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

1



**1.**

●

●

**2.**

●

●

time

●

●

A.

1.

2.

3.

4.

5.

B.

Testing

5.

1.

2.

**Approaches Considered**

I considered the following approaches as part of this marathon match:

Convolutional Neural Network (CNN)

Random Forest

**Final Approach**

I selected Random Forest as my final approach. The reasons are:

I have no prior experience with CNN

I did not find simple enough library for CNN which I could learn using in such a short

The match started while I was still competing in CMAP2 match, so I had even less

time to build the solution

The form of the match was very similar to the SpaceNet Challenge, which I solved

using Random Forest, so I could reuse huge part of the code from that challenge by

simple copy-paste without any change.

The detailed description of my solution follows. The basic steps are:

Training

Conversion of images

Detection of the “area of interest” (AOI)

Segmentation of AOI

Assignment of features

Build classification random forest

1. – 4. The same as in training

Classification using the random forest

C. Postprocessing

Remove “isolated” segments

Remove segments contained inside of other segments

**Training**

*A.1 Conversion of images* [doppm.cpp]

All the PNG images are converted to PPM format using ImageMagick 7.0.5. During this

conversion, I also add brightness to the images. The exact conversion command applied to

each image is

magick convert N.png -modulate 2000% -define png:color-type=2 -depth 8 N.ppm

*A.2 Detection of the “area of interest” (AOI)* [pluca\_vm.cpp, lines 788-1040]

2



•

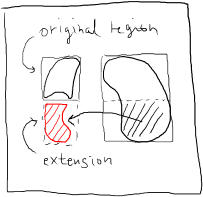
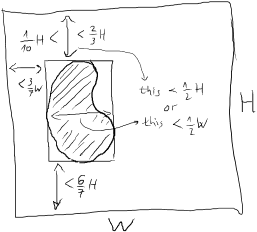
•

•

•

•

•



In each 512x512 image, AOI is determined. Area out of AOI is ignored. Basically, AOI is

meant to contain convex hull of each lung. The determination of AOI works as follows:

Each pixel is considered “dark”, if its intensity is below the threshold DARK defined in

[line 55].

4-connected regions of dark pixels are detected [lines 795-825]

These regions are sorted by size in descending order [line 826]. The first regions

which satisfy some specific conditions [lines 835-836] are considered as “left lung”,

and “right lung”, respectively. These conditions involve only the coordinates of the

bounding box of the region with respect to the image dimensions (the figure depicts

the conditions for the left lung; the conditions for the right lung are symmetric). The

conditions were deduced by visual inspection of the images and in most cases, lungs

are detected correctly. In cases when there are no lungs in the image, some

“random” objects are sometimes detected as lungs in this way.

In some cases, both lungs are connected, which means the previous steps returns

both lungs as either “left lung” or “right lung”. In this case (that is, if a specific

condition is met, see [lines 860 and 883]) the algorithm splits the region into two

regions per the *x*-coordinate [lines 861-882, 884-905].

In some cases, one lung region is substantially taller than

the other (i.e., the bounding box is of greater height). Very

often this happens when there is a tumor under or above

the shorter region. Therefore, if some specific conditions

are met [lines 910, 924, 938, 952], the shorter region is

extended to match the height of the taller region [907-966].

The extension reflects the pixels from the taller region into

the shorter region so that the new bounding box has the

original width, only the height changes.

[lines 969-1039] All the pixels which lie outside of the convex hull of each lung region

are replaced with black color. More precisely, each pixel is labeled with

“1” if it lies in the left lung [line 1012],

3



o   
o   
o

o

997],

o

o

0.

1.

2.

3.

“2” if it lies in the right lung [line 1018],

“3” if it lies in the convex hull of the left lung (but not in the left lung) [line 997],

“4” if it lies in the convex hull of the right lung (but not in the right lung) [line

“5” if it lies in the convex hull of the union of both lungs (but is not labeled with

“1”, “2”, “3”, nor “4”) [line 997]

“0” otherwise [line 991].

The pixels with labels “0” and “5” are replaced with black color [line 1010], that

effectively means they are ignored. The labels and lung positions are stored because

some features are calculated out of these.

*A.3 Segmentation of AOI* [pluca\_vm.cpp, lines 1284-1404]

Segmentation is performed using “Graph Based Image Segmentation” external library [line

1284]. This library provides segment\_image() method. The result of this method is a new

image in which each detected segment is of different color. The only change I made to the

library code was that originally the colors were picked by random – I changed it so that the

colors are consecutive integers [segment-image.h, lines 28-41]. There are 3 parameters

(sigma, k, min\_size) which affect the segmentation. I tried different combinations and selected

the one giving promising results.

For each detected segment, some statistic is calculated [lines 1294-1332] to be used in

feature assignment. For each segment, the polygon which bounds the segment is calculated

[lines 1338-1404]. This polygon is always simple – the pixels lying inside of the segment,

which do not belong to this segment, are ignored. (Because of this, some of the resulting

polygons lie inside of other polygons – these are removed in step C.)

To avoid the resulting CSV-file being too big, each polygon is simplified to contain at most

100 vertices [line 1418]. This simplification is performed using “psimpl” external library and its

method simplify\_douglas\_peucker\_n<2>().

*A.4 Assignment of features* [pluca\_vm.cpp, lines 1423-1567]

To each segment detected in the previous step, 56 features are assigned. Many of the

features are the same as I used in SpaceNet Challenge and some of them maybe do not

make sense in this contest. I decided to retain all the features, because in Random Forest, a

few irrelevant features do not negatively affect classification. The features are numbered

(0-based) and the list of all the features follows (some features are derived from other

features, the notation *Fi* refers to the *i*-th feature):

Total number of pixels in the segment

Perimeter of the polygon bounding the segment (before the simplification of the

polygon)

Average intensity of the pixels of the segment

Total number of segments in the slice in which this segment occurs

4. *F*1 / sqrt(*F*0)

Perimeter of the polygon when simplified with tolerance 0.9 (simplification using

4



5.

7.

9.

external library “psimpl” and its method simplify\_douglas\_peucker<2>())

6. *F*5 / sqrt(*F*0)

Perimeter of the polygon when simplified with tolerance 1.4

8. *F*7 / sqrt(*F*0)

Perimeter of the polygon when simplified with tolerance 2.5

10. *F*9 / sqrt(*F*0)

11. Number of vertices of the polygon when simplified with tolerance 2.5

12. Number of vertices of the polygon when simplified with tolerance 5

13. (Perimeter of the polygon when simplified to 20 vertices) / *F*5

14. (Perimeter of the polygon when simplified to 10 vertices) / *F*5

15. (Perimeter of the polygon when simplified to 5 vertices) / *F*5

16. Sum of the absolute values of the negative exterior angles of the polygon when

simplified to 100 vertices (i.e., “non-convexity” measure)

17. The same as *F*16, but with only 20 vertices

18. The same as *F*16, with 10 vertices

19. The same as *F*16, with 5 vertices

20. Difference of the greatest and the smallest interior angle of the polygon when

simplified to 5 vertices

21. Minimum of abs(*F*2 – *F’*2), where *F’* ranges over all the neighboring segments

22. Ratio of the area of the polygon and the area of the minimal rectangle containing the

polygon

23. Order of the slice (with respect to the given scan)

24. *F*23 / (total number of the slices for the given scan)

25. Slice z (extracted from (0020.0032) CT tag)

26. Slice dx (extracted from (0028.0030) CT tag)

27. Slice x0 (extracted from (0020.0032) CT tag)

28. Slice y0 (extracted from (0020.0032) CT tag)

29. *x* coordinate of the centroid of the segment

30. *y* coordinate of the centroid of the segment

31. *F*29 transformed to mm using pixelToMm() method   
32. *F*30 transformed to mm using pixelToMm() method

33. *F*0 × *F*26 × (slice dy)

34. *F*1 × (*F*26 + (slice dy)) / 2

35. Number of neighboring segments

36. Area of the left lung (detected in step A.2)

37. Area of the right lung

38. Area of the convex hull of the left lung

39. Area of the convex hull of the right lung

40. Area of the convex hull of the union of both lungs

41. abs(*F*36 – *F*37) / (*F*36 + *F*37)

42. (*F*36 + *F*37) / (*F*38 + *F*39)

43. (Number of pixels of the segment labeled with “1” or “2”) / *F*0 (see end of A.2)

5



lung)

lung)

*B.1 – 4*

44. (Number of pixels of the segment labeled with “1”, “2”, “3”, or “4”) / *F*0

45. (Number of pixels of the segment labeled with “5”) / *F*0

46. (*F*29 – (*x* coordinate of the centroid of the left lung)) or ((*x* coordinate of the centroid of

the right lung) – *F*29), according to the distance of the segment centroid from the

centroids of the left and right lung

47. (*F*30 – (*y* coordinate of the centroid of the left lung)) or (*F*30 – (*y* coordinate of the

centroid of the right lung)), according to the distance of the segment centroid from the

centroids of the left and right lung

48. (*x* coordinate of the centroid of the right lung) – (*x* coordinate of the centroid of the left

49. (*y* coordinate of the centroid of the right lung) – (*y* coordinate of the centroid of the left

50. Average intensity of the corresponding pixels in the previous slice

51. Average intensity of the corresponding pixels in the next slice

52. *F*50 + *F*51

53. *F*2 – *F*50

54. *F*51 – *F*2

55. *F*54 – *F*53

*A.5 Build classification random forest* [prienik.cpp; pluca\_vm.cpp, lines 1140-1194]

The output of the previous steps applied to the training images are two CSV files:

polygons.csv and features.csv. The first one contains the polygons bounding the segments (in

the same format as required in the problem statement), the second one containing the

features assigned to each segment (each line of the file corresponding to one segment). The

script prienik.cpp takes these two files as input. Its output is the file traindata.csv, which

contains all the data from features.csv, but to each line one real number is appended – the

ratio of the polygon area lying in the tumor. This is calculated using the ground truth data. For

calculation of the area of the intersection of the polygon with the tumor polygons, external

library “pia” is used.

Classification random forest is built over the polygons in traindata.csv, with the features

described above. There were 908402 such polygons detected on all the slices. The target

variable is 1 if the ratio of the polygon area lying in the tumor is greater than 0.5, 0 otherwise.

The final set up was 1000 trees, each built over 600000 randomly selected polygons (with

replacement). When splitting the nodes, 5 random features were tried each time to find the

best splitting. The cost function was sqrt(*x*(1 – *x*)). Nodes with size 5 or less are not split.

**Testing**

The same process as in A.1 – 4 was applied to the images in the test set.

*B.5 Classification using the random forest*

[post.cpp]

6



•

•

•

•

•

•

•

Each tree assigns to each testing polygon the ratio of polygons with target variable 1 among

training polygons which lie in the same leave of the tree. If the average among all the trees is

greater than a specific threshold (which was 0.14 in the final set up), the polygon is

considered as a candidate for tumor, all the other polygons are discarded.

**Postprocessing**

*C.1 Remove “isolated” segments*

During the previous phase, the 3D information was almost completely ignored since the slices

were treated separately. The only 3D information / connection between neighboring slices

was extracted into features 23, 24, 25 and 50 – 55. Thus, there are several “isolated”

polygons among the tumor candidates, i.e., the polygons such that on the previous and on the

next slice there is no tumor candidate in the same position. Such candidates are removed in

this step: For the candidate polygon *P* to qualify as a tumor, there must be a candidate

polygon in the previous or in the next slice such that the area of their intersection is more than

0.3 × (area of *P*).

*C.2 Remove segments contained inside of other segments* [inter.cpp]

As mentioned in A.3, some of the detected polygons lie entirely inside of some other

polygons. These are removed at the end. There are also some other small intersections (due

to the use of path simplification and because the segmentation generates 4-connected

regions). I did not remove these intersections, as the score gain would be very small (maybe

around 1000) and it would be necessary to change the polygons in this case.

**Effect of specific steps on score**

To get a better insight of how particular steps affect the score, here is a summary of how my

provisional score changed after improving the algorithm:

Only the first 36 features, without A.2, without C: score less than 80000

Adding A.2, but without “extension”, discarding only pixels with label “0” (retaining

pixels with label “5”), using the first 46 features: around 160000

Discarding also label “5”, using the first 50 features: around 180000

Adding C.1: 188000

Introducing last 6 features: 200000

Adding lung “extension”: 217000

Adding C.2: 225000

To get a glimpse of the importance of specific features, in the following table, the second

column is the number of times each feature was selected for splitting, the first column is the

sum of the sizes of the nodes each feature split. Both columns are sorted in descending

order, so the most important features should be on the top.

283364103 2 | 71958 24

7



●

●

●

**4.**

●

266074125 52 | 71018 29

245823399 23 | 68967 23

239599934 43 | 67872 46

224213980 50 | 67242 31

213495710 24 | 65409 47

210583468 51 | 65349 2

183691086 25 | 62951 25

180984146 55 | 61479 42

145316365 29 | 59685 32

127098441 31 | 59492 37

119865505 54 | 57990 36

119316204 38 | 57062 48

114625510 47 | 56904 50

112300388 46 | 55938 30

112222776 40 | 55871 41

109244291 42 | 55626 49

107321644 36 | 53886 52

104123894 34 | 53765 40

92712210 53 | 52714 0

90292888 21 | 52632 51

86131308 5 | 52530 15

85404977 41 | 52344 39

81131646 9 | 52117 3

79982468 4 | 48828 55

77386373 7 | 48605 38

77221804 37 | 48179 4

75100887 32 | 47522 21

74339128 39 | 47238 53

74082796 30 | 46853 8

72943744 48 | 46608 17

71616798 15 | 46278 13

69109970 49 | 46109 14

68812454 33 | 45436 20

66489606 1 | 45269 5

65263832 3 | 44563 18

63817700 17 | 43949 10

62681010 14 | 43868 1

60218798 10 | 43085 54

58893795 0 | 42681 9

56945224 8 | 42161 33

53582794 22 | 41898 43

53287740 6 | 41718 7

51748234 20 | 41580 6

49684350 13 | 40889 22

44649664 26 | 39231 16

37739044 28 | 31719 34

37611994 18 | 27131 11

34601741 27 | 20855 35

33798301 11 | 20136 12

30412752 16 | 19667 19

28163767 12 | 17904 26

21421145 45 | 16178 27

20482968 35 | 15115 28

19067630 44 | 2842 45

13438860 19 | 635 44

**3.**  **Open Source Resources, Frameworks and Libraries**

● ImageMagick, [http://www.imagemagick.org,](http://www.imagemagick.org/) [Apache 2.0](http://www.imagemagick.org)

Graph Based Image Segmentation, [http://cs.brown.edu/~pff/segment/,](http://cs.brown.edu/%7Epff/segment/) [GPL v2](http://cs.brown.edu/~pff/segment/)

psimpl, [http://sourceforge.net/projects/psimpl/,](http://sourceforge.net/projects/psimpl/) MPL 1.1

pia, [https://github.com/jjgreen/pia,](https://github.com/jjgreen/pia) MIT

**Potential Algorithm Improvements**

The segmentation algorithm I have used was very simple and probably some other

segmentation algorithms will give better results. One could also try some

3D-segmentation.

My lung detection algorithm is quite trivial, one can use training data and machine

8



●

●

**5.**

1.

2.

**6.**

1.

2.

Testing

3.

4.

learning to detect lungs more precisely.

I did not pay enough attention to the feature engineering. Probably better features

may be designed.

**Deployment Guide**

Install ImageMagick 7.0.5

Based on your system, select appropriate binary release from

[http://www.imagemagick.org/script/download.php,](http://www.imagemagick.org/script/download.php) or install from source

[(http://www.imagemagick.org/script/install-source.php;](http://www.imagemagick.org/script/install-source.php) when installing from

source, it is necessary to install some dependencies; I followed this:

[http://askubuntu.com/questions/745660/imagemagick-png-delegate-install-](http://askubuntu.com/questions/745660/imagemagick-png-delegate-install)

problems).

Compile all the \*.cpp scripts (make sure all the header files from the external libraries

are presented in the same folder):

g++ -std=gnu++11 -O3 doppm.cpp -o doppm

g++ -std=gnu++11 -O3 pluca\_vm.cpp -o pluca\_vm   
g++ -std=gnu++11 -O3 prienik.cpp -o prienik

g++ -std=gnu++11 -O3 post.cpp -o post   
g++ -std=gnu++11 -O3 inter.cpp -o inter

g++ -std=gnu++11 -O3 undoppm.cpp -o undoppm

**Final Verification**

(Let [trF] and [teF] be the path to the training and testing data folder.)

Training

./doppm [trF]

./pluca\_vm train 1 0.5 50 70 100 2 1 1 [trF] [trF] polygons.csv   
./prienik [trF] polygons.csv features.csv traindata.csv

./doppm [teF]

./pluca\_vm test 1 0.5 50 70 100 2 1000 600000 [teF] [teF] test.csv

Postprocessing

./post 0.3 1 test.csv test\_post.csv   
./inter test\_post.csv [Output file]

Remove temporary files

./undoppm [teF]   
./undoppm [trF]

rm rs.csv rs2.csv test.csv test\_post.csv polygons.csv features.csv   
rm prediktors.txt