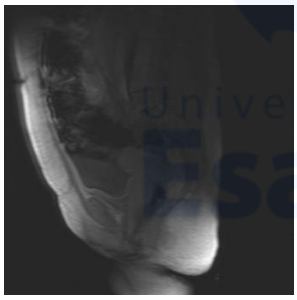
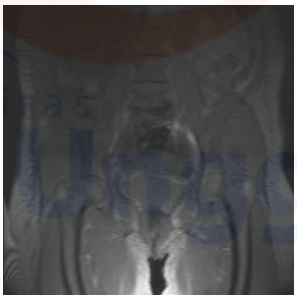
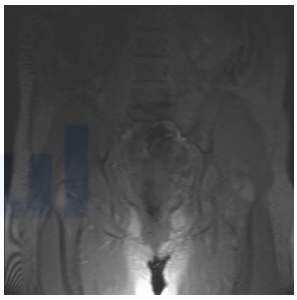
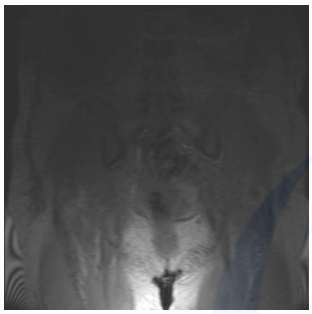
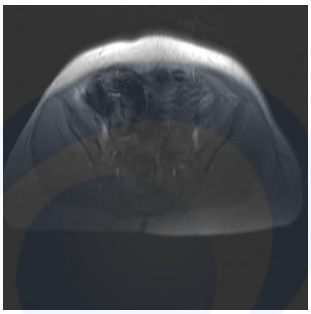
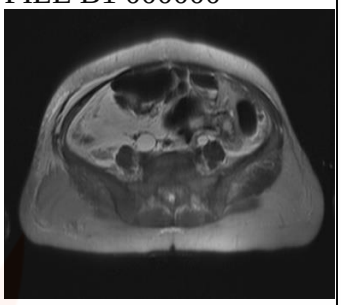
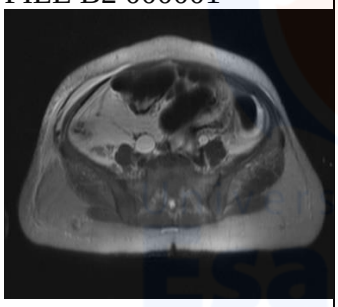
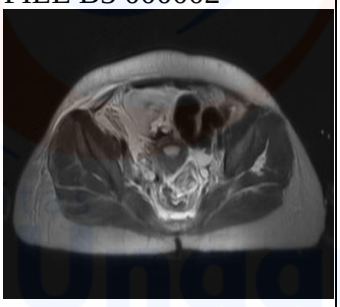
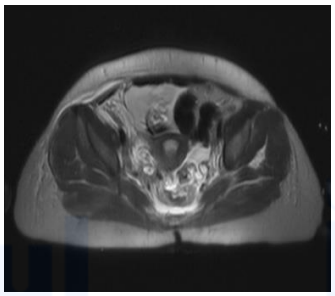
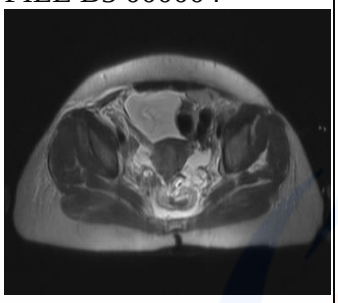
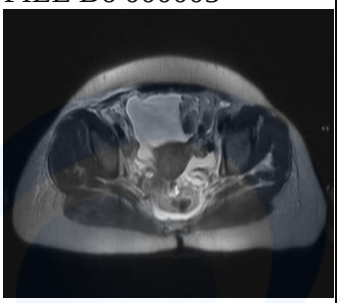
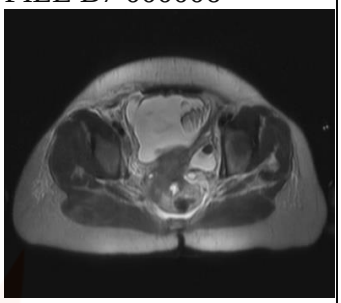
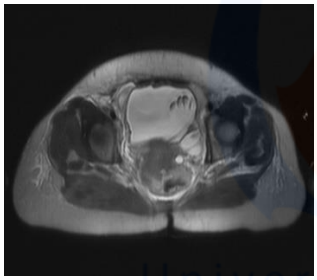
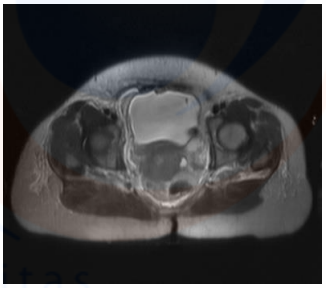
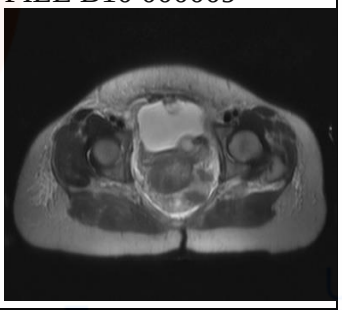
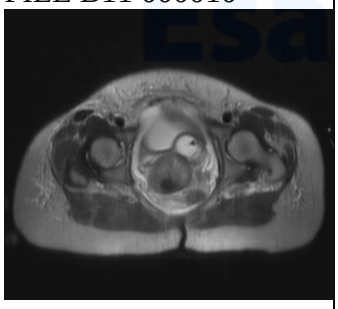
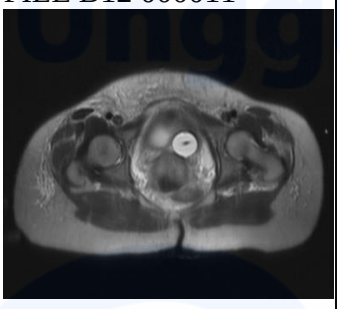
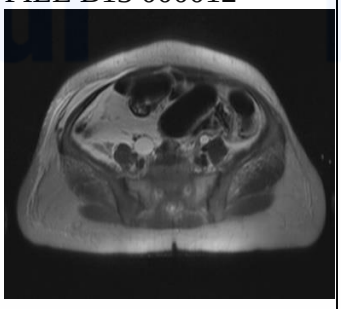
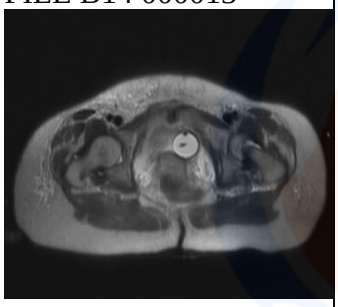
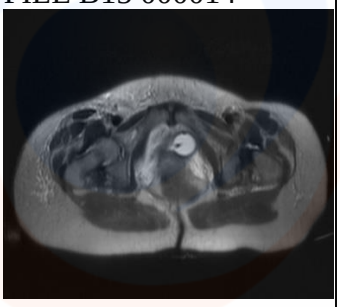
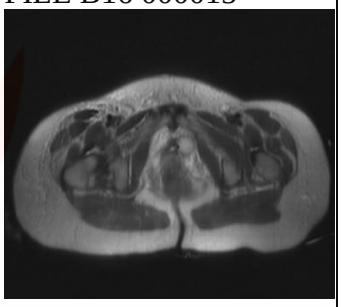
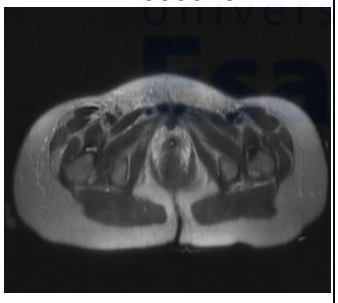
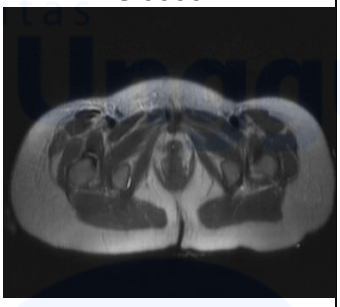
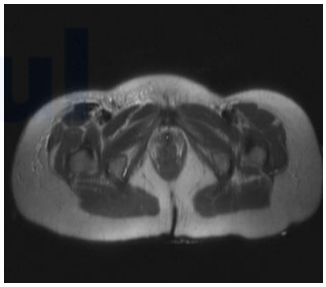
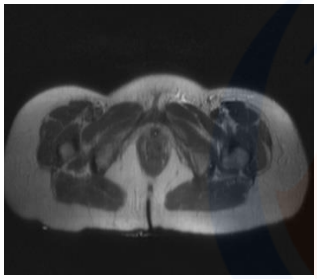
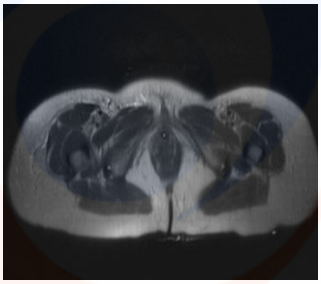
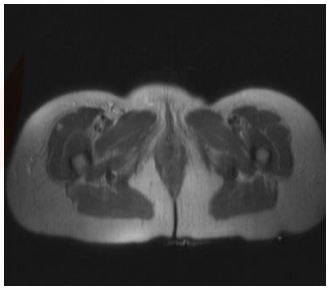
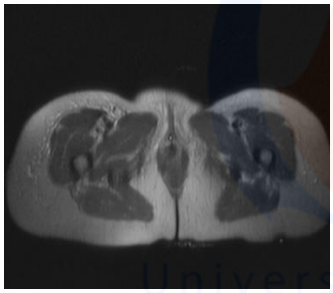
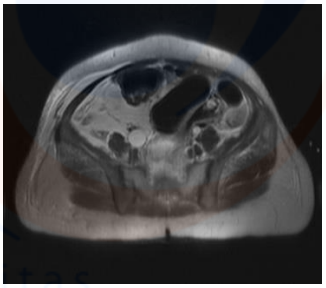
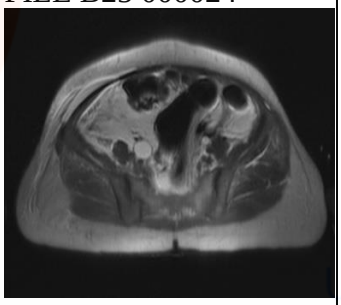
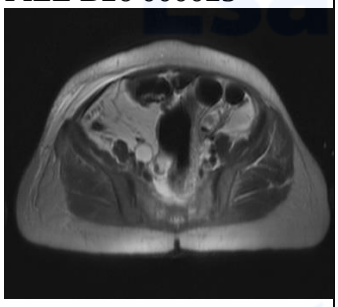
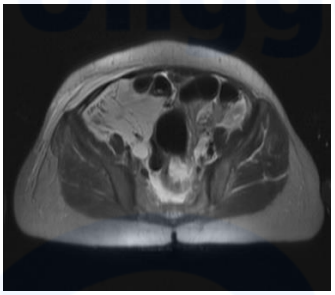
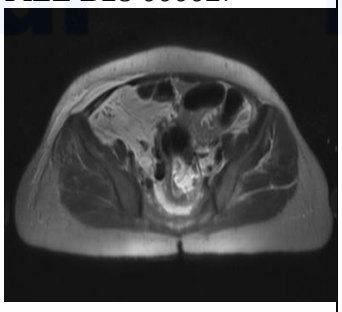
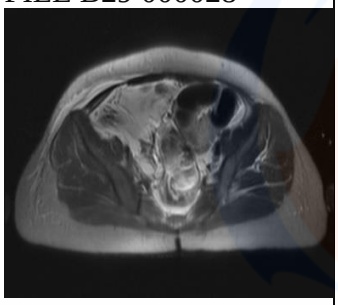
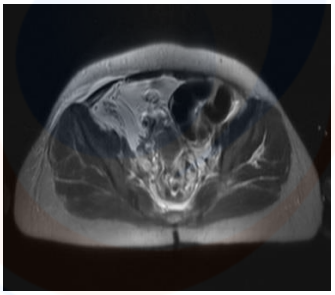
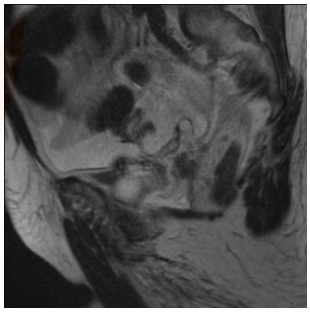
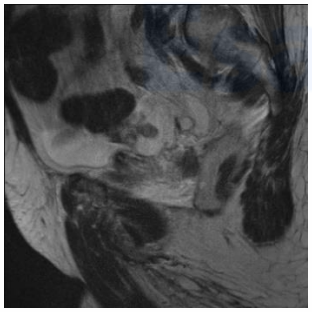
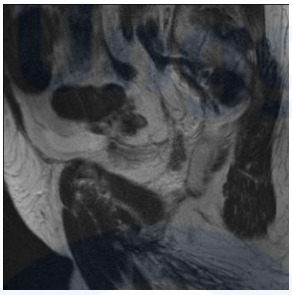
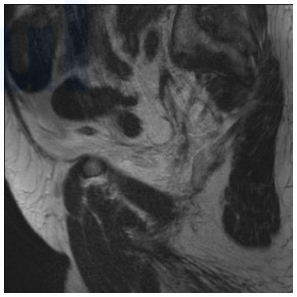
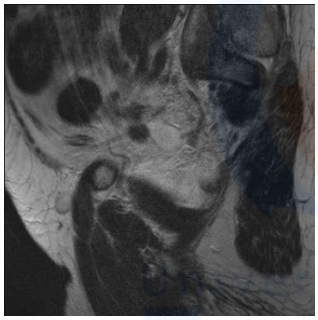
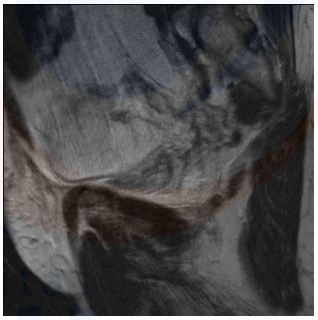
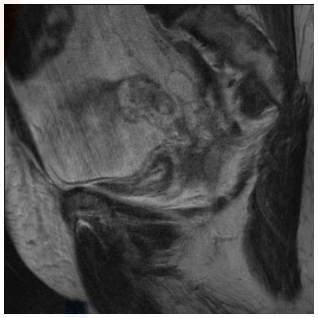
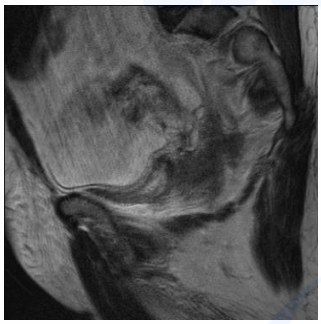
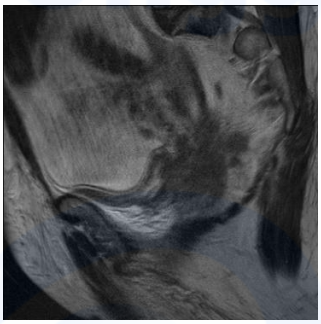
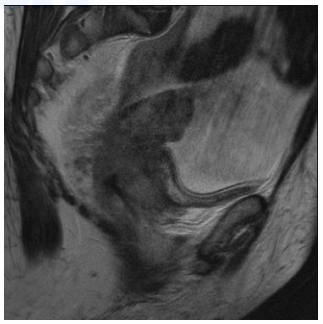
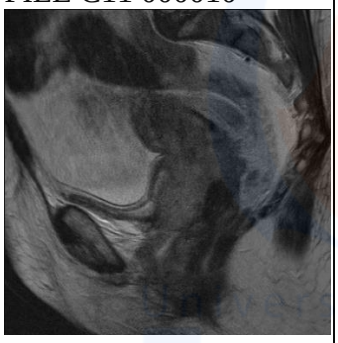
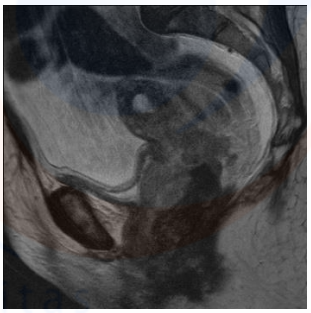
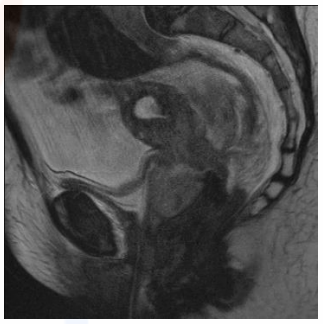
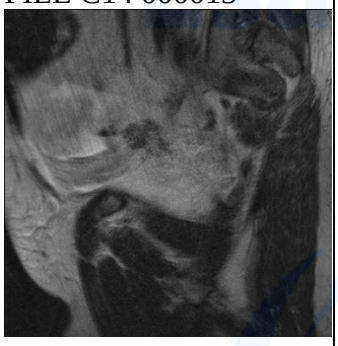
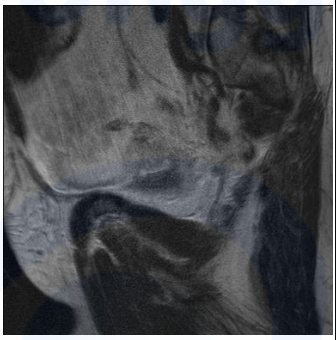
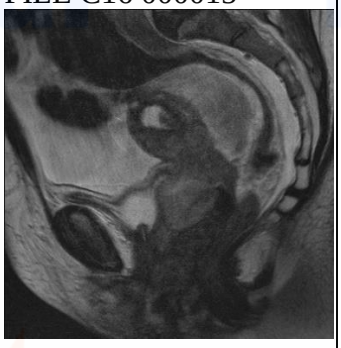
LAMPIRAN

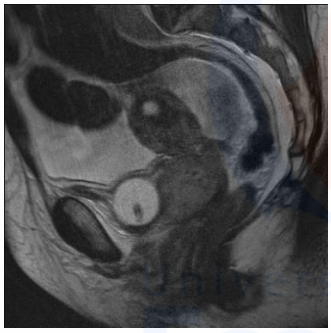
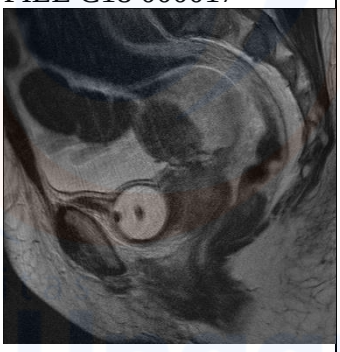
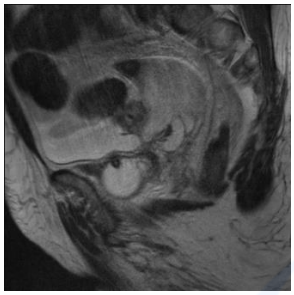
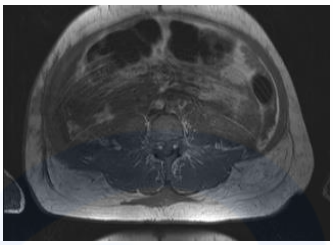
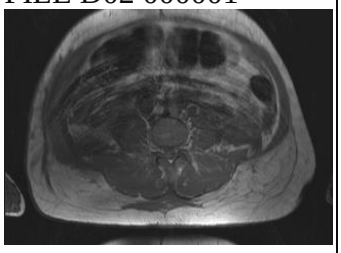
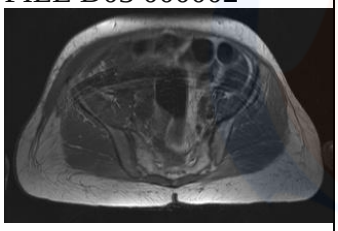
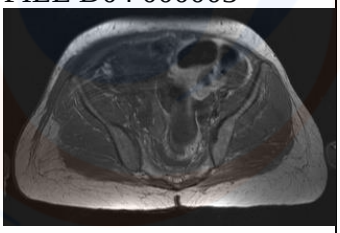
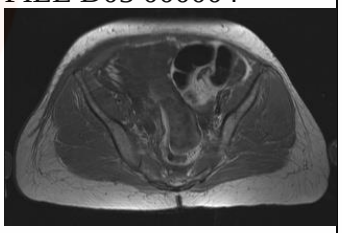
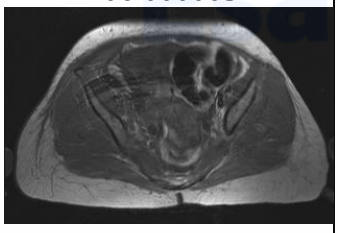
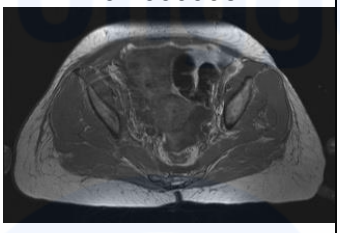
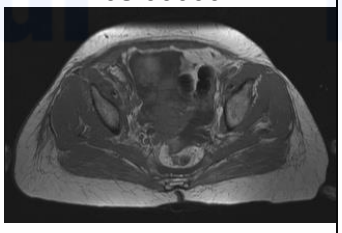
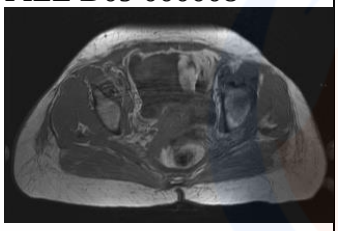
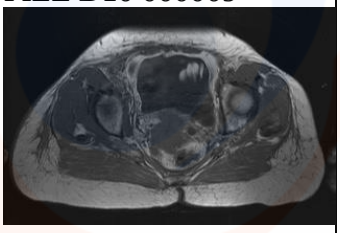
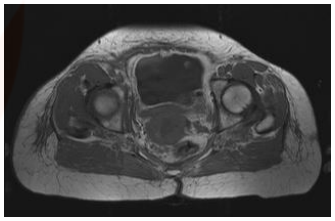
1. Dataset (cancerimagingarchive.net)

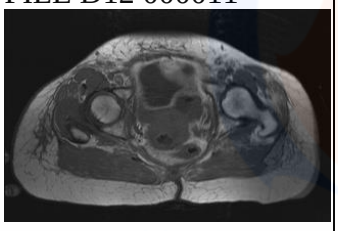
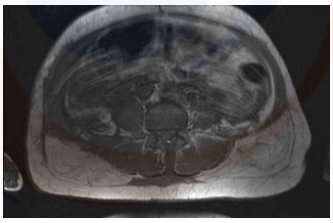
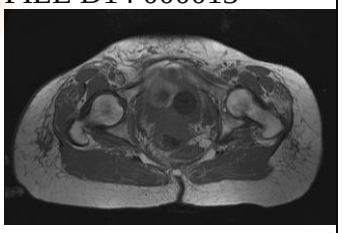
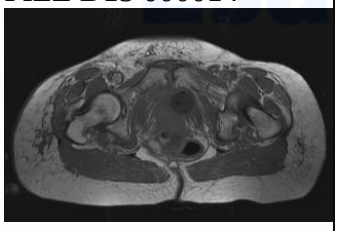
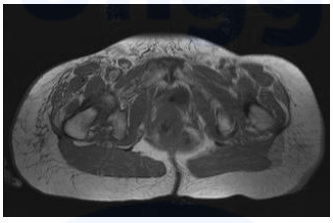
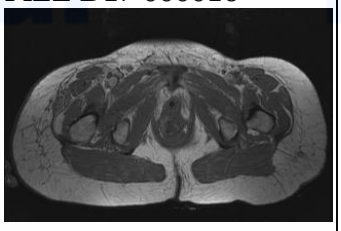
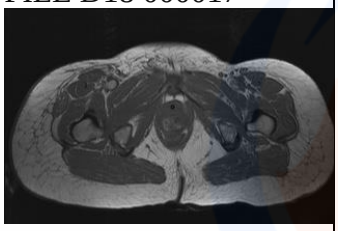
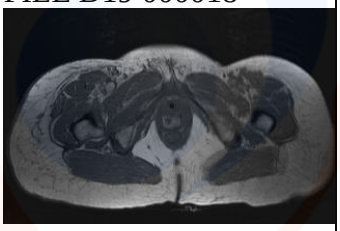
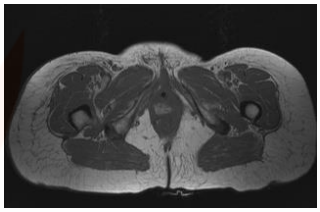
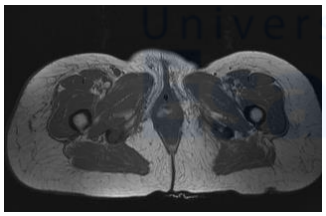
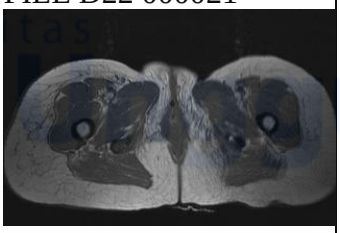
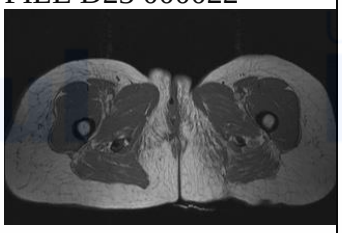
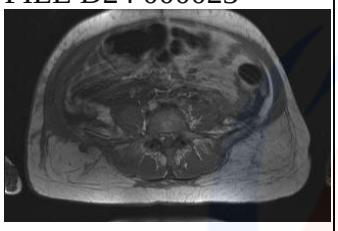
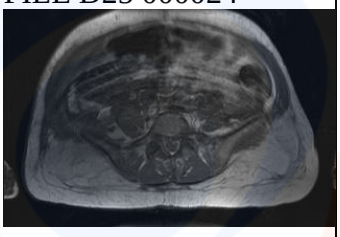
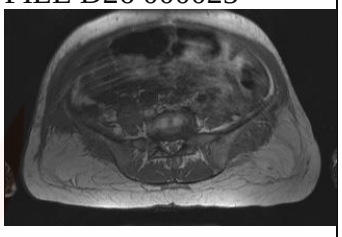
FILE A1 000000 	FILE A2 000001 	FILE A3 000002 
FILE A4 000003 	FILE A5 000004 	FILE B1 000000 
FILE B2 000001 	FILE B3 000002 	FILE B4 000003 
FILE B5 000004 	FILE B6 000005 	FILE B7 000006 

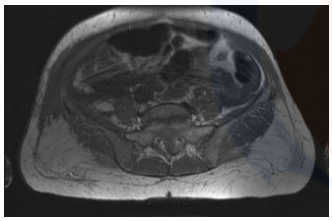
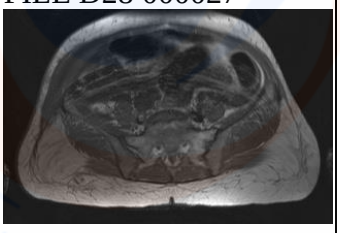
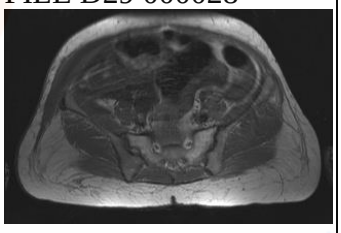
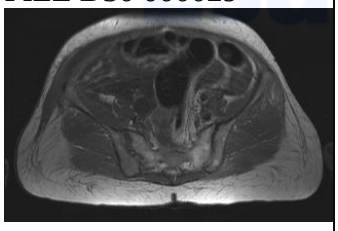
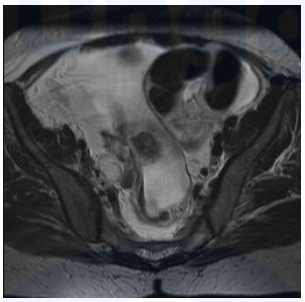
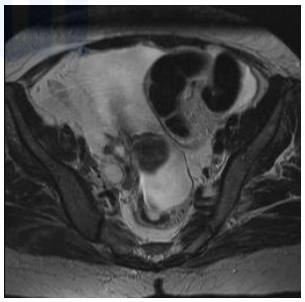
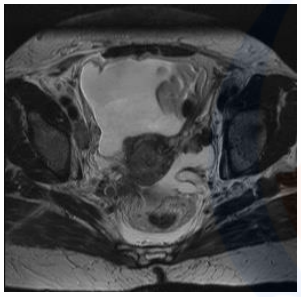
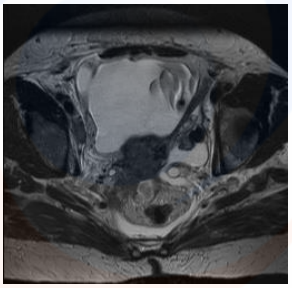
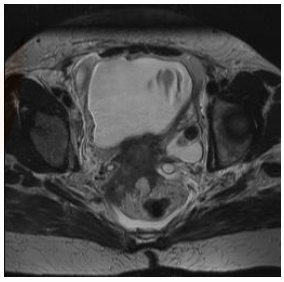
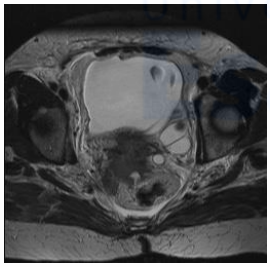
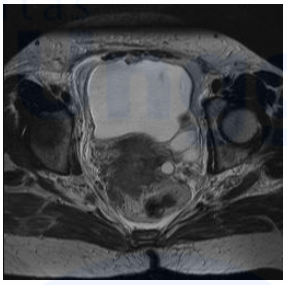
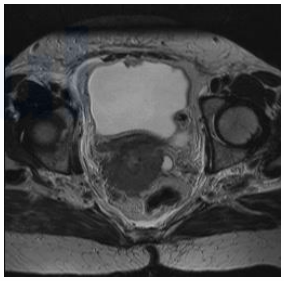
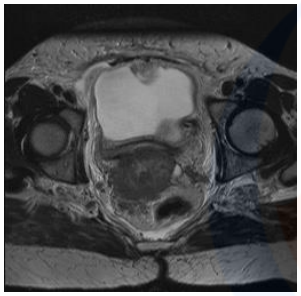
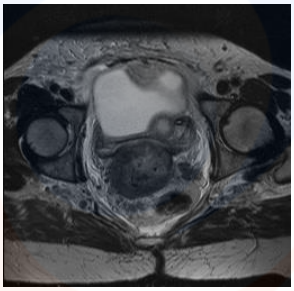
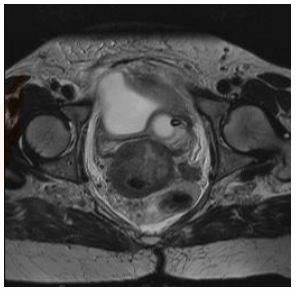
FILE B8 000007 	FILE B9 000008 	FILE B10 000009 
FILE B11 000010 	FILE B12 000011 	FILE B13 000012 
FILE B14 000013 	FILE B15 000014 	FILE B16 000015 
FILE B17 000016 	FILE B18 000017 	FILE B19 000018 
FILE B20 000019 	FILE B21 000020 	FILE B22 000021 

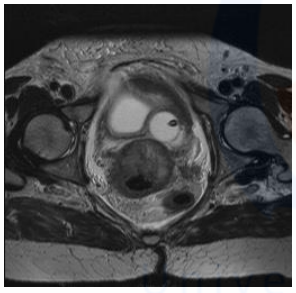
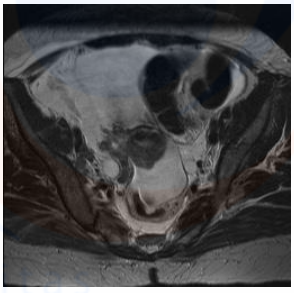
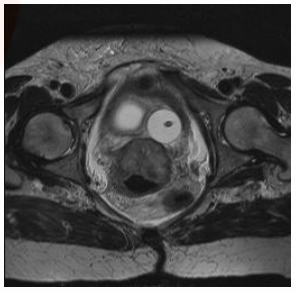
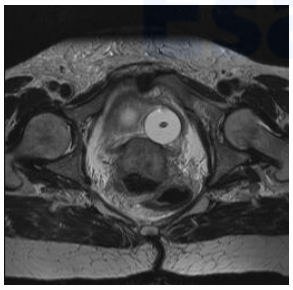
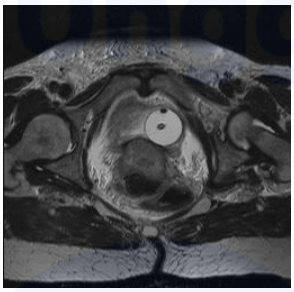
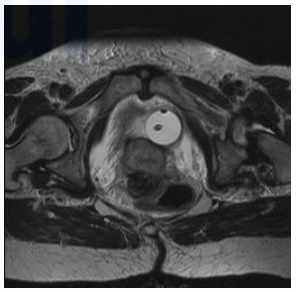
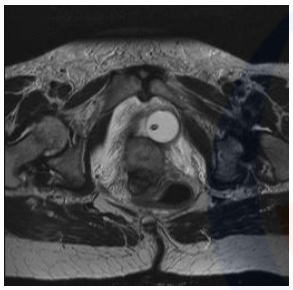
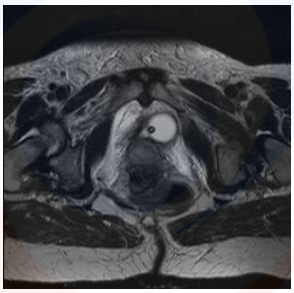
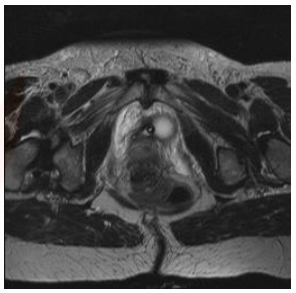
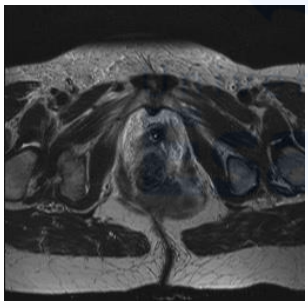
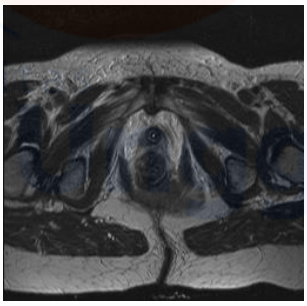
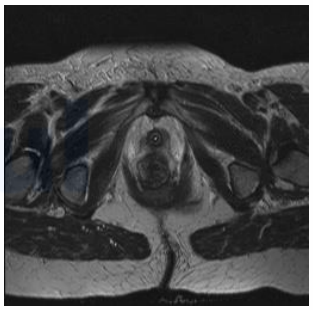
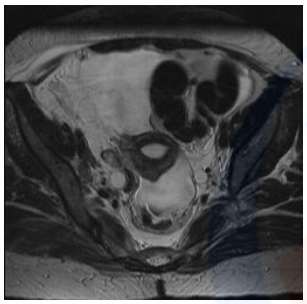
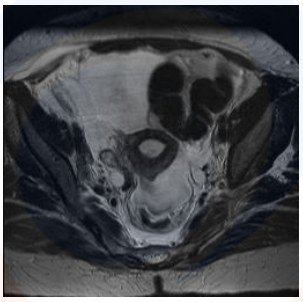
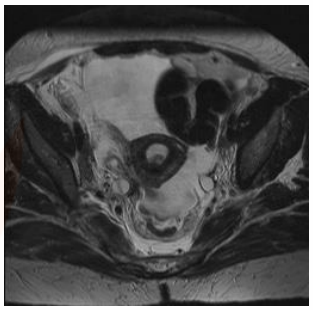
FILE B23 000022 	FILE B24 000023 	FILE B25 000024 
FILE B26 000025 	FILE B27 000026 	FILE B28 000027 
FILE B29 000028 	FILE B30 000029 	FILE C01 000000 
FILE C02 000001 	FILE C03 000002 	FILE C04 000003 

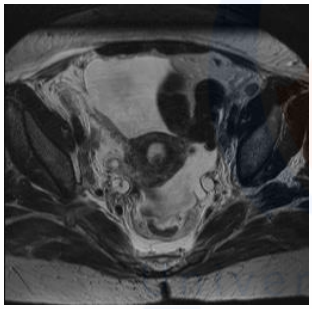
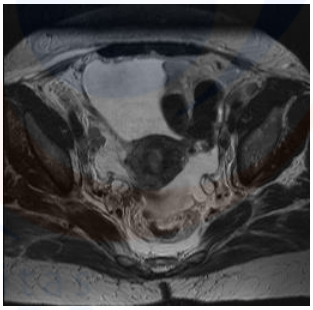
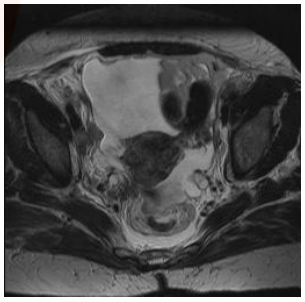
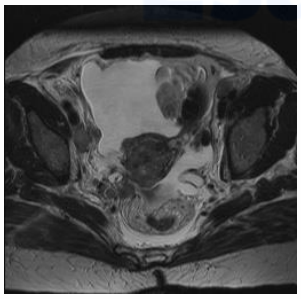
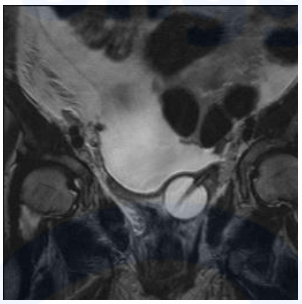
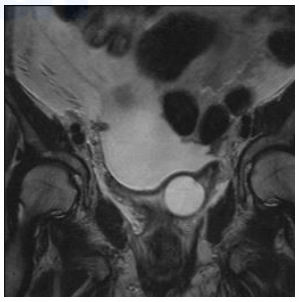
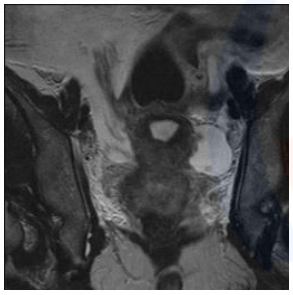
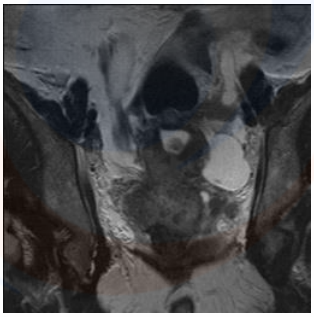
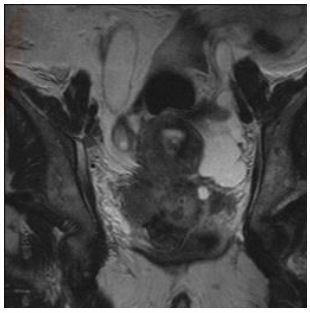
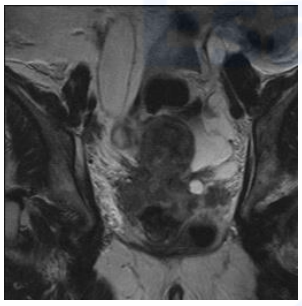
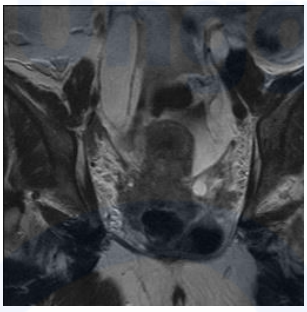
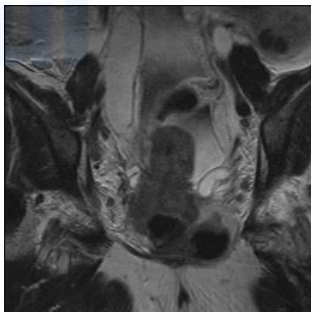
FILE C05 000004 	FILE C06 000005 	FILE C07 000006 
FILE C08 000007 	FILE C09 000008 	FILE C10 000009 
FILE C11 000010 	FILE C12 000011 	FILE C13 000012 
FILE C14 000013 	FILE C15 000014 	FILE C16 000015 

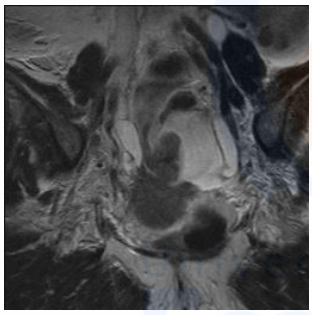
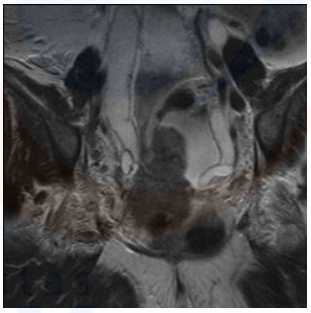
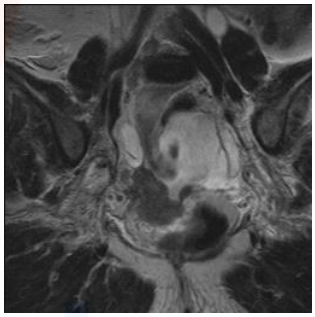
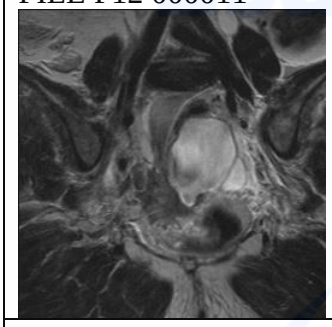
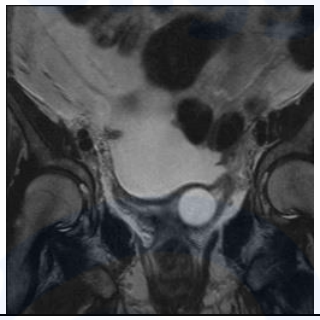
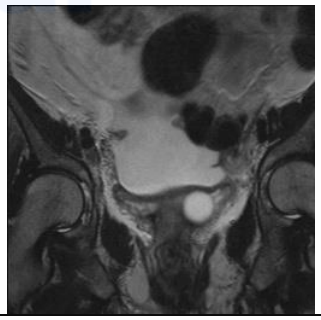
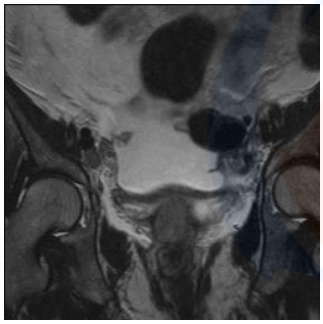
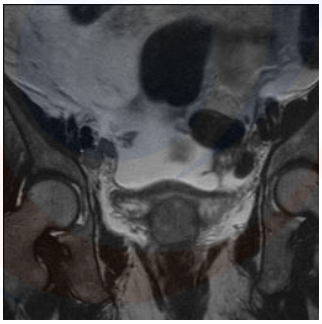
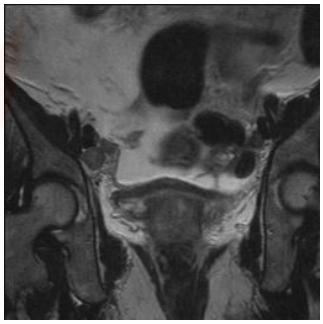
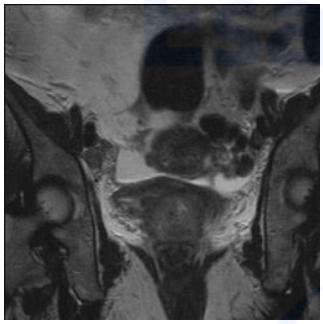
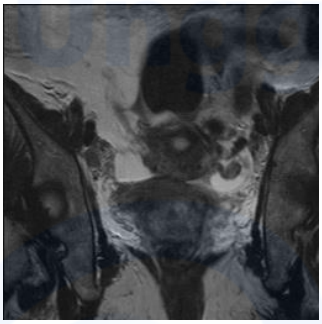
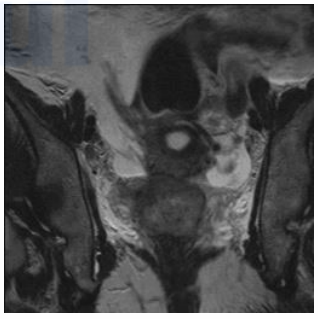
FILE C17 000016 	FILE C18 000017 	FILE C19 000018 
FILE C20 000019 	FILE D01 000000 	FILE D02 000001 
FILE D03 000002 	FILE D04 000003 	FILE D05 000004 
FILE D06 000005 	FILE D07 000006 	FILE D08 000007 
FILE D09 000008 	FILE D10 000009 	FILE D11 000010 

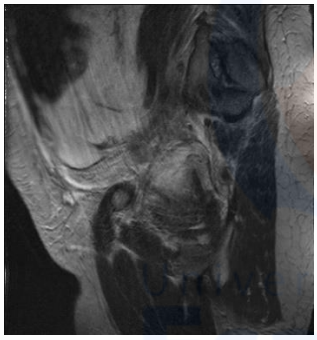
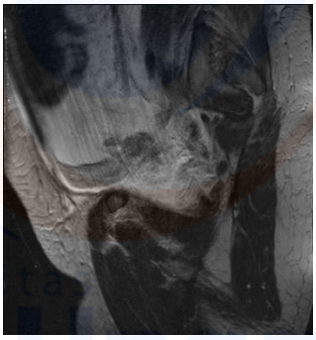
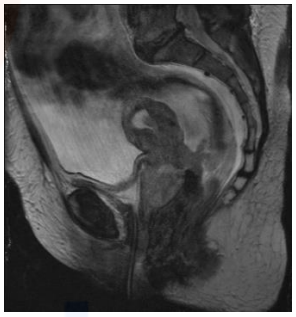
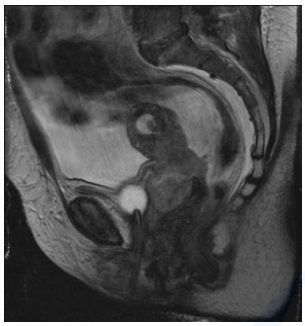
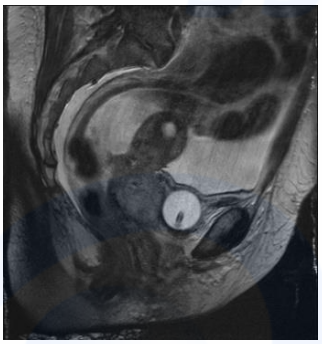
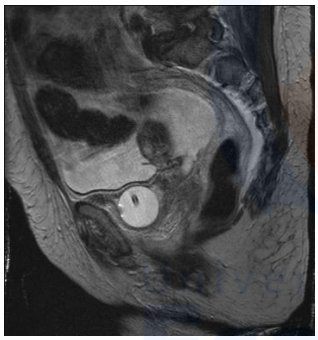
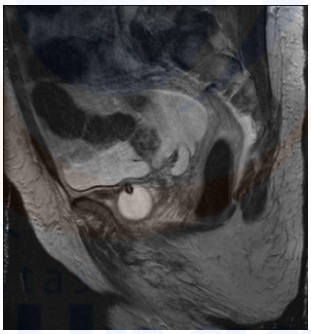
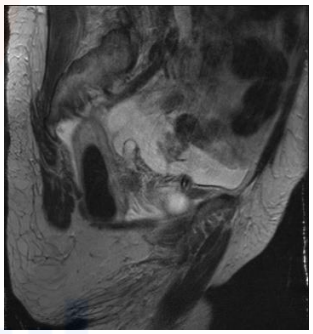
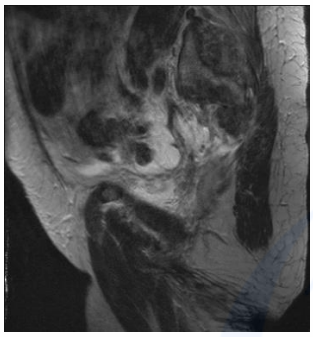
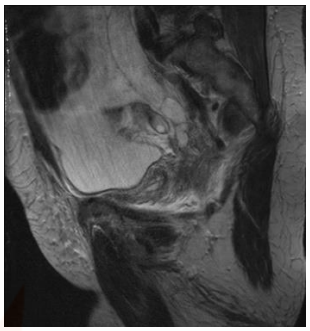
FILE D12 000011 	FILE D13 000012 	FILE D14 000013 
FILE D15 000014 	FILE D16 000015 	FILE D17 000016 
FILE D18 000017 	FILE D19 000018 	FILE D20 000019 
FILE D21 000020 	FILE D22 000021 	FILE D23 000022 
FILE D24 000023 	FILE D25 000024 	FILE D26 000025 

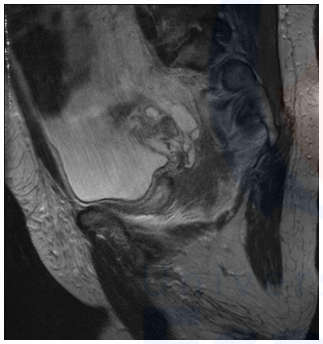
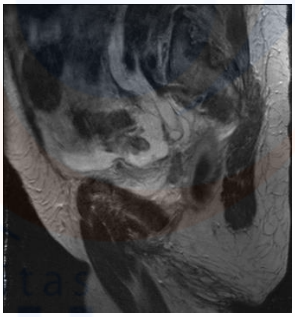
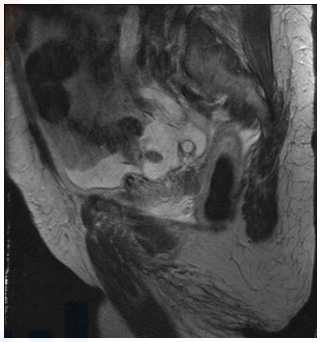
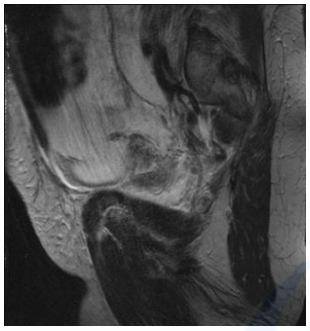
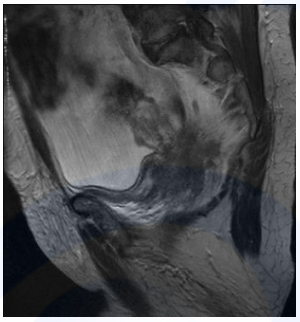
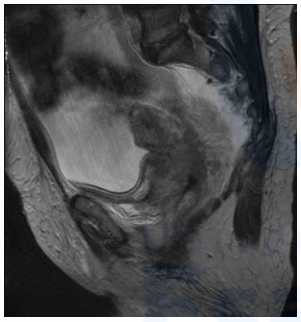
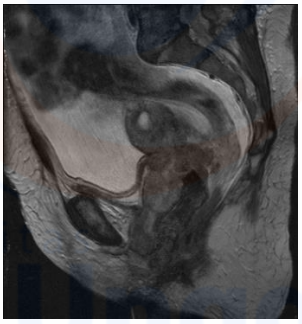
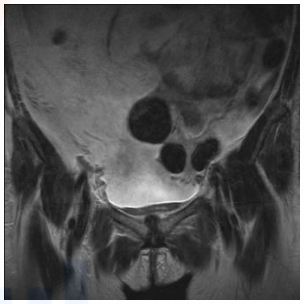
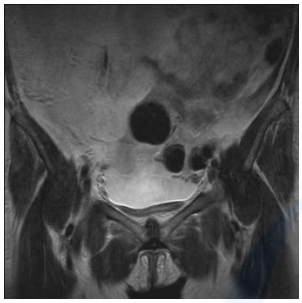
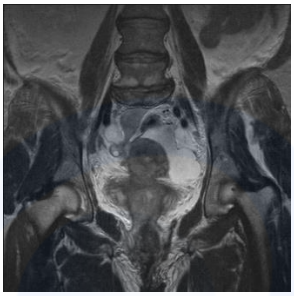
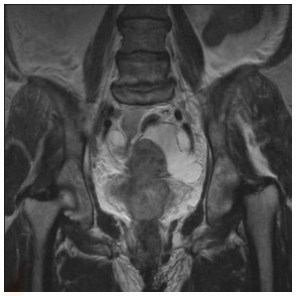
FILE D27 000026 	FILE D28 000027 	FILE D29 000028 
FILE D30 000029 	FILE E01 000000 	FILE E02 000001 
FILE E03 000002 	FILE E04 000003 	FILE E05 000004 
FILE E06 000005 	FILE E07 000006 	FILE E08 000007 
FILE E09 000008 	FILE E10 000009 	FILE E11 000010 

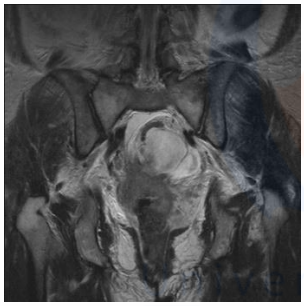
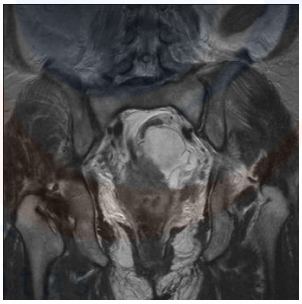
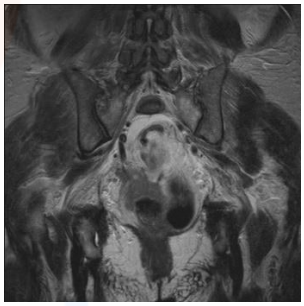
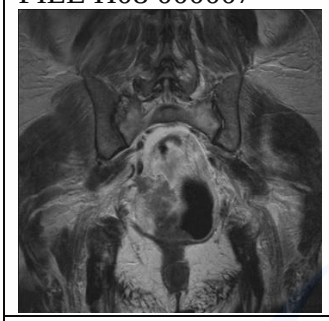
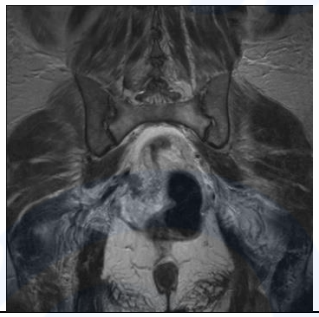
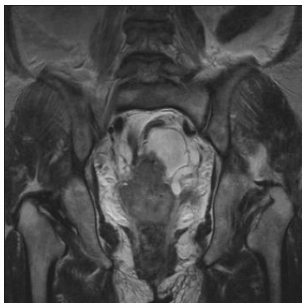
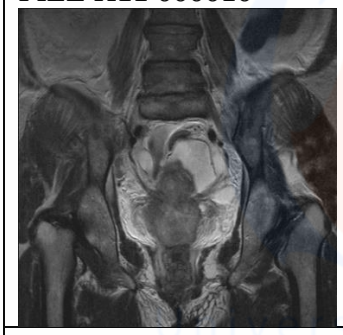
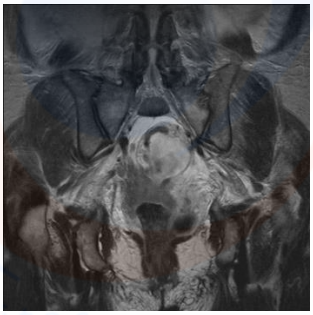
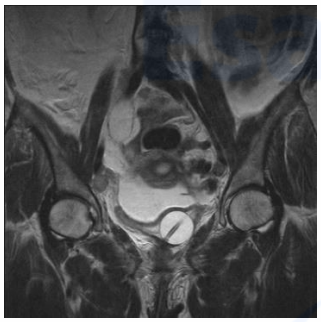
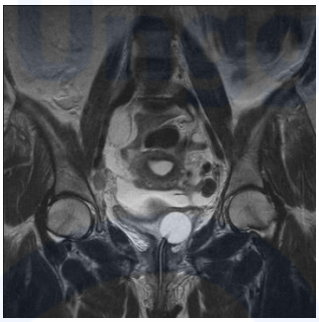
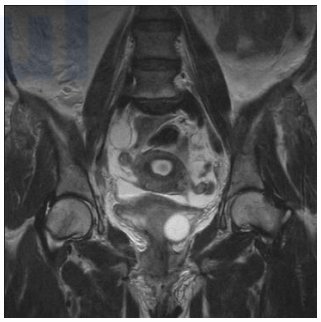
FILE E12 000011 	FILE E13 000012 	FILE E14 000013 
FILE E15 000014 	FILE E16 000015 	FILE E17 000016 
FILE E18 000017 	FILE E19 000018 	FILE E20 000019 
FILE E21 000020 	FILE E22 000021 	FILE E23 000022 
FILE E24 000023 	FILE E25 000024 	FILE E26 000025 

FILE E27 000026 	FILE E28 000027 	FILE E29 000028 
FILE E30 000029 	FILE F01 000000 	FILE F02 000001 
FILE F03 000002 	FILE F04 000003 	FILE F05 000004 
FILE F06 000005 	FILE F07 000006 	FILE F08 000007 

FILE F09 000008 	FILE F10 000009 	FILE F11 000010 
FILE F12 000011 	FILE F13 000012 	FILE F14 000013 
FILE F15 000014 	FILE F16 000015 	FILE F17 000016 
FILE F18 000017 	FILE F19 000018 	FILE F20 000019 

FILE G01 000000 	FILE G02 000001 	FILE G03 000002 
FILE G04 000003 	FILE G05 000004 	FILE G06 000005 
FILE G07 000006 	FILE G08 000007 	FILE G09 000008 
FILE G10 000009 	FILE G11 000010 	FILE G12 000011 

FILE G13 000012 	FILE G14 000013 	FILE G15 000014 
FILE G16 000015 	FILE G17 000016 	FILE G18 000017 
FILE G19 000018 	FILE G20 000019 	FILE H01 000000 
FILE H02 000001 	FILE H03 000002 	FILE H04 000003 

FILE H05 000004 	FILE H06 000005 	FILE H07 000006 
FILE H08 000007 	FILE H09 000008 	FILE H10 000009 
FILE H11 000010 	FILE H12 000011 	FILE H13 000012 
FILE H14 000013 	FILE H15 000014 	FILE H16 000015 

2. Source Code
   a. Build Data Set

```
from main import config
from imutils import paths
import random
import shutil
import os

imagePaths =
list(paths.list_images(config.ORIG_INPUT_DATASET))
random.seed(42)
random.shuffle(imagePaths)


i = int(len(imagePaths) * config.TRAIN_SPLIT)
trainPaths = imagePaths[:i]
testPaths = imagePaths[i:]

i = int(len(trainPaths) * config.VAL_SPLIT)
valPaths = trainPaths[:i]
trainPaths = trainPaths[i:]

datasets = [
    ("training", trainPaths, config.TRAIN_PATH),
    ("validation", valPaths, config.VAL_PATH),
    ("testing", testPaths, config.TEST_PATH)
]

for (dType, imagePaths, baseOutput) in datasets:

    print("[INFO] building '{}' split".format(dType))


    if not os.path.exists(baseOutput):
            print("[INFO] 'creating {}' directory".format(baseOutput))
            os.makedirs(baseOutput)


    for inputPath in imagePaths:

            filename = inputPath.split(os.path.sep)[-1]
            label = filename[-5:-4]
```

```
                labelPath = os.path.sep.join([baseOutput, label])


                if not os.path.exists(labelPath):
                        print("[INFO] 'creating {}'
directory".format(labelPath))
                        os.makedirs(labelPath)

                p = os.path.sep.join([labelPath, filename])
                shutil.copy2(inputPath, p)
```

b. Config

```
import os

ORIG_INPUT_DATASET = "datasets/orig"


BASE_PATH = "datasets/idc"


TRAIN_PATH = os.path.sep.join([BASE_PATH, "training"])
VAL_PATH = os.path.sep.join([BASE_PATH, "validation"])
TEST_PATH = os.path.sep.join([BASE_PATH, "testing"])

TRAIN_SPLIT = 0.8


VAL_SPLIT = 0.1
```

c. Model

```
from keras.models import Sequential
from keras.layers.normalization import BatchNormalization
from keras.layers.convolutional import SeparableConv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation
from keras.layers.core import Flatten
from keras.layers.core import Dropout
from keras.layers.core import Dense
from keras import backend as K

class CancerNet:
    @staticmethod
    def build(width, height, depth, classes):
```

```
        model = Sequential()
        inputShape = (height, width, depth)
        chanDim = -1

        if K.image_data_format() == "channels_first":
            inputShape = (depth, height, width)
            chanDim = 1

        model.add(SeparableConv2D(32, (3, 3), padding="same",
            input_shape=inputShape))
        model.add(Activation("relu"))
        model.add(BatchNormalization(axis=chanDim))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Dropout(0.2))

        model.add(SeparableConv2D(64, (3, 3),
padding="same"))
        model.add(Activation("relu"))
        model.add(BatchNormalization(axis=chanDim))
        model.add(SeparableConv2D(64, (3, 3),
padding="same"))
        model.add(Activation("relu"))
        model.add(BatchNormalization(axis=chanDim))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Dropout(0.2))

        model.add(SeparableConv2D(128, (3, 3),
padding="same"))
        model.add(Activation("relu"))
        model.add(BatchNormalization(axis=chanDim))
        model.add(SeparableConv2D(128, (3, 3),
padding="same"))
        model.add(Activation("relu"))
        model.add(BatchNormalization(axis=chanDim))
        model.add(SeparableConv2D(128, (3, 3),
padding="same"))
        model.add(Activation("relu"))
        model.add(BatchNormalization(axis=chanDim))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Dropout(0.2))

        model.add(Flatten())
        model.add(Dense(256))
        model.add(Activation("relu"))
        model.add(BatchNormalization())
```

```
model.add(Dropout(0.2))

model.add(Dense(256))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.2))


model.add(Dense(256))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.2))


model.add(Dense(256))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.2))


model.add(Dense(256))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.2))


model.add(Dense(classes))
model.add(Activation("softmax"))
model.summary()
model.compile(loss='mse', optimizer='rmsprop')

return model
```

d. Train

```
# USAGE
# python train_model.py

# set the matplotlib backend so figures can be saved in the
background
import matplotlib
matplotlib.use("Agg")


from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import LearningRateScheduler
```

```
from keras.optimizers import Adagrad
from keras.utils.vis_utils import plot_model
from keras.utils import np_utils
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from main.cancernet import CancerNet
from main import config
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import argparse
import os


ap = argparse.ArgumentParser()
ap.add_argument("-p", "--plot", type=str, default="plot.png",
    help="path to output loss/accuracy plot")
args = vars(ap.parse_args())

NUM_EPOCHS = 30
INIT_LR = 1e-2
BS = 32

trainPaths = list(paths.list_images(config.TRAIN_PATH))
totalTrain = len(trainPaths)
totalVal = len(list(paths.list_images(config.VAL_PATH)))
totalTest = len(list(paths.list_images(config.TEST_PATH)))


trainLabels = [int(p.split(os.path.sep)[-2]) for p in trainPaths]
trainLabels = np_utils.to_categorical(trainLabels)
classTotals = trainLabels.sum(axis=0)
classWeight = classTotals.max() / classTotals


trainAug = ImageDataGenerator(
    rescale=1 / 255.0,
    rotation_range=20,
    zoom_range=0.05,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.05,
    horizontal_flip=True,
    vertical_flip=True,
    fill_mode="nearest")
```

```
valAug = ImageDataGenerator(rescale=1 / 255.0)


trainGen = trainAug.flow_from_directory(
    config.TRAIN_PATH,
    class_mode="categorical",
    target_size=(48, 48),
    color_mode="rgb",
    shuffle=True,
    batch_size=BS)


valGen = valAug.flow_from_directory(
    config.VAL_PATH,
    class_mode="categorical",
    target_size=(48, 48),
    color_mode="rgb",
    shuffle=False,
    batch_size=BS)


testGen = valAug.flow_from_directory(
    config.TEST_PATH,
    class_mode="categorical",
    target_size=(48, 48),
    color_mode="rgb",
    shuffle=False,
    batch_size=BS)


model = CancerNet3.build(width=48, height=48, depth=3,
    classes=2)
opt = Adagrad(lr=INIT_LR, decay=INIT_LR / NUM_EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
    metrics=["accuracy"])


H = model.fit_generator(
    trainGen,
    steps_per_epoch=totalTrain // BS,
    validation_data=valGen,
    validation_steps=totalVal // BS,
    class_weight=classWeight,
```

```
    epochs=NUM_EPOCHS)

print("[INFO] evaluating network...")
testGen.reset()
predIdxs = model.predict_generator(testGen,
    steps=(totalTest // BS) + 1)

predIdxs = np.argmax(predIdxs, axis=1)


print(classification_report(testGen.classes, predIdxs,
    target_names=testGen.class_indices.keys()))


cm = confusion_matrix(testGen.classes, predIdxs)
total = sum(sum(cm))
acc = (cm[0, 0] + cm[1, 1]) / total
sensitivity = cm[0, 0] / (cm[0, 0] + cm[0, 1])
specificity = cm[1, 1] / (cm[1, 0] + cm[1, 1])


print(cm)
print("acc: {:.4f}".format(acc))
print("sensitivity: {:.4f}".format(sensitivity))
print("specificity: {:.4f}".format(specificity))


N = NUM_EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["acc"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_acc"], label="val_acc")
plt.title("Training Loss and Accuracy on Dataset")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig(args["plot"])


plot_model(model, to_file='vgg.png')
```