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



used.

●

●

●

1

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

I decided that Deep Learning is a suitable way to attack the problem from the beginning and the

variation in my approaches was coming from the difference in the modules and networks I’ve

An end-to-end 2D approach using CNN where the input is an image of a slice and the

output is a mask classifying each pixel into {background, body, lung, …, radiomics\_gtv}

I also tried taking advantage of the slice ordering and trained a network using 3D

convolutions with 7 stacked slices as an input and a 2D output mask with the same

classes as the above one.

A chain of two 2D convolutional neural networks, the first one takes as an input a

512x512x1 image of a slice and segments the lung area (using a mask as an output),

and the second one takes as an input 256x256x3 (3 consecutive image slices) and

outputs a binary mask 512x512x1, classifying each pixel in {radiomics\_gtv, background)

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

My final approach was based on the last one from the previous section which varied itself given

the network architectures I’ve used.

**Preprocessing**

The images were normalized into [0,1] by dividing each pixel value by the max pixel value in the

training dataset. Additionally, each pixel value of an image was centered by subtracting the

mean value of the image. No further preprocessing was made.

**Lung Segmenter CNN**

I’ve used a CNN with several 2d convolutions as an encoder and almost mirrored 2d convolution

transposed layers as a decoder. All layers using a ReLU activation and a dropout value of 0.5

during training. The input to the network was 512x512x1 and the output 512x512x1. I was also

experimenting with adding batch normalization layers but I got no significant improvement. The

number of filters used for the convolutional layers vary from 16 to 256. A stride of 1 was used.

Downsampling was achieved by max-pooling. The output mask contained a probability of each

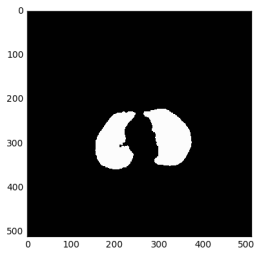
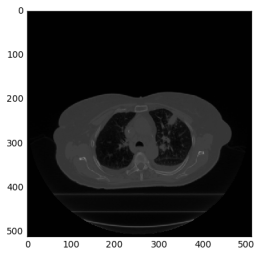
pixel being {background, lung}. Then every pixel having a probability higher than 0.5 was

classified as belonging to a lung.

The ground truth values for the mask were obtained by setting each pixel located inside the lung

polygon to 1 and all others to 0.

Figure 1: Original image



2

Figure 2: Predicted lung mask

**Network Architecture:**

Convolution(filters=16, kernel=[5,5]), MaxPool(kernel=[2,2], stride=[2,2]), ReLu, Dropout(0.5)

Convolution(filters=32, kernel=[5,5]), MaxPool(kernel=[2,2], stride=[2,2]), ReLu, Dropout(0.5)

Convolution(filters=64, kernel=[5,5]), MaxPool(kernel=[2,2], stride=[2,2]), ReLu, Dropout(0.5)

Convolution(filters=128, kernel=[5,5]), MaxPool(kernel=[2,2], stride=[2,2]), ReLu, Dropout(0.5)

Convolution(filters=256, kernel=[5,5]), ReLu, Dropout(0.5)

ConvolutionTranspose(filters=128, kernel=[5,5]), stride=[2,2]), ReLu, Dropout(0.5)

ConvolutionTranspose(filters=64, kernel=[5,5]), stride=[2,2]), ReLu, Dropout(0.5)

ConvolutionTranspose(filters=32, kernel=[5,5]), stride=[2,2]), ReLu, Dropout(0.5)

ConvolutionTranspose(filters=16, kernel=[5,5]), stride=[2,2]), ReLu, Dropout(0.5)

Convolution(filters=2, kernel=[1,1])

**Lung Slicer**

The goal of the Lung Segmenter module was to remove the unimportant regions of the image.

Hence, the lung slicer module is taking the output lung classification mask and is producing a

downsampled image for each slice where non-lung pixel values are zeroed-out. This is achieved

by simply multiplying the predicted mask by the image. As the bounding box of the lungs is

significantly smaller than the image size most of the resulting 512x512 image contains

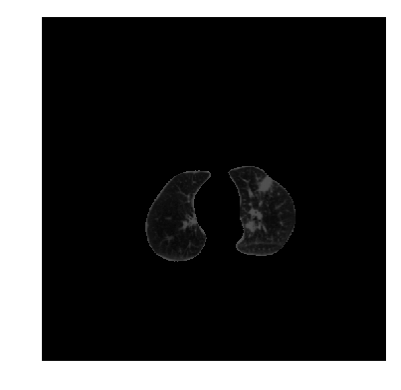
background. To reduce some computations of the next module, for each scan I’m finding a

minimal bounding box that contains all lung pixels from all lung areas and has the same

coordinates for each slice in a scan. It was indeed important that all slices from a scan had the

same bounding box coordinates, since the next module takes a stack of 3 slices as an input.

The bounding box coordinates were obtained by taking the min and max lung pixel coordinates



3

in each direction from all slices. The resulting image was padded with zeroes equally from each

side to a desired shape of 256x256.

Figure 3: Extracted lung regions

**Tumour Segmenter CNN**

Having extracted the lungs and filtering out all unimportant pixels, the next step I did is to train a

CNN that takes as an input a tensor with shape 224x224x3, consisting of 3 consecutive image

slices and outputs a mask with shape 512x512x1 having values of 1 for each pixel predicted as

part of a tumour. The network architecture consists of a VGG-19 encoder which training starts

from a pre-trained checkpoint on the ImageNet dataset. The decoder part consists of several

convolution transposed layers that are upsampling the output to the desired shape. The decoder

is using ReLu activations and dropout during training with a value of 0.5. The number of filters

used for each layer vary from 16 to 256. I found that a kernel size of 5x5 achieves the best

results. The ground truth masks for the network training were obtained by setting the pixels

located inside the tumour polygons to 1 and all others to 0.

Since the background-to-tumour pixel ratio was significantly imbalanced I used a different

weights for the background and tumour classes in my loss function. Misclassifying a tumour

pixel was penalized 8 times higher. The loss function used was Softmax Cross Entropy.

The final prediction contours are obtained from the prediction masks by finding the contours of



, Apache 2.0 open source license

, BSD License

, Apache 2.0 open



●

●

0.15.

**3.**

●



●



●



●



**4.**

●

helpful.

●

●

●

**5.**

4

each polygon formed inside the mask. I used skimage and Sutherland-Hodgman polygon

clipping for that purpose.

**Network Architecture:**

VGG-19

ConvolutionTranspose(filters=128, kernel=[5,5]), stride=[2,2]), ReLu, Dropout(0.5)

ConvolutionTranspose(filters=64, kernel=[5,5]), stride=[2,2]), ReLu, Dropout(0.5)

ConvolutionTranspose(filters=32, kernel=[5,5]), stride=[2,2]), ReLu, Dropout(0.5)

ConvolutionTranspose(filters=32, kernel=[5,5]), stride=[2,2]), ReLu, Dropout(0.5)

Convolution(filters=2, kernel=[1,1])

**Postprocessing**

I tried few different postprocessing steps to reduce the FPs. They were mostly based on filtering

polygons with average low probability to belong to a tumour. However, the only postprocessing

steps I left in the final solution were to

remove predicted regions containing less than 15 pixels.

remove every predicted as a tumour pixel that had a normalized intensity lower than

**Open Source Resources, Frameworks and Libraries**

Tensorflow, <https://www.tensorflow.org/>

scikit-image, <http://scikit-image.org/docs/dev/api/skimage.html>

TF Slim, <https://github.com/tensorflow/models/tree/master/slim>

source license

VGG-19 model checkpoint:

<http://download.tensorflow.org/models/vgg_19_2016_08_28.tar.gz>

**Potential Algorithm Improvements**

I believe one can get significant boost from improving the lung segmentation

component. It was regularly missing tumour areas, hence putting an upper bound on

the tumour segmenter performance. Some sort of post-processing should be also

A pre-trained network can be used for the lung segmentation component.

There is a significant number of FPs that can be reduced by using a region proposal

network.

I noticed that the training of the tumour network was very unstable and I had problems

finding an optimal checkpoint. I haven’t had the time to investigate the issue.

**Deployment Guide**

I’m using a python 3.5 so an installation of the requirements would be enough.



**6.**

5

pip install -r ./requirements.txt

You can also check ./compile.sh

**Final Verification**

My solution consists of several python scripts executed one after another. Please take a look at

./train.sh and test.sh scripts.