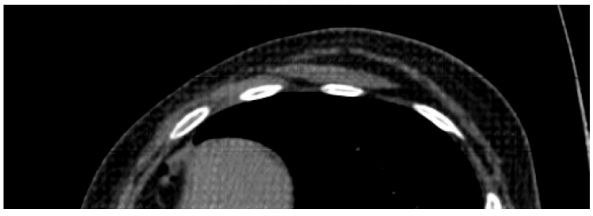
sa model Predicted Image : PSNR=27.2945
SSIM=0.8404
RMSE=298.3129

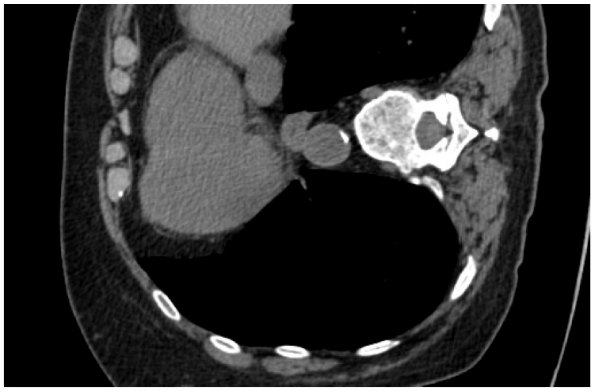
QD Image

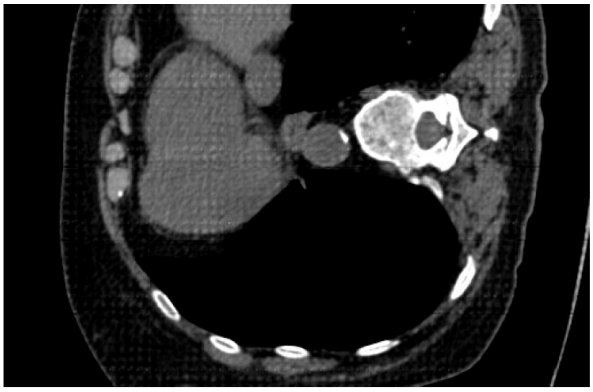
sa model Predicted Image : PSNR=27.0916
SSIM=0.8126
RMSE=312.5817

QD Image



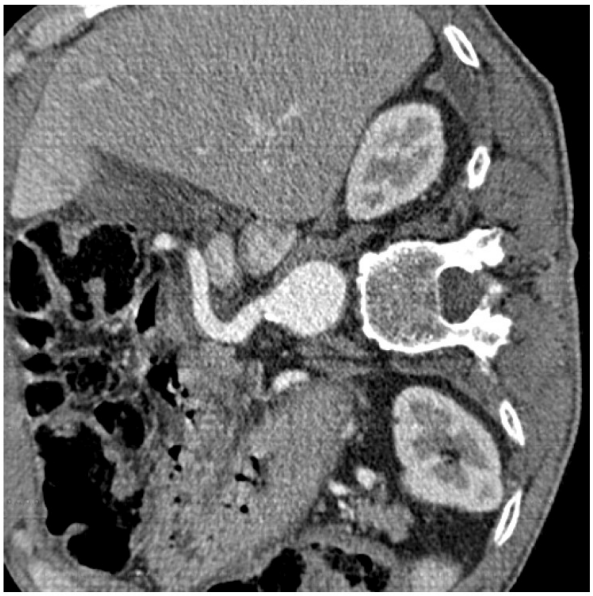






QD Image

sa model Predicted Image : PSNR=24.0689
SSIM=0.6803
RMSE=626.948



QD Image

sa model Predicted Image : PSNR=24.0001
SSIM=0.6469
RMSE=636.9604

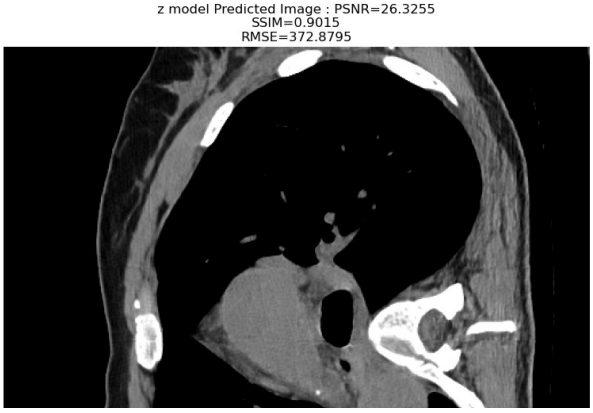
Model 4 : Z Model

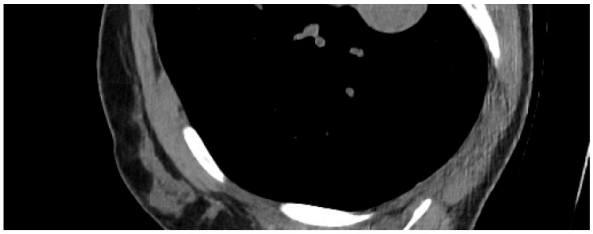
```
sys.path.append('../denoising-models/hformer_pytorch')
from torchinfo import summary

from z_model import ZModel

z_model = ZModel(num_channels=16).cuda()
z_model.load_state_dict(torch.load('../denoising-models/hformer_pytorch/weights/z_model_159.pth'))
z_model.eval()
print('model summary\n', summary(z_model, input_size=(64, 64, 64, 1)))
```

```
Using cache found in C:\Users\Tarun/.cache\torch\hub\mateuszbuda_brain-segmentation-p
Using cache found in C:\Users\Tarun/.cache\torch\hub\mateuszbuda_brain-segmentation-p
model summary
 ==========================================================================================
Layer (type:depth-idx)                             Output Shape              Param #
==========================================================================================
ZModel                                             [64, 64, 64, 1]           --
├─InputProjectionLayer: 1-1                        [64, 16, 64, 64]          --
│    └─Conv2d: 2-1                                 [64, 16, 64, 64]          160
├─UNet: 1-2                                        [64, 16, 64, 64]          --
│    └─Sequential: 2-2                             [64, 16, 64, 64]          --
│    │    └─Conv2d: 3-1                            [64, 16, 64, 64]          2,304
│    │    └─BatchNorm2d: 3-2                       [64, 16, 64, 64]          32
│    │    └─ReLU: 3-3                              [64, 16, 64, 64]          --
│    │    └─Conv2d: 3-4                            [64, 16, 64, 64]          2,304
│    │    └─BatchNorm2d: 3-5                       [64, 16, 64, 64]          32
│    │    └─ReLU: 3-6                              [64, 16, 64, 64]          --
│    └─MaxPool2d: 2-3                              [64, 16, 32, 32]          --
│    └─Sequential: 2-4                             [64, 32, 32, 32]          --
│    │    └─Conv2d: 3-7                            [64, 32, 32, 32]          4,608
│    │    └─BatchNorm2d: 3-8                       [64, 32, 32, 32]          64
│    │    └─ReLU: 3-9                              [64, 32, 32, 32]          --
│    │    └─Conv2d: 3-10                           [64, 32, 32, 32]          9,216
│    │    └─BatchNorm2d: 3-11                      [64, 32, 32, 32]          64
│    │    └─ReLU: 3-12                             [64, 32, 32, 32]          --
│    └─MaxPool2d: 2-5                              [64, 32, 16, 16]          --
│    └─Sequential: 2-6                             [64, 64, 16, 16]          --
│    │    └─Conv2d: 3-13                           [64, 64, 16, 16]          18,432
│    │    └─BatchNorm2d: 3-14                      [64, 64, 16, 16]          128
│    │    └─ReLU: 3-15                             [64, 64, 16, 16]          --
│    │    └─Conv2d: 3-16                           [64, 64, 16, 16]          36,864
│    │    └─BatchNorm2d: 3-17                      [64, 64, 16, 16]          128
│    │    └─ReLU: 3-18                             [64, 64, 16, 16]          --
│    └─MaxPool2d: 2-7                              [64, 64, 8, 8]            --
│    └─Sequential: 2-8                             [64, 128, 8, 8]           --
│    │    └─Conv2d: 3-19                           [64, 128, 8, 8]           73,728
│    │    └─BatchNorm2d: 3-20                      [64, 128, 8, 8]           256
│    │    └─ReLU: 3-21                             [64, 128, 8, 8]           --
```

```
|    |    └─Conv2d: 3-22                      [64, 128, 8, 8]           147,456
|    |    └─BatchNorm2d: 3-23                 [64, 128, 8, 8]           256
|    |    └─ReLU: 3-24                        [64, 128, 8, 8]           --
|    └─MaxPool2d: 2-9                         [64, 128, 4, 4]           --
|    └─Sequential: 2-10                       [64, 256, 4, 4]           --
|    |    └─Conv2d: 3-25                      [64, 256, 4, 4]           294,912
|    |    └─BatchNorm2d: 3-26                 [64, 256, 4, 4]           512
|    |    └─ReLU: 3-27                        [64, 256, 4, 4]           --
|    |    └─Conv2d: 3-28                      [64, 256, 4, 4]           589,824
|    |    └─BatchNorm2d: 3-29                 [64, 256, 4, 4]           512
|    |    └─ReLU: 3-30                        [64, 256, 4, 4]           --
|    └─ConvTranspose2d: 2-11                  [64, 128, 8, 8]           131,200
|    └─Sequential: 2-12                       [64, 128, 8, 8]           --
|    |    └─Conv2d: 3-31                      [64, 128, 8, 8]           294,912
|    |    └─BatchNorm2d: 3-32                 [64, 128, 8, 8]           256
|    |    └─ReLU: 3-33                        [64, 128, 8, 8]           --
|    |    └─Conv2d: 3-34                      [64, 128, 8, 8]           147,456
|    |    └─BatchNorm2d: 3-35                 [64, 128, 8, 8]           256
|    |    └─ReLU: 3-36                        [64, 128, 8, 8]           --
|    └─ConvTranspose2d: 2-13                  [64, 64, 16, 16]          32,832
```

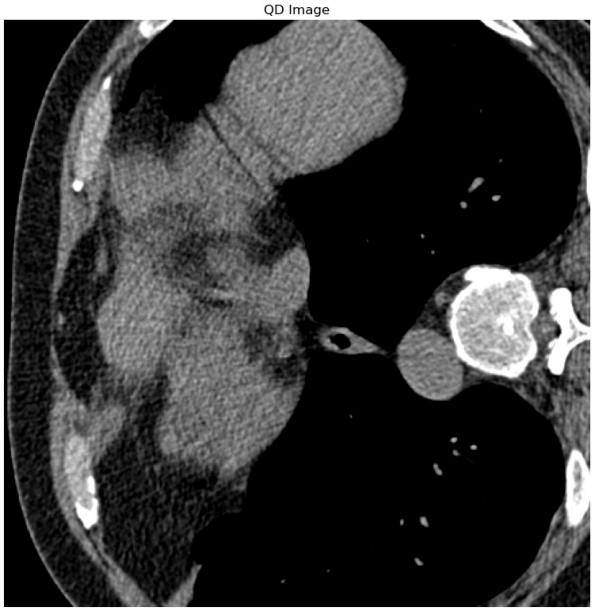
```
z_prediction_patches = []

with torch.no_grad():
    for i, data in enumerate(noisy_image_patches_array):
        noisy = data

        predictions = z_model(torch.unsqueeze(torch.from_numpy(noisy.numpy()), dim=0).to('cuda')).cpu()

        z_prediction_patches.append(predictions.detach().cpu())

z_prediction_patches = np.concatenate(z_prediction_patches, axis=0)
z_predictions = np.expand_dims(reconstruct_image_from_patches(z_prediction_patches[0:64], 8), axis=0)


for i in range(1, int(z_prediction_patches.shape[0] / 64)):
    reconstructed_image = reconstruct_image_from_patches(z_prediction_patches[i * 64 : i * 64 + 64], num
    reconstructed_image = np.expand_dims(reconstructed_image, axis=0)

    z_predictions= np.append(z_predictions, reconstructed_image, axis=0)
visualize_predictions(z_model, noisy_array,  len(noisy_array), z_predictions, "z model")
```





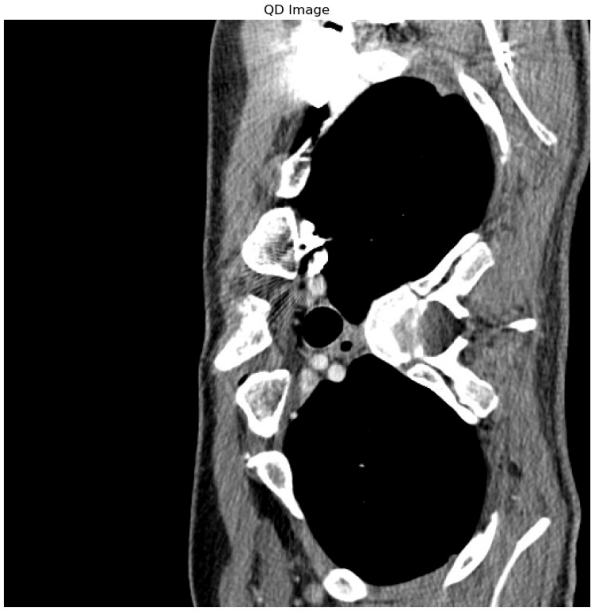
z model Predicted Image : PSNR=32.8672
SSIM=0.9278
RMSE=82.6796

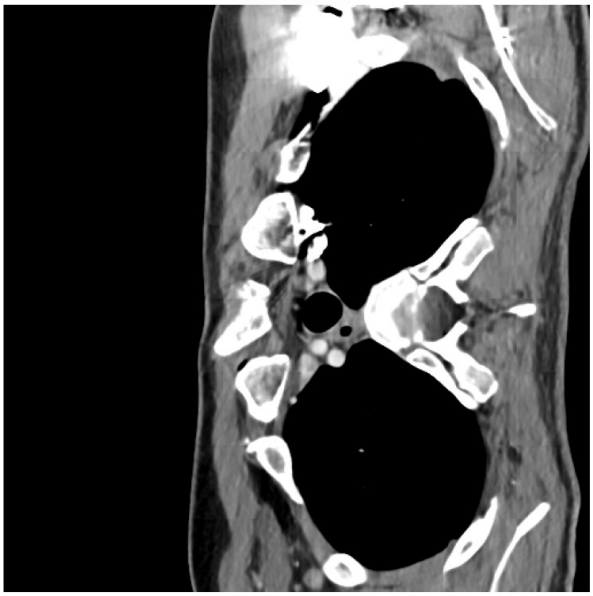
QD Image



z model Predicted Image : PSNR=34.0757
SSIM=0.9649
RMSE=62.5972

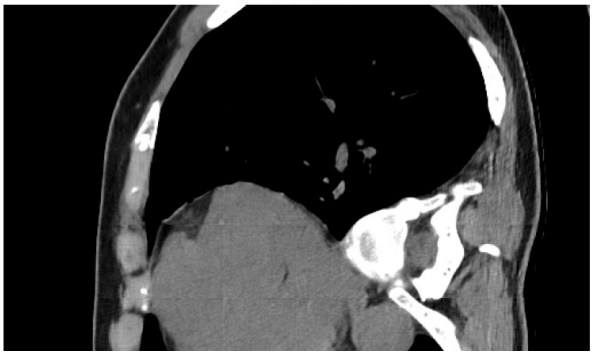
QD Image

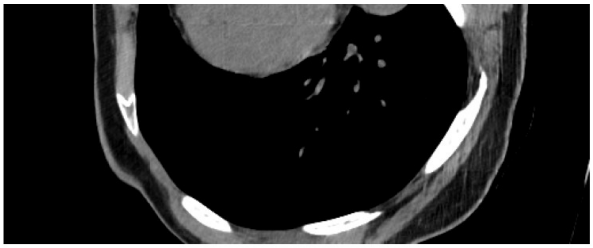




z model Predicted Image : PSNR=25.6435
SSIM=0.8803
RMSE=436.2883

QD Image





z model Predicted Image : PSNR=37.4587
SSIM=0.9524
RMSE=28.7242

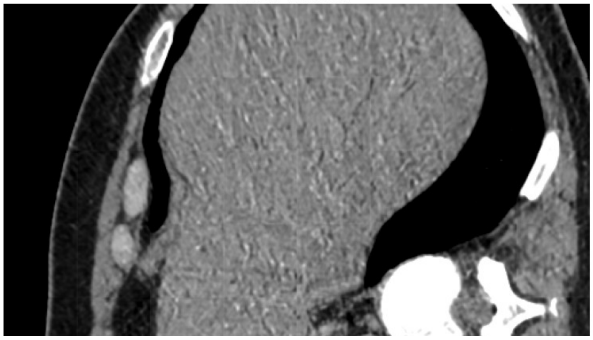
QD Image

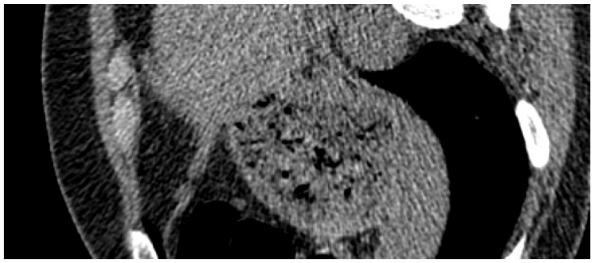
z model Predicted Image : PSNR=32.3487
SSIM=0.9464
RMSE=93.1638

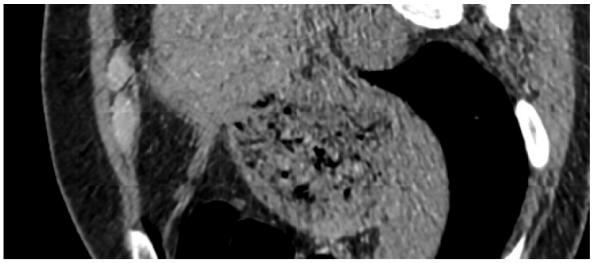
QD Image

z model Predicted Image : PSNR=28.0054
SSIM=0.8742
RMSE=253.2674

QD Image

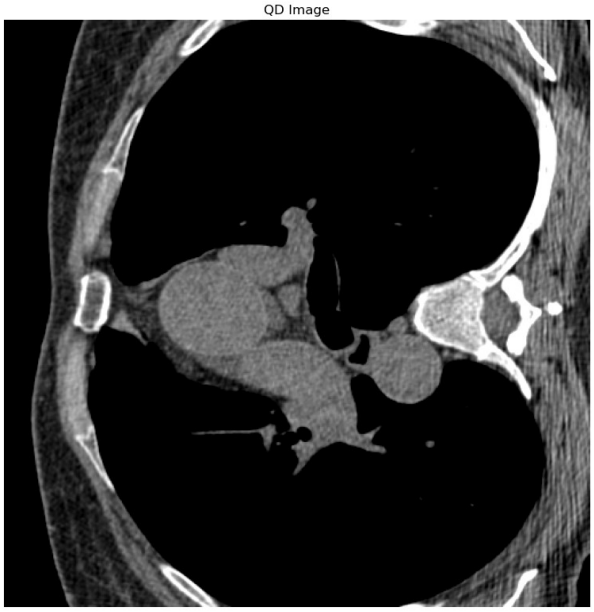






z model Predicted Image : PSNR=35.354
SSIM=0.9636
RMSE=46.6361

QD Image

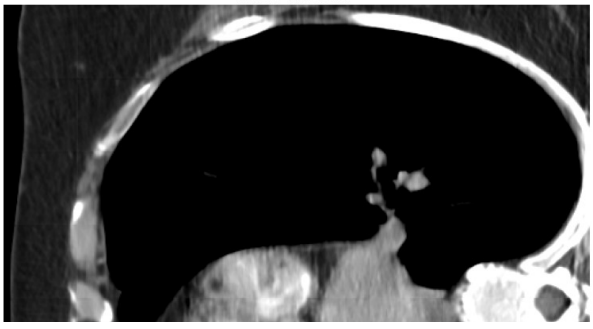


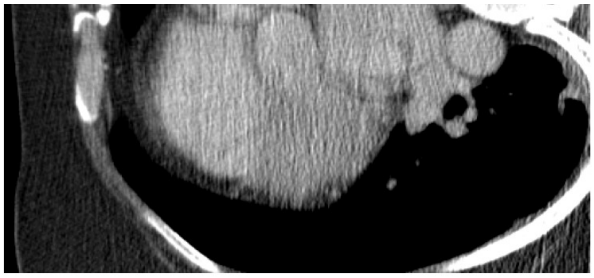
z model Predicted Image : PSNR=31.834
SSIM=0.8805
RMSE=104.8861

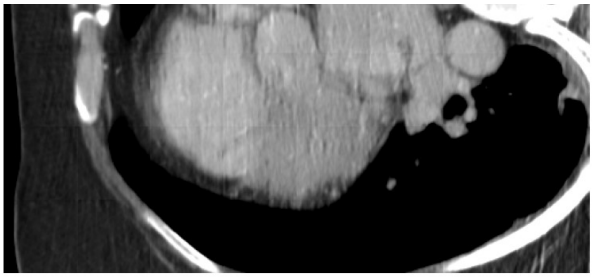
QD Image

z model Predicted Image : PSNR=31.2864
SSIM=0.9128
RMSE=118.9826

QD Image

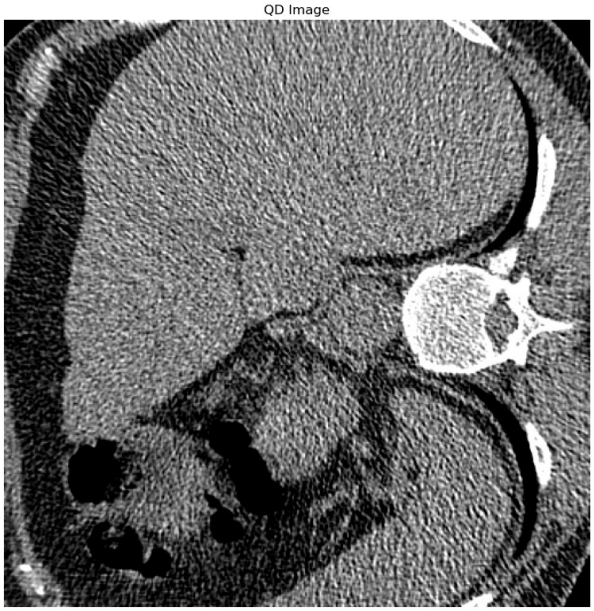


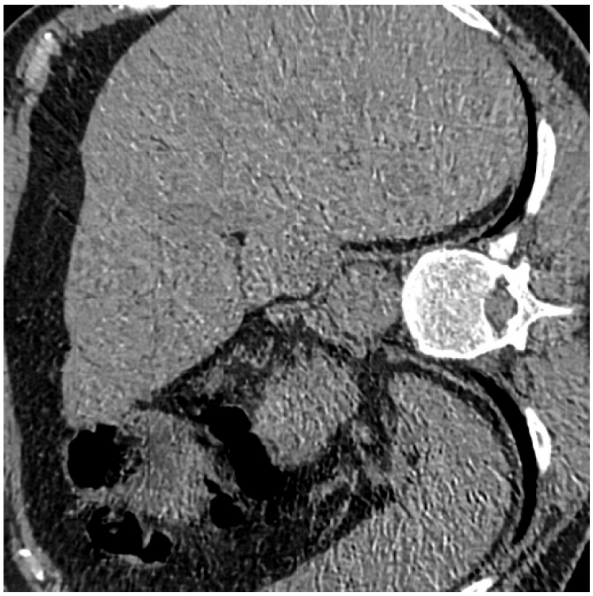




z model Predicted Image : PSNR=21.5394
SSIM=0.7742
RMSE=1122.4785

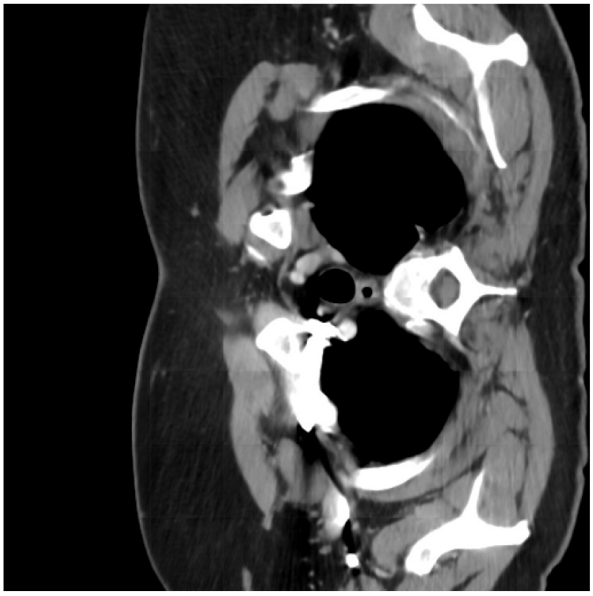
QD Image





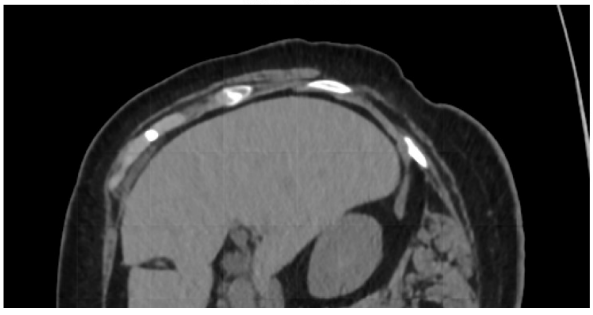
z model Predicted Image : PSNR=34.3931
SSIM=0.9388
RMSE=58.1853

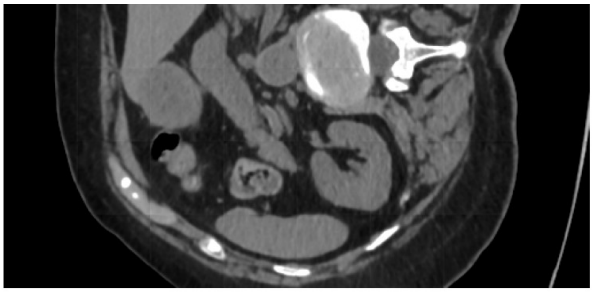
QD Image



z model Predicted Image : PSNR=33.5065
SSIM=0.9142
RMSE=71.3623

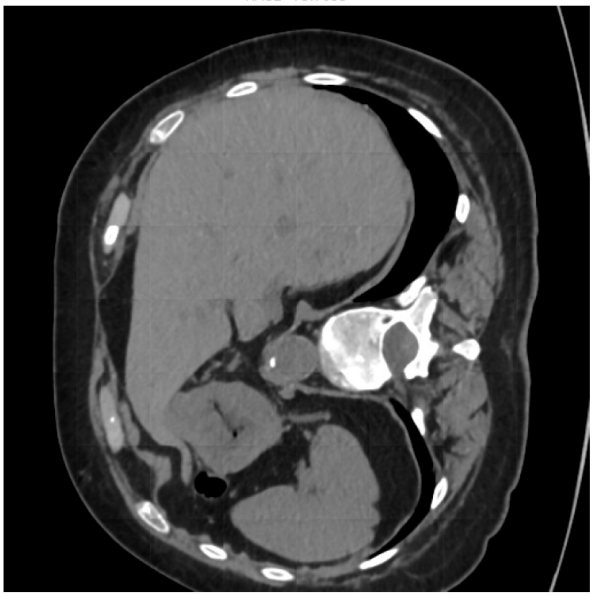
QD Image





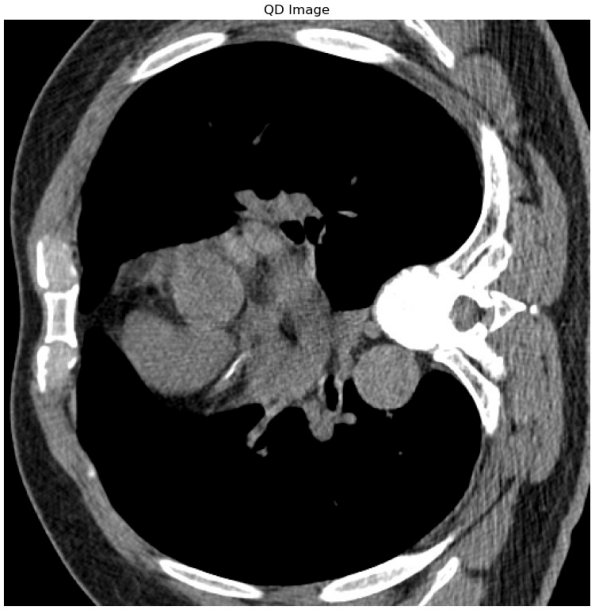
z model Predicted Image : PSNR=33.3628
SSIM=0.9015
RMSE=73.7633

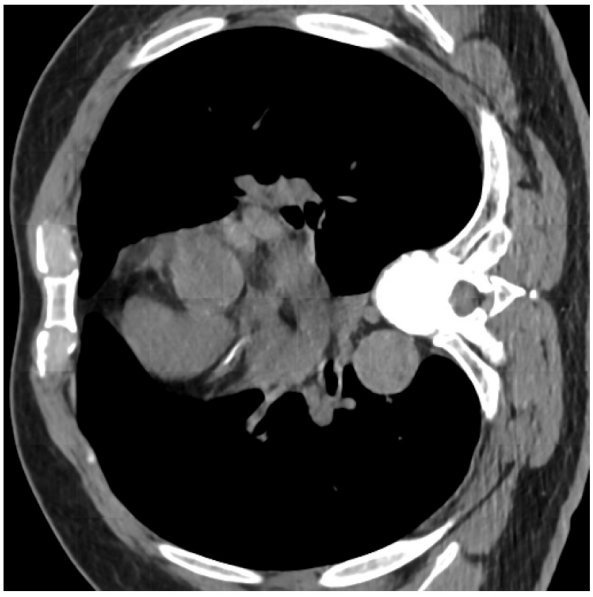
QD Image



z model Predicted Image : PSNR=33.9679
SSIM=0.9497
RMSE=64.1691

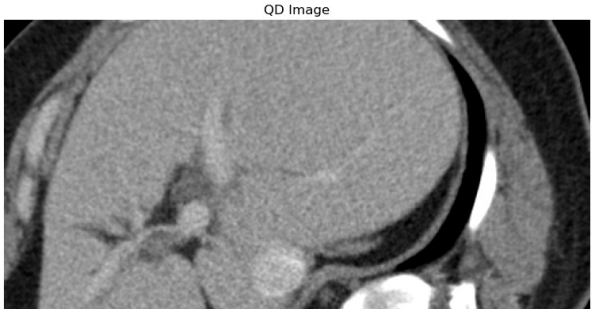
QD Image

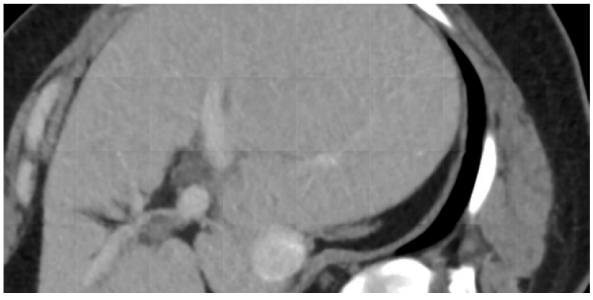


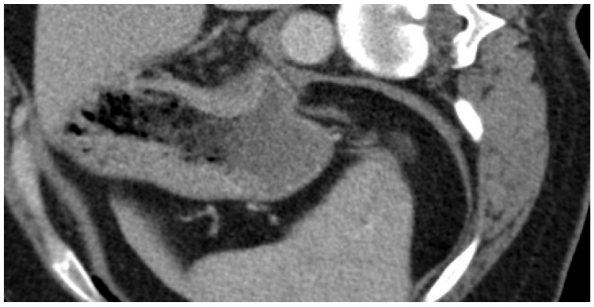


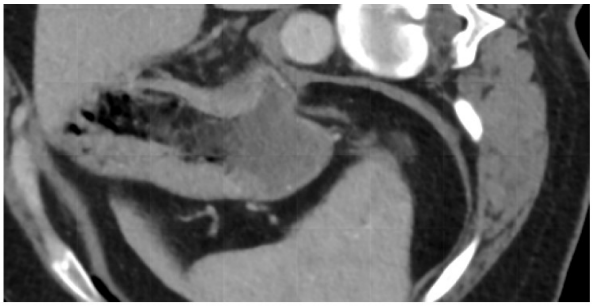
z model Predicted Image : PSNR=33.8642
SSIM=0.8909
RMSE=65.7204

QD Image







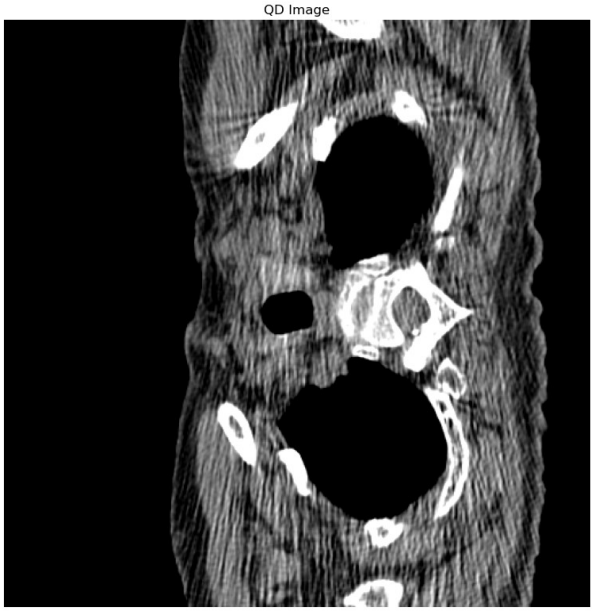


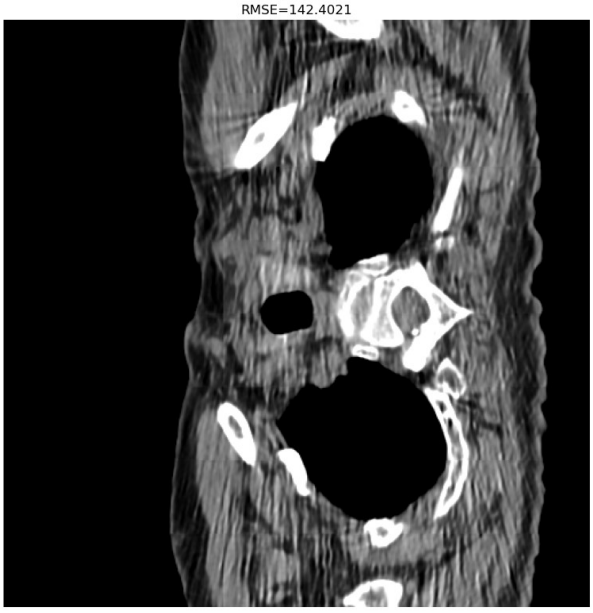
z model Predicted Image : PSNR=33.5935
SSIM=0.9475
RMSE=69.9468

QD Image

z model Predicted Image : PSNR=30.506
SSIM=0.9565
RMSE=142.4021

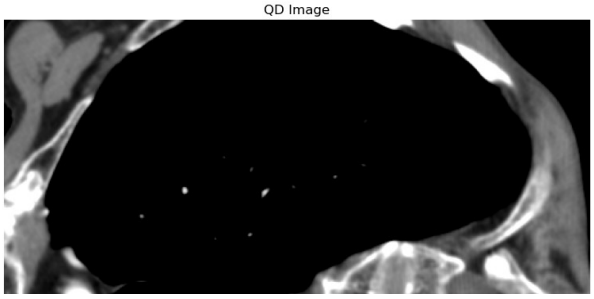
QD Image

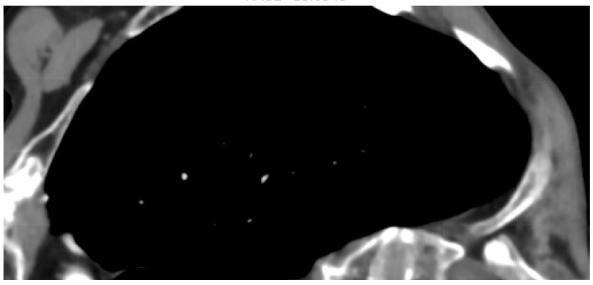


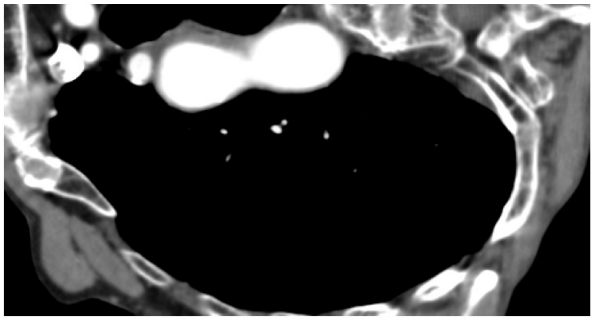


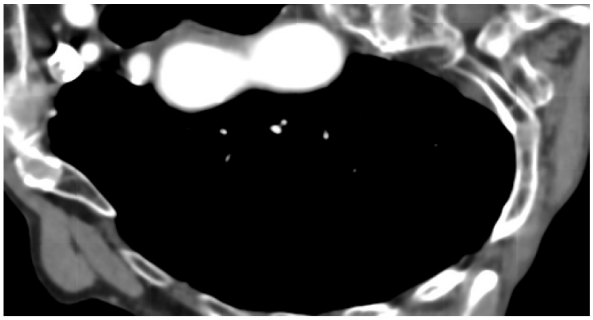
z model Predicted Image : PSNR=40.8836
SSIM=0.9864
RMSE=13.0545

QD Image



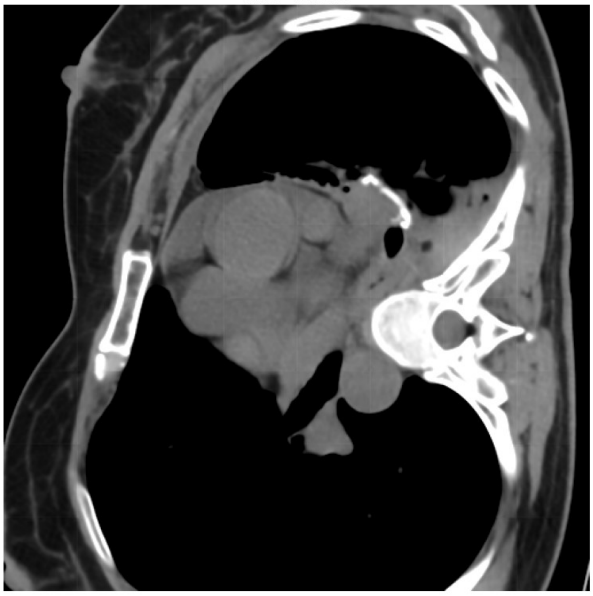






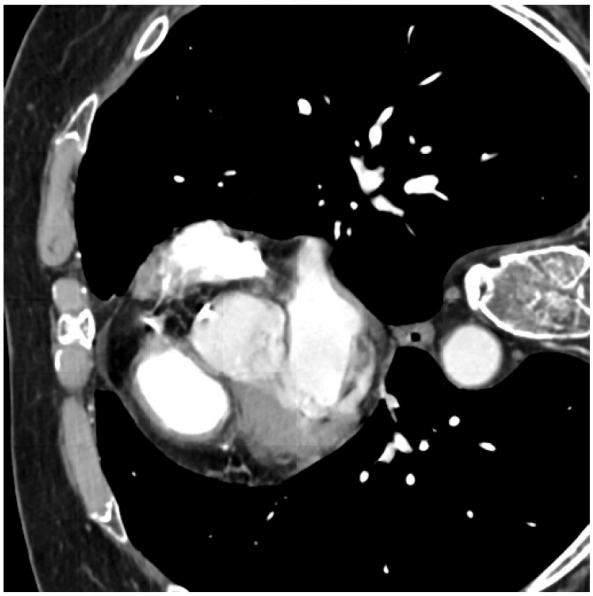
z model Predicted Image : PSNR=39.4386
SSIM=0.9704
RMSE=18.2078

QD Image



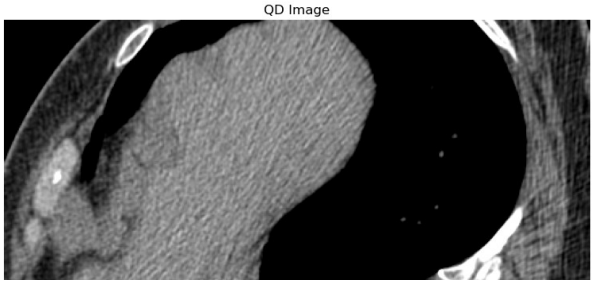
z model Predicted Image : PSNR=34.5582
SSIM=0.9618
RMSE=56.0151

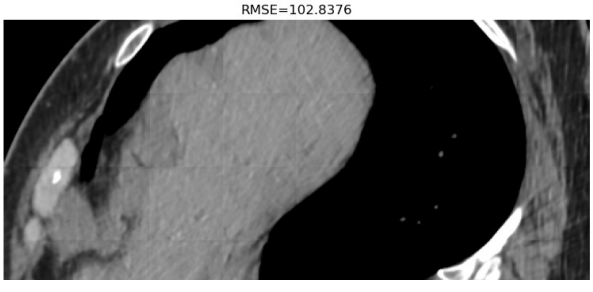
QD Image

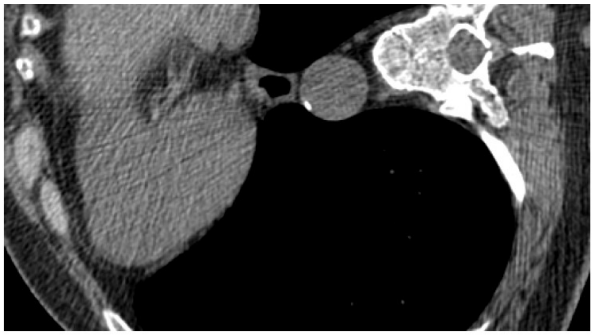


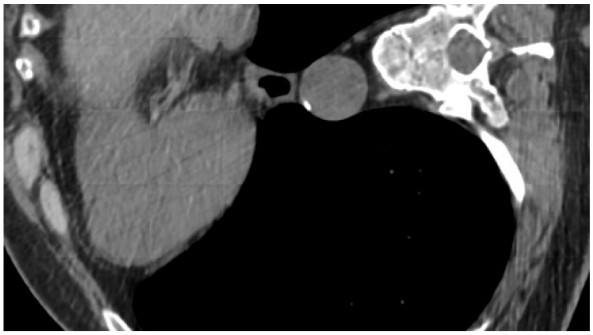
z model Predicted Image : PSNR=31.9197
SSIM=0.8973
RMSE=102.8376

QD Image



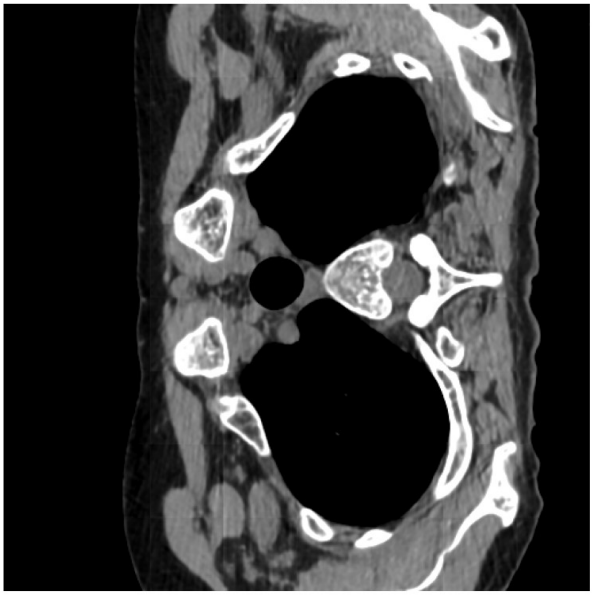






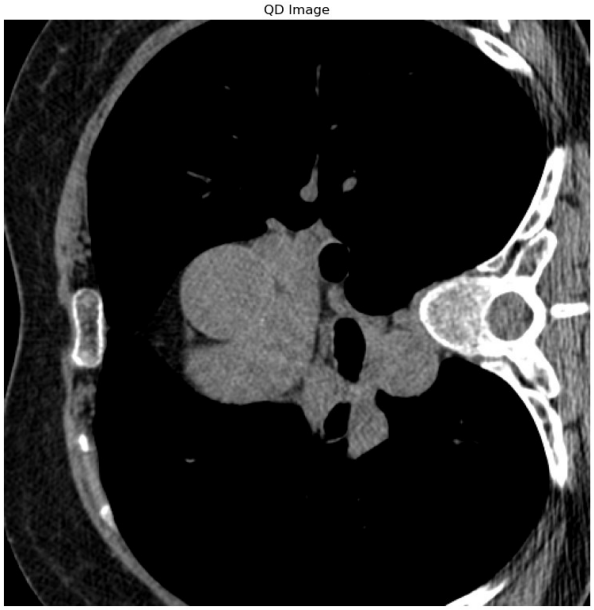
z model Predicted Image : PSNR=34.4536
SSIM=0.9607
RMSE=57.3799

QD Image



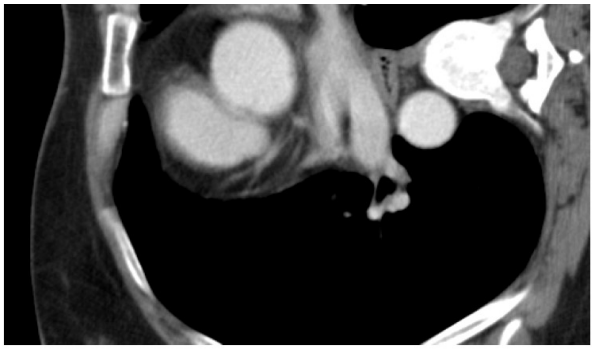
z model Predicted Image : PSNR=36.0778
SSIM=0.9663
RMSE=39.477

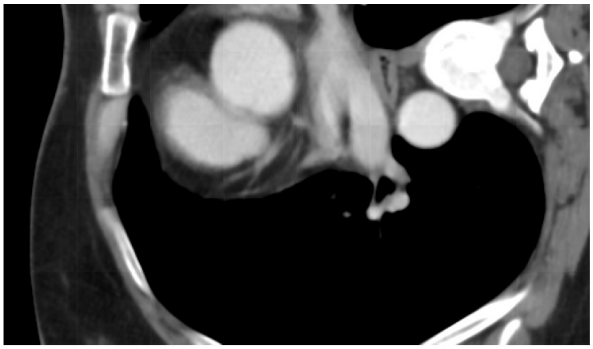
QD Image



z model Predicted Image : PSNR=39.7096
SSIM=0.9776
RMSE=17.1066

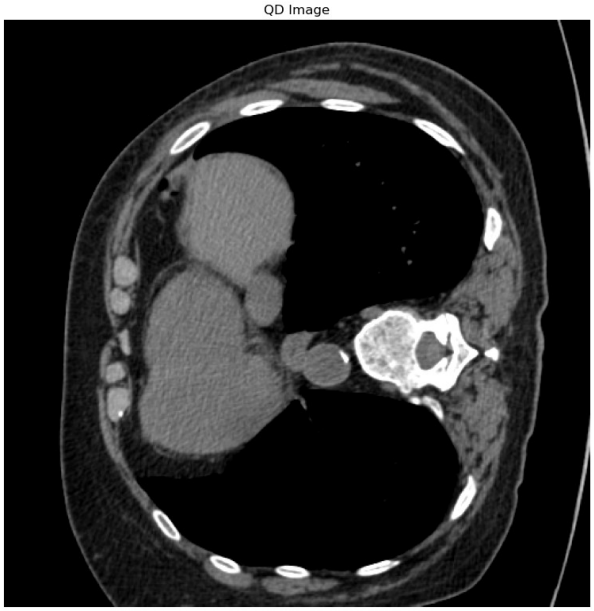
QD Image

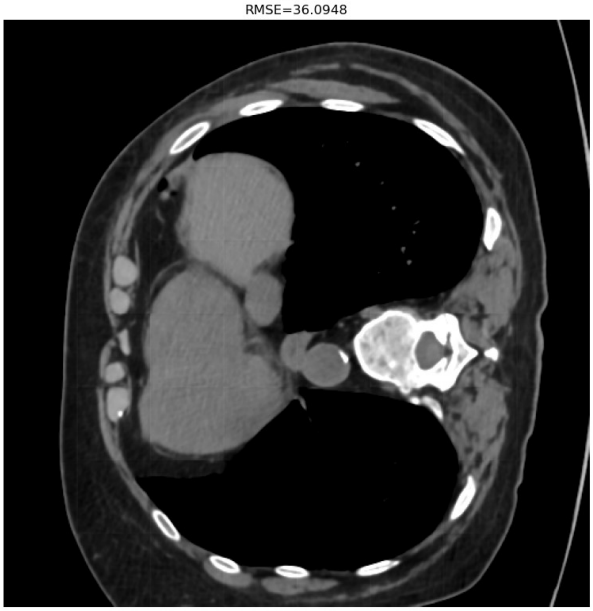




z model Predicted Image : PSNR=36.4668
SSIM=0.9574
RMSE=36.0948

QD Image



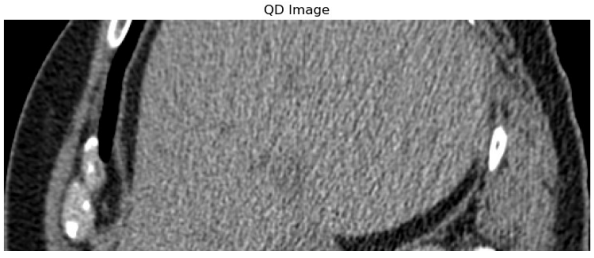


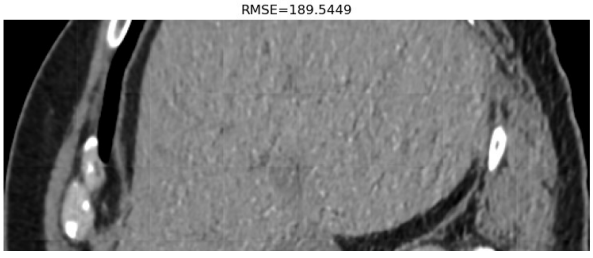
z model Predicted Image : PSNR=27.7374
SSIM=0.8463
RMSE=269.3885

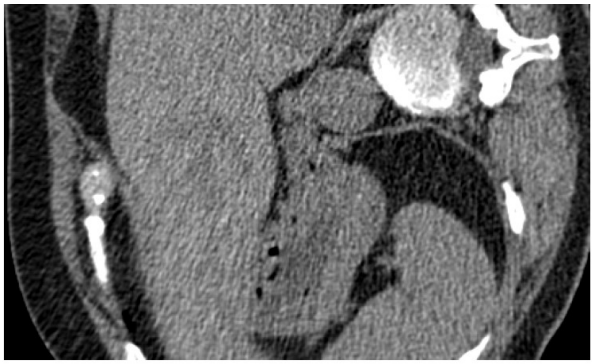
QD Image

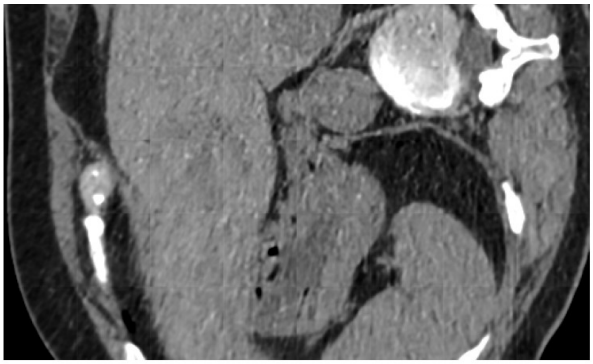
z model Predicted Image : PSNR=29.2641
SSIM=0.8448
RMSE=189.5449

QD Image









Model 5 : W Model

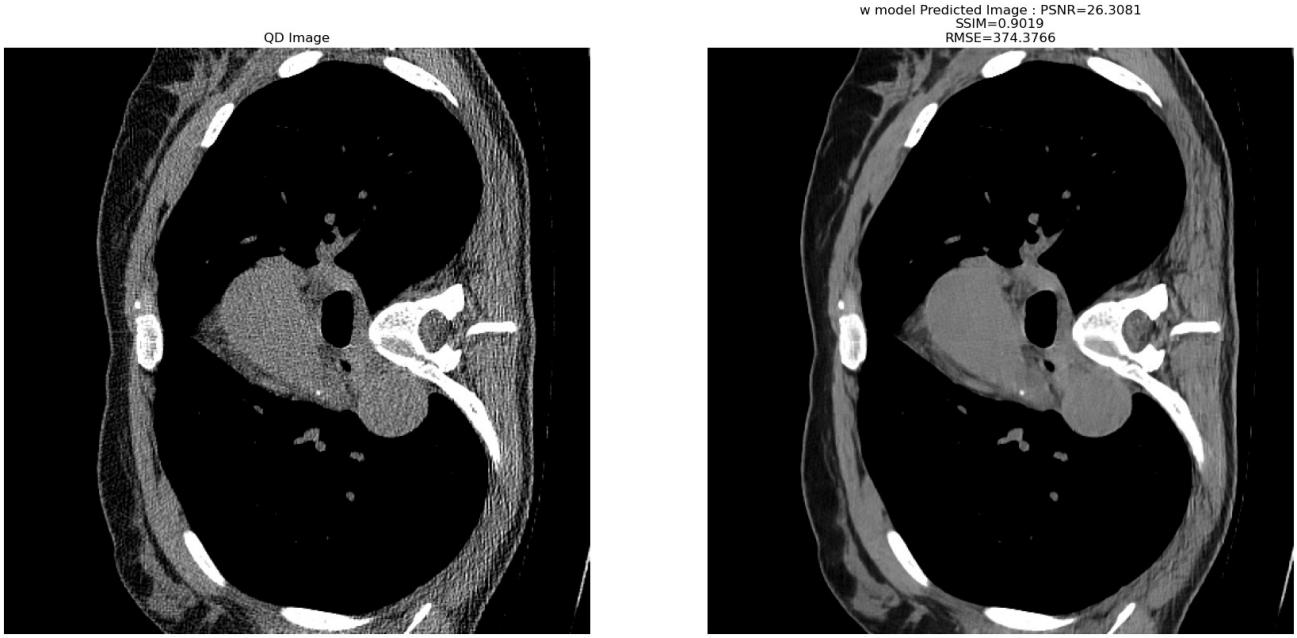
```
sys.path.append('../denoising-models/hformer_pytorch')
from torchinfo import summary

from w_model import WModel

w_model = WModel(num_channels=32).cuda()
w_model.load_state_dict(torch.load('../denoising-models/hformer_pytorch/weights/model_278.pth'))
w_model.eval()
print('model summary\n', summary(w_model, input_size=(64, 64, 64, 1)))
```

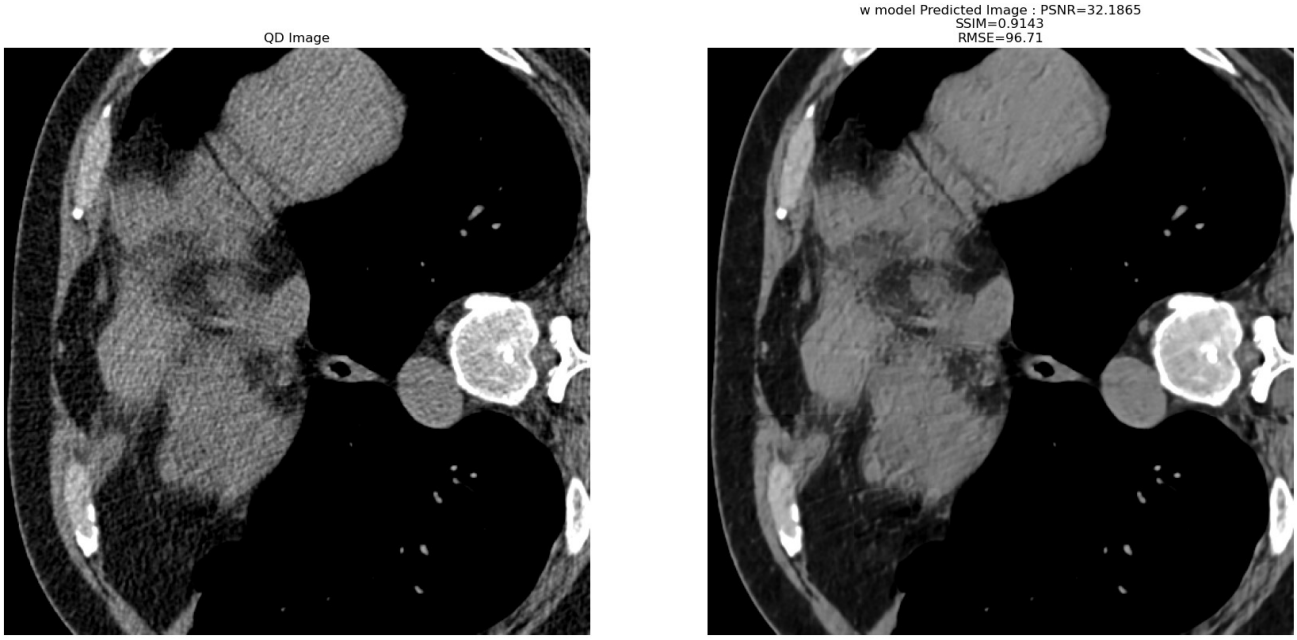
```
Using cache found in C:\Users\Tarun/.cache\torch\hub\mateuszbuda_brain-segmentation-p
Using cache found in C:\Users\Tarun/.cache\torch\hub\mateuszbuda_brain-segmentation-p
model summary
 ==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
WModel                                   [64, 64, 64, 1]           --
├─InputProjectionLayer: 1-1              [64, 32, 64, 64]          --
│    └─Conv2d: 2-1                       [64, 32, 64, 64]          320
├─UNet: 1-2                              [64, 32, 64, 64]          --
│    └─Sequential: 2-2                   [64, 32, 64, 64]          --
│    │    └─Conv2d: 3-1                  [64, 32, 64, 64]          9,216
│    │    └─BatchNorm2d: 3-2             [64, 32, 64, 64]          64
│    │    └─ReLU: 3-3                    [64, 32, 64, 64]          --
│    │    └─Conv2d: 3-4                  [64, 32, 64, 64]          9,216
│    │    └─BatchNorm2d: 3-5             [64, 32, 64, 64]          64
│    │    └─ReLU: 3-6                    [64, 32, 64, 64]          --
│    └─MaxPool2d: 2-3                    [64, 32, 32, 32]          --
│    └─Sequential: 2-4                   [64, 64, 32, 32]          --
│    │    └─Conv2d: 3-7                  [64, 64, 32, 32]          18,432
│    │    └─BatchNorm2d: 3-8             [64, 64, 32, 32]          128
```

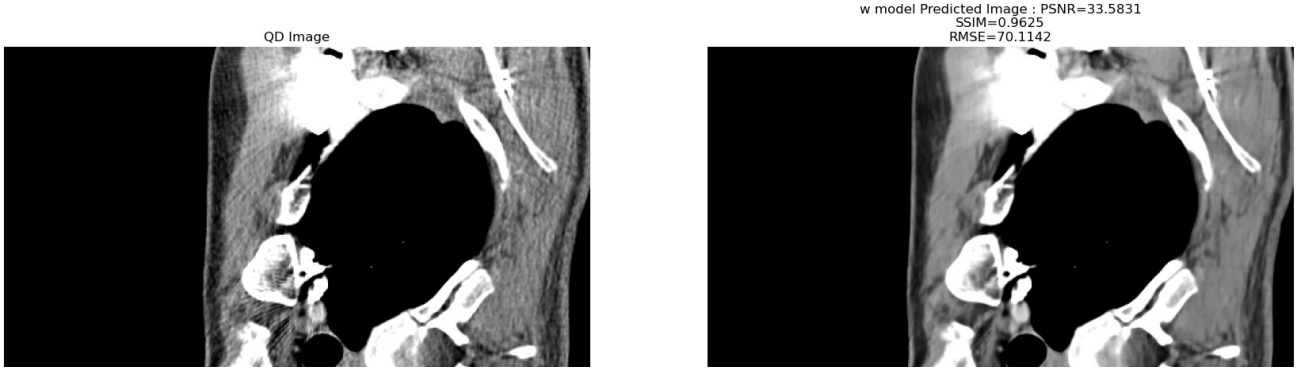
```
|    |    BatchNorm2d: 3-8                   [64, 64, 32, 32]          128
|    |    └─ReLU: 3-9                       [64, 64, 32, 32]          --
|    |    └─Conv2d: 3-10                    [64, 64, 32, 32]          36,864
|    |    └─BatchNorm2d: 3-11               [64, 64, 32, 32]          128
|    |    └─ReLU: 3-12                      [64, 64, 32, 32]          --
|    └─MaxPool2d: 2-5                       [64, 64, 16, 16]          --
|    └─Sequential: 2-6                      [64, 128, 16, 16]         --
|    |    └─Conv2d: 3-13                    [64, 128, 16, 16]         73,728
|    |    └─BatchNorm2d: 3-14               [64, 128, 16, 16]         256
|    |    └─ReLU: 3-15                      [64, 128, 16, 16]         --
|    |    └─Conv2d: 3-16                    [64, 128, 16, 16]         147,456
|    |    └─BatchNorm2d: 3-17               [64, 128, 16, 16]         256
|    |    └─ReLU: 3-18                      [64, 128, 16, 16]         --
|    └─MaxPool2d: 2-7                       [64, 128, 8, 8]           --
|    └─Sequential: 2-8                      [64, 256, 8, 8]           --
|    |    └─Conv2d: 3-19                    [64, 256, 8, 8]           294,912
|    |    └─BatchNorm2d: 3-20               [64, 256, 8, 8]           512
|    |    └─ReLU: 3-21                      [64, 256, 8, 8]           --
|    |    └─Conv2d: 3-22                    [64, 256, 8, 8]           589,824
|    |    └─BatchNorm2d: 3-23               [64, 256, 8, 8]           512
|    |    └─ReLU: 3-24                      [64, 256, 8, 8]           --
|    └─MaxPool2d: 2-9                       [64, 256, 4, 4]           --
|    └─Sequential: 2-10                     [64, 512, 4, 4]           --
|    |    └─Conv2d: 3-25                    [64, 512, 4, 4]           1,179,648
|    |    └─BatchNorm2d: 3-26               [64, 512, 4, 4]           1,024
|    |    └─ReLU: 3-27                      [64, 512, 4, 4]           --
|    |    └─Conv2d: 3-28                    [64, 512, 4, 4]           2,359,296
|    |    └─BatchNorm2d: 3-29               [64, 512, 4, 4]           1,024
|    |    └─ReLU: 3-30                      [64, 512, 4, 4]           --
|    └─ConvTranspose2d: 2-11                [64, 256, 8, 8]           524,544
|    └─Sequential: 2-12                     [64, 256, 8, 8]           --
|    |    └─Conv2d: 3-31                    [64, 256, 8, 8]           1,179,648
|    |    └─BatchNorm2d: 3-32               [64, 256, 8, 8]           512
|    |    └─ReLU: 3-33                      [64, 256, 8, 8]           --
|    |    └─Conv2d: 3-34                    [64, 256, 8, 8]           589,824
|    |    └─BatchNorm2d: 3-35               [64, 256, 8, 8]           512
|    |    └─ReLU: 3-36                      [64, 256, 8, 8]           --
|    └─ConvTranspose2d: 2-13                [64, 128, 16, 16]         131,200
```

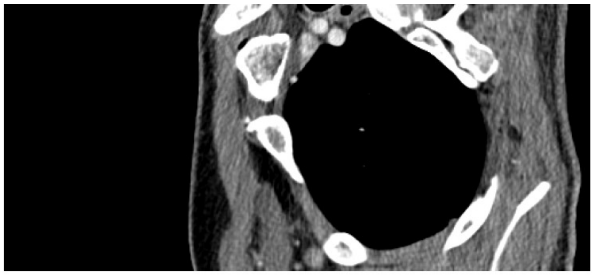
```
w_prediction_patches = []

with torch.no_grad():
    for i, data in enumerate(noisy_image_patches_array):
        noisy = data

        predictions = w_model(torch.unsqueeze(torch.from_numpy(noisy.numpy()), dim=0).to('cuda')).cpu()

        w_prediction_patches.append(predictions.detach().cpu())

w_prediction_patches = np.concatenate(w_prediction_patches, axis=0)

w_predictions = np.expand_dims(reconstruct_image_from_patches(w_prediction_patches[0:64], 8), axis=0)


for i in range(1, int(w_prediction_patches.shape[0] / 64)):
```

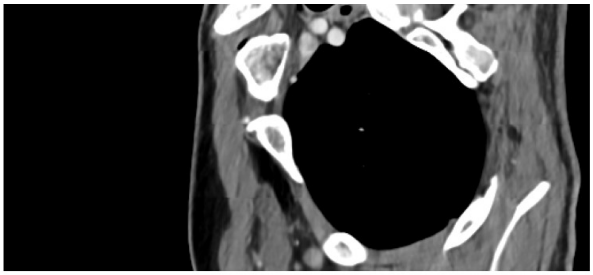
```
    reconstructed_image = reconstruct_image_from_patches(w_prediction_patches[i * 64 : i * 64 + 64], num
    reconstructed_image = np.expand_dims(reconstructed_image, axis=0)

    w_predictions= np.append(w_predictions, reconstructed_image, axis=0)

visualize_predictions(w_model, noisy_array,  len(noisy_array), w_predictions, "w model")
```





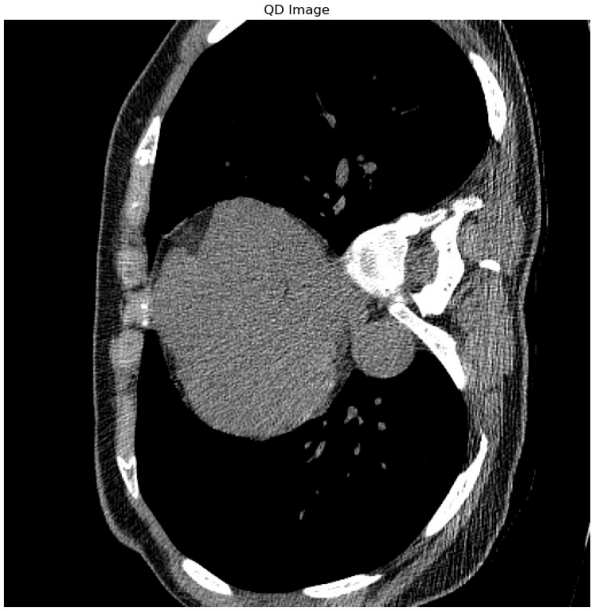






w model Predicted Image : PSNR=25.5771
SSIM=0.8765
RMSE=443.0038

QD Image

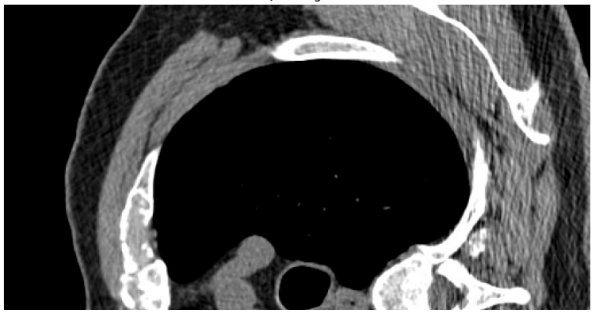


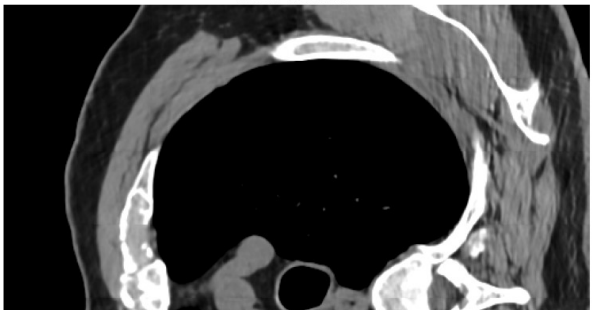
w model Predicted Image : PSNR=37.1836
SSIM=0.9182
RMSE=30.6024

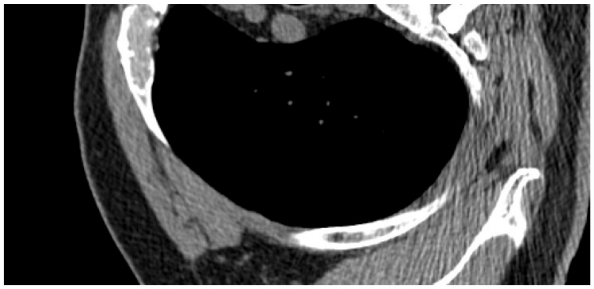
QD Image

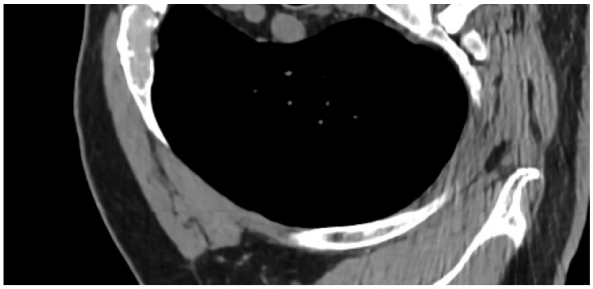
w model Predicted Image : PSNR=31.7072
SSIM=0.9357
RMSE=107.9932

QD Image



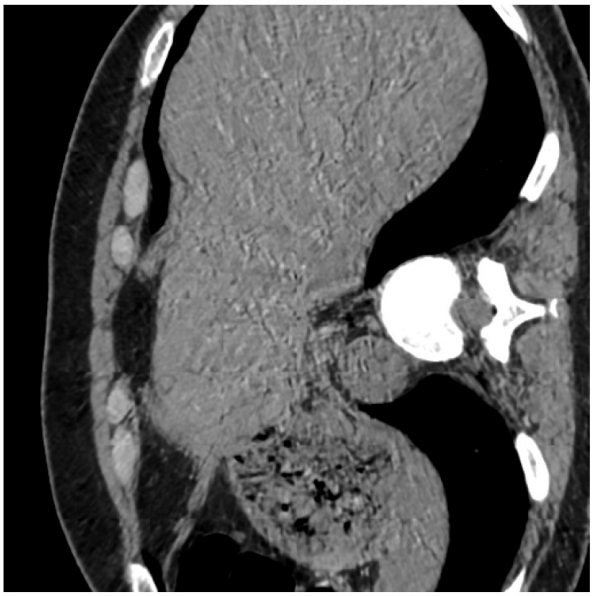






w model Predicted Image : PSNR=27.1027
SSIM=0.8404
RMSE=311.7794

QD Image

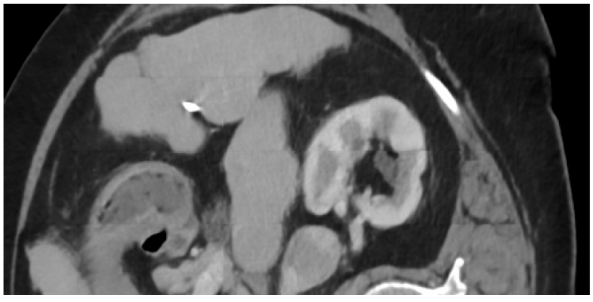


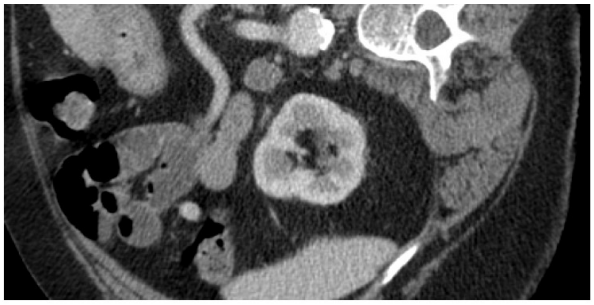
w model Predicted Image : PSNR=34.8294
SSIM=0.9612
RMSE=52.6238

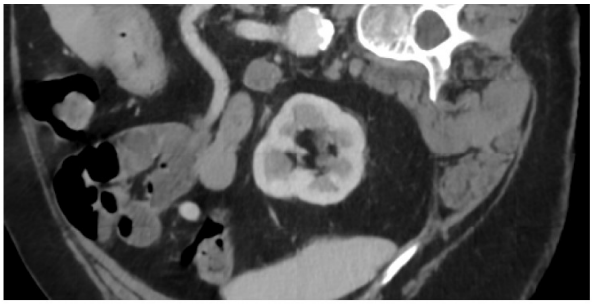
QD Image

w model Predicted Image : PSNR=31.3435
SSIM=0.8579
RMSE=117.4285

QD Image

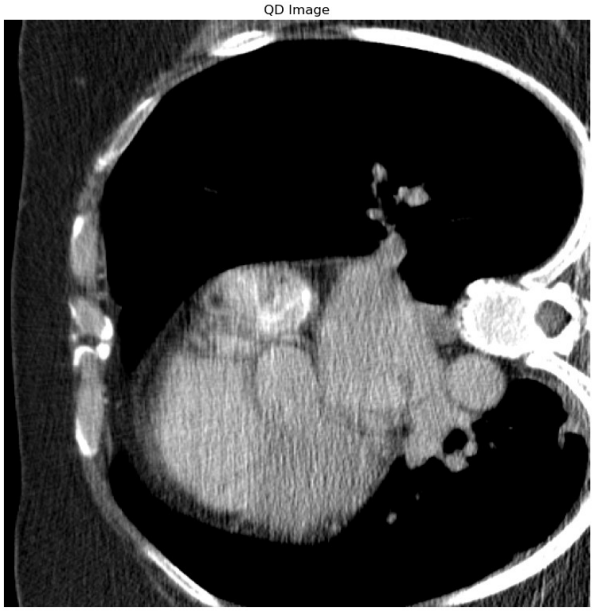


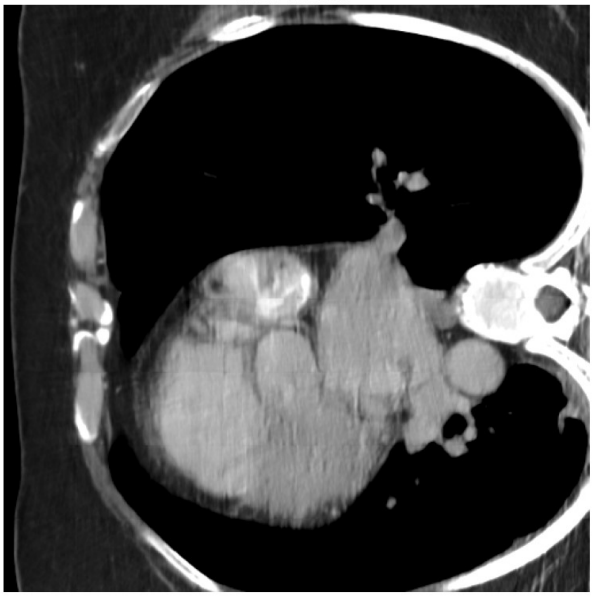




w model Predicted Image : PSNR=30.8824
SSIM=0.8948
RMSE=130.5819

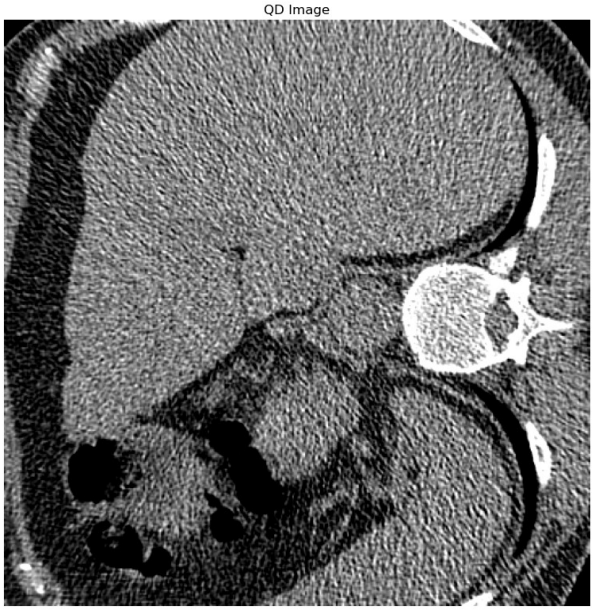
QD Image

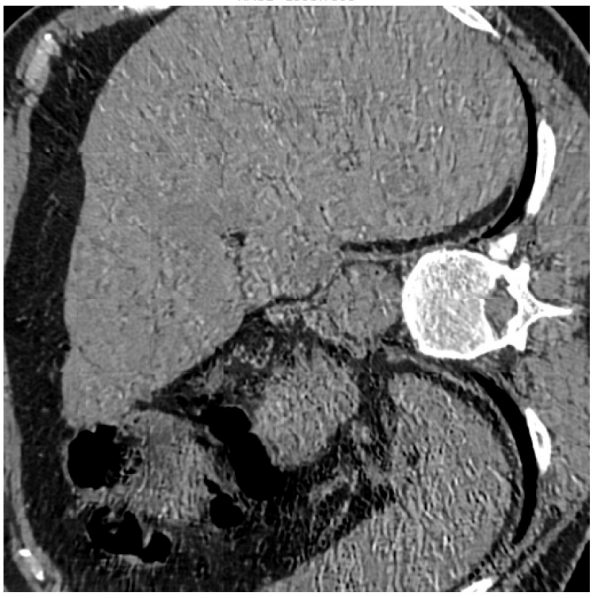




w model Predicted Image : PSNR=20.5994
SSIM=0.7082
RMSE=1393.7395

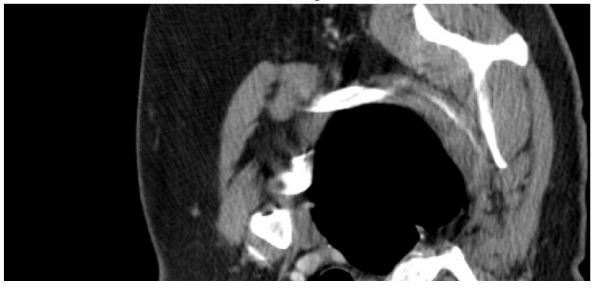
QD Image

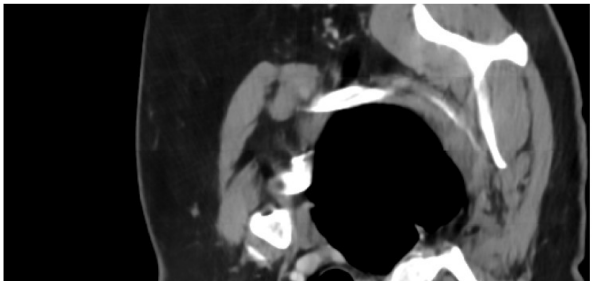


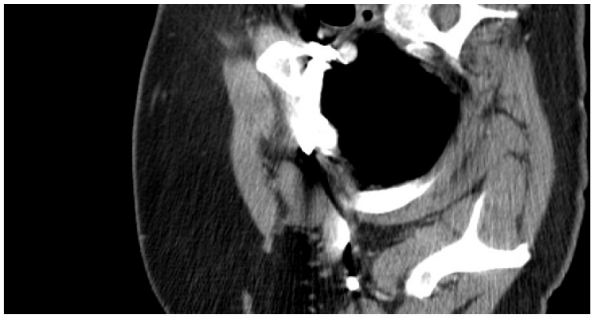


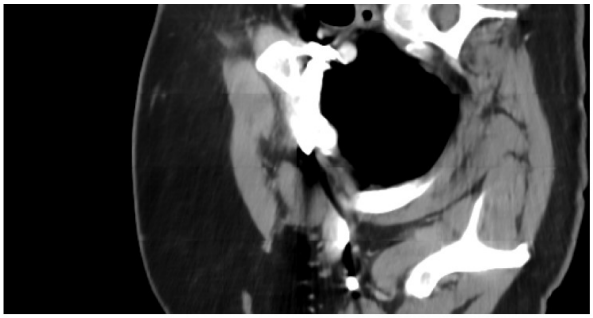
w model Predicted Image : PSNR=34.0396
SSIM=0.9246
RMSE=63.119

QD Image







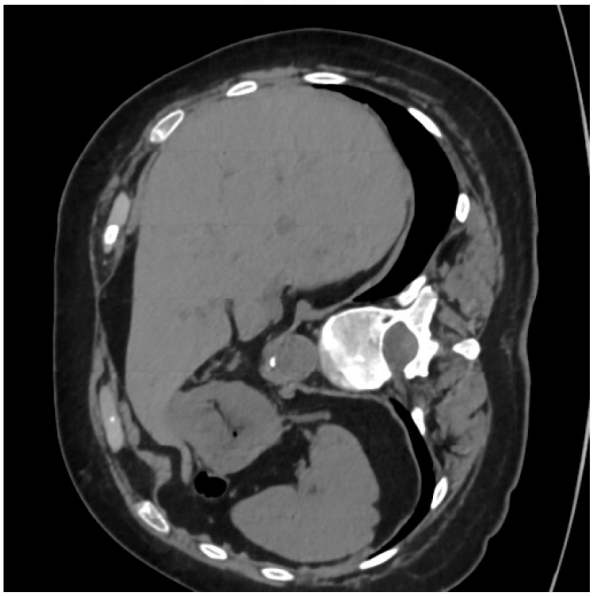


w model Predicted Image : PSNR=33.2459
SSIM=0.903
RMSE=75.7763

QD Image

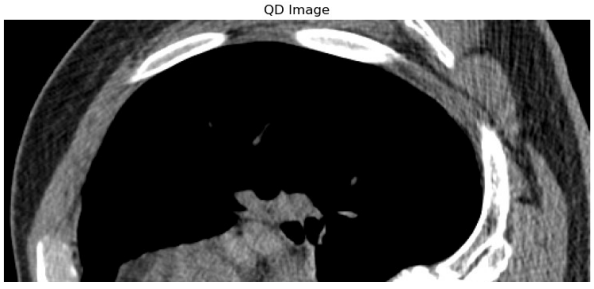
w model Predicted Image : PSNR=33.1083
SSIM=0.8874
RMSE=78.2148

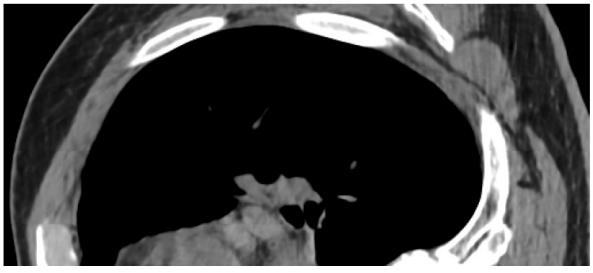
QD Image

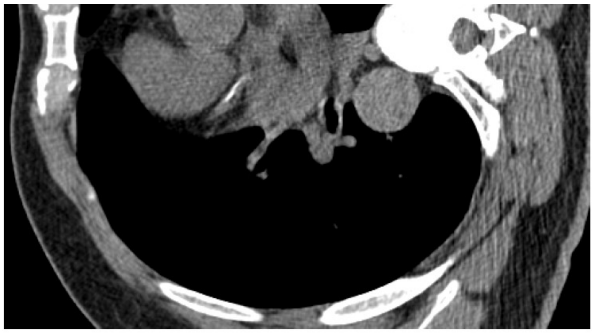


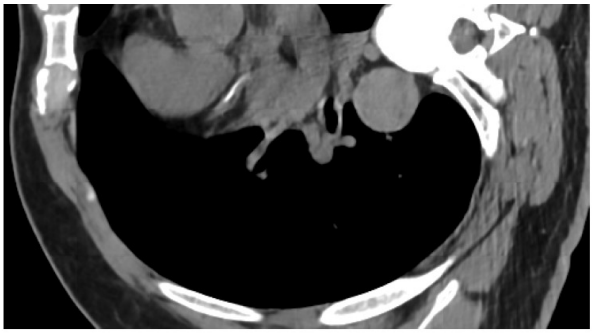
w model Predicted Image : PSNR=33.3518
SSIM=0.9434
RMSE=73.9496

QD Image



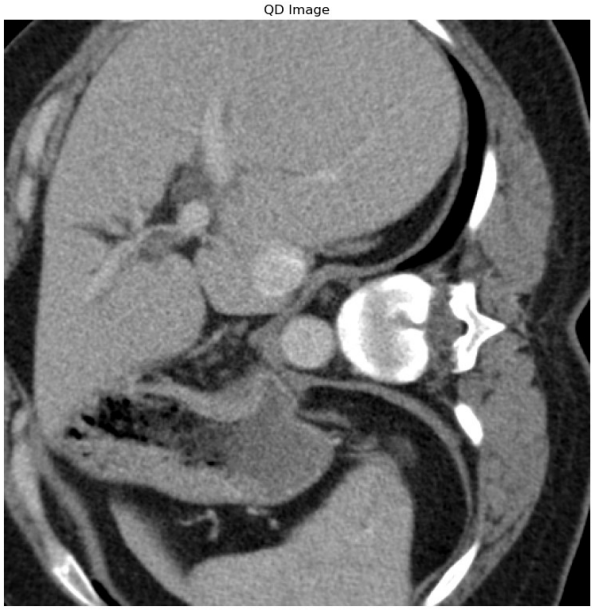


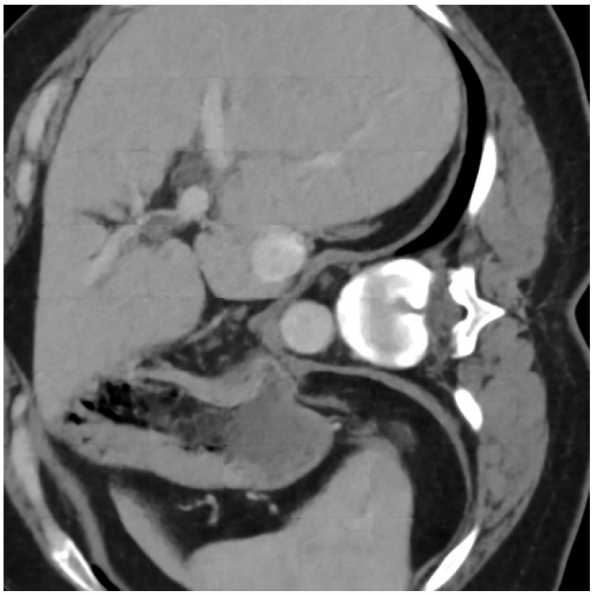




w model Predicted Image : PSNR=33.3203
SSIM=0.8582
RMSE=74.4886

QD Image



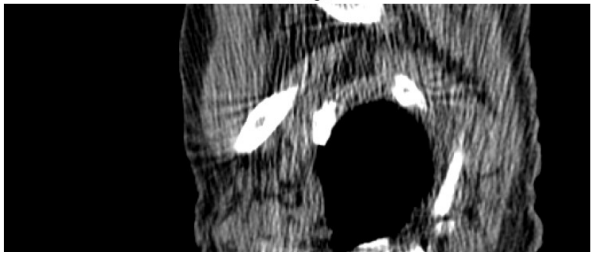


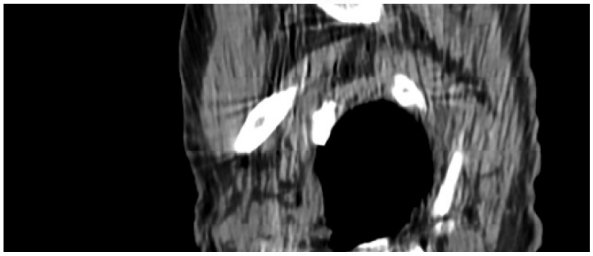
w model Predicted Image : PSNR=33.0199
SSIM=0.9401
RMSE=79.8241

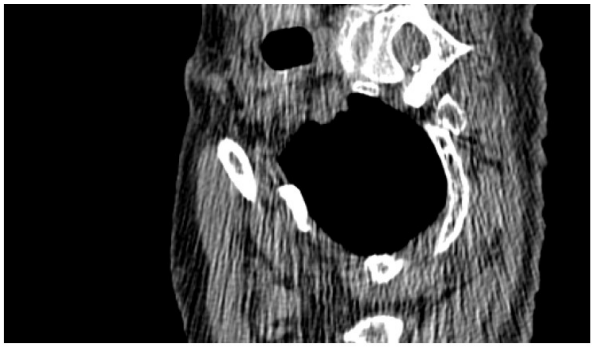
QD Image

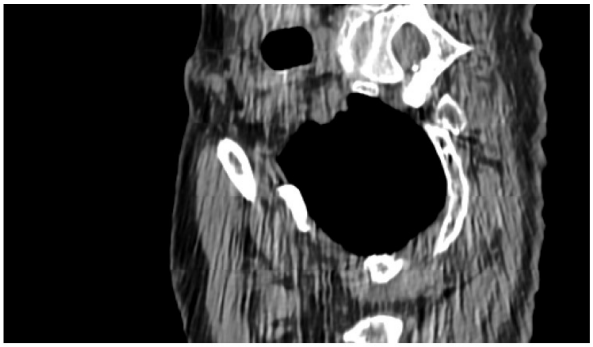
w model Predicted Image : PSNR=29.7835
SSIM=0.9513
RMSE=168.1784

QD Image



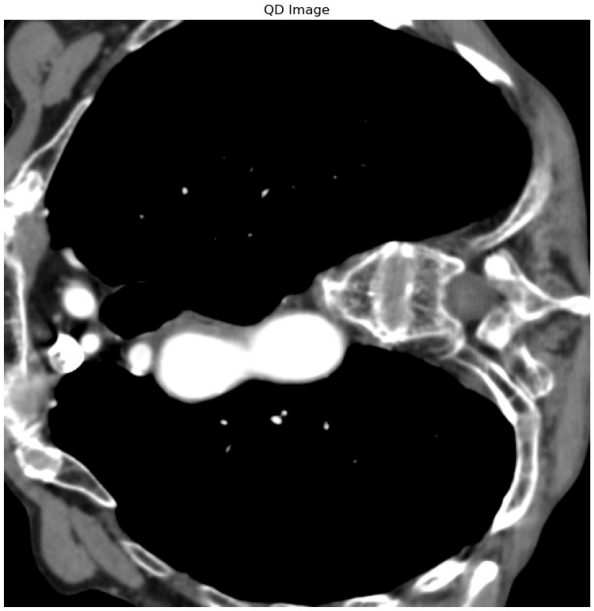


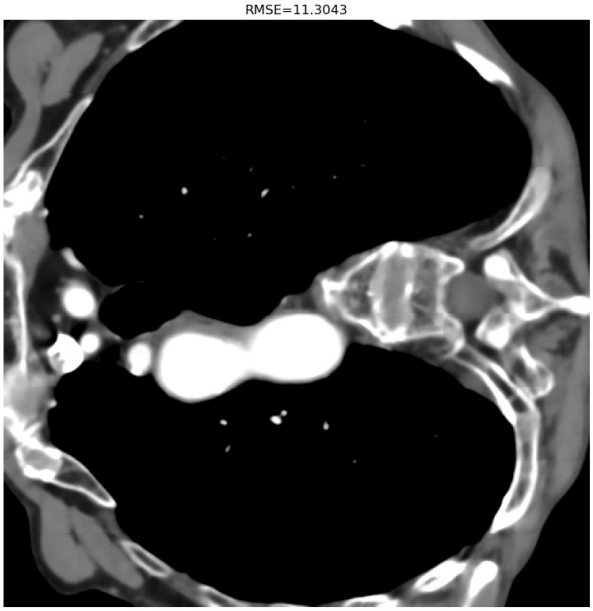




w model Predicted Image : PSNR=41.5088
SSIM=0.9928
RMSE=11.3043

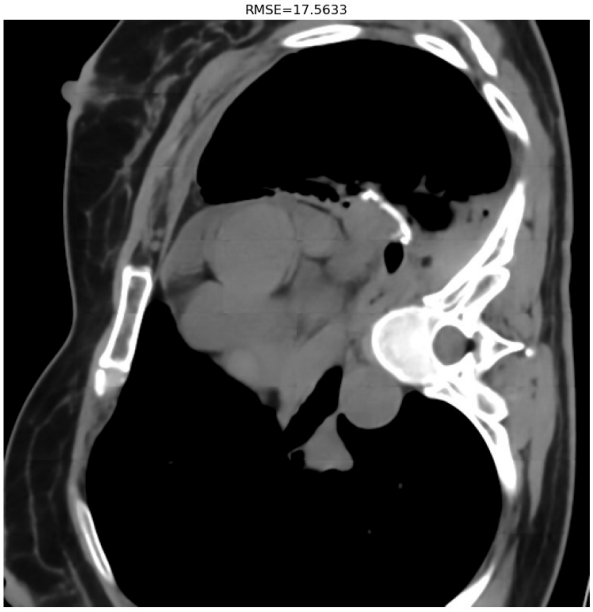
QD Image





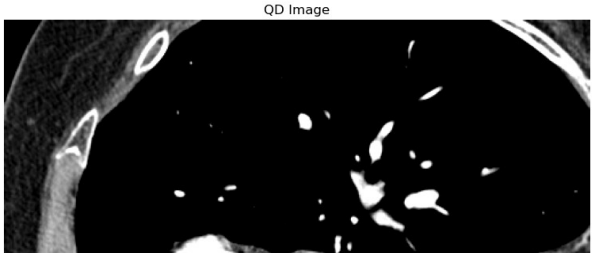
w model Predicted Image : PSNR=39.5951
SSIM=0.975
RMSE=17.5633

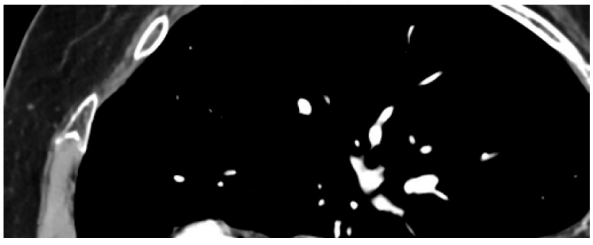
QD Image

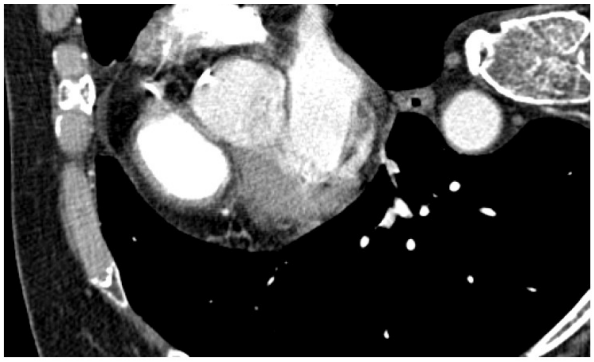


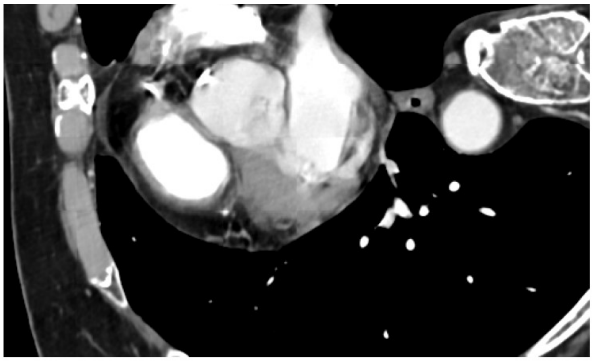
w model Predicted Image : PSNR=34.0714
SSIM=0.954
RMSE=62.6582

QD Image



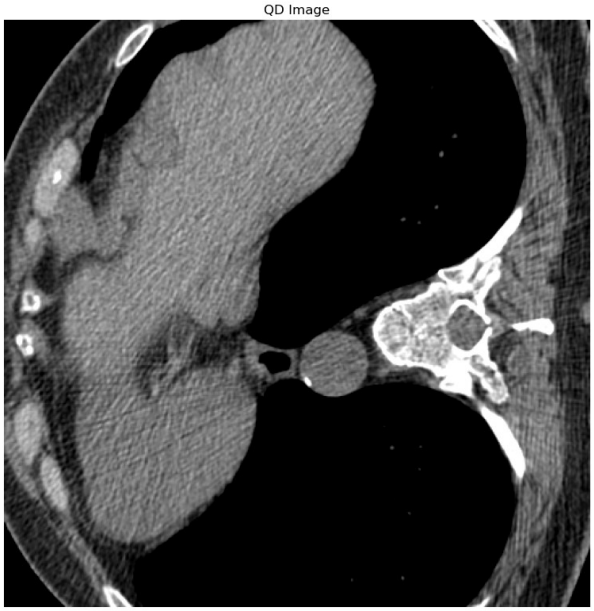






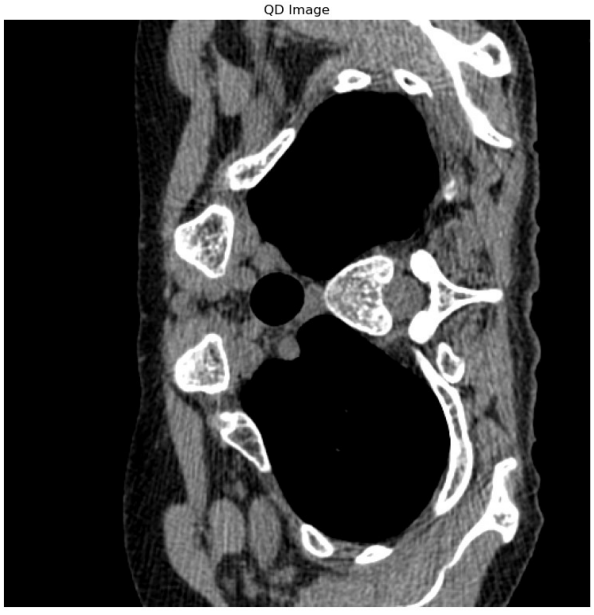
w model Predicted Image : PSNR=31.4061
SSIM=0.8819
RMSE=115.7475

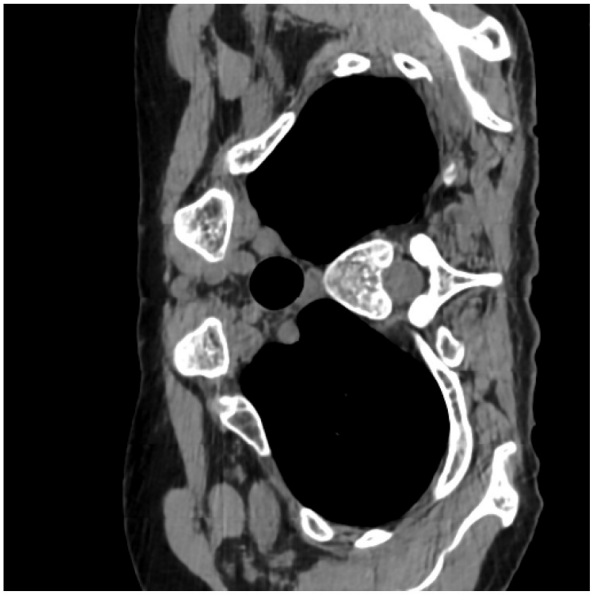
QD Image



w model Predicted Image : PSNR=33.9317
SSIM=0.9555
RMSE=64.7074

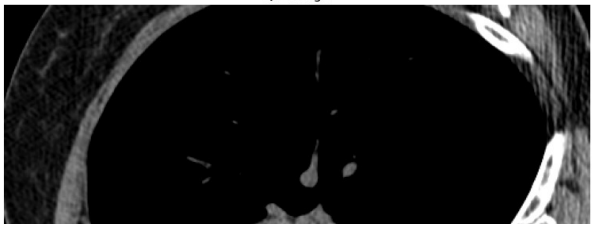
QD Image

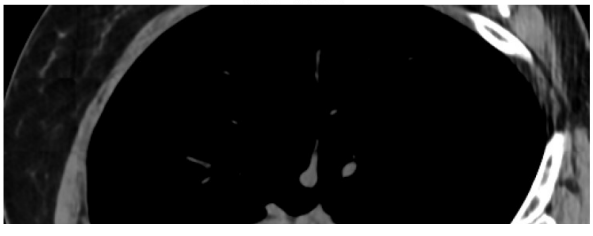


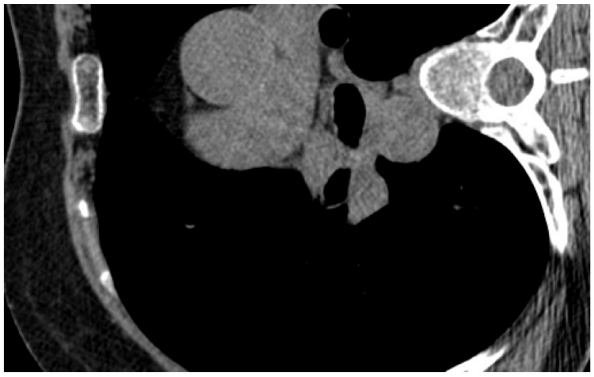


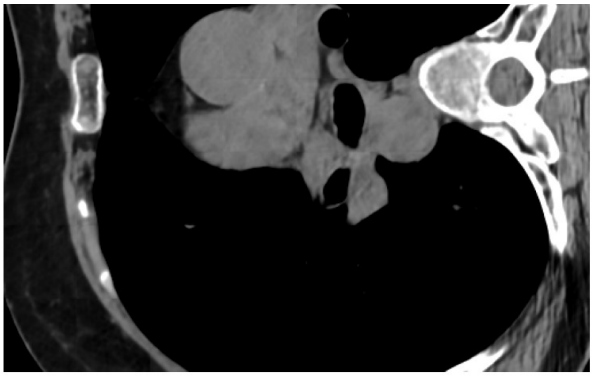
w model Predicted Image : PSNR=35.169
SSIM=0.9544
RMSE=48.6657

QD Image



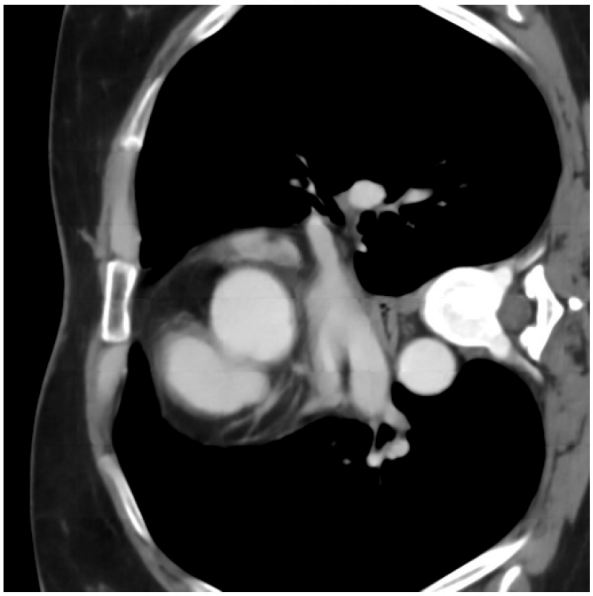






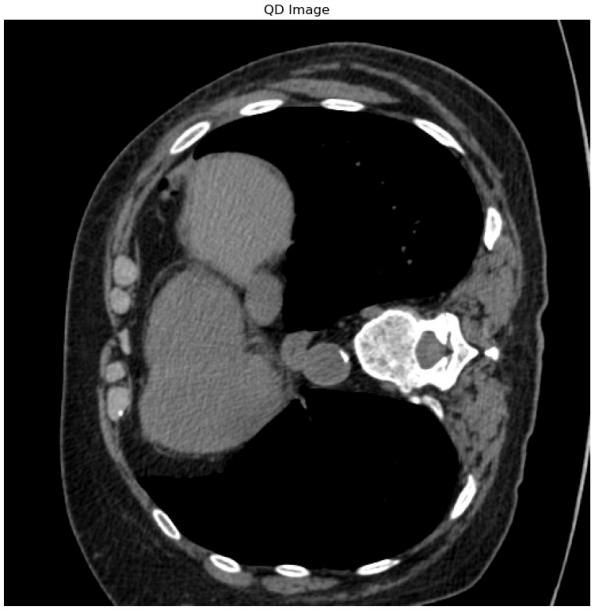
w model Predicted Image : PSNR=39.8881
SSIM=0.9756
RMSE=16.4175

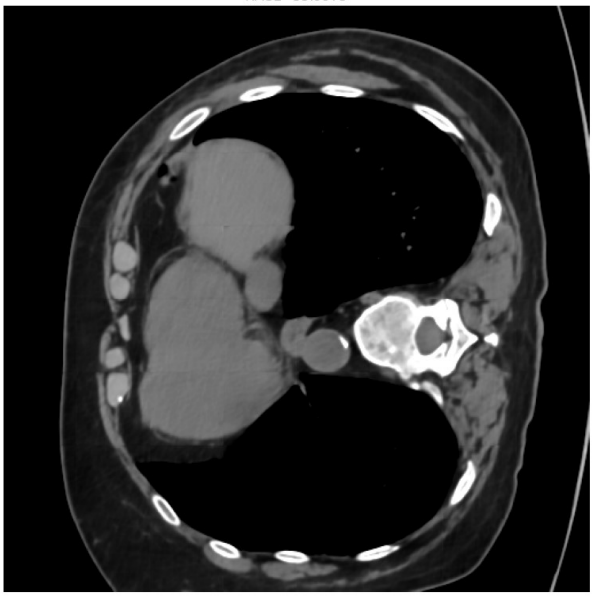
QD Image



w model Predicted Image : PSNR=36.4785
SSIM=0.9499
RMSE=35.9975

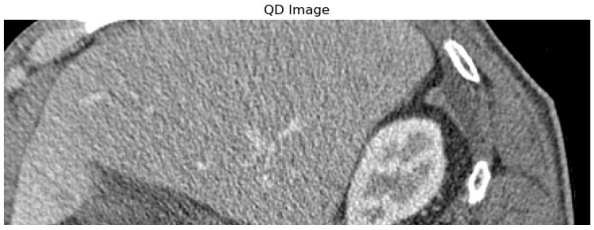
QD Image

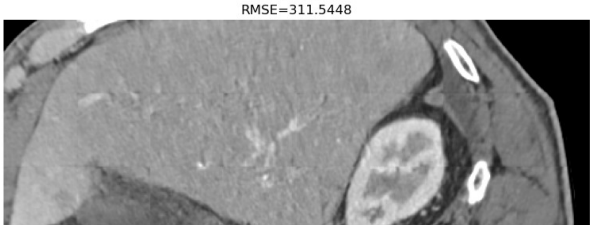


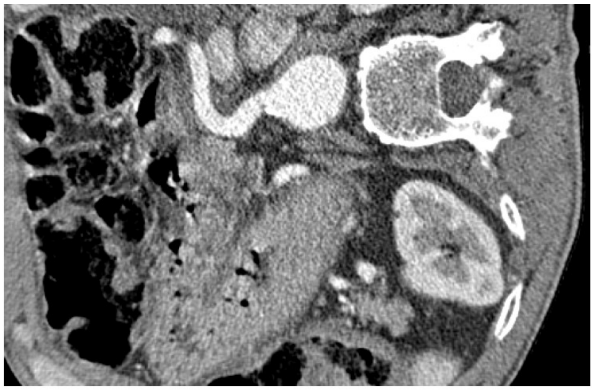


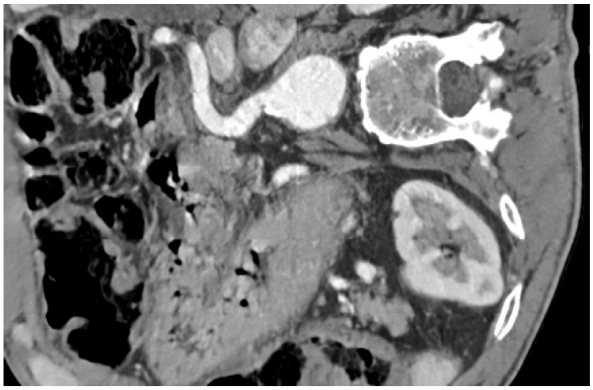
w model Predicted Image : PSNR=27.106
SSIM=0.8185
RMSE=311.5448

QD Image









QD Image

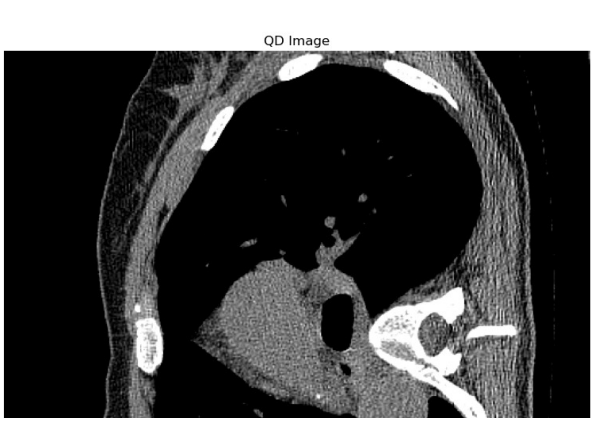
w model Predicted Image : PSNR=28.7437
SSIM=0.8207
RMSE=213.6733

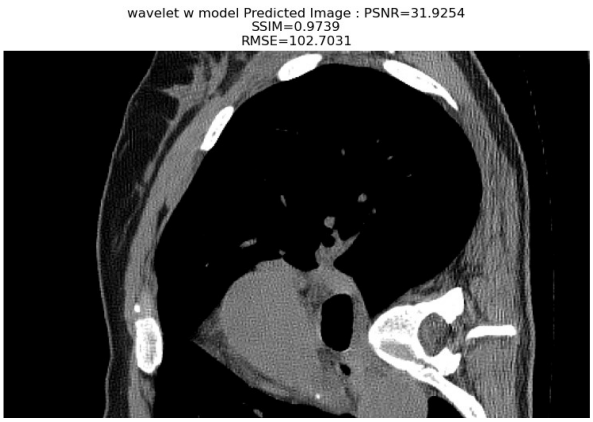
```
# Wavelet with wmodel

# W model with wavelet
from pytorch_wavelets import DTCWTForward, DTCWTInverse
dwt = DTCWTForward(J=3).cuda()
idwt = DTCWTInverse().cuda()
wavelet_w_model_prediction_patches =[]


with torch.no_grad():
    for i in range(noisy_image_patches_array.shape[0] // 64):
        noisy = noisy_image_patches_array[i * 64 : i * 64 + 64]
        noisy = torch.from_numpy(noisy.cpu().numpy()).to('cuda')
        prediction = w_model(noisy)

        prediction_img = torch.transpose(prediction, 1, 3)
        transposed_noisy_image = torch.transpose(noisy, 1, 3)
```

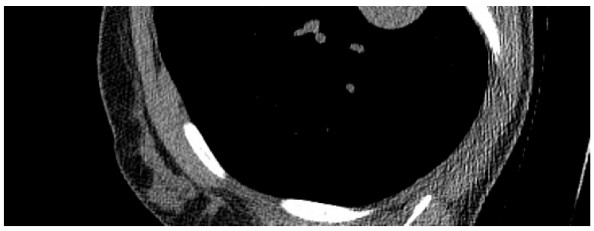
```
        transposed_noisy_image = torch.transpose(noisy, 1, 3)

        prediction_approx, prediction_high_freq = dwt(prediction_img.cuda())
        prediction_high_freq_low, prediction_high_freq_mid, prediction_high_freq_coarse = prediction_hig

        noisy_approx, noisy_high_freq = dwt(transposed_noisy_image.cuda())

        noisy_high_freq_fine, noisy_high_freq_mid, noisy_high_freq_coarse = noisy_high_freq[0], noisy_hi

        reconstructed_prediction_image_with_high_freq_swap = idwt((noisy_approx, noisy_high_freq))
        reconstructed_prediction_image_with_high_freq_swap = torch.transpose(reconstructed_prediction_im


        #denoise_high_freq = torch.transpose(torch.from_numpy(denoised_high_freq), 1, 3)
        #denoised_noisy_approx, denoised_high_freq = dwt(denoise_high_freq.cuda())
        #noisy_high_freq_fine, noisy_high_freq_mid, noisy_high_freq_coarse = denoised_high_freq[0], denc

        wavelet_high_freq_swapped = [None] * 3
        wavelet_high_freq_swapped[0] =noisy_high_freq_fine
        wavelet_high_freq_swapped[1] =prediction_high_freq_mid
        wavelet_high_freq_swapped[2] =prediction_high_freq_coarse

        reconstructed_prediction_image = idwt((prediction_approx,wavelet_high_freq_swapped))
        reconstructed_prediction_image = torch.transpose(reconstructed_prediction_image, 1, 3)
        wavelet_w_model_prediction_patches.append(reconstructed_prediction_image.detach().cpu().numpy())

wavelet_w_predictions = [None] * noisy_array.shape[0]
def reconstruct(patches,  num_images):
    num_patches_per_image = patches.shape[0] // num_images
    for i in range(num_images):

        image_patches = patches[i * num_patches_per_image:i * num_patches_per_image + num_patches_per_im

        reconstruct_image = (reconstruct_image_from_patches(image_patches, 8))
        wavelet_w_predictions[i]  = reconstruct_image
wavelet_w_model_prediction_patches = np.concatenate(wavelet_w_model_prediction_patches)
reconstruct(wavelet_w_model_prediction_patches, 28)
```

```
WARNING:tensorflow:From C:\Users\Tarun\AppData\Local\Temp\ipykernel_11168\1428046907.
Instructions for updating:
Use tf.identity instead.
```

```
visualize_predictions(w_model, noisy_array,  len(noisy_array), wavelet_w_predictions, "wavelet w model")
```

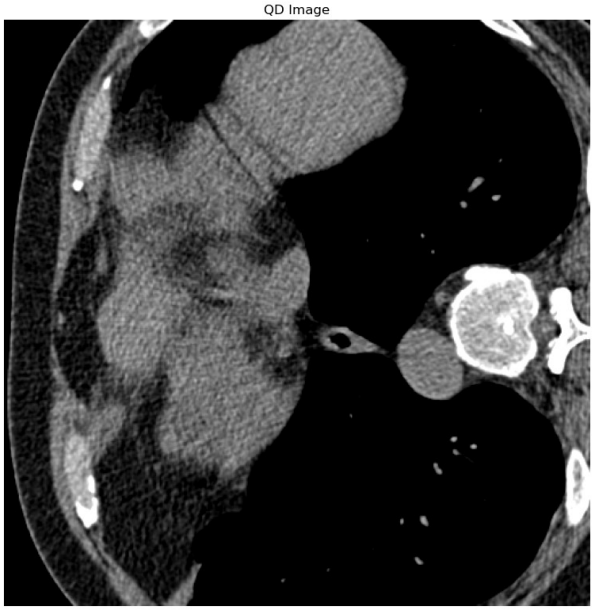






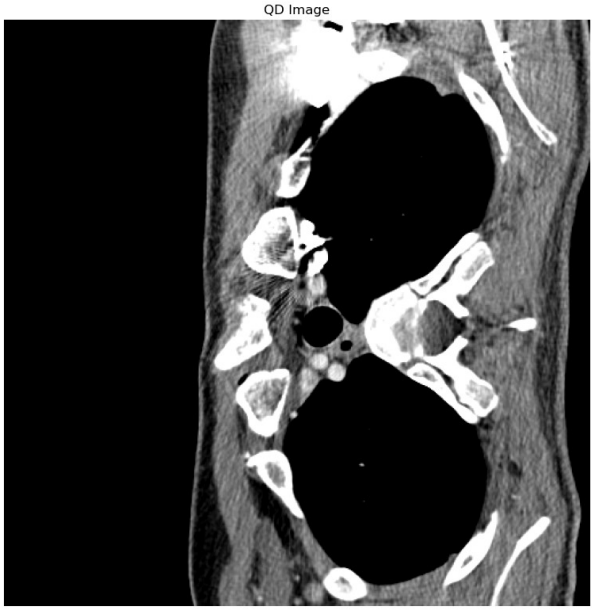
wavelet w model Predicted Image : PSNR=33.8063
SSIM=0.9421
RMSE=66.6025

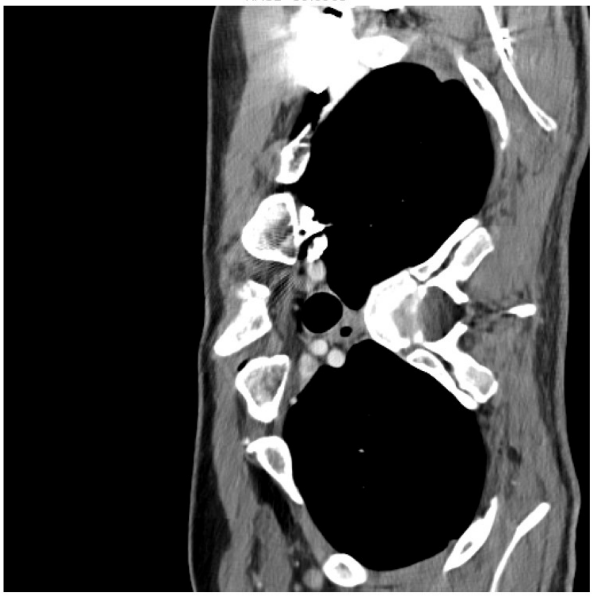
QD Image



wavelet w model Predicted Image : PSNR=36.0317
SSIM=0.9785
RMSE=39.8983

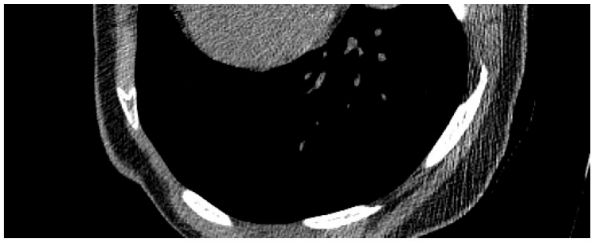
QD Image





wavelet w model Predicted Image : PSNR=30.9012
SSIM=0.967
RMSE=130.0179

QD Image



wavelet w model Predicted Image : PSNR=37.9865
SSIM=0.9336
RMSE=25.4373

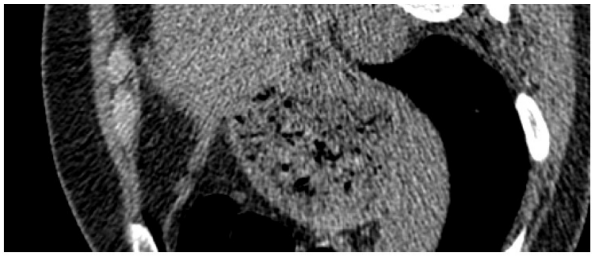
QD Image

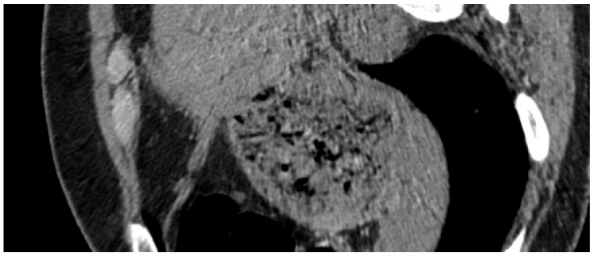
wavelet w model Predicted Image : PSNR=34.1222
SSIM=0.9626
RMSE=61.9297

QD Image

wavelet w model Predicted Image : PSNR=29.0101
SSIM=0.9011
RMSE=200.9599

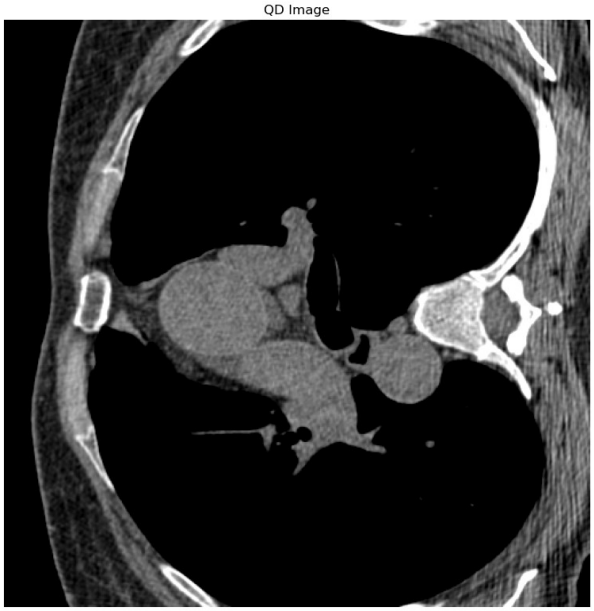
QD Image





wavelet w model Predicted Image : PSNR=36.9632
SSIM=0.9753
RMSE=32.1958

QD Image

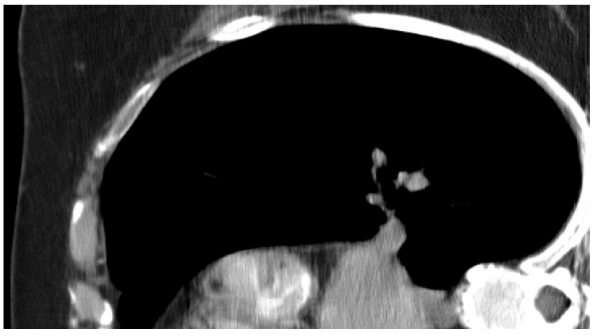


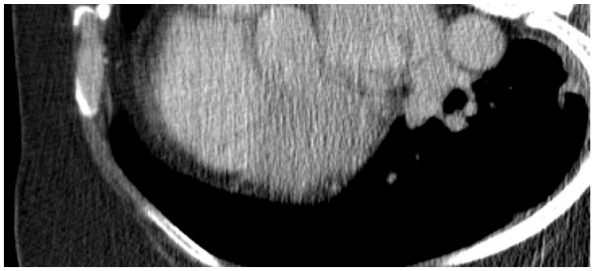
wavelet w model Predicted Image : PSNR=33.5503
SSIM=0.918
RMSE=70.647

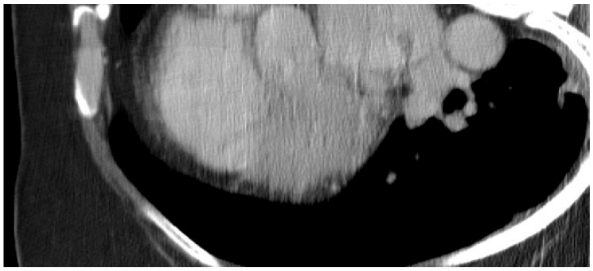
QD Image

wavelet w model Predicted Image : PSNR=33.8728
SSIM=0.9505
RMSE=65.5907

QD Image

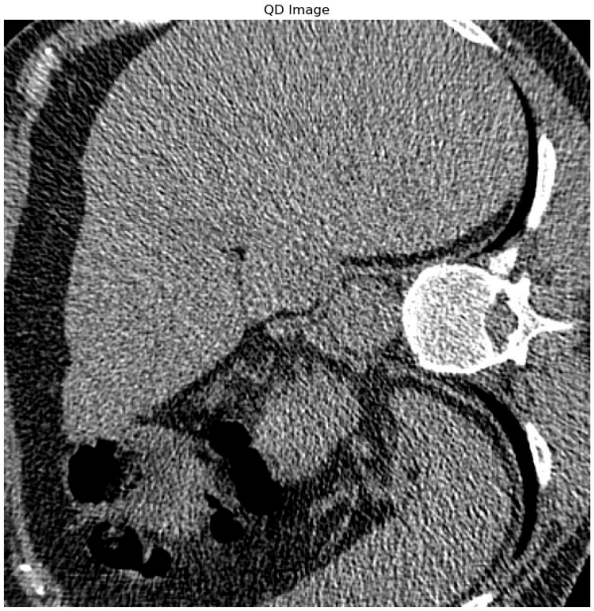


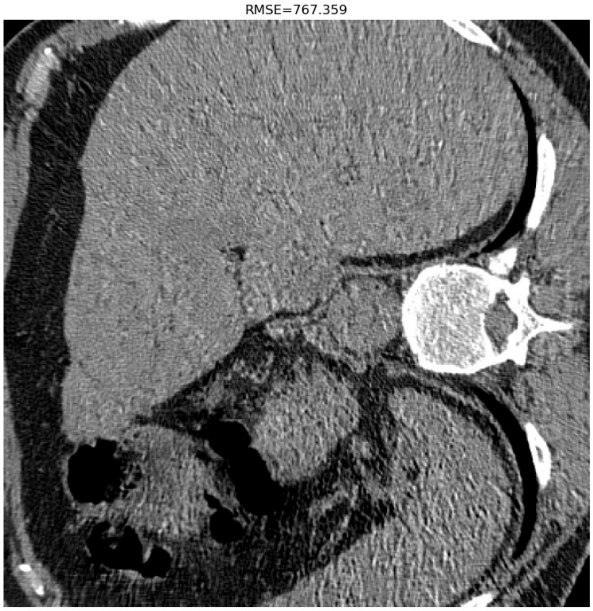




wavelet w model Predicted Image : PSNR=23.1912
SSIM=0.8522
RMSE=767.359

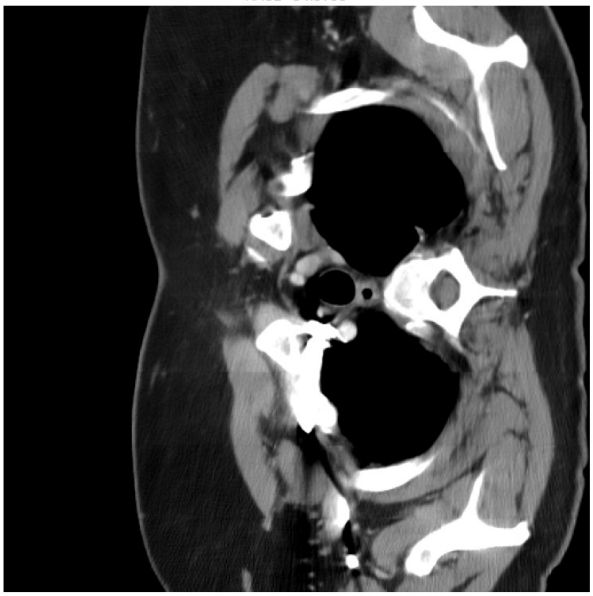
QD Image





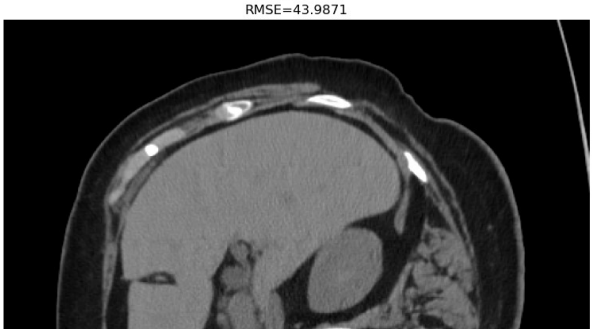
wavelet w model Predicted Image : PSNR=36.6535
SSIM=0.9619
RMSE=34.5753

QD Image



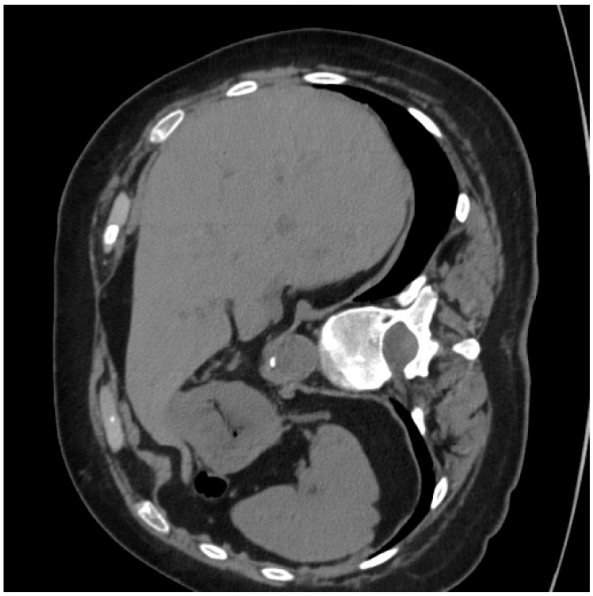
wavelet w model Predicted Image : PSNR=35.6079
SSIM=0.9456
RMSE=43.9871

QD Image



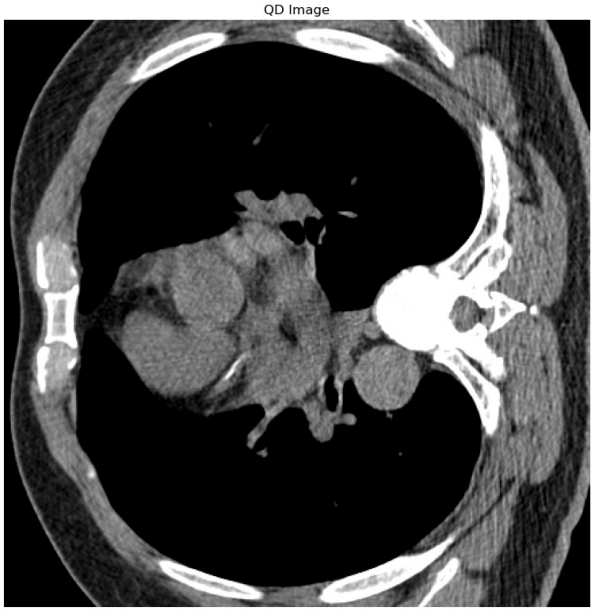
wavelet w model Predicted Image : PSNR=35.3456
SSIM=0.9355
RMSE=46.7263

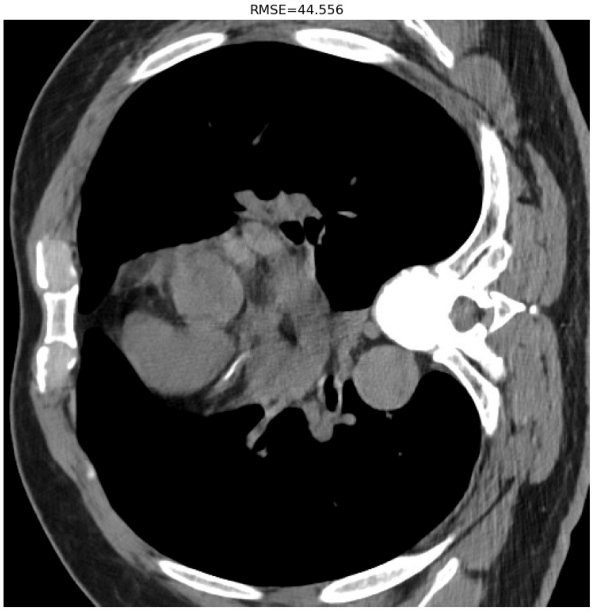
QD Image



wavelet w model Predicted Image : PSNR=35.5521
SSIM=0.966
RMSE=44.556

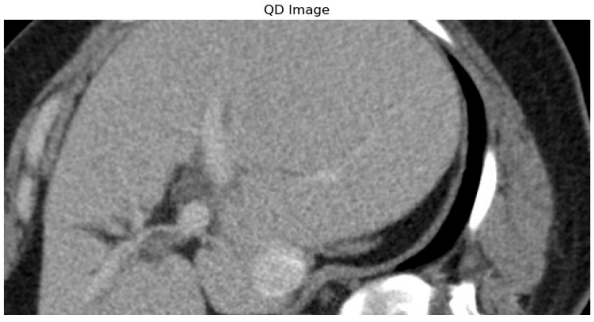
QD Image

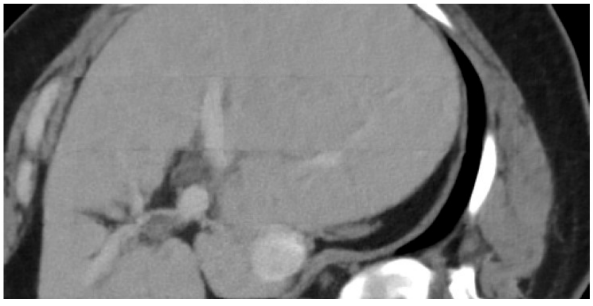


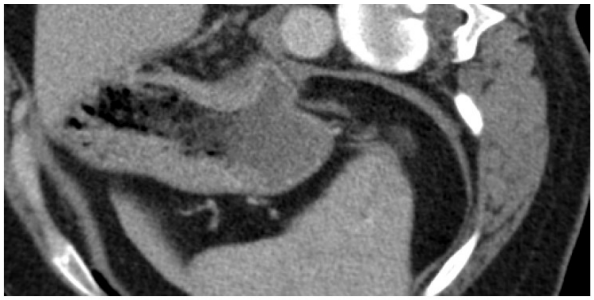


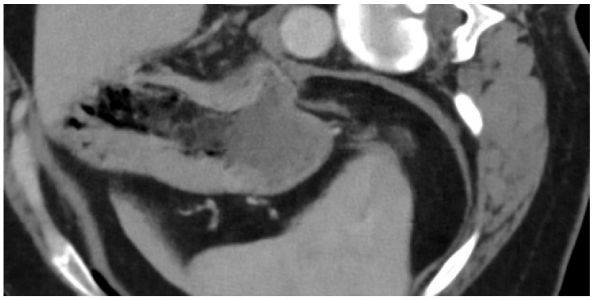
wavelet w model Predicted Image : PSNR=34.6586
SSIM=0.8981
RMSE=54.7348

QD Image







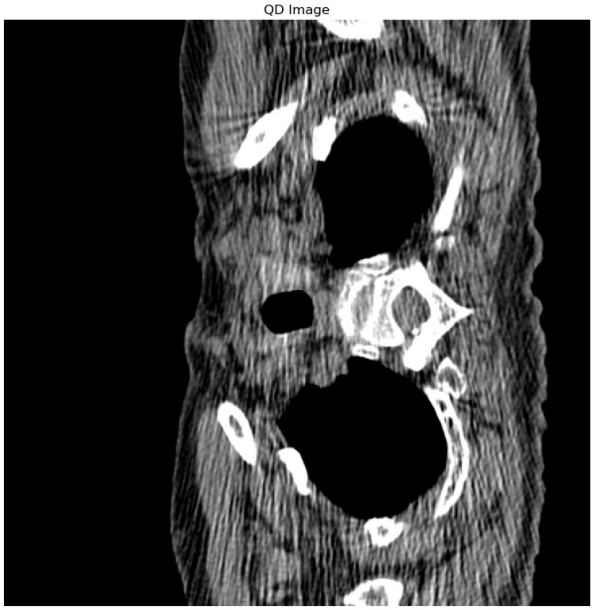


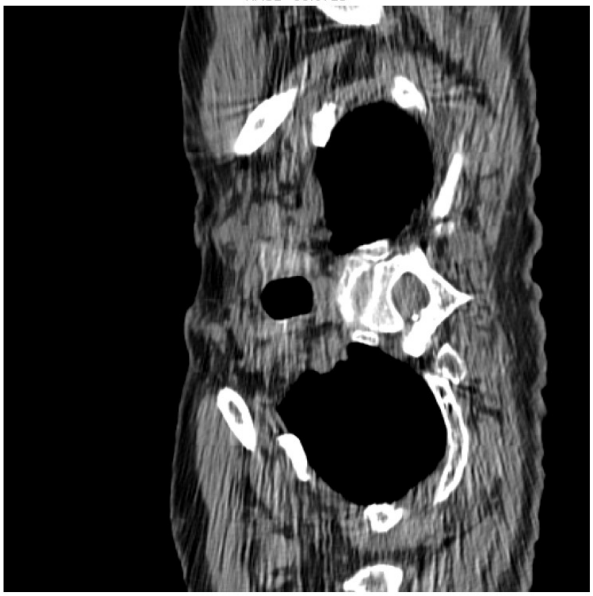
wavelet w model Predicted Image : PSNR=35.1322
SSIM=0.9639
RMSE=49.0799

QD Image

wavelet w model Predicted Image : PSNR=32.5633
SSIM=0.9738
RMSE=88.6723

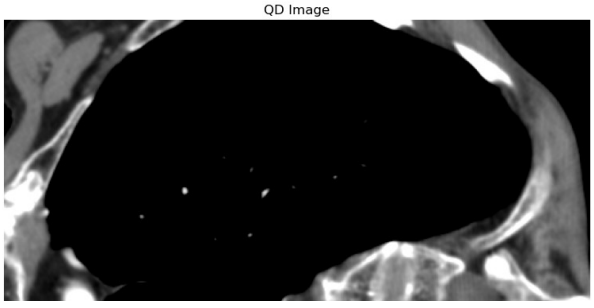
QD Image

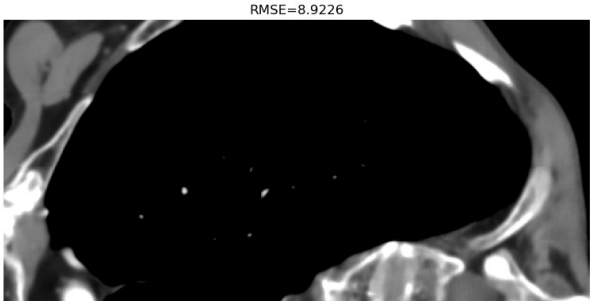


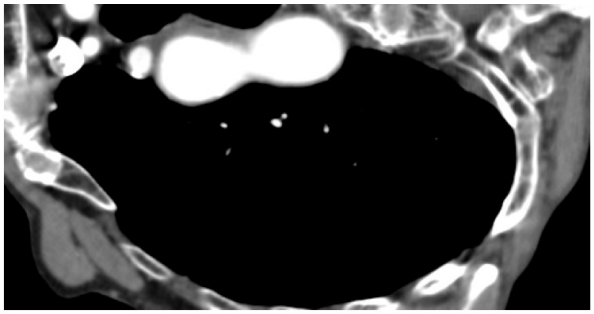


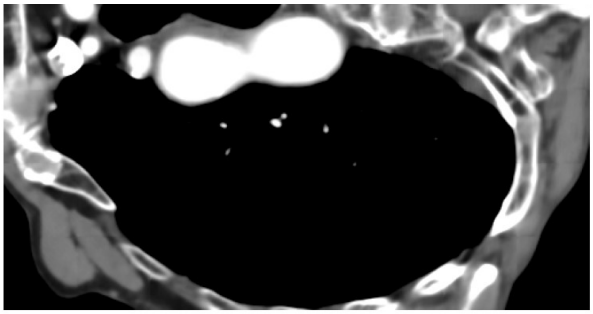
wavelet w model Predicted Image : PSNR=42.5363
SSIM=0.9939
RMSE=8.9226

QD Image



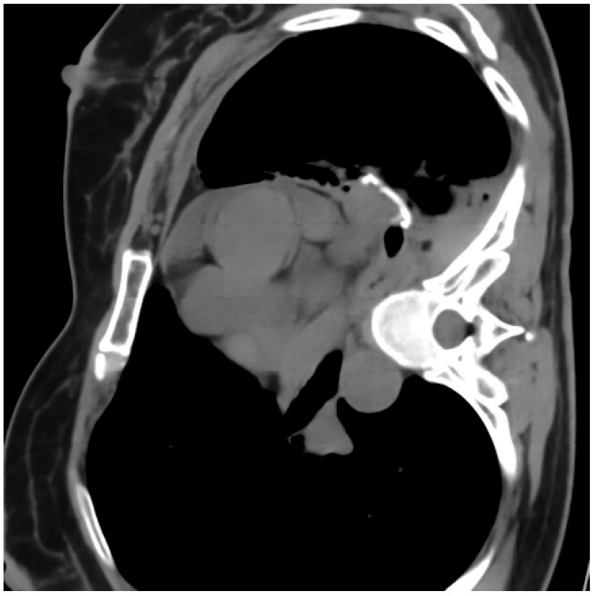






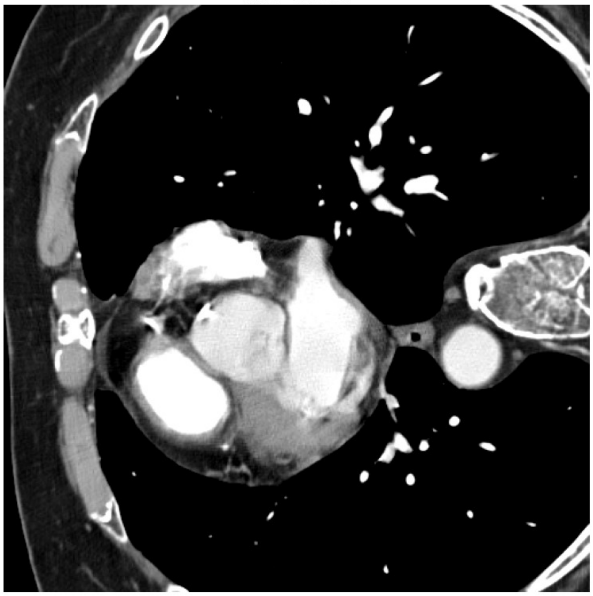
wavelet w model Predicted Image : PSNR=40.7613
SSIM=0.9798
RMSE=13.4275

QD Image



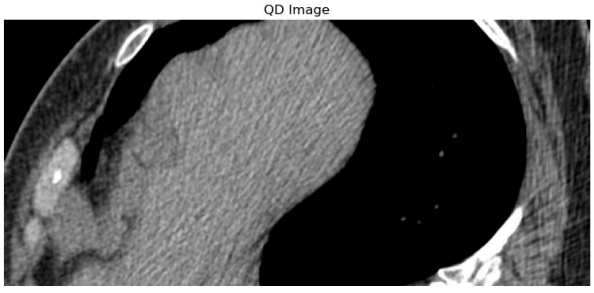
wavelet w model Predicted Image : PSNR=36.2547
SSIM=0.9727
RMSE=37.9012

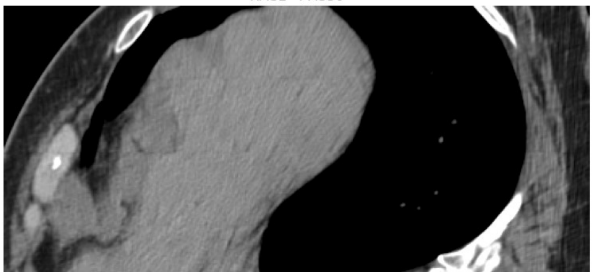
QD Image

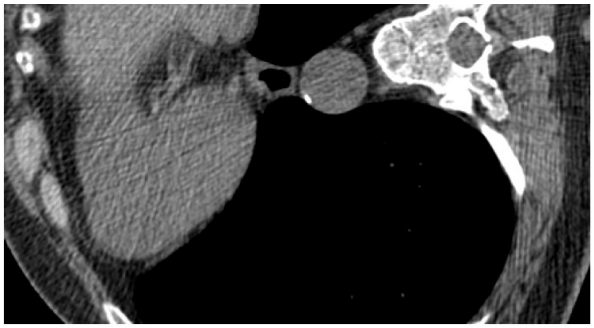


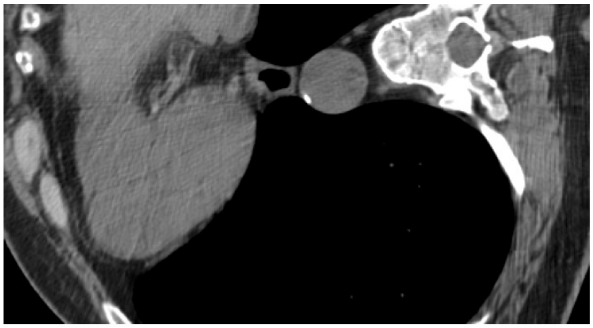
wavelet w model Predicted Image : PSNR=33.1574
SSIM=0.9238
RMSE=77.336

QD Image



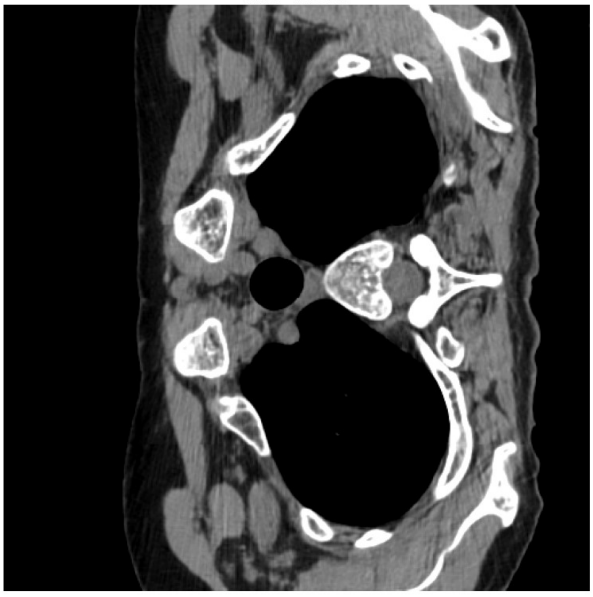






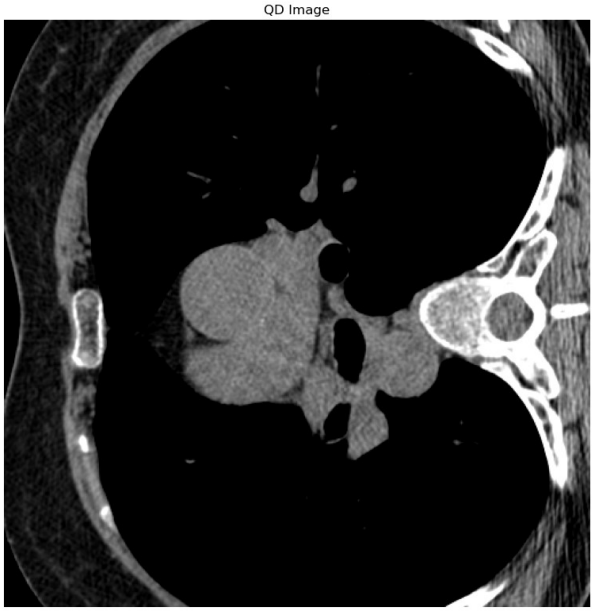
wavelet w model Predicted Image : PSNR=35.7041
SSIM=0.9703
RMSE=43.0239

QD Image



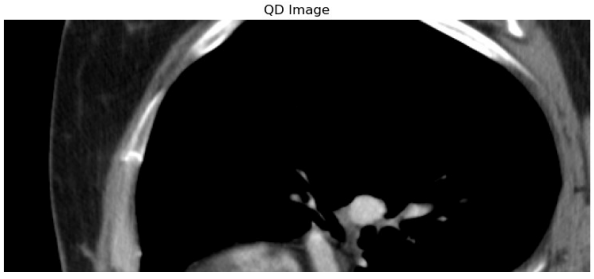
wavelet w model Predicted Image : PSNR=37.036
SSIM=0.9688
RMSE=31.6606

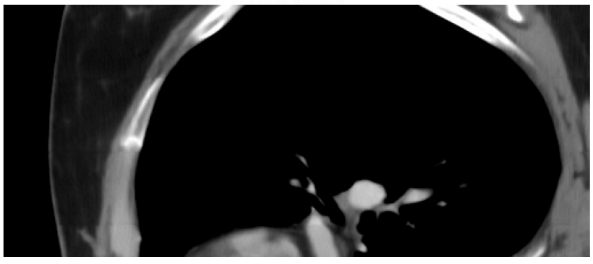
QD Image

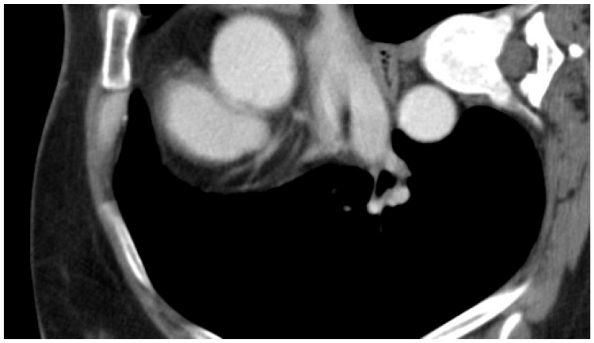


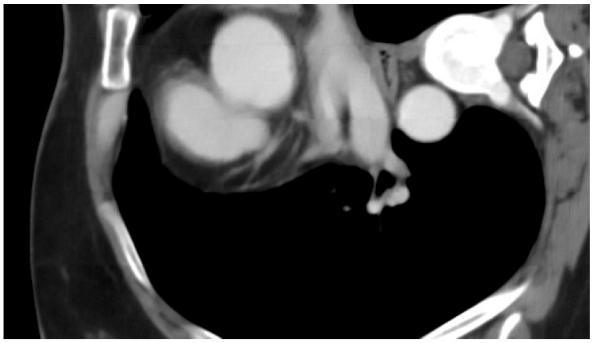
wavelet w model Predicted Image : PSNR=41.3679
SSIM=0.9833
RMSE=11.677

QD Image



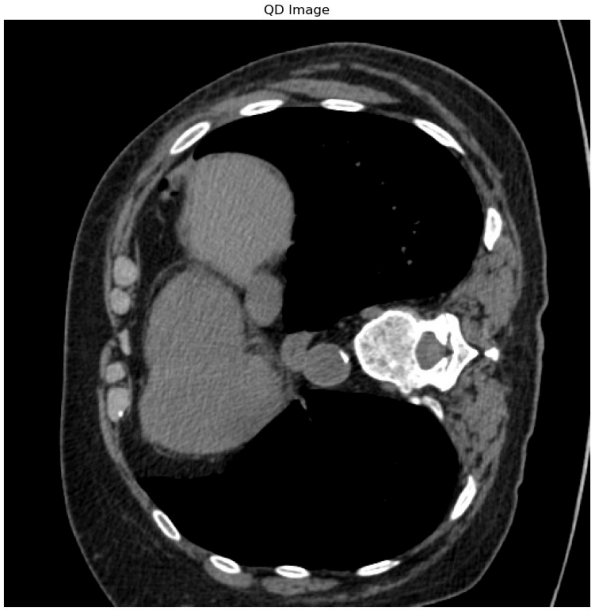


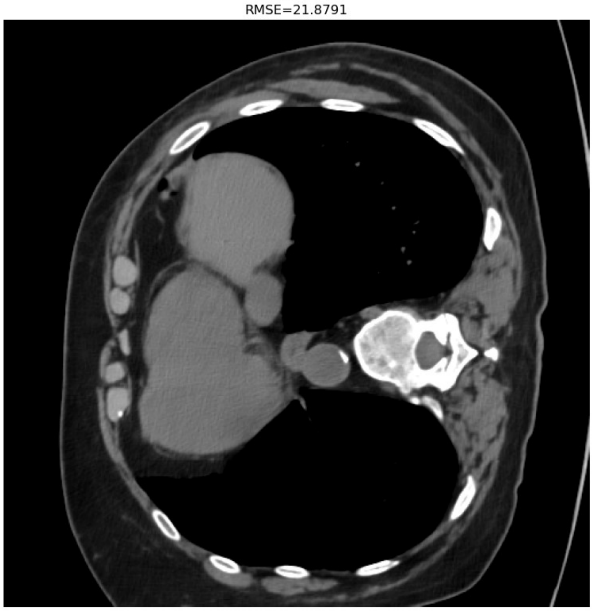




wavelet w model Predicted Image : PSNR=38.6409
SSIM=0.9702
RMSE=21.8791

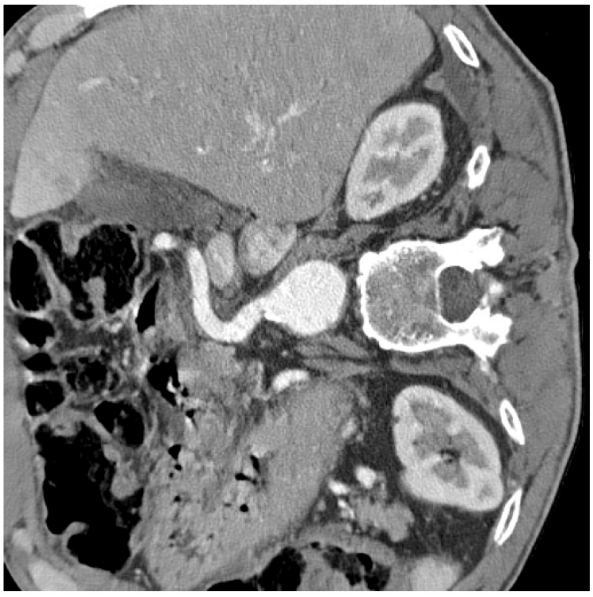
QD Image





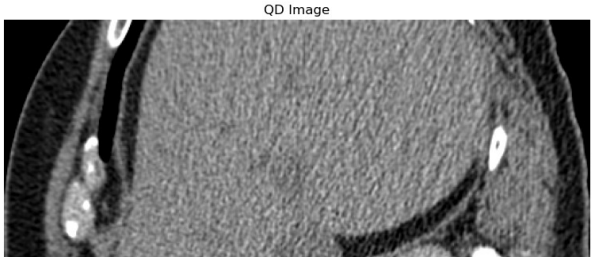
wavelet w model Predicted Image : PSNR=29.6992
SSIM=0.9051
RMSE=171.4736

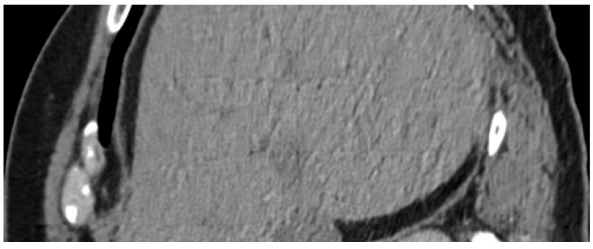
QD Image

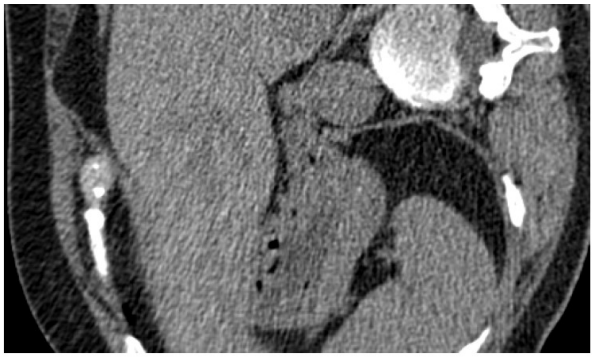


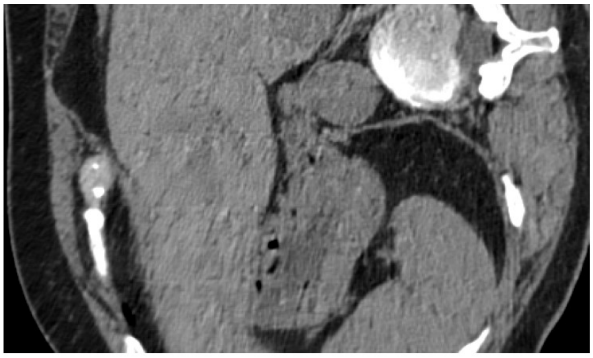
wavelet w model Predicted Image : PSNR=30.3391
SSIM=0.8801
RMSE=147.9809

QD Image









Part V : Side by side comparison of all models

```
from prettytable import PrettyTable

pt = PrettyTable()
pt.field_names = ["Model", "PSNR", "SSIM", "MSE"]

hformer_metrics = get_average_metrics(hformer_predictions,  noisy_array)
x_metrics= get_average_metrics(np.concatenate(x_model_prediction_patches, axis=0), noisy_array)
y_metrics= get_average_metrics(np.concatenate(y_model_prediction_patches, axis=0),  noisy_array)
sa_metrics = get_average_metrics(sa_predictions,  noisy_array)
z_metrics = get_average_metrics(z_predictions,  noisy_array)
w_metrics = get_average_metrics(w_predictions,  noisy_array)
wavelet_w_metrics = get_average_metrics(wavelet_w_predictions, noisy_array)
```

```
Predicted average gt-predicted PSNR -> 32.99037891246741
Predicted average gt-predicted SSIM -> 0.916629681798598
Predicted average gt-predicted MSE-> 185.87344353967302
Predicted average gt-predicted PSNR -> 8.728047851372207
Predicted average gt-predicted SSIM -> 0.332313969732048
Predicted average gt-predicted MSE-> 22949.383764070844
Predicted average gt-predicted PSNR -> 4.6170096551702695
Predicted average gt-predicted SSIM -> 0.0001014814520292278
Predicted average gt-predicted MSE-> 55501.08398167486
Predicted average gt-predicted PSNR -> 26.12000539421971
Predicted average gt-predicted SSIM -> 0.7756457741615084
Predicted average gt-predicted MSE-> 413.72842258304127
Predicted average gt-predicted PSNR -> 32.877298514970875
Predicted average gt-predicted SSIM -> 0.9232649297146059
Predicted average gt-predicted MSE-> 145.19556484099186
```

```
Predicted average gt-predicted PSNR -> 32.471087393232125
Predicted average gt-predicted SSIM -> 0.9093229017453929
Predicted average gt-predicted MSE-> 165.67989832823488
Predicted average gt-predicted PSNR -> 34.73367343929717
Predicted average gt-predicted SSIM -> 0.9482066967907415
Predicted average gt-predicted MSE-> 88.90021109734904
```

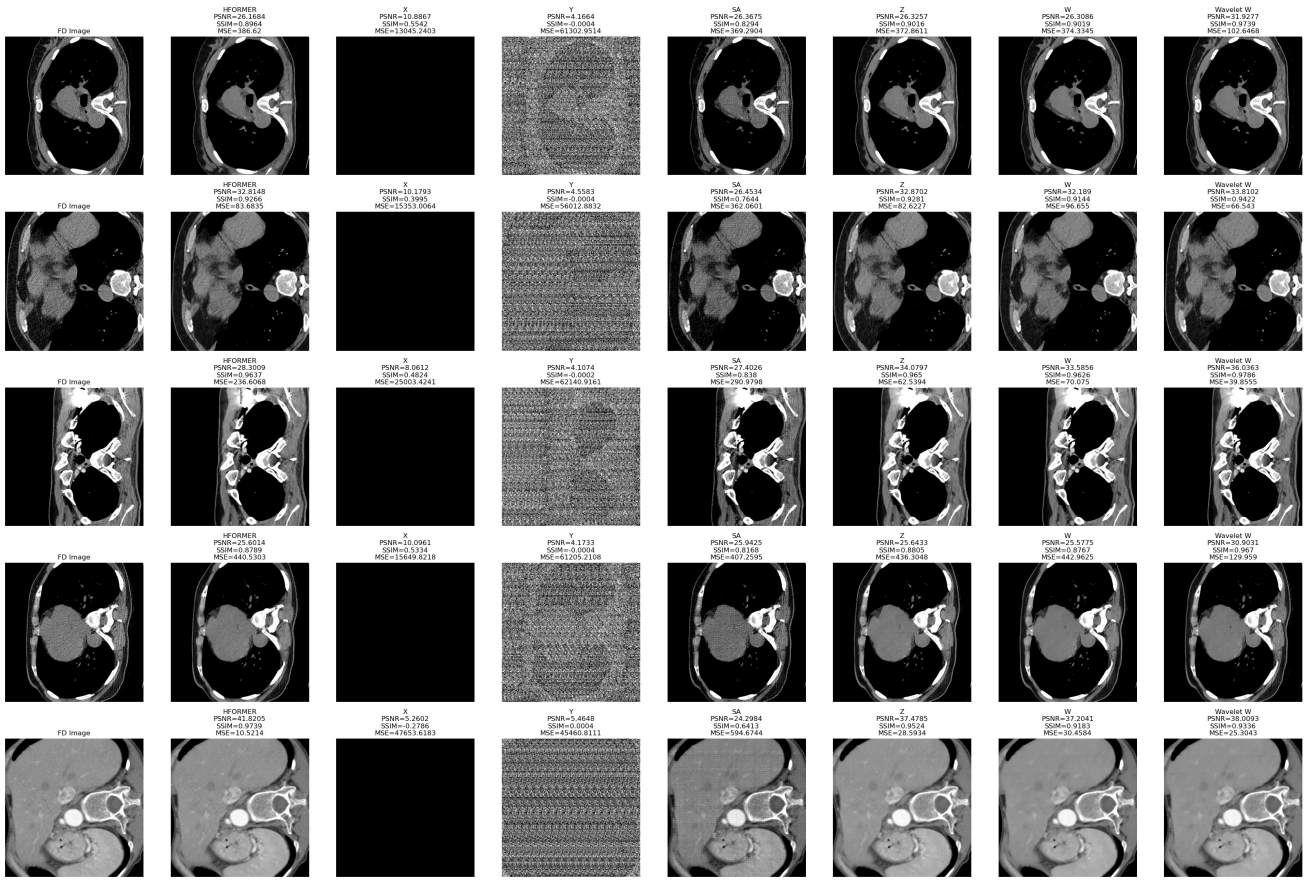
```
pt.add_row(["Original X-y pairs (No Model)", '-', '-', "-"])
pt.add_row(["Hformer",str(hformer_metrics[0]), str(hformer_metrics[1]), str(round(hformer_metrics[2], 4)
pt.add_row(["X Model",str(x_metrics[0]), str(x_metrics[1]), str(round(x_metrics[2], 4))])
pt.add_row(["Y Model",str(y_metrics[0]), str(y_metrics[1]), str(round(y_metrics[2], 4))])
pt.add_row(["SA Model",str(sa_metrics[0]), str(sa_metrics[1]), str(round(sa_metrics[2], 4))])
pt.add_row(["Z Model",str(z_metrics[0]), str(z_metrics[1]), str(round(z_metrics[2], 4))])
pt.add_row(["W Model",str(w_metrics[0]), str(w_metrics[1]), str(round(w_metrics[2], 4))])
pt.add_row(["W Wavelet Model",str(wavelet_w_metrics[0]), str(wavelet_w_metrics[1]), str(round(wavelet_w_

print(pt)
```

```
+-------------------------------+---------+--------+------------+
|             Model             |   PSNR  |  SSIM  |    MSE     |
+-------------------------------+---------+--------+------------+
| Original X-y pairs (No Model) |    -    |   -    |     -      |
|            Hformer            | 32.9904 | 0.9166 |  185.8734  |
|            X Model            |  8.728  | 0.3323 | 22949.3838 |
|            Y Model            |  4.617  | 0.0001 | 55501.084  |
|            SA Model           |  26.12  | 0.7756 |  413.7284  |
|            Z Model            | 32.8773 | 0.9233 |  145.1956  |
|            W Model            | 32.4711 | 0.9093 |  165.6799  |
|        W Wavelet Model         | 34.7337 | 0.9482 |  88.9002   |
+-------------------------------+---------+--------+------------+
```

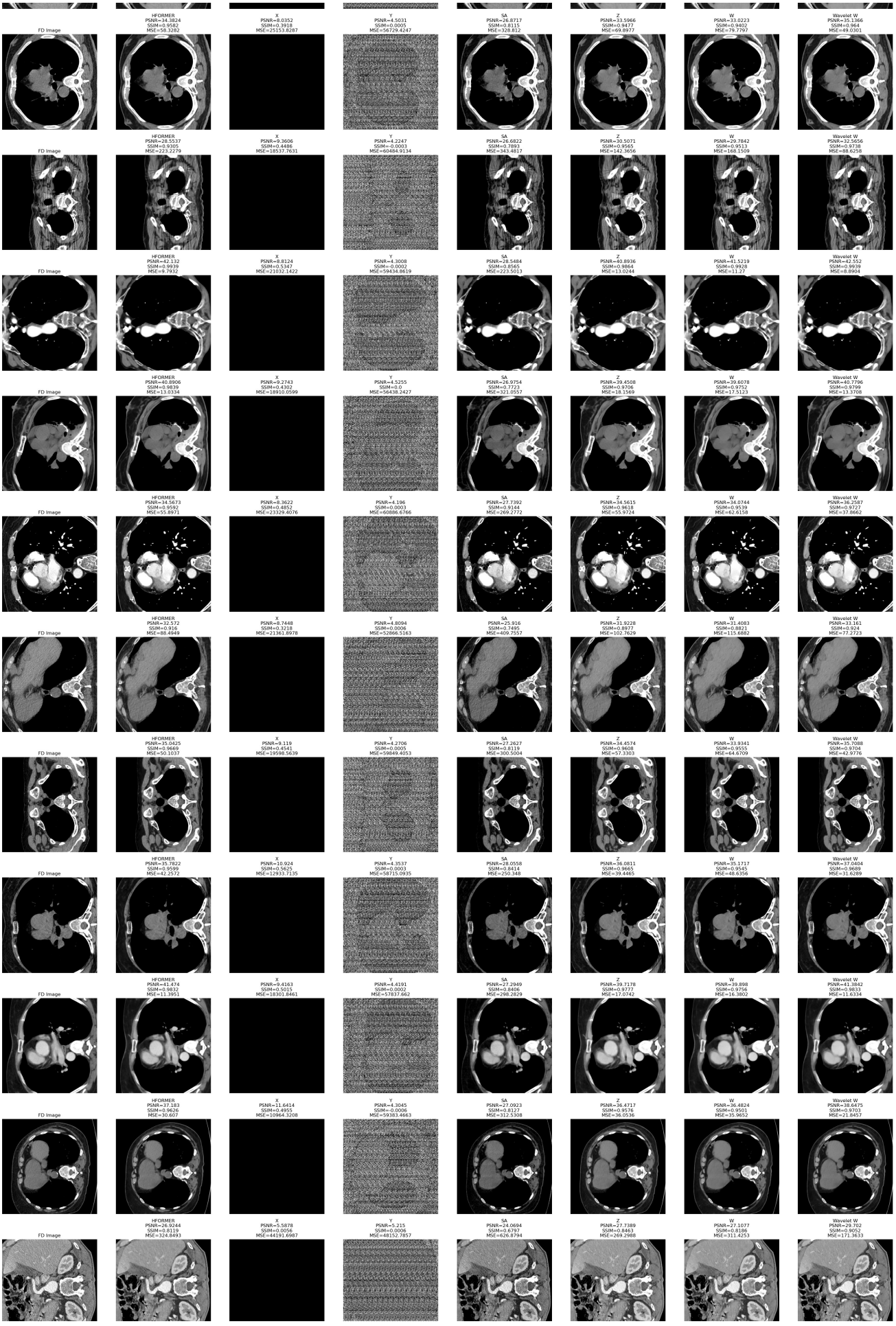
Part 6 : Output of predictions of all 4 models side by side for direct visualize comparison

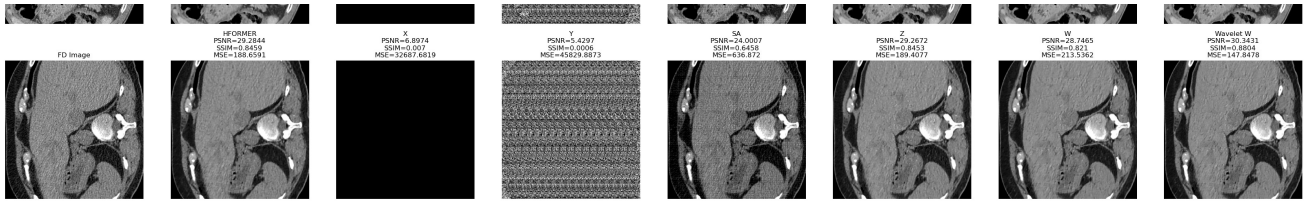
```
def visualize_predictions_all_models(X_test,  n, hformer_predictions, x_predictions, y_predictions, sa_p
    random_numbers = list(range(n))  # not very random
    for i in random_numbers:
        gt_image= X_test[i]

        hformer_pred = hformer_predictions[i]
        x_pred = x_predictions[i]
        y_pred = y_predictions[i]
        sa_pred = sa_predictions[i]
        z_pred = z_predictions[i]
        w_pred = w_predictions[i]
        wavelet_w_pred = wavelet_w_predictions[i]

        models = ["HFORMER", "X", "Y", "SA",  "Z", "W", "Wavelet W"]
        predictions = [hformer_pred, x_pred, y_pred, sa_pred, z_pred, w_pred, wavelet_w_pred]
```

```
    # Display QD and FD images
    f, axarr = plt.subplots(1, 1 + len(models), figsize=(41,41))



    axarr[0].imshow(trunc(denormalize(gt_image)), cmap='gray', vmin=-160.0, vmax=240.0)
    axarr[0].set_title("FD Image")
    axarr[0].set_axis_off()

    for j, (model_name, predicted_image) in enumerate(zip(models, predictions), start=1):
        if predicted_image.shape[-1] == 3:
            predicted_image = rgb2gray(predicted_image)

        psnr_recon = calculate_psnr(trunc(denormalize(gt_image)), trunc(denormalize(predicted_image)
        ssim_recon = calculate_ssim(trunc(denormalize(gt_image)), trunc(denormalize(predicted_image)
        rmse_recon = calculate_rmse(trunc(denormalize(gt_image)), trunc(denormalize(predicted_image)

        psnr_recon = round(psnr_recon, 4)
        ssim_recon = round(ssim_recon, 4)
        mse_recon = round(rmse_recon, 4)

        axarr[j].imshow(trunc(denormalize(predicted_image)), cmap='gray', vmin=-160.0, vmax=240.0)
        axarr[j].set_title("{}\nPSNR={}\nSSIM={}\nMSE={}".format(model_name, psnr_recon, ssim_recon,
        axarr[j].set_axis_off()

    plt.savefig('../../output/novel_comparison_lodopabct/combined_outputs_image_index_{}.png'.format
    plt.show()


visualize_predictions_all_models(noisy_array,  len(noisy_array), hformer_predictions, np.concatenate(x_m
```




FD Image
HFORMER
PSNR=32.2659
SSIM=0.9454
MSE=94.9583
X
PSNR=8.6443
SSIM=0.4224
MSE=21861.8709
Y
PSNR=4.4328
SSIM=0.0005
MSE=57655.9252
SA
PSNR=26.703
SSIM=0.8122
MSE=341.8368
Z
PSNR=32.3512
SSIM=0.9465
MSE=93.1105
W
PSNR=31.7096
SSIM=0.9357
MSE=107.9356
Wavelet W
PSNR=34.1252
SSIM=0.9626
MSE=61.6872
FD Image
HFORMER
PSNR=26.4636
SSIM=0.8109
MSE=361.2106
X
PSNR=6.6802
SSIM=0.1716
MSE=27296.0388
Y
PSNR=4.8648
SSIM=0.0
MSE=52195.8952
SA
PSNR=24.9777
SSIM=0.7051
MSE=508.5662
Z
PSNR=28.007
SSIM=0.8746
MSE=253.1738
W
PSNR=27.1037
SSIM=0.8407
MSE=311.7087
Wavelet W
PSNR=29.0124
SSIM=0.9013
MSE=200.855
FD Image
HFORMER
PSNR=35.6378
SSIM=0.9714
MSE=43.6862
X
PSNR=10.8561
SSIM=0.5423
MSE=13137.4986
Y
PSNR=4.4488
SSIM=-0.0
MSE=57443.4298
SA
PSNR=27.7271
SSIM=0.8256
MSE=270.0291
Z
PSNR=35.3584
SSIM=0.9638
MSE=46.5892
W
PSNR=34.8329
SSIM=0.9613
MSE=52.5817
Wavelet W
PSNR=36.967
SSIM=0.9754
MSE=32.1677
FD Image
HFORMER
PSNR=32.3639
SSIM=0.8886
MSE=92.8388
X
PSNR=7.9647
SSIM=0.0919
MSE=25565.5075
Y
PSNR=5.1998
SSIM=-0.0001
MSE=48321.0196
SA
PSNR=24.0788
SSIM=0.7004
MSE=625.5241
Z
PSNR=31.8382
SSIM=0.8806
MSE=104.7857
W
PSNR=31.347
SSIM=0.858
MSE=117.3333
Wavelet W
PSNR=33.5557
SSIM=0.918
MSE=70.5593
FD Image
HFORMER
PSNR=31.1313
SSIM=0.902
MSE=123.3081
X
PSNR=8.3301
SSIM=0.4146
MSE=23502.0704
Y
PSNR=4.5733
SSIM=0.0004
MSE=55819.6447
SA
PSNR=26.4222
SSIM=0.8758
MSE=364.6692
Z
PSNR=31.2882
SSIM=0.9128
MSE=118.9311
W
PSNR=30.8842
SSIM=0.8949
MSE=130.5255
Wavelet W
PSNR=33.8752
SSIM=0.9503
MSE=65.5536
FD Image
HFORMER
PSNR=19.1269
SSIM=0.5665
MSE=1956.2586
X
PSNR=6.5612
SSIM=0.0194
MSE=35318.7173
Y
PSNR=5.1896
SSIM=0.0002
MSE=48435.1525
SA
PSNR=22.7435
SSIM=0.6753
MSE=850.6863
Z
PSNR=21.54
SSIM=0.7744
MSE=1122.3213
W
PSNR=20.5995
SSIM=0.7084
MSE=1393.6862
Wavelet W
PSNR=23.1917
SSIM=0.8523
MSE=767.2808
FD Image
HFORMER
PSNR=34.4765
SSIM=0.9316
MSE=57.0778
X
PSNR=8.859
SSIM=0.3801
MSE=20807.2842
Y
PSNR=4.3398
SSIM=-0.0003
MSE=58903.3505
SA
PSNR=25.9933
SSIM=0.7981
MSE=402.5206
Z
PSNR=34.3974
SSIM=0.9388
MSE=58.1272
W
PSNR=34.0442
SSIM=0.9246
MSE=63.0518
Wavelet W
PSNR=36.6601
SSIM=0.9619
MSE=34.5229
FD Image
HFORMER
PSNR=34.1495
SSIM=0.9282
MSE=61.5418
X
PSNR=10.2639
SSIM=0.3267
MSE=15056.6875
Y
PSNR=4.6127
SSIM=-0.0001
MSE=55316.1743
SA
PSNR=25.5639
SSIM=0.7296
MSE=444.3513
Z
PSNR=33.5099
SSIM=0.9145
MSE=71.3066
W
PSNR=33.2503
SSIM=0.9032
MSE=75.6983
Wavelet W
PSNR=35.6135
SSIM=0.9457
MSE=43.931
FD Image
HFORMER
PSNR=34.0243
SSIM=0.9189
MSE=63.3414
X
PSNR=9.5935
SSIM=0.287
MSE=17570.0697
Y
PSNR=4.6076
SSIM=0.0001
MSE=55380.8289
SA
PSNR=25.5908
SSIM=0.7364
MSE=441.6115
Z
PSNR=33.3663
SSIM=0.9018
MSE=73.7035
W
PSNR=33.1122
SSIM=0.8877
MSE=78.1446
Wavelet W
PSNR=35.351
SSIM=0.9357
MSE=46.6684
FD Image
HFORMER
PSNR=34.6217
SSIM=0.9598
MSE=55.2017
X
PSNR=8.5387
SSIM=0.3791
MSE=22400.2655
Y
PSNR=4.5808
SSIM=0.0004
MSE=55723.3379
SA
PSNR=26.6014
SSIM=0.7828
MSE=349.928
Z
PSNR=33.9711
SSIM=0.9499
MSE=64.1228
W
PSNR=33.3553
SSIM=0.9436
MSE=73.89
Wavelet W
PSNR=35.5567
SSIM=0.9661
MSE=44.5088
FD Image
HFORMER
PSNR=35.9747
SSIM=0.9306
MSE=40.4251
X
PSNR=6.4351
SSIM=-0.0598
MSE=36358.7001
Y
PSNR=5.4038
SSIM=0.0002
MSE=46103.8848
SA
PSNR=23.9854
SSIM=0.6613
MSE=639.1116
Z
PSNR=33.8727
SSIM=0.891
MSE=65.5919
W
PSNR=33.3275
SSIM=0.8583
MSE=74.3659
Wavelet W
PSNR=34.6684
SSIM=0.8982
MSE=54.6106


FD Image
HFORMER
PSNR=34.3824
SSIM=0.9582
MSE=58.3282
X
PSNR=8.0352
SSIM=0.3918
MSE=25153.8287
Y
PSNR=4.5031
SSIM=0.0005
MSE=56729.4247
SA
PSNR=26.8717
SSIM=0.8115
MSE=328.812
Z
PSNR=33.5966
SSIM=0.9477
MSE=69.8977
W
PSNR=33.0223
SSIM=0.9402
MSE=79.7797
Wavelet W
PSNR=35.1366
SSIM=0.964
MSE=49.0301
FD Image
HFORMER
PSNR=28.5537
SSIM=0.9305
MSE=223.2279
X
PSNR=9.3606
SSIM=0.4486
MSE=18537.7631
Y
PSNR=4.2247
SSIM=-0.0003
MSE=60484.9134
SA
PSNR=26.6822
SSIM=0.7893
MSE=343.4817
Z
PSNR=30.5071
SSIM=0.9565
MSE=142.3656
W
PSNR=29.7842
SSIM=0.9513
MSE=168.1509
Wavelet W
PSNR=32.5656
SSIM=0.9738
MSE=88.6258
FD Image
HFORMER
PSNR=42.132
SSIM=0.9939
MSE=9.7932
X
PSNR=8.8124
SSIM=0.5347
MSE=21032.1422
Y
PSNR=4.3008
SSIM=-0.0002
MSE=59434.8619
SA
PSNR=28.5484
SSIM=0.8565
MSE=223.5013
Z
PSNR=40.8936
SSIM=0.9864
MSE=13.0244
W
PSNR=41.5219
SSIM=0.9928
MSE=11.27
Wavelet W
PSNR=42.552
SSIM=0.9939
MSE=8.8904
FD Image
HFORMER
PSNR=40.8906
SSIM=0.9839
MSE=13.0334
X
PSNR=9.2743
SSIM=0.4302
MSE=18910.0599
Y
PSNR=4.5255
SSIM=0.0
MSE=56438.2427
SA
PSNR=26.9754
SSIM=0.7723
MSE=321.0557
Z
PSNR=39.4508
SSIM=0.9706
MSE=18.1569
W
PSNR=39.6078
SSIM=0.9752
MSE=17.5123
Wavelet W
PSNR=40.7796
SSIM=0.9799
MSE=13.3708
FD Image
HFORMER
PSNR=34.5673
SSIM=0.9592
MSE=55.8971
X
PSNR=8.3622
SSIM=0.4852
MSE=23329.4076
Y
PSNR=4.196
SSIM=0.0003
MSE=60866.6766
SA
PSNR=27.7392
SSIM=0.9144
MSE=269.2772
Z
PSNR=34.5615
SSIM=0.9618
MSE=55.9724
W
PSNR=34.0744
SSIM=0.9539
MSE=62.6158
Wavelet W
PSNR=36.2587
SSIM=0.9727
MSE=37.8662
FD Image
HFORMER
PSNR=32.572
SSIM=0.916
MSE=88.4949
X
PSNR=8.7448
SSIM=0.3218
MSE=21361.8978
Y
PSNR=4.8094
SSIM=0.0006
MSE=52866.5163
SA
PSNR=25.916
SSIM=0.7495
MSE=409.7557
Z
PSNR=31.9228
SSIM=0.8977
MSE=102.7629
W
PSNR=31.4083
SSIM=0.8821
MSE=115.6882
Wavelet W
PSNR=33.161
SSIM=0.9234
MSE=77.2723
FD Image
HFORMER
PSNR=35.0425
SSIM=0.9669
MSE=50.1037
X
PSNR=9.119
SSIM=0.4541
MSE=19598.5639
Y
PSNR=4.2706
SSIM=0.0005
MSE=59849.4053
SA
PSNR=27.2627
SSIM=0.8119
MSE=300.5004
Z
PSNR=33.4574
SSIM=0.9608
MSE=57.3303
W
PSNR=33.9341
SSIM=0.9555
MSE=64.6709
Wavelet W
PSNR=35.7088
SSIM=0.9704
MSE=42.9776
FD Image
HFORMER
PSNR=35.7822
SSIM=0.9599
MSE=42.2572
X
PSNR=10.924
SSIM=0.5625
MSE=12933.7135
Y
PSNR=4.3537
SSIM=0.0003
MSE=58715.0935
SA
PSNR=28.0558
SSIM=0.8414
MSE=250.348
Z
PSNR=36.0811
SSIM=0.9665
MSE=39.4465
W
PSNR=35.1717
SSIM=0.9545
MSE=48.6356
Wavelet W
PSNR=37.0404
SSIM=0.9689
MSE=31.6289
FD Image
HFORMER
PSNR=41.474
SSIM=0.9832
MSE=11.3951
X
PSNR=9.4163
SSIM=0.5015
MSE=18301.8461
Y
PSNR=4.4191
SSIM=0.0002
MSE=57837.662
SA
PSNR=27.2949
SSIM=0.8406
MSE=298.2829
Z
PSNR=39.7178
SSIM=0.9777
MSE=17.0742
W
PSNR=39.898
SSIM=0.9756
MSE=16.3802
Wavelet W
PSNR=41.3842
SSIM=0.9833
MSE=11.6334
FD Image
HFORMER
PSNR=37.183
SSIM=0.9626
MSE=30.607
X
PSNR=11.6414
SSIM=0.4955
MSE=10964.3208
Y
PSNR=4.3045
SSIM=-0.0006
MSE=59383.4663
SA
PSNR=27.0923
SSIM=0.8127
MSE=312.5308
Z
PSNR=36.4717
SSIM=0.9576
MSE=36.0536
W
PSNR=36.4824
SSIM=0.9501
MSE=35.9652
Wavelet W
PSNR=38.6475
SSIM=0.9703
MSE=21.8457
FD Image
HFORMER
PSNR=26.9244
SSIM=0.8119
MSE=324.8493
X
PSNR=5.5978
SSIM=0.0056
MSE=44191.6987
Y
PSNR=5.215
SSIM=0.0006
MSE=48152.7857
SA
PSNR=24.0694
SSIM=0.6797
MSE=626.8794
Z
PSNR=27.7389
SSIM=0.8463
MSE=269.2988
W
PSNR=27.1077
SSIM=0.8186
MSE=311.4253
Wavelet W
PSNR=29.702
SSIM=0.9052
MSE=171.3633


FD Image
HFORMER
PSNR=29.2844
SSIM=0.8459
MSE=188.6591
X
PSNR=6.8974
SSIM=0.007
MSE=32687.6819
Y
PSNR=5.4297
SSIM=0.0006
MSE=45829.8873
SA
PSNR=24.0007
SSIM=0.6458
MSE=636.872
Z
PSNR=29.2672
SSIM=0.8453
MSE=189.4077
W
PSNR=28.7465
SSIM=0.821
MSE=213.5362
Wavelet W
PSNR=30.3431
SSIM=0.8804
MSE=147.8478