LIVER TUMOUR SEGMENTATION FROM CT IMAGES USING CONVOLUTIONAL NEURAL NETWORK

CREATIVE AND INNOVATIVE PROJECT

Submitted by

Subhiksha Sai Subramanian 2019103067
Vasudha E 2019103073
Sneha S 2019103583

in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

COLLEGE OF ENGINEERING, GUINDY

ANNA UNIVERSITY, CHENNAI 600 025.

JUNE 2022

ACKNOWLEDGEMENT

First and foremost, we would like to express our sincere gratitude to our project guide, **Dr. Lalitha Devi K**, Teaching Fellow, Department of Computer Science and Engineering, College of Engineering Guindy, Chennai for her constant source of inspiration. We thank her for the continuous support and guidance which was instrumental in taking the project to successful completion.

We are grateful to **Dr.S.Valli** , Professor and Head, Department of Computer Science and Engineering, College of Engineering Guindy, Chennai for her support and for providing necessary facilities to carry out for our project.

We would also like to thank our friends and family for their encouragement and continued support. We would also like to thank the Almighty for giving us the moral strength to accomplish our task.

SUBHIKSHA SAI SUBRAMANIAN **VASUDHA E** **SNEHA S**

TABLE OF CONTENTS

ABSTRACT

The liver is an important organ, helping humans eliminate waste, digest food, preserve vitamins and minerals, and energy materials. Liver tumors are abnormal masses of tissues when cells reproduce at an increased rate. Both benign and malignant tumors form in the liver. Liver Cancer is one of the main causes of human death. Physicians prefer automatic segmentation methods, owing to the large number of slices in computed tomography sequence. Accurate automatic segmentation methods, however, are difficult to obtain, due to the noise in scan sequence, similarities in pixel intensity between tumors and surrounding tissues, and differences in size, position, and shape of tumors from one patient to another.

1.INTRODUCTION

1.1. OBJECTIVES

Our objective here is to create an automatic segmentation model using deep learning algorithms for separating the liver from the CT scans, and to detect and segment the tumors present in the liver.

1.2. PROBLEM STATEMENT

Liver Tumor is one of the main causes of human death. Detection of liver tumors early using Computed Tomography (CT) could prevent millions of patient's deaths. Reading a CT scan is a time-consuming process that requires specialized radiologists and can still be prone to errors. Therefore, the need to read, detect and evaluate CT scans for liver tumors automatically, quickly, and accurately has significantly increased in the recent years.

1.3. CHALLENGES IN THE SYSTEM:

- As the shape and size of tumours vary from person to person, it is difficult to to predict tumours using a machine learning algorithm.
- Each CT scan consists of multiple slices and volumes which makes it computationally intensive to pre-process.

1.4. SIGNIFICANCE OF UNET ARCHITECTURE:

- U-NET architecture can be used for image localization, which helps in predicting the image pixel by pixel. It also achieves good performance on very different biomedical segmentation applications.
- It can be used for any reasonable image masking task
- High accuracy is given proper training, adequate dataset, and training time
- This architecture is input image size agnostic since it does not contain fully connected layers

1.5. SCOPE OF THE MODEL:

This method of tumor segmentation can eliminate the tedious work of radiologists or hepatologists in segmenting the tumor manually. Although this technology is still in it's initial phase, it can be used as a reference to aid doctors until better technology is developed.

2. LITERATURE SURVEY

Before deep learning was widely introduced, many methods have been proposed for the liver segmentation of abdominal CT images based on graphics, morphology, and traditional machine learning. Among automatic liver segmentation algorithms, model-based methods have proved to be the most effective one, where prior anatomical knowledge of the target organ is incorporated into the segmentation process. Shi et al. [2] introduced a novel framework for accurate and robust liver segmentation in portal phase of abdominal CT images based on active shape models (ASMs). The highlight was a new multilevel local region-based SSC (MLR-SSC) to increase the flexibility of shape prior models and capture the detailed local shape information more faithfully. During the past few decades, they mainly focused on developing algorithms such as level set, watershed, statistical shape model, region growing, active contour model, threshold processing, graph cuts and traditional machine learning methods that require manually extract tumor features. Massoptier et al. [3] proposed an automatic liver tumor segmentation algorithm based

on statistical shape model, which used the active contour technique of gradient vector flow to obtain smooth and natural liver tumor segmentation results without the need of interaction. Wong et al. [4] proposed a semi-automatic tumor segmentation method based on region growing method. They first sketch the region of interest of tumor manually and calculate the seed point and feature vector in it, then regional growing algorithm was performed to mark the tumor voxels. Incorporating knowledge-based constraints into the growing method ensures the segmented tumor size and shape are within a reasonable range. Zhou et al. [5] proposed a unified level set method (LSM) for liver tumor segmentation. They used local pixel intensity clustering combined with hidden Markov random field to construct a unified LSM. Then, regional information and edge information were used to acquire the tumor contour, so that the problem of edge leakage can be solved.

With the rapid development of deep learning and its blossom in the field of computer vision, the direction of research in the field of medical image segmentation has also begun to transform to deep learning. In the field of liver image segmentation, more and more methods based on deep learning have also appeared. In general, liver and tumor extraction approaches can be classified into three categories: manual segmentation, semi-automated segmentation, and automated segmentation. Manual segmentation is a subjective, poorly reproducible, and time-consuming approach. Later, Wong et al. [6] proposed a semi-automatic tumor segmentation method based on region growing method. They first sketch the region of interest of tumor manually and calculate the seed point and feature vector in it, then regional growing algorithm was performed to mark the tumor voxels. Incorporating knowledge-based constraints into the growing method ensures the segmented tumor size and shape are within a reasonable range.

Besides, a mounting interest continues for achieving automatic segmentation via deep-learning techniques. Compared with traditional methods, the convolutional neural network (CNN) has been proven effective in processing images. Especially the fully convolutional neural network (FCN) has achieved excellent results in medical image identification, classification, and segmentation. Lately, Dou et al. [7] presented a novel and efficient 3D fully convolutional network equipped with a 3D deep supervision mechanism for 3D image segmentation. Christ et al. [9] proposed a method based on cascading two fully convolutional neural networks (FCN). They first trained an FCN to segment the liver from the abdominal images and used the

segmented liver region as input of the second segmentation network, then the tumor segmentation results can be acquired by the second FCN. Finally, they used 3D conditional random field to optimize the tumor segmentation results.

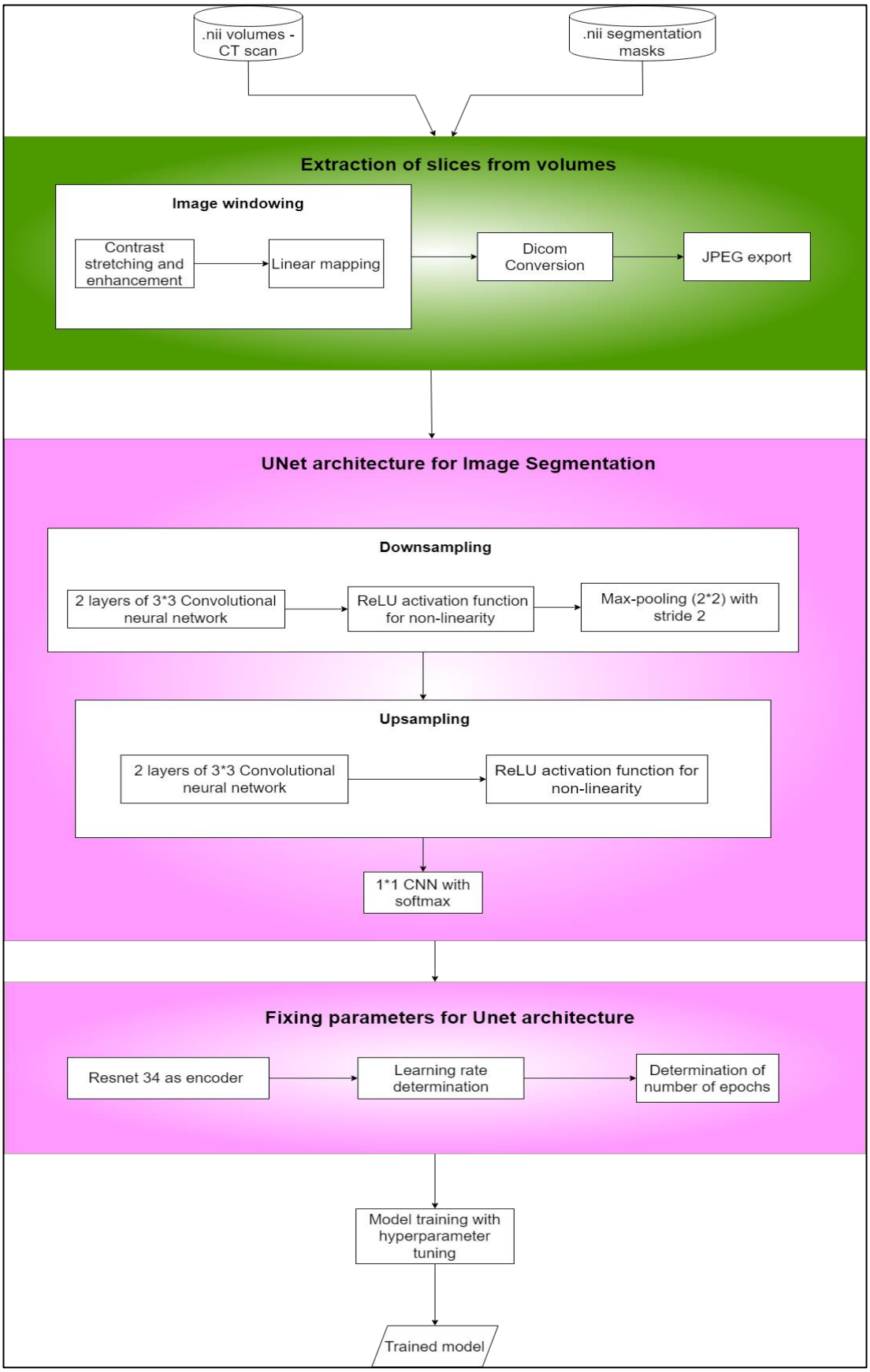
Chen et al [1] developed a cascaded adversarial training system to segment liver tumors from abdominal CT volumes. The liver tumor segmentation challenge was divided into a two cascade binary segmentation tasks and we designed two networks to segment the liver and liver tumor, respectively. He first used multi-plane integrate network to segment the liver tissue from 3D CT abdominal volumes and then extracted the tumors in the liver region by develop a deep 3D densely connected fully convolutional neural network with adversarial training strategy. The networks use a multi-plane convolution operation, which balanced the computing memory consumption and receptive field. They also introduced dense connection to capture more accurate tumor features followed with multi-scale feature fusion technique to reduce the miss segmented results. Adversarial training strategy is used to minimize the output of network with ground truth, which further boosts the final segmentation result. The Experimental results show that our method achieved a best Dice score of 68.4% for tumor segmentation.

3. PROPOSED APPROACH

CNN is an excellent feature extractor, therefore utilizing it to classify medical images can avoid complicated and expensive feature engineering. In this present work we have implemented an automatic segmentation model using deep learning algorithms for separating the liver from the CT scans, and to detect and segment the tumors present in the liver. CNN is used here to segment liver from the CT scans and to detect the presence of tumors in the liver. The model is trained with the help of a dataset that contains the CT scan images and the expected masks that are to be generated. The trained model is later validated using a dataset that contains a CT scan image volume and the predicted mask is compared to the expected mask to find the Dice Loss and IoU loss (Intersection over Union loss)

4. SYSTEM DESIGN

4.1. BLOCK DIAGRAM


.nii volumes - CT scan
.nii segmentation masks
Extraction of slices from volumes
Image windowing
Contrast stretching and enhancement
Linear mapping
Dicom Conversion
JPEG export
UNet architecture for Image Segmentation
Downsampling
2 layers of 3*3 Convolutional neural network
ReLU activation function for non-linearity
Max-pooling (2*2) with stride 2
Upsampling
2 layers of 3*3 Convolutional neural network
ReLU activation function for non-linearity
1*1 CNN with softmax
Fixing parameters for Unet architecture
Resnet 34 as encoder
Learning rate determination
Determination of number of epochs
Model training with hyperparameter tuning
Trained model

4.2. SYSTEM REQUIREMENTS

- 12GB NVIDIA Tesla T4 GPU
- System RAM: 12.68 GB
- Disk Space: 77GB
- Cpu: Intel(R) Xeon(R) CPU @ 2.20GHz
- OS: Ubuntu 18.04.3 LTS

5. DETAILED ARCHITECTURE

1.EXTRACTION OF SLICES FROM CT VOLUMES:

1. 1 Take volumes of CT scans and segmentation masks as input.

1. 2 Image Windowing:

1.2.1 For each slice, apply contrast stretching and enhancement.

1.2.2 Followed by linear mapping for brightness.

1.3 Obtain DICOM intermediates.

1.4 Perform pixel grouping and scaling

1.5 Generate JPG images

The deep learning model will be trained on these enhanced images.

2.LOADING SAMPLES & FITTING FOR UNET ARCHITECTURE:

2.1. Create a Data Block template to hold the Images and the corresponding masks together.

2.2. Using the template, create a data source of from training samples to be used by Data Loader

2.3. Load the training images as batches using the Data Loader.

2.4. Instantiate the Unet model which has following Encoder-Decoder architecture

2.4.1 Downsampling (Encoding) Architecture

2.4.1.1 Two layers of 3x3 CNN.

2.4.1.2 ReLU for activation.

2.4.1.3 Max pooling layer with stride of 2.

2.4.2 Upsampling (Decoder) Architecture

2.4.2.1 Two layers of 3x3 CNN.

2.4.2.2 ReLU for activation.

2.4.2.3 CNN layer with softmax as activation function.

2.5 Print the model summary to check the layers

2.6 Fit the model for the images loaded by the Data Loader

3.DETERMINATION OF HYPERPARAMETERS AND TRAINING:

3.1 Apply ResNet 34 as encoder.

3.2 Fixing Hyperparameters:

3.2.1 Determine learning rate.

3.2.2 Determine number of epochs

3.3 Train the model

3.4 Save the model

3.5 Load the model for testing

3.6 Giving an unknown slice of CT scan as input, test the model for segmentation.

3.7 Obtain segmented image.

6. IMPLEMENTATION

This project aims to predict a mask for a given CT scan. Several slices of the CT scan volume are generated and for the slice that depicts the liver fragment accurately, a mask is generated with three different colours (purple, aqua, yellow) denoting the background, liver and the tumours respectively.

Importing Required Libraries

```
import os
import glob
import cv2
import imageio
import numpy as np
import pandas as pd
import nibabel as nib
import matplotlib.pyplot as plt
from tqdm.notebook import tqdm
from ipywidgets import *
from PIL import Image
from matplotlib.pyplot import figure
from fastai.basics import *
from fastai.vision.all import *
from fastai.data.transforms import *
```

Adding all files to a dataset

```
file_list = []
for dirname, _, filenames in os.walk('../input/liver-tumor-segmentation'):
    for filename in filenames:
        file_list.append((dirname, filename))

for dirname, _, filenames in os.walk('../input/liver-tumor-segmentation-part-2'):
    for filename in filenames:
        file_list.append((dirname, filename))

df_files = pd.DataFrame(file_list, columns =['dirname', 'filename'])
df_files.sort_values(by=['filename'], ascending=True)
```

Opt:

[3]:

	dirname	filename
89	../input/liver-tumor-segmentation/segmentations	segmentation-0.nii
81	../input/liver-tumor-segmentation/segmentations	segmentation-1.nii
142	../input/liver-tumor-segmentation/segmentations	segmentation-10.nii
31	../input/liver-tumor-segmentation/segmentations	segmentation-100.nii
45	../input/liver-tumor-segmentation/segmentations	segmentation-101.nii
...	...	...
243	../input/liver-tumor-segmentation-part-2/volume_pt6	volume-95.nii
222	../input/liver-tumor-segmentation-part-2/volume_pt6	volume-96.nii
217	../input/liver-tumor-segmentation-part-2/volume_pt6	volume-97.nii
252	../input/liver-tumor-segmentation-part-2/volume_pt6	volume-98.nii
231	../input/liver-tumor-segmentation-part-2/volume_pt6	volume-99.nii

262 rows × 2 columns

Adding relation between segmentation and volume (complete CT):

```
df_files["mask_dirname"]  = ""
df_files["mask_filename"] = ""

for i in range(131):
    ct = f"volume-{i}.nii"
    mask = f"segmentation-{i}.nii"

    df_files.loc[df_files['filename'] == ct, 'mask_filename'] = mask
    df_files.loc[df_files['filename'] == ct, 'mask_dirname'] = "../input/liver-tumor-segmentation/segmentations"

df_files = df_files[df_files.mask_filename != ''].sort_values(by=['filename']).reset_index(drop=True)

df_files
```

Opt:

	dirname	filename	mask_dirname	mask_filename
0	../input/liver-tumor-segmentation/volume_pt1	volume-0.nii	../input/liver-tumor-segmentation/segmentations	segmentation-0.nii
1	../input/liver-tumor-segmentation/volume_pt1	volume-1.nii	../input/liver-tumor-segmentation/segmentations	segmentation-1.nii
2	../input/liver-tumor-segmentation/volume_pt1	volume-10.nii	../input/liver-tumor-segmentation/segmentations	segmentation-10.nii
3	../input/liver-tumor-segmentation-part-2/volume_pt6	volume-100.nii	../input/liver-tumor-segmentation/segmentations	segmentation-100.nii
4	../input/liver-tumor-segmentation-part-2/volume_pt8	volume-101.nii	../input/liver-tumor-segmentation/segmentations	segmentation-101.nii
...	...	...	...	...
126	../input/liver-tumor-segmentation-part-2/volume_pt6	volume-95.nii	../input/liver-tumor-segmentation/segmentations	segmentation-95.nii
127	../input/liver-tumor-segmentation-part-2/volume_pt6	volume-96.nii	../input/liver-tumor-segmentation/segmentations	segmentation-96.nii
128	../input/liver-tumor-segmentation-part-2/volume_pt6	volume-97.nii	../input/liver-tumor-segmentation/segmentations	segmentation-97.nii
129	../input/liver-tumor-segmentation-part-2/volume_pt6	volume-98.nii	../input/liver-tumor-segmentation/segmentations	segmentation-98.nii
130	../input/liver-tumor-segmentation-part-2/volume_pt6	volume-99.nii	../input/liver-tumor-segmentation/segmentations	segmentation-99.nii

131 rows × 4 columns

Function to read .nii file and return pixel array:

```
def read_nii(filepath):
    '''
    Reads .nii file and returns pixel array
    '''
    ct_scan = nib.load(filepath)
    array   = ct_scan.get_fdata()
    array   = np.rot90(np.array(array))
    return(array)
```

Getting pixel array:

```
sample = 0
sample_ct = read_nii(df_files.loc[sample,'dirname']+"/"+df_files.loc[sample,'filename'])
sample_mask = read_nii(df_files.loc[sample,'mask_dirname']+"/"+df_files.loc[sample,'mask_filename'])

print(f'CT Shape:   {sample_ct.shape}\nMask Shape: {sample_mask.shape}')
```

```
print(np.amin(sample_ct), np.amax(sample_ct))
print(np.amin(sample_mask), np.amax(sample_mask))
```

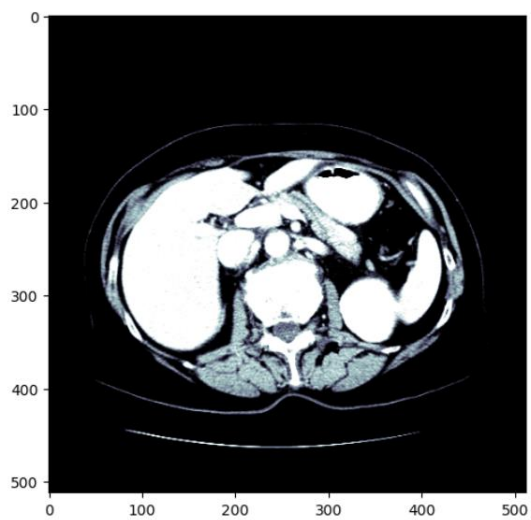
Opt:

```
CT Shape:   (512, 512, 75)
Mask Shape: (512, 512, 75)

-3024.0 1410.0
0.0 2.0
```

Function to convert NIfTi to DICOM:

```
dicom_windows = types.SimpleNamespace(
    brain=(80,40),
    subdural=(254,100),
    stroke=(8,32),
    brain_bone=(2800,600),
    brain_soft=(375,40),
    lungs=(1500,-600),
    mediastinum=(350,50),
    abdomen_soft=(400,50),
    liver=(150,30),
    spine_soft=(250,50),
    spine_bone=(1800,400),
    custom = (200,60)
)

@patch
def windowed(self:Tensor, w, l):
    px = self.clone()
    px_min = l - w//2
    px_max = l + w//2
    px[px<px_min] = px_min
    px[px>px_max] = px_max
    return (px-px_min) / (px_max-px_min)

figure(figsize=(8, 6), dpi=100)

plt.imshow(tensor(sample_ct[..., 55].astype(np.float32)).windowed(*dicom_windows.liver), cmap=plt.cm.bone);
```

Sample Opt:



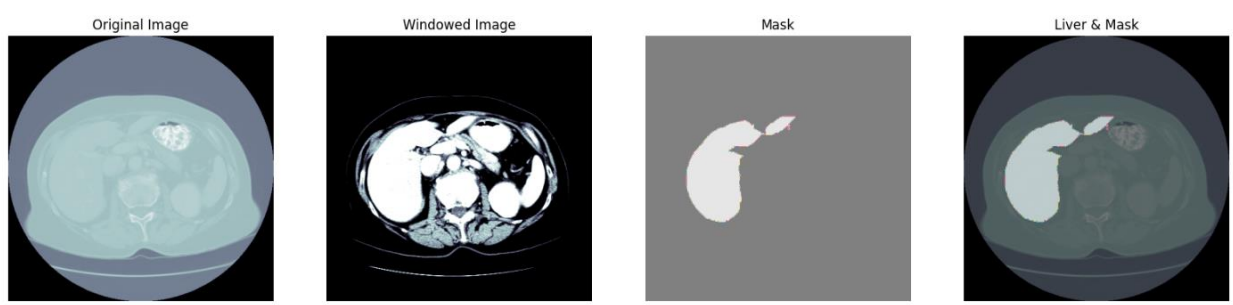
Function to plot Original Image, Windowed Image, Mask and Liver & Mask:

```
def plot_sample(array_list, color_map = 'nipy_spectral'):
    fig = plt.figure(figsize=(20,16), dpi=100)

    plt.subplot(1,4,1)
    plt.imshow(array_list[0], cmap='bone')
    plt.title('Original Image')
    plt.axis('off')

    plt.subplot(1,4,2)
    plt.imshow(tensor(array_list[0].astype(np.float32)).windowed(*dicom_windows.liver), cmap='bone');
    plt.title('Windowed Image')
    plt.axis('off')

    plt.subplot(1,4,3)
    plt.imshow(array_list[1], alpha=0.5, cmap=color_map)
    plt.title('Mask')
    plt.axis('off')

    plt.subplot(1,4,4)
    plt.imshow(array_list[0], cmap='bone')
    plt.imshow(array_list[1], alpha=0.5, cmap=color_map)
    plt.title('Liver & Mask')
    plt.axis('off')

    plt.show()
```

Function call:

```
sample = 55

sample_slice = tensor(sample_ct[...,sample].astype(np.float32))

plot_sample([sample_ct[..., sample],
             sample_mask[..., sample]])
```

Output:

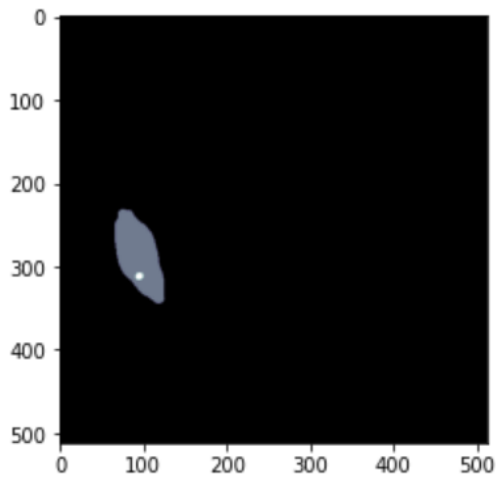


Check the mask values:

```
# Check the mask values
mask = Image.fromarray(sample_mask[...,sample].astype('uint8'), mode="L")
print(mask.shape)
unique, counts = np.unique(mask, return_counts=True)
print( np.array((unique, counts)).T)
plt.imshow(mask , cmap = 'bone')
```

Output:

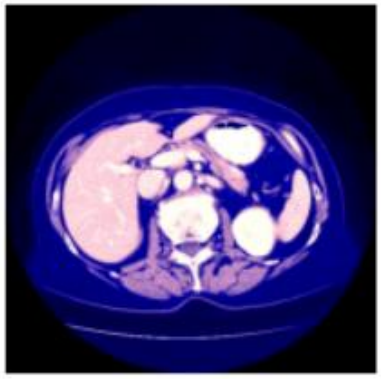
```
(512, 512)
[[     0 257907]
 [     1   4173]
 [     2     64]]
<matplotlib.image.AxesImage at 0x7fc899014cd0>
```



Function to convert DICOM to JPG:

```
class TensorCTScan(TensorImageBW): _show_args = {'cmap':'bone'}

@patch
def freqhist_bins(self:Tensor, n_bins=100):
    imsd = self.view(-1).sort()[0]
    t = torch.cat([tensor([0.001]),
                   torch.arange(n_bins).float()/n_bins+(1/2/n_bins),
                   tensor([0.999])])
    t = (len(imsd)*t).long()
    return imsd[t].unique()

@patch
def hist_scaled(self:Tensor, brks=None):
    if self.device.type=='cuda': return self.hist_scaled_pt(brks)
    if brks is None: brks = self.freqhist_bins()
    ys = np.linspace(0., 1., len(brks))
    x = self.numpy().flatten()
    x = np.interp(x, brks.numpy(), ys)
    return tensor(x).reshape(self.shape).clamp(0.,1.)


@patch
def to_nchan(x:Tensor, wins, bins=None):
    res = [x.windowed(*win) for win in wins]
    if not isinstance(bins,int) or bins!=0: res.append(x.hist_scaled(bins).clamp(0,1))
    dim = [0,1][x.dim()==3]
    return TensorCTScan(torch.stack(res, dim=dim))

@patch
def save_jpg(x:(Tensor), path, wins, bins=None, quality=120):
    fn = Path(path).with_suffix('.jpg')
    x = (x.to_nchan(wins, bins)*255).byte()
    im = Image.fromarray(x.permute(1,2,0).numpy(), mode=['RGB','CMYK'][x.shape[0]==4])
    im.save(fn, quality=quality)

_,axs = subplots(1,1)

sample_slice.save_jpg('test.jpg', [dicom_windows.liver, dicom_windows.custom])
show_image(Image.open('test.jpg'), ax=axs[0], figsize=(8, 6))
```

```
<AxesSubplot:>
```



Generating JPG for all:

```
GENERATE_JPG_FILES = True

if (GENERATE_JPG_FILES) :

    path = Path(".")

    os.makedirs('train_images',exist_ok=True)
    os.makedirs('train_masks',exist_ok=True)

    for ii in tqdm(range(0,len(df_files),3)):
        curr_ct        = read_nii(df_files.loc[ii,'dirname']+"/"+df_files.loc[ii,'filename'])
        curr_mask      = read_nii(df_files.loc[ii,'mask_dirname']+"/"+df_files.loc[ii,'mask_filename'])
        curr_file_name = str(df_files.loc[ii,'filename']).split('.')[0]
        curr_dim       = curr_ct.shape[2]
        for curr_slice in range(0,curr_dim,2):
            data = tensor(curr_ct[...,curr_slice].astype(np.float32))
            mask = Image.fromarray(curr_mask[...,curr_slice].astype('uint8'), mode="L")
            data.save_jpg(f"train_images/{curr_file_name}_slice_{curr_slice}.jpg", [dicom_windows.liver,dicom_windows.custom])
            mask.save(f"train_masks/{curr_file_name}_slice_{curr_slice}_mask.png")
else:
    path = Path("../input/liver-segmentation-with-fastai-v2")
```

Output:

```
100% ################################################## 44/44 [14:01<00:00, 28.94s/it]
```

Model Training

```
bs = 16
im_size = 128

codes = np.array(["background","liver","tumor"])

def get_x(fname:Path):
    return fname

def label_func(x):
    return path/'train_masks'/f'{x.stem}_mask.png'

tfms = [IntToFloatTensor(),Normalize()]

db = DataBlock(blocks=(ImageBlock(),MaskBlock(codes)),  #codes = {"Backround": 0,"Liver": 1,"Tumor": 2}
               batch_tfms=tfms,
               splitter=RandomSplitter(),
               item_tfms=[Resize(im_size)],
               get_items=get_image_files,
               get_y=label_func
              )
```

```
# ../output/kaggle/working/train_images.zip
# ds = db.datasets(source=path/'train_images.zip')
ds = db.datasets(source='./train_images')
print(len(ds))
print(ds)
```

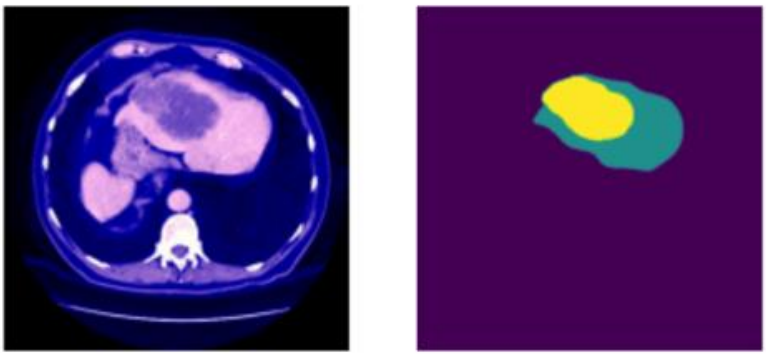
Output:

```
1869
(#1869) [(PILImage mode=RGB size=512x512, PILMask mode=L size=512x512),(PILImage mode=RGB size=512x512, PILMask mode=L size=512x512),(PILImage mode=RGB
```

Image and corresponding mask in the training sample

```
[ ]  idx=48
     imgs = [ds[idx][0],ds[idx][1]]
     fig,axs = plt.subplots(1, 2)
     for i,ax in enumerate(axs.flatten()):
         ax.axis('off')
         ax.imshow(imgs[i]) #, cmap='gray'
```

Output:



Encoded values for background, liver and tumor

```
unique, counts = np.unique(array(ds[idx][1]), return_counts=True)

print( np.array((unique, counts)).T)
```

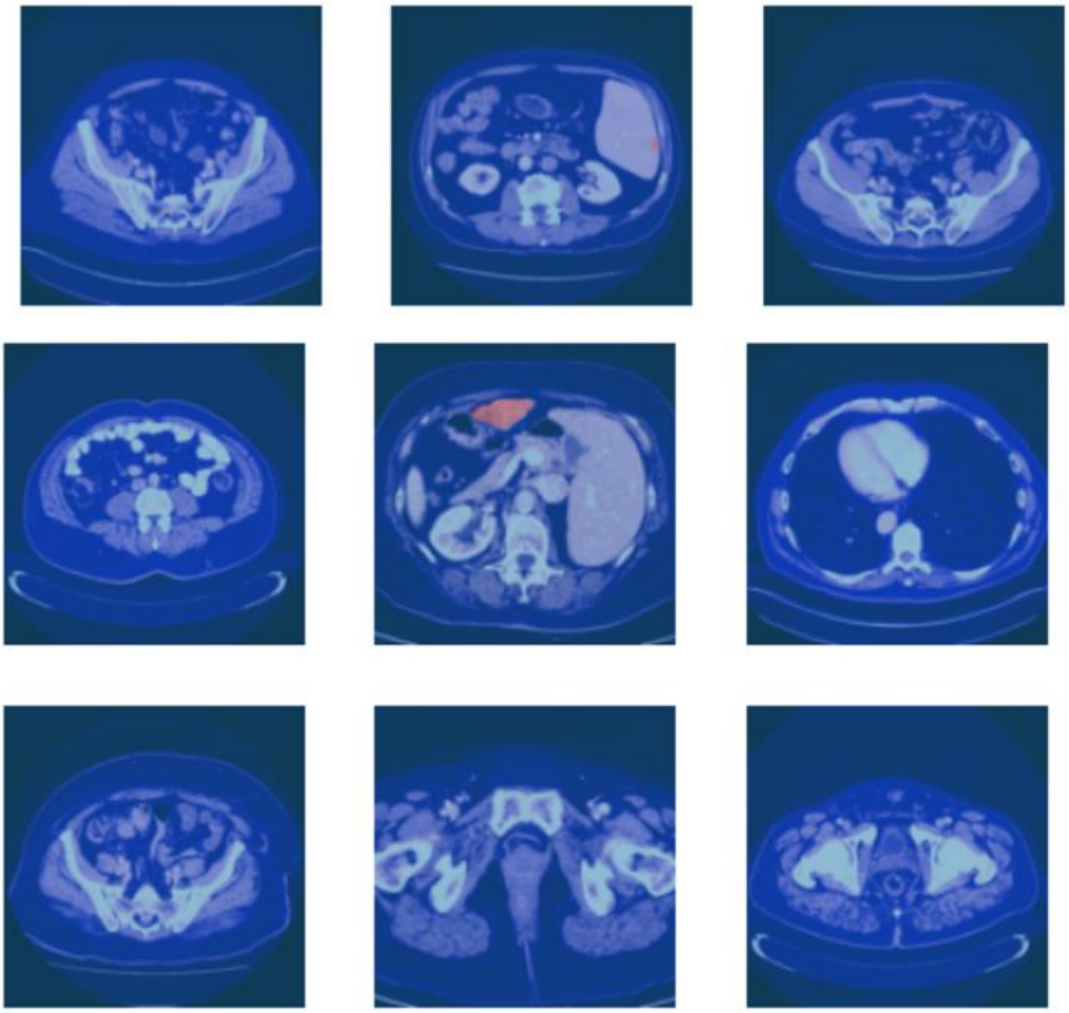
Output:

```
[[      0 240441]
 [      1  12901]
 [      2   8802]]
```

Viewing training sample as a batch

```
dls = db.dataloaders(path/'train_images',bs = bs) #, num_workers=0
dls.show_batch()
```

Output:



```
[ ] def foreground_acc(inp, targ, bkg_idx=0, axis=1):  # exclude a background from metric
        "Computes non-background accuracy for multiclass segmentation"
        targ = targ.squeeze(1)
        mask = targ != bkg_idx
        return (inp.argmax(dim=axis)[mask]==targ[mask]).float().mean()

    def cust_foreground_acc(inp, targ):  # # include a background into the metric
        return foreground_acc(inp=inp, targ=targ, bkg_idx=3, axis=1) # 3 is a dummy value to include the background which is 0
```

Importing unet_learner from fastai

```
[ ] learn = unet_learner(dls, resnet34, loss_func=CrossEntropyLossFlat(axis=1), metrics=[foreground_acc, cust_foreground_acc])
```

Output:

```
Downloading: "https://download.pytorch.org/models/resnet34-333f7ec4.pth" to /root/.cache/torch/hub/checkpoints/resnet34-333f7ec4.pth
100% [=========================] 83.3M/83.3M [00:00<00:00, 130MB/s]
```

Training model with hyper parameter tuning – number of epochs : 5

```
learn.fine_tune(5, wd=0.1, cbs=SaveModelCallback() )
```

Output:

epoch	train_loss	valid_loss	foreground_acc	cust_foreground_acc	time
0	0.055949	0.074643	0.568199	0.988721	00:47

```
Better model found at epoch 0 with valid_loss value: 0.07464338839054108.
```

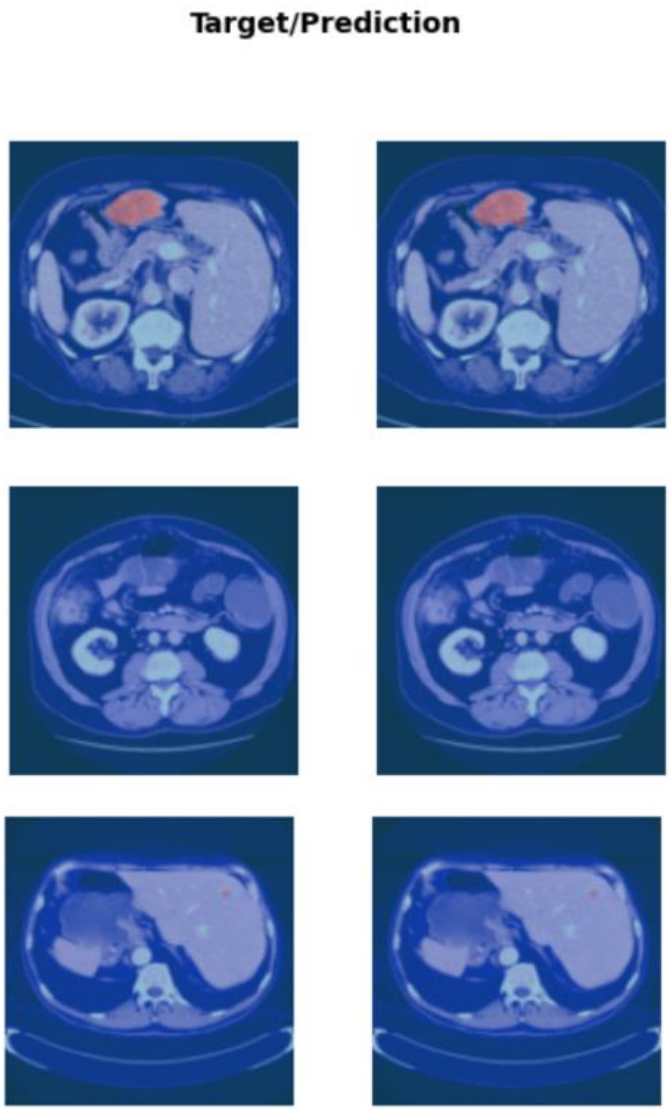
epoch	train_loss	valid_loss	foreground_acc	cust_foreground_acc	time
0	0.011018	0.011883	0.934516	0.995735	00:49
1	0.008316	0.008421	0.935173	0.997021	00:49
2	0.006285	0.007325	0.948141	0.997547	00:49
3	0.004609	0.004661	0.957442	0.998172	00:48
4	0.003410	0.004478	0.953530	0.998272	00:49

```
Better model found at epoch 0 with valid_loss value: 0.011882836930453777.
Better model found at epoch 1 with valid_loss value: 0.008420741185545921.
Better model found at epoch 2 with valid_loss value: 0.007324928883463144.
Better model found at epoch 3 with valid_loss value: 0.004660581238567829.
Better model found at epoch 4 with valid_loss value: 0.004478479735553265.
```

Training results

```
learn.show_results()
```

Output:



Saving the learned model

```
[ ]  # Save the model
     learn.export(path/f'Liver_segmentation')
```

```
[ ]  import gc
     del learn
     gc.collect()
     torch.cuda.empty_cache()
```

Predicting the model:

```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import glob

import nibabel as nib
import cv2
import imageio
from tqdm.notebook import tqdm
from ipywidgets import *
from PIL import Image


import fastai;
print(fastai.__version__)
from fastai.basics import *
from fastai.vision.all import *
from fastai.data.transforms import *
```

```
2.2.5
```

```
# Create a meta file for nii files processing

file_list = []
for dirname, _, filenames in os.walk(r'/content/drive/MyDrive/LITS Challenge/Training Batch 1'):
    for filename in filenames:
#         print(os.path.join(dirname, filename))
        file_list.append((dirname,filename))

for dirname, _, filenames in os.walk(r'/content/drive/MyDrive/LITS Challenge/Training Batch 2'):
    for filename in filenames:
        file_list.append((dirname,filename))

df_files = pd.DataFrame(file_list, columns =['dirname', 'filename'])


# Map CT scan and label
df_files["mask_dirname"] = "" ; df_files["mask_filename"] = ""
```

```
for i in range(131):
    ct = f"volume-{i}.nii"
    mask = f"segmentation-{i}.nii"

    df_files.loc[df_files['filename'] == ct, 'mask_filename'] = mask
    p=""
    if i<=27 :
      p = r"/content/drive/MyDrive/LITS Challenge/Training Batch 1"
    else :
      p = r"/content/drive/MyDrive/LITS Challenge/Training Batch 2"
    df_files.loc[df_files['filename'] == ct, 'mask_dirname'] = p

df_files_test= df_files[df_files.mask_filename==''']
# drop segment rows
df_files = df_files[df_files.mask_filename != ''].sort_values(by=['filename']).reset_index(drop=True)
print(len(df_files))

# function used to read nii files and convert into a numpy array
def read_nii(filepath):
    '''
    Reads .nii file and returns pixel array
    '''
    ct_scan = nib.load(filepath)
    array   = ct_scan.get_fdata()
```

```
    array   = np.rot90(np.array(array))
    return(array)
```

```
131
```

```
[ ] # df_files=df_files[100:131]
    # df_files

    # first 20 data points
    df_file = df_files[0:20]
```

```
# Load saved model
bs = 16
im_size = 128

# the labels used for the classes
# When predicting the model predicts it in terms of indices (ie 0 --> background, 1 --> liver ...)
codes = np.array(["background","liver","tumor"])

# the default pathb
path = './'

def get_x(fname:Path):
    return fname

def label_func(x):
    return path/'train_masks'/f'{x.stem}_mask.png'
```

```
def foreground_acc(inp, targ, bkg_idx=0, axis=1):  # exclude a background from metric
    "Computes non-background accuracy for multiclass segmentation"
    targ = targ.squeeze(1)
    mask = targ != bkg_idx
    return (inp.argmax(dim=axis)[mask]==targ[mask]).float().mean()

def cust_foreground_acc(inp, targ):  # # include a background into the metric
    return foreground_acc(inp=inp, targ=targ, bkg_idx=3, axis=1) # 3 is a dummy value to include the background which is 0
```

Loading the model:

```
# loading the tensor flow model
tfms = [Resize(im_size), IntToFloatTensor(),Normalize()]
learn0 = load_learner('/content/Liver_segmentation',cpu=False )
learn0.dls.transform = tfms
```

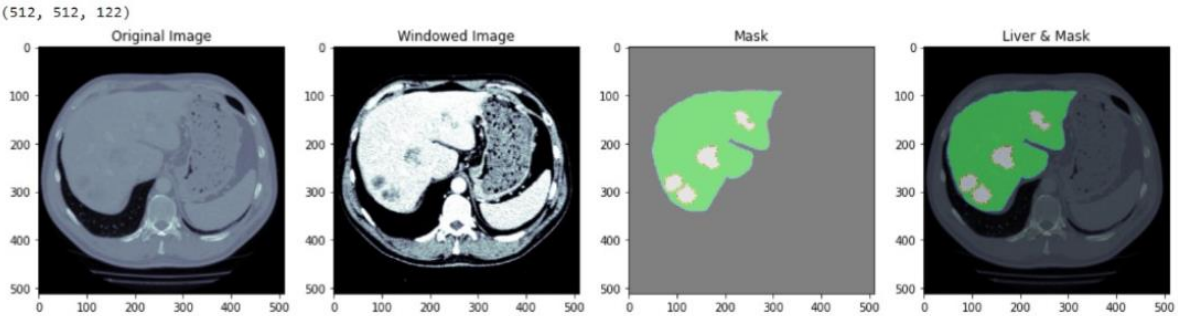
```
def nii_tfm(fn,wins):

    test_nii  = read_nii(fn)
    curr_dim  = test_nii.shape[2] # 512, 512, curr_dim
    slices = []

    for curr_slice in range(curr_dim):
        data = tensor(test_nii[...,curr_slice].astype(np.float32))
        data = (data.to_nchan(wins)*255).byte()
        slices.append(TensorImage(data))

    return slices
```

Selecting a slice in test volume:

```
[34] # test number
     tst = 66

     # slice number
     test_slice_idx = 85


     test_nii   = read_nii(df_files.loc[tst,'dirname']+"/"+df_files.loc[tst,'filename'])
     test_mask  = read_nii(df_files.loc[tst,'mask_dirname']+"/"+df_files.loc[tst,'mask_filename'])
     print(test_nii.shape)

     sample_slice = tensor(test_nii[...,test_slice_idx].astype(np.float32))

     plot_sample([test_nii[...,test_slice_idx], test_mask[...,test_slice_idx]])
```

Plot of original image, windowed image, expected mask and imposing it on the liver:

```
(512, 512, 122)
```



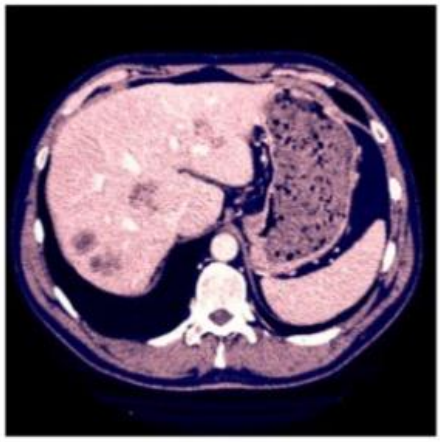
Generating slices for a CT scan:

```
[35] # Prepare a nii test file for prediction

     test_files = nii_tfm(df_files.loc[tst,'dirname']+"/"+df_files.loc[tst,'filename'],[dicom_windows.liver, dicom_windows.custom])
     print("Number of test slices: ",len(test_files))

     Number of test slices:  122
```

Choosing a slice:

```
# Check an input for a test file
#show_image(test_files[0])
show_image(test_files[test_slice_idx])
#show_image(test_files[501])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f51685d61d0>
```



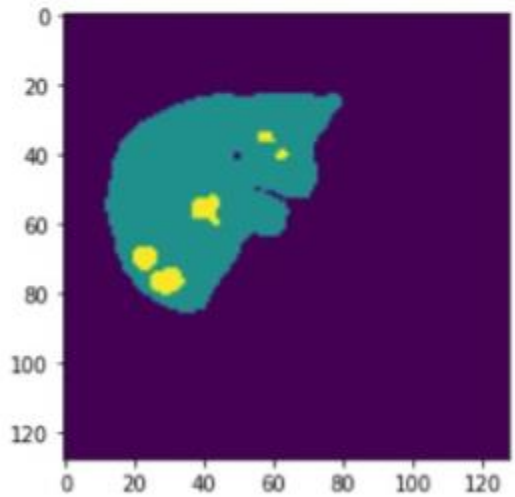
Predicting a mask for the given slice:

```
[37] # Get predictions for a Test file

     test_dl = learn0.dls.test_dl(test_files)
     preds,y = learn0.get_preds(dl=test_dl)

     #predicted_mask = np.argmax(preds, axis=1)
     #print(type(predicted_mask))
     #print(predicted_mask[test_slice_idx].shape)
     #plt.imshow(predicted_mask[test_slice_idx])
```

The Generated Mask:

```
predicted_mask = np.argmax(preds, axis=1)
print(type(predicted_mask))
#print(predicted_mask[test_slice_idx].shape)
plt.imshow(predicted_mask[test_slice_idx])
#plt.imshow(predicted_mask[0])
```

```
<class 'fastai.torch_core.TensorImage'>
<matplotlib.image.AxesImage at 0x7f5168562290>
```




```
[39] #a=np.array(predicted_mask[0])
     a=np.array(predicted_mask[test_slice_idx])
     unique, counts = np.unique(a, return_counts=True)
     print( np.array((unique, counts)).T)

     np.amin(a),np.amax(a),
```

```
[[    0 13639]
 [    1  2571]
 [    2   174]]
(0, 2)
```

METRICS FOR EVALUATION:

DiceLoss:

```
[62] #PyTorch
     class DiceLoss(nn.Module):
         def __init__(self, weight=None, size_average=True):
             super(DiceLoss, self).__init__()

         def forward(self, inputs, targets, smooth=1):

             #comment out if your model contains a sigmoid or equivalent activation layer
             #inputs = F.sigmoid(inputs)

             #flatten label and prediction tensors
             #inputs = inputs.view(-1)
             #targets = targets.view(-1)

             intersection = (inputs * targets).sum()
             dice = (2.*intersection + smooth)/(inputs.sum() + targets.sum() + smooth)

             return 1 - dice
```

```
print(abs(DiceLoss().forward(numdata, res)))
```

Intersection Over Union:

```
class IoULoss(nn.Module):
    def __init__(self, weight=None, size_average=True):
        super(IoULoss, self).__init__()

    def forward(self, inputs, targets, smooth=1):

        #comment out if your model contains a sigmoid or equivalent activation la
        #inputs = F.sigmoid(inputs)

        #flatten label and prediction tensors
        #inputs = inputs.view(-1)
        #targets = targets.view(-1)

        #intersection is equivalent to True Positive count
        #union is the mutually inclusive area of all labels & predictions
        intersection = (inputs * targets).sum()
        total = (inputs + targets).sum()
        union = total - intersection

        IoU = (intersection + smooth)/(union + smooth)

        return 1 - IoU
```

```
print(abs(IoULoss().forward(numdata, res)))
```

7. RESULTS & DISCUSSIONS

Different scenarios are tested on the model and the output is tabulated as shown below.

Evaluation metrics is chosen to be Dice Loss and IoU Loss.

- Dice Loss:

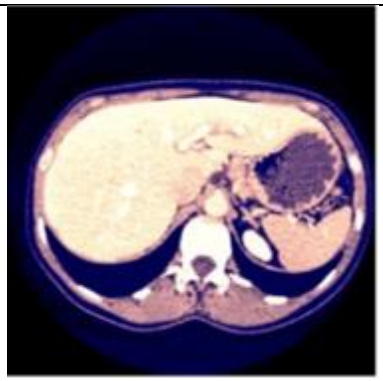
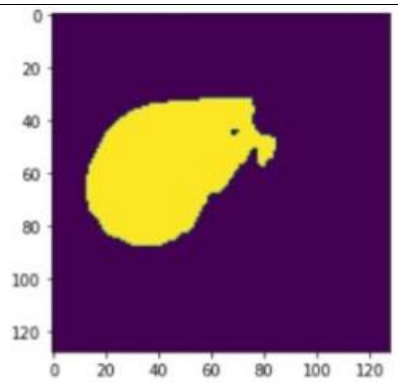
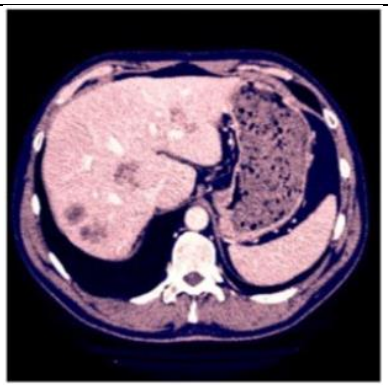
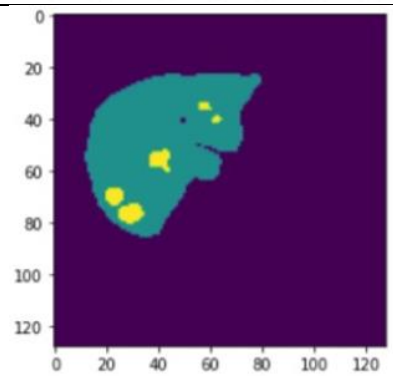
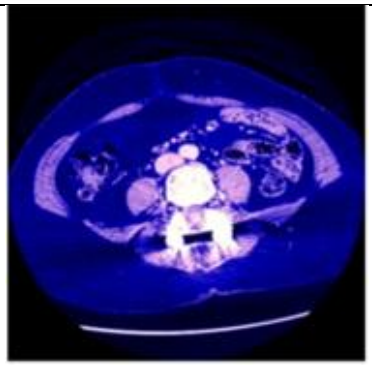
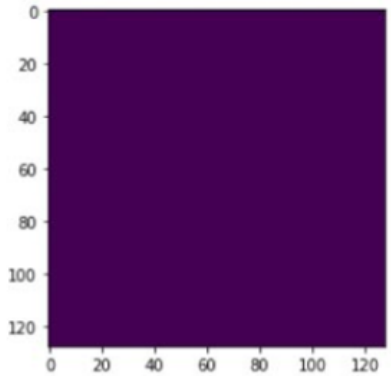
The Dice coefficient is widely used metric in computer vision community to calculate the similarity between two images. Later in 2016, it has also been adapted as loss function known as Dice Loss

$$1 - \frac{2 \times |y \cap y - pred|}{|y + y - pred|}$$

- IoU Loss:

Intersection over Union (IoU), also known as the Jaccard index, is the most popular evaluation metric for tasks such as segmentation, object detection and tracking.

$$1 - \frac{|y \cap y - pred|}{|y \cup y - pred|}$$

Image	Input Scan Image	Final Output	Evaluation Metrics
Only Liver, No lesion			IoU Loss: 0.19102 Dice Loss: 0.10561
Liver with lesion			IoU Loss: 0.11256 Dice Loss: 0.10986
No liver present in the segmented slice			IoU Loss: 0.0 Dice Loss: 0.0

7.1. RESULT ANALYSIS

The above table shows the output of the model on various scenarios. The three scenarios which occur are: Image with both Liver and Lesion, Image with only Liver, Image with neither Liver

nor Lesion. The colors, Purple, Aqua and Yellow are used to indicate the segmented results (Background, Liver and Lesion).

The Losses for the image which has both Liver and Lesion (IoU Loss - 0.11256, Dice Loss - 0.10986) and the image which has only Liver (IoU Loss – 0.19102, Dice Loss – 0.10561) are similar, whereas the loss for the image which has neither Liver nor Lesion is 0. This can be explained by the fact that the model is trained explicitly to segment out the liver and the lesion parts and does not mask out any other organ, thus avoiding conflicts by unwanted results.

8. CONCLUSION & FUTURE WORK

Thus, we have developed a deep learning model using ResNet architecture with CT images for image segmentation. The model was trained under predominant cases to edge out the liver and lesion of all sizes for early detection of tumor.

Our future work will focus on automating the segmentation of other organs apart from liver by both semantic and instance segmentation methods.

REFERENCES

1.Chen, L., Song, H., Wang, C., Cui, Y., Yang, J., Hu, X. and Zhang, L., 2019. Liver tumour segmentation in CT volumes using an adversarial densely connected network. BMC bioinformatics, 20(16), pp.1-13. (Base paper)

2.C. F. Shi, Y. Z. Cheng, F. Liu, Y. D. Wang, J. Bai, and S. Tamura, “A hierarchical local region-based sparse shape composition for liver segmentation in CT scans,” Pattern Recognition, vol. 50, pp. 88–106, 2016.

3.Massoptier L, Casciaro S. A new fully automatic and robust algorithm for fast segmentation of liver tissue and tumors from ct scans. Eur Radiol. 2008;18(8):1658

4.Wong D, Liu J, Fengshou Y, Tian Q, Xiong W, Zhou J, Qi Y, Han T, Venkatesh S, Wang S-c. A semi-automated method for liver tumor segmentation based on 2d region growing with knowledge-based constraints. In: MICCAI Workshop, vol. 41. Berlin: Springer-Verlag Berlin Heidelberg; 2008

5.Zheng Z, Zhang X, Xu H, Liang W, Zheng S, Shi Y. A unified level set framework combining hybrid algorithms for liver and liver tumour segmentation in CT images. BioMed Res Int. 2018;2018

6.Wong D, Liu J, Fengshou Y, Tian Q, Xiong W, Zhou J, Qi Y, Han T, Venkatesh S, Wang S-c. A semi-automated method for liver tumour segmentation based on 2d region growing with knowledge-based constraints. In: MICCAI Workshop, vol. 41. Berlin: Springer-Verlag Berlin Heidelberg; 2008.

7.Q. Dou, L. Yu, H. Chen et al., “3D deeply supervised network for automated segmentation of volumetric medical images,” Medical Image Analysis, vol. 41, pp. 40–54, 2017.

8.F. Lu, F. Wu, P. Hu, Z. Peng, and D. Kong, “Automatic 3D liver location and segmentation via convolutional neural network and graph cut,” International Journal for Computer Assisted Radiology and Surgery, vol. 12, no. 2, pp. 171–182, 2017.

9.Christ PF, Ettlinger F, Grün F, Elshaera MEA, Lipkova J, Schlecht S, Ahmaddy F, Tatavarty S, Bickel M, Bilic P, et al. Automatic liver and tumor segmentation of ct and mri volumes using cascaded fully convolutional neural networks. 2017.

10.Qiangguo Jin, Zhaopeng Meng, Changming Sun, Hui Cui and Ran Su, “RA-UNet: A Hybrid Deep Attention-Aware Network to Extract Liver and Tumor in CT Scans,”. Bioeng, Biotechnol, 2020.

11.Meng, L., Zhang, Q. and Bu, S., 2021. Two-Stage Liver and Tumour Segmentation Algorithm Based on Convolutional Neural Network. Diagnostics, 11(10), p.1806.

12.Almotairi, S., Kareem, G., Aouf, M., Almutairi, B. and Salem, M.A.M., 2020. Liver tumour segmentation in CT scans using modified SegNet. Sensors, 20(5), p.1516.

13.Chlebus, G., Schenk, A., Moltz, J.H., van Ginneken, B., Hahn, H.K. and Meine, H., 2018. Automatic liver tumour segmentation in CT with fully convolutional neural networks and object-based postprocessing. Scientific reports, 8(1), pp.1-7.

14.Zhang, Y., Pan, X., Li, C. and Wu, T., 2020. 3D liver and tumour segmentation with CNNs based on region and distance metrics. Applied Sciences, 10(11), p.3794.

15.Sumathy, B., Dadheech, P., Jain, M., Saxena, A., Hemalatha, S., Liu, W. and Nuagah, S.J., 2022. A Liver Damage Prediction Using Partial Differential Segmentation with Improved Convolutional Neural Network. Journal of Healthcare Engineering, 2022.