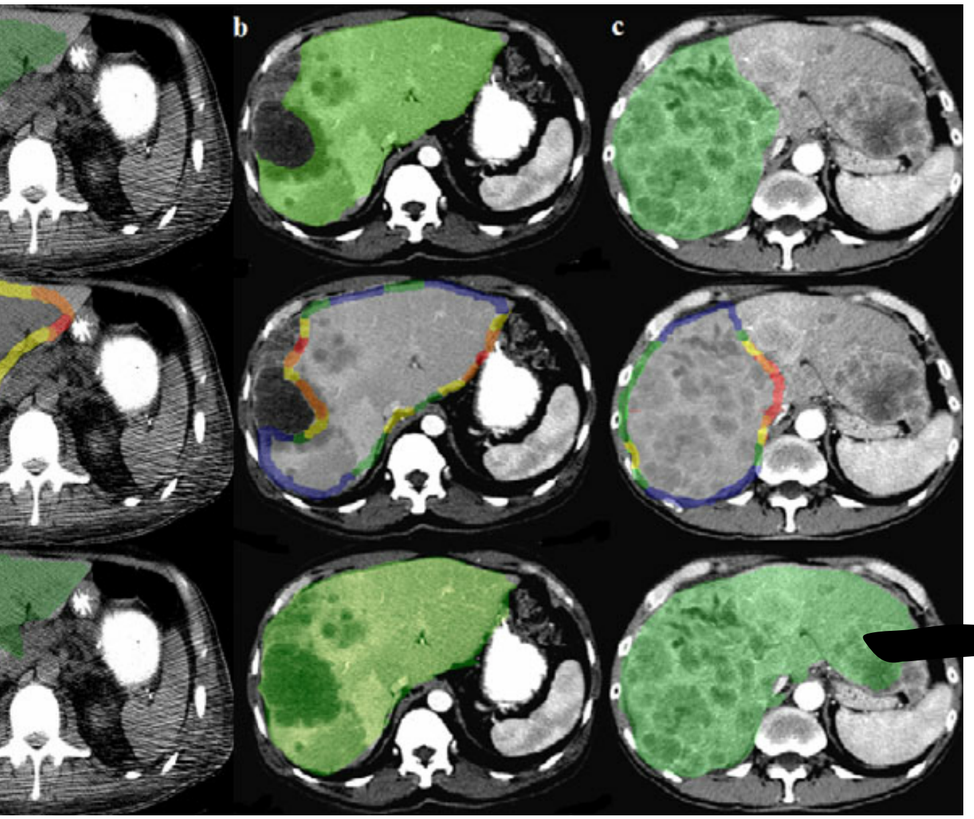
Data science project



AI HEALTHCARE IMAGING

GROUP 4

LIVER SEGMENTATION

TEACHER:
PAIROTE SATIRACOO

USING MONAI AND PYTORCH

Vipoo | Rasita | Nareupon | Atcharawan | Ketsiree

WELCO-ME TO TODAY'S CLASS!

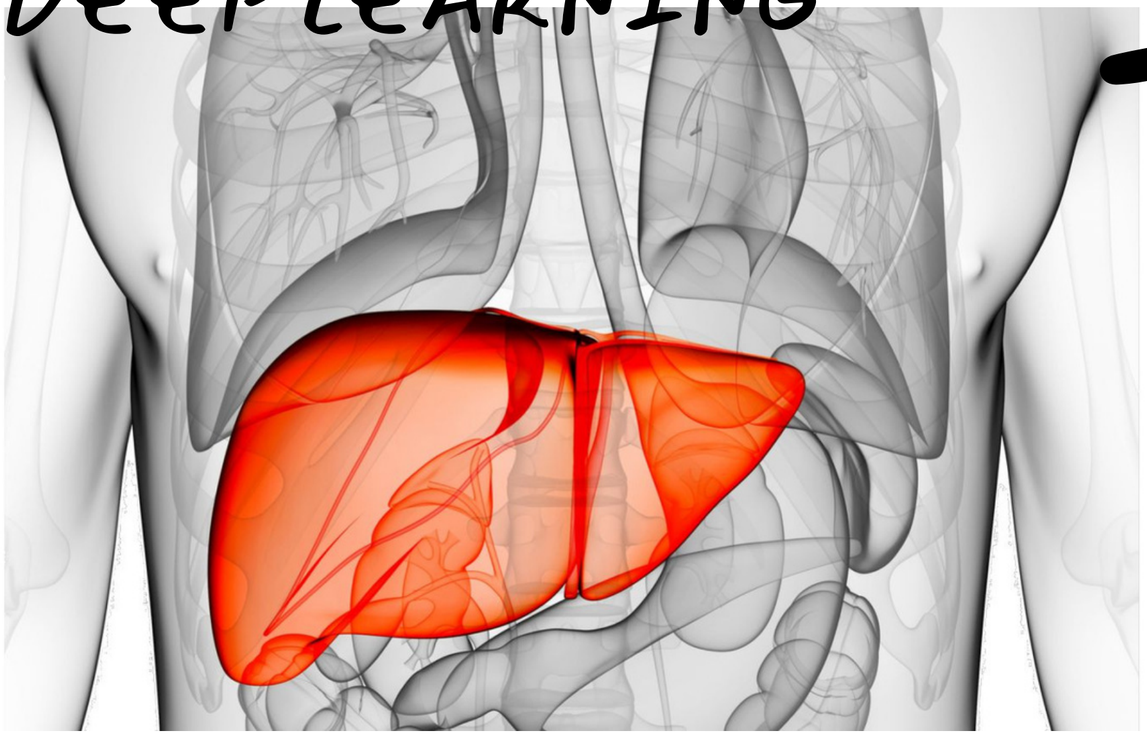
TODAY'S AGENDA

1. INTRODUCTION
2. SOFTWARE INSTALLATION
3. ABOUT THIS DATASET
4. PREPARE AND PREPROCESS THE DATA.
5. TRAINING AND TESTING
6. RESULT

LIVER SEGMENTION

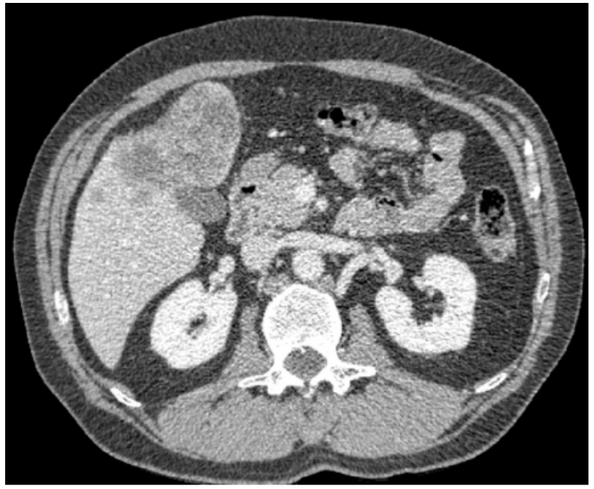
INTRODUCTION

DEEP LEARNING



WHAT IS THERE IN IMAGE AND WHERE?

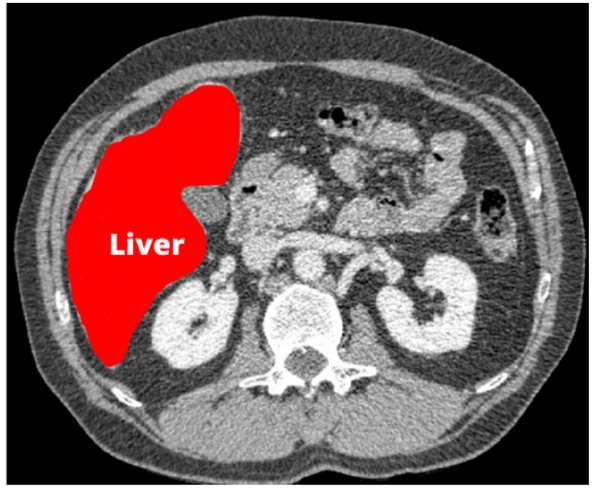
WHICH PIXELS BELONGS TO WHICH OBJECT?



Classification

Liver: 99.99%

Object Detection



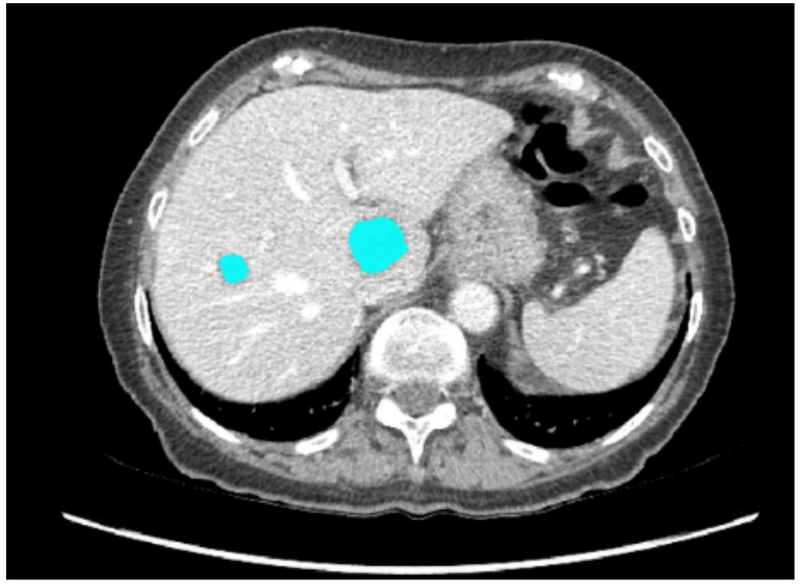
Segmentation

IMAGE CLASSIFICATION

IMAGE DETECTION

IMAGE SEGMENTATION

IMAGE SEGMENTATION TYPES



Semantic Segmentation

Tumor/No Tumor

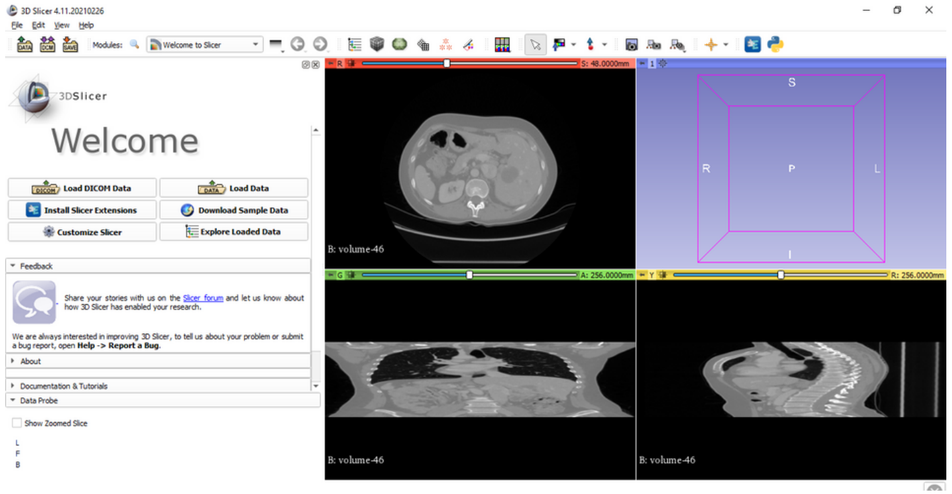
Instance Segmentation

Tumor/No Tumor/How many

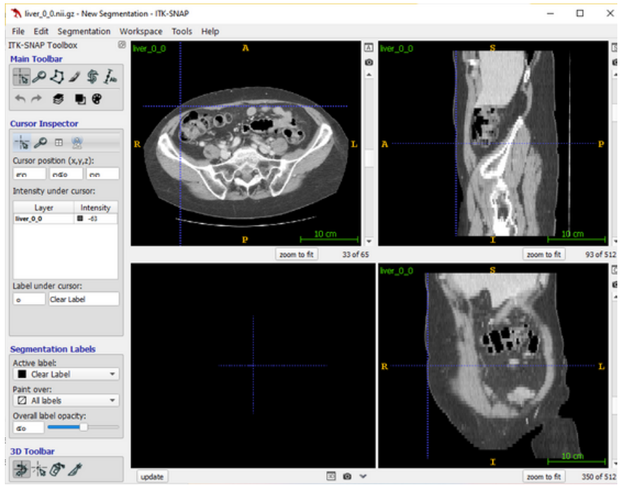
IN OUR CASE, SEMANTIC SEGMENTATION WILL BE USED BECAUSE WE NEED TO KNOW WHETHER THE SLICE CONTAINS A LIVER OR NOT.

SOFTWARE INSTALLATION

- *PROGRAMMING LANGUAGE: PYTHON*
- *3D VISUALIZATION: 3D SLICER*
- *TOOL FOR SEGMENTATION: ITK SNAP*



3D SLICER



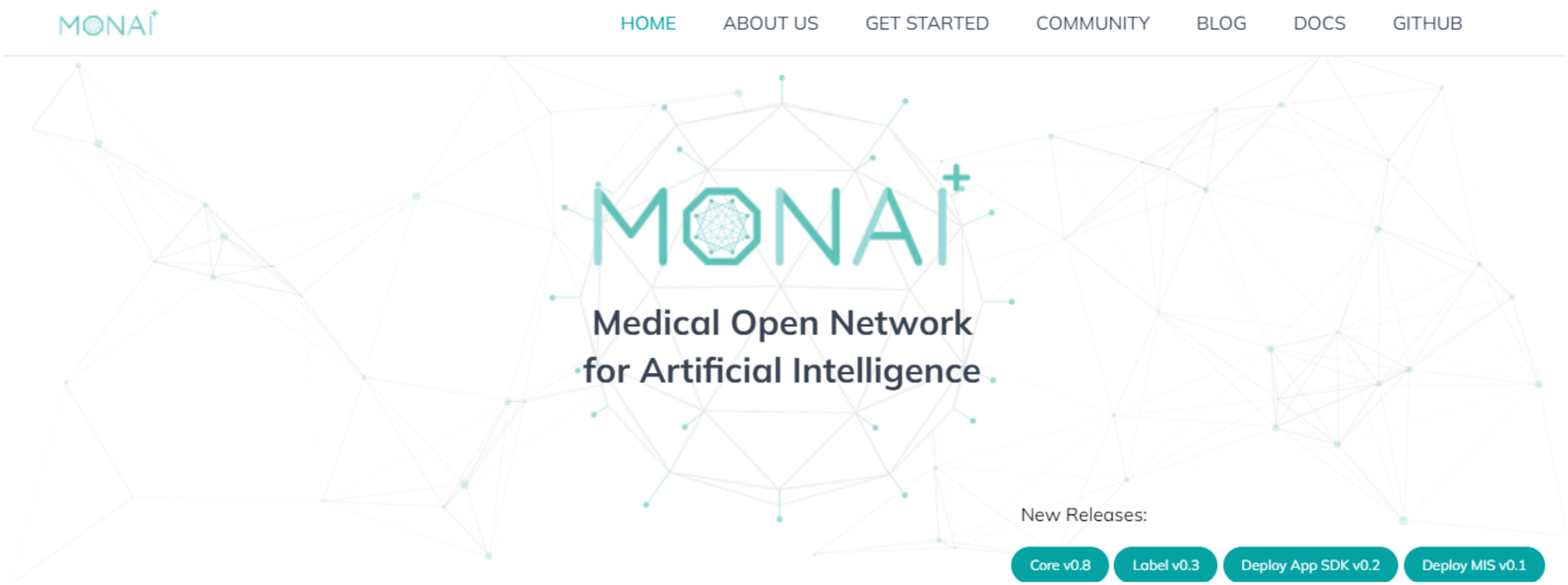
ITK SNAP

INSTALLATION AND INCLUDES

- INSTALL CUDA
- INSTALL CUDNN
- INSTALL MONAI
- INSTALL PYTORCH

*requirements

MATPLOTLIB
NUMPY
TQDM
GLOB2
DICOM2NIFTI
PYTEST-SHUTIL
NIBABEL



WHAT IS MONAI ?

NVIDIA เปิดตัว MEDICAL OPEN NETWORK FOR AI (MONAI) เฟรมเวิร์ค โอเพนซอร์สสำหรับ AI วิเคราะห์ภาพถ่ายทางการแพทย์ (MEDICAL IMAGING เช่น X-RAY, CT, MRI, ULTRASOUND) ตัวมันเองพัฒนาจาก PYTORCH

การเตรียมข้อมูล

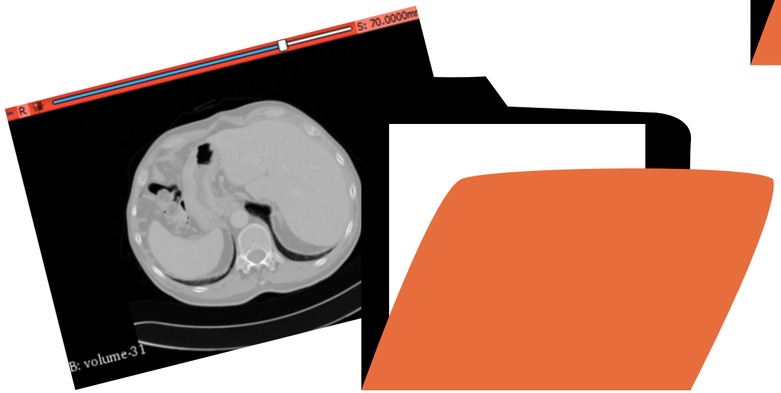
PREPARING DATA

*STEP 1:

1. Download Dataset
2. Dividing the data into train/test folders
3. Convert nifties into dicom files
4. Move every 65 slices into a folder
5. Reconvert the groups of 65 slices into nifti files
6. Delete the empty volumes

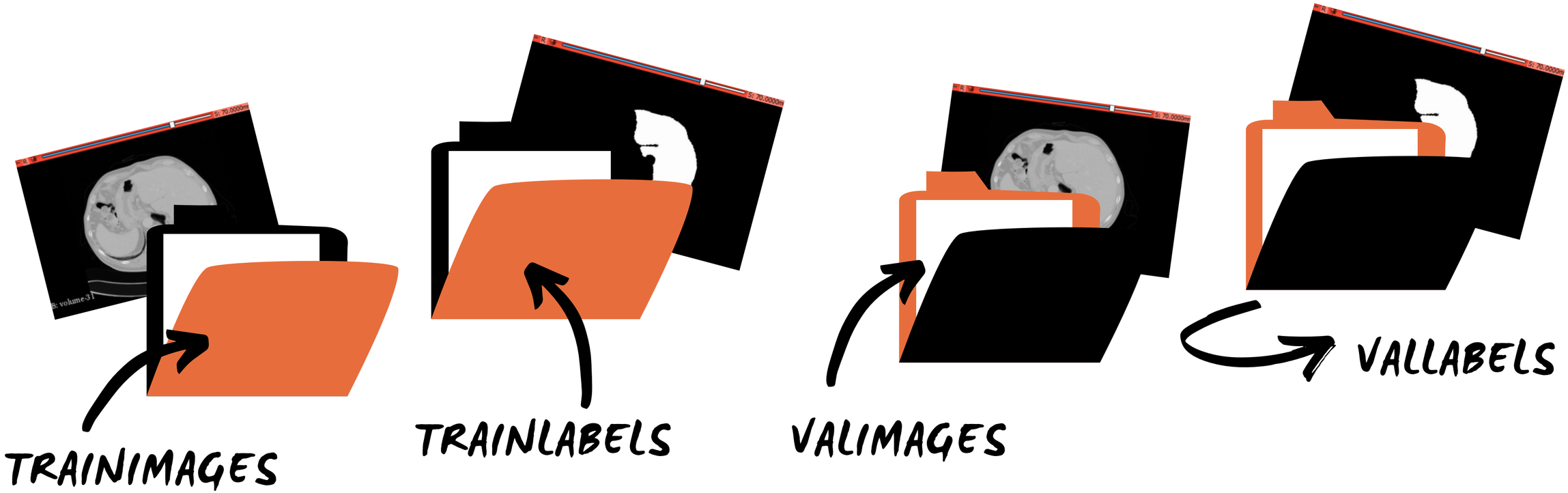
DOWNLOAD DATASET FROM KAGGLE

SEGMENTATIONS



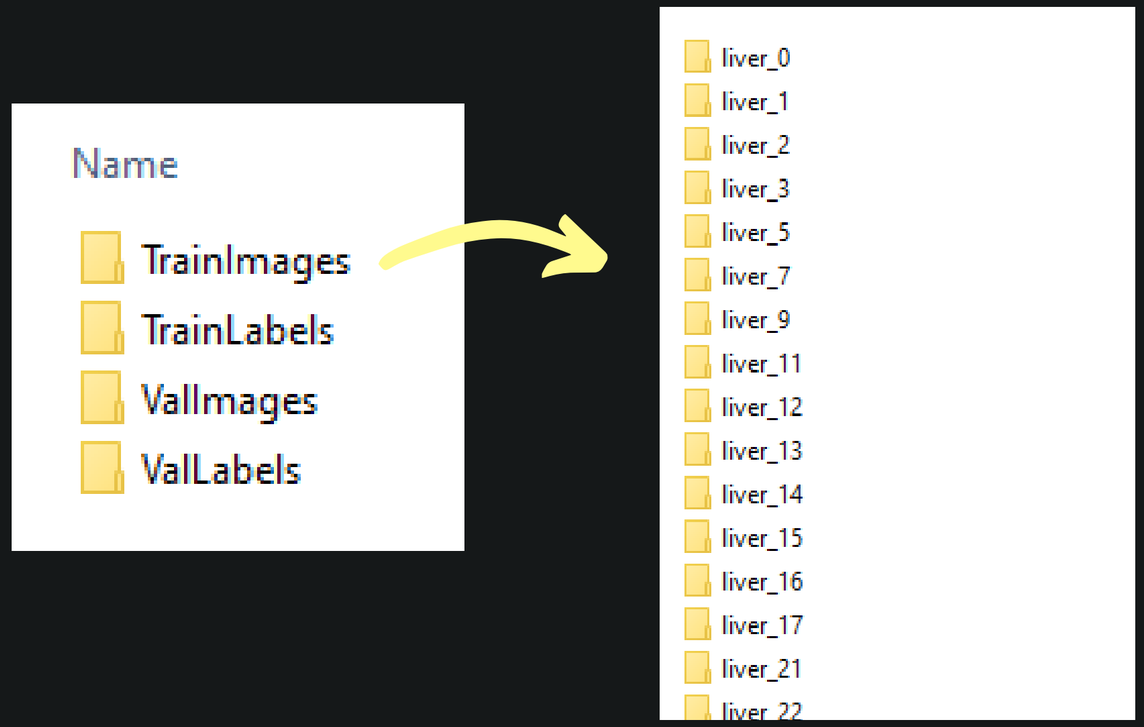
VOLUME

DIVIDING THE DATA INTO TRAIN/TEST FOLDERS

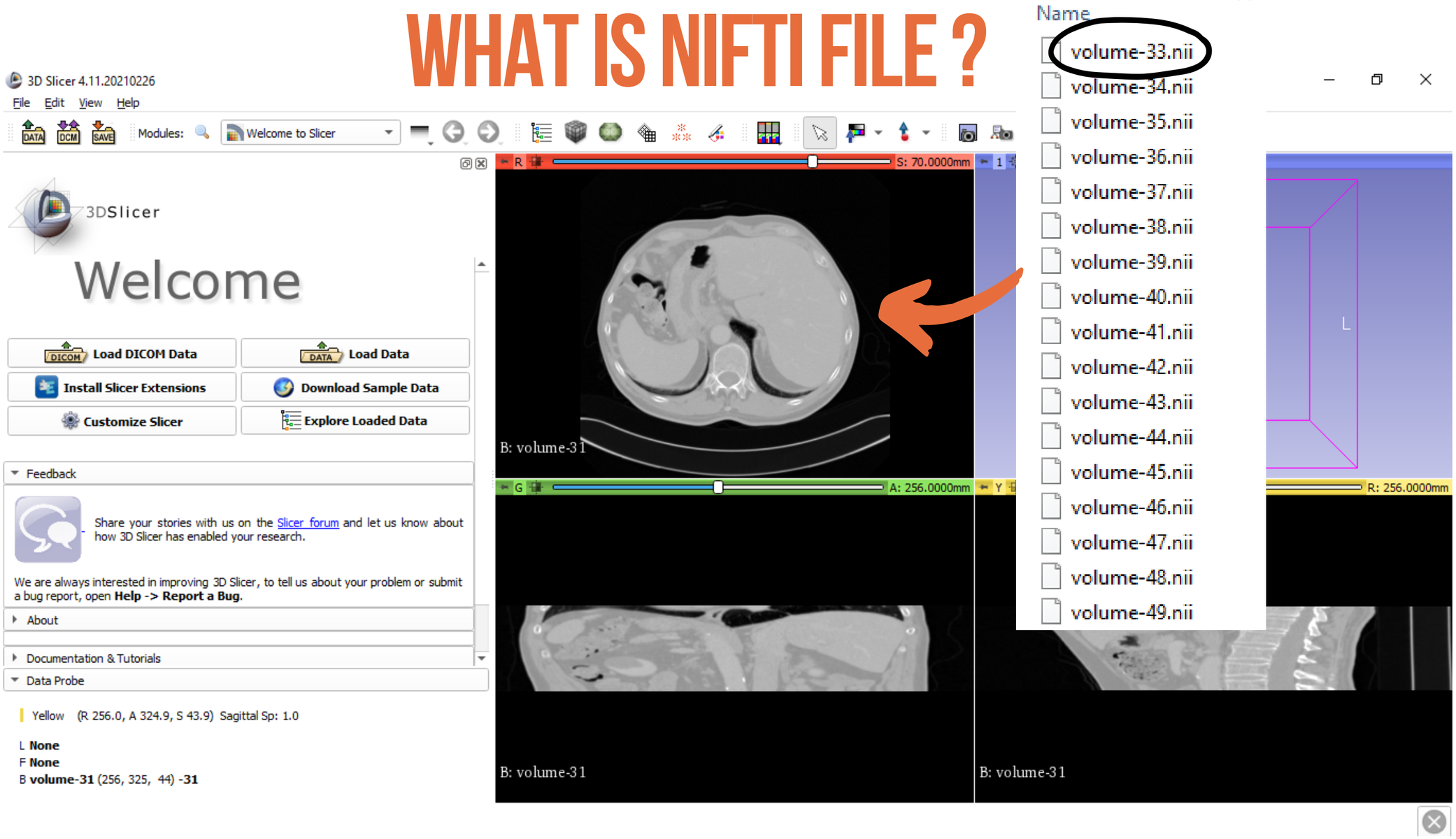


```python
from glob import glob
import shutil
import os
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split

labels_path = 'D:/Liver Segmentation/Kaggle/nifti_raw/segmentations'
images_path = 'D:/Liver Segmentation/Kaggle/nifti_raw/volume'
data = {'Labels': glob(labels_path + '/*'),
        'Images': glob(images_path + '/*')}
df = pd.DataFrame(data)

Images = df[['Images']]
Labels = df['Labels']
TrainImages, ValImages, TrainLabels, ValLabels = train_test_split(Images, Labels, test_size=0.06,
random_state=6305093)

def move_directory(path_name, path, destination):
    os.mkdir(destination)
    for sorce in path:
        patient_name = os.path.basename(os.path.normpath(sorce))
        letter = [char for char in patient_name]
        num = [int(s) for s in letter if s.isdigit()]
        string = map(str, num)
        index = ''.join(string)
        os.mkdir(os.path.join('D:\Liver Segmentation\Kaggle\Dicom_sep', path_name, 'liver_'+ index))
        shutil.move(sorce, destination)
```

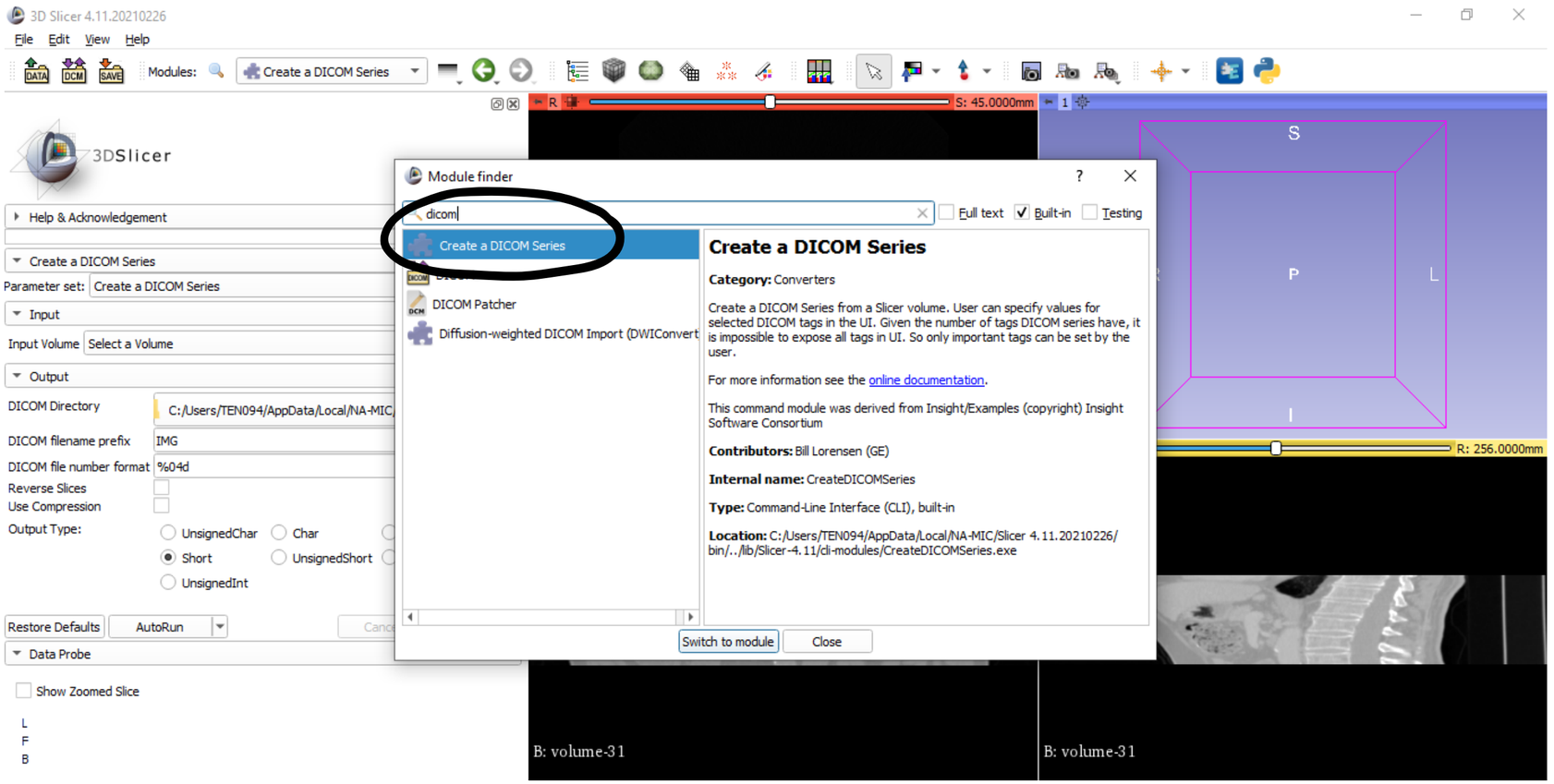
```
move_directory('TrainImages',TrainImages['Images'], 'D:/Liver
Segmentation/Kaggle/nifti_sep/TrainImages')

move_directory('ValImages',ValImages['Images'], 'D:/Liver Segmentation/Kaggle/nifti_sep/ValImages')

move_directory('TrainLabels',TrainLabels, 'D:/Liver Segmentation/Kaggle/nifti_sep/TrainLabels')

move_directory('ValLabels',ValLabels, 'D:/Liver Segmentation/Kaggle/nifti_sep/ValLabels')
```



WHAT IS NIFTI FILE ?

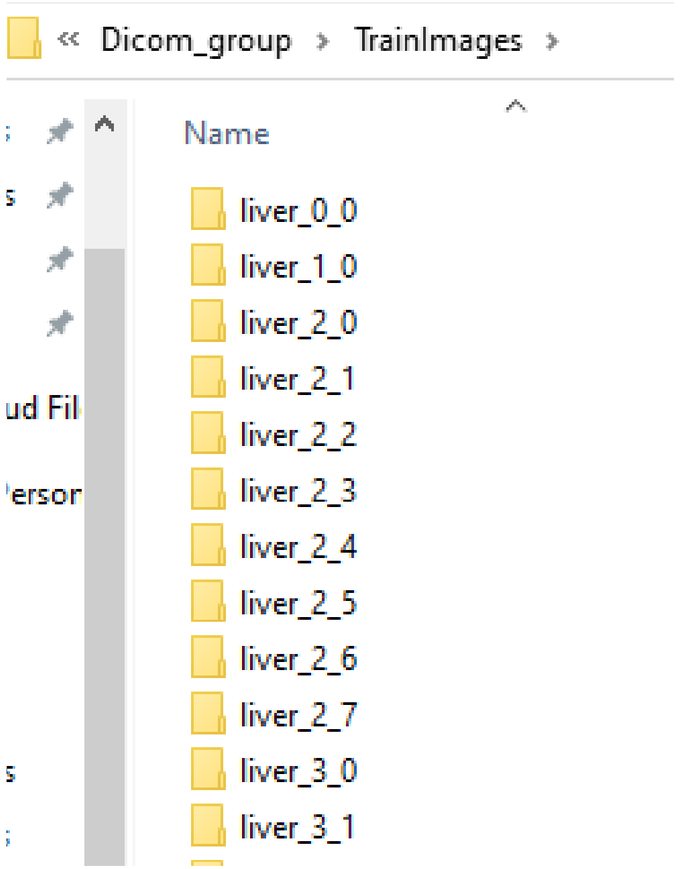


CONVERT NIFTIES INTO DICOM FILES



MOVE EVERY 65 SLICES INTO A FOLDER

```
for patient in glob(in_path + '/*'):
    patient_name = os.path.basename(os.path.normpath(patient))
    number_folders = int(len(glob(patient + '/*'))/65)

    for i in range(number_folders):
        output_path_name = os.path.join(out_path, patient_name + '_' + str(i))
        os.mkdir(output_path_name)
        for i, file in enumerate(glob(patient + '/*')):
            if i == 65 +1:
                break
            shutil.move(file, output_path_name)
```



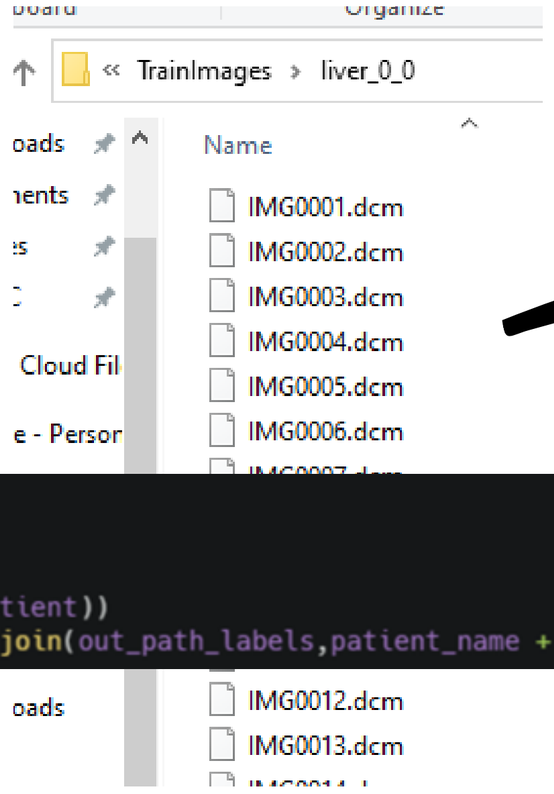
RECONVERT THE GROUPS OF 65 SLICES INTO NIFTI FILES

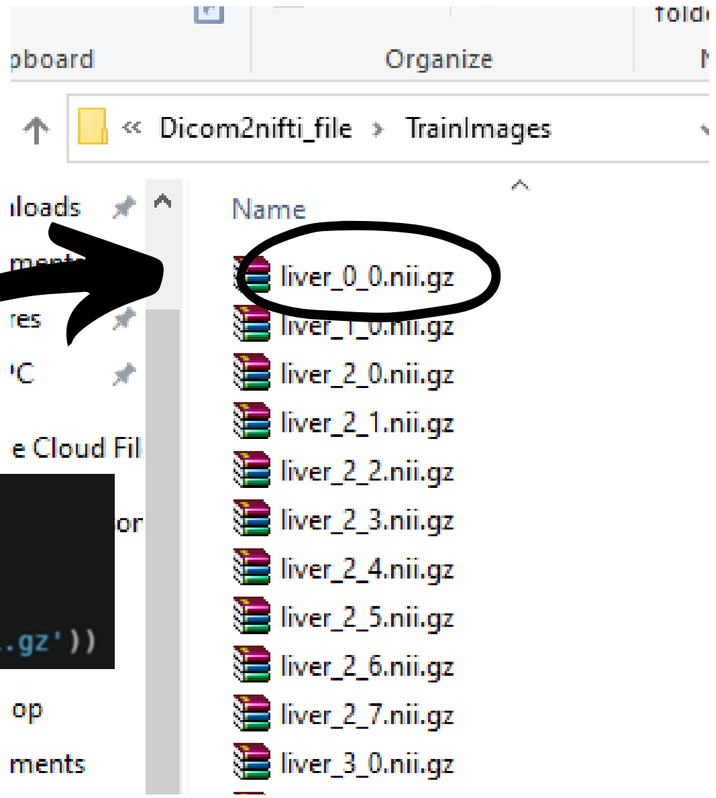
PACKAGE : DICOM2NIFTI

dicom2nifti 2.3.2

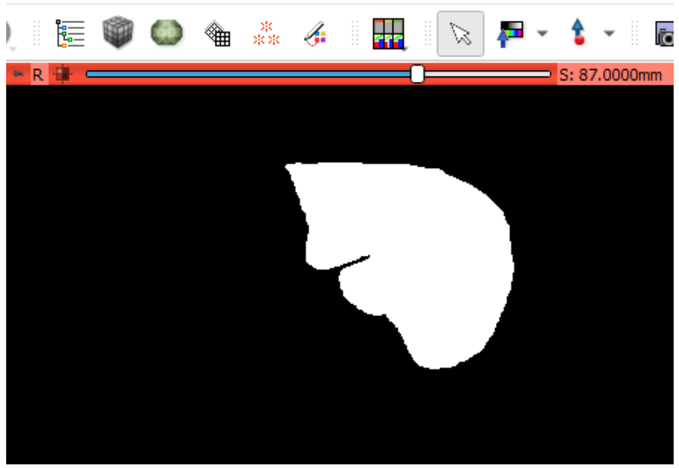
```
pip install dicom2nifti
```

```
for patient in list_labels:
    patient_name = os.path.basename(os.path.normpath(patient))
    dicom2nifti.dicom_series_to_nifti(patient, os.path.join(out_path_labels,patient_name + '.nii.gz'))
```





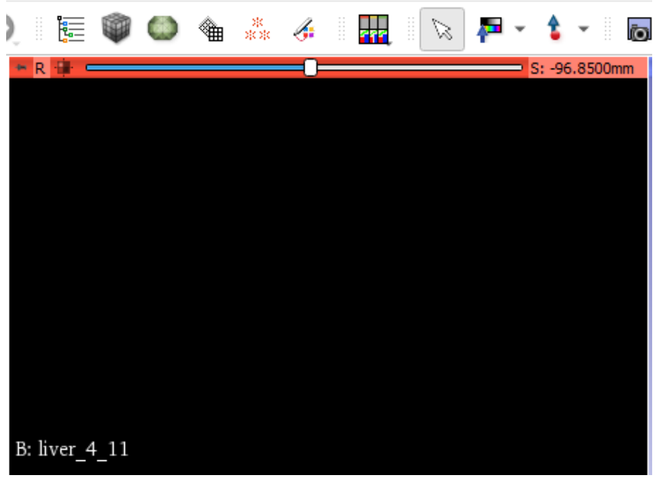
DELETE THE EMPTY FILE



LEN(NP_UNIQUE) != 1

NOT EMPTY

```
for patient in list_labels:
    nifti_file = nib.load(patient)
    fdata = nifti_file.get_fdata()
    np_unique = np.unique(fdata)
    if len(np_unique) == 1:
        shutil.move(patient, Output_bin_path)
```



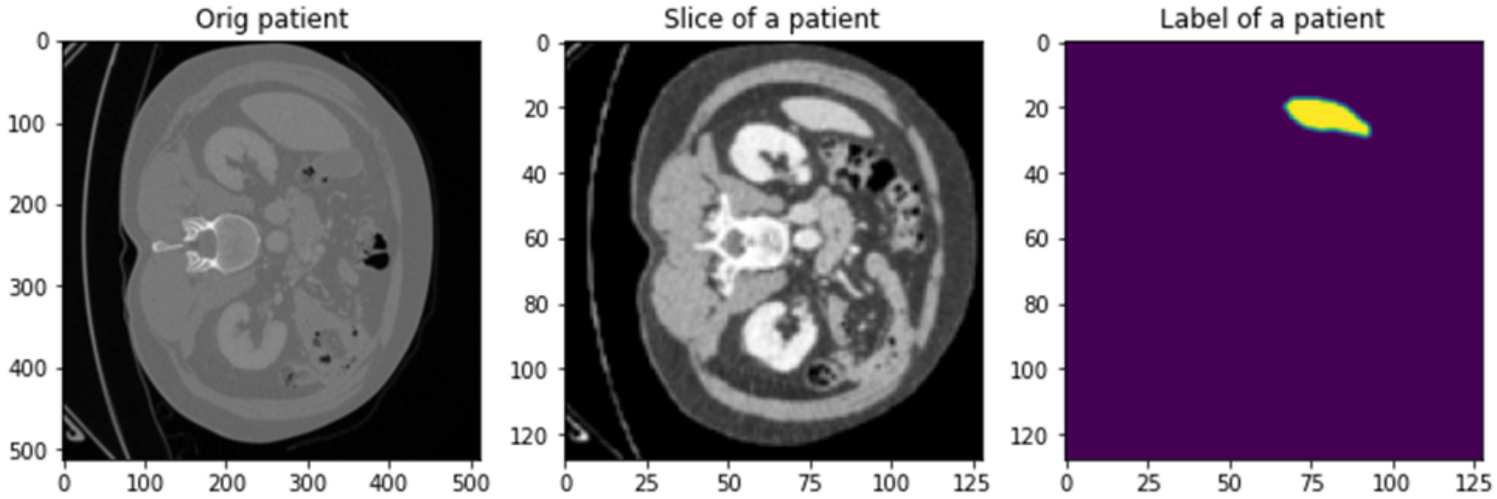
EMPTY FILE

LEN(NP_UNIQUE) == 1

ปรับแต่งข้อมูลก่อนนำไปTRAIN

PREPOCESSING

*STEP 2:



```
def prepare(in_dir, pixdim=(1.5, 1.5, 1.0), a_min=-200, a_max=200, spatial_size=[128,128,64],
cache=False):
    set_determinism(seed=0)

    path_train_volumes = sorted(glob(os.path.join(in_dir, "TrainVolumes", "*.nii.gz")))
    path_train_segmentation = sorted(glob(os.path.join(in_dir, "TrainSegmentation", "*.nii.gz")))

    path_test_volumes = sorted(glob(os.path.join(in_dir, "TestVolumes", "*.nii.gz")))
    path_test_segmentation = sorted(glob(os.path.join(in_dir, "TestSegmentation", "*.nii.gz")))

    train_files = [{"vol": image_name, "seg": label_name} for image_name, label_name in
zip(path_train_volumes, path_train_segmentation)]
    test_files = [{"vol": image_name, "seg": label_name} for image_name, label_name in
zip(path_test_volumes, path_test_segmentation)]

    train_transforms = Compose(
        [
            LoadImaged(keys=["vol", "seg"]),
            AddChanneld(keys=["vol", "seg"]),
            Spacingd(keys=["vol", "seg"], pixdim=pixdim, mode=("bilinear", "nearest")),
            Orientationd(keys=["vol", "seg"], axcodes="RAS"),
            ScaleIntensityRanged(keys=["vol"], a_min=a_min, a_max=a_max, b_min=0.0, b_max=1.0,
clip=True),
            CropForegroundd(keys=["vol", "seg"], source_key="vol"),
            Resized(keys=["vol", "seg"], spatial_size=spatial_size),
            ToTensord(keys=["vol", "seg"]),

        ]
    )

    test_transforms = Compose(
        [
            LoadImaged(keys=["vol", "seg"]),
            AddChanneld(keys=["vol", "seg"]),
            Spacingd(keys=["vol", "seg"], pixdim=pixdim, mode=("bilinear", "nearest")),
            Orientationd(keys=["vol", "seg"], axcodes="RAS"),
            ScaleIntensityRanged(keys=["vol"], a_min=a_min, a_max=a_max,b_min=0.0, b_max=1.0,
clip=True),
            CropForegroundd(keys=['vol', 'seg'], source_key='vol'),
            Resized(keys=["vol", "seg"], spatial_size=spatial_size),
            ToTensord(keys=["vol", "seg"]),
        ]
    )
```

Compose() เป็นฟังก์สำหรับรวมการเปลี่ยนแปลงทั้งหมดแล้วนำไปใช้กับรูปภาพในครั้งดียว

- ***ADDCHANNELD***- เพิ่มแชนแนลสำหรับแยกพื้นหลังกับพื้นที่ระบาย
- ***LOADIMAGED***- เป็นการโหลดข้อมูลมาก่อนเริ่มปรับแต่ง
- ***RESIZED***- ปรับขนาดรูปภาพ*(รูปภาพในที่นี้หมายถึงกลุ่มของสไลด์)*
- ***TOTENSORD***- แปลงเป็น***TENSOR***เตรียมสำหรับขั้นตอน***TRAIN***
- ***SPACINGD***- ใช้สำหรับปรับ***VOXEL SPACING***ให้เท่ากันทั้งหมด
- ***ORIENTATIOND***- หมุนรูปภาพ
- ***SCALEINTENSITYRANGED***- ทำสองอย่างด้วยกันคือปรับสีภาพให้ชัดขึ้นและกำหนดค่าความเข้มแสงให้อยู่ระหว่าง ***0*** ถึง ***1***
- ***CROPFOREGROUNDD***- ตัดส่วนว่างที่ไม่จำเป็นออกจากรูปภาพจะเห็นได้ว่าข้างหลังฟังก์จะลงท้ายด้วยตัว***D***เพื่อใช้กับข้อมูล***DICTIONARY***

TRAIN / TEST THE MODEL

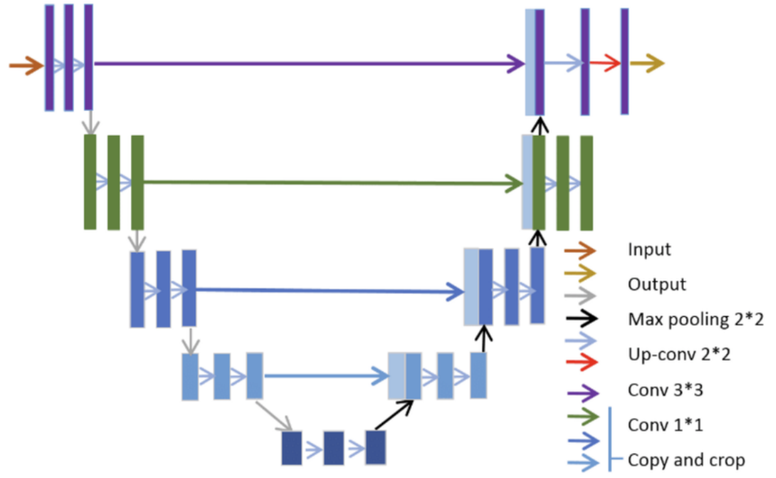
*STEP 3:

1. The U-Net Model
2. Loss Function
3. Train/Test the Model

THE U-NET MODEL *FROM MONAI*

```
model = UNet(
    dimensions=3,
    in_channels=1,
    out_channels=2,
    channels=(16, 32, 64, 128, 256),
    strides=(2, 2, 2, 2),
    num_res_units=2,
    norm=Norm.BATCH,
).to(device)
```

UNET คือหนึ่ง **NEURAL NETWORK ARCHITECTURE** ที่พัฒนามาจาก **CONVOLUTIONAL NEURAL NETWORK** แบบดั้งเดิม เพื่อวิเคราะห์โรคหรือระบุจุดของความผิดปกติในภาพถ่าย โดยรูปแบบสถาปัตยกรรมของ **NETWORK** นี้ มีรูปแบบที่ย่อและขยายข้อมูลในแต่ละ **LAYERS** ลักษณะคล้ายรูปตัว **U**



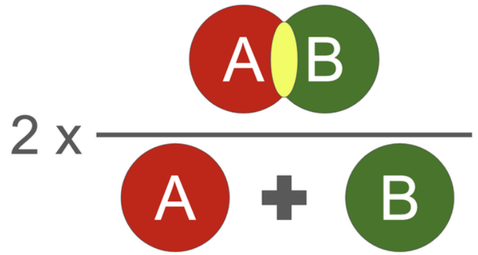
UNET ถูกนำมาแก้ปัญหานี้โดยเฉพาะ เพราะว่าเทคนิคนี้ สามารถระบุพื้นที่และแยกเส้นขอบได้ดี ในระดับ **PIXEL SPACE** จึงเหมาะการโจทย์ **IMAGE SEGMENTATION** หรือ **IMAGE CLASSIFICATION** โดย **INPUT** และ **OUTPUT** ของ **UNET** เป็นขนาดเท่ากัน

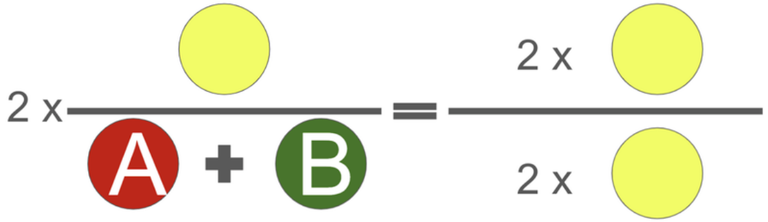
LOSS FUNCTION

DICE LOSS

FROM MONAI

DICE





ทางด้านการเทรนของโมเดลที่เราสร้างคือการหา ***DICE COEFFICIENT*** หรือเรียกอีกอย่างนึงคือ ***F1 SCORE*** สามารถเขียนได้เป็นสมการ โดย ***TP*** คือ ***TRUE POSITIVE FP*** คือ ***FALSE POSITIVE***และ ***FN*** คือ ***FALSE NEGATIVE***

$$Dice = \frac{2 \times TP}{(TP + FP) + (TP + FN)}$$

```
loss_function = DiceLoss(to_onehot_y=True, sigmoid=True, squared_pred=True)
```

DICE LOSS = 1 - DICE

หลักการคือยิ่งภาพมีการเหมือนกันมากค่าจะเข้าใกล้ ***1*** ยิ่งภาพที่เรา ***PREDICT*** ออกมาจากภาพ ***MRI*** มีความคล้ายภาพที่ ***LABEL DICE COEFFICIENT*** จะมีค่ามาก ส่วน ***DICE LOSS*** คือการเอา ***1*** ไปลบกับ ***DICE COEFFICIENT*** ยิ่งค่าเข้าใกล้***0***แสดงว่า***MODEL***ที่หามามีประสิทธิภาพดี ในงานนี้เราจะใช้ฟังก์ชัน ***DICELOSS()*** ในการหา ***DICE LOSS***

TRAIN/TEST THE MODEL

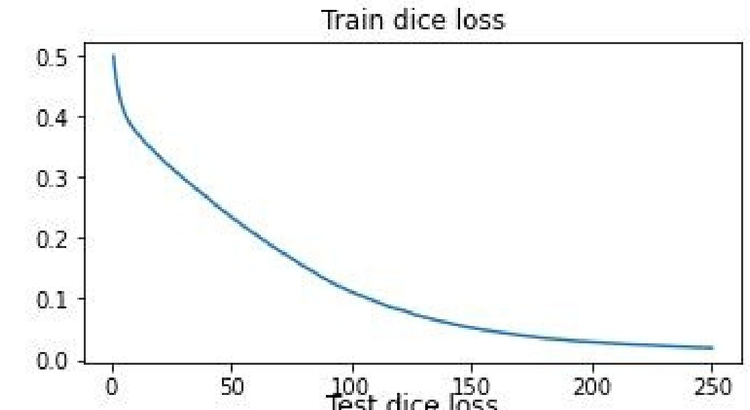
In 1 epoch

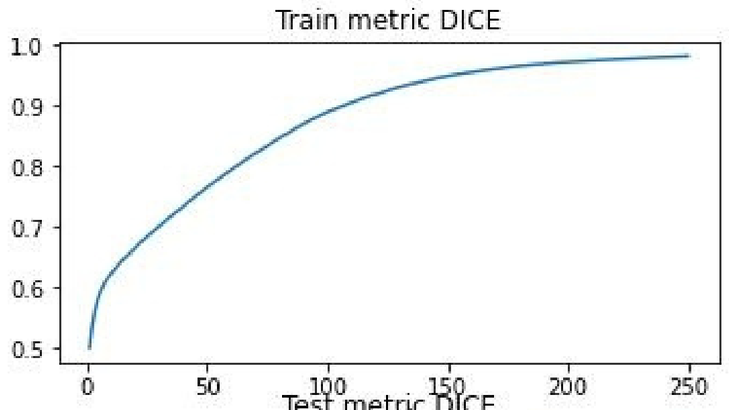
- Start Training
- Run Through every Train data
  - In each Train data
    - Seperate Train image and label
    - Find Train prediction using model
    - Find loss and dice of prediction and label of Train
    - Combine loss and dice to the old value
- Find avg loss and avg matrix of Train
- Save avg loss and matrix of train in directory
- go into if statement every epoch
  - Stop Training
  - Run Through every Test data
    - In each Test data
      - Seperate Test image and label
      - Find Test prediction using model
      - Find loss and dice of prediction and label of Test
      - Combine loss and dice to the old value
  - Find avg loss and avg matrix of Test
  - Save avg loss and avg matrix of test in directory
  - if avg matrix of test>best metrix
    - save the value over best matrix
    - save the model in directory over old model

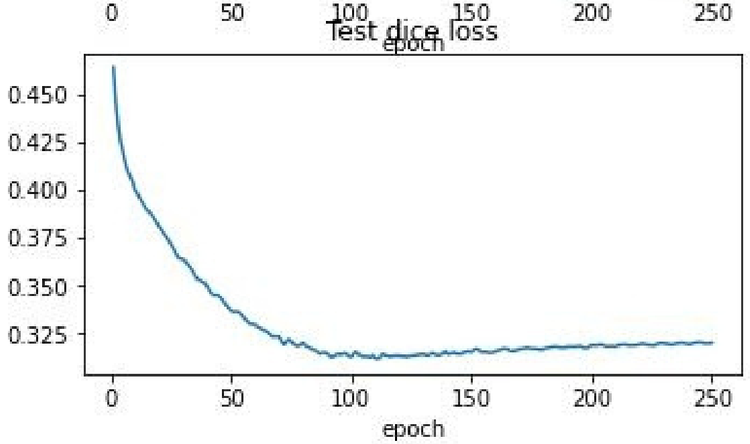
```
106/110, Train_loss: 0.0626
Train_dice: 0.9374
107/110, Train_loss: 0.0324
Train_dice: 0.9676
108/110, Train_loss: 0.0358
Train_dice: 0.9642
109/110, Train_loss: 0.0096
Train_dice: 0.9904
110/110, Train_loss: 0.0099
Train_dice: 0.9901
--------------------
Epoch_loss: 0.0191
Epoch_metric: 0.9809
test_loss_epoch: 0.3199
test_dice_epoch: 0.8971
current epoch: 250 current mean dice: 0.9195
best mean dice: 0.9147 at epoch: 248
train completed, best_metric: 0.9147 at epoch: 248
```

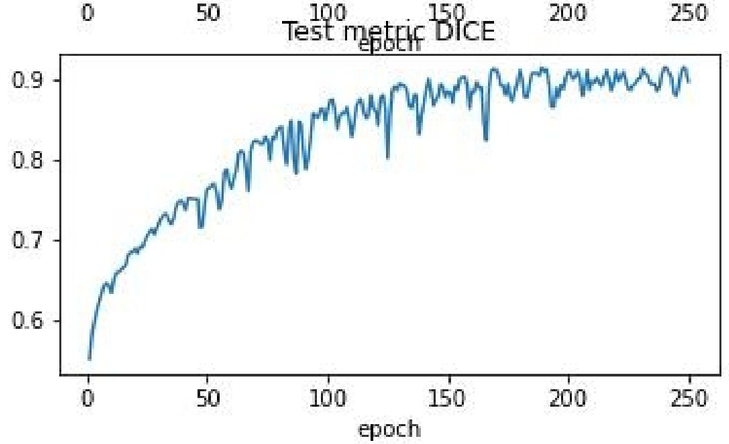
RESULT

*STEP 4:









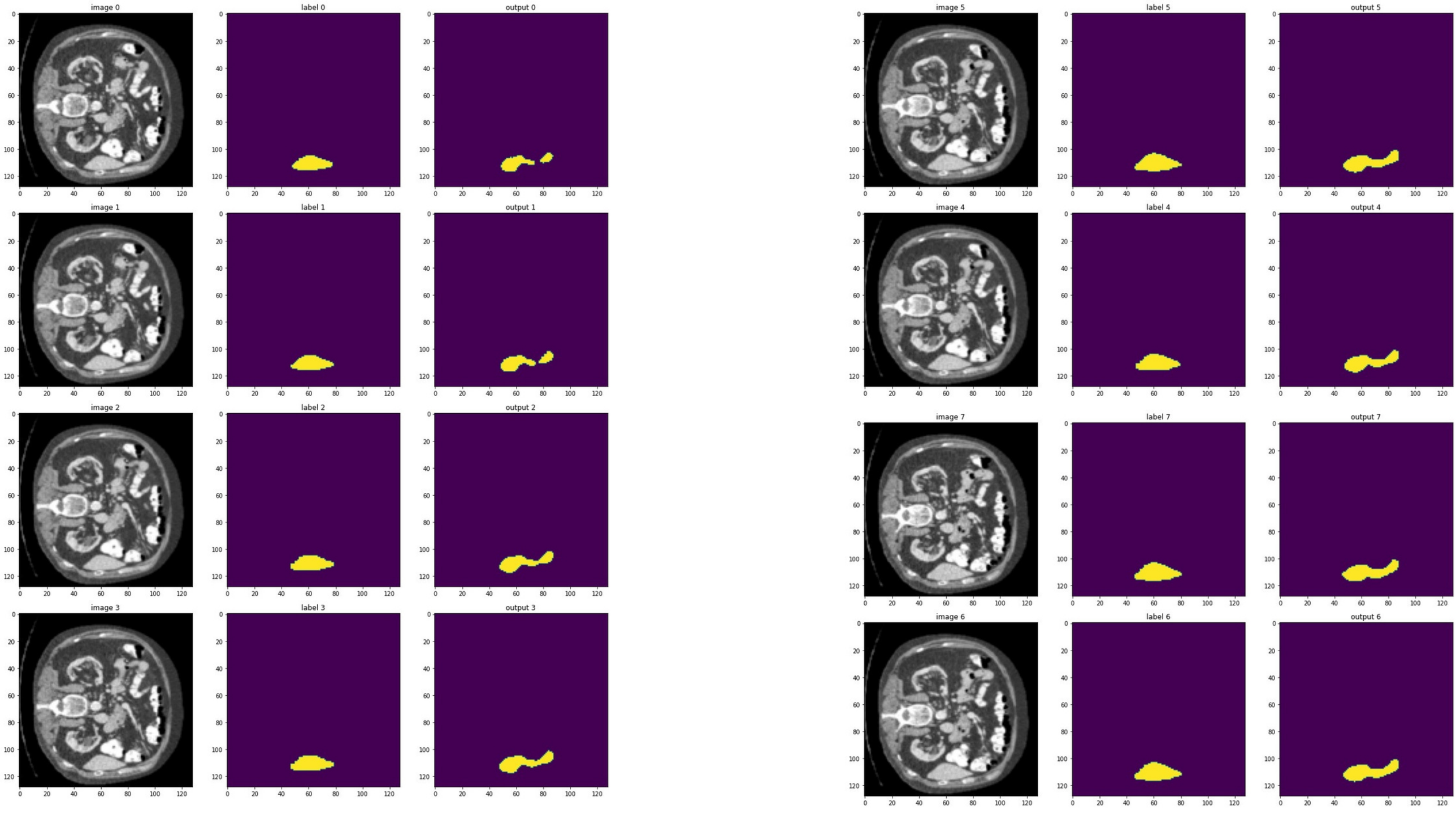


image 0
label 0
output 0
image 1
label 1
output 1
image 2
label 2
output 2
image 3
label 3
output 3
image 5
label 5
output 5
image 4
label 4
output 4
image 7
label 7
output 7
image 6
label 6
output 6

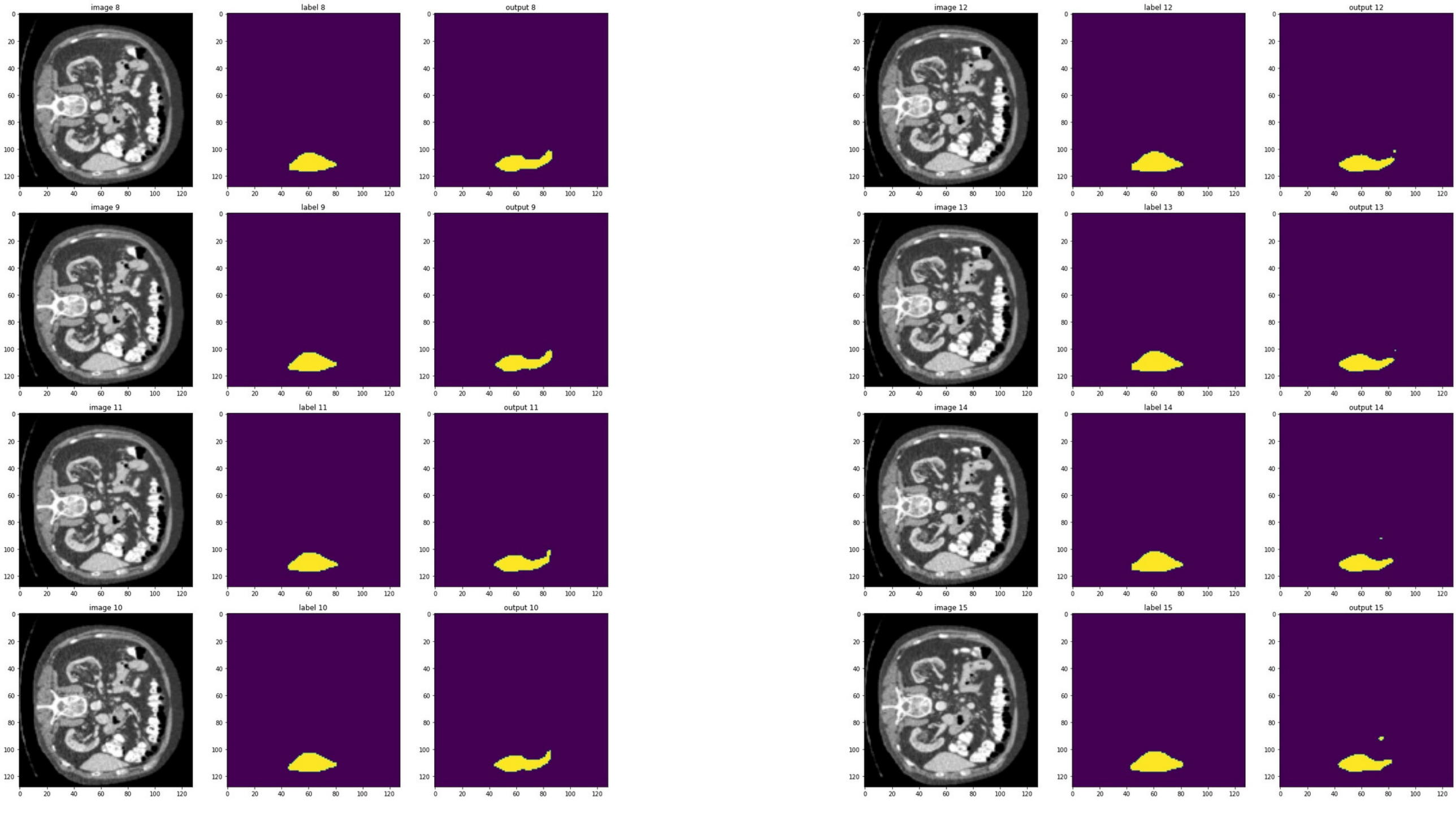


image 8
label 8
output 8
image 12
label 12
output 12
image 9
label 9
output 9
image 13
label 13
output 13
image 11
label 11
output 11
image 14
label 14
output 14
image 10
label 10
output 10
image 15
label 15
output 15

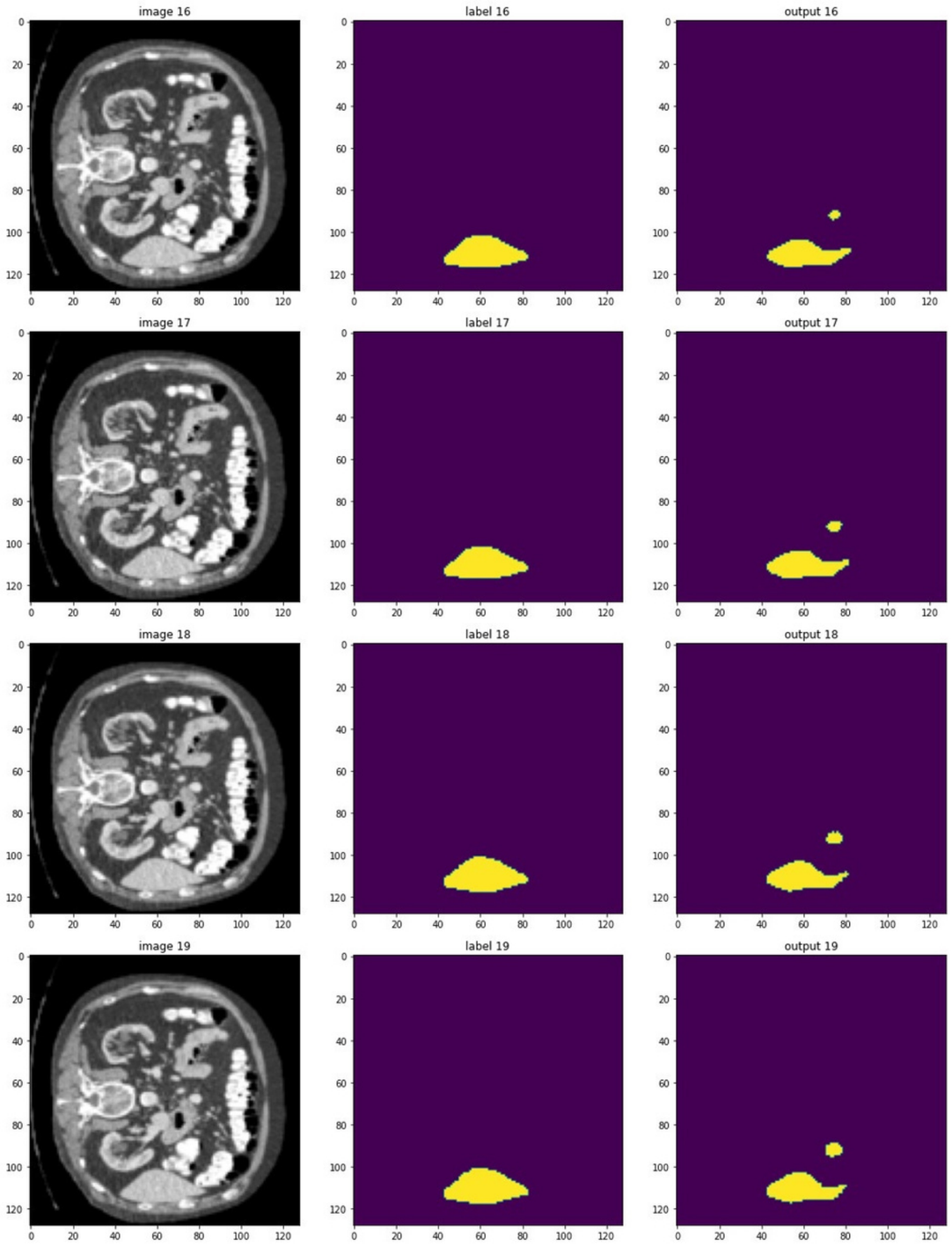


image 16
label 16
output 16
image 17
label 17
output 17
image 18
label 18
output 18
image 19
label 19
output 19

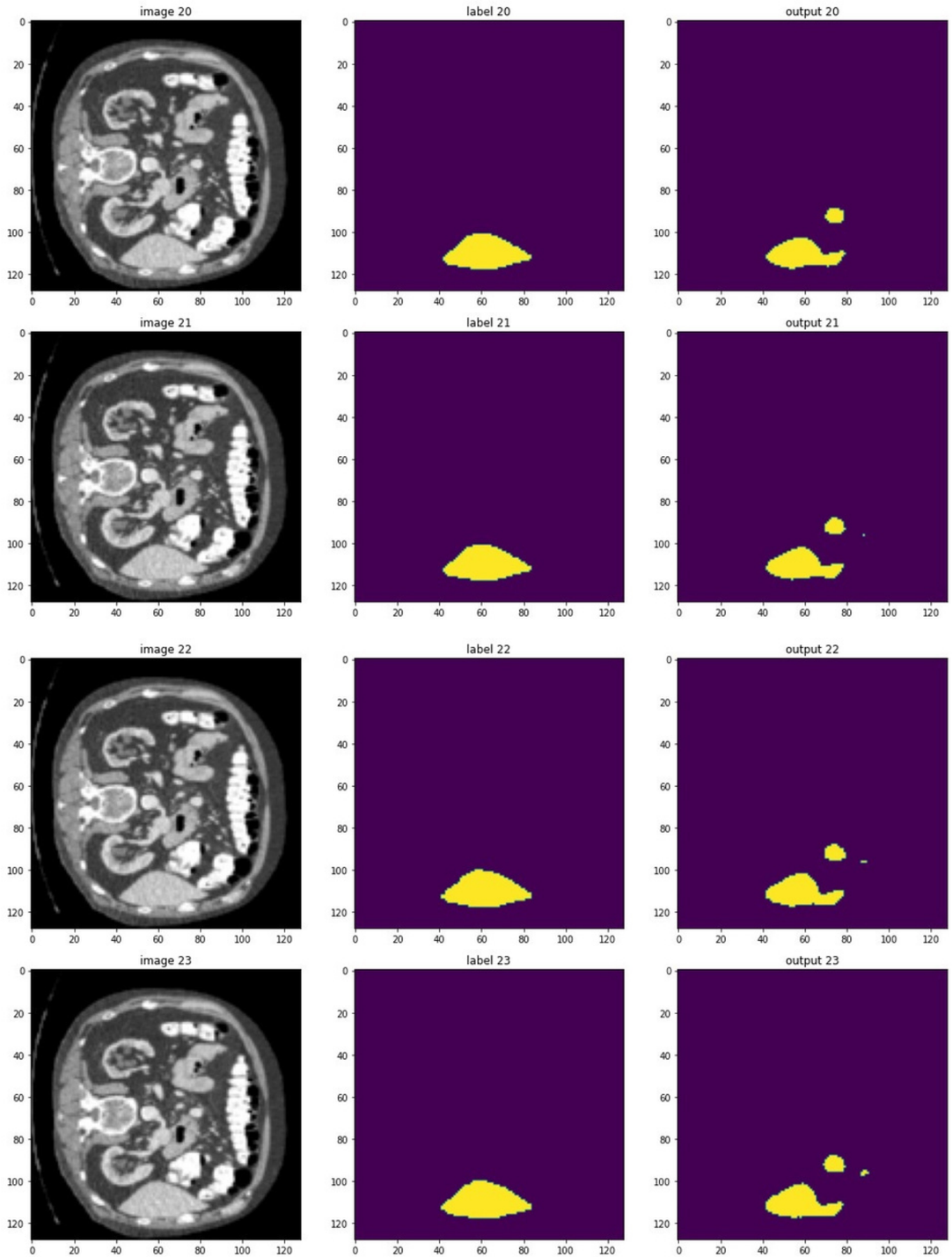


image 20
label 20
output 20
image 21
label 21
output 21
image 22
label 22
output 22
image 23
label 23
output 23

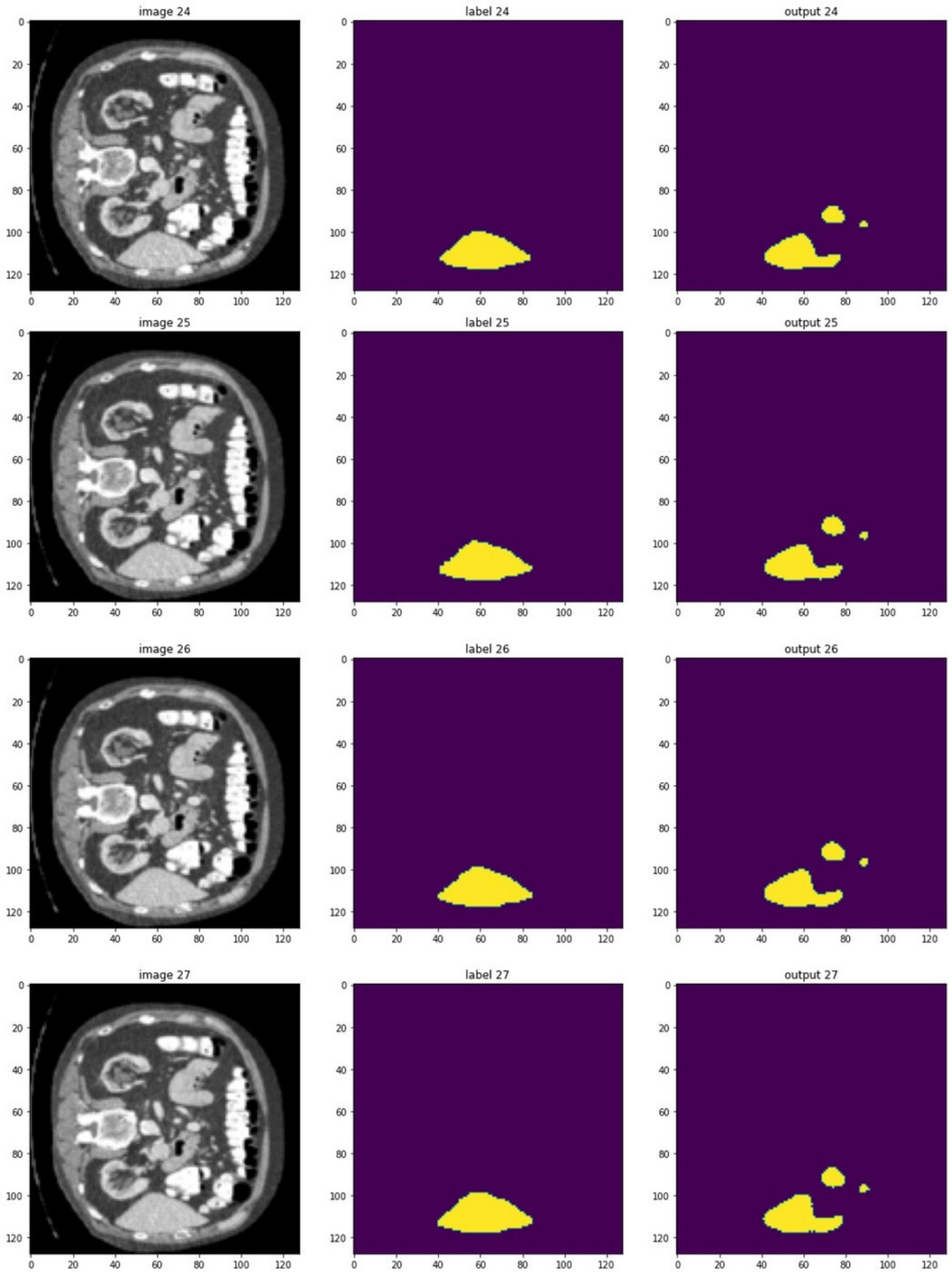


image 24
label 24
output 24
image 25
label 25
output 25
image 26
label 26
output 26
image 27
label 27
output 27

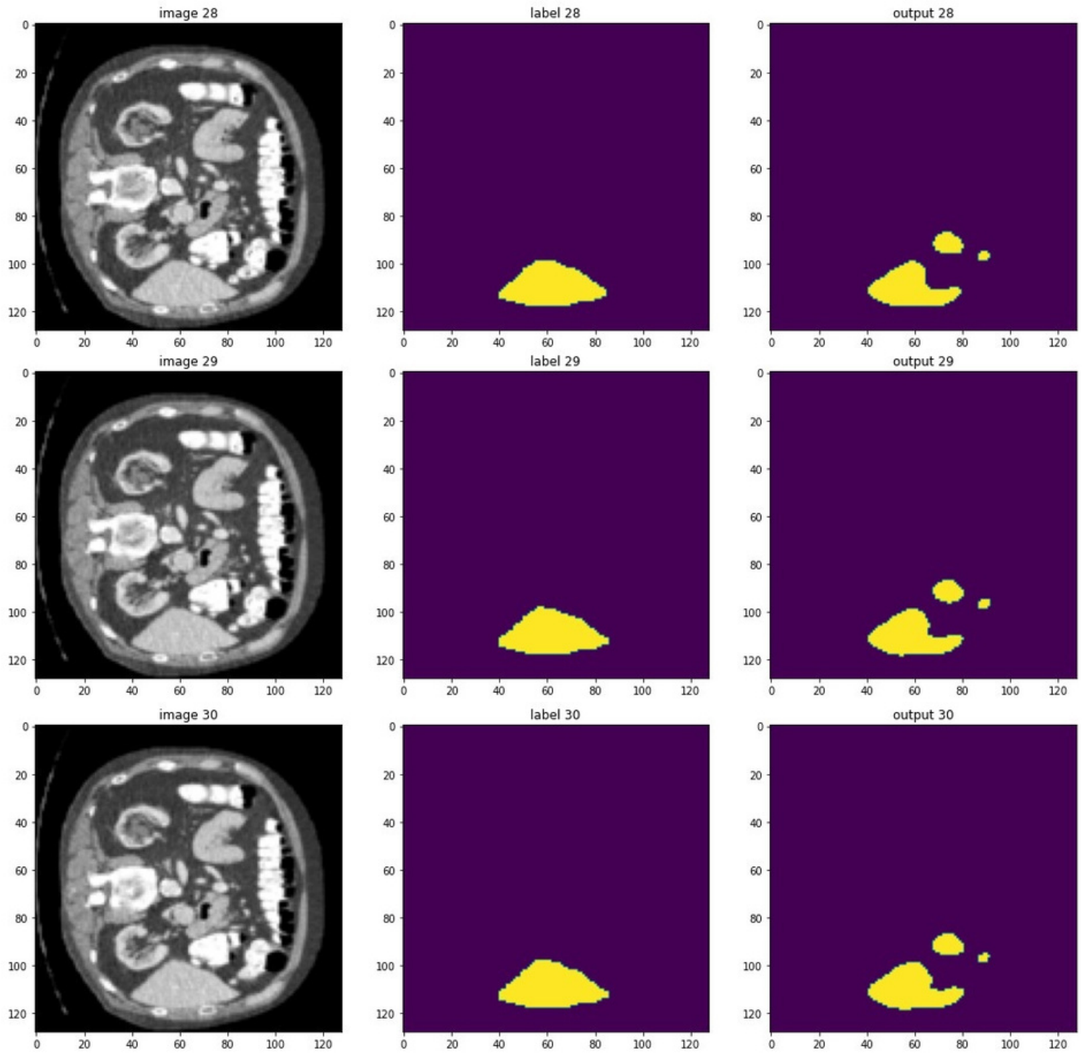


image 28
label 28
output 28
image 29
label 29
output 29
image 30
label 30
output 30

FIRST PAGE

BIBLIOGRAPHY

MOKHTARI, M. E. (2022, April 19). **Preprocessing 3D Volumes for Tumor Segmentation Using Monai and PyTorch**. From pycad: https://pycad.co/preprocessing-3d-volumes-for-tumor-segmentation-using-monai-and-pytorch/

MOKHTARI, M. E. (2022, April 14). **Preprocessing 3D Volumes for Tumor Segmentation using PyTorch and Monai | Part 1/2**. From pycad: https://www.youtube.com/watch?v=83FLt4fPNGs&list=PLQCkKRar9trOubxZm_gfqWmggW4MgVeqq

MOKHTARI, M. E. (2022, April 14). **Preprocessing 3D Volumes for Tumor Segmentation using PyTorch and Monai | Part 2/2.** Retrieved from pycad: https://www.youtube.com/watch?v=hqgZuatm8eE&list=PLQCkKRar9trOubxZm_gfqWmggW4MgVeqq&index=3

MOKHTARI, M. E. (2022, April 14). **Automatic Liver Segmentation Using PyTorch and Monai**. Retrieved from pycad: https://www.youtube.com/watch?v=AU4KlXKKnac&list=PLQCkKRar9trODKvJr2B2A3P2m-9Wk_ziF

Pycad. (2022, April 19). **The Difference Between Dice and Dice Loss**. From pycad: https://pycad.co /the-difference-between-dice-and-dice-loss/

SECOND PAGE

BIBLIOGRAPHY

Wikipedia. (2022, April 19). **U-net**. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/U-Net

สุรพงษ์ กนกทิพย์สถาพร. (14 เมษายน 2565). **Confusion Matrix คืออะไร Metrics คืออะไร Accuracy, Precision, Recall, F1 Score ต่างกันอย่างไร – Metrics ep.1**. เข้าถึงได้จาก BUA Labs: https://www.bualabs.com/archives/1 9 6 8 / what-is-confusion-matrix-what-is-metrics-accuracy-precision-recall-f1-score-difference-metrics-ep-1/

สุรพงษ์ กนกทิพย์สถาพร. (14 เมษายน 2565). **พัฒนาโปรแกรม AI การแพทย์ วินิจฉัยภาวะปอดรั่ว (Pneumothorax) อัตโนมัติ จากฟิล์ม X-Ray โดยใช้ Machine Learning, Deep Neural Network – Image Segmentation ep.2**. เข้าถึงได้จาก BUA Labs: https://www.bualabs.com/archives /1 9 6 8 / what-is-confusion-matrix-what-is-metrics-accuracy-precision-recall-f1 -score-difference-metrics-ep-1/

DO YOU HAVE

ANY QUESTIONS?

Feel free to make this an open discussion for questions or clarifications before proceeding.