A GAN based Approach Medical Image Segmentation

In recent years, with the rapid development of deep learning theory and hardware, deep learning is more and more widely used in the field of medical image processing, and has greatly improved the performance of traditional methods. Medical image segmentation still is an important research field in the area of computer vision and machine learning. In deep learning specially Generative adversarial network (GAN) has revolutionized the computer vision domain with its unique architecture, that the model itself does evaluation using discriminator. The basic structure of GAN is that there is a deep learning based generator that generates the output and there is a discriminator that compares the generated output with real image and computes the loss of the model. The U-Net based GAN has the tendency to effectively and efficiently do segmentation from the complex structures like medical imaging. The GAN architecture is shown in the figure 1.

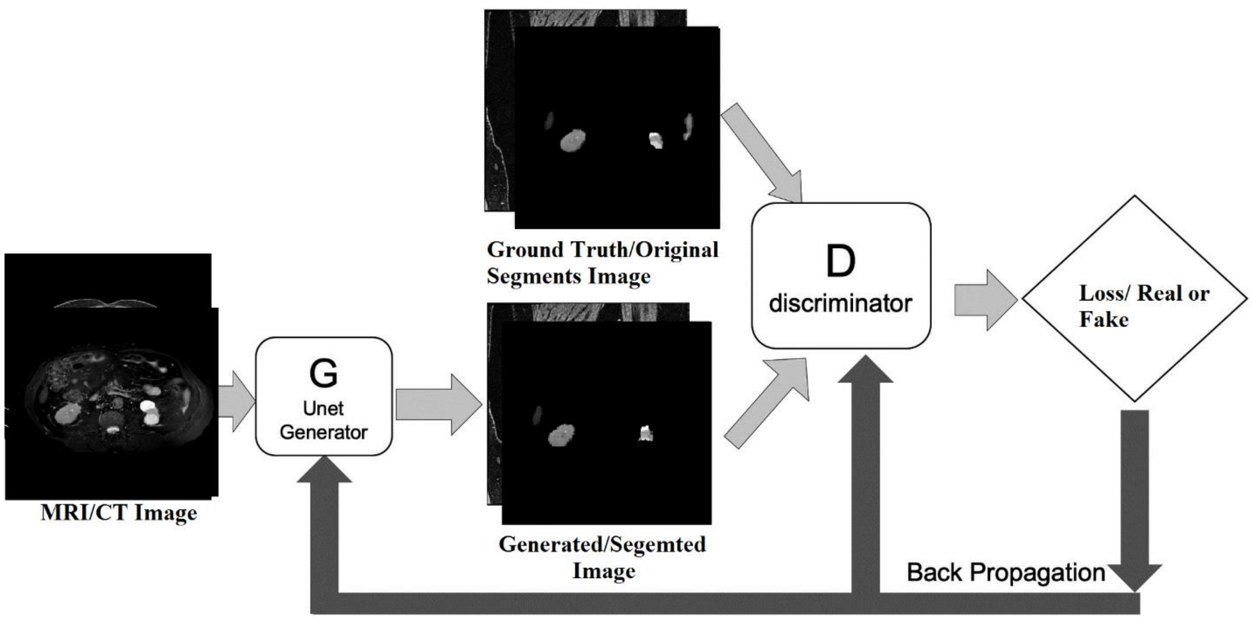
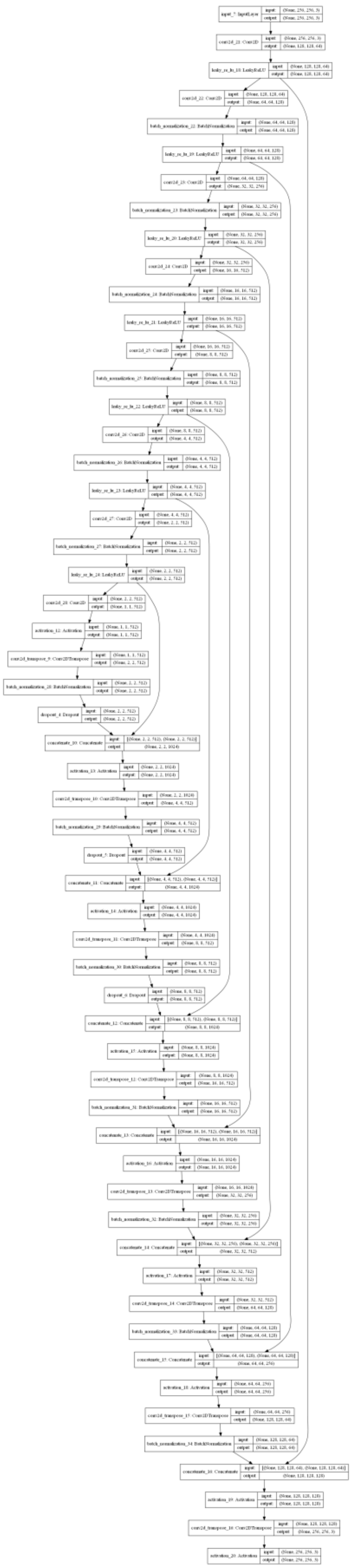


Figure 1. GAN Architecture

The U-Net based GAN consists of of Auto-Encoder Generators and neural network based discriminator. The detailed architecture of Generator is shown in the details below

Generator Architecture:

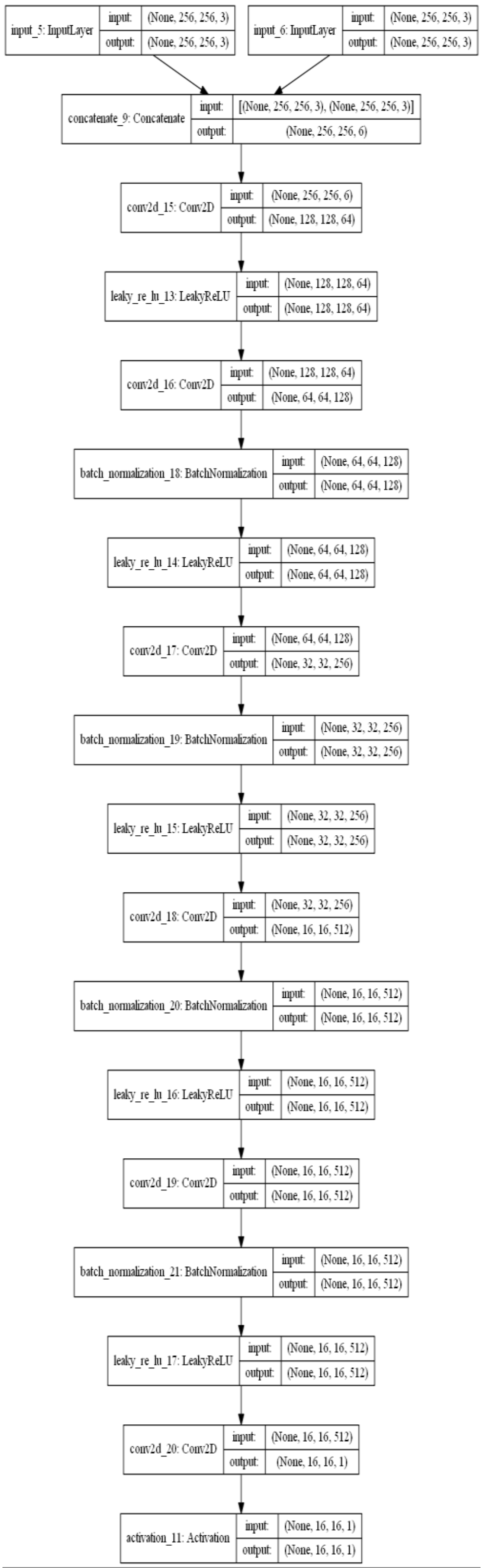
Following is the generator architecture



Model: "Generator"

Layer (type)	Output Shape	Param #	Connected to
input_13 (InputLayer)	(None, 256, 256, 3)	0	
conv2d_36 (Conv2D)	(None, 128, 128, 64)	3136	input_13[0][0]
leaky_re_lu_30 (LeakyReLU)	(None, 128, 128, 64)	0	conv2d_36[0][0]
conv2d_37 (Conv2D)	(None, 64, 64, 128)	131200	leaky_re_lu_30[0][0]
batch_normalization_39 (BatchNo	(None, 64, 64, 128)	512	conv2d_37[0][0]
leaky_re_lu_31 (LeakyReLU)	(None, 64, 64, 128)	0	batch_normalization_39[0][0]
conv2d_38 (Conv2D)	(None, 32, 32, 256)	524544	leaky_re_lu_31[0][0]
batch_normalization_40 (BatchNo	(None, 32, 32, 256)	1024	conv2d_38[0][0]
leaky_re_lu_32 (LeakyReLU)	(None, 32, 32, 256)	0	batch_normalization_40[0][0]
conv2d_39 (Conv2D)	(None, 16, 16, 512)	2097664	leaky_re_lu_32[0][0]
batch_normalization_41 (BatchNo	(None, 16, 16, 512)	2048	conv2d_39[0][0]
leaky_re_lu_33 (LeakyReLU)	(None, 16, 16, 512)	0	batch_normalization_41[0][0]
conv2d_40 (Conv2D)	(None, 8, 8, 512)	4194816	leaky_re_lu_33[0][0]
batch_normalization_42 (BatchNo	(None, 8, 8, 512)	2048	conv2d_40[0][0]
leaky_re_lu_34 (LeakyReLU)	(None, 8, 8, 512)	0	batch_normalization_42[0][0]
conv2d_41 (Conv2D)	(None, 4, 4, 512)	4194816	leaky_re_lu_34[0][0]
batch_normalization_43 (BatchNo	(None, 4, 4, 512)	2048	conv2d_41[0][0]
leaky_re_lu_35 (LeakyReLU)	(None, 4, 4, 512)	0	batch_normalization_43[0][0]
conv2d_42 (Conv2D)	(None, 2, 2, 512)	4194816	leaky_re_lu_35[0][0]
batch_normalization_44 (BatchNo	(None, 2, 2, 512)	2048	conv2d_42[0][0]
leaky_re_lu_36 (LeakyReLU)	(None, 2, 2, 512)	0	batch_normalization_44[0][0]
conv2d_43 (Conv2D)	(None, 1, 1, 512)	4194816	leaky_re_lu_36[0][0]
activation_22 (Activation)	(None, 1, 1, 512)	0	conv2d_43[0][0]
conv2d_transpose_17 (Conv2DTran	(None, 2, 2, 512)	4194816	activation_22[0][0]
batch_normalization_45 (BatchNo	(None, 2, 2, 512)	2048	conv2d_transpose_17[0][0]
dropout_7 (Dropout)	(None, 2, 2, 512)	0	batch_normalization_45[0][0]
concatenate_19 (Concatenate)	(None, 2, 2, 1024)	0	dropout_7[0][0] leaky_re_lu_36[0][0]
activation_23 (Activation)	(None, 2, 2, 1024)	0	concatenate_19[0][0]
conv2d_transpose_18 (Conv2DTran	(None, 4, 4, 512)	8389120	activation_23[0][0]
batch_normalization_46 (BatchNo	(None, 4, 4, 512)	2048	conv2d_transpose_18[0][0]
dropout_8 (Dropout)	(None, 4, 4, 512)	0	batch_normalization_46[0][0]
concatenate_20 (Concatenate)	(None, 4, 4, 1024)	0	dropout_8[0][0] leaky_re_lu_35[0][0]
activation_24 (Activation)	(None, 4, 4, 1024)	0	concatenate_20[0][0]
conv2d_transpose_19 (Conv2DTran	(None, 8, 8, 512)	8389120	activation_24[0][0]
batch_normalization_47 (BatchNo	(None, 8, 8, 512)	2048	conv2d_transpose_19[0][0]
dropout_9 (Dropout)	(None, 8, 8, 512)	0	batch_normalization_47[0][0]
concatenate_21 (Concatenate)	(None, 8, 8, 1024)	0	dropout_9[0][0] leaky_re_lu_34[0][0]
activation_25 (Activation)	(None, 8, 8, 1024)	0	concatenate_21[0][0]
conv2d_transpose_20 (Conv2DTran	(None, 16, 16, 512)	8389120	activation_25[0][0]
batch_normalization_48 (BatchNo	(None, 16, 16, 512)	2048	conv2d_transpose_20[0][0]
concatenate_22 (Concatenate)	(None, 16, 16, 1024)	0	batch_normalization_48[0][0] leaky_re_lu_33[0][0]
activation_26 (Activation)	(None, 16, 16, 1024)	0	concatenate_22[0][0]
conv2d_transpose_21 (Conv2DTran	(None, 32, 32, 256)	4194560	activation_26[0][0]
batch_normalization_49 (BatchNo	(None, 32, 32, 256)	1024	conv2d_transpose_21[0][0]
concatenate_23 (Concatenate)	(None, 32, 32, 512)	0	batch_normalization_49[0][0] leaky_re_lu_32[0][0]
activation_27 (Activation)	(None, 32, 32, 512)	0	concatenate_23[0][0]
conv2d_transpose_22 (Conv2DTran	(None, 64, 64, 128)	1048704	activation_27[0][0]
batch_normalization_50 (BatchNo	(None, 64, 64, 128)	512	conv2d_transpose_22[0][0]
concatenate_24 (Concatenate)	(None, 64, 64, 256)	0	batch_normalization_50[0][0] leaky_re_lu_31[0][0]
activation_28 (Activation)	(None, 64, 64, 256)	0	concatenate_24[0][0]
conv2d_transpose_23 (Conv2DTran	(None, 128, 128, 64)	262208	activation_28[0][0]
batch_normalization_51 (BatchNo	(None, 128, 128, 64)	256	conv2d_transpose_23[0][0]
concatenate_25 (Concatenate)	(None, 128, 128, 128	0	batch_normalization_51[0][0] leaky_re_lu_30[0][0]

Discriminator Architecture:



Model: "Discriminator"

Layer (type)	Output Shape	Param #	Connected to
input_11 (InputLayer)	(None, 256, 256, 3)	0	
input_12 (InputLayer)	(None, 256, 256, 3)	0	
concatenate_18 (Concatenate)	(None, 256, 256, 6)	0	input_11[0][0] input_12[0][0]
conv2d_30 (Conv2D)	(None, 128, 128, 64)	6208	concatenate_18[0][0]
leaky_re_lu_25 (LeakyReLU)	(None, 128, 128, 64)	0	conv2d_30[0][0]
conv2d_31 (Conv2D)	(None, 64, 64, 128)	131200	leaky_re_lu_25[0][0]
batch_normalization_35 (BatchNo	(None, 64, 64, 128)	512	conv2d_31[0][0]
leaky_re_lu_26 (LeakyReLU)	(None, 64, 64, 128)	0	batch_normalization_35[0][0]
conv2d_32 (Conv2D)	(None, 32, 32, 256)	524544	leaky_re_lu_26[0][0]
batch_normalization_36 (BatchNo	(None, 32, 32, 256)	1024	conv2d_32[0][0]
leaky_re_lu_27 (LeakyReLU)	(None, 32, 32, 256)	0	batch_normalization_36[0][0]
conv2d_33 (Conv2D)	(None, 16, 16, 512)	2097664	leaky_re_lu_27[0][0]
batch_normalization_37 (BatchNo	(None, 16, 16, 512)	2048	conv2d_33[0][0]
leaky_re_lu_28 (LeakyReLU)	(None, 16, 16, 512)	0	batch_normalization_37[0][0]
conv2d_34 (Conv2D)	(None, 16, 16, 512)	4194816	leaky_re_lu_28[0][0]
batch_normalization_38 (BatchNo	(None, 16, 16, 512)	2048	conv2d_34[0][0]
leaky_re_lu_29 (LeakyReLU)	(None, 16, 16, 512)	0	batch_normalization_38[0][0]
conv2d_35 (Conv2D)	(None, 16, 16, 1)	8193	leaky_re_lu_29[0][0]
activation_21 (Activation)	(None, 16, 16, 1)	0	conv2d_35[0][0]

Total params: 6,968,257
Trainable params: 6,965,441
Non-trainable params: 2,816

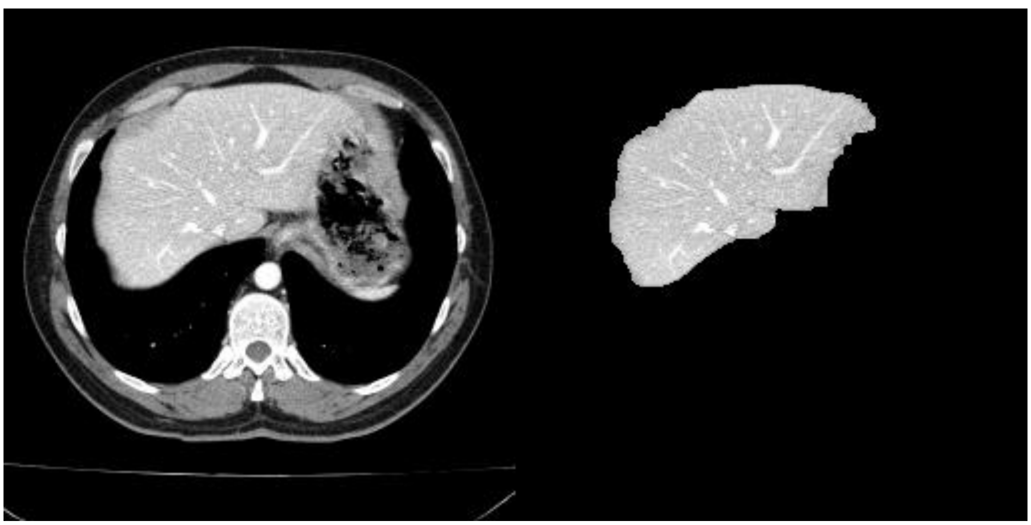
Materials and Method

Dataset:

The CHAOS Grand Challenge consist of Two databases Abdominal CT and MRI (T1 and T2 weighted). Each data set in these two databases corresponds to a series of DICOM images belonging to a single patient. The data sets are collected retrospectively and randomly from the PACS of DEU Hospital. There is no connection between the data sets obtained from CT and MR databases (i.e. they are acquired from different patients and not registered).

Preprocessing:

The dataset consists of DICOM (.dcm) image. In the preprocessing steps, the DICOM CT images were subject to file format conversion to portable network graphics (PNG). The PNG file format was chosen to preserve the image quality, as it is a lossless format. Now we have the masked image as a ground truth for training so we prepare dataset in such a way that we generate the concatenated image of both the original input and the segmented image side by side as showin in example below. Here in left side the original image and on the right hand side the segmented part extracted using ground truth is available.



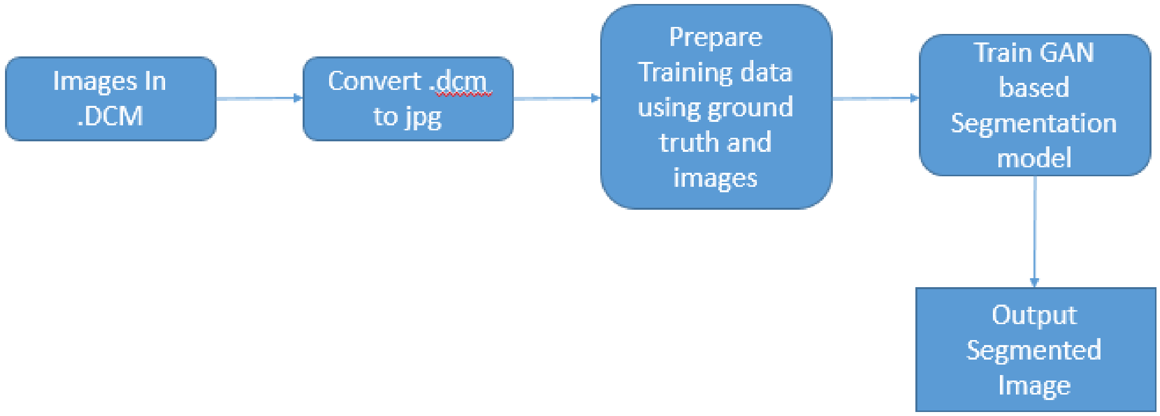
Training and Classification

Collectively we have 4144 samples of CT and MRI images out of which 3730 (2587 of CT and 1143 of MRI) images are used for training set and 287 sample images of CT and 127 sample images of MRI is taken for testing process. After successful preparation of dataset with concatenated images we feed this data to our patched based Pix2Pix GAN to train the model. Our model has the ability to train the model efficiently that it can do segmentation both MRI and CT images as both images have different kind of structure and ground truth images.

The main steps involved in the project are

1. The dataset of images is in Dicom (.dcm) format we first need to convert it to normal image format like jpeg or png. Using some function and prebuild libraries in python pydicom we have converted the images from .dcm to .jpg.

2. We have the masked image as a ground truth for training so we prepare dataset in such a way that we generate the concatenated image of both the original input and the segmented image side by side.
3. Feed the data to train the model.
4. Testing using trained model.



Tools and Technology used:

1. Python 3.6 or higher
2. Tensorflow
3. Keras.
4. Pydicom
5. Pillow
6. OpenCV

Results

To authenticate the validity of results we calculate Mean IOU between the ground truth and the predicted segmented result. The table 1 compares the calculated mean IOU using proposed method with other research works i.e. U-NET and SegNet. The final mean IoU scores were 76.14% using U-Net, 68.88% using SegNet and 85.75% using proposed method. The loss error based on binary cross entropy for U-Net, SegNet and proposed method were, respectively, 0.002, 0.053, 0.000. As illustrated in table 1, the predicted qualitative results using proposed method were closer to the ground truth than those obtained using SegNet and U-Net. Moreover the results predicted in pictorial form using test sample

Method	Accuracy (IOU)	Loss (Binary Cross Entropy
U-NET	76.14	0.002
SegNet	68.88	0.053
Proposed Methodology (GAN)	**85.75**	**0.000**

Table 1. Results Comparison

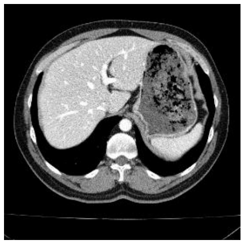
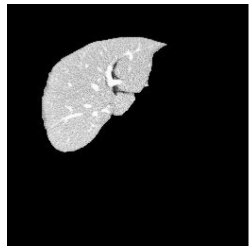
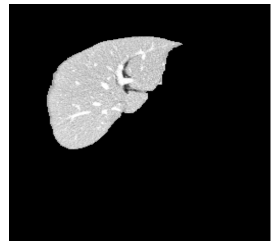
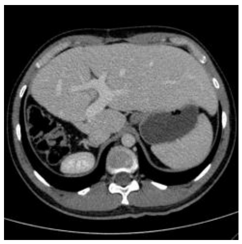
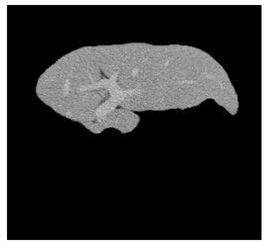
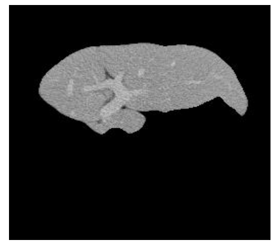
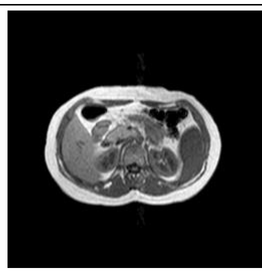
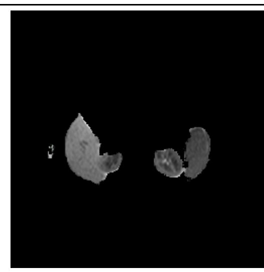
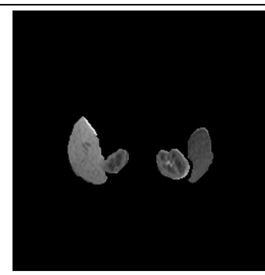
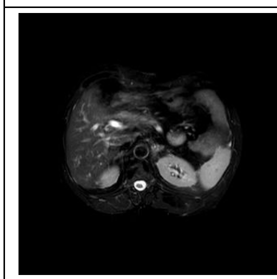
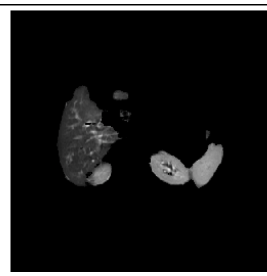
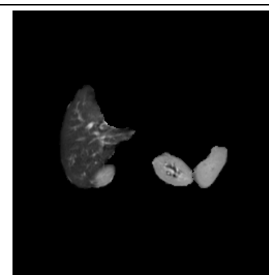
	Input	Predicted	Ground Truth
CT			
			
MRI			
			

Table 2. Visual result comparison with ground truth