LUNG CANCER NODULE DETECTION & CLASSIFICATION USING DEEP CNN

Project Guide: Dr. R. Malmathanraj, Dept. ECE, NIT Trichy

Manoj Thiraviam Kumar (108114053)
Nagaraj Archak (108114060)
Vijay Ravi (108114104)

PREPROCESSING

The CT scan image data requires preprocessing before it is fed to the convolutional neural network (CNN). Each images are 512*512*X, where X is the number of slices present in the CT scan of the patient and it depends upon the resolution of the CT scanner. The images are heterogeneous in nature and therefore need to be preprocessed before being fed to the CNN. The following are the steps being followed for preprocessing CT scan DICOM images:

- Obtaining Slice Thickness
- Convert pixels to Hounsfield Units (HU)
- Resampling
- Normalisation
- Zero Centering
- Segmentation

DEPENDENCIES

```
1   import pydicom                          # Read DICOM Files
2   import os                               # Do Directory Operations
3   import pandas as pd                     # Data Analysis
4   import scipy.ndimage                    # Image Processing Dependency
5   import matplotlib.pyplot as plt         # PLotting Graphs & Images
6   import numpy as np                      # Array Operations
```

These are the dependencies that need to be imported in order to implement the respective functions.

SLICE THICKNESS

```
17  def SliceThick(path):
18      slices = [pydicom.read_file(path + '/' + s) for s in os.listdir(path)]
19      slices.sort(key = lambda x: float(x.ImagePositionPatient[2]))
20      try:
21          slice_thickness = np.abs(slices[0].ImagePositionPatient[2] - slices[1].ImagePositionPatient[2])
22      except:
23          slice_thickness = np.abs(slices[0].SliceLocation - slices[1].SliceLocation)
24
25      for s in slices:
26          s.SliceThickness = slice_thickness
27
28      print "Obtaining Slice Thickness...Done."
29
30      return slices
```

data_dir = 'FYP/stage1/'

patients = os.listdir(data_dir)

path = data_dir + patient

Image Position: ['-177.500000', '-177.500000', '-324.109985']

Slice Location: "-324.109985"

Here, Image Position and Slice Location are the metadata that come inbuilt with the dataset.

HOUNSFIELD UNITS

```
33  def HU(Slices):
34      image = np.stack([s.pixel_array for s in Slices])
35      image = image.astype(np.int16)
36
37      image[image == -2000] = 0
38
39      for slice_number in range(len(Slices)):
40
41          intercept = Slices[slice_number].RescaleIntercept
42          slope = Slices[slice_number].RescaleSlope
43
44          if slope != 1:
45              image[slice_number] = slope * image[slice_number].astype(np.float64)
46              image[slice_number] = image[slice_number].astype(np.int16)
47
48          image[slice_number] += np.int16(intercept)
49
50      print "Converting to Hounsfield Units...Done."
51
52      return np.array(image, dtype=np.int16)
```

Substance	HU
Air	−1000
Lung	−500
Fat	−100 to −50
Water	0
CSF	15
Kidney	30
Blood	+30 to +45
Muscle	+10 to +40
Grey matter	+37 to +45
White matter	+20 to +30
Liver	+40 to +60
Soft Tissue, Contrast	+100 to +300
Bone	+700 (cancellous bone) to +3000 (cortical bone)

Hounsfield Units (HU) is also called CT Numbers and it is a quantitative scale for describing radio density. CT scanners have cylindrical scanning bounds but the image that they output is a square. The pixels that call outside this bound have a fixed -2000 value. Conversion from pixel data to Hounsfield Units:

$$\mathbf{HU = Data(x , y) \times RS + RI}$$

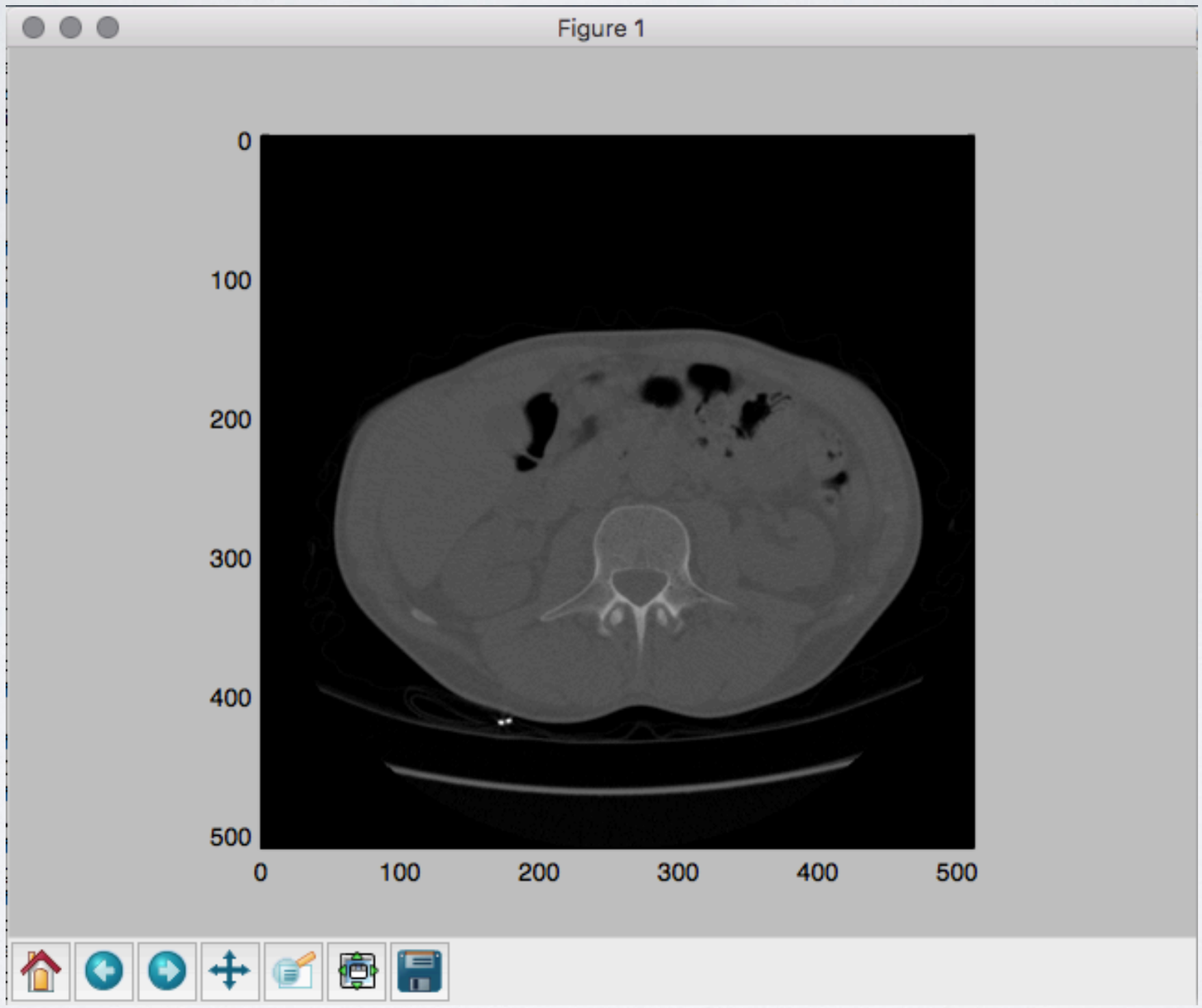
where,

Data(x , y) = Pixel value at coordinates x and y.

RS = Rescale Slope

RI = Rescale Intercept

AFTER CONVERSION TO HU



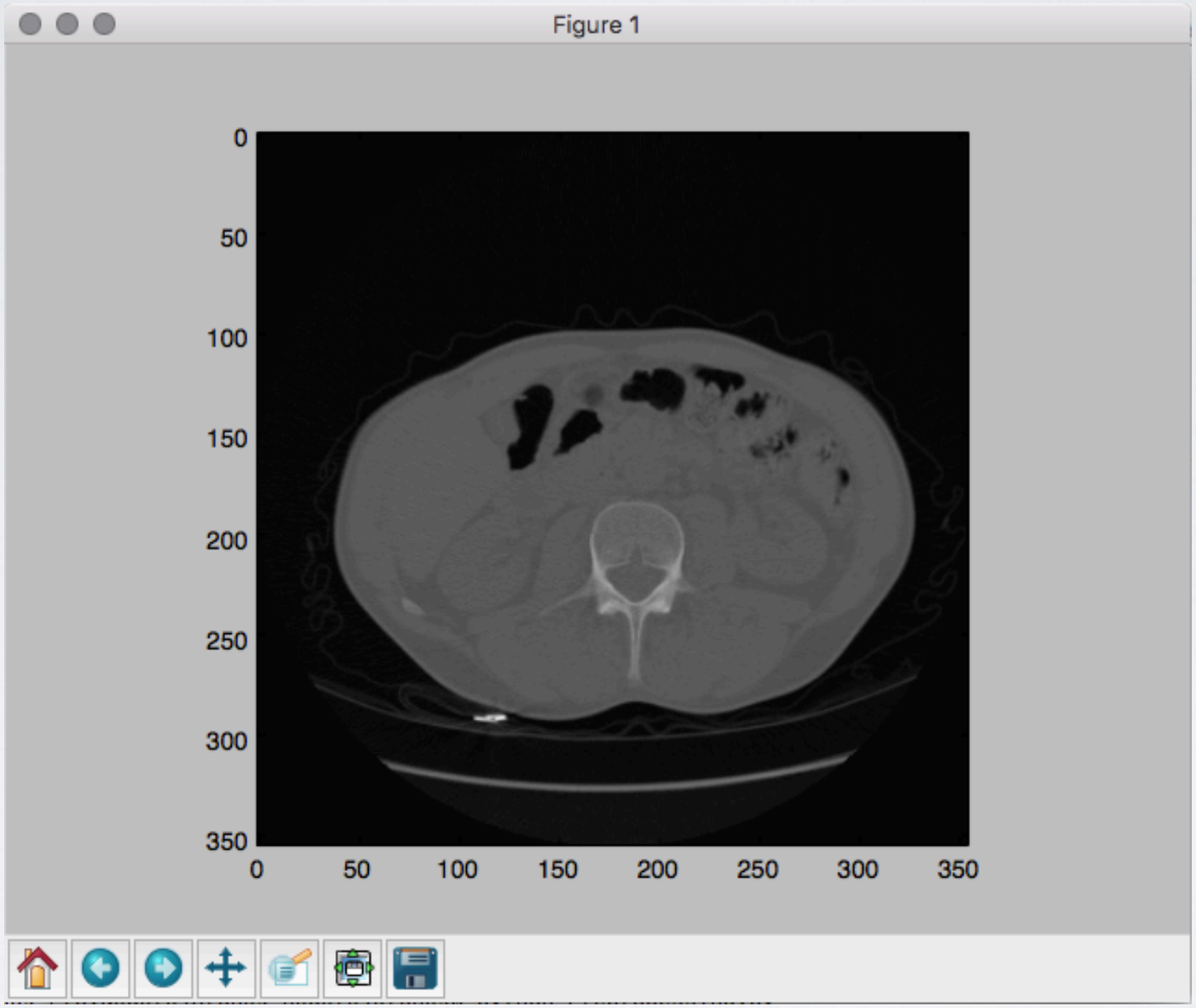
RESAMPLING

```
54  def Resample(image, scan, new_spacing=[1,1,1]):
55
56      spacing_list = [scan[0].SliceThickness]
57      spacing_list.extend(scan[0].PixelSpacing)
58      spacing = np.array(spacing_list, dtype=np.float32)
59
60      resize_factor = spacing / new_spacing
61      new_real_shape = image.shape * resize_factor
62      new_shape = np.round(new_real_shape)
63      real_resize_factor = new_shape / image.shape
64      new_spacing = spacing / real_resize_factor
65
66      image = scipy.ndimage.interpolation.zoom(image, real_resize_factor, mode='nearest')
67
68      print "Resampling...Done."
69
70      return image, new_spacing
```

Different CT scans may have different pixel spacing. One scan may have [2.5, 0.5, 0.5] while another may have [1.5, 0.725, 0.725]. Here 2.5 and 1.5 depict the spacing between slices of CT scans. These value varies from scans of one patient to another, so they'll have to be standardised in order to be accepted by the CNN.

The entire dataset is resampled to an isotropic resolution. All DICOM images will be resampled to 1mm x 1mm x 1mm pixel spacing resolution.

AFTER RESAMPLING



NORMALISATION

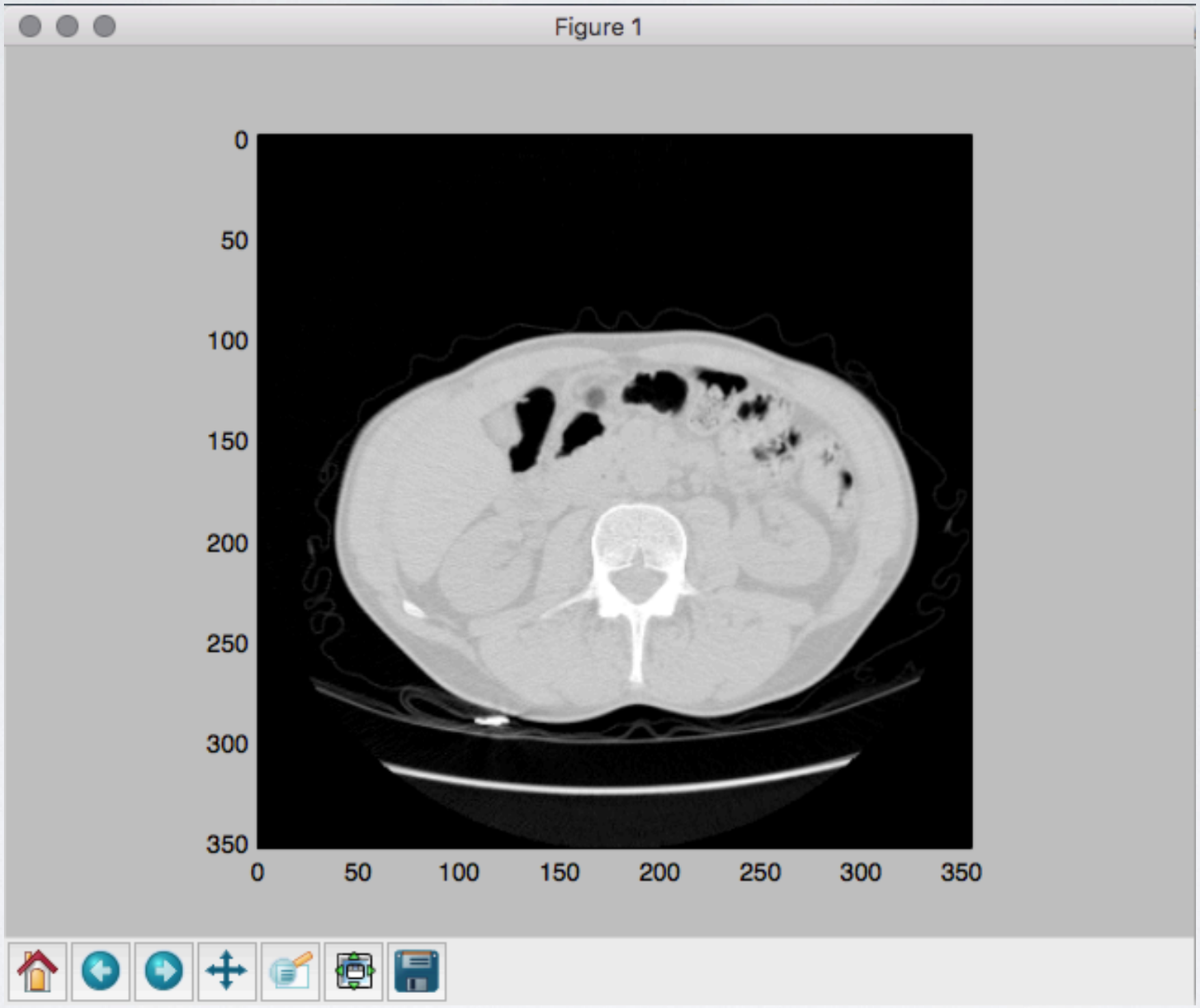
```
72    MIN_BOUND = -1000.0
73    MAX_BOUND = 400.0
74
75    def Normalize(image):
76        image = (image - MIN_BOUND) / (MAX_BOUND - MIN_BOUND)
77        image[image>1] = 1.
78        image[image<0] = 0.
79        return image
```

Normalisation is a process that changes the range of pixel intensity values.

The current pixel Hounsfield Units may range from -1000 to 2000. They will have to be normalised with a upper limit of 400 as anything above that is merely bones and unwanted noise.

Data(x , y) = [Data(x , y) - MinBound] / [MaxBound - MinBound]

AFTER NORMALISING

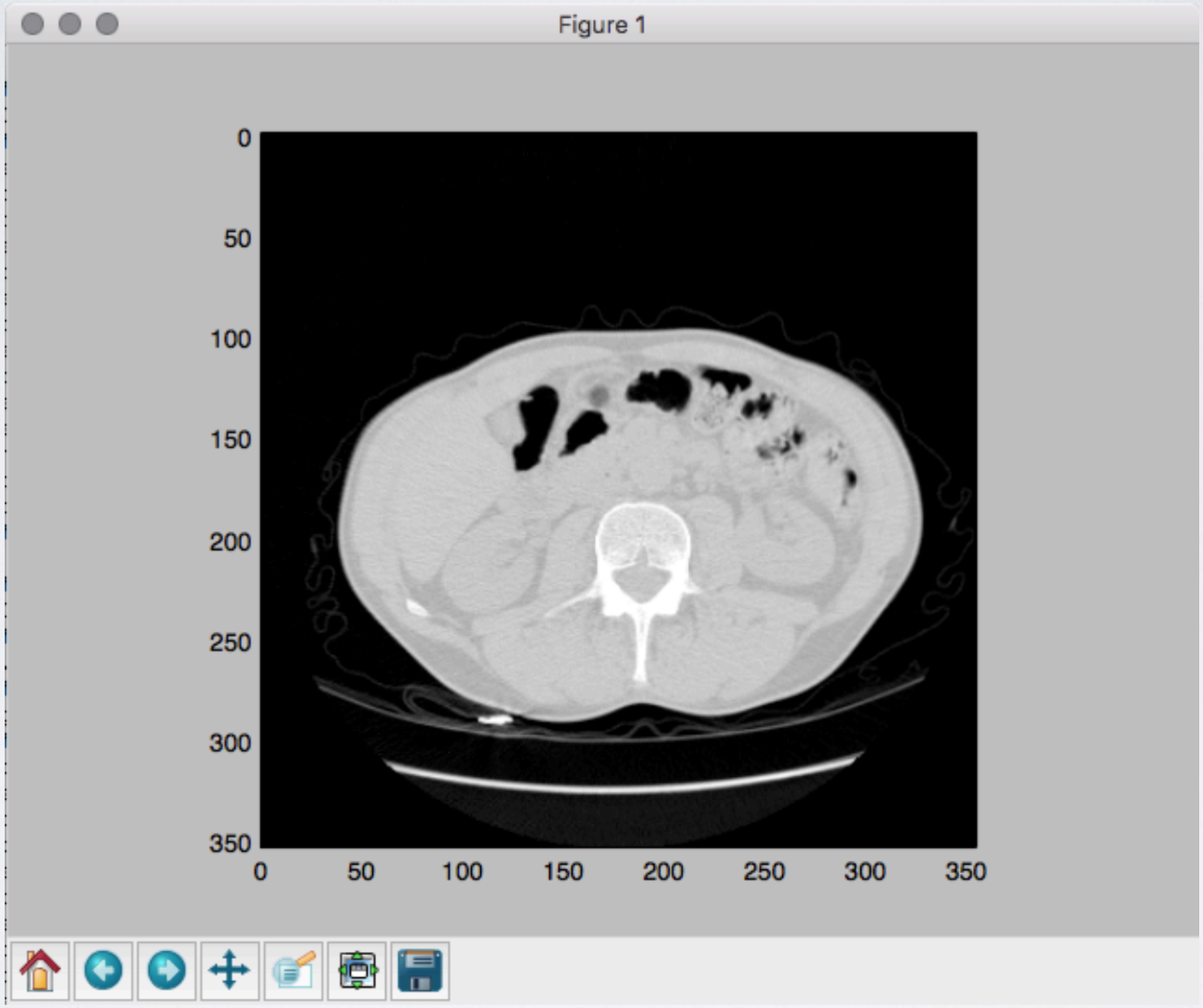


ZERO CENTERING

```
81    PIXEL_MEAN = 0.25
82
83    def zeroCenter(image):
84        image = image - PIXEL_MEAN
85        return image
```

The pixel data will have to be zero centered in order to obtain a zero mean. This is accomplished by simply subtracting PixelMean from pixel Data(x , y). The pixel mean for the Kaggle Data Set is found to be 0.25.

AFTER ZERO CENTERING



FUTURE WORK

Implemented:

- Finding Slice Thickness
- Conversion to HU
- Resampling
- Normalising
- Zero Centering

To Be Implemented:

- Segmentation
- 3D Plotting
- Train CNN

REFERENCES

Literature Review

- Stewart, B.W.K.P. and Wild, C.P., 2017. World cancer report 2014. *Health*.
- Rao, P., Pereira, N.A. and Srinivasan, R., 2016, December. Convolutional neural networks for lung cancer screening in computed tomography (CT) scans. In *Contemporary Computing and Informatics (IC3I), 2016 2nd International Conference on* (pp. 489-493). IEEE.
- Chon, A., Balachandar, N. and Lu, P., 2017. Deep convolutional neural networks for lung cancer detection. tech. rep., Stanford University. (classification AUC of 0.83)
- Wang, Z., Xu, H. and Sun, M., 2017, December. Deep Learning Based Nodule Detection from Pulmonary CT Images. In *Computational Intelligence and Design (ISCID), 2017 10th International Symposium on* (Vol. 1, pp. 370-373). IEEE.
- Dandıl, E., Çakiroğlu, M., Ekşi, Z., Özkan, M., Kurt, Ö.K. and Canan, A., 2014, August. Artificial neural network-based classification system for lung nodules on computed tomography scans. In *Soft computing and pattern recognition (soCPar), 2014 6th international conference of* (pp. 382-386). IEEE.

REFERENCES

Frameworks, Dependencies & Datasets

- https://www.tensorflow.org/versions/r1.2/get_started/mnist/pros
- https://www.tensorflow.org/versions/r1.2/get_started/summaries_and_tensorboard
- Lung Image Database Consortium (https://wiki.cancerimagingarchive.net/display/Public/LIDC-IDRI#cda78258407d41af86614bf0c054cbbc).
- Lung Nodule Detection (https://luna16.grand-challenge.org/home/).
- Kaggle Data Science Bowl 2017 Dataset (https://www.kaggle.com/c/data-science-bowl-2017).