Automatic lung nodule detection

Octavi Font

June, 2018

Contents

Abstract

This Master's final thesis details the implementation of a computer-aided diagnosis (CADx) for lung nodule detection. The aim of this system is to provide assistance to radiologists for early diagnosis of lung cancer. The system's pipeline consists of 4 main steps:

- scan preprocessing
- lung segmentation
- nodule segmentation
- false positive reduction

Each part of the system is assessed quantitatively. Also, the system as a whole is compared with the state of the art following the metrics established in the LUNA grand challenge.

Chapter 1

Introduction

Some brief intro as to how I am planning to organize this thing. Let's try to add a citation Jacobs [2015]. And I've added something else, let's see if the auto build system picks it up. Does it now? Maybe it is working?

1.1 Clinical context

1.1.1 Lung cancer

Lung cancer is the most deadly cancer in the world. Bring some figures and talk about trends. Explain why this was collected, essentially the whole reasoning behind it was to provide a good benchmark to easily compare CAD systems

1.1.2 Computed Tomography

Basically talk about the technique and how it has been changing diagnosis recently.

1.1.3 Lung cancer screening with CT

Talk about the NLST study and NELSON. Reduction of 20% in mortality if screened, so early detection is important to improve the outcomes.

1.1.4 Lung nodules

Explain nodule types. Solid and subsolid.

1.2 Lung nodule CAD

1.2.1 Objectives

Explain why it would be useful (reduce workload, reduce intra-variability for radiologists). Also cheaper. Explain why historically they haven't worked (mention main problems a system like this faces) and why I think now is a good time to create a system that improves upon the existing state of the art.

1.2.2 Shortcomings

Explain what are the main things that fail

1.2.3 Pipeline

Even though there has been much effort in developing new techniques to improve the performance of CAD systems due to the availability of annotated datasets and challenges (NLST, ISBI, LUNA, DSB2017), the published systems tend to be brittle and very much focused on demonstrating good results on those specific challenges but useless as an integrated system. Also, what is not available tends to be proprietary systems, which might be good, but who knows really.

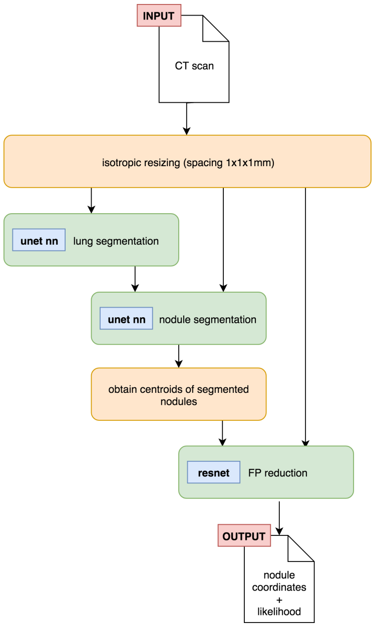
INPUT
CT scan
isotropic resizing (spacing 1x1x1mm)
unet nn
lung segmentation
unet nn
nodule segmentation
obtain centroids of segmented nodules
resnet
FP reduction
OUTPUT
nodule coordinates + likelihood

Figure 1.1: the lucanode pipeline

One of the improvements that I wanted to bring to the state of the art was to prepare a system which could be easily deployed in a real system. To achieve this I had to automate the scan preprocessing and prepare a full pipeline that could later on be integrated in a real system. In fact, this integration with a system has been performed by Albert , that has a queue which picks up the scan and returns a CSV with the annotated nodules to check for.

What do we need to do:

- preprocessing: basically reading the ct scan in a SimpleITK compatible format and rescale it to 1x1x1mm
- lung segmentation: using the input from before, segment the lung and get a segmentation mask
- nodule segmentation: using the isotropic scan and the lung segmentation, compute the segmentation mask for the nodules, then measure the centroids

of the labels and convert those coordinates to real world coordinates.

- fp reduction: Using the scan and the centroids in the previous step, apply the nodule classifier and retrieve a probability for each of the nodules. Once we have this probability per candidate, discard any that are below a required threshold. If instead of using a probability threshold what we are interested is in a false positive rate, use the numbers in the evaluation phase to basically determine how the probability maps to a specific FPR, and adjust the output candidates with that.

To run this basically I've packaed everything in a conda environment. This has allowed me to list all the necessary packages and provide an easy way to create environments with all the necessary dependencies, even stuff like CUDA libraries, which is not native python, can be easily installed using conda. This also makes it very easy to then create a Docker image that has all the necessary packages to run this stuff.

What else? Well, the docker image contains the weights of the different neural networks. I've basically just included the best network for each of the steps, based on the evaluation of the results. Both the code, dependencies and weights is included in a Docker image, which can also have GPU support (very much recommended) by using nvidia-docker.

Once that is built, we have a ready to go image, which only needs to mount 2 volumes (folders) for the input image and the output result. Then it's just a matter of running a command and all of this code can be easily run. Apart from the ease in reproducibility (not only the final script can be executed, but everything else, such as evaluation scripts and the like), we gain a very convenient way to distribute the results and an even better way to test our system in other datasets with minimum hassle, since the whole pipeline has been integrated.

Currently on an i7 7700, 32GB of RAM, GTX 1080Ti, evaluating a scan from start to finish requires around 2mins of processing time.

1.2.4 Metrics

Small section to introduce the metrics I'll use and what are they used for and what drawbacks they have:

```
- DICE
- FROC
- Average FROC
- AUC
- TP, FP, sensitivity and F1
```

1.2.5 Lung segmentation

I might just put this after nodule segmentation and false positive reduction, since it basically just an addendum on nodule segmentation that needs to be done for the pipeline to work in an end to end fashin. Interestingly, this chapter could serve to demonstrate the transferability of deep learning techniques to other domains, which is not a bad thing. Essentially the network and everything is exactly the same thing as the nodule segmentation, but using the lung masks as ground truth, instead of nodule masks, so the problem is actually simpler.

Not much really. Basically the idea is that, if the previous network works well for something as complicated as segmenting nodules, segmenting the lungs themselves should be easier, but basically the same concepts should apply.

1.2.6 Nodule detection

I could say that based on the work I did in the LUNA challenge chapter, best approach right now seems UNET based. Explain again that for this part of the system what we are interested in is basically something with very high sensitivity. And finally I guess say that I went for a 2D network cause the images are big, it is a very deep network, and I wanted to avoid as much technical trouble as possible, especially since it was a first for me.

1.2.7 FP reduction

Similarly to an object detection problem (Hosang et al. [2016]), we've divided our pipeline in two phases: candidate proposal and false positive reduction. As we have seen in the previous chapter, our UNET-based proposal network primed sensitivity above all else, but now we need a classifier with high precision so that the signal-to-noise ratio of the system will be high enough to prove useful to a radiologist.

One of the main benefits of performing a previous step to detect candidates is the fact that the search space is reduced and that makes it computationally feasible to run image recognition algorithms with high computational costs within a reasonable timeframe.

In this chapter we'll cover two different approaches to false positive reduction. The first one will be a classifier trained on features manually extracted from the previous segmentation phase of the pipeline. The second one is based on a volumetric ResNet (Chen et al. [2018]). The original 2D version of this deep neural network (Wu et al. [2017]) achieved a deeper architecture bypassing the vanishing/exploding gradients problem (Bengio et al. [1994], Glorot and Bengio) by using a combination of normalization techniques (Ioffe and Szegedy [2015], LeCun et al. [2012], He [2014]) and the use of residuals.

1.3 State of the art

1.3.1 The LUNA grand challenge

This chapter will serve as an introduction to what is the LUNA grand challenge, its dataset, competition tracks and metrics. After that is out of the way, I'll go over the current top 20 and do a survey of the different techniques that compound the state of the art for this kind of problem. This will serve as an introduction to what I amb about to do.

Basically talk about the technique and how it has been changing diagnosis recently. This could be a copy pasta of Arindra et al. [2017] and explain a bit on how they've reworked on the LIDC dataset to prepare the data, what it does and what is missing (malignancy!), which is actually available in LIDC.

What is this dataset and why is it useful to evaluate CAD systems

Talk about the tracks and metrics. Again, this appears in Arindra et al. [2017], so I don't know how much I want to add

Interesting to go over the top 20 of LUNA as it stands right now. Thankfully most of the systems are closed so I don't have to explain them, but for the open ones, it would be good to go over the methods they present, and basically argument why I chose what I did Talk about the top 20. Basically put a table with the methods, describe them slightly. Then divide method by groups and expand more on that.

1.3.2 Nodule detection track

Review of the top20. Cover here any deep learning content I might have to.

1.3.3 FP reduction

Review of the top20 (again paper and all). Basically cover here any deep learning content I might have to.

1.4 Techniques?

1.4.1 Image registration

LUNA and other registration methods? ### Image segmentation Talk about LUNA and radiomics? ### Image recognition RESNET?

1.5 Outline

Talk about the chapters, and how the work is organized.

Chapter 2

Methods

2.1 Lung segmentation

Pretty much the same as the nodule segmentation, but actually with less preprocessing. I have to check whether or not I did stuff like laplacians and augmentations (I don't think so, really).

Mention that the lung segmentation performed in Luna is based on the method presented in Van Rikxoort et al. [2009].

Also, in terms of how I am going to evaluate this, 2 measures. One is the typical DICE score, which OK, is good. Problem is, the segmentation itself, although is based on state of the art methods (and here I should really ask with Mario how they were performed, and add this to the introduction of this section). So, since this is basically a preprocessing step, and what I am interested in is to actually detect nodules, what I am going to do is compare how much nodule mass the segmentation is cutting out, compared to the nodule mass lost in the original masks. And if it is close enough, basically I'm going to consider this good enough.

2.2 Nodule detection

Explain the basis of the unet network I am using. Explain also the batch normalization and relu layers I've introduced on the convolutional layers of the network. Also explain all the variations, both in terms of augmentation and preprocessing that I am applying.

Finally talk about the actual evaluation system, which in this case it ain't even visual. Mostly just sensitivity and average FP per scan, which is not nearly as important.

Also interesting and should be mentioned, we want to know if the variants are diverse. That is, if different variations of the network capture different nodules (to test if an ensamble would be a worthwhile approach).

2.3 False Positive reduction

2.3.1 Handpicked feature classifier

2.3.1.1 Selected features

As seen in the previous chapter, the probability map obtained by the segmented slices is not informative enough to calculate the likelihood of the predictions, but the shape of the labels themselves potentially hold information that can help us distinguish between real and false nodules. To explain this concept visually, we can compare the segmented nodules A and C in Figure 2.1. The first one is an example of a large nodule, mostly round, mostly contiguous in the Z-axis. Nodule C, on the contrary, while having a round segmentation in the axial plane, is almost flat, which typically translates to a false positive. Another frequent source of false positives are caused by the presence of airways in the lung. On a single slice they can be easily mistaken for a nodule, but if we pay attention to their coronal and sagittal projections we will appreciate large displacements, forming an elliptical shape. This effect can be observed to some degree in nodule B, and more agressively in nodule D.

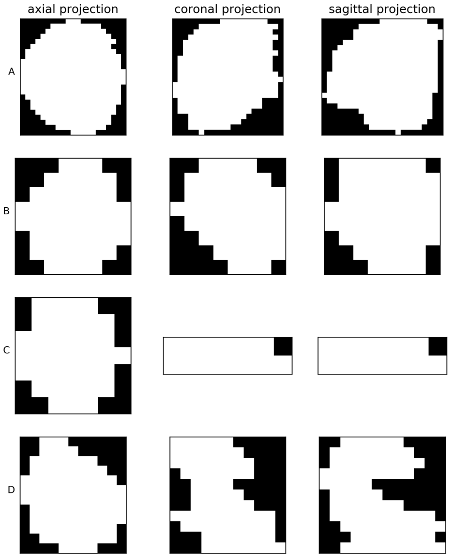

Figure 2.1: axial, coronal and sagittal projections of 4 nodule masks as segmented by our U-Net network. Even though the axial projection is similar in all the examples, the sagittal and coronal views offer a much larger degree of variance.

Based on the visual inspection of the masks obtained by our segmentation, we engineered the following features to characterize the nodules:

diameter mesures diameter (in mm) of the bounding box in the axial plane.

layers measures number of contiguous layers of the bounding box in the z-axis.

squareness measures how similar the shape is between the axial and its ortogonal planes. Values range between 0 and 1. 0 means ratio between axial and the ortogonal planes (sagittal and coronal) is the same. 1 would mean that one side is completely square, while the other flat. Formulated as:

$$squareness(length, width, depth) = abs\left(\frac{min\{width, length\}}{max\{width, length\}} - \frac{min\{depth, \frac{width+length}{2}\}}{max\{depth, \frac{width+length}{2}\}}\right)$$

extent measures the ratio between masked and unmasked area in a labeled bounding box. Formulated as:

$$extent = \frac{num\ masked\ pixels\ of\ bbox}{num\ total\ pixels\ of\ bbox}$$

axial eccentricity measures the geometric eccentricity of the segmented nodule projected on the axial plane. 0 would indicate the projection is a perfect circle.

sagittal eccentricity measures the geometric eccentricity of the segmented nodule projected on the sagittal plane. 0 would indicate the projection is a perfect circle.

It should be noted that these features are only capturing basic information about the shape of the segmentations. This model ignores texture or other finer-grained features based on shape.

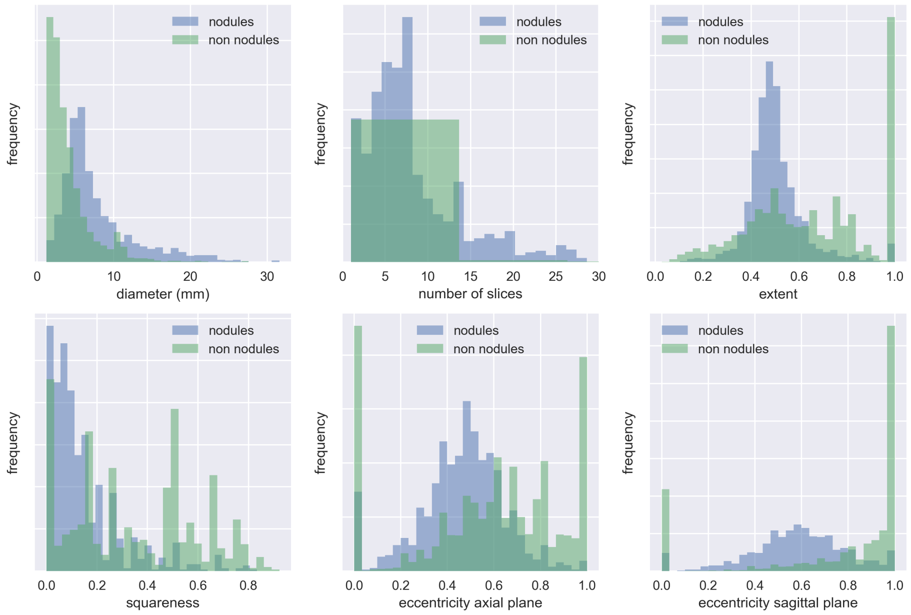

Figure 2.2: frequency distribution of the nodule candidates features, obtained by segmenting the entire LUNA dataset with the *augmented, 3ch, batch normalized, bce unet*. The histograms of TP and FP are overlapped and normalized.

2.3.1.2 Training the model

We're going to train multiple binary classifiers with the features presented above and compare their performance quantitatively employing the AUROC. We're also going to plot the entire ROC curve to qualitativaly assess the behaviour of the classifier as the false positive rate increases. The tests will be performed both on the training and test sets, so we can also compare the performance of both side-by-side and assess the tendency to overfit of each of the classifiers.

The training and testing will be performed on the candidates obtained by the segmentation network *augmentation_normalization_bce_3ch_laplacian_f6c98ba* from the previous chapter. Candidates from subsets 0 to 8 will be used as training data, while candidates in subset 9 will serve as our test dataset. We're not going to tune hyperparameters on the classifiers, so no validation set will be employed. This basically leaves us a dataset with a 4 to 1 ratio in FP vs TP that we will not rebalance. More details about the dataset can be found in Table 2.1.

Table 2.1: Baseline from running the segmentation network *augmentation_normalization_bce_3ch_laplacian_f6c98ba*. The classifier will be trained and evaluated on the features extracted form those candidates.

	Training (subsets 0 to 8)	**Test (subset 9)**
number of scans	776	84
number of candidates	5415	599
TP	1032	93
FP	4383	506
average FP per scan	5.6482	6.0238

We've selected a list of 5 classification algorithms (see Table 2.2), from simple logistic regression models to more advanced tree boosting classifiers, in an attempt to understand what sort of classification strategy works best both in terms of performance and generalization. We've used the `scikit-learn` (Nielsen [2016]) implementation of those algorithms, initialized with default parameters, for training and evaluation purposes.

Table 2.2: Types of classifiers trained on the candidates' dataset

Classifiers
Logistic regression
Decision tree
Random forest
AdaBoost
Gradient boosting

2.3.2 Radiomics based classifier

All the actual features extracted: - original_firstorder_10Percentile - original_firstorder_90Percentile - original_firstorder_Energy - original_firstorder_Entropy - original_firstorder_InterquartileRange - original_firstorder_Kurtosis - original_firstorder_Maximum - original_firstorder_Mean - original_firstorder_MeanAbsoluteDeviation - original_firstorder_Median - original_firstorder_Minimum - original_firstorder_Range - original_firstorder_RobustMeanAbsoluteDeviation - original_firstorder_RootMeanSquared - original_firstorder_Skewness - original_firstorder_TotalEnergy - original_firstorder_Uniformity - original_firstorder_Variance - original_glcm_Autocorrelation - original_glcm_ClusterProminence - original_glcm_ClusterShade - original_glcm_ClusterTendency - original_glcm_Contrast - original_glcm_Correlation - original_glcm_DifferenceAverage - original_glcm_DifferenceEntropy - original_glcm_DifferenceVariance - original_glcm_Id - original_glcm_Idm - original_glcm_Idmn - original_glcm_Idn - original_glcm_Imc1 - original_glcm_Imc2 - original_glcm_InverseVariance - original_glcm_JointAverage - original_glcm_JointEnergy - original_glcm_JointEntropy - original_glcm_MaximumProbability - original_glcm_SumAverage - original_glcm_SumEntropy - original_glcm_SumSquares - original_gldm_DependenceEntropy - original_gldm_DependenceNonUniformity - original_gldm_DependenceNonUniformityNormalized - original_gldm_DependenceVariance - original_gldm_GrayLevelNonUniformity - original_gldm_GrayLevelVariance - original_gldm_HighGrayLevelEmphasis - original_gldm_LargeDependenceEmphasis - original_gldm_LargeDependenceHighGrayLevelEmphasis - original_gldm_LargeDependenceLowGrayLevelEmphasis - original_gldm_LowGrayLevelEmphasis - original_gldm_SmallDependenceEmphasis - original_gldm_SmallDependenceHighGrayLevelEmphasis - original_gldm_SmallDependenceLowGrayLevelEmphasis - original_glrlm_GrayLevelNonUniformity - original_glrlm_GrayLevelNonUniformityNormalized - original_glrlm_GrayLevelVariance - original_glrlm_HighGrayLevelRunEmphasis - original_glrlm_LongRunEmphasis - original_glrlm_LongRunHighGrayLevelEmphasis - original_glrlm_LongRunLowGrayLevelEmphasis - original_glrlm_LowGrayLevelRunEmphasis - original_glrlm_RunEntropy - original_glrlm_RunLengthNonUniformity - original_glrlm_RunLengthNonUniformityNormalized - original_glrlm_RunPercentage - original_glrlm_RunVariance - original_glrlm_ShortRunEmphasis - original_glrlm_ShortRunHighGrayLevelEmphasis - original_glrlm_ShortRunLowGrayLevelEmphasis - original_glszm_GrayLevelNonUniformity - original_glszm_GrayLevelNonUniformityNormalized - original_glszm_GrayLevelVariance - original_glszm_HighGrayLevelZoneEmphasis - original_glszm_LargeAreaEmphasis - original_glszm_LargeAreaHighGrayLevelEmphasis - original_glszm_LargeAreaLowGrayLevelEmphasis - original_glszm_LowGrayLevelZoneEmphasis - original_glszm_SizeZoneNonUniformity - original_glszm_SizeZoneNonUniformityNormalized - original_glszm_SmallAreaEmphasis - original_glszm_SmallAreaHighGrayLevelEmphasis - original_glszm_SmallAreaLowGrayLevelEmphasis - original_glszm_ZoneEntropy - original_glszm_ZonePercentage - original_glszm_ZoneVariance - original_ngtdm_Busyness - original_ngtdm_Coarseness - original_ngtdm_Complexity - original_ngtdm_Contrast - original_ngtdm_Strength - original_shape_Elongation - original_shape_Flatness - original_shape_LeastAxis - original_shape_MajorAxis - original_shape_Maximum2DDiameterColumn - original_shape_Maximum2DDiameterRow - original_shape_Maximum2DDiameterSlice - original_shape_Maximum3DDiameter - original_shape_MinorAxis - original_shape_Sphericity - original_shape_SurfaceArea - original_shape_SurfaceVolumeRatio - original_shape_Volume

2.3.3 ResNet based classifier

We're going to train multiple volumetric ResNet networks with different depths and compare their performance quantitatively emplying the AUROC. Similarly to what we've done in the manual feature classifier, we'll also plot the entire ROC curve of the classifier. As before, both training and testing curves will be plotted side by side, to assess the overfitting of the model.

Regarding the network architecture itself, we introduced the suggestions by Chen et al. [2018] and added a batch normalization and ReLU layer before each convolutional layer on the residual module, to facilitate convergence and weight stability while training. The same network was trained on different layer depths: 34, 50, 101 and 152.

As training data we will use the annotations provided by LUNA for the false positive reduction track of the challenge. They contain the world coordinates of the candidate centroid and a label indicating whether or not it is a nodule. See Table 2.3 for details regarding the distribution of this dataset. We will evaluate the model against the candidates obtained by the segmentation network *augmentation_normalization_bce_3ch_laplacian_f6c98ba*, just as in the previous section, so that we can compare the performance between the two different methods.

Table 2.3: Number of entries per class in the candidate annotations dataset, divided by split. The class imbalance between the two categories is very prominent, which we'll have to take into account when training the network.

dataset split	FP	TP	ratio
training (subsets 0 to 7)	603345	1218	495 to 1
validation (subset 8)	74293	195	381 to 1
test (subset 9)	75780	144	526 to 1

Since we are not using an ensemble of multiple models, the volumetric patch we will use as input should capture the entirety of the nodule. Based on the data observed in Figure 2.2, the dataset does not contain diameters above 32mm, so we will fix the input resolution to be `32x32x32x1`. The scans have been rescaled to a spacing of 1x1x1mm and the images only have 1 color channel, with values corresponding to the Hounsfield value of the voxel (no normalization or clipping applied in the preprocessing).

The training is performed for a maximum of 50 epochs, only saving the weights in the iterations with better validation loss. We're using Adam (Kingma and Ba [2014]) as our method for stochastic optimization, initialized to a learning rate of `1e-3`. Early stopping is applied if the validation loss is not shown to improve in 10 consecutive epochs. The batch size for resnets {34, 50 and 101} was 64, while the batch size for resnet 152 was 32 due to memory constrains on the GPU side. Binary crossentropy was used as the loss function. The hardware employed during training consisted on an Intel i7 7700, 32GB of RAM and a Nvidia 1080Ti GPU.

To offset the data imbalance observed in the dataset (see Table 2.3) we will oversample the nodule annotations with replacement so the training and validation ratio is 2 to 1 (FP vs TP). This effectively means that a nodule annotation will be seen during training 250 times per each non-nodule one, which could very well induce the network to overfit. We mitigate this effect by using 3D image augmentation. As detailed in Table 2.4, affine transformations are randomly applied to the input cube before passing it to the neural network. Since this transformations would be lossy if applied to the actual cube of 32x32x32, we actually retrieve a larger cut of 46x46x46, apply the augmentation, and return a centered view of 32 pixels per side. The augmentation cube side needs to be larger than the diagonal of the input one for this to be valid. Also important, the augmentations are randomly applied to each sample each time and the dataset is shuffled on each epoch.

Table 2.4: Range of transformations randomly applied to both the axial and coronal planes of the input volume

transformation	range
rotation	[-90º, +90º]
shearing	[-20%, +20%]
scaling	[-10%, +10%]
flip vertically	[True, False]
flip horizontally	[True, False]
translation width	[-2px, +2px]
translation height	[-2px, +2px]

It should also be noted that the training and validation have been performed on a smaller fraction (35%) of the original data. This is the case purely due to hardware limitations when performing the experiment. Basically, extracting small patches of data from a much larger image is only fast if said image is already loaded, so we reduced the dataset size until it could fit in memory (32GB). Preloading the scans in-memory instead of reading them from disk supposed a speed-up larger than 2 orders of magnitude per epoch, so we considered the trade-off worthwhile.

2.4 Overall Discussion

Maybe explain a bit of the tradeoffs? Essentially the feature based is much easier to explain (the features are still based on the magic performed by the UNET, which is not a good thing in terms of reasoning about the algorithm, but basically it stops there, whereas the NN is another black box on top of an already black box). Still, performance improvements compound in a system with long pipelines, so it can't be dismissed.

Also interesting is seeing how we hit a wall in terms of performance after a certain amount of layers. It seems like the network saturates at some point. Basically this could be due to A) image size and, surely also helps, the fact that we are working with a reduced dataset, which won't help us to train properly the model (although at least that makes it cheaper and faster, which is also nice and can't be dismissed).

Chapter 3

Results

The results chapter is divided in five sections. The first three report the metrics for each individual problem of the CAD pipeline, paying especial emphasis to the differential in performance between different approaches. The 4th section compares the metrics of the system against other competitors of the LUNA grand challenge to contextualize it within the state of the art. Finally, there is a 5th section that assesses qualitatively the efforts placed to integrate the system into a clinical context.

3.1 Lung segmentation

The results of using the U-Net for lung segmentation purposes show a network that achieves a Dice score over 98% in 40 epochs of training, although it only takes two epochs for the scores to be above 96%. As we can see in Table 3.1, there are no signs of overfitting. In fact, the Dice coeficients for both the validation and testing sets are 0.5% better than the results on the training dataset, although this could be explained due to the extra variability found in the larger training dataset (616 CT scans vs 88 each on the validation and test splits).

Table 3.1: mean Dice coefficient for each dataset split. Each axial slice in the dataset is evaluated individually and then the mean is calculated in two steps. First the mean over the whole scan and then the mean over the dataset .

dataset split	**Dice score**
training	0.977392
validation	0.983458
test	0.984037

The U-Net lung segmentation is especially accurate in the superior and middle lobe of the lung, as we can appreciate in the bottom left slice on Figure 3.1. Irregular areas with lower contrast, like the one in the left lung shown in the bottom right slice (same figure) can confuse the network. The trend is towards more expansive masks, which in our case is a valid trade-off, since this is only a preprocessing step done towards reducing the complexity of the nodule segmentation task, and we don't want to discard potential lung mass which may contain a nodule.

Lower lung lobes are generally those with lower Dice scores (see top left and right slices in Figure 3.1). It is also possible to observe holes inside a segmented lung (bottom right), which could be fixed by applying a morphological closing in the mask.

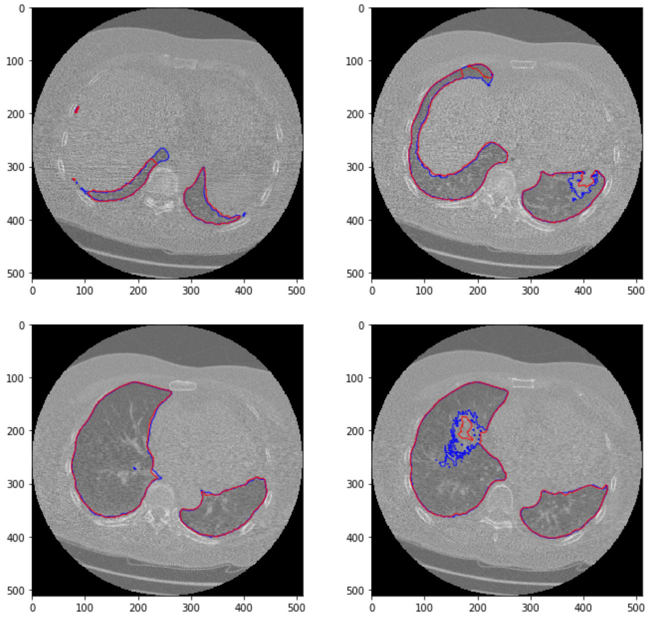

Figure 3.1: Axial slices of a lung with both masks superposed. The blue segmentation corresponds to the ground truth used to train the U-Net, while the red segmentation is the prediction returned by the model.

3.2 Nodule detection

Our main metric for the nodule detection module is its sensitivity. This will determine the upper limit performance of the system. We will also keep track of the number of false positives returned, since this directly affects the complexity of the false positive reduction module. More false positives will require a more complex model in order to be competitive with the state of the art.

In Table 3.2 we have both metrics divided by the loss function used during the training phase and the different image processing variations applied. We haven't included the figures of the network trained without batch normalization as they were not very telling themselves, but we still wanted to report those negative results, because they were the key that allowed the network to learn. As we can see in Figure 3.2, a U-Net without normalization, neither on the input image nor on its convolutional layers, is incapable of learning the true representation of a nodule. Basically the only information it can extract form the original image is a rough segmentation of the lung parenchyma, which happens to be the area of major contrast in the original slice (as a reminder, a lung mask is applied as a preprocessing step, fixing the value of any voxel outside the lung tissue to -4000HU). This was the key discovery that made this approach feasible.

Both loss functions achieve similar top sensitivity scores in Table 3.2. In general, binary cross entropy displays a more stable behaviour during training and it penalizes false positives more heavily, as we can see from looking at the false positive rates of both networks (3 to 1 ratio favoring binary cross entropy). The downside of this heavy penalization of false positives is a slight drop in sensitivity (*0.915* vs *0.930*) that caps the maximum performance of the system.

It is also worth mentioning the effect of applying augmentation to mitigate overfitting. If we take a look at the differences between the augmented binary cross entropy variation vs the non-augmented (Table 3.2) the differential in training sensitivity barely achieves a 0.5%, but the gap between the training and testing scores goes from a 19% to 11%, and down to 6% in its best performing variation.

We also analyzed whether the different variations introduced diversity in the nodules detected by the network. As we can see in Figure 3.3, the nature of each variation is purely additive. The network is able to detect more nodules while still discerning the previous subset. In practice, this would mean that we might need to develop an entirely different model to detect the nodules we are currently missing, so that an ensemble based on both models would beat their individual performance.

Finally, we compared the individual performance of our best segmentation networks against the results presented in the LUNA16 challenge survey (Arindra et al. [2017]) in Table 3.3. Our network is able to beat the individual systems described in the paper both in sensitivity and in number of false positives reported by scan. It is especially in this last metric where the differences are the

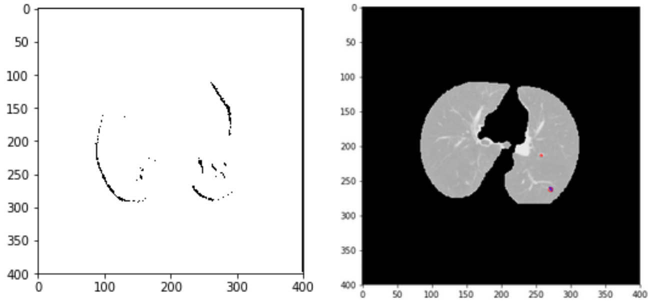

Figure 3.2: nodule segmentation network results when the U-Net is trained without applying batch normalization on each convolutional layer on the left. On the right we see the output of the same network after enabling batch normalization. The red contour corresponds to the predicted mask while the blue markings match the ground truth used for training.

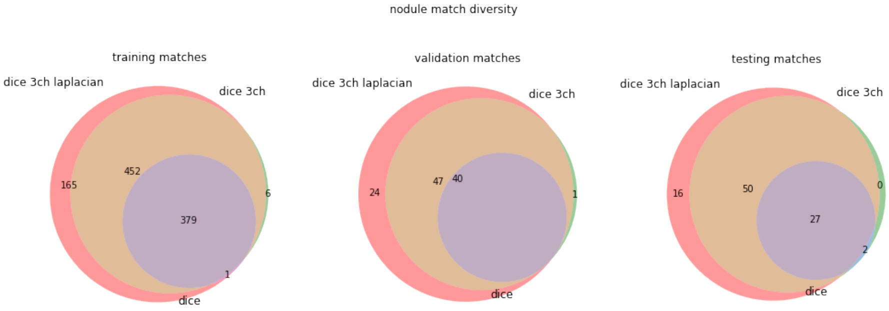

Figure 3.3: This venn diagram showcases the additive nature of the variations performed in the U-Net. The lack of diversity in its predictions discouraged the use of an ensemble to increase its overall performance.

Table 3.2: U-Net nodule segmentation variations along with their sensitivity and number of FP per slice. It should be noted that nodule candidates may refer to the same annotation, so it is possible for the sensitivity to take values over 1.

			sensitivity	FP
			mean	mean
loss function	variation	set		
crossentropy	no augmentation, normalization	test	0.783051	7.329545
		train	0.977351	6.707865
		validation	0.796944	6.840909
crossentropy	augmentation, normalization	test	0.859275	6.011364
		train	0.972629	5.703652
		validation	0.922778	5.488636
crossentropy	augmentation, normalization, 3ch, laplacian	**test**	**0.915490**	**5.750000**
		train	0.974303	5.515449
		validation	0.940417	5.181818
dice	no augmentation, normalization	test	0.740254	7.125000
		train	0.828008	7.063202
		validation	0.795972	6.784091
dice	augmentation, normalization	test	0.339407	1.443182
		train	0.390669	2.252809
		validation	0.399306	1.750000
dice	augmentation, normalization, 3ch	test	0.803672	34.125000
		train	0.818526	34.228933
		validation	0.806389	33.715909
dice	augmentation, normalization, 3ch, laplacian	**test**	**0.930791**	**15.420455**
		train	0.944604	17.234551
		validation	1.044861	14.193182

most striking. The U-Net trained with binary cross entropy is able to match the sensitivity of ETROCAD (best reported) within a 1%, but it is able to do so with 47 times lesser amount of candidates. The levels of accuracy provided by our network will, in fact, enable us to develop false positive reduction methods based on the nodule segmentation themselves, and still be competitive in the general LUNA scoreboard.

Table 3.3: Candidate detection systems performance as reported by Arindra et al. [2017]. Even though each individual system is offering worse performance than our custom U-Net, an ensemble combining them reported sensitivity rates up to 0.983.

system	sensitivity	avg num candidates / scan
ISICAD	0.856	335.9
SubsolidCAD	0.361	290.6
LargeCAD	0.318	47.6
M5L	0.768	22.2
ETROCAD	0.929	333.0
lucanode bce	0.915	7.0
lucanode dice	0.930	18.0

3.3 FP Reduction

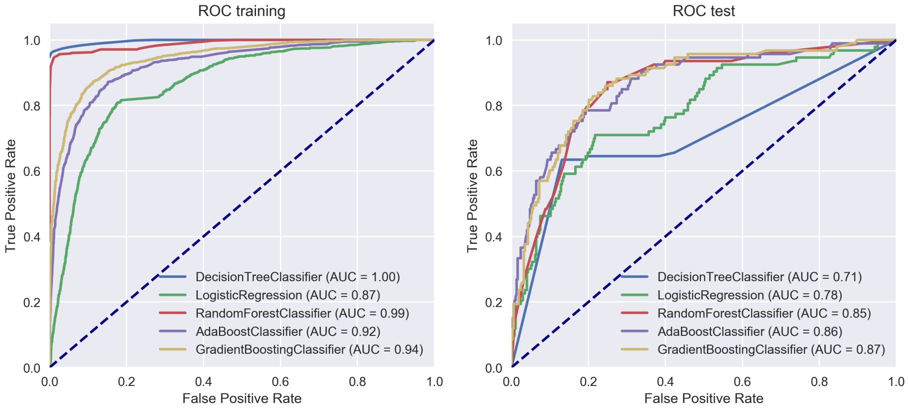

Figure 3.4: ROC curves and AUC of the handpicked feature classifiers

Explain how the classifiers compare. Basically explain the overfitting effect of the tree classifiers and how the boosting algorithms seem to be best and overfit less

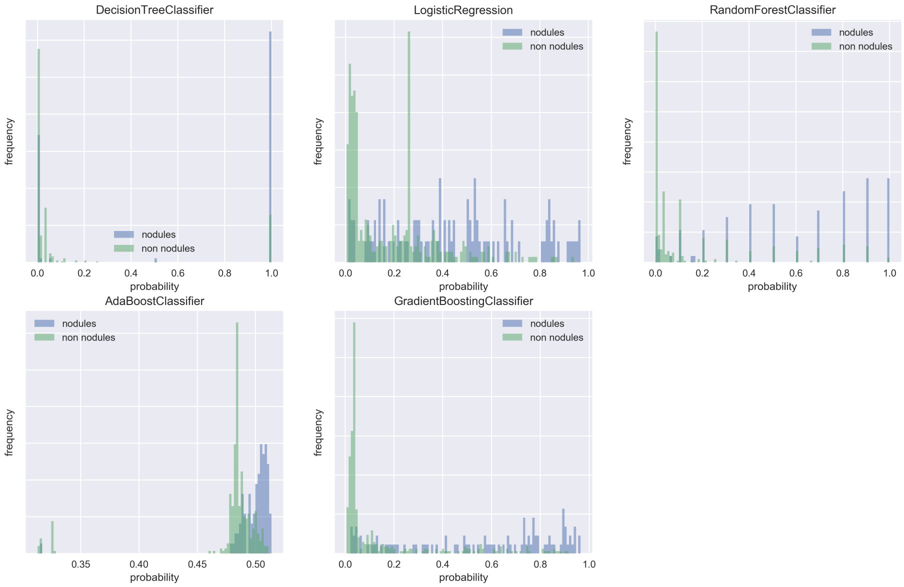

Figure 3.5: histogram pdf handpicked features

Based on the probability distribution of the histogram we can demonstrate that both boosting algorithms are the ones that are better at separating between both groups. Basically it can be seen on the graphs that they follow two different distributions, with the least overlap between them.

In here the distinction is not as straightforward. There is basically a sweet spot at 50. 101 and 152 seem too deep for the amount of data we were training it with (just 35% of the half a milion annotations we had to start with), and really are not helping much. Basically it plateaus.

Again, put here the probability histograms. In this case, the differences are more subtle, as they should be, since the curves are much closer between them. It should be noted though that the overfitting effect is much lower than what we've observed with the previous method. Again, this is to be expected since we have a much larger dataset to train the classifier with. In fact, paradoxically, the better our segmentation is, the less data we have to train the dataset, which might make our FP reduction worse. Which is why it could be interesting to decouple both parts. At the same time, if we use features engineered from the segmentations, they are coupled through and through, so that approach will always have those limitations.

Basically, comparison side by side of both methods. resnet is better, as it should be, since it is a much more complex model, with N features (look up how many, actually), compared to the 6? of the other. Also important, it overfits less, again probably due to availability of data. The differences don't seem that major, but

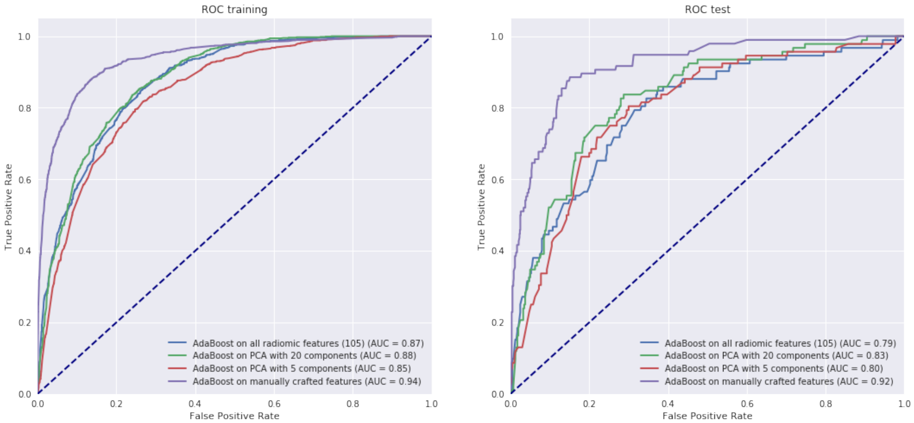

Figure 3.6: ROC curves and AUC of the classifier based on radiomics

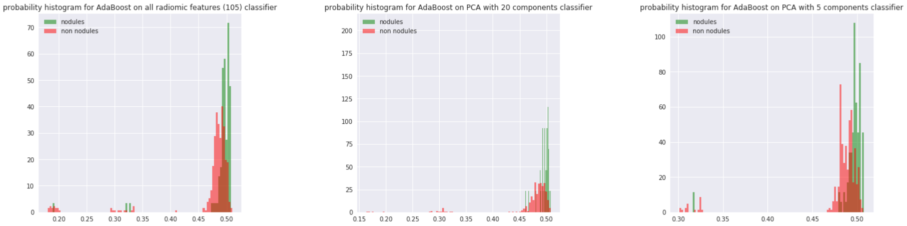

Figure 3.7: histogram pdf radiomics classifier

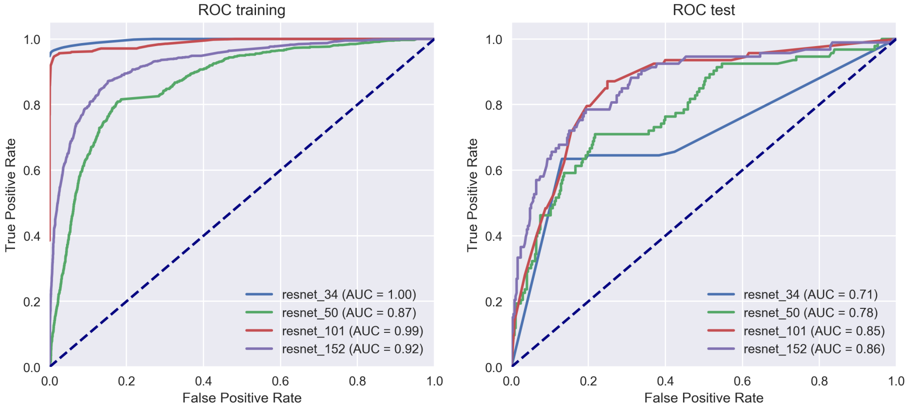

Figure 3.8: ROC curves and AUC of the residual networks

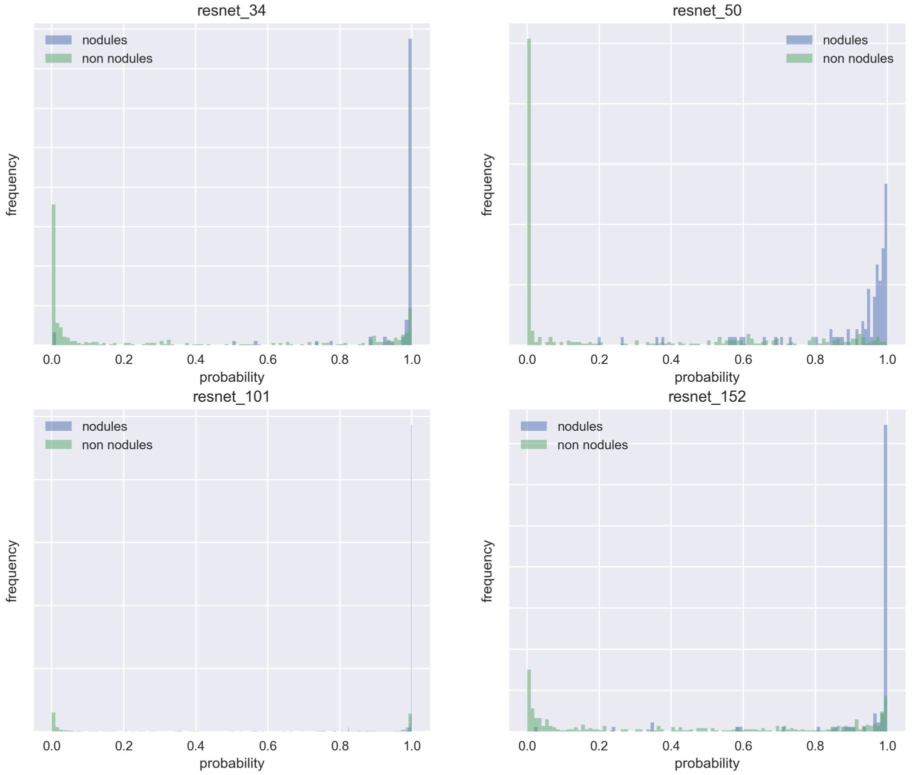

Figure 3.9: histogram pdf residual networks

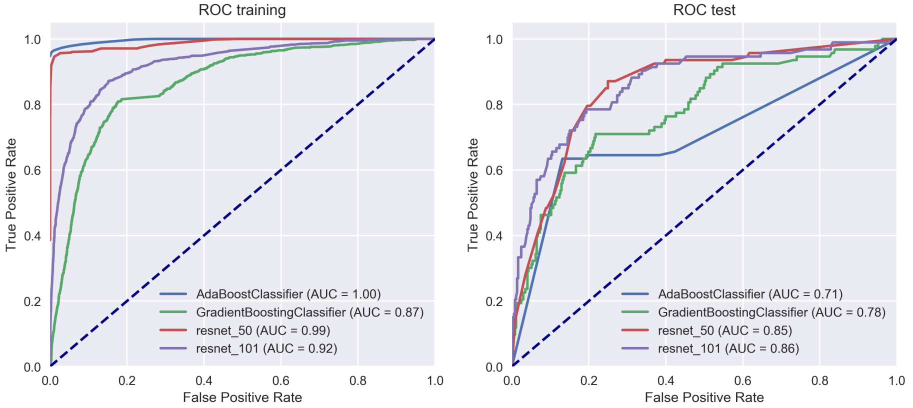

Figure 3.10: ROC curves and AUC comparing the best 2 variations of FP reduction method

basically they compound with whatever performance the segmentation has, so a few percent points drop on the AUC are important. Also, the slope looks better, as it will be able to achieve better performance at lower FPR, which is very important for a system such as this. We need a slope as flat as possible, so that the results are very good even with very low rates of false positives.

3.4 LUNA performance comparative

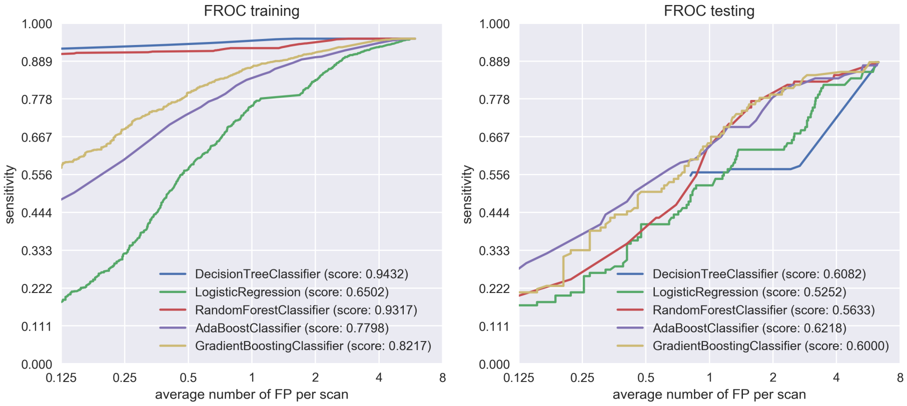

Figure 3.11: FROC curves and averaged sensitivity at selected FP rates of the handpicked features classifier

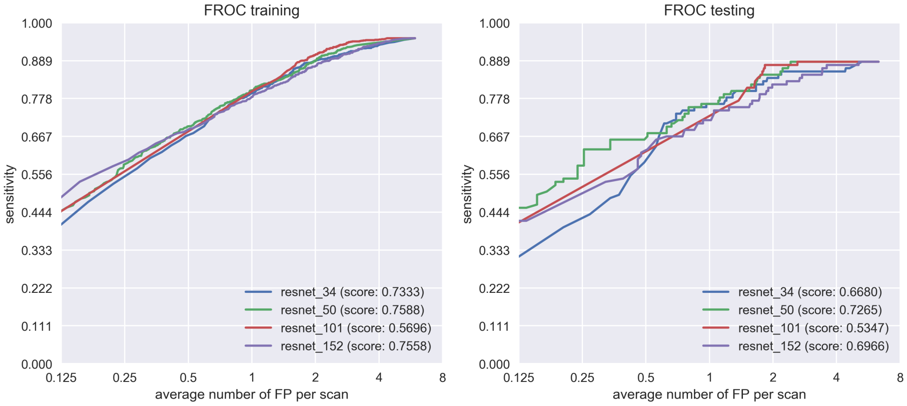

Figure 3.12: FROC curves and averaged sensitivity at selected FP rates of the residual networks

I can actually draw the curves manually of different competing systems by retrieving the numbers from the table in the LUNA paper, which really, would

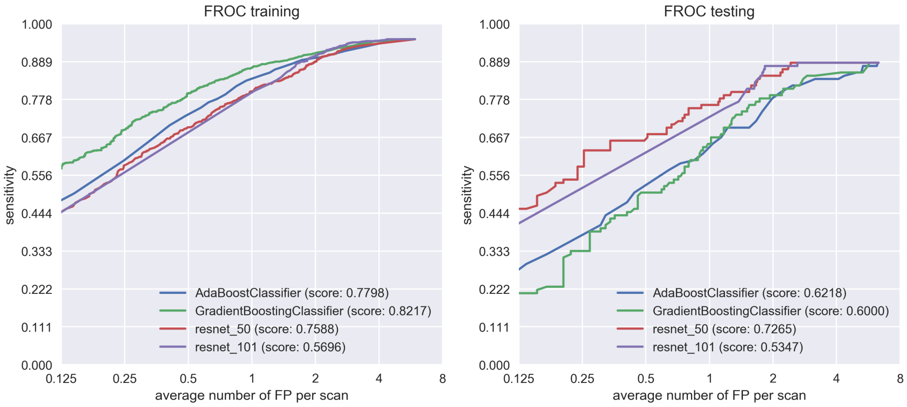

Figure 3.13: FROC curves and averaged sensitivity at selected FP rates comparing the best 2 variations of FP reduction method

be the best approach to show my perf VS other systems.

3.5 Integration into a clinical workflow

Hello

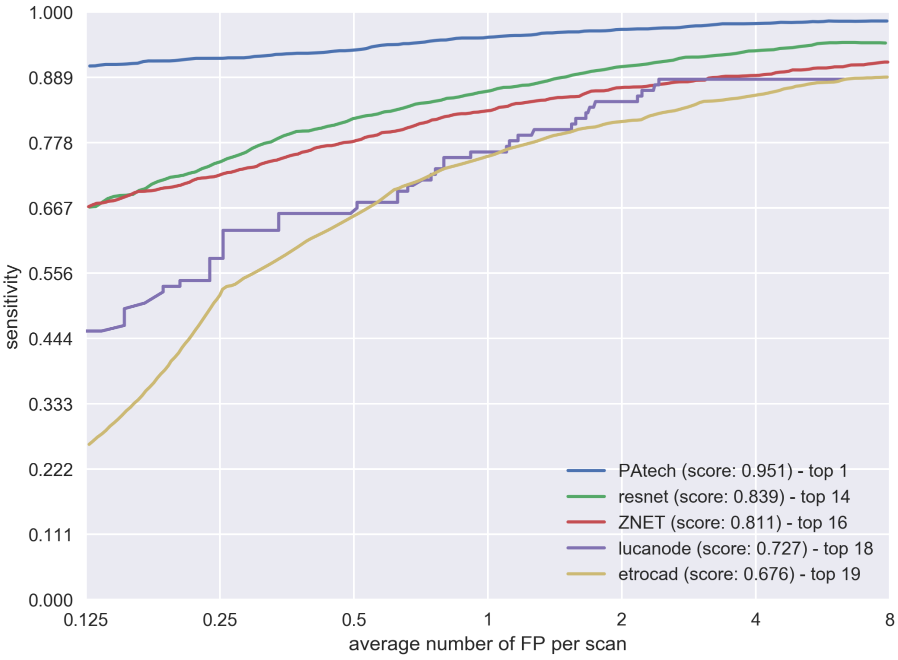

Figure 3.14: Best lucanode iteration, trained with bce on segmentation, resnet50 in the FP reduction vs other approaches. I've added top 1, immeddiately below and top14-16, which is more or less where we could reach with some improvements

Chapter 4

Conclusions

4.1 Lung segmentation

Edges need more training. Especially end of lung is hard for the segmentation. Still, this could be easily fixed by retraining, we just need to rebalance the dataset to oversample on the last 20% of the lung, which is otherwise underrepresented. Also, it would be easy to apply some expansion on the mask, and maybe a bit of smoothing, to correct out rough edges. Finally, we know that masks are not supposed to have holes inside them, so that could be automatically removed. Also, no filtering has been applied to the original image, nor image augmentation of any kind. Probably both ideas would help a lot.

Even though this part can be improved, in the grand scheme of the system, it wouldn't yield much benefit. Basically we are barely losing any significant mass of the segmented nodules with the new masks, so the possible gains are minimal. We are much better off focusing on a better FP reduction.

Maybe discuss about the failings of the current system, such as the holes that sometimes appear inside a lung. Basically all of this stuff could be corrected quite easily by applying some transformations on the mask to fix this obviously wrong problems. Also worth noting is the fact that some expansion of the mask could be done, to diminish problems on the parenchyma, which has nodules that thend to be cancerous (moreso than other areas in the lungs). Also basically say that we could relax our requirements in this part as long as we don't miss nodules nor too much mass, which really would screw us on the overall performance of the system. 'Cause this is a big one on this system. Since it is actually multiple problems in one, you need to know where to put the effort, and this is not really one of the areas that would result in big wins for the system, so let's rule it out and worry about other stuff instead, like FP reduction.

4.2 Nodule detection

In here Ok, so basically based on the results, speculate on what is missing. Also mention that we don't really have a conclusive view of wheter having a lesser FP rate (despite lower sensitivity) is a worthwhile tradeoff or not.

Also failings of the current system actually include airways and nodules which are too flat, so basically say that yes, 3D would be better, but the big question is if 3D would actually increase the sensitivity of the system or what. That really is the key. 'Cause if it does not, training a 3D network is VERY expensive (which by the way, should be mentioned in the methods part of this) and not only is it expensive to train, it is expensive to run and evaluate, so really, beware 3D fanboys.

4.3 FP reduction

4.4 Overall

We are not doing ensembling, which as the LUNA paper demonstrates, it improves the performances of those systems by a huge margin. As we've seen, variations on our approaches offer only incremental improvements, but they are not very good in terms of diversity. It would be interesting to actually investigate further if the different FP reduction approaches could be combined to offer better performance. So far it is a bit of an unknown quantity that angle.

Basically as we can see with the results, the nodule detection and segmentation part is very competitive. Basically it shows all the work I've performed, with the variations and filters I've applied. It is also quite novel the fact that the architecture takes whole slices, instead of using patches on the slice, which is more typical and also easier to train. Also very important, how crucial it is to introduce normalization and augmentation to train systems. It is absolutely necessary and it makes a world of difference. Also, performance is key. If you don't have a performing system, you're dead in the water, you can't train fast and if you don't, you just don't have the feedback to develop a system such as this. Lots of trial and error, which is a bit disheartening when trying to explain why the system works as it does.

Chapter 5

General discussion

5.1 Lung segmentation

Mention that the initial idea was to perform a multiatlas lung segmentation, along the lines of Rohlfing et al. [2004] and Van Rikxoort et al. [2009], but that I could not justify the cost and complexity to do this, when the neural network performed very well with very little overhead, and it was also fast to run.

Also mention here that it is very interesting how the architecture translates so well, even though the data is different. It was supposed to, but it is quite fantastic that, just by writing a different data loader, you can train a network that works on an entirely different problem and still yield state of the art results.

For the problems exposed in Figure 3.1, my approach would be to rebalance the dataset, to prioritize slices from the inferior lobe. Also, applying a laplacian of gaussians as part of the preprocessing would help to increase the initial contrast of the image, which would make the task easier to start with. If anything, some augmentation could be used as well so that the artificial rebalance of the dataset does not cause overfitting. Finally, there are some obvious errors, such as holes inside the masks, which could be dealt with with some image processing, such as closing any masks returned by the algorithm. It would be safe and would increase the scores a little bit.

At the end of the day, all of this is pending work because it is just not the bottleneck of the system. Other areas are much more of a priority (FP reduction) if we want to increase the overall score of the system. Basically in here, even if the segmentation is not perfect, we can apply mask dilation over the automated segmentation and work with that.

Bibliography

Arindra, A., Setio, A., Traverso, A., et al. 2017. Validation, comparison, and combination of algorithms for automatic detection of pulmonary nodules in computed tomography images: The LUNA16 challenge..

Bengio, Y., Simard, P., and Frasconi, P. 1994. Learning Long Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks 5*, 2, 157–166.

Chen, H., Dou, Q., Yu, L., Qin, J., and Heng, P.A. 2018. VoxResNet: Deep voxelwise residual networks for brain segmentation from 3D MR images. *NeuroImage 170*, 446–455.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks..

He, K. 2014. Delving Deep into Rectifiers : Surpassing Human-Level Performance on ImageNet Classification..

Hosang, J., Benenson, R., Dollar, P., and Schiele, B. 2016. What Makes for Effective Detection Proposals? *IEEE Transactions on Pattern Analysis and Machine Intelligence 38*, 4, 814–830.

Ioffe, S. and Szegedy, C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift..

Jacobs, C. 2015. *Automatic detection and characterization of pulmonary nodules in thoracic CT scans.*.

Kingma, D.P. and Ba, J. 2014. Adam: A Method for Stochastic Optimization. 1–15.

LeCun, Y.A., Bottou, L., Orr, G.B., and Müller, K.R. 2012. Efficient backprop. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 7700 LECTU*, 9–48.

Nielsen, D. 2016. Tree Boosting With XGBoost Why Does XGBoost Win "Every" Machine Learning Competition? *NTNU Tech Report* December, 2016.

ROHLFING, T., BRANDT, R., MENZEL, R., AND MAURER, C.R. 2004. Evaluation of atlas selection strategies for atlas-based image segmentation with application to confocal microscopy images of bee brains. *NeuroImage 21*, 4, 1428–1442.

VAN RIKXOORT, E.M., DE HOOP, B., VIERGEVER, M.A., PROKOP, M., AND VAN GINNEKEN, B. 2009. Automatic lung segmentation from thoracic computed tomography scans using a hybrid approach with error detection. *Medical Physics 36*, 7, 2934–2947.

WU, S., ZHONG, S., AND LIU, Y. 2017. Deep residual learning for image recognition. 1–17.