**HMS Lung Cancer 1 Marathon Match - Solution Description**



●

●

●

●

●

●

●

**1.**  **Approaches Considered**

Use image features to train a random forest of binary classification trees, which will be used to   
predict if a single point belongs or not to a tumor.

Combine individual pixel predictions into regions, that will be returned as the answer.

**2.**  **Final Approach**

**General Observations**

My final approach follows the initial idea previously described, with few extra steps to overcome   
problems that I observed during the contest.

One of the “problems” I found was that the sets (tumor / not tumor) are very imbalanced,   
obviously with much more pixels belonging to “not tumor” class. Two strategies were used to

mitigate this problem:

○ discard as many pixels as possible that we can easily mark as “not tumor”, before the   
 classification process;

○ randomly subsample “not tumor” pixels.

**Image Processing**

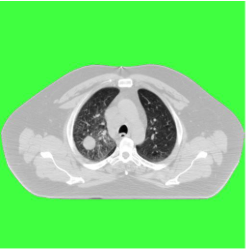
Each slice image is processed, reading the PNG file and converting to an array of pixel values,   
together with other properties that will be used later in different parts of the solution.

First step here is to limit pixel values range into [1, 1279]. Initially a dynamic (based on the   
content) conversion was implemented. But tests showed that a fixed conversion worked better.

○ Value = Raw Value – 23

○ If Value < 1 ==> Value = 1

○ If Value > 1279 ==> Value = 1279



The next step is to perform a flood fill from   
the image borders, marking all pixels outside

the body with the special value of zero. All   
these pixels (with zero value) will be ignored

later in training and testing steps, as they   
obviously couldn’t contain tumor regions. In

the Figure 1 green pixels denote pixels   
outside the body, which are discarded from

all image subsequent analysis.

***Figure 1*** *– A sample slice image, with “outside of the*   
 *body” discarded pixels in green.*

A simple “center” of the body is estimated by averaging all non-zero pixels present in the image.   
Then a randomized hill-climb process is executed, to find a near-vertical line that splits the image

image

final

A

processing



●

●

●

Figure

4

an

of

the

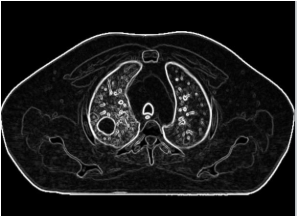
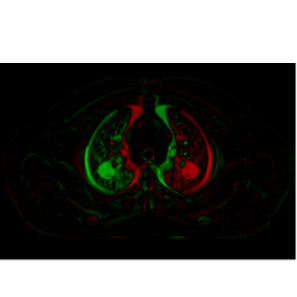
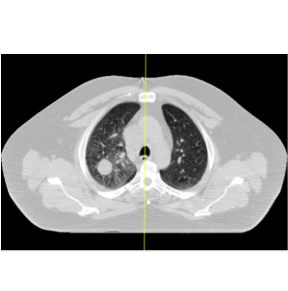
“edge

●

●

●

●



into two symmetric sides. The implementation used was very simple, and there should be room   
for improvement here.

This “symmetry line” is then averaged using 5 consecutive slices (-2 to +2, from a given slice).   
Example shown in Figure 2.

After that, a comparison (subtraction) between each pixel to its symmetric pixel is made,   
generating another array of values, ranging between [-1279, 1279]

***Figure 2*** *– In yellow, the symmetry line found during image*   
 *preprocessing phase.*

***Figure 3*** *– A “mirror image”, subtracting each pixel value by its*   
*symmetric pixel (green pixels for positive values, and red pixels*

*for negative).*

step   
applies a simple 3x3 edge detector

filter. The produced values will be   
later used by the feature extracting

process.

contains

example

produced

image”.

***Figure 4*** *– Result of the edge detection process.*

**STEP 1 – Lung Detector Training**

In order to reduce the number of pixels we need to deal with, the solution tries to detect and then   
discard slice images above (neck / head) or below the lung, where we won’t find lung tumors.

Using the training set in which lung regions as present, a binary classifier is built to detect if a   
single slice image (not pixels, we are talking about the whole slice image here) contains or not a

part of patient’s lungs.

Since there are few samples (one sample is one slice image), a lot of features were used.   
Basically, the features are histogram characteristics of the image as a whole, and from different

regions (close the center for example).

**STEP 2 – Tumor Detector Training**



●

●

●

●

●

●

●

●

This is the main training step, which aims to build a random forest that classifies pixels   
individually, as belonging or not to a tumor.

First the trained lung detection classifier (described in STEP 1) is applied for each slice image   
from a patient. This step produces the probability of each slice image to contain portions of lungs.

It takes then the largest range that contains all slice images with predicted probability above 80%.   
Any adjacent slices (in both directions, top / bottom of the body) with more than 50% of prediction

value are also added to the range. And finally, 4 extra images (in each Z direction) are added as a   
margin. For some patients, this step reduces the number of slice images to less than half of

original provided set.

Ground truth regions are processed and all pixels inside the tumor are used. For other (not tumor)   
pixels, a random subsample (1 in each 36) is applied.

For each sampled pixel, the following features are collected:

○ For different square regions around the pixel (3x3, 5x5, 7x7, …, 33x33), the average,   
 variance and skewness of pixel gray values;

○ The same properties (average, variance and skewness) for slightly different size squares,   
 but using the “mirror” and “edge” values, instead of gray values;

○ The same properties for different size squares, using two “3D” generated channels. One   
 of these generated channels is the average for each pixel with the pixel values on same

position from the slice above and the slice below the current one. The other uses the   
 difference between the current pixel value with the pixel from the slices above and below;

○ Histogram of pixel values for another set of square regions around the pixel. Pixel values   
 are classified into 12 bins (<700, >= 1100, and uniformly into 10 bins between these

values);

○ Pixel distances to vertical and horizontal center;

○ Pixel distances in each of 4 directions to the closest “dark” pixel (2 values are used as   
 “dark”, 200 and 500). The idea here was to find distance/relative position to lung pixels;

○ Number of “dark” pixels in each of 4 directions;

○ If contrast was used or not (taken from clinical file);

○ Global slice properties (height, percentage of “dark pixels”).

My final submission used 128 trees, but 64 or even 32 trees produce very similar final results, as   
individual pixel prediction is later combined.

This step generates a file with information about tumor shape present on ground truth. It contains   
the dimensions of tumor bounding box, the proportion between the height and the width of this   
rectangle, and the proportion between the tumor area and the bounding box area. All the

information is per slice, for slices that contains parts of a tumor.

**STEP 3 – Tumor Detector Testing**

The first step here is to apply the lung detector classifier (exactly the same procedure described   
in the beginning of STEP 2) to find the range of slices that will be considered.

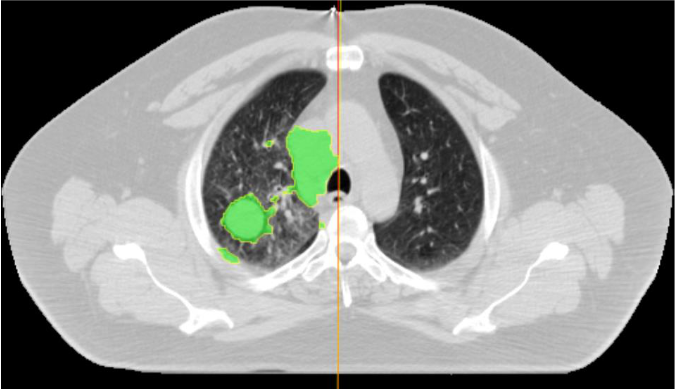
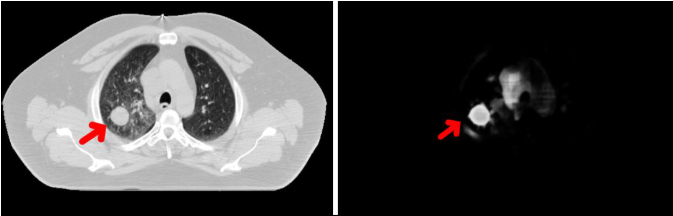
Then each pixel is evaluated using the main classifier built on STEP 2. The raw output of this   
process can be observed in Figure 5.

***Figure 5*** *– In the left side a testing image (with a tumor pointed by the red arrow). In the right side is the result of individual*   
 *pixel prediction. Lighter (closer to white) pixels mean higher chance of belonging to a tumor. Darker pixels mean lower*



●

●



*probabilities, with black meaning 0% predicted chance of belonging to a tumor. Note that other regions (part of what I believ e*   
 *to be the heart) has medium values, not as high as the tumor though.*

A threshold is defined (final submission used 65, in a [0, 255] scale), and pixels with probability of   
belonging to a tumor above this level are connected. The result can be observed in Figure 6

(grouped pixels in green).

***Figure 6*** *– Regions (in green) containing connect pixels with predicted probability of belonging to a tumor higher than a*   
*threshold. There are serval regions is this example. The largest ones are the correct (tumor) one, and the one that covers part*

*of the heart.*

Then a heuristic value is given to each region, based on the average of predicted values inside   
the region. This “region value” is later updated, based on the properties of the region (dimensions,

proportion), comparing the observed values of the current region to the information collected from   
training set (last item in STEP 2 description). Uncommon shapes (too small, too large, too wide,

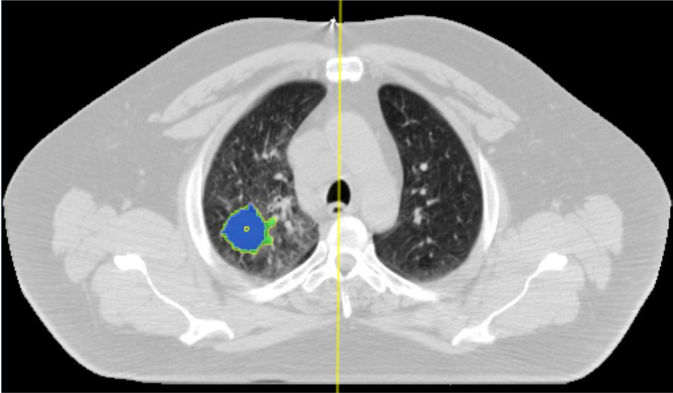
etc.) are penalized, and highly unlike shapes are discarded.

So far, the algorithm worked in a single slice. At this point, it combines the regions from all the   
slices, grouping the ones that share a common area from vertically adjacent slices. Finally, the



●

●



group (exactly one per patient) of regions with the higher value is selected.

Although the solution has an answer ready to be returned at this point, I noticed looking into   
example images that the predicted regions sometimes contains false positive pixels with not-so-   
high predicted value, but that are connected to the correct (high value) pixels. The opposite

situation also happened: pixels very “similar” (same gray level) and connected to the correctly   
predicted region, that for some reason received low prediction values, and were excluded from

the answer. In order to deal with that, I added an additional refinement phase that repeats the   
delimitation of the returned region. It starts by finding the “center” of the tumor, and repeats the

search for connected pixels, but instead of using only the predicted individual pixel value, it uses:   
 ○ predicted pixel values (high values have higher chance to be correct);

○ gray level difference to the “center” of the tumor (pixels with very similar gray levels have   
 higher chance to also belong to the tumor);

○ distance (in a slice) to the center of the tumor (pixels closer to the center haver higher   
 chance);

○ distance (number of slices) to the vertical center of the tumor.

Using these properties together, produces more regular-shaped regions, which improved the

accuracy of the prediction in the cases where the “raw” prediction was “correct” but the score was

low because of too many false positive / false negative pixels. The Figure 7 shows an example of   
this refinement process.

***Figure 7*** *– In blue the final “refined” region, and in green the initially predicted region. The yellow circle is the estimated*   
 *“center” of the tumor, from which the pixel expansion process starts.*

**Open Source Resources, Frameworks and Libraries**



**3.**

●

●

●

●

files.

3.

My solution did not use any library or open source resource.

**4.**  **Potential Algorithm Improvements**

Improve symmetry check between the two lungs, finding left and right lungs shape;

Detect organs (left / right lung, heart, etc.) in each slice, and use this information during feature

extraction process (inside and/or distance to each organ);

Add other criteria to discard “non-tumor” pixels, based on pixel intensity, position, etc.;

Use a better strategy (other than subsampling) to overcame the imbalance between the number

of “tumor” and “not tumor” pixels.

**5.**  **Deployment Guide**

The solution is implemented in Java, with the code organized in about fifteen “.java” files. Compiling is

a straightforward java compilation (javac), and the deployment is just copying the generated class

**6.**  **Final Verification**

1. Setup folder / file structure as in provided ZIP files, with the PNG version of the data.

2. Put clinical file in a separated folder.

In the testing/validation VM I provided the model files, so no training was necessary. But running

the training process is also straightforward:

a. Execute “LungDetectorTrainer”, which will produce a serialized random forest file named

“rfLung.dat”.

b. Execute “TumorTracerTrainer”, which will produce a serialized version of the main

classifier named “rfTumor.dat”. This step also produces a text file with general information

about tumor shapes collected from training set, named “infoTumor.dat”.

4. Execute “TumorTracerTester”, which will produce the expected CSV file.