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



[place](https://community.topcoder.com/longcontest/stats/?module=ViewOverview&rd=16870)

**model**

**Pre-processing.**

**selection**



, [[2]](https://community.topcoder.com/longcontest/?module=ViewProblemStatement&rd=16870&pm=14511)

.



**basic**



**1.**

●

**Data**

●

1

This report describes a solution to automatically find lung cancer cells in tomography scans. The lung

cancer automatic segmentation problem was proposed in the Harvard Medical School challenge [[1]](https://www.topcoder.com/blog/improve-lung-cancer-treatment-with-algorithms/)

The provided solution was based on image segmentation with deep learning and the model used was

the [SegNet](http://mi.eng.cam.ac.uk/projects/segnet/)

. The SegNet model was initially designed to segment urban road scenes but

could successfully segment lung cancer cells. SegNet is a relatively lightweight segmentation model,

where it was easy to train and deploy solutions with a commercial video card, GTX 1070. Despite its

small size this network can provide reasonable results. If more powerful machines are available it is

recommend the use of “bigger” networks, see 5. Potential Algorithms Improvements. The solution

presented here achieved a final score of 146,679.48 which was placed in the [7th](https://community.topcoder.com/longcontest/stats/?module=ViewOverview&rd=16870)

. Up to 180k

could be achieved with more training time.

**Approaches Considered**

. This was probably the most important evaluation of the problem. From

the total 38,913 of the images (slices) only 4,100 presented cancer cells. In other

words, only 10.54% of the images presented the class which we were interested to find.

This is a classic imbalance problem where only a small part of the dataset presents the

examples we are looking for. To solve this problem the classes were balanced

artificially choosing all images with cancer cells (4,100) and randomly choosing other 5

thousands images without cancer cells. Therefore the training dataset had roughly 11

thousand images. This data selection yield the best results. Other selections were

evaluated but they provided worse results. Training the model only with cancer images

(4,100) provided half the score and training the model with all images (38,913) required

4 to 5 times more time to reach a lower score.

The image size and the image region of interest were also taken in

consideration. The original input images size is 512 x 512. From a visual inspection it

was possible to realize that most part of the images were not from the patient and in

most images the tomography equipment was visible. Considering that the tumor is just

a small portion of the image the whole image presented useless information that could

interfere in the learning process. As a common practice in data science the

pre-processing phase aims to select the data relevant to the problem reducing the data

dimensionality. Therefore the original images were cropped from 100:360 and 100:420

and the resulting images resolution was 260 x 320. The cropped range was defined

through visual inspection and it is a potential improvement point.

Several model sizes were tested to understand its influence in the



**model.**



●

**Size**

**of**

**the**

**●**

**2.**

●

●

●

**3.**

●



,



2

results. Basically, the size of the model does not have great influence. The

SegNet-basic model has already a considerable size. If the number of layers are

reduced the accuracy slightly decreases and if the number of layers increase the

accuracy slightly increases. The same behaviour was found when changing the patch

sizes and output feature maps. Summing up: it is possible to increase the score when

increasing the number of layers, patch sizes and output feature maps but the

computational cost is higher than the improvements.

**Influence**  **of**  **lung**  **segmentation.**  The influence of the lung segmentation was

evaluated as well. Basically the accuracy was slightly lower when the lung and cancer

cells were segmented. As only a small portion of the scans had the information to

segment the lung this approach reduced the available training set. Therefore this

evaluation was biased once the size of the dataset has great influence in the results.

**Final Approach**

The following reasons were the main aspects that defined the presented solution.

The lack of processing power prevented the use of bigger deep learning models. State

of the art segmentation methods required a large amount of GPU memory, usually

higher than 12GB. The commercial video card used in this work had 8GB of memory.

The small power processing was the main reason to choose the SegNet-basic model.

The SegNet-basic is a smaller version of SegNet and presents a smaller accuracy.

Another reason to choose SegNet-basic was its smaller training and inference time,

which would allow testing different approaches within the 3 weeks deadline of the

challenge. Even choosing the smaller model, a modification usually required several

hours to be evaluated.

PS: After the final submission a bug in the code was found. The **radiomics\_lm\_sum** label was

used to find cancer cells contours. This is a major issue once this wrongly label 1359 images as

images with cancer cells. This number represents ~33% of all images with cancer cells.

**Open Source Resources, Frameworks and Libraries**

The only software used to develop this solution was a version of caffe-segnet modified to work

with the NVidia cudnn5 library. Caffe-segnet is licensed for non-commercial use. From the

github repository: “This software is released under a creative commons license which allows for

personal and research use only. For a commercial license please contact the authors.”

caffe-segnet, <https://github.com/TimoSaemann/caffe-segnet-cudnn5>

<https://creativecommons.org/licenses/by-nc/4.0/>

The caffe-segnet-cudnn5 was only used to improve testing and inference time. Therefore the

original caffe-segnet version will likely work with the provided solution.

**Potential Algorithm Improvements**



**data**



same than Caffe. Please follow the instructions from [caffe](http://caffe.berkeleyvision.org/installation.html)

**images**

not be enough to train these models. Suggested models are: [DeconvNet](http://cvlab.postech.ac.kr/research/deconvnet/)

**models**

[website](http://caffe.berkeleyvision.org/installation.html)



**4.**

●

**Bigger**



.

●

**More**

●

●

●

**5.**



1.

2.

3

Three possible improvements were identified during the development of this solution:

. Using a more accurate deep learning segmentation model will likely

improve the score. However this should be done with caution once the dataset size may

, [DeepLab-v2](http://liangchiehchen.com/projects/DeepLab.html)

and [PSPNet](http://appsrv.cse.cuhk.edu.hk/~hszhao/projects/pspnet/index.html)

. It is a common sense in deep learning that increasing the dataset can

improve results. Therefore, artificially generating images to enlarge the dataset could

improve the score. Flipping, rotating and translating the images and ground truths are

example of techniques to generate more images.

**Pre-processing**

. Usually deep learning models receive the input data without

major pre-processing modifications. This is the opposite of what was done with man

engineered solutions. The main pre-processing improvement would be the development

of a solution to center the patient in the image before cropping it. This solution should

also take in consideration the final resolution size which should be the smallest as

possible. Removing saturation points from the images seems to be a good idea,

however visual clues from inference results suggest that these saturation points are not

an issue.

PS: After the final submission some evaluations were done and the initial results suggests that:

using the “bigger” SegNet model does influence the accuracy. It increases the score in

several thousand points (~20k) but not enough to reach top-5 score.

using data augmentation to increase the training dataset has a major influence. Flipping

and rotating images and ground truths could increase score close to one hundred

thousand (~80k).

**Deployment Guide**

Only the caffe-segnet is required to build and install. Its installation instructions are exactly the

. Although it is possible to run

this solution in CPUs, it is strongly recommended to use a high-end NVidia GPUs to

substantially reduce the training and inference time. The presented solution was developed in

Ubuntu 16.04 (Xenial) and it was not tested in other platforms.

Please install caffe-segnet inside the source directory of the provided source code.

clone the provided source-code

clone and install caffe-segnet inside the source code directory.

**Final Verification**



Modify the tag **max\_iter**

Modify the tag **snapshot\_prefix**

Modify the tag **net**

Modify the **source**

Modify the **root\_path**



**6.**

1.

2.

3.

a.

4.

a.

b.

c.

5.

a.

b.

6.

7.

8.

9.

a.

a.

a.

4

Run all the commands from the source directory.

To train the model:

Separate the images in train and test ( see data\_selection directory )

variable from scripts/generate\_gt.py and

scripts/generate\_train\_files.py to point to your training dataset.

Generate ground truth:

python scripts/generate\_gt.py

Create the dataset, training and inference directories:

mkdir caffe/dataset/

mkdir caffe/models/training/

mkdir caffe/models/inference/

Generate the caffe training files and move them to caffe/dataset/:

python scripts/generate\_train\_files.py

mv \*.txt caffe/dataset/

tag from caffe/models/lungnet\_basic\_train.prototxt to point to

caffe/dataset/train\_mix.txt

from caffe/models/lungnet\_basic\_solver.prototxt to point to the

training file caffe/models/lungnet\_basic\_train.prototxt.

from caffe/models/lungnet\_basic\_solver.prototxt to

point to caffe/models/training/.

from caffe/models/lungnet\_basic\_solver.prototxt to 120000.

10. Train caffe model up to 120k iterations:

caffe-segnet/build/tools/caffe train -solver

caffe/models/lungnet\_basic\_solver.prototxt

11. Modify the variable **caffe\_root** from caffe/scripts/compute\_bn\_statistics.py to point to

your caffe-segnet directory.

12. Adjust the BN weights:

python caffe/scripts/compute\_bn\_statistics.py

caffe/models/lungnet\_basic\_train.prototxt

caffe/models/training/lungnet\_basic\_iter\_120000.caffemodel

caffe/models/inference/

13. Rename BN weights file:

mv caffe/models/inference/test\_weights.caffemodel

caffe/models/inference/lungnet\_weights.caffemodel

To test the model:



Modify the variable **root\_path**

Modify the **source**

Modify the variable **text\_file\_name**

Modify the variable **root\_path**

Modify the variable **caffe\_root**

Move the **inference.txt**

Modify the variable **root\_path**



1.

2.

a.

3.

4.

5.

6.

7.

8.

a.

9.

a.

5

from scripts/convert\_provisional.py to point to your

inference directory.

Convert provisional images:

python scripts/convert\_provisional.py

file to caffe/dataset/

from caffe/scripts/test\_segmentation.py to point to your

caffe-segnet directory.

from caffe/scripts/test\_segmentation.py to point to your

inference directory.

from caffe/scripts/test\_segmentation.py to point to

caffe/dataset/inference.txt

tag from caffe/models/lungnet\_basic\_solver.prototxt to point to

caffe/dataset/inference.txt

Run inference:

python caffe/scripts/test\_segmentation.py --model

caffe/models/lungnet\_basic\_inference.prototxt --weights

caffe/models/inference/lungnet\_weights.caffemodel

from scripts/generate\_bb.py to point to your inference

directory.

10. Generate the bounding boxes (result.cvs file):

python scripts/generate\_bb.py