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 Lung cancer

Lung cancer is the most deadly cancer in the world. Bring some figures and talk about trends

1.2 Computed Tomography

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

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.

1.4 Lung nodule CAD

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.

1.5 Deep learning in medical imaging

A brief introduction on deep learning in computer vision, how it started and how it is currently being used to

1.6 Lung cancer detection challenges

Talk about the ongoing competitions about lung cancer, based on the LIDC study (most of them). Talk about LUNA, Kaggle, ANODE09, the ISBI. Also the different variants, with just detection, inferring malignancy, etc. I wouldn't expand too much apart from this and just create a chapter to really drill into what they use and why it works.

1.7 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

1.8 Pipeline:

Describe the pipeline for a CAD system. The 3 problems we are facing and how they stack together. Also mention how we are scoring each of this problems individually.

1.9 Outline

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

Chapter 2

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.

2.1 The LUNA grand challenge

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

2.2 The LUNA dataset

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.

2.3 Grand challenge competition tracks

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

2.4 A survey of the state of the art

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.

Chapter 3

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.

3.1 Introduction

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.

3.2 Methods

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).

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.

3.3 Results

Put dice score of both. Put lost nodule mass in both, compare the numbers. Then actually evaluate the cuts of some of the slices and basically show where it fails more.

3.4 Discussion

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.

Chapter 4

Nodule segmentation

4.1 Introduction

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.

4.2 Methods

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).

4.3 Results

Put the comparison tables. Also put the venn diagrams. There is no diversity. Finally plot examples of what it is that each variation captures exclusively, so that we can explain what it is we are adding on each step and also what is that we are missing most (how could we improve it). See Table 4.1

4.4 Discussion

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.

Table 4.1: Nodule segmentation network

network	set	sensitivity mean	FP mean
augmentation normalization bce	test	0.859275	6.011364
	train	0.972629	5.703652
	validation	0.922778	5.488636
augmentation normalization bce 3ch laplacian	test	0.915490	5.750000
	train	0.974303	5.515449
	validation	0.940417	5.181818
augmentation normalization dice	test	0.339407	1.443182
	train	0.390669	2.252809
	validation	0.399306	1.750000
augmentation normalization dice 3ch	test	0.803672	34.125000
	train	0.818526	34.228933
	validation	0.806389	33.715909
augmentation normalization dice 3ch laplacian	test	0.930791	15.420455
	train	0.944604	17.234551
	validation	1.044861	14.193182
no augmentation normalization binary crossentropy	test	0.783051	7.329545
	train	0.977351	6.707865
	validation	0.796944	6.840909
no augmentation normalization dice	test	0.740254	7.125000
	train	0.828008	7.063202
	validation	0.795972	6.784091
unet 3ch axial 400x400 laplacian	test	0.771234	85.488636
	train	0.801148	84.466292
	validation	0.861389	82.840909

Chapter 5

False Positive reduction

5.1 Introduction

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.

5.2 Handpicked feature classifier

5.2.1 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 5.1. 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.

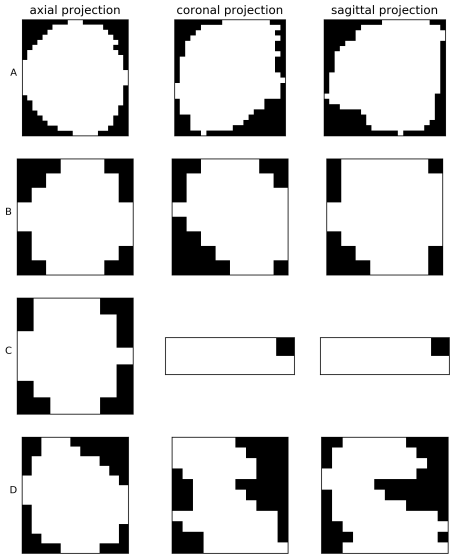

Figure 5.1: axial, coronal and sagittal projections of 4 nodule masks as segmented by our U-Net network. Even though the axial projection is similar in all the examples, the sagittal and coronal views offer a much larger degree of variance.

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.

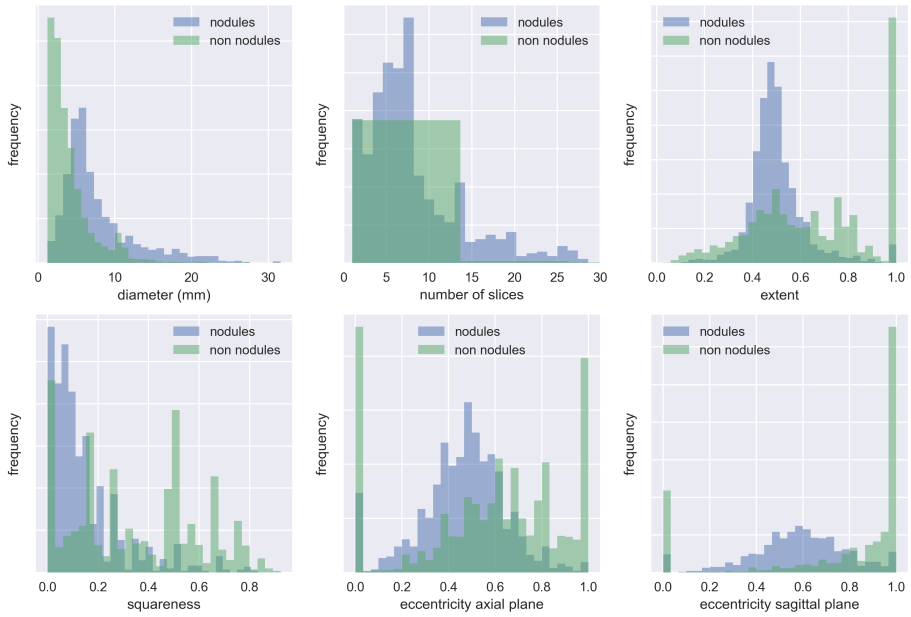

Figure 5.2: frequency distribution of the nodule candidates features, obtained by segmenting the entire LUNA dataset with the *augmented, 3ch, batch normalized, bce unet*. The histograms of TP and FP are overlapped and normalized.

5.2.2 Methods

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 5.1.

Table 5.1: 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 5.2), 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 5.2: Types of classifiers trained on the candidates' dataset

Classifiers
Logistic regression
Decision tree
Random forest
AdaBoost
Gradient boosting

5.3 ResNet based classifier

5.3.1 Methods

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 5.3 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 5.3: 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 5.2, 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 5.3) 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 5.4, 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 5.4: 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.

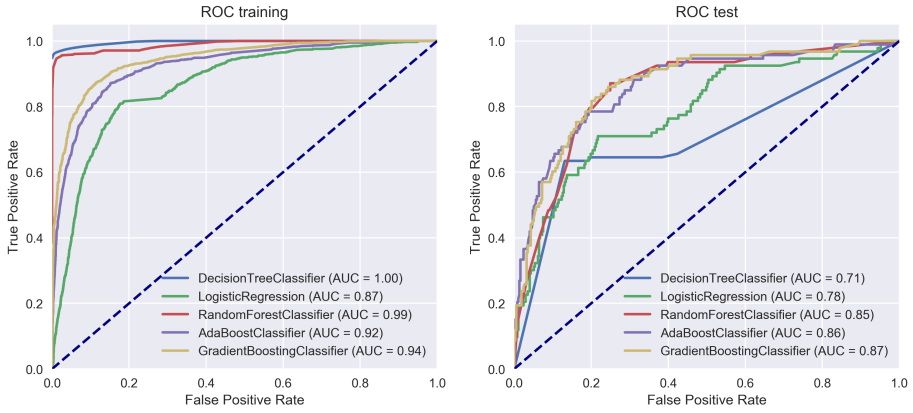

Figure 5.3: ROC curves and AUC of the handpicked feature classifiers

5.4 Results

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

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

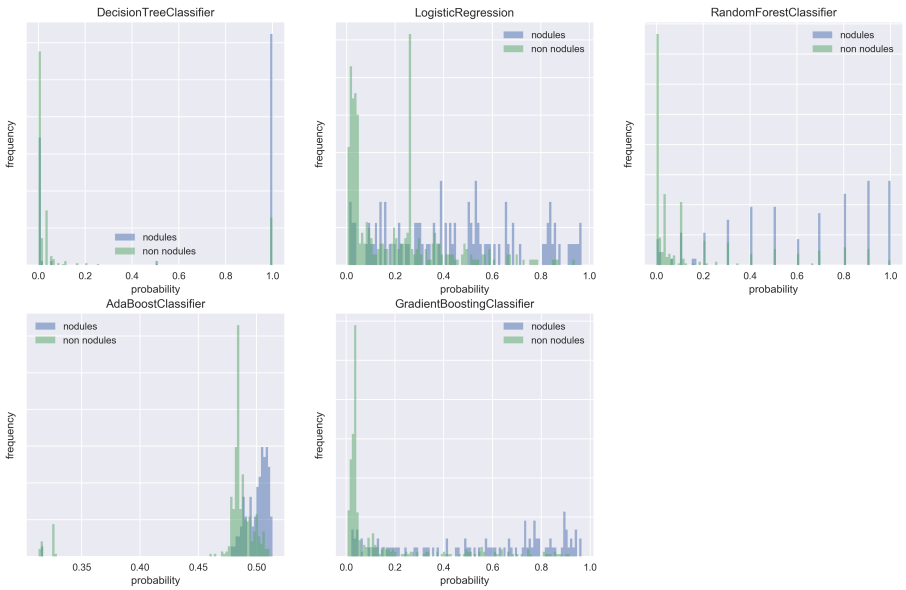

Figure 5.4: histogram pdf handpicked features

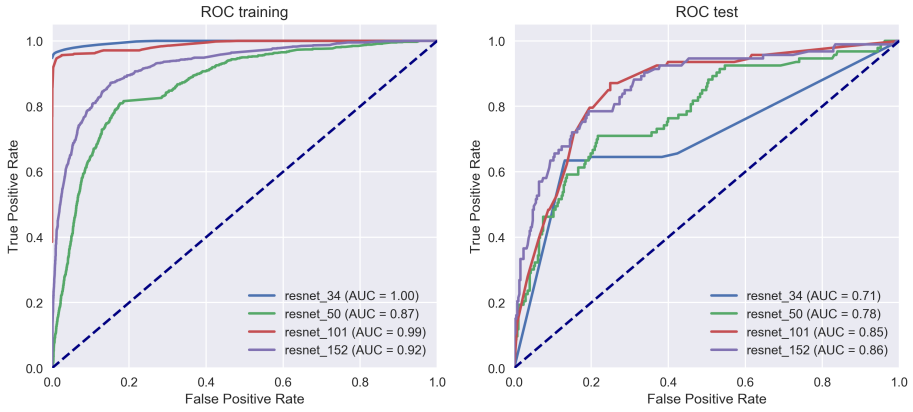

Figure 5.5: ROC curves and AUC of the residual networks

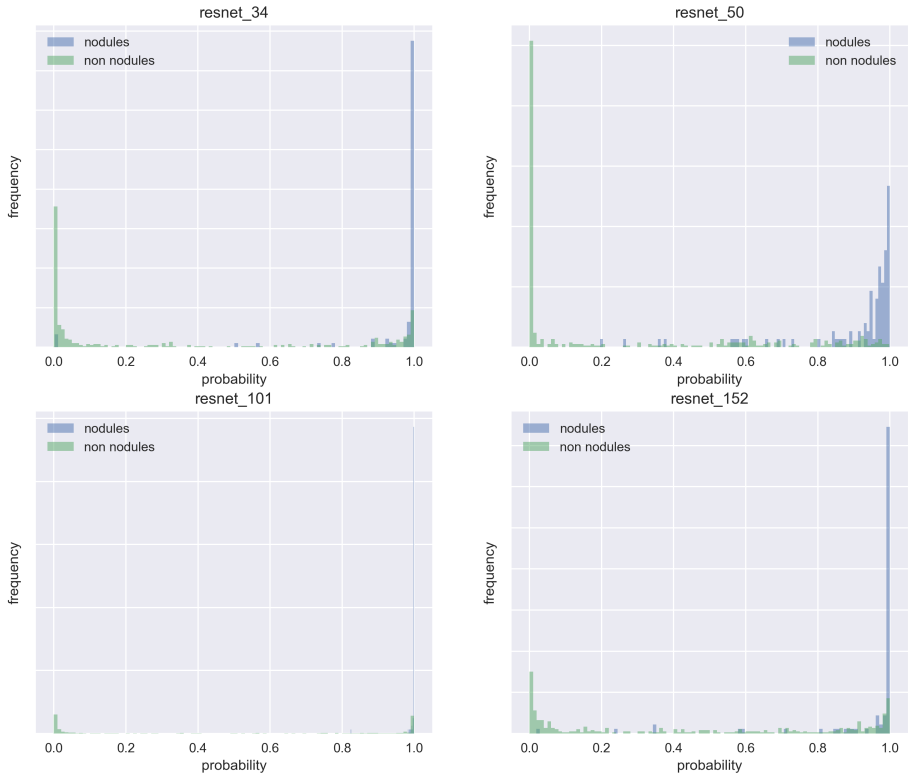

Figure 5.6: histogram pdf residual networks

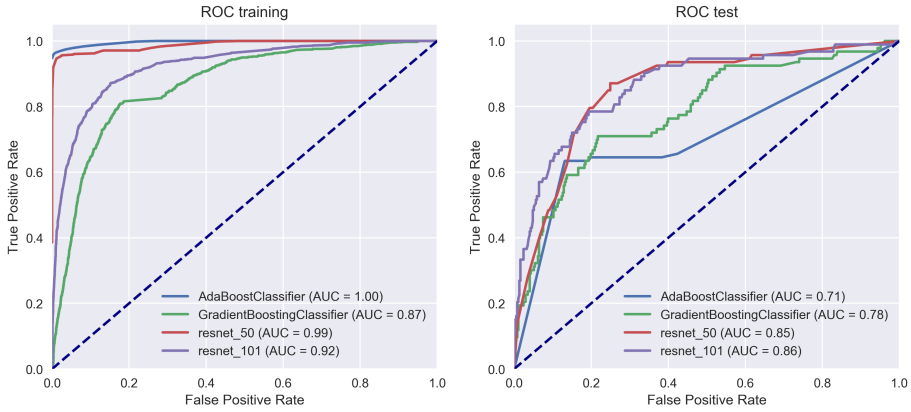

Figure 5.7: ROC curves and AUC comparing the best 2 variations of FP reduction method

probably due to availability of data. The differences don't seem that major, but 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.

5.5 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 6

Integrating the CAD 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.

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

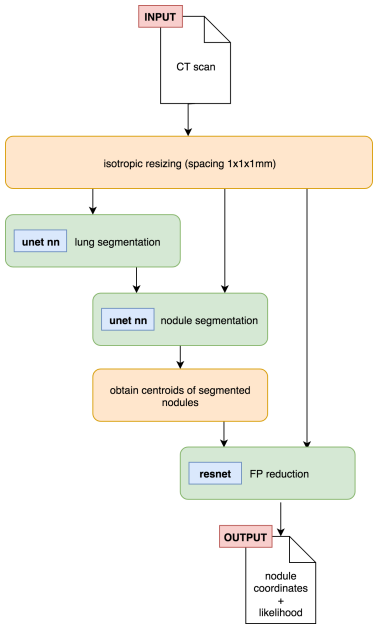

Figure 6.1: the lucanode pipeline

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.

Chapter 7

Evaluating lucanode on LUNA

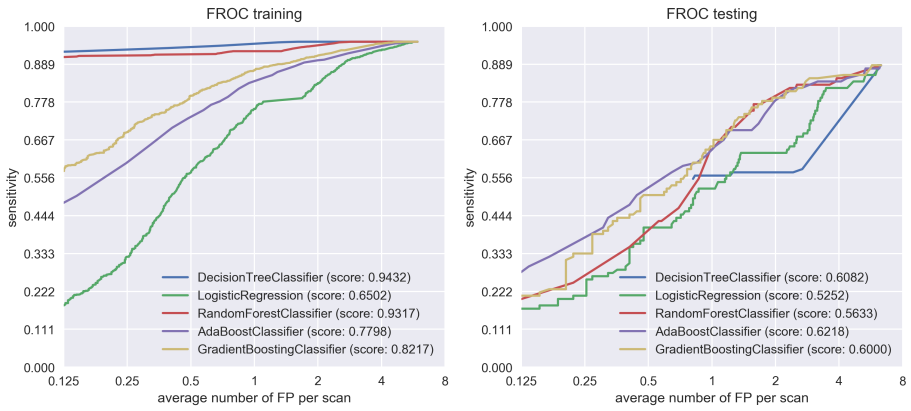

Figure 7.1: FROC curves and averaged sensitivity at selected FP rates of the handpicked features classifier

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 be the best approach to show my perf VS other systems. # Evaluating lucanode on EURECAT

TODO. Or probably not, let's be honest.

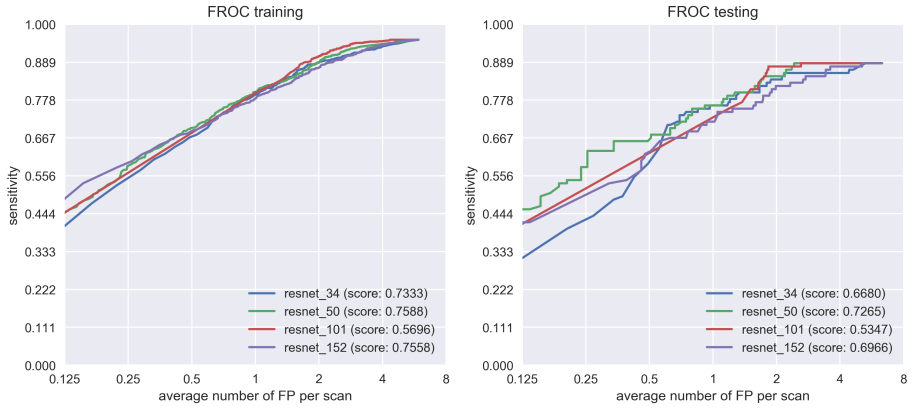

Figure 7.2: FROC curves and averaged sensitivity at selected FP rates of the residual networks

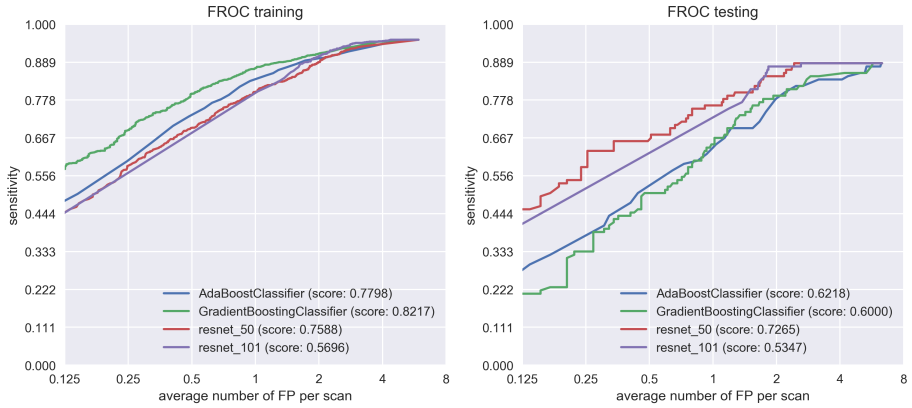

Figure 7.3: FROC curves and averaged sensitivity at selected FP rates comparing the best 2 variations of FP reduction method

Conclusions

TODO

General discussion

TODO

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.

Wu, S., Zhong, S., and Liu, Y. 2017. Deep residual learning for image recognition. 1–17.