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

2.2 The LUNA dataset

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

2.3 Grand challenge competition tracks

Talk about the tracks and metrics

2.4 A survey of the state of the art

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

TODO

Chapter 4

Nodule segmentation

TODO

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 (**add some ref based on Wu et al. [2017] that would properly explain the concept**).

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 come in handy. The intuition here is that we've trained what essentially is a 2D network on a volumetric problem, and after visual inspection of the obtained segmentations, once of the most frequent cases of erroneous segmentation is when the network confuses airways with nodules

put a graph of an axial projection of a nodule mask + actual axial layers AND axial projection of FP air vessel + actual axial layers, so the displacement can be properly viewed

Based on exploting this idea, what I am looking for are spherical and symmetric labels. Basically I want to be as close as possible to the original masks. I come up with the following features:

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.

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.

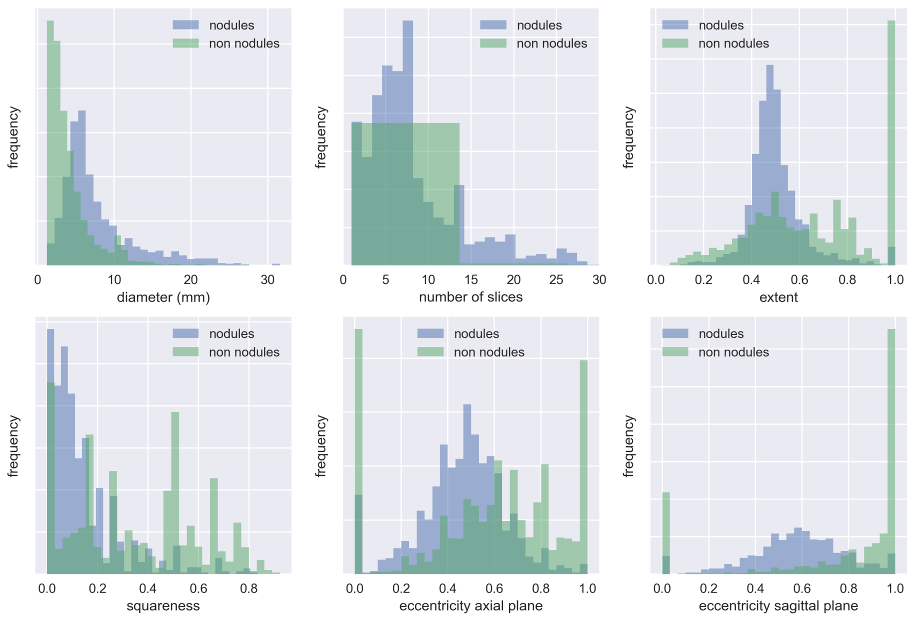


Figure 5.1: frequency distribution of the nodule candidates features, obtained by segmenting the entire LUNA dataset with the *augmented, 3ch, batch normalized, bce unet*. The histograms of TP and FP are overlapped and normalized.

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 Introduction

Why resnet? Set a bit of introduction as why are they good and how they've impacted the image recognition competitions.

5.3.2 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.

As training data we will use the annotations provided by LUNA for the false positive reduction track of the challenge. 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 we'll be able to compare the performance within variations of the same method and between different methods.

Table 5.3:

annotations dataset
something
something
something

Since we will not use an ensemble of multiple networks, we ReLU for nonlinearity batch normalization before

Resnet image input 32x32x32x1

The annotations contain the world coordinates of the candidate centroid and a label indicating whether or not it is a nodule.

Basically here say that in terms of evaluation, same method as before is employed. Preparation of the dataset though is quite different.

candidates_v2.csv is used, which is a list of coordinates with an annotation, whether or not it is a nodule. VERY imbalanced. Then talk about the subsets that I've employed and how I'm using just a window of 32x32x32 to capture the area centered around that point.

Talk about the augmentation procedure in 3D in detail. Explain what I use and what not. Talk about the trigonometry involved so I can actually use the full extent of affine transformations. Also discuss the problems due to memory constraints, and how I opted to use a smaller fraction of the dataset instead of hitting disk (much too slow). Put the back of the envelope numbers on how much memory would be needed to do things properly.

Once the model is trained, say that I evaluate the candidates obtained in the segmentation and output a probability. With that I calculate the ROC and its AUC, and that I can compare directly with the results I've obtained from the previous classifier, which is nice. Using the 1x1x1mm isotropic resize of the scan.

5.4 Results

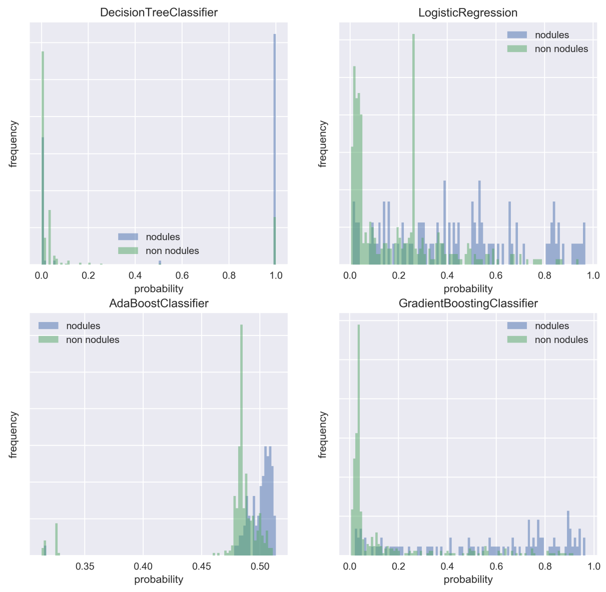


Figure 5.2: histogram pdf handpicked features

It would be nice to have the filters activation patterns for a nodule and a non-nodule. Explain this better plz. I'm using

5.5 Overall Discussion

Put here some comparison between

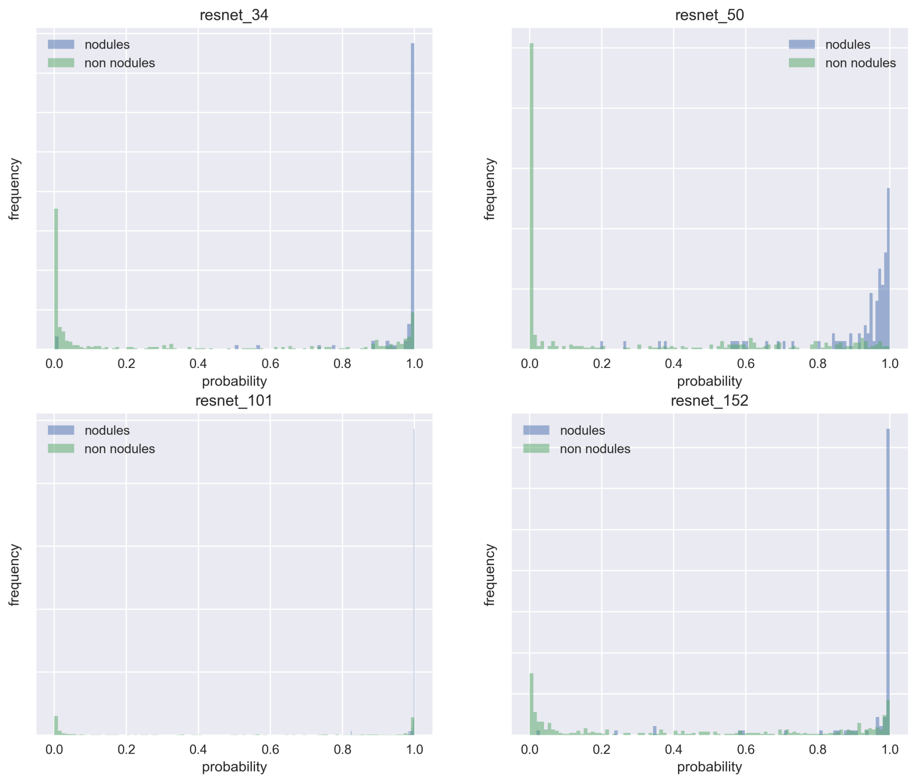


Figure 5.3: histogram pdf residual networks

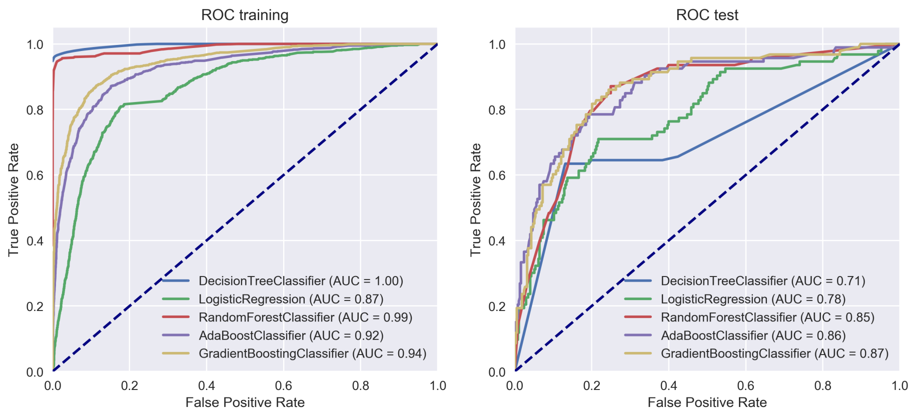


Figure 5.4: ROC curves and AUC of the handpicked feature classifiers

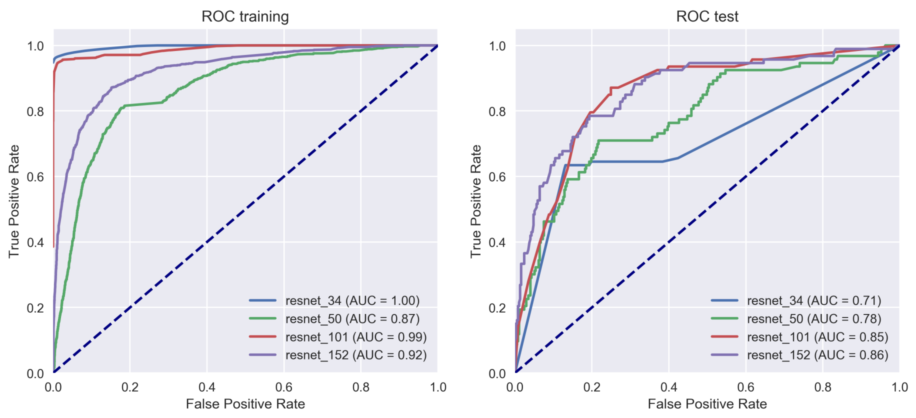


Figure 5.5: ROC curves and AUC of the residual networks

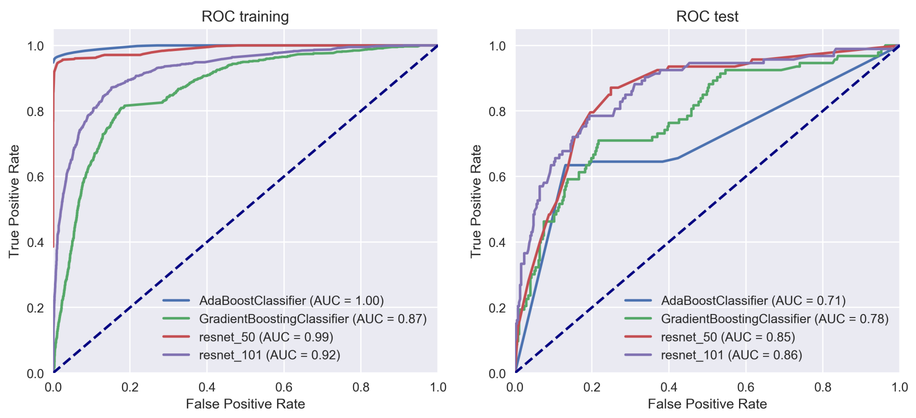


Figure 5.6: ROC curves and AUC comparing the best 2 variations of FP reduction method

Chapter 6

Integrating the CAD pipeline

Cover the necessary steps to package the system as a readymade product. Basically we have to automate the preprocessing steps I've performed when training the network and then I am good to go.

Also, talk about the Dockerized thing and the support for GPU, etc. Basically important to have something easily deployable in production on-site or whatever really.

I'll need some performance numbers with my stuff to basically tell how much time I need to analyze a scan. Also, break down results by step as well. We want to know what is more expensive.

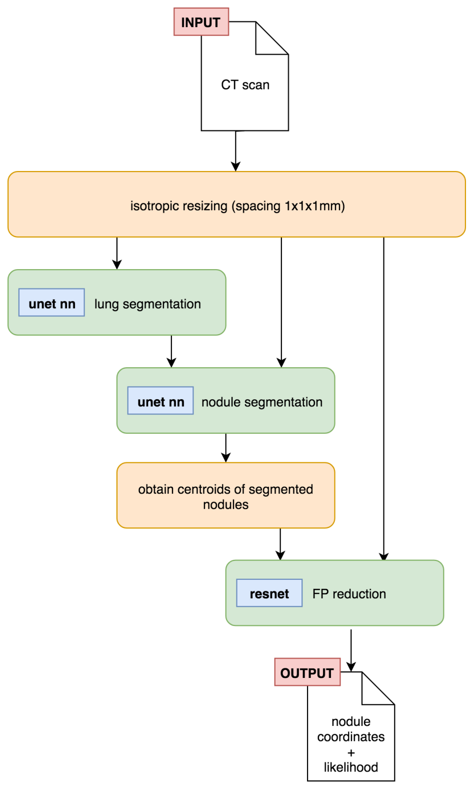

INPUT
CT scan
isotropic resizing (spacing 1x1x1mm)
unet nn
lung segmentation
unet nn
nodule segmentation
obtain centroids of segmented nodules
resnet
FP reduction
OUTPUT
nodule coordinates + likelihood

Figure 6.1: the lucanode pipeline

Chapter 7

Evaluating lucanode on LUNA

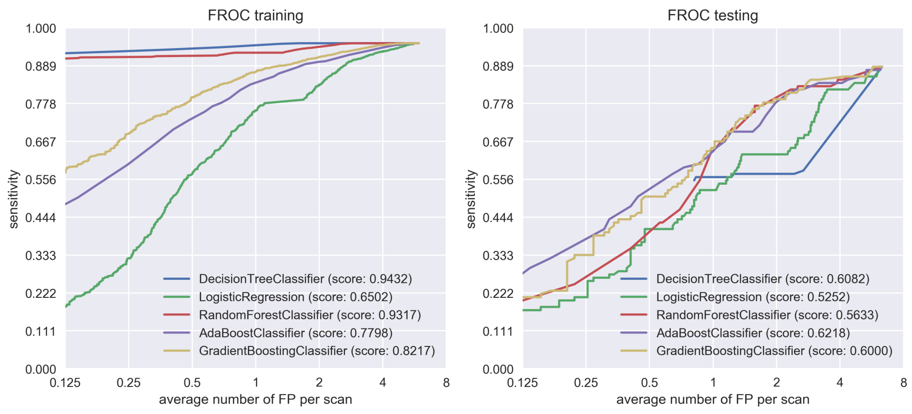


Figure 7.1: FROC curves and averaged sensitivity at selected FP rates of the handpicked features classifier

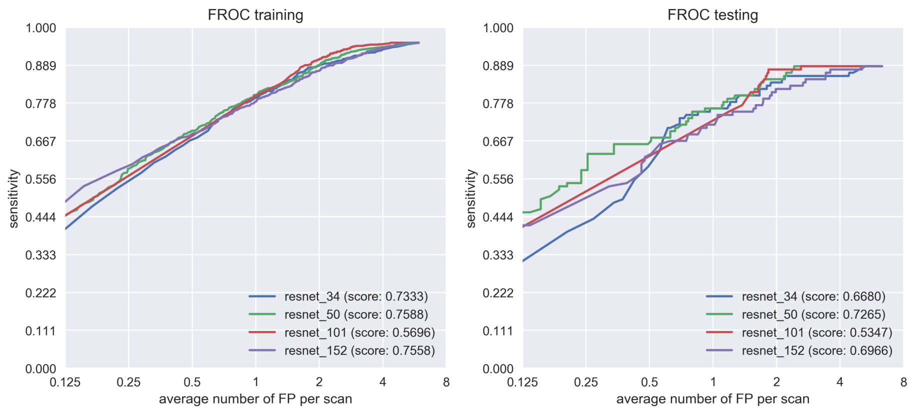


Figure 7.2: FROC curves and averaged sensitivity at selected FP rates of the residual networks

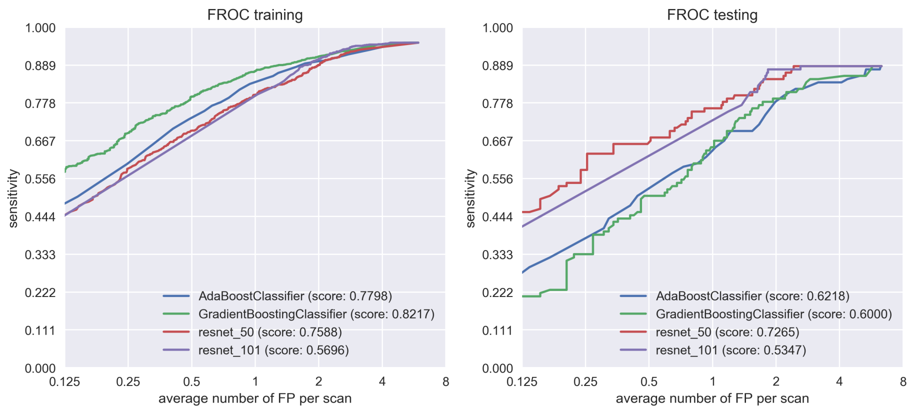


Figure 7.3: FROC curves and averaged sensitivity at selected FP rates comparing the best 2 variations of FP reduction method

Chapter 8

Evaluating lucanode on EURECAT

TODO

Conclusions

TODO

General discussion

TODO

Chapter 9

Placeholder

Basically, examples on how to put Tables, figures, links and citations.

To put an end to this project we will do a performance assessment on LdShake (http://ldshake.upf.edu) (Hernández-Leo et al. [2011]). LdShake is a web application to share and edit learning design solutions mainly tailored at teachers. LdShake is built on top of the PHP Elgg framework (http://elgg.org/), which makes development of social networks easier. It uses a fair amount of javascript, which is mainly used to add interactivity to the webpage, and elements like text editors. Once loaded, though, the pages don't make use of AJAX to modify the content dynamically. The platform is also quite sparse in image resources, so each page weights much less than the typical 1.5MB we saw in the introductory chapter.

Overall, we could say that LdShake is lighter than the typical website, has less resources (around 40 per page instead of a 100) and does not load content dynamically (although the content generation is, indeed, dynamic). Despite these differences, I still believe that LdShake performance analysis can be easily extrapolated to perform similar assessments on other websites.

9.1 Setting up the testing environment

First of all, we have to prepare our testing environment. I have created a VM running Ubuntu Server 12.04 (same OS as production system) and configured following the instructions that appear the ldshake tarball (http://sourceforge.net/projects/ldshake/).

Once the VM is up and running, we will also have to add a traffic shaper to replicate the network conditions our users are most likely to have. To do this, I will use OSX Network Link Conditioner. Network Link Conditioner is an

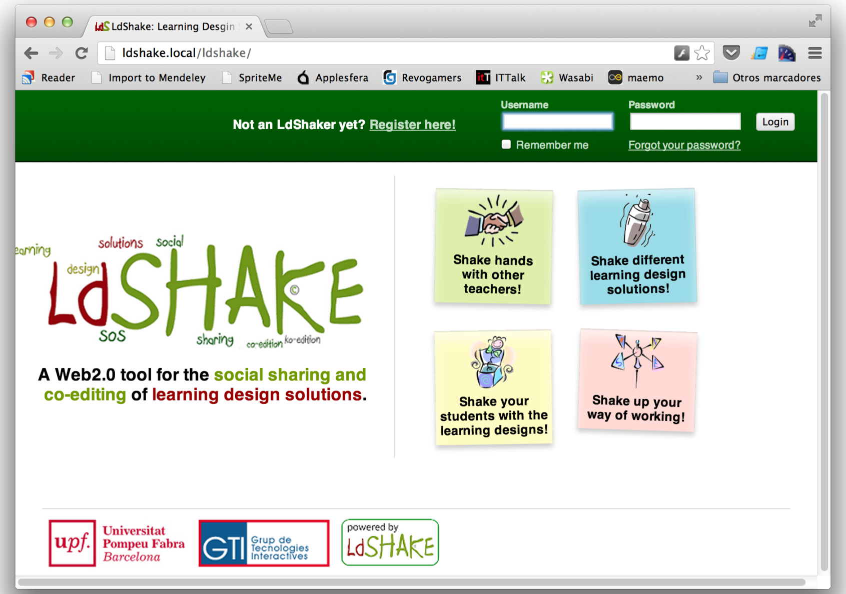
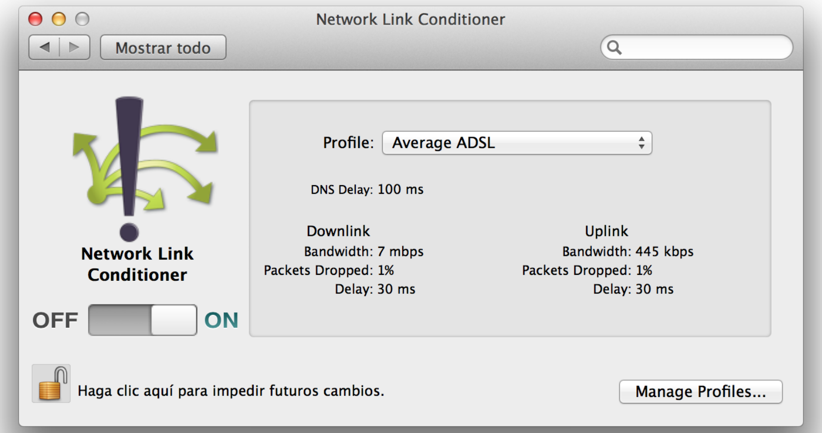

LdShake: Learning Desgin
ldshake.local/ldshake/
Reader
Import to Mendeley
SpriteMe
Applesfera
Revogamers
ITTalk
Wasabi
maemo
Otros marcadores
Not an LdShaker yet? Register here!
Username
Password
Login
Remember me
Forgot your password?
LdSHAKE
A Web2.0 tool for the social sharing and co-editing of learning design solutions.
Shake hands with other teachers!
Shake different learning design solutions!
Shake your students with the learning designs!
Shake up your way of working!
Universitat Pompeu Fabra Barcelona
GTI Grup de Tecnologies Interactives
powered by LdSHAKE

Figure 9.1: LdShake running from the local VM

optional developer tool in Xcode that artificially alters the link capabilities of our networking connections. It can be installed by opening XCode and accessing `XCode > Open Developer Tool > More developer tools...`.



Profile: Average ADSL

DNS Delay: 100 ms

Downlink
Bandwidth: 7 mbps
Packets Dropped: 1%
Delay: 30 ms

Uplink
Bandwidth: 445 kbps
Packets Dropped: 1%
Delay: 30 ms

Haga clic aquí para impedir futuros cambios.

Figure 9.2: Network Link Conditioner with the test settings

Now that we have the traffic shaper installed, we have to set up the network conditions for our tests. To do so, I have looked up a 2012 study on the average Spanish broadband connections (http://www.testdevelocidad.es/estudio-velocidad-2012/). It basically states that a typical user has a *7.1Mbps downlink* and a *435Kbps uplink*. I have also added a *60ms latency*, which is on par with the FCC study numbers on the network layer chapter and a *package loss of 1%*, which is also a typical value in a wired connection.

If we ping the local VM from our traffic shaped machine, we can certify that everything is working as expected:

```
$ ping -c 1 ldshake.local
PING ldshake.local (172.16.246.134): 56 data bytes
64 bytes from 172.16.246.134: icmp_seq=0 ttl=64 time=63.399 ms

--- ldshake.local ping statistics ---
1 packets transmitted, 1 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 63.399/63.399/63.399/0.000 ms
```

9.2 LdShake architectural overview

LdShake uses the MVC pattern. This means that presentation data, like html templates, scripts, stylesheets and images should be kept apart from the business logic, which makes front-end changes easier to deploy without affecting the back-end.

9.3 Initial assessment

Since LdShake uses templates to construct each view and shares a lot of resources between them, we will only do performance assessment on a subset of views, which will be:

- login page
- first steps page
- create ldshake (new content) page
- browse available ldshakes (content) page

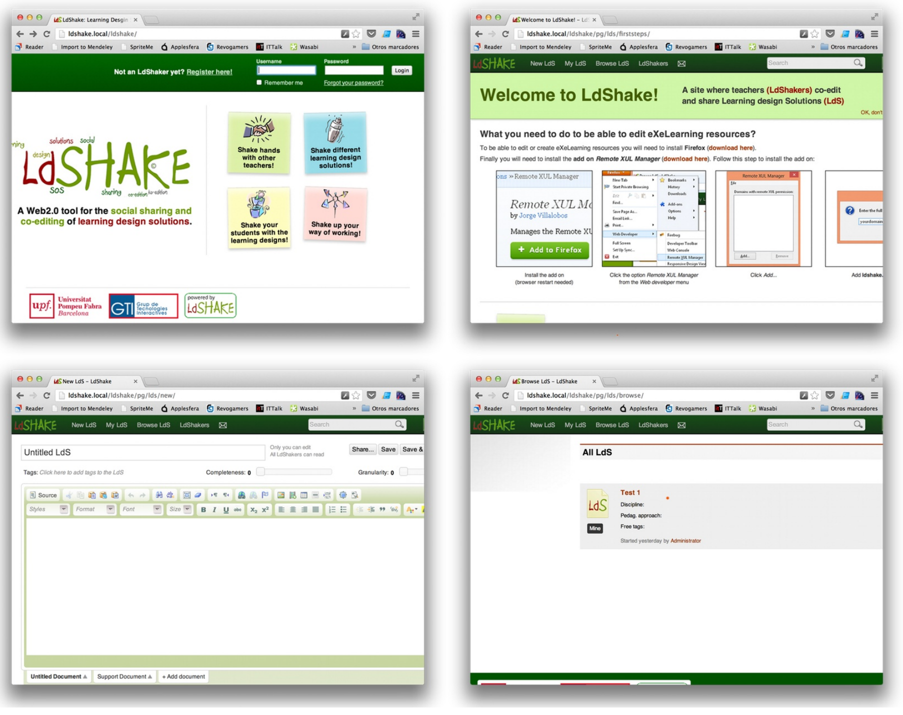

Not an LdShaker yet? Register here!
Username
Password
Login
Remember me
Forgot your password?
A Web2.0 tool for the social sharing and co-editing of learning design solutions.
Shake hands with other teachers!
Shake different learning design solutions!
Shake your students with the learning designs!
Shake up your way of working!
Welcome to LdShake!
A site where teachers (LdShakers) co-edit and share Learning design Solutions (LdS)
What you need to do to be able to edit eXeLearning resources?
Add to Firefox
New LdS
My LdS
Browse LdS
LdShakers
Untitled LdS
Share...
Save
Untitled Document
Support Document
+ Add document
All LdS
Test 1

Figure 9.3: LdShake views to be tested. From left to right: login, first steps, create content, browse content

Using the Chrome developer tools and the PageSpeed Insights extension (https://developers.google.com/speed/pagespeed/insights) we will run an analysis on those pages and detect the problems they may have. The given punctuation is a custom index created by Google that gives us a good idea of how optimized a webpage is (from a front-end point of view). Those indexes were first introduced with the Yahoo UI tools and correlate greatly with decreasing page load times, so they serve as a good measure to quantify the front-end optimization of a webpage.

9.3.1 Login view

Score: 69 / 100

Description: The login view is a very simple html page, with some inlined javascript for the google analytics and some images.

Problems:

1. There are no cache headers. Images don't have any cache information.
2. Usage of redirects. Some images are misnamed (.PNG instead of .png). This triggers a costly `301 Moved Permanently` redirect.
3. Images could use better compression (-28% size).
4. Images could be scaled down (-65% size).

9.3.2 First steps view

Score: 72 / 100

Description: The first steps view is an introductory page to the site, showing some functionality and the necessary extensions to be installed to get a full experience.

Problems:

1. There are no cache headers. Images don't have any cache information. Stylesheets or javascript don't have any cache information either.
2. Usage of redirects. Some images are misnamed (.PNG instead of .png).
3. CSS sprites could be employed.
4. Images could use better compression (-28% size).
5. Images could be scaled down (-65% size).
6. Javascript and CSS are not minified.
7. Some javascript resources are very small and should be inlined.

9.3.3 Browse view

Score: 78 / 100

Description: The browse view displays a list with all the content created by LdShake users.

Problems: Same as previous view.

9.3.4 New LdShake view

Score: 83 / 100

Description: Rich text javascript editor.

Problems: Same as previous view. The mark improves because it has less unoptimized images.

9.4 Performance optimizations

I have performed 2 rounds of optimizations. In the first one I have applied most of the HTTP 1.1 optimizations techniques discussed in the 4th chapter. In the second one, apart from the HTTP 1.1 optimizations, I have enabled SPDYv3 as the application protocol (instead of using HTTP).

To solve most of the issues pointed by the initial assessment I have performed the following changes in LdShake:

- Enable mod_expires in apache.
- Concatenate commonly used javascript files.
- Serve static css instead of dynamic. Delete parameter to allow hashing.
- Minify all javascript and CSS files.
- Optimize images for better compression
- Rename images to avoid `301` redirects.

The default `.htaccess` provided by elgg already configured caching and Etags. By default, files are cached with a 10 year expiration date. That basically forces us to deliver any kind of update with a different filename, but as we will see, caching does wonders to improve the performance of the website. It *is* worth it.

The default template used by all views loaded jquery, jquery-ui and another pair of custom scripts every time. I have joined them and minified it, so this should also help.

The rest of the steps consists on applying UglifyJS to the javascript files of the project and YUICompressor to the CSS. I have also recompressed the website images and added some CSS sprites where they could be applied. To generate the sprites and the modified CSS I have used the spriteme bookmarklet (http://spriteme.org/), created by Steve Souders.

With all this optimizations performed, it is time to see how the new version scores:

Table 9.1: PageSpeed scored after the optimizations

View	Score
Login view	90 / 100
First steps view	96 / 100
Browse view	99 / 100
New LdShake view	98 / 100

In a last attempt to improve performance even further, I have decided to configure `mod_spdy` and test the webapp over SPDY, to see how much of an improvement in brings over HTTP. To install `mod_spdy` on apache, I have downloaded the packages from the project site (https://code.google.com/p/mod-spdy/) and added the following to the apache config file (`/etc/apache2/apache2.conf`):

```
<IfModule spdy_module>
    SpdyEnabled on
    SpdyDebugUseSpdyForNonSslConnections 2
</IfModule>
```

With this, we are basically telling the server to enable SPDY and serve it over unencrypted connections by default. This means that the client must connect using SPDY directly, since the connection won't upgrade from HTTP to SPDY automatically. We can force Chrome into a SPDY only mode by running it with the following parameters:

```
--use-spdy=no-ssl
```

I am aware that SPDY would usually be employed over HTTPS, using ALPN to upgrade the connection to SPDY, but for testing purposes, I wanted to see how SPDY performed raw.

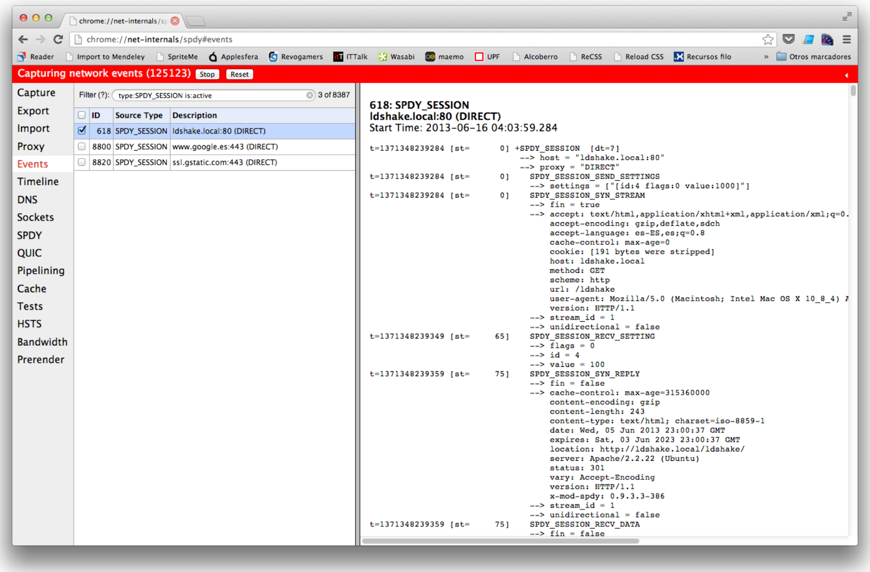
9.5 Measuring

Apart from this performance index given by tools like PageSpeed Insights or YSlow (http://developer.yahoo.com/yslow/), I wanted to have real numbers to quantify the speed increase between versions. In order to do so, I have performed the following experiment:

- For each view:
    - Reload the page 10 times without cache. Record onLoad time.
    - Reload the page 10 times from cache. Record onLoad time.

This test has been done on all 4 views and for all 3 versions of the site:

- Original site. HTTP 1.1
- Optimized site. HTTP 1.1



Filter (?): type:SPDY_SESSION is:active

ID	Source Type	Description
618	SPDY_SESSION	ldshake.local:80 (DIRECT)
8800	SPDY_SESSION	www.google.es:443 (DIRECT)
8820	SPDY_SESSION	ssl.gstatic.com:443 (DIRECT)

618: SPDY_SESSION
ldshake.local:80 (DIRECT)
Start Time: 2013-06-16 04:03:59.284

```
t=1371348239284 [st=     0] +SPDY_SESSION  [dt=?]
                             --> host = "ldshake.local:80"
                             --> proxy = "DIRECT"
t=1371348239284 [st=     0]    SPDY_SESSION_SEND_SETTINGS
                               --> settings = ["[id:4 flags:0 value:1000]"]
t=1371348239284 [st=     0]    SPDY_SESSION_SYN_STREAM
                               --> fin = true
                               --> accept: text/html,application/xhtml+xml,application/xml;q=0.
                                   accept-encoding: gzip,deflate,sdch
                                   accept-language: es-ES,es;q=0.8
                                   cache-control: max-age=0
                                   cookie: [191 bytes were stripped]
                                   host: ldshake.local
                                   method: GET
                                   scheme: http
                                   url: /ldshake
                                   user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_4) A
                                   version: HTTP/1.1
                               --> stream_id = 1
                               --> unidirectional = false
t=1371348239349 [st=    65]    SPDY_SESSION_RECV_SETTING
                               --> flags = 0
                               --> id = 4
                               --> value = 100
t=1371348239359 [st=    75]    SPDY_SESSION_SYN_REPLY
                               --> fin = false
                               --> cache-control: max-age=315360000
                                   content-encoding: gzip
                                   content-length: 243
                                   content-type: text/html; charset=iso-8859-1
                                   date: Wed, 05 Jun 2013 23:00:37 GMT
                                   expires: Sat, 03 Jun 2023 23:00:37 GMT
                                   location: http://ldshake.local/ldshake/
                                   server: Apache/2.2.22 (Ubuntu)
                                   status: 301
                                   vary: Accept-Encoding
                                   version: HTTP/1.1
                                   x-mod-spdy: 0.9.3.3-386
                               --> stream_id = 1
                               --> unidirectional = false
t=1371348239359 [st=    75]    SPDY_SESSION_RECV_DATA
                               --> fin = false
```

Figure 9.4: LdShake on SPDY. Session as captured by Chrome net-internals

- Optimized site. SPDY

I have extracted the average and median onLoad times per case, and I also have recorded the number of requests and transfer size per load. All the data can be accessed from http://goo.gl/HibSN. This url points to a Google Docs spreadsheet with the complete version of the experiment.

Table 9.2: Median page load times per view and optimization. Percentage indicates decrease in PLT against baseline

View	B. F. PLT	B. C. PLT	O. F. PLT	O. C. PLT	SPDY F. PLT	SPDY C. PLT
Login	771ms	606ms	715ms (-7.2%)	282ms (-54%)	498ms (-35%)	262ms(-57%)
First Steps	1205ms	944ms	912ms (-24%)	236ms (-75%)	799ms (-33%)	244ms (-74%)
Browse	1165ms	870ms	958ms (-18%)	228ms (-74%)	755ms (-35%)	240ms (-72%)
New content	2095ms	1100ms	1250ms (-40%)	382ms (-65%)	1495ms (-29%)	417ms (-62%)

We have obtained a substantial improvement in prime page load time in both cases. In this test, since there is more data to download, SPDY can shine with the use of multiplexing. I would expect that bigger websites with many more resources would increase the SPDY lead by a larger margin.

The improvement in cache PLT is spectacular. It really shows how much can cache improve the user experience. In this case, only the html had to be downloaded from the server. Since all the other resources has already been fetched and stored, they loaded instantly. In this case, since the connection only has to download an small HTML file, SPDY does not show any real performance improvement with standard HTTP 1.1. In fact, it performs a little bit worse.

Table 9.4: Number of requests per view.

View	B. Fresh req	B. Cached req	O. Fresh req	O. Cached req
Login	13	13	12	12
First Steps	35	35	27	23
Browse	25	25	22	16
New content	39	39	34	26

From the transfer size and requests tables we can also see the decrease in size and number of requests. This is mainly thanks to the better compression (minification and better image encoding) and also thanks to the spriting of images in the CSS and the concatenation of javascript files. I have not included the transfer sizes for SPDY since the Chrome Dev Tools does not report them. The number of requests for SPDY is the same than those for the optimized HTTP site.

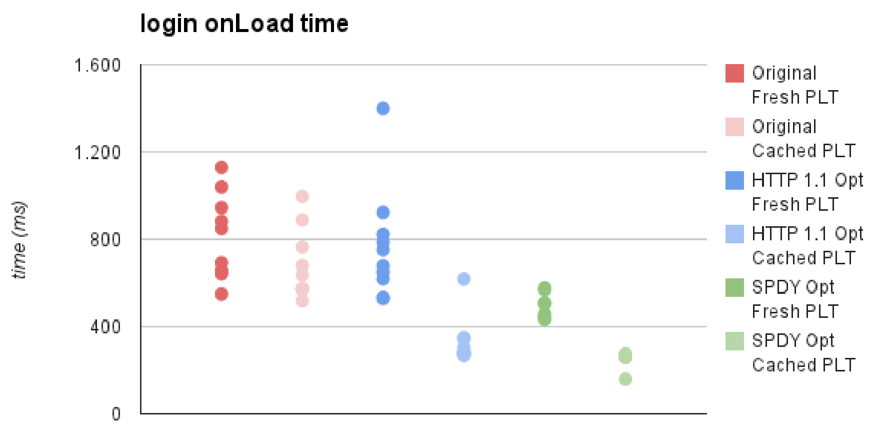


Figure 9.5: Sample variance for the login view test

In the last four figures, we can appreciate the variance of the results in each experiment. As we can see, the original version of each view has higher variances and higher page load times, while the optimized versions all show lower variances and lower page load times, especially the cached view.

The variance can be explained mostly due to our package drop configuration in the traffic shaper (1% out, 1% in). The optimized versions show lower variances since the transferred bytes and requests per view has decreased due to the performed optimizations.

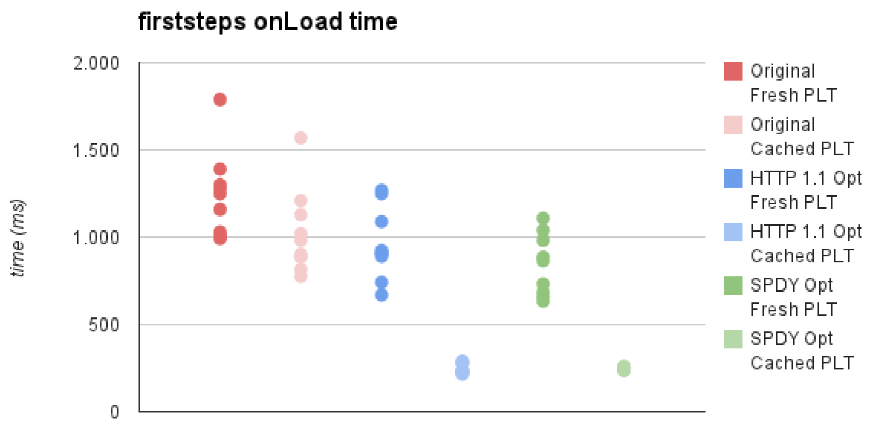


Figure 9.6: Sample variance for the first steps view test

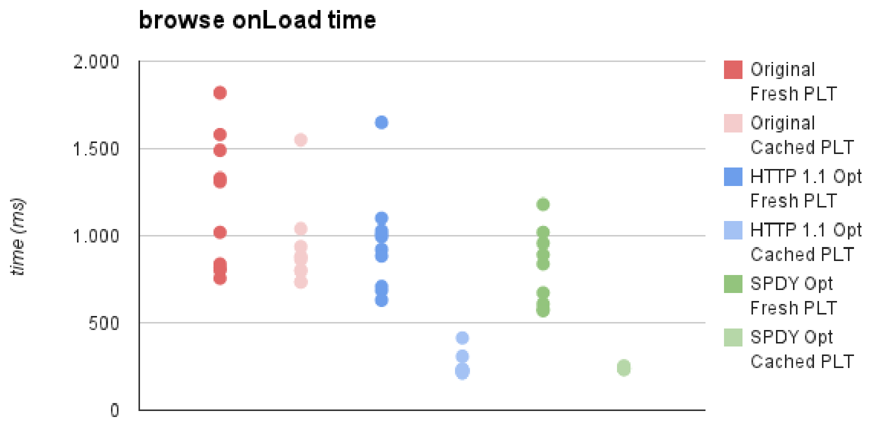


Figure 9.7: Sample variance for the browse view test

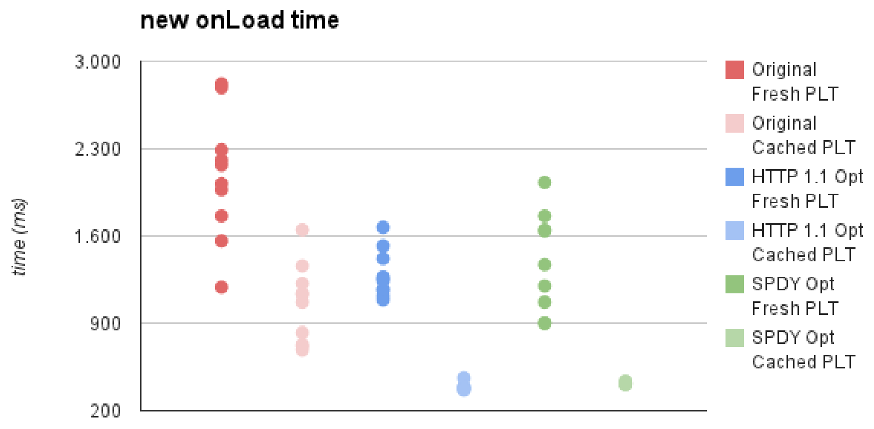


Figure 9.8: Sample variance for the new content view test

9.6 Summary

In this chapter we have finally put to use the attained knowledge in the previous ones. We have seen that it is indeed possible to improve the delivery performance of our websites without changing much of our service, and it has also become apparent that HTTP 2.0 performs better than 1.1, but not noticeably better for lighter pages such as LdShake views. It would be interesting to test HTTP 2.0 on pages filled with multimedia content, like newspapers or flickr. It would also be interesting to make use of its new push capabilities and see how they compare with the typical request-response model.

Bibliography

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..

HERNÁNDEZ-LEO, D., ROMEO, L., CARRALERO, M.A., ET AL. 2011. LdShake: Learning design solutions sharing and co-edition. *Computers & Education 57*, 4, 2249–2260.

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.*.

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.