# CS 640 - Artificial Intelligence

## Data Science Bowl 2017

### Final Report

**Team: Rohit Agrawal, Pratikkumar Patel, Wenjun Shen, Tianqi Xu**

## Description:

In the United States, lung cancer strikes 225,000 people every year, and accounts for $12 billion in health care costs. Early detection is critical to give patients the best chance at recovery and survival.

One year ago, the office of the U.S. Vice President spearheaded a bold new initiative, the Cancer Moon Shot, to make a decade's worth of progress in cancer prevention, diagnosis, and treatment in just 5 years.

In 2017, the Data Science Bowl will be a critical milestone in support of the Cancer Moonshot by convening the data science and medical communities to develop lung cancer detection algorithms.

Using a dataset of thousands of high-resolution lung scans provided by the National Cancer Institute, participants will develop algorithms that accurately determine when lesions in the lungs are cancerous. This will dramatically reduce the false positive rate that plagues the current detection technology, get patients earlier access to life-saving interventions, and give radiologists more time to spend with their patients.

Our team first studied how to process the CT scan image in the tutorial and Kaggle kernel. Combining our knowledge in convolutional neural network (CNN), we trained a network that can efficiently distinguishes between lung cancer images and healthy lung images.

## Pre-Processing:

Since the data size of those CT images is really big and brute force calculation without any preprocessing can take inconceivably long, we implemented two different preprocessing methods both of which significantly decreased the amount of computation we will do in the training processing phase. The first method is masking the lung area, which removed the lung area pixel data where nodule won't exist. The second method is reducing the number of the scans and resolution of the scans.

### 1) 3D Lung Masking:

In our first part of pre-processing, we followed the 3D Pre-Processing Kernel from Kaggle which covered the following topics:

- Loading DICOM Files
- Adding Missing Metadata
- Converting Pixel Values to Hounsfield Units (HU)
- Resampling to Isomorphic Resolution & Removing Variance in Scanner Resolution
- 3D-Plotting & Visualization
- Lung Segmentation
- Normalization
- Zero Centering Scans

We first load the CT scans with DICOM format. The standard file format in medical imaging is DICOM. They contain a lot of metadata such as the pixel size etc. This pixel size/coarseness of the scan differs from scan to scan (e.g. the distance between slices may differ), which can hurt performance of CNN approaches. We deal with this problem by doing isomorphic resampling. For each patient folder, we have dozens of CT Scans as DICOM files. The Hounsfield Unit is an important measure

unit of radio density. It is used to describe the accuracy of the CT scan.

Since the raw data is too gigantic to be processed, we need to segment the lungs and other non-cancer tissues and bones out. We have followed below process to do that:

1. We start by reading all the slices for a patient and then sort them by using image position of patient.
2. Next, we get the pixels Hounsfield Units. Following table can be referred for the HU (Source: Wikipedia)
3. In order to get the HU, we first convert the data to INT16. This is possible because the values are low <32k
4. Next, we set the outside-of-scan pixels to 0. The intercept is usually -1024 and hence, the air is approximately 0.
5. Next, we get the intercept and the slope values and check if slope is 1 to properly align it
6. Next step is to segment the lung mask. The original image values are not actually binary but 1 and 2 and hence the first step to segment out the lungs is to convert the image to true binary. In the original image 0 is treated as the background and we don't want that.
7. We pick the pixel in the very corner to determine which label is AIR. But this technique might not work all the time and an improvement could be pick multiple background labels from around the patient more resistant to trays on which the patient lays cutting the air around the person in half.
8. Next, we fill the air around person.
9. We then fill the lung structures which is equivalent of performing a Morphological Closing operation. In order to do this, we determine the largest solid structures and retain the slices which contains the lung parts.
10. We then convert the image to binary and perform inversion so that the lung values become 1.

11. Next step is to remove the air pockets inside the body.
12. Once we segment the lung we project the segmented mask.

Set thresholds to the HU unit will help us differentiate between the cancer region and other tissues.

| 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) |

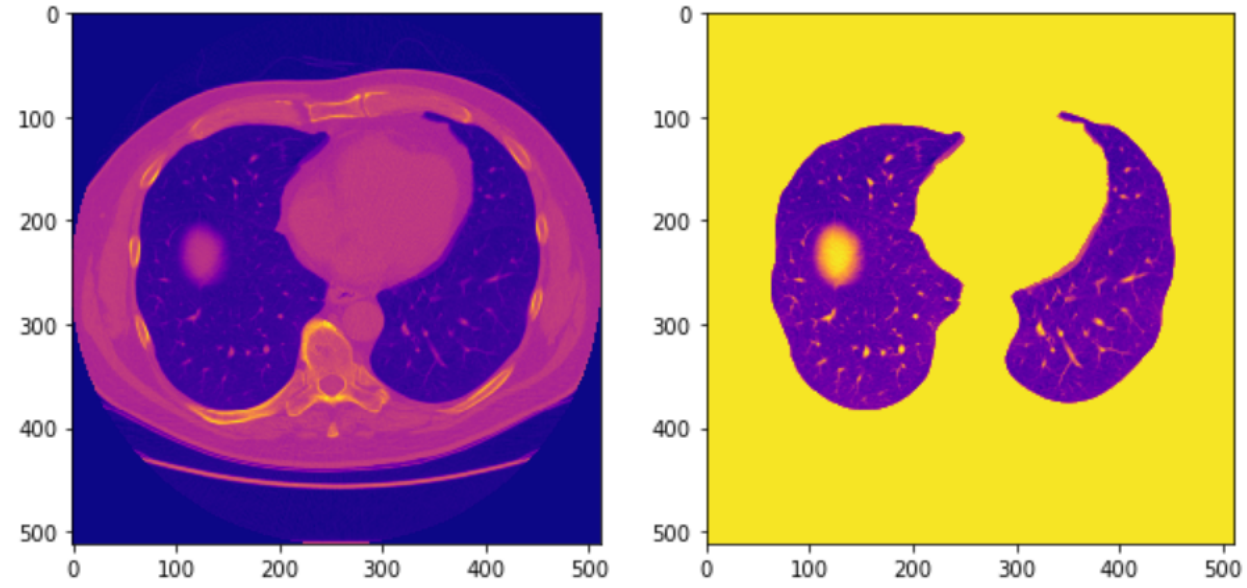
**HU Values Table**

Source: Wikipedia

## Visualization:

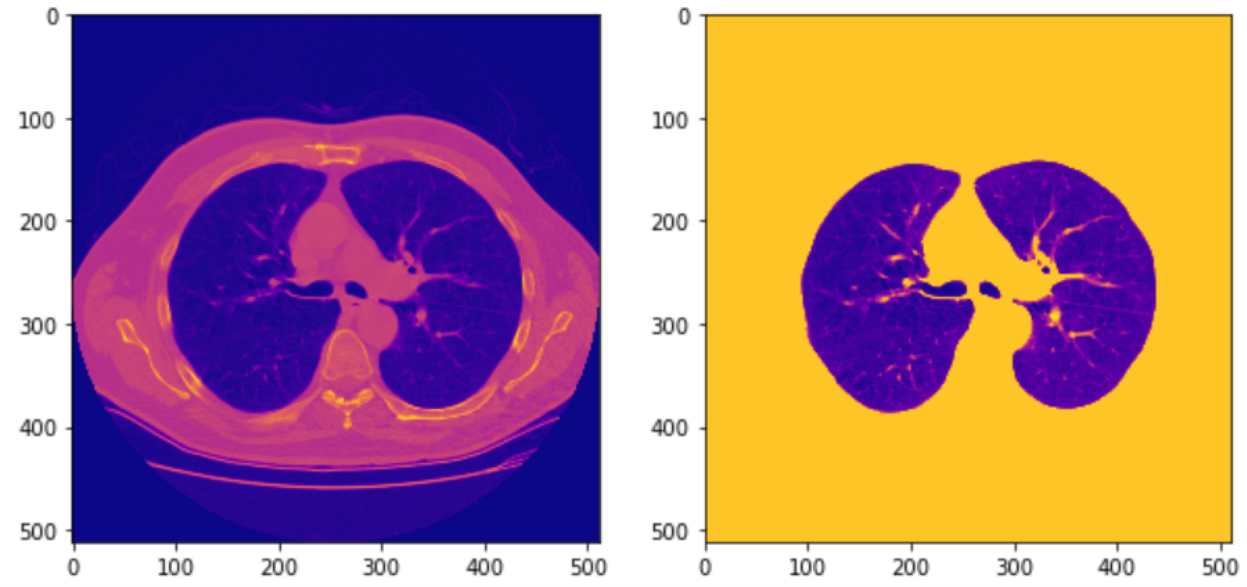
We also visualized the segmentation to help us understand the problem.

```
Patiend ID: 0c59313f52304e25d5a7dcf9877633b1
```



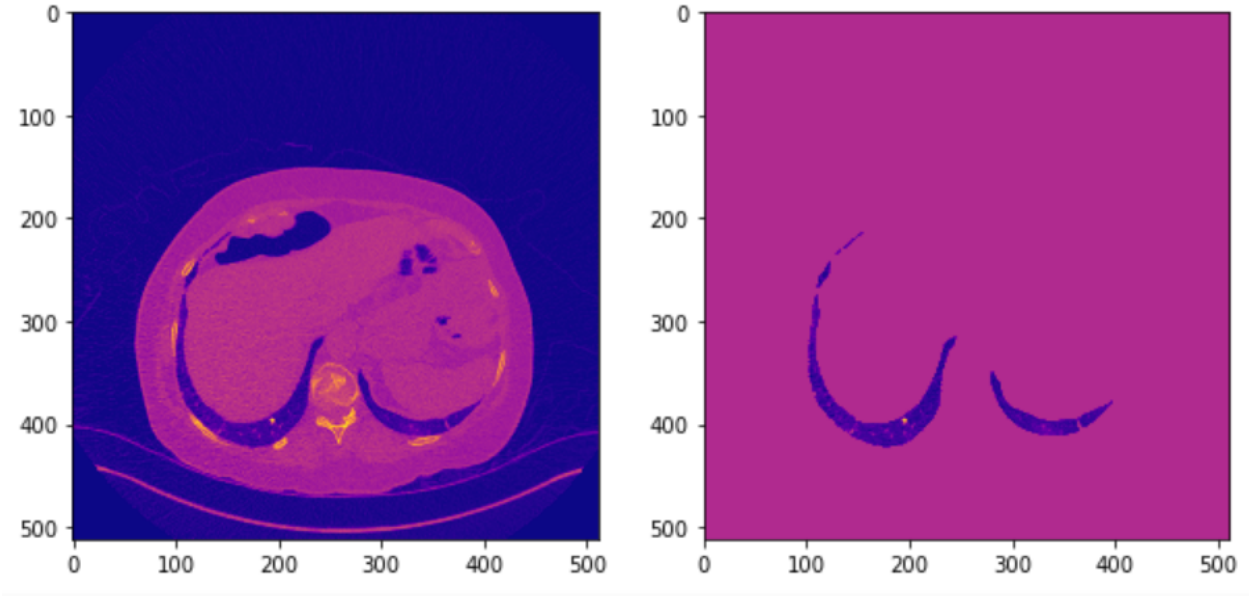

**Cancer Nodule Visible**

```
Patiend ID: 0c37613214faddf8701ca41e6d43f56e
```




**Cancer Nodule Not Visible**

Patiend ID: 0d06d764d3c07572074d468b4cff954f

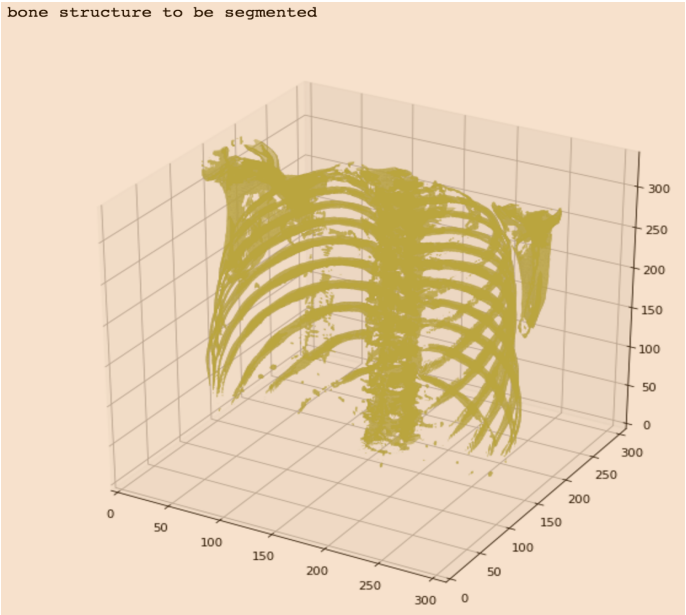


**Lungs Not Segmented**

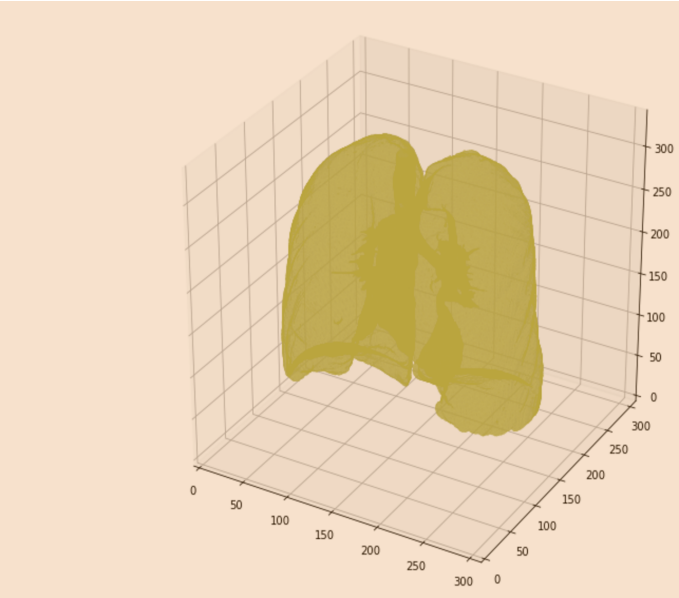
Above sample images are from 3 different patients. We have shown only single slices from their CT scan. A combined view of all slices can show us a 3D picture as shown below.

Here is the 3D picture of the other tissues we need to mask out:

**Bones:**

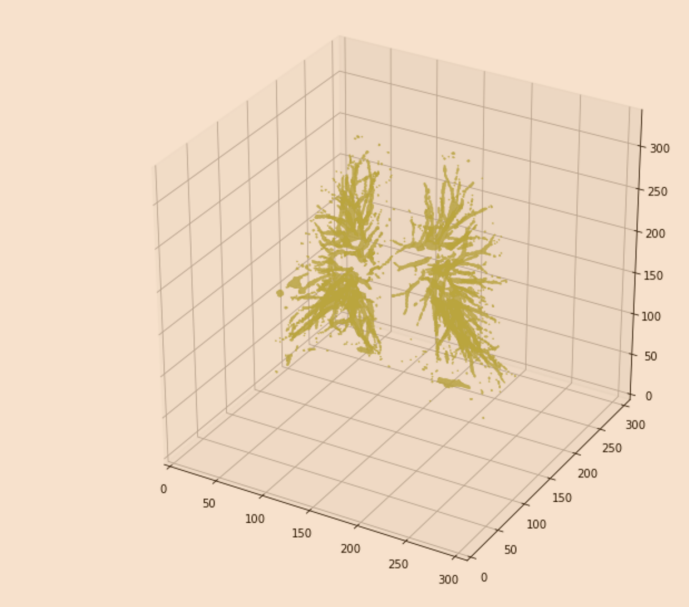


**Lungs:**




**Before Segmentation**

Here is the picture of the lung after the segmentation, and the data size is significantly reduced:




**After Segmentation**

Lastly, we apply the normalization and zero centering to cut out the area of the image. The areas of the lung being cut off are not needed for cancer detection.

## 2) Image Number & Data Resolution Reduction

Here we referred the preprocessing method from the sentdex's kaggle kernel, which reduced the data size from 512 x 512 to 50 x 50. The downsizing still keeps the essence of those images, so it doesn't affect our nodule detection much.

Now we need to solve the inconsistency and the size of the number of CT scans for each patient. Each patient in the original sample has 100+ scans, which is unable to be processed quickly and it is problematic when we pass the data into neural networks if each patient has different number of scans. In this case, we set the number of scans for each patient to 20. To do that, we take the current list of slices and combine them together into a list of slices. And we chunk it into a list of lists of data with the same size.

After thresholding, the color to gray-scale pictures (thresholding the color will make the training easier). In order to train our network, we need to know which image is positive for cancer. We import the **stag1_labels.csv,** which contains all the patient IDs and 1 or 0 indicating whether they have cancers or not, and save the image data along with the label into a NPY file. That's all the steps for one patient, and there are 1595 patients in stage 1 dataset. We iterate that many times and save the processed data into separate folders so that it can be used later.

| 1 | id | cancer |
|---|---|---|
| 2 | 0015ceb851d7251b8f399e39779d1e7d | 1 |
| 3 | 0030a160d58723ff36d73f41b170ec21 | 0 |
| 4 | 003f41c78e6acfa92430a057ac0b306e | 0 |
| 5 | 006b96310a37b36cccb2ab48d10b49a3 | 1 |
| 6 | 008464bb8521d09a42985dd8add3d0d2 | 1 |
| 7 | 0092c13f9e00a3717fdc940641f00015 | 0 |
| 8 | 00986bebc45e12038ef0ce3e9962b51a | 0 |
| 9 | 00cba091fa4ad62cc3200a657aeb957e | 0 |
| 10 | 00edff4f51a893d80dae2d42a7f45ad1 | 1 |
| 11 | 0121c2845f2b7df060945b072b2515d7 | 0 |

## 3) Plotting & Viewing 3D Nodules

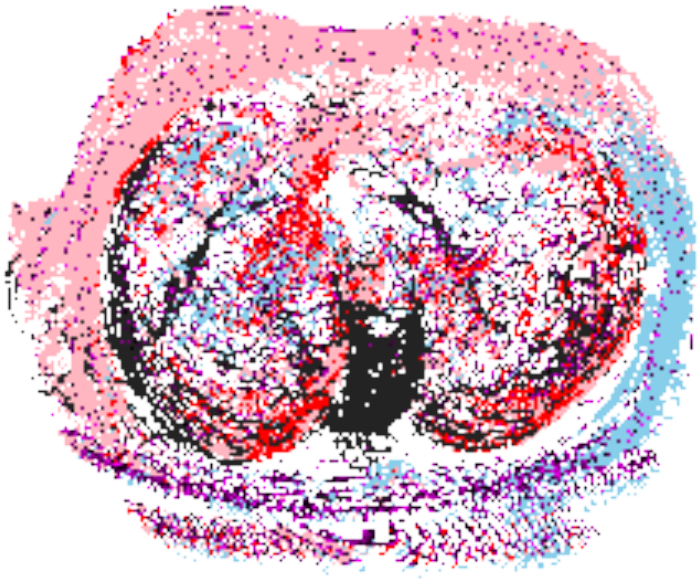
Here are the steps to view the RAW MHD Files in 3D Viewer:

- Download & Install Fiji
- Drag one. MHD file to Fiji status bar
- Click on the new image window. Press "control +" to zoom in, "control -" to zoom out.
- In the Menubar, click "Image > Stacks > Orthogonal Views" to see it. Scroll to go through all slices. Notice that when your cursor moves around in the image window, the Fiji Status Bar shows you the XY coordinates and the value, this value is the same as that in Python NUMPY array.
- In the Menubar, click "Plugins > 3D viewer", in the "Add ..." window, change "Resampling factor" to 1, click "OK", click "OK" to convert to 8-bit.
- Click on the "ImageJ 3D Viewer", In the Menubar, click "Edit > Adjust threshold". Drag threshold bar to around 150.
- Drag the object to rotate it. Hold "shift" and drag to move it. Scroll to zoom.
- In the Menubar, click "View > Start animation" to activate it.
- Click on the image stack, in the Menubar, click "Image > Adjust > threshold", check "Dark background", move the first threshold bar to around -400, click "Apply > OK > Yes".
- Either use step 5 to open another 3D viewer, or click on the current "ImageJ 3D Viewer", in the Menubar click "add > from image" to add another object. Click on the object and "shift" drag to move it. Thresholding on the raw image then creating 3D object gives you different rendering.

## RGB Visualization:

DICOM Standards:
http://dicom.nema.org/standard.html

It is possible to do a RGB Visualization of the Lung from the information captured in the DICOM Images.



**RGBA Visualization**

## Training & Testing Approach:

During our preprocessing of given DCM files, we looked for patient id in given label data and if patient ID was not available, we put processed information in a separate folder that was to be submitted as validation file.

This gave us a set of approx. 1300 files that had label data available. So, we split these files into 80-20 set for training and testing purposes.

In order to train our model, we used Tensorflow. First step was to create a Convolutional Neural Network using the Tensorflow library by feeding in the pre-processed and aligned Stage 1 data. We utilized IVCGPU clusters to train the NN on our data.

Picking 80% of the files and training the 3D convolutional layers all together was exhausting the memory, so we picked 1 3D image at a time and improving the fit using cross entropy on final SoftMax layer. After the model was trained, we used the rest of 20% files to test our model and predict the label and find accuracy of the model. This was done after each epoch to see if accuracy was diverging for training data and testing data (suggesting we have overfitting problem).

This gave us an output of label per image when it came to validation files.

We generated output files after each epoch and saved along with the accuracy we derived from the test data. This approach helped us pick the best submission file for the project.

## Experiments:

Our initial attempt was to build an 8-layer 3D Convolutional Neural Network (CNN). We designed the kernel sizes of each layer to cover information from the whole image. At the last layer, we got one output as prediction, which was the log probability of being cancer. We assess the model performance using sigmoid cross entropy loss.

Another attempt was to modify the 2D Convolutional Networks using Tensorflow, so that it can accept 3D images. We took 3x3x3 convolutions on the initial image, and producing 32 outputs. In the next step, we took 3x3x3 convolutions of the 32 inputs and made 64 outputs. We used 2 max pooling layers with strides size equals to 2 along each axis, so the image size was reduced to 13x13x5 from 50x50x20. At this point, we had 13x13x5 sized images, and 64 of them, we had 13x13x5x64 =54080 inputs. And then we outputted 54080 inputs to 1024 nodes in the fully connected layer. Then, the output layer was 1024 layers, to 2 possible nodes, which were either 1 or 0. We set the learning rate of the optimizer to 0.001 and number of epochs to 10. For each epoch, we ran the optimizer and cost against input data, and outputted the loss. Then, we can find the epoch that minimizes the cost.

We also experimented by creating different other models by changing the parameters like number of nodes, size of images etc.

## Observations:

We found that the Training Data itself was not enough to successfully train the CNN to identify the Cancerous Nodules. In most of the cases, the trained model always ends up predicting 0 which symbolizes NO Cancer. The highest accuracy we achieved is around 76%. The nodules in the lungs were very small to be successfully picked up by the code in order for it to successfully train the CNN to identify them.

## Results:

$$\text{LogLoss} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right],$$

where

- n is the number of patients in the test set
- \\$\hat{y}_i$ is the predicted probability of the image belonging to a patient with cancer
- \\$y_i$ is 1 if the diagnosis is cancer, 0 otherwise
- \\$log$( \\) is the natural $base$ logarithm

Note: the actual submitted predicted probabilities are replaced with \\$max(min(p, 1-10^{-15}$,10^{-15})\\). A smaller log loss is better.

The predictions are scored based on the log loss function as shown in figure above. Although most of the models built by us achieved a score less than the Benchmark, the best result we achieved is slightly better than the 0.5 Benchmark. The final position on the Leaderboard was 307th. We could train our models on LUNA 2016 dataset to improve accuracy since the training data given to us is not significantly sufficient. The current detecting method could be used as an accessory for doctors to decide whether a patient has cancer or not. In medical industry, false negative rate must be extremely low or even be eliminated, because predicting a cancerous patient as non-cancerous can result in a destructive consequence. In future, we should also take into account minimizing false negative rate.

## Credits & Bibliography:

https://www.kaggle.com/arnavkj95/data-science-bowl-2017/candidate-generation-and-luna16-preprocessing

https://www.kaggle.com/anokas/data-science-bowl-2017/exploratory-data-analysis-4

https://www.kaggle.com/mtfall/data-science-bowl-2017/my-idea-for-lung-segmentation

https://www.kaggle.com/armamut/data-science-bowl-2017/getting-the-lungs-right

https://www.kaggle.com/rashmitarouy01/data-science-bowl-2017/ttuparthanrash-data-science-bowl-2017

https://www.kaggle.com/gzuidhof/data-science-bowl-2017/full-preprocessing-tutorial

https://www.kaggle.com/the1owl/data-science-bowl-2017/explore-digital-imaging-and-communications-in-med

https://www.kaggle.com/sentdex/data-science-bowl-2017/first-pass-through-data-w-3d-convnet

https://www.kaggle.com/rodenluo/data-science-bowl-2017/crop-save-and-view-nodules-in-3d

https://www.kaggle.com/kmader/data-science-bowl-2017/simple-single-image-features-for-cancer