Appendix 1

Codes for the used network

```
import numpy as np
import os
import skimage.io as io
import skimage.transform as trans
from scipy import ndimage
from tensorflow.keras.models import *
from tensorflow.keras.layers import *
from tensorflow.keras.optimizers import *
from tensorflow.keras.callbacks import ModelCheckpoint, LearningRateScheduler
from tensorflow.keras import backend as keras
from tensorflow.keras import backend as K
import tensorflow as tf
from tensorflow.keras.applications import *

smooth = 1
def dice_coef(y_true, y_pred):
  y_true_f = K.flatten(y_true)
  y_pred_f = K.flatten(y_pred)
  intersection = K.sum(y_true_f * y_pred_f)
  return (2. * intersection + smooth) / (K.sum(y_true_f * y_true_f) + K.sum(y_pred_f * y_pred_f) + smooth)

def dice_coef_loss(y_true, y_pred):
  return 1. - dice_coef(y_true, y_pred)

from tensorflow.keras import regularizers

def DenseUnet(pretrained_weights = None,input_size = (256,256,1)):
  inputs = Input(input_size)

  conv1_1 = Conv2D(64, 3, activation='relu', padding='same', kernel_initializer='he_normal')(inputs)
  BatchNorm1_1 = BatchNormalization(axis=3, gamma_regularizer=regularizers.l2(1e-4),
            beta_regularizer=regularizers.l2(1e-4))(conv1_1)
  ReLU1_1 = Activation('relu')(BatchNorm1_1)
  conv1_2 = Conv2D(64, 3, activation='relu', padding='same', kernel_initializer='he_normal')(ReLU1_1)
  drop1_2 = Dropout(0)(conv1_2)#
  # Merge1 = merge([conv1_1,drop1_2], mode = 'concat', concat_axis = 3)
  Merge1 = Concatenate(axis=3)([conv1_1, drop1_2])

  pool1 = MaxPooling2D(pool_size=(2, 2))(Merge1)

  conv2_1 = Conv2D(128, 3, activation='relu', padding='same', kernel_initializer='he_normal')(pool1)
  BatchNorm2_1 = BatchNormalization(axis=3, gamma_regularizer=regularizers.l2(1e-4),
            beta_regularizer=regularizers.l2(1e-4))(conv2_1)
  ReLU2_1 = Activation('relu')(BatchNorm2_1)
  conv2_2 = Conv2D(128, 3, activation='relu', padding='same', kernel_initializer='he_normal')(ReLU2_1)
```


https://dx.doi.org/10.21037/qims-22-1198

```
drop2_2 = Dropout(0)(conv2_2)#
# Merge2 = merge([conv2_1,drop2_2], mode = 'concat', concat_axis = 3)
Merge2 = Concatenate(axis=3)([conv2_1, drop2_2])
pool2 = MaxPooling2D(pool_size=(2, 2))(Merge2)
conv3_1 = Conv2D(256, 3, activation='relu', padding='same', kernel_initializer='he_normal')(pool2)
BatchNorm3_1 = BatchNormalization(axis=3, gamma_regularizer=regularizers.l2(1e-4),
            beta_regularizer=regularizers.l2(1e-4))(conv3_1)
ReLU3_1 = Activation('relu')(BatchNorm3_1)
conv3_2 = Conv2D(256, 3, activation='relu', padding='same', kernel_initializer='he_normal')(ReLU3_1)
drop3_2 = Dropout(0)(conv3_2)#
# Merge3 = merge([conv3_1,drop3_2], mode = 'concat', concat_axis = 3)
Merge3 = Concatenate(axis=3)([conv3_1, drop3_2])

pool3 = MaxPooling2D(pool_size=(2, 2))(Merge3)

conv4_1 = Conv2D(512, 3, activation='relu', padding='same', kernel_initializer='he_normal')(pool3)
BatchNorm4_1 = BatchNormalization(axis=3, gamma_regularizer=regularizers.l2(1e-4),
            beta_regularizer=regularizers.l2(1e-4))(conv4_1)
ReLU4_1 = Activation('relu')(BatchNorm4_1)
conv4_2 = Conv2D(512, 3, activation='relu', padding='same', kernel_initializer='he_normal')(ReLU4_1)
drop4_2 = Dropout(0)(conv4_2)#
# Merge4 = merge([conv4_1,drop4_2], mode = 'concat', concat_axis = 3)
Merge4 = Concatenate(axis=3)([conv4_1, drop4_2])
drop4 = Dropout(0.5)(Merge4)
pool4 = MaxPooling2D(pool_size=(2, 2))(drop4)

# conv5_1 = Conv2D(1024, 3, activation='relu', padding='same', kernel_initializer='he_normal')(pool4)
# BatchNorm5_1 = BatchNormalization(axis=3, gamma_regularizer=regularizers.l2(1e-4),
#              beta_regularizer=regularizers.l2(1e-4))(conv5_1)
# ReLU5_1 = Activation('relu')(BatchNorm5_1)
# conv5_2 = Conv2D(1024, 3, activation='relu', padding='same', kernel_initializer='he_normal')(ReLU5_1)
# drop5_2 = Dropout(0)(conv5_2)#
# # Merge5 = merge([conv5_1,drop5_2], mode = 'concat', concat_axis = 3)
# Merge5 = Concatenate(axis=3)([conv5_2, drop5_2])
# drop5 = Dropout(0.5)(Merge5)
# DAC
branch1 = Conv2D(512, 3, activation='relu', padding= 'same', dilation_rate=1)(pool4)
branch2 = Conv2D(512, 3, activation='relu', padding='same', dilation_rate=3)(pool4)
branch2 = Conv2D(512, 1, activation='relu', padding='same', dilation_rate=1)(branch2)

branch3 = Conv2D(512, 3, activation='relu', padding='same', dilation_rate=1)(pool4)
branch3 = Conv2D(512, 3, activation='relu', padding='same', dilation_rate=3)(branch3)
branch3 = Conv2D(512, 1, activation='relu', padding='same', dilation_rate=1)(branch3)

branch4 = Conv2D(512, 3, activation='relu', padding='same', dilation_rate=1)(pool4)
branch4 = Conv2D(512, 3, activation='relu', padding='same', dilation_rate=3)(branch4)
branch4 = Conv2D(512, 3, activation='relu', padding='same', dilation_rate=5)(branch4)
branch4 = Conv2D(512, 1, activation='relu', padding='same', dilation_rate=1)(branch4)
```


https://dx.doi.org/10.21037/qims-22-1198

```
# pool4 = Concatenate(axis=3)([branch1, branch2, branch3, branch4, pool4])
pool4 = branch1 + branch2 + branch3 + branch4

# RMP

up6 = Conv2D(512, 2, activation='relu', padding='same', kernel_initializer='he_normal')(
  UpSampling2D(size=(2, 2))(pool4))
# merge6 = merge([drop4,up6], mode = 'concat', concat_axis = 3)
# drop4_at = cbam_block(drop4, name = "4")
merge6 = Concatenate(axis=3)([drop4, up6])

conv6_1 = Conv2D(512, 3, activation='relu', padding='same', kernel_initializer='he_normal')(merge6)
BatchNorm6_1 = BatchNormalization(axis=3, gamma_regularizer=regularizers.l2(1e-4),
            beta_regularizer=regularizers.l2(1e-4))(conv6_1)
ReLU6_1 = Activation('relu')(BatchNorm6_1)
conv6_2 = Conv2D(512, 3, activation='relu', padding='same', kernel_initializer='he_normal')(ReLU6_1)
drop6_2 = Dropout(0)(conv6_2)#

# Merge6 = merge([conv6_1,drop6_2], mode = 'concat', concat_axis = 3)
Merge6 = Concatenate(axis=3)([conv6_1, drop6_2])

up7 = Conv2D(256, 2, activation='relu', padding='same', kernel_initializer='he_normal')(
  UpSampling2D(size=(2, 2))(Merge6))
# merge7 = merge([Merge3,up7], mode = 'concat', concat_axis = 3)
# Merge3_at = cbam_block(Merge3, name = '3')
merge7 = Concatenate(axis=3)([Merge3, up7])

conv7_1 = Conv2D(256, 3, activation='relu', padding='same', kernel_initializer='he_normal')(merge7)
BatchNorm7_1 = BatchNormalization(axis=3, gamma_regularizer=regularizers.l2(1e-4),
            beta_regularizer=regularizers.l2(1e-4))(conv7_1)
ReLU7_1 = Activation('relu')(BatchNorm7_1)
conv7_2 = Conv2D(256, 3, activation='relu', padding='same', kernel_initializer='he_normal')(ReLU7_1)
drop7_2 = Dropout(0)(conv7_2)#
# Merge7 = merge([conv7_1,drop7_2], mode = 'concat', concat_axis = 3)
Merge7 = Concatenate(axis=3)([conv7_1, drop7_2])


up8 = Conv2D(128, 2, activation='relu', padding='same', kernel_initializer='he_normal')(
  UpSampling2D(size=(2, 2))(Merge7))
# merge8 = merge([Merge2,up8], mode = 'concat', concat_axis = 3)
# Merge2_at = cbam_block(Merge2, name = '2')
merge8 = Concatenate(axis=3)([Merge2, up8])

conv8_1 = Conv2D(128, 3, activation='relu', padding='same', kernel_initializer='he_normal')(merge8)
BatchNorm8_1 = BatchNormalization(axis=3, gamma_regularizer=regularizers.l2(1e-4),
            beta_regularizer=regularizers.l2(1e-4))(conv8_1)
ReLU8_1 = Activation('relu')(BatchNorm8_1)
conv8_2 = Conv2D(128, 3, activation='relu', padding='same', kernel_initializer='he_normal')(ReLU8_1)
```


https://dx.doi.org/10.21037/qims-22-1198

```
drop8_2 = Dropout(0)(conv8_2)#
# Merge8 = merge([conv8_1,drop8_2], mode = 'concat', concat_axis = 3)
Merge8 = Concatenate(axis=3)([conv8_1, drop8_2])

up9 = Conv2D(64, 2, activation='relu', padding='same', kernel_initializer='he_normal')(
  UpSampling2D(size=(2, 2))(Merge8))
# merge9 = merge([Merge1,up9], mode = 'concat', concat_axis = 3)
# Merge1_at = cbam_block(Merge1, name = '1')
merge9 = Concatenate(axis=3)([Merge1, up9])

conv9_1 = Conv2D(64, 3, activation='relu', padding='same', kernel_initializer='he_normal')(merge9)
BatchNorm9_1 = BatchNormalization(axis=3, gamma_regularizer=regularizers.l2(1e-4),
            beta_regularizer=regularizers.l2(1e-4))(conv9_1)
ReLU9_1 = Activation('relu')(BatchNorm9_1)
conv9_2 = Conv2D(64, 3, activation='relu', padding='same', kernel_initializer='he_normal')(ReLU9_1)
drop9_2 = Dropout(0)(conv9_2)#
# Merge9 = merge([conv9_1,drop9_2], mode = 'concat', concat_axis = 3)
Merge9 = Concatenate(axis=3)([conv9_1, drop9_2])

conv9 = Conv2D(2, 3, activation='relu', padding='same', kernel_initializer='he_normal')(Merge9)
conv10 = Conv2D(1, 1, activation='sigmoid')(conv9) # sigmoid
# conv10 = Conv2D(1, 1, activation = 'softmax')(conv9)#sigmoid

model = Model(inputs, conv10)
model.compile(optimizer=Adam(lr=3e-4), loss=dice_coef_loss, metrics=[dice_coef, 'acc'])

model.summary()

if (pretrained_weights):
  model.load_weights(pretrained_weights)

return model
```


https://dx.doi.org/10.21037/qims-22-1198

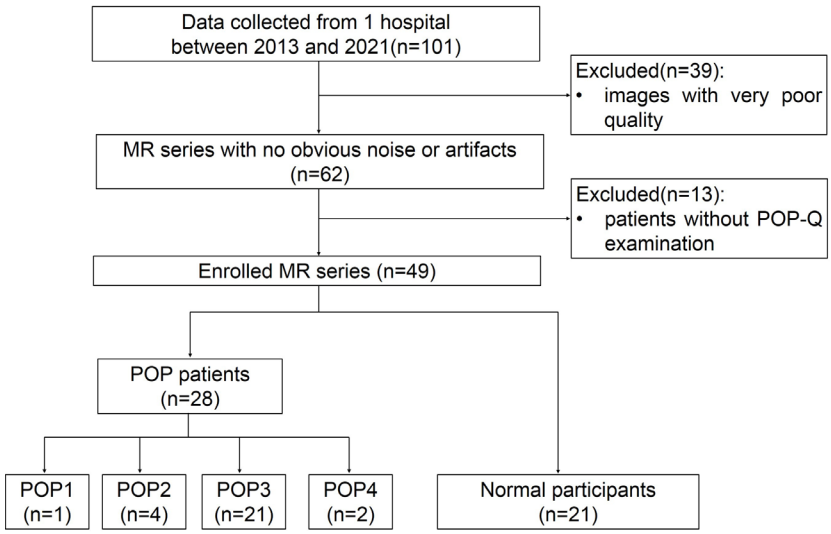

Figure S1 Flowchart of data collection. MR, magnetic resonance; POP, pelvic organ prolapse; POP-Q, pelvic organ prolapse quantification; POP 1, patients with stage 1 pelvic organ prolapse; POP2, patients with stage 2 pelvic organ prolapse; POP3, patients with stage 3 pelvic organ prolapse; POP4, patients with stage 4 pelvic organ prolapse.

Table S1 Training time of the improved DenseUnet and compared models for 3 structures

Models	LAM	EAS	IOM
Unet	0:26'54"	0:19'6"	0:31'4"
Unet++	0:56'17"	0:40'33"	1:5'3"
ResUnet	1:45'13"	1:16'43"	2:1'42"
DenseUnet	1:51'24"	0:24'32"	0:45'2"
DenseUnet†	2:26'4"	0:38'26"	1:0'39"

†, denotes the improved DenseUet. All values are described as hour:minute'second. LAM, levator ani muscle; EAS, external anal sphincter; IOM, internal obturator muscle.

https://dx.doi.org/10.21037/qims-22-1198