Plant Recognition: Bringing Deep Learning to iOS
— Final Report —

Cutmore Ashley, Dalyac Alexandre, Douglas Stewart,
Haughian Gerard, Leigh Simon, McCormac John
{ac7513, ad6813, sd3112, gh413, sjl213, bjm113}@ic.ac.uk

Supervisor: Dr William Knottenbelt and Mr Jack Kelly
Course: CO530/533, Imperial College London

May 16, 2014

Abstract

Automatic species recognition of plants in natural scenery is an unmet challenge with considerable user demand from the smartphone app market. LeafSnap, developed by Columbia University's machine vision lab, has attracted 150,000 downloads but has received poor reviews due to its inability to deal with cluttered backgrounds and within-class variance. We propose WhatPlant, an iOS App powered by a cutting edge convolutional neural network to overcome LeafSnap's shortcomings. We achieve a 29% top-5 classification accuracy on 259 plant species, with images of plants taken in natural scenery. This report accompanies the official release of our App on Apple's App Store, and sets out the implementation of our work from beginning to end.

Contents

1 Introduction **4**

2 Specification **5**
2.1 Original Specification . . . 5
2.2 Interpretation . . . 5
2.2.1 iOS 7 Application . . . 6
2.2.2 Plant Classification . . . 6
2.2.3 Server . . . 7
2.3 Changes to Original Specification . . . 7
2.3.1 Added . . . 7
2.3.2 Removed . . . 8
2.4 Revised Schedule . . . 8

3 System Architecture **9**

4 Product Design **10**

5 Methodology **11**
5.1 Division Of Work . . . 11
5.2 Agile Development . . . 11
5.2.1 Anatomy of the Sprints . . . 11
5.3 Source Control . . . 12
5.4 Release Management . . . 12
5.5 Testing . . . 14
5.5.1 iOS 7 Application . . . 14
5.5.2 Servers . . . 15
5.5.3 Plant Classification . . . 16
5.5.4 Overall Coverage . . . 16

6 Implementation **17**
6.1 iOS 7 Application . . . 17
6.2 Servers . . . 17
6.2.1 Request Server . . . 17
6.2.2 Worker Server . . . 18
6.3 Plant Classification . . . 19

7 Final Product **21**
7.1 Product Evaluation . . . 21
7.2 Comparative Performance . . . 22
7.3 Further Development . . . 22
7.4 Future Opportunities . . . 23
7.5 Project Evaluation . . . 23

A Code Coverage Reports **25**
A.1 Plant Classification Coverage Report . . . 25
A.2 Request Server Coverage Report . . . 25
A.3 Worker Server Coverage Report . . . 25
A.4 iOS Application Unit Tests . . . 26

B Logbook **27**
B.1 Detailed Work Breakdown . . . 27
B.2 Minutes of Group Meetings . . . 34
B.2.1 13/12/2013 13:00 . . . 34
B.2.2 16/01/2014 12:00 . . . 36
B.2.3 16/01/2014 13:00 . . . 38
B.2.4 17/01/2014 18:00 . . . 40
B.2.5 22/01/2014 14:00 . . . 42
B.2.6 27/01/2014 15:00 . . . 44
B.2.7 27/01/2014 16:30 . . . 46
B.2.8 29/01/2014 14:00 . . . 48

B.2.9 30/01/2014 15:00 50
B.2.10 31/01/2014 10:00 52
B.2.11 17/02/2014 13:00 53
B.2.12 20/02/2014 15:00 54
B.2.13 26/02/2014 14:00 55
B.2.14 10/03/2014 13:00 56
B.2.15 12/03/2014 14:00 58
B.2.16 25/03/2014 17:00 59
B.2.17 12/05/2014 10:00 60
B.2.18 13/05/2014 10:00 61
B.3 Git Commit Logs 62

1 Introduction

Accurate and portable plant identification has many motivations: the pragmatic farmer identifying a weed, families in the park curious about their natural surroundings, conservationists keen to gather accurate botanical data, citizen science projects for the control of invasive species. For a human to visually identify plants requires years of specialised training, but recent advances in computer vision and artificial intelligence make it possible to take a machine through comparable training in a matter of days. This opens the possibility of enabling anybody with a smartphone to identify plants with a reasonable degree of accuracy, and allow such data collection and analysis to be available to non-experts en masse.

This report describes the creation of an iOS app for recognising plant species based on images taken in natural scenery. The project is divided into three components: an iOS app front-end, a server system to enable communication between the app and our classifier, and a Convolutional Neural Network to classify the images. This report takes the reader through the following sections, each of which branch off into the three project components:

- **Specification** : The original outline of the project, our interpretation of it, and the specific tasks we set for ourselves.
- **System Architecture** : An overview describing how the three components of our product fit together and interact.
- **Product Design** : The look and feel of our product to the user.
- **Methodology** : Our production development strategy, including unit testing.
- **Implementation** : The biggest challenges we encountered and how we dealt with them.
- **Final Product** : Evaluation of the performance of our product, and the commercial opportunities it presents.

Overall, the project presented ambitious goals - since, to our knowledge, no app for recognising plants in natural scenery exists - but the goals were met: our app can classify 259 plant species with 29% top-5 accuracy, and is currently being downloaded and used by iOS users on Apple's App store.

2 Specification

2.1 Original Specification

The original proposal stated that "The ultimate aim is to produce a smart phone application which can automatically identify a plant from a photo". There already exists an iOS app which promises to do this [10] - but only if supplied with a photo of a leaf against a white background. Hence, our key objective became the production of an improved alternative that addresses the weaknesses of the existing solution.

The original proposal considered three areas for development: the smartphone application, the server and the classification engine. Two different directions were suggested for the classification engine: manually writing feature detectors, or automatic detection of features (i.e. machine learning). Possible features and implementation details were proposed such as uploading a group of images per classification, linking results to 3rd party reference sites e.g Wikipedia and server job queue management.

2.2 Interpretation

With a very broad and open specification the first item on our agenda was to give the project a focused direction that we felt minimised compromise, was achievable and got us excited. Group meetings, discussions with our project supervisors and a system of Goal-Orientated Capture [Figure 1] allowed us to come to a mutual understanding of the specific goals of the project.

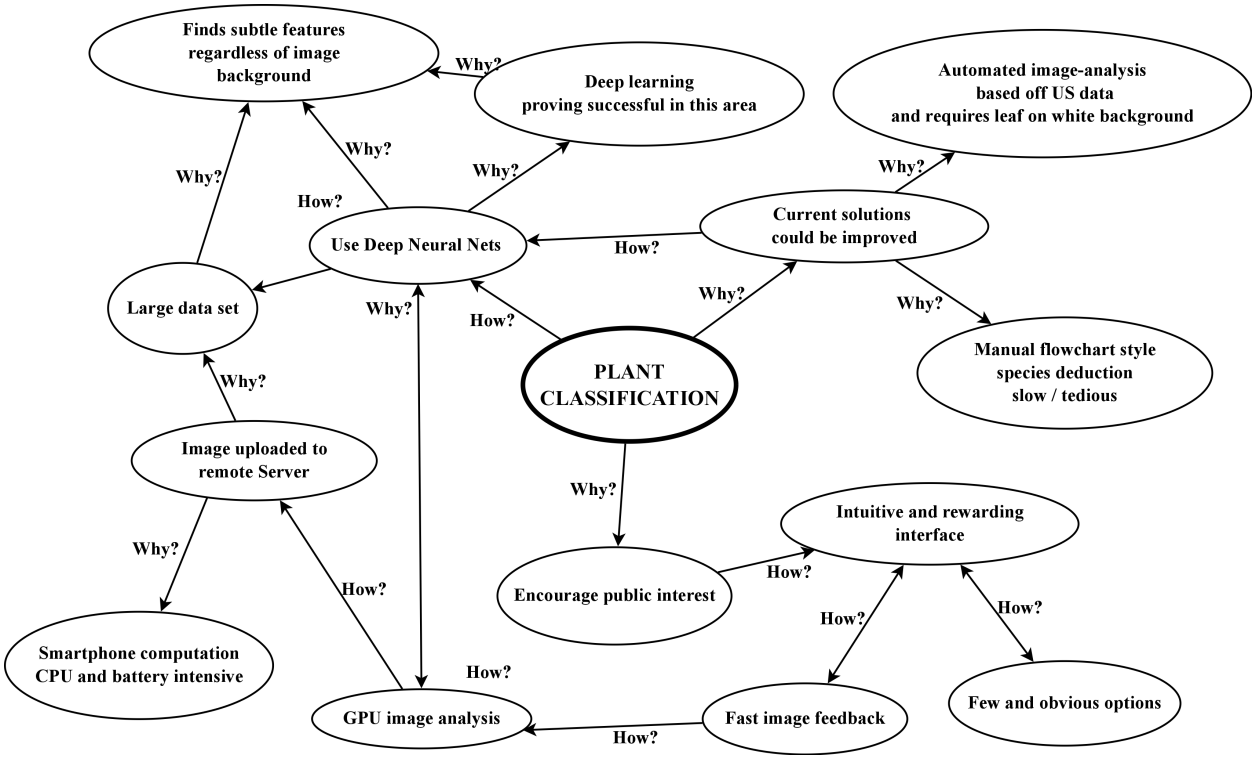


Figure 1: Goal-Orientated Capture

Central to this project is software enabling accurate and automatic identification of a plant from a single or multiple images of the plant (leaf, flower, fruit, or the entire specimen) in its natural environment. Ideally, the process of identification should not require manipulation of the plant or its surroundings and should also only take a short period of time (on the order of seconds). To achieve this we took a deep learning approach using Alex Krizhevsky's cuda-convnet which utilises the GPU available to us during the project.

The smartphone app was developed for iOS7 and designed to be intuitive to use: leading a user naturally from taking a photo to an understandable result. The server maintains a database of jobs and results, and communicates with the app using HTTP.

To help the team structure the project we translated the above goals into sets of deliverables for each of the project's three main areas: iOS Application, Plant Classification, and Server. We have differentiated these deliverables across two dimensions. Firstly, we assigned them to either

an Essential or Supplementary category. Essential deliverables were seen as key to the project's success, whereas Supplementary deliverables typically represent additional features that could be added. Secondly, each deliverable is given a High, Medium or Low priority to help the team plan the project's time-line. Each deliverable and its corresponding category and priority is listed below. Deliverables that were were added after the original goal capture meeting are highlighted.(with a Category '+'). In the tables below, we show the final state of each deliverable and the estimated completion date (ECD).

2.2.1 iOS 7 Application

Cat	Description	Priority	ECD	Status
E	**iOS 7 application that can take and store photos.**	H	Sprint1	Complete
E	**Application can upload photos to server.**	H	Sprint1	Complete
E	**Application can receive and handle classification result from server.**	H	Sprint1	Complete
E	**Classification result can be displayed to user.**	H	Sprint1	Complete
S	*Photo geographical location stored.*	L	Sprint1	Complete
S	*HTML5 web interface available for desktop and non-iOS mobile.*	M	Sprint3	Removed
S	*Link to web (e.g. Wikipedia) entry for species.*	M	Sprint3	Complete
S	*Application released on Apple's App Store.*	M	Sprint3	Complete
S	*Comparison image displayed.*	M	Sprint3	Removed
S	*User feedback of result returned to server.*	L	Sprint3	Removed
+	**User can select one of four plant components (e.g. Leaf, Fruit, Flower) to improve result accuracy.**	-	Sprint3	Complete

2.2.2 Plant Classification

Cat	Description	Priority	ECD	Status
E	**A neural network that processes an image and returns a classification.**	H	Sprint1	Complete
E	**Neural network capable of handling required image resolution.**	H	Sprint1	Complete
E	**Determine image resolution that produces a relevant result.**	H	Sprint1	Complete
E	**Interface integrated with the rest of the system.**	H	Sprint1	Complete
E	**Result produced within an acceptable time interval.**	M	Sprint2	Complete
E	**Acceptable classification time for users determined.**	M	Sprint2	Complete
S	*Image manipulation scripts to augment training set.*	M	Sprint2	Complete
S	*Neural network capable of processing multiple images for a single plant.*	M	Sprint3	Complete
+	**A neural net to retag ImageNet data by plant component type.**	-	Sprint2	Complete
+	**A training database with taxonomical bucketing algorithm.**	-	Sprint2	Complete

2.2.3 Server

Cat	Description	Priority	ECD	Status
E	**Define RESTful interface allowing image and image meta-data to be received from iOS 7 application and response delivered from server.**	H	Sprint1	Complete
E	**Define interface between application-facing server and plant classification software.**	H	Sprint1	Complete
E	**Classification requests forwarded to our neural network for processing.**	H	Sprint1	Complete
E	**Classification results received back from the neural network.**	H	Sprint1	Complete
E	**Plant classifications stored in database.**	H	Sprint1	Complete
E	**Classification passed back to user/iOS 7 application.**	H	Sprint1	Complete
E	**Plant images and appropriate meta-data stored in database.**	M	Sprint1	Complete
E	**Basic queue for requests implemented.**	M	Sprint1	Complete
S	*Images hashed and stored in database.*	L	Sprint2	Removed
S	*Requests can be sent to multiple neural networks.*	L	Sprint3	Removed
S	*Open API provided for 3rd parties to interact with our backend.*	L	Sprint3	Removed
S	*Hashed image used to check previous uploads of same image.*	L	Sprint3	Removed
S	*Duplicate images not stored or re-processed.*	L	Sprint3	Removed
S	*Requests to connect to server authenticated.*	L	Sprint3	Removed
S	*Stored classification results returned on hash collision.*	L	Sprint3	Removed
S	*Result accuracy tracked over classification versions.*	L	Sprint3	Removed
+	**Separate 'Worker' server executes combine script when all images of plant received.**	-	Sprint3	Complete

2.3 Changes to Original Specification

2.3.1 Added

- **User can select one of four plant components (e.g. Leaf, Fruit, Flower) to improve result accuracy.**

  Given the disparate nature of features between images (consider a flower vs. a leaf), we felt that training separate smaller neural networks on each of the components could improve classification performance. This motivated us to allow the user to capture four different components of each specimen.

- **A neural network to retag ImageNet data by plant component type.**

  The ImageNet data we had was tagged by species, but not by 'component type' (i.e. Leaf, Fruit, Flower, Entire). We overcame this issue by using an alternate dataset (provided by PlantClef [2], which did have these tags) to train a separate neural network to tag each of our two million images from ImageNet, before inserting it into our training database.

- **A training database with taxonomical bucketing algorithm.**

  The motivation for this bucketing algorithm was to ensure our neural network wasn't burdened having to learn to recognise species for which we had a limited training subset (i.e. few images of an individual species). Bucketing those images into a higher level in our taxonomy tree meant we could decrease our neural network error rate while maximising our use of available training data. We wrote a bucketing algorithm which could dynamically bucket plant species into buckets which contained a minimum threshold of images which could be classified based on a taxonomy tree we constructed using WordNet [15].

- **Separate 'Worker' server executes combine script when all images of plant received.**

  To ensure modularity and scalability of our architecture, it was decided early in Sprint 1 to create independent 'Request' and 'Worker' servers. These servers run on distributed machines and both connect to the same database.

2.3.2 Removed

- **User feedback of result returned to server.**
  It was noted in our meeting with the Natural History Museum that the general public often misclassified plants, and that even experts regularly misclassified. Thus, we removed crowd-sourced classifications from the iOS application because it was felt that it cluttered the design and would not add to our classification accuracy.
- **Image Hashing and Duplicate Image Detection**
  The changes to the iOS application made hashing uploaded images unnecessary. The application does not allow users to upload the same image twice.
- **HTML5 Interface and API** These were omitted as they were non-essential and it was felt that development time would be better spent on the other deliverables.

2.4 Revised Schedule

There was one major, unforeseen issue in Sprint 1 which caused a significant impact on our original proposed schedule. This also affected later Sprints: the main source for our neural network training data, ImageNet, experienced technical difficulties with their web site; causing it to be inaccessible for a full week. Once the site was available again, we then were further delayed for five full days downloading a 1.2TB tar file of images. This too impacted our ability to progress with many Plant Classification tasks for Sprint 1. All other tasks proceeded as scheduled, taking into account revised goals as iterations of the product were developed.

3 System Architecture

The specification presented two essential features for our architecture: an iOS app and a 'Worker' server capable of classifying images sent to it. In particular, the Worker server needed to have a powerful GPU to support NVIDIA's CUDA platform [9]. To help us complete our project, the Computing Support Group provided us with a dedicated lab machine with a powerful GPU.

A number of different configurations were discussed initially, including having the iOS app communicate directly with the Worker server, which itself would be linked to a data store. However, this early proposal was rejected as it could not scale effectively. Instead, the final architecture includes a 'Request' server, which acts as an intermediary between the iOS app and the Worker server. If the app becomes popular then this would allow us to distribute the image classification tasks across a number of different Worker servers.

The Request server runs on a virtual machine and communicates with the iOS app using HTTP GET, POST and PUT requests. To store the metadata associated with each image we use the MongoDB NoSQL document database, because it can scale rapidly and executes reads and writes quickly. Furthermore, our data is not strongly 'relational', meaning an SQL solution was not necessary. The MongoDB database runs on the Request server but is accessible from the Worker server.

Upon receiving an image from the iOS application, the Request server modifies our MongoDB collections and forwards the image to the Worker server. Concurrently, the Worker server is polling the database for unclassified images, which it then runs through our neural network. Finally, when all the images of a certain plant have been analysed, the results are aggregated using our Bayesian Classifier. The iOS application is then responsible for requesting results.

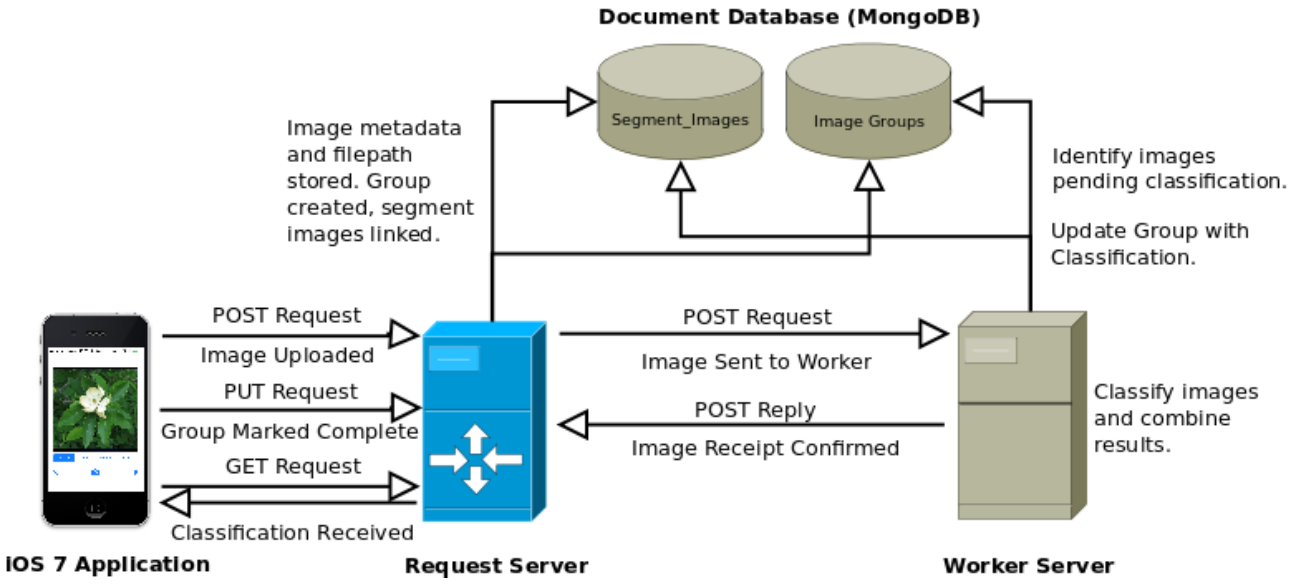


Figure 2: System Context Diagram

4 Product Design

The design of the product will determine the user's experience of using our software long before a result is shown. A prototype application was built during the first sprint that could take images, upload these to the server and receive and display a result. This was made possible by the organisation of 'protocol meetings', to specify the interface between the application and the server. In an extensive UI design session during sprint 2 we prototyped different interface work-flows, sketching ideas onto paper wire-frames. As we planned to release on the App Store, our designs had to stay aligned with Apple's detailed guidelines. The user interface was simplified to minimise the options available to the user and we reduced each classification session to three views: Home, Capture and Review. An initial version of the updated design was completed by the end of Sprint 3. In this version, photos can be taken and reviewed for different plant components before being saved and uploaded in the background during one continuous camera session.

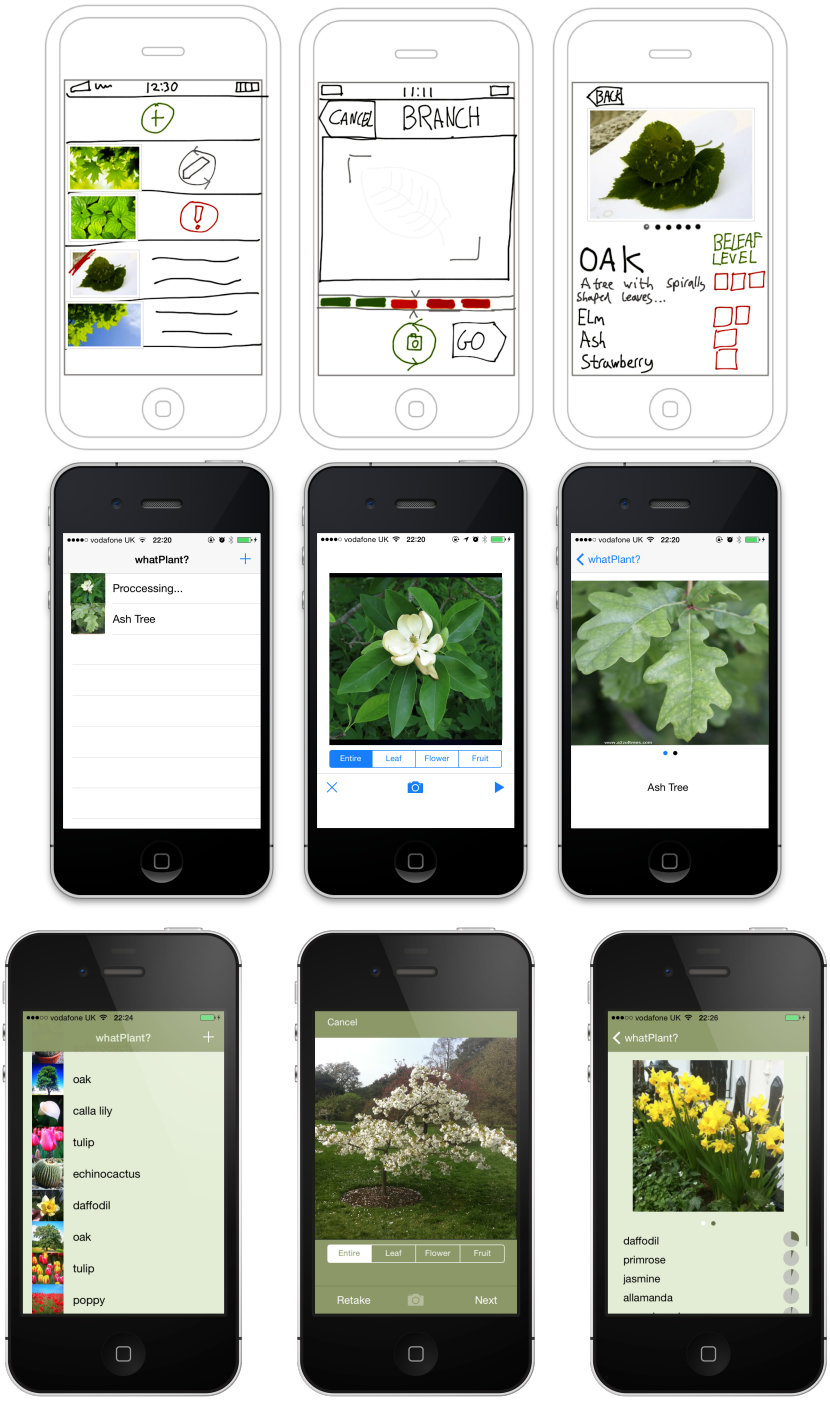


Figure 3: Evolution of the App

5 Methodology

5.1 Division Of Work

We balanced the existing experience and skill set of each team member with their desired learning outcomes in order to best allocate resources and ensure timely delivery of each component of the specification. Due to his prior experience in software project management, Gerard Haughian was nominated as Group Leader and Scrum Master and accepted responsibility for Code Integration and Release Management.

The table below shows the initial allocation of individuals to areas:

	Leader	Doc Editor	Servers	Frontend	Machine Learning
Alexandre Dalyac	-	-	-	-	X
Ashley Cutmore	-	-	-	X	-
Gerard Haughian	X	-	-	-	X
John McCormac	-	-	-	-	X
Simon Leigh	-	X	X	-	-
Stewart Douglas	-	-	X	-	-

5.2 Agile Development

The project management methodology of choice for this project was Agile software development with Scrum. Some group members have previous industry experience using Scrum and recognise the key benefits of using it and its popularity; often being the methodology of choice in the industry. Agile was favoured over other methodologies such as Waterfall due to the ability to incrementally produce a stable working product at the end of each sprint. This is beneficial as having something tangible to demo to stakeholders enabled the team to implement any feedback straight back into the product, rapidly respond to issues, and incrementally improve upon the product. We aimed to maximise code coverage by unit testing and using a sophisticated release management strategy that automated the majority of our unit and integration testing. This is explained in greater detail in section 5.4.

5.2.1 Anatomy of the Sprints

The sprints were two weeks in duration. This length was favoured due to the complexity of the system and the number of components that needed to be developed. We also recognised that other academic commitments needed to be taken into account. The team completed three full sprints and one half sized sprint.

Backlog The backlog was central to the project. The requirements were written up as user stories with associated acceptance tests. The backlog was updated before each sprint and frozen during sprints. It was contained within the project management tool VersionOne [17]. This tool enabled sprint planning and progress tracking of the project over time.

Each user story was assigned a high-level estimate of the number of hours required to complete that story. We used buckets of 1, 2, 4, and 8 hours as the scale of choice for these estimates. The backlog was prioritised in order of importance of that feature making it into the final product.

Scrum Board VersionOne offered the necessary functionality to act as our online scrum board. Having one piece of software manage as many aspects of our project as possible was desired over using multiple applications. Having a central application ensured that we had accurate and up to date information on the progress of our project and the tasks each user was responsible for.

Daily Scrum During weekdays the team met at a designated time for daily scrums. These where between 5-10 minutes in duration and run by our Scrum Master, Gerard Haughian. Traditionally all members of the team would be physically present for this stand-up however if a team member was working remotely they would use Google Hangout to join the scrum.

Sprint Procedures Each sprint consisted of five stages:

- **Backlog Review:** Prioritise the backlog and select user stories that will form the sprint backlog. Add selected user stories to a new sprint on VersionOne.

- **Estimations and Committals:** Prioritise the sprint backlog and assign estimations to the user stories.
- **Tasking Up:** Break down user stories into tasks and put those tasks on VersionOne assigning team members to work on them.
- **Product Demo:** At the end of the sprint, demo the product to Mr. Kelly and Dr. Knottenbelt.
- **Retrospective:** As sprints were completed, we discussed what went well and identified areas of improvement for the next sprint. We quickly reviewed overall progress on the product backlog to ensure we where still on schedule to complete the project.

5.3 Source Control

We used git as our version control software of choice for a number of reasons: decentralised local actions that are extremely fast; almost everything committed is recoverable; cheap and easy branching encourages frequent use. Additionally, the release management strategy below highlights ways in which we attempt to alleviate one issue with git's decentralised system which could potentially lead to state in which there is no sense of a latest version of our code base. Git integrates well with GitLab which all members of the team have experience with. GitLab also enables setting up web hooks which we can utilise to trigger automated unit tests.

5.4 Release Management

A simple, well defined release management process was put into place that ensured we always maintained a stable, well tested and structured version of our code base. This guaranteed we always had a deployable product from early in the project and that development work completed during sprints did not affect our stable code base.

We had tiered release management strategy which consists of having three git branches, namely: DEV, QA and Master. When a team member began working on a task, they first checkout the DEV branch, then create a new branch from DEV named appropriately for the task they are working on. As each developer worked on completing a task, they could check code in and out of their task branch without affecting the main DEV branch. Once tasks were completed the code was merged and pushed into DEV, triggering automated unit tests.

This approach encouraged any new code checked into the DEV branch to have appropriate unit tests associated with it. This strategy assisted in detecting issues early and ensured all components of the system remained in a compatible state. Once all tasks required for a sprint where completed, the release manager pushed and merged all changes into the QA branch for Unit, System and Integration testing. At the end of each sprint, all new code was pushed and merged with the Master branch by the release manager. The Master branch always reflected a stable, well tested and incremental prototype of our system.

Both the QA and Master branches where protected so that only the release manager could push code to those branches. It is expected that a few days prior to a sprints due date, all code is checked into QA for adequate testing before the release date.

Figure 4: Project Velocity Tracking

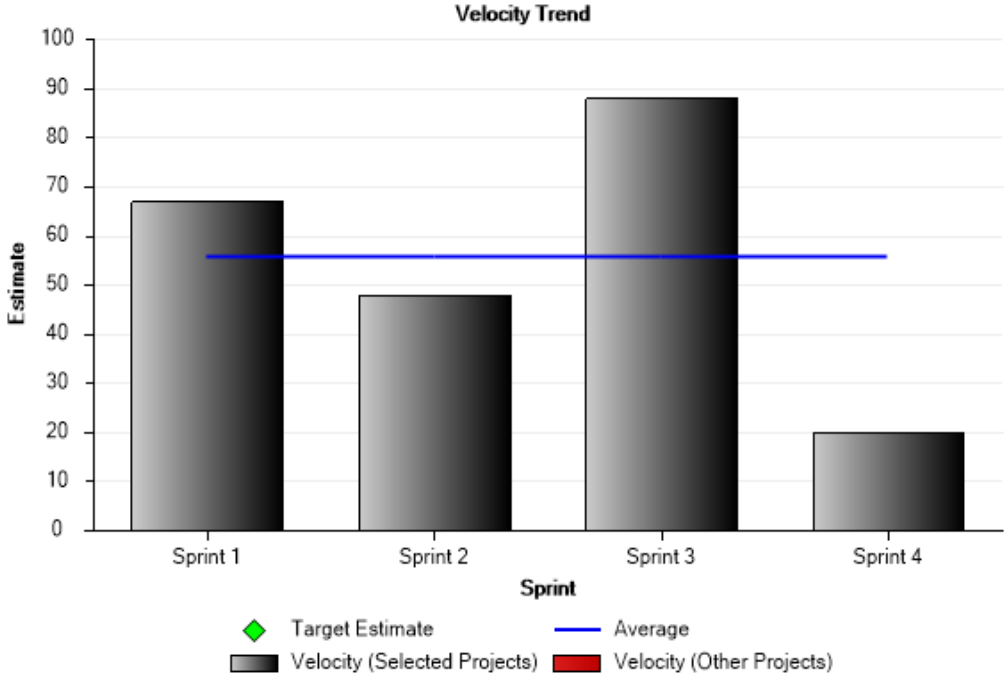
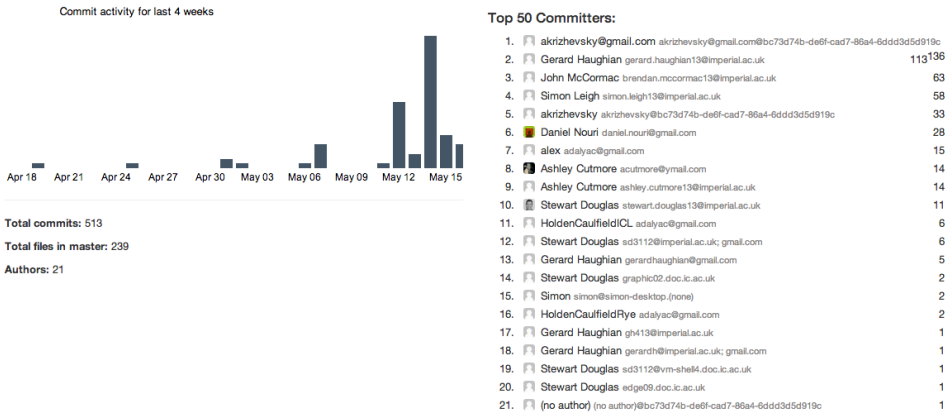


Figure 5: Git Commit Statistics



5.5 Testing

The combination of several distinct pieces of software and multiple languages necessitated various tools for unit testing. A Continuous Integration server was built which automated unit testing after every check-in to our DEV branch. Tests were triggered at the end of each sprint when code was checked in to our QA branch, ensuring the integrity of our code base was maintained. Emails were despatched by the Continuous Integration server to notify group members of the results of these tests. Our integration and system testing involved using our prototype and verifying that a complete roundtrip classification was successful. Each subsection that follows highlights in detail the unit testing strategy employed for each component of the system.

5.5.1 iOS 7 Application

The iOS application is written in Objective-C and makes use of Apple's range of frameworks for accessing hardware features on the iPhone (including camera and GPS) and the drawing of basic UI elements (buttons, images, animations). Xcode (Apple's IDE) is the main workhorse for developing the iOS application and comes with Apple's unit testing framework (xctest) built in.

Taking an initial Black-Box approach and writing tests for core classes before beginning to implement any functionality helped define exactly what each class was responsible for and how it would expose its functionality. As development of the classes proceeded a more White-Box approach was adopted. For example the core server interaction class has private methods for processing the server's JSON response, in Objective-C this could be exposed and tested with a variety of response data in isolation. This mix of the two approaches is described as a Grey-Box approach.

Unit tests should be both quick to execute and isolated of dependencies, to encourage frequent use and increase confidence that test failures point to a specific issue. To remove the delay and add isolation to the networking code the OHHTTPStubs library was used to 'catch' calls made to the underlying Network API and redirected to a simple stub of the API that could return a range of HTTP responses instantly. Code coverage statistics were produced in Xcode by simply passing the appropriate flag to the built-in compiler, LLVM. Cover Story, an open source program, provides the ability to analyse and generate reports from the coverage files. Unfortunately branch coverage statistics are not produced. There are currently 37 unit tests and take $\sim$ 0.3seconds to run.

Module	Statement Coverage	Branch Coverage
Database	77.6% - 111/143	na
FileSystem	98.3 % - 59/60	na
Networking	85.2% - 253/297	na
Image Capture	86.7% - 91/105	na
Total Coverage	84.8% - 514/605	na

Table 1: Code Coverage for iOS 7 application

5.5.2 Servers

A Grey-Box testing approach was used, given that multiple libraries were used for which we did not have visibility of all internal structure. Partition testing was used to select and test boundary cases and try to ensure effective, high quality unit tests.

Mocha [3] was selected as the unit testing framework for the Servers, as it enabled straightforward testing of our asynchronous logic and the necessary integration with node.js, Express and MongoDB. Istanbul [1] was selected as the code coverage tool for our server-side Javascript. It has integration with Mocha (allowing tests to be run and coverage to be computed in one operation), tracks statement, branch, and function coverage and is able to produce easily parsed coverage outputs in XML, JSON or HTML.

The combination of Istanbul and Mocha enabled straightforward creation and automation of our unit tests for the Servers.

Module	Statement Coverage	Branch Coverage
Configuration Parsing	100% - 52/52	100% - 22/22
POST and GET Routes	81% - 102/126	77% - 26/34
Server Instantiation	88% - 46/52	75% - 9/12
Total Coverage	86% - 200/230	87% - 57/68

Table 2: Code Coverage for Request Server

Module	Statement Coverage	Branch Coverage
Configuration Parsing	100% - 52/52	100% - 22/22
Routes	88% - 44/50	62% - 5/8
Server Instantiation	90% - 52/58	86% - 12/14
Update Poll	88% - 95/108	82% - 31/38
Total Coverage	91% - 243/268	85% - 70/82

Table 3: Code Coverage for Worker Server

5.5.3 Plant Classification

Our plant classification framework uses a combination of CUDA C++ and python. The CUDA elements of Alex Krizhevsky's CUDA convnet library (as modified by Daniel Nouri to include dropout) have required no modification and so have not been tested. The modules which we have edited have all been written in python; hence we use a combination of the python unittest and coverage modules. We use a grey-box testing approach, ensuring a high code coverage of the individual units, while also ensuring all of the key specification criteria are met. Many of the test cases used partition testing to try and catch real world 'edge cases'.

In the early stages of development, many very small one off scripts were written. These scripts do not form an integral part of our classification system and so have not been included under our test coverage framework. To properly test the infrastructure requires both the appropriate MongoDB server to be running, and a GPU to test our neural network run scripts.

Module	Statement Coverage	Branch Coverage
Image Manipulation and Batching	83% - 163/196	81% - 47/58
Network Data Providers	96% - 48/50	80% - 8/10
Component Classification (run script)	89% - 143/161	73% - 32/44
Classification Post Processing	83% - 74/89	77% - 23/30
Component Tagging	78% - 109/140	91% - 29/32
Database Querying	93% - 51/55	79% - 11/14
Total Coverage	85% - 588/691	78% - 146/188

Table 4: Code Coverage for Plant Classification

5.5.4 Overall Coverage

The overall coverage is given by taking the arithmetic mean across the sections.

Module	Statement Coverage	Branch Coverage
iOS 7 Application	85% - 514/605	na
Request Server	86% - 200/230	87% - 57/68
Worker Server	91% - 243/268	85% - 70/82
Plant Classification	85% - 588/691	78% - 146/188
Total Coverage	87% - 1545/1794	83% - 273/338

Table 5: Overall Code Coverage

6 Implementation

Below, we list the challenges involved in the implementation of each area of the project and the solutions we developed to overcome them.

6.1 iOS 7 Application

Challenge: *Concurrency*

To ensure a positive user experience, multi-threading was required to keep the UI interactive and responsive while other tasks were performed in the background. Multi-threaded code can be beautifully simple when threads can lock access to shared data or are given immutable data, however this is not always possible when users are involved. For example: A 'specimen' object is needed by the network thread to generate the appropriate network requests to send to the server and possibly make updates. What should happen if the user decides to 'swipe-to-delete' that specimen from the table during the network task? To the user it would make little sense why this action might be blocked and even if every line of the network code was protected by an 'if object still exists' guard it would still not be thread safe. To overcome this, when a user deletes a specimen it is only made invisible. It appears to have been deleted but still resides in memory, except it now has its 'deleted' flag set. The actual deletion can be done at a safe, controlled time such as app start up or termination.

Challenge: *App Store Submission*

The second main challenge was releasing the app on the App Store. It was a long and involved process requiring continual monitoring, discussions, and a number of non-code related tasks, such as designing a logo and brand. One particularly difficult element of our submission was the requirement for our whole infrastructure, from the server to the classifier, to be up and running for the duration: the submitted App could be tested at any time by Apple. This unfortunately led to a significant delay in training our neural network, as well as pending improvements for performance of both it and the Worker server. We also set up a number of systems to alert us to any issues with the server, to ensure it continued working as expected. The benefit of this hiatus was that it proved that the full system could be up and running for an extended period of time, supplying classifications to outside users. Our application was approved before the submission of this report, and is now scheduled to go up on the App store.

6.2 Servers

The main considerations we faced in selecting the software stack for use in our servers were: scalability, stability, compatibility with our database, and ease of implementation.

Both servers are implemented using Node.js with the Express web application framework. MongoDB is our chosen database. Node.js is built on Google Chrome's JavaScript V8 runtime environment. It was selected since it is well supported and easy to develop in. It is highly scalable and easily implements asynchronous data calls making it well suited to our project [13]. MongoDB is a NoSQL document-orientated database which stores data in a compact JSON format known as Binary JSON. MongoDB interacts well with Node.js and the JSON format allows easy message passing with the rest of the stack.

6.2.1 Request Server

Challenge: *Concurrency and Scalability*

As an iOS application may well have a large number of concurrent users, we required a solution that would be able to support high simultaneous demand. Since our classification engine requires special hardware and must devote significant resources to classification, we isolate the Request server on another machine so that it may maintain significant, dedicated resources to accept and respond to HTTP requests received from the iOS application.

Node.js was selected as the environment as it has been proven to cope with many concurrent connections (in fact, up to 1m concurrent connections have been demonstrated [14]) and allows the use of full stack JSON along with MongoDB.

MongoDB was chosen as our database solution as the document model fits well with the data that we must store. Additionally, the ability to shard MongoDB across multiple instances gave us the confidence that we would be able to easily scale out our database if demand proved sufficient.

RESTful HTTP interaction was selected as the means by which the Request Server, Worker Server and App would communicate. We chose this mature and portable standard to ensure that we would be easily able to introduce new platforms, and to enable the production of an open API, should it be desirable in future.

Challenge: *Grouping Multiple Images for Classification*

As we allow the user to take multiple images of a single specimen, we must aggregate these segment images into a group, for which we can track the classification state. Apple impose significant restrictions on the ability of iOS applications to uniquely identify hardware or application users. Thus we accomplish the aggregation of data and session tracking by having the Request Server creating a new group document in the MongoDB database when the first image of a new set is received from the iOS application. The initial HTTP response that is sent to the iOS application then contains the unique 'GroupID'. Further image submissions for the same specimen then have the GroupID embedded in the HTTP POST request made by the iOS application.

MongoDB, as a document based NoSQL database, is an ideal fit for our needs, as we embed the multiple individual segment 'ObjectIDs' that represent each image in one 'super' document representing the group of images for a single specimen. Queries and updates can then be made to all documents belonging to a group, or to individual documents within that group. We may then store the ultimate classification against the 'Group' document.

Once the user confirms via the iOS application that they have finished uploading images of a specimen, an HTTP PUT request is sent to the Request Server containing the unique ID of the 'Group' which is now complete. The appropriate document is then updated in the database, so that the Worker server may then combine the individual classifications for each image into one classification for the whole group.

6.2.2 Worker Server

Challenge: *Receipt of Images*

The Worker Server oversees the storage of user images and the querying of the neural net. It listens for HTTP POST requests from the Request Server, and on receipt of a new image stores the image in a directory created specifically for that image's group. At the same time as it is listening for new images to be sent to it, the Worker Server polls the MongoDB database for images which need to be sent to the neural network for classification. If such an image is found then a link to the images location is passed to the neural network, which genertes a .pickle data file for classification. Some of these actions need to be executed synchronously: a new group folder must be generated and a new image saved into this folder prior to passing the image to the neural net. Like the Request server it makes extensive use of *formidable* for extracting form information. It also makes use of of the *mkdirp* library to synchronously create a new directory. When a new image has been received the Worker server will update the MongoDB database, and the image's relevant entries will be updated to reflect the fact that it is now ready for classification.

Challenge: *Synchronous Execution*

As noted in the previous point, many of the Worker Server tasks must be completed synchronously. For example, we should not attempt to pass multiple images through the net at the same time. This is complicated by the fact that Node.js is asynchronous by default, and so we needed to rely on third-party libraries to make sure certain sections of the code blocked until completion. Initially we used the *async* library, but this depended on using timeouts which led to longer response times. For our final implementation we use JavaScript 'promises', which allow us to write synchronous code is a concise and readable way. The *Q* library was chosen because it was well documented.

6.3 Plant Classification

Challenge: *Computer Vision*

There are three considerable computer vision problems: number and complexity of features, image segmentation, and the need for translation invariance. Specifying at the pixel level what features define each class would require expert botanical knowledge of each species and enormous effort spent hard-coding detectors for each feature. Secondly, in order to recognise a plant in an image, the target needs to be segmented from the image. If the plant is in the wild, conventional segmentation approaches based on edge and corner detection are of little use; hence why LeafSnap requires the target to be placed on a white background. Thirdly, the need for translation invariance, as an object remains the same regardless of where it is located within an image.

We opted for a Deep Learning approach to classification, because it can automatically learn features. It learns features that work best on a training set, so with wildlife training images, features are learned that deal with segmentation. We specifically chose a Convolutional Neural Network (CNN) implementation, the architecture of which is inspired by that of the human visual cortex, and is specifically built to replicate feature detection across all dimensions. This is known to deal efficiently with dimension hopping by achieving translation invariance. Although there are many alternative methodologies (SIFT, Random Forests, and SVMs to name just a few), the record breaking performance of Alex Krishevsky's cuda-convnet [5] project on the same data source we had available [16] and with a comparable number of classes, made it an ideal library upon which to base our classifier.

Challenge: *Data Processing*

Deep learning itself is distinguished from other machine learning techniques by not simply learning the relative importance of features, but by learning the features themselves. The additional information needed for this raises the desired volume of training data. The PlantCLEF dataset which was originally intended to underlie training, provided a meagre average of 12 images per species; our data sourcing pursuits managed to bring this number up to 2,000 per species. The only significantly large labelled dataset was ImageNet.

To ensure our neural network wasnt burdened having to learn to recognise species for which we had a limited training subset. A Bucketing algorithim was created in order to bucket those images into a higher level in our taxonomy tree which we constructed using WordNet [15]. This meant we could decrease our neural network error rate while maximising our use of available training data. Outlined below is the pseudocode for the bucketing algorithm.

```
1: procedure BUCKETING
2:     NumPlants ← global hash table with aggregated count of unique plants
3:     leafNodes ← set containing all leaf nodes in the plant taxonomy tree
4:     for all nodes ∈ leafNodes do UPDATE-DESCENDANT-COUNT(nodes.PathToRoot)
5:     end for
6:     for all nodes ∈ leafNodes do ASSIGN-BUCKETS(nodes.PathToRoot)
7:     end for
8: end procedure
```

```
1: procedure UPDATE-DESCENDANT-COUNT(path)
2:     count ← 0
3:     for all nodes ∈ path do
4:         count ← count + NumPlants[node]
5:         UPDATE Bucket = node, BucketSpecies = Species, Count += count
6:         FROM plants
7:         WHERE SynsetID = node
8:     end for
9: end procedure
```

```
1: procedure ASSIGN-BUCKETS(path)
2:     for i ← 0...path.length do
```

```
3:          bucket ← path[i]
4:          count ← NumPlants[bucket]
5:          if count >= threshold then
6:              UPDATE Bucket = bucket, BucketSpecies = species
7:              FROM plants
8:              WHERE SynsetID IN path[0...i]
9:              break
10:         end if
11:     end for
12: end procedure
```

Our final bucketing parameters for the network set a minimum threshold per class of 2,000 images, which resulted in total 1.2 million images, and 259 separate plant classes.

Our training images, originally in JPEG format were scaled (uniformly so as to preserve the aspect ratio) and cropped into square 256x256 3-channel images. These were then batched into groups of 128 and stored in python numpy arrays; this reduced preprocessing requirements making the CPU available for the more computationally intensive data augmentation stage (discussed below). The image size of 256 was chosen for a number of reasons: the cuda-convnet library was optimised for 256x256 images, we felt there was enough detail for plants of different species to be recognised at that resolution, and finally in the highly uncompressed numpy arrays, a larger image size would have used more space than our 2TB hard disk would allow.

Data augmentation was also an essential part of preventing overfitting on our neural networks; by using label-preserving transformations, such as extracting subpatches from images (in our case, 224x224 patches) taking horizontal reflections, and altering the intensities of the RGB channels with Principal Component Analysis. We used these techniques to increase the variation between successive training batches, and also take advantage the otherwise free CPU while the GPU was training the network weights.

Challenge: *Network Architecture*

We experimented with two flavours of architecture. The first was a single large network trained on the entire labelled dataset, the second was a combination of multiple smaller networks each trained to classify a specific plant component (Leaf, Flower, Fruit, or Entire). We felt the multiple smaller network would have the advantage of learning a more specific set of features, as well as leveraging information provided to us from the user about the exact type of image being uploaded, which could help to improve accuracy. However two key potential drawbacks of smaller component networks were also apparent. Firstly, each component network would only have a small subset of the images which would work to counter performance gains of the more specific feature sets. Secondly, on a single GPU system, the turnaround time for each user request would be much slower, as a new network would need to be loaded into memory for each component type submitted.

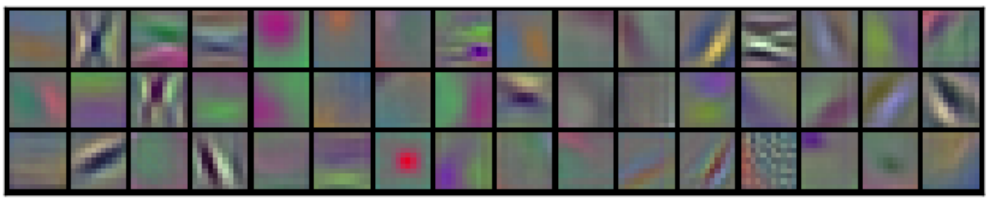


Figure 6: Filters learned by the plant recognisers first convolutional layer

The ImageNet data we had available, was tagged by species (such as Oak), but not by component type (Leaf, Flower, Fruit etc). We therefore trained a separate network on these separate component types (using PlantClef data [2], which was tagged appropriately), and used that network to tag each of our 1.2 million images from ImageNet. The component network itself was based on the CIFAR training network [4], but modified to accept 256x256 resolution images. We reserved 15% of our component training data for accuracy testing,

which achieved a 12% top-1 error rate, giving us confidence to let the tagging proceed as planned.

We trained the Flower component network as an initial test, as it had the largest subset of images and hence we believed would exhibit the best performance. However, after over a week of training, the best top-5 error rate achieved was 28%, which was not significantly better than the performance of the single large net and had the drawbacks already discussed. Given that this was the best performance we were likely to achieve, we decided to use the single large network architecture for our production implementation.

The simpler, single large net was based heavily on Krishevsky's ImageNet architecture [5]. The network consisted of 5 convolutional layers (with max pooling and local response normalization), and 3 fully connected layers. The network use rectified linear unit neurons, which have been shown to significantly improve training times [5]. The network was trained for 12 days on a single GeForce GTX 780, with 259 different plant classes. Each class has on average approximately 4,500 separate images. The top-5 error rate achieved for 102,400 test and cross-validation set cases was 29%.

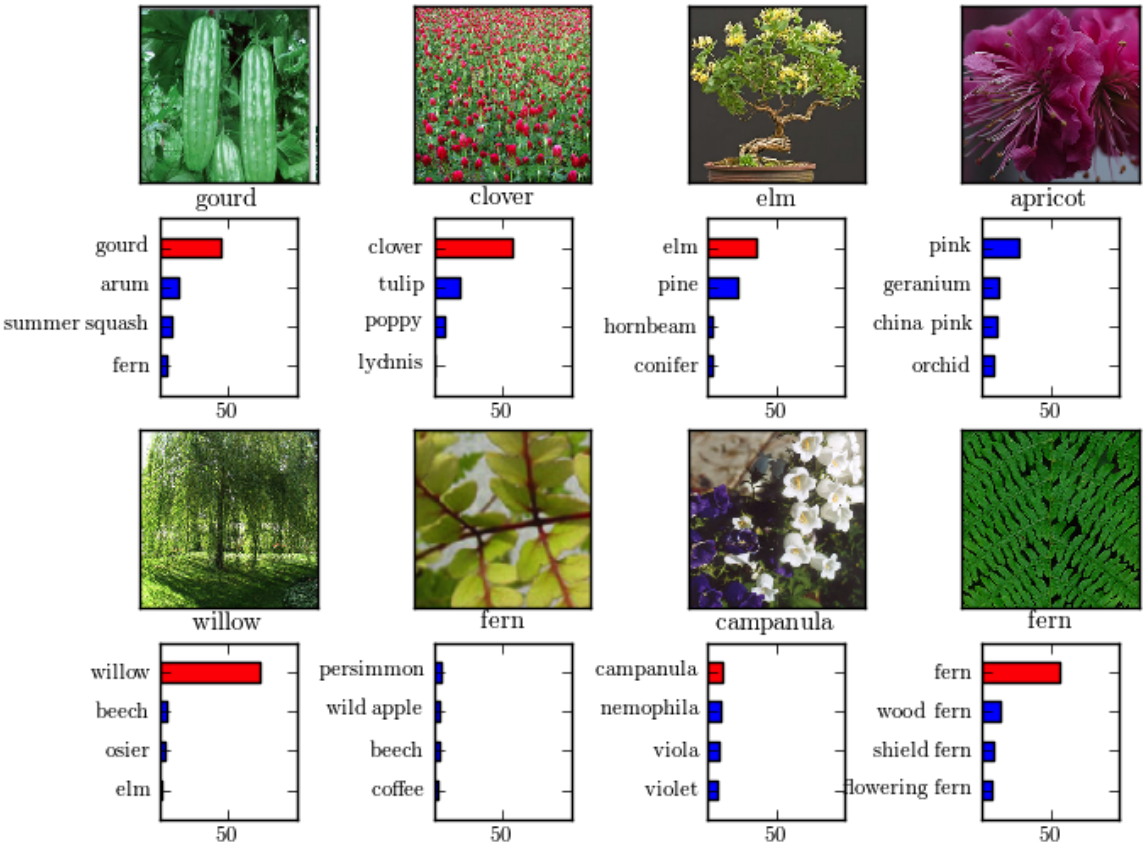


Figure 7: Some random predictions of previously unseen plant images

7 Final Product

7.1 Product Evaluation

We have succeeded in releasing an iOS App for recognising plant species in the wild. We have succeeded in developing a deep-learning convolutional neural network that can classify plants. Every task agreed, and highlighted in the Specification section as Essential have been achieved. We have in effect, delivered a product that meets the original specification. This includes creating the first ever App that automatically recognises diverse species of plants in the wild.

7.2 Comparative Performance

The performance of our app can be compared to its two competitors, LeafSnap and Google Goggles. Ours is the only one which can recognise plants in the wild. LeafSnap can only recognise leaves, and needs the leaf to be torn off and placed in front of a white background. Google Goggles aspires to recognise any class of object (e.g. painting, monument, plant, animal etc), but fails on plants. In fact, its "overview and requirements" page warns the reader that its performance is "not so good" with plants.

Image Set	Number of Images Tested	Top-5 Error	Top-1 Error
Test Batch	88,320	29.0%	54.6%
Cross-Validation Batch	14,080	29.4%	55.2%
Overall	102,400	29.1%	54.7%

Table 6: Overall Performance of our Neural Network

7.3 Further Development

We have identified several areas where we feel we that optimisations are possible.

- **Ensuring High Server Availability and Robust Error Checking**
  Users of an iOS application have a justifiable expectation that the application works, regardless of the time of day. In our case, this means keeping our servers up and running 24 hours a day. This provides many challenges, but one in particular is the handling of error conditions occurring within our code. We cannot simply allow a server that experiences an error to gracefully exit and log the failure for later inspection. The server code should continue to run and receive and respond to the requests of other users, whilst logging the error. We have tried to accomplish this by catching all errors and logging them, but there remains a danger that whilst the server code continues to run, it or the database may be left in an inconsistent state that may result in undefined behaviour. We believe that a priority in further development would be the creation of monitoring scripts that monitor the health and consistency of items in the database and the results being produced by the Classification Engine and Request and Worker servers. These scripts would cleanse the database of old, unclassified jobs and remove any erroneous data, as well as monitoring the speed and viability of responses sent to the App.

- **Enforcing Schema on a NoSQL Database to Prevent Inconsistent State**
  Whilst the use of MongoDB provides us with great speed and scalability, the lack of an enforced schema means that is is incumbent upon the developer to ensure that all documents inserted or updated are consistent with their proposed data schema. We have been successful in this goal, but it has required significant collaborative effort. A point for further development would be the production of schema configuration files, within which variables and constants could be contained and read in by each code component that made use of the database. In this manner we would enforce greater schema consistency across our code-base.

- **App Responding to Server-Side Error Conditions**
  The user-facing server protocol we devised during the early project meetings, though sufficient for our needs, did not included error handling. The App currently responds to errors by waiting for a set time period and retrying silently (without notifying the user). We could improve upon this by agreeing on both an error message the server could return on consistent classification failure (e.g image corruption) and a time-out period to stop retrying. The spinning classification processing icon could then turn into an exclamation mark and when 'tapped' by the user, an error message displayed informing of the possible cause (network issues, image error). Finding the right balance for the time-out will require user research and prolonged software testing to ensure the time is long enough for the server to respond even when under pressure but short enough for the user to not become frustrated from lack of further information.

7.4 Future Opportunities

The commercial opportunities of WhatPlant are also of interest. A user spotting a pleasing plant in a park needs to identify its species in order to order the specimen in a shop. This places WhatPlant at a significant vantage point in the gardening consumer life cycle. The App could be augmented with hyperlinks to purchase the recognised plant specimen on partner e-commerce platforms and earn a commission on a per-click and per-purchase basis. We have already have many signups on our website, whatplant.github.io, which suggests a strong level of interest among the general public.

Another opportunity lies in the creation of citizen science initiatives for the control of invasive species. The GPS mapping capability of a smartphone could be coupled with the app's species detection to monitor ecosystems and flag new species.

The most attractive aspect of our project is its modularity: WhatPlant could be forked to produce WhatDog, WhatCat, WhatFish, WhatBird, WhatFabric, WhatDrink with relative ease, considering the data availabilities on ImageNet. The servers would require no modification, the machine learning would only require retraining, and the UI would only require minor adjustments such as a new logo.

7.5 Project Evaluation

Overall, we will certainly classify this project as a success. We all feel that we have learned a great deal about both the technical implementation of our chosen software, and how modern software development techniques are put into practice and the benefits they offer. The group worked very well as a team, all remaining dedicated to the project, putting in the extra hours and respecting each other's opinions. We are proud of what we have produced, and each of the group can attest that they contributed their fair share of work to the end result.

Finally, we would like to thank our supervisors Dr William Knottenbelt and Mr Jack Kelly for their constant support and ineffable enthusiasm.

References

[1] Krishnan Anantheswaran, *istanbul: A Javascript code coverage tool written in JS* URL: `http://gotwarlost.github.io/istanbul/`, last accessed 9th March 2014.

[2] H. Goeau, A. Joly, P. Bonnet, *LifeCLEF 2014, Plant Task.* URL: `http://www.imageclef.org/node/179`, last accessed: 11th March 2014.

[3] TJ Holowaychuk, *Mocha - the fun, simple, flexible JavaScript test framework* URL: `http://visionmedia.github.io/mocha/#getting-started`, last accessed 9th March 2014.

[4] A. Krizhevsky, *cuda-convnet, High-performance C++/CUDA implementation of convolutional neural networks.* URL: `http://code.google.com/p/cuda-convnet/`, last accessed 11th March 2014.

[5] A. Krizhevsky, I. Sutskever, G.E. Hinton, *ImageNet Classication with Deep Convolutional Neural Networks.* URL: `https://www.cs.toronto.edu/~hinton/absps/imagenet.pdf`, last accessed 11th March 2014.

[6] Alex Rodriguez (IBM), *RESTful Web services: The basics*, 06 November 2008. URL: `https://www.ibm.com/developerworks/webservices/library/ws-restful/`, last accessed 9th March 2014.

[7] Apple Appstore, URL `https://itunes.apple.com/gb/app/leafsnap/id430649829?mt=8`

[8] Apple Inc. *"80% of devices are using iOS 7."*, 26th January 2014, URL: `https://developer.apple.com/support/appstore/`, last accessed: 30th January 2014

[9] CUDA Parallel Computing *"CUDA is NVIDIAs parallel computing architecture that enables dramatic increases in computing performance by harnessing the power of the GPU (graphics processing unit)."*, URL: `http://www.nvidia.co.uk/object/cuda-parallel-computing-uk.html`, last accessed: 13th May 2014

[10] The Editors, *Scientific American Staff Picks: 10 apps for Your Smart Phone or Tablet*, 25th December 2012. URL: `http://www.scientificamerican.com/article/scientific-american-staff-picks-10-apps-for-smart-phone-tablet/?page=6`, last accessed 30th January 2014.

[11] formidable: A node.js module for parsing form data, especially file uploads. *"A node.js module for parsing form data, especially file uploads."*, URL: `https://github.com/felixge/node-formidable`, last accessed: 12th May 2014

[12] ImageNet Fine-Grained Challenge, URL: `http://www.image-net.org/challenges/LSVRC/2013/`

[13] Joyent Inc. *"Node's goal is to provide an easy way to build scalable network programs"*, URL: `http://nodejs.org/about/`, last accessed: 30th January 2014

[14] Node.js w/1M concurrent connections URL: `http://blog.caustik.com/2012/08/19/node-js-w1m-concurrent-connections/`, last accessed: 13th May 2014

[15] Princeton University, *WordNet: An Electronic Lexical Database. Princeton University. 2010.* URL: `http://wordnet.princeton.edu`, last accessed: 12th March 2014.

[16] Stanford Vision Lab, Stanford University, Princeton University *Image-Net: Image-Net 2014.* URL: `http://www.image-net.org`, last accessed: 15th May 2014.

[17] VersionOne, URL: `http://www.versionone.com`, last accessed: 30th January 2014.

[18] WJLA.com, *Mobile Tree Identification App Leafsnap Already Downloaded 150,000 Times*, 8th June 2011, URL: `http://wj.la/mGc1kj`, last accessed: 30th January 2014.

A Code Coverage Reports

A.1 Plant Classification Coverage Report

Coverage report: 84%

Module	*statements*	*missing*	*excluded*	*branches*	*partial*	*coverage*
/homes/bjm113/beLeaf/ML/bucketing/mongoHelperFunctions	55	4	0	14	3	90%
/homes/bjm113/beLeaf/ML/combine	89	15	0	30	7	82%
/homes/bjm113/beLeaf/ML/cuda_convnet/plantdataproviders	50	2	20	10	2	93%
/homes/bjm113/beLeaf/ML/noccn/noccn/dataset	196	33	0	58	9	83%
/homes/bjm113/beLeaf/ML/noccn/noccn/tag	140	31	0	32	3	78%
/homes/bjm113/beLeaf/ML/run	161	18	0	44	10	85%
Total	**691**	**103**	**20**	**188**	**34**	**84%**

coverage.py v3.7.1

Figure 8: Sample Plant Classification Coverage Report.

A.2 Request Server Coverage Report

Code coverage report for **All files**

Statements: **86.96%** (200 / 230) Branches: **83.82%** (57 / 68) Functions: **100%** (26 / 26) Lines: **86.96%** (200 / 230) Ignored: **5 statements, 1 function, 3 branches**

File	Statements		Branches		Functions		Lines	
AppServer/	88.46%	(46 / 52)	75.00%	(9 / 12)	100.00%	(3 / 3)	88.46%	(46 / 52)
AppServer/config_parser/	100.00%	(52 / 52)	100.00%	(22 / 22)	100.00%	(1 / 1)	100.00%	(52 / 52)
AppServer/routes/	80.95%	(102 / 126)	76.47%	(26 / 34)	100.00%	(22 / 22)	80.95%	(102 / 126)

Generated by istanbul at Wed May 14 2014 16:36:47 GMT+0100 (BST)

Figure 9: Request Server Coverage Report.

A.3 Worker Server Coverage Report

Code coverage report for **All files**

Statements: **90.67%** (243 / 268) Branches: **85.37%** (70 / 82) Functions: **83.33%** (30 / 36) Lines: **90.77%** (236 / 260) Ignored: **13 statements, 3 functions, 8 branches**

File	Statements		Branches		Functions		Lines	
GraphicServer/	88.55%	(147 / 166)	82.69%	(43 / 52)	79.31%	(23 / 29)	88.68%	(141 / 159)
GraphicServer/config_parser/	100.00%	(52 / 52)	100.00%	(22 / 22)	100.00%	(1 / 1)	100.00%	(52 / 52)
GraphicServer/routes/	88.00%	(44 / 50)	62.50%	(5 / 8)	100.00%	(6 / 6)	87.76%	(43 / 49)

Generated by istanbul at Fri May 16 2014 01:24:55 GMT+0100 (BST)

Figure 10: Request Server Coverage Report.

A.4 iOS Application Unit Tests

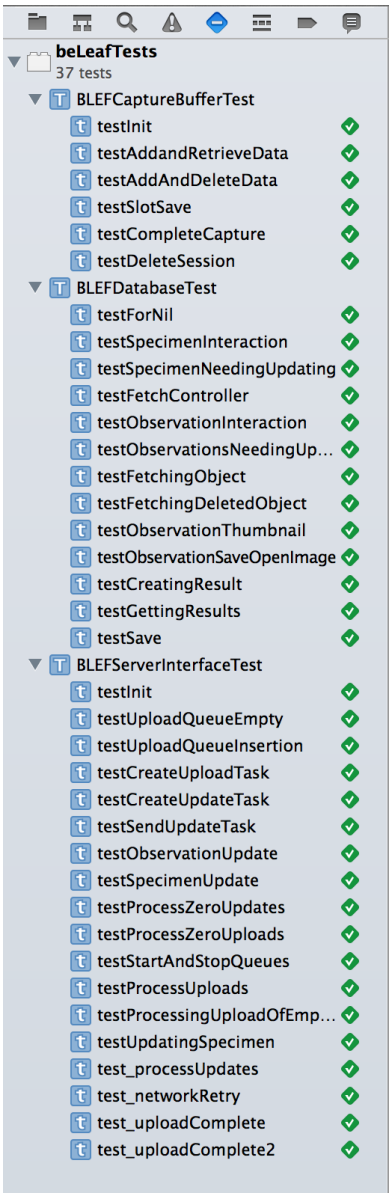


Figure 11: Xcode test suite.

B Logbook

B.1 Detailed Work Breakdown

Week Ending	Gerard	Alex	Simon	Ashley	John	Stewart
24th Jan	20	15	10	5	10	5
31st Jan	15	5	20	15	18	15
7th Feb	25	10	20	10	10	5
summary of work done:	Wrote startup script to configure an environment with all components and optional stubs if necessary. Wrote a script to parse mocha unit test results in an email friendly way. Wrote a script to convert image meta-data from xml to json and populate a MongoDB collection. Installed necessary software on VM and graphic02	Wrote a bash script that automates installation and setup of neural net to make it operable for all team members. Practiced training neural net on PlantClef images by varying parameters.	Various pieces of configuration on the server (mongoDB, permissions). Implemented POST request requirement on AppServer, and linked to MongoDB. Implemented Configuration file parsing for node AppServer.	Re-wrote iPhone app to have much more flexible logic and control flow. App now has custom camera view with 'segment selection'.	Created neural network using plantclef data to categorize an image of a plant into different plant components (i.e. entire, leaf, flower, fruit, branch, stem). Wrote a tagging script which uses the above network to tag xml files corresponding to images with the component of the shot, and a batching system which takes xml data and jpgs and stores this into python pickled numpy arrays with meta-data for efficient training.	Learned about node streams and integrating with databases. Tried to implement sockets.
14th Feb	20	15	15	20	20	15

summary of work done:	Followed up with NHM re:collaboration. Increased stability of CI Server. Identified appropriate code coverage framework for NodeJs. Wrote a python script that will convert Alex's bucketing algo results into documents and insert them into MongoDB. Extended the xml2json util script to handle new logic of bucketing information being stored in MongoDB.	Implemented object-oriented graph class in python to represent taxon tree. Wrote a bucketing algorithm that determines which classes can and should be classified based on number of images available and parent-child relationships, and generates a data structure optimised for mongodb querying. Tested graph building methods and added code to enforce tree structure.	Improved parsing of configuration files. Implemented HTTP GET request on AppServer. Migrated AppServer to use native Mongo driver.	Implmented a queue processing system for the app for both uploads and result requests. After user permission promot, the location of taken photos will also collected and stored in the DB.	Trained 6 (very basic) model networks for each component type (created and stored accompanying batch data also). Created a main script which takes an image and tag, loads the appropriate network (based on a config file and the tag), and outputs the probabilities of top 5 most likely species. Also created a 'test' stub equivalent for testing purposes.	Implemented the graphic02 node server such that it can listen for an HTTP request. Integrated John's python script.
21st Feb	10	0	10	10	15	10
summary of work done:	Installed and configured code coverage tool for NodeJS code. Extended startup script to start up in pseudo-dist mode and quiet mode. Continued to work on testing infrastructure		Resolved mongo connection issue on graphic. Altered config files to allow pseudo-dist mode.	Cleaned up result page design for App Demo. Unit Test and Coverage Reports for xCode project up and running. 98	Reviewed image data and extracted required subsets of plants from imagenet. Created taxon tree of the plant data, with accompanying metadata. Customised tagging script for our imagenet data and tagged xml files with their component type using our trained net.	Used John's python script to append JSON to the VM mongo. Exported node files to a single plant.js to allow for code coverage.

28th Feb	35	0	10	15	20	10
summary of work done:	wrote script to convert image meta data xml files into json, inserting them into Mongodb with appropiate wordnet columns etc. stored wordnet and taxonomy tree in mogodb(scripts provided). wore a scirpt to update the path from root to all nodes of tanonomy tree. worte bucketing algo javascript queries/functions to dynamically pull image batches from mongo for training neural nets. wrote help function script in python to facilitate interacting with mongodb and pulling in bucketed data and a function to dynamically exclude synsets from the bucketing algo.		Wrote unit tests for config parser. Established how to connect to mongo and how to use Istanbul and Mocha coverage and test frameworks in asynchronous manner. Wrote unit tests for server and routes (REST API) for appServer. Refactored code in server to enable easier testing.	Completed Unit Test coverage for App's core functinality (Database and Network Classes). Began initial structuring for the new design.	Created test suite framework for machine learning modules. Created data augmentation script with PCA analysis to mimic natural lighting changes. Reviewed and excluded some synsets from database. Outlined and began work on combining multiple results into one classification through bayesian system. Began work on pulling bucketed classes into batching script for training a network.	Began unit testing for node and started code coverage
7th Mar	10	10	15	30	25	15

summary of work done:	Completed unit test coverage for bucketing algo	Wrote tests for augmenting data provider, creating dummy images, making sure forward increment, downward increment, flipping work.	Rewrote code for AppServer and adjusted database scheme to contain both group and segment collections, in order to interface correctly with the revised version of the iPhone App. Revised and added to test coverage to ensure new logic was being tested effectively. Started to make insertions into Report2.	Finished moving App's UI to new design structure. Modified database and networking to be compatible with new classification architechture. App automatically continues performing network actions on launch. Added more unit tests to cover these changes.	Batched image net data, and trained and supervised 'one-big-net' model. Run script now stores a pickle object of the result. Batcher now creates a super set of the classes from a combination of nets and filters. Added combine script to use a baysian optimal classifier to assemble final classification. Added unit tests for all of the above.	Updated graphic server so that, upon receiving a post request it creates a new 'group' directory to save images, and updates mongo to say the segment image has been received. Began work on polling mongo for 128 images.
14th Mar	10	10	10	10	10	15
summary of work done:	Mostly report 2 work, and small bug fixes	report 2, more work on data provider	Creation of content for Report 2, draft reading, further unit tests written.	Moved image-capture logic into seperate class, increasing line-count for code coverage.	Built prototype templates of a smaller architecture for four separate networks. Some minor bug fixes in the combine and run script.	Created separate process, poll.js which polls Mongo for new images in graphic and gets their location. Implemented skeleton queueing logic
21st Mar	0	20	10	20	5	18

summary of work done:		Trained multiple versions of a leaf network, using different network architectures, dropout levels, component tagging thresholds, data augmentation hyperparameters.	Testing of interaction between App and Request server. Adjustment of structure of JSON responses to restrict data being passed to iPhone to only that necessary for the results to be parsed.	Adding ability to change debug/production URL within the settings.app - Photos taken are center-cropped to 512x512 before being saved and uploaded. App can now generate and send notification of the 'upload-session' being 'completed' ready for final classification. Corrected JSON parser to conform to new result response. Many UI improvments: displays 'beleaf' level as a piechart, results are ordered by confidence, proccessing results have an activity indicator. Fixed resultview scroll-bounce-bug. Tried different layout/visibility options for the camera interface.	Working with the Node server to ensure the script can be run from other locations on the central graphic02 machine.	Tested executing python run.py and combine.py scripts from Node server. Each group is augmented with a new field upon receipt of an image e.g. leaf: '/path/to/-file', which are then combined when the user presses 'upload'. Queuing logic now working satisfactorily.
28th Mar	0	0	0	0	0	0
4th Apr	0	0	0	2	0	0
summary of work done:				App sends 'PUT' on user cancel		
11th Apr	0	0	0	0	5	15

summary of work done:					Built and tried to train a network on flower components only. Error rate was worse than that of the one big net after a week of training	Built team website using Bootstrap, including signup field to track interest
18th Apr	0	0	0	0	20	0
summary of work done:					Built a prediction suite to test the error rates by plant and overall of our nets. Calculated network error for current network. Built a small wrapper process which keeps a single net in persistent memory and can accept request via a client process. Modified the training parameters to enlarge the test set to an arbitrary number of batches. Modified the provider classes to take random image crops and augment luminosity via pca. Started training another big network to improve performance and test with prediction suite	

25th Apr	0	0	0	5	0	0
summary of work done:				Updated UI theme		
2nd May	0	0	0	0	0	0
9th May	0	0	0	2	0	0
summary of work done:				Fixed interface autolayout constrains. Add swipe-to-change-segment on camera view		
16th May	0	0	10	6	0	0
summary of work done:			Updated test suite for NodeJS, ensured dependencies were locked at working versions. Created README, rationalised folder structure, adjusted Makefile. Implemented kill script for old and cancelled submissions.	Updated unit tests. Worker Server debugging session		

B.2 Minutes of Group Meetings

B.2.1 13/12/2013 13:00

__Unify our approach to collecting data.__

Matt wants to just look at leaves to keep it within the scope of the project.

Alex disagreed. Suggested deep neural networks will allow for training.

It was pointed out that one does not want to overshoot with data set.

Jack pointed out that one scenario is that one group focusses on a similar but different pattern classification type (i.e. malarial vs normal blood cells).

Jack is agnostic as to in which direction we go. Pattern classification is important but otherwise the subject of the classification is no big deal.

Dataset does already exist for plants.

Jack will resolve issues with the graphical machines such that they will be rebooted into linux and we can ssh in.

Option given as to whether to do any app side or just work on image classification side of stuff.

Stewart raised the point of competing – we can take part in PlantCLEF.

Jack advised that a mistake he made on group project was to arrogantly decide he knew all about version control and did not go to software engineering lectures. The first two reports are a very dry box ticking exercise. We must complete tables to put into the reports. The only way of finding out what needs to go into them is to go to the lectures.

Python already comes with unit test frameworks. Life may be harder with the deep neural network testing.

NIPS conference has just occurred and a whole bunch of new papers have been released.

Things to do:

* Email Natural History Museum, see if we can get them on board as a client. GH group agreed to do this
* NHM have a community crowdsourcing urban tree survey into which we would be able to contribute.
* Also perhaps with speaking with LeafSNAP.

GH asked if we are expected to include test coverage for the neural network. Can we actually test for this? Is it not that our algorithm would be rubbish it it didn't work?

JACK: Don t need to unit test the whole stack. Just need to test the stuff that we write. Don t need to unit test CUDA-CONVNET.

Unit test the functions on their own and then once they are plugged together to be a net, the concept of unit testing becomes vapid as the result speaks to the success of the stack.

Embrace and love testing early and often.

JACK: Software Engineering lectures are drenched in technicalities. Just try to extract the salient points. The first two reports will require technicalities to be sprinkled between them.

External contact requests to be sent via Jack such that we don t have duplication and mixed signals.

Jack: Jeff Hinton Coursera course on ML worth going through.

Alex: Start at lecture 3, he also has downloaded everything.

GH: Start to try and get some data – what do we need, what do we not need?

B.2.2 16/01/2014 12:00

Attendees:

Gerard
Stewart
Simon
Ashley
John

Apologies: Alex D

Gerard suggested there are three elements:

Machine Learning,
Server,
UI,

but also fourth element INTEGRATION.

Someone needs to work on the fourth element of Integrating the different components.

We could feed this into continuous integration. Gerard suggests building auto-tester so that whole thing gets built and tested whenever change is checked in.

Everyone agreed that we do need the formal Integration component.

Ger will take ownership that everything does come together in one piece.

John created a git repository for himself to test things – should he make this sub-module of master?

The thinking is that we have individual Repos and sub-modules within the master that will only have the sub-modules updated with a stable version of the individual workers repo (for their section).

Question raised: how do we unit test the app and neural network? Neural network particularly hard to unit test, perhaps we just have to look at input/output and error rates.

Ashley mentioned that there are other libraries that automate touch, such that we could automate the testing of the touch interface of the app.

Ger suggested the higher level unit testing is how does flow work from one module to the other.

We should produce a formal specification for what the server can accept and how it will respond. This should be absolutely formalised. This should include the specification of timeouts – i.e., the server will timeout after 60 seconds.

Ger mentioned he had thought about how we handle a user submitting the photo, we need to categorise it.

Simon suggested one thread monitoring input, one monitoring output and

tying them together to mark responses.

John suggested how we handle the storage of items uploaded. If we get a good collection, do we store it in a db and then use it to retrain the Neural net. It was agreed that we do want to do this but it should be a stretch goal.

Stretch goal: If you hash each image, and you then see if a result has been computed for that hash in the past, if the neural network version was recent (or acceptable under some criteria tbd) then you don t need to load the neural net and can give a really fast response.

We agree that the bare minimum.

We agreed that we would prefer not to have joint meetings, it is a detriment to jack s time, not us.

Each separate group investigate what it would take to gain the minimal requirements reference Report 1 wiki page for what we need to focus on and reference.

B.2.3 16/01/2014 13:00

```
Meeting Information
-------------------

*Objective*: Kick off Meeting for Jack to introduce project
    requirements
*Date*: 16/01/2014
*Location*: DOC Meeting Room
*Chair*: JK
*Minute Taker*: SL
*Attendees*: Entire Group, Jack Kelly, The Others!


> Jack said we've all seen the shared gitlab where we can use the
    wiki and share ideas and access computers etc.

* For us doing Plant Recognition, we need to know PlantCLEF don't
    allow external training.

> At the end of last year it was unfair as one PC had 1gb ram and one
    had 3gb.
> Now we have graphic05 and graphic02 which have equivalent spec with
    GTX780s.

>Jack:
>       will forward any problems to CSG but let him know if any more
    issues.
>       Jack away from 22-29th January.

* 1st report due on the 31st, we should meet separately on one of the
    two days after he is back.

* Report 1 is box ticking exercise, just go through the description on
    the website, extract all single points needed in report and make
    sure that they are in there.

* Game is to separate out the tasks of what is essential vs stretch
    goals. Then we should send draft to Jack to fine tune the balance of
     what is in each section.

* Jack will forward a first report from last year so that we get a
    sense of what is expected.

> Gerard:
>       Raised the point that we don't really know what should be
    included.

> Jack suggested that we need to send an email with a rough list by
    Monday morning of what we think should be included vs what we think
    should be a stretch goal.

* Flag anything that we think is going to take a long time or might be
    risky.

* For example, if we've tried something and found it difficult, then
    we should highlight that this might be problematic.

* To confirm, Jack will send report 1 to us today, by Monday we will
    send our thoughts on what we should do/not do.
```

> Ashley asked if we need to interact with Will at any point this term.

> Jack suggested that it s worth meeting with him at some point this term, but no need before report 1.

> Contract between supervisor and group is nice way of thinking about it. But real purpose is to define our goals so that later in the term we can be judged on whether we achieved the goals that we set out to achieve.

> It s main purpose is to make concrete in everyone s minds what the aims of the project are. Marking determined on whether we achieve the goals or not.

> Complex beast, mainly based upon your presentation. Do not expect everyone to read word for word through report and code.

* PRESENTATION MUST BE GOOD. It is worth doing the sexier option on screen as it will be marked on wow factor in presentation.

* NHM are up for meeting us, we just need to agree when this be done.

* We should get back to the NHM with a date for this meeting.

> John said the data from LeafCLEF is all french stuff, so it mightn t translate that well to UK Flora.

> We are not just getting up and showing our competition trophy! We need to consider UK stuff as well.

* LeafSNAP will have US data we should just go and get our data from wherever we can.

* The question was raised as to how we enter PlantCLEF, it is believed that we have to just enter the results of our run on their test data. TO BE CONFIRMED.

> Jack: we will no longer have shared meetings, just whenever we have a problem or want to meet we should let Jack know and we will do this individually.

> EVERYONE LEFT BUT OUR GROUP

* STRETCH GOAL: ALEX made the pointer regarding us to invasive species, do we want to be able to flag invasive species in locations that are not expected and raise these for investigation by a user/ institution based upon the probabilities

* We need to sit down and hammer out a rough wiki on what we are going to do in each different area.

B.2.4 17/01/2014 18:00

```
Meeting Information
--------------------

*Objective*: Decide upon minimum and stretch deliverables for Plant
    Recogniser.
*Date*: 17/01/2014
*Location*: SCR
*Chair*: GH
*Minute Taker*: SL
*Attendees*: Entire Group


> Ashley pointed out that in the report Jack sent there are two tiers
    of things (essential/non-essential) also split into (low/med/high)
    risk.

## 1. Deliverables


### FrontEnd:

#### Essential Requirements:

1. iOS7 iPhone app that can:
2. Take Photos.
3. Store Photos.
4. Upload Photos to server.
5. Receive and handle probable classification from server.
6. Display classification results to user.

#### Non-Essential Requirements:

__General__:
1. HTML-5 web interface that can be accessed from any browser and
    allows one time image upload and response.

__App__:
1. Requests image of each probable result for user comparison, displays
     to user.
2. Requests and displays short text description of species to user.
3. Allows user to connect to wikipedia entry for species.
4. Highlights to user potentially invasive species.
5. Highlights to user poisonous/edible species.
6. Allow user feedback (i.e., allow user to say they are  unhappy
    /  happy  with result).
7. Crowdsource tagged images from users.

8. Furthest stretch is app store / android app.

### Machine Learning


#### Essential Requirements:


1. Deep Neural Network utilising the Cuda-Conv implementation that can
    take an image and return a probability Matrix.
2. Network can handle image of a quality/resolution detailed enough to
    produce relevant probability Matrix.
```

3. Neural Network can produce a result within an acceptable time interval (to be determined) to allow interface with Server.
* [DO WE INCLUDE A PERFORMANCE METRIC HERE FOR DISTINGUISHING BETWEEN SPECIES]

Non-Essential:
*[DO WE INCLUDE A PERFORMANCE METRIC HERE FOR DISTINGUISHING BETWEEN SPECIES]
1. Network able to compete in LifeCLEF.
2. Outline the levers that were used to optimise the neural network, providing useful information for others in future.

Server / DB

Essential Requirements:

1. Receive image information from App.
2. Implement basic Queue for requests.
3. Store image in Database with appropriate tagging information and results (geo, time etc.).
4. Request that Neural network process the information.
5. Receive output from Neural Network.
6. Pass information back to user.

Non-Essential Requirements:

1. Authenticate requests.
2. Ability to send requests to multiple Neural Networks.
3. Hash Image.
4. Compare Hash to previous uploads.
5. Check Neural network version used to tag image if existing hash in database.
6. Vary responses to requests based upon previous hashes and age of network used to tag.
7. Implement open API to our backend.

> Stewart pointed out that the stakeholders hasn t been addressed.

> Alex pointed out about adding stuff to wiki if we discover further useful information.

B.2.5 22/01/2014 14:00

```
Meeting Information
-------------------

*Objective*: Firm up how to operate effectively as a group and set out
    architecture.
*Date*: 22/01/2014
*Location*: DOC Meeting Room
*Chair*: GH
*Minute Taker*: SL
*Attendees*: Entire Group

## x1. Where and how to keep minutes?


> AC:     Do not need to be super detailed, just covering what we have
    said and decided.
> SD:     GitLab for the minutes.


   * Create a separate repo for the minutes on Git, all minutes will be
       stored there. SL to own that action point.
   * Documentation repo. Also put reports in here.

## x2. Discussed what method to use for IM:


   * IRC -> A bot to log when people are working, a room to discuss
       things.
   * We agree to keep track of our own time by logging when we are
       working, and adding to spreadsheet. Gerard will take care of
       setting this up.

## x3. Discussed how to use github issues board:


> GH:     Once we've had meeting with will, GH will put each
    requirement into tasks and put them on the issue board. Then there
    is an issue for each piece of work. Then GH will schedule that into
    sprints.

> AC:     Suggest including duration in issue header.

### WIKI:

> AC:      Anything actually being documented should be put on there (i
    .e. server API).

## 4. Working as a team

> SD: Scrum each day? Think we'll establish that once we start.


   * We will have official weekly meeting on a Wednesday.
   * Monday Scrum  /Wednesday Proper Meeting /Fridays Scrum
   * 12 hour during week SLA.
   * By Monday lunchtime after weekend.
   * Raise on Wednesday meeting any specific unavailability.

>Ashley: 07590005295
```

```
>Stewart: 07770280997
>Gerard: 07753598038
>Alexandre: 07794033992
>John:    07590607299
>Simon: 07711516675

## 2/1. Release Management


> GH: Three branches: Master, QA, DEV.

>      Branch from DEV for a particular issue, once finished, check back
     in to DEV.
>      Once finished with an issue, check into QA branch. User and
    Integration Testing.
>      Release to Master at end of Sprint. That is then latest stable
    release.
>
>      We should have a second party signing off the code before being
    pushed to QA.
>      Unit tests should be done at Dev level before QA commit.
>      We will go with the flatter (directories for each thing)
    structure for everything, one repo with the three branches and
    subdirectories.



   * Gerard will keep up to date the git commands for how to checkout
      etc.

Gerard owns release management and will push to QA and master etc.
Everyone else just pushes to DEV.


   * Can we password protect master - GH will investigate?


## 3. System Architecture.


httpServer is keeping track of the images uploaded and responses given
    and adding info to DB.

plantClassifier on Graphic02 has interface server and the plant
    classification.

gateKeeper will check whether something is organic or not.


## x. First Report

Report one is currently the priority and is due in 8 days time.

Ashley has started writing HTML5 interface,
```

B.2.6 27/01/2014 15:00

Meeting Information

Objective: Discuss with WJK the objectives for the Project.
Date: 27/01/2014
Location: 371 - WJK Office
Chair: WJK
Minute Taker: SL
Attendees: GH, JM, AC, SL

> WK:

Why is it challenging?
How are we going to demo it?
What is the audience going to see, how is it going to impress them?
How are you going to make it happen?
When we stand up and talk to people, what are we going to stay, what story are we going to tell, and what is its culmination?
Panel present at demo, at least two people see all of the projects, everyone on the panel gets input. People outside of the are need to understand what we have been doing.
People don't set out importance of the area, examples of mistakes (forex example), you must set the scene and not just bombard with data and flip around with columns!
Students have done Poker robots, they haven't said what, why a challenge?
Tell the story! Keep people engaged!
Nice focussed story, with challenges that can be explained.
Be careful with black box Neural network. When it is the same data it was trained on!

> GH:

Emailed Will the requirements, so we will step through them.

> WK:

iPad app small extension from iPhone app.
Once decision made for iOS, might as well add iPad as an extension.
Will thought our spec was incredibly broad.
Generate excitement in user.
Don't care about essential and non-essential requirements. Choosing 1 of 3 items.
Don't care what was agreed with supervisor. He has input but is not the determining factor.
Doesn't matter what's essential and non-essential. Create an engaging and exciting experience.
As we go probably specs will change, if we start doing really well in competition then we can seriously concentrate on that.
It is ultimately about what is actually acheived rather than ticking requirement boxes.
Non-essential requirements is massive, gives us plenty of freedom.
WORK BACKWARDS FROM THE PRESENTATION.

What are we going to show people that is going to make them go "wooow".
iSeeNotes. An app that recognises and plays sheet music.
Would you be too dissapointed with the right hand of baa baa black sheep, rather than Chopin's fantasia.
Presentation was beautiful, guessing what music is etc.
Always consider what is something going to look like for somebody in the audience, what makes it worth paying attention to.
List of requirements is fine, but incredibly ambitious.
Essential requirements is fine, and has the right focus.
Don't be scared if we discover one of those non-essentials starts to sap all of our time.
Endemic vs non-endemic species would be a great extension of such a work.
We must motivate it well.

> JM:

We are using Scrum.
We were going to use GitLab for issue tracking.

> WK:

How are we going to submit Burndown chart etc.
We should take advantage of stuff.
JIRA or TRELLO for issue tracking.
We must be tracking the things that we are supposed to be tracking when doing a scrum methodology.
Where are we, what's to do, what have we done etc?
Must have momentum up going into the term.
Be very aware of black box stuff, we have a lovely black box, we feed it stuff and it tells us stuff.
We must convey a very good understanding of the technology that we are uses.
Project is as much about what kind of insight do we get from using this software, what do we know now that we did not know before?
Where are the challenges and what matters in terms of the performance of something like this.
Do not forget testing: we want as much automated testing as man in the street testing.
Imperial do not own our IP.
Bring prototypes to Jack and to Will regularly.
Even if we just bring a pen and paper mock up of how the system will look, Will doesn't want the first time he sees the software to be when we demo it.
Better that as we go to share with Jack and Will.
Will does want to talk and be involved.
Small frequent meetings that are no more than 15-30 minutes than get completely sideswiped by some giant thing at the end.
This needs to be front loaded. Progress must be made now, if we don't make it now we will end up in trouble.
We should consider doing things in pairs, it can often be more fun.

B.2.7 27/01/2014 16:30

```
Meeting Information
--------------------

*Objective*: Firm up how to operate effectively as a group and set out
    architecture.
*Date*: 27/01/2014
*Location*: DOC Labs
*Chair*: GH
*Minute Taker*: SL
*Attendees*: AC, SD, JM, GH, SL

## 1. Report 1.

> GH: Everyone familiar with the report, let's decide who is doing each
     section. We have copied the titles from the sample report so let's
    model it on that.

  * Introduction done in conclusion. Not yet assigned.
  * Ashley will draw the diagram for the specification. (2.1). Server
     comes from the goal orientated capture.
  * Stewart will double check 2.2, remove the last column of
     dependencies.
  * Priorities plugged into table on Wednesday.
  * Gerard owns section 3.

> AC: Commits should explain behaviour, not just what you have typed.
> JM: One of the issues with Git is that there is no "latest version"
    worth mentioning that we have overcome this. We have a release
    management system.

  * Simon will type up section 4.1 (balancing learning outcome and
     workload required etc).
  * Simon will add comments about scrum master and code integration.
  * Gerard will oversee the Schedule section 4.2 based upon the sprint
     schedule.
  * Ashley will do section 5 App Store.
  * Alex will do section 5 Plant Data.
  * Simon and Stewart will look at Server/Scalability section 5.
  * John will do the Neural Network of section 5.

  ```
          Neural Network (John)
          Plant data (Alex)
          App Store (Ashley)
          Developer account
          Scalability (Simon & Stewart)
                  * Server
                  * Extensibility for large numbers of people
                  * Becomes target for hacker
                  * API issues
          Hardware
                  * Need a specific graphics stack i.e. Nvidia GPU
                  * Storage space for large amounts of data,
                     scalability
                  * High availability server
  ```

  * Naturally splitting software/hardware environment project
     boundaries in section 6. How and why? ie, into groups. Consolidate
```

```
    as team afterward - Wednesday.

> Discussion:
            We want to target general public primarily, and increase
               interest in these things, and as a result, we must
               make the app as easily accessible as possible.

  * Stewart will own this section 6 about stakeholders and produce
    something.

> AC: Natural history museum are interested...

  * Alex will do the state of the market summary.

## 2. Database Structure.

   * John will have a look at the database characteristics and let us
     know what we should record from a taxanomic point of view.

## 3. Decide on two wow-factor items for the project.

   * App store acceptance is a good goal for a wow-factor.
   * Make people download the app, or get HTML5 app on phone, to take
     the picture and get the information back themselves.

> We need to pick some species that we want to identify.

   * Simon will go to the nursery and see if there is any nice looking
     plants.
```

B.2.8 29/01/2014 14:00

```
Meeting Information
-------------------

*Objective*: Firm up how to operate effectively as a group and set out
    architecture.
*Date*: 29/01/2014
*Location*: DOC Labs
*Chair*: GH
*Minute Taker*: SL
*Attendees*: All Group Members

## 4. Report 1.

    * Ashley to increase the size of the diagram on Goal orientated
      capture.
    * John will reword section 2.1 above the diagram.
    * Stewart will add intro paragraph to section 2.2.
    * Simon will make the tables in section 2 smaller font. Aiming for
      one page.
    * Simon will alter the italicisition or bold of Essential/supp to
      aid clarity.
    * Stewart will re-word image manipulation words in Plant
      Classification 2.2.2.
    * We will all look at and consider the wording of that section.
    * Gerard will update the VC part in 3.4 with a short intro covering
      Git.
    * John will try and do a flow chart to replace/supplement text in
      3.4.1
    * John add supplemental requirement to 2.2 detailing multiple image
      input.
    * John will try a rewrite on 5.2 and 5.1 to squeeze ML and Plant
      Data.
    * Stewart to make change to 5.4 regarding using commercial service
      to overcome rapid growth.
    * Simon will tweak the mongoDB section to make it scale properly.
    * Simon will work out how to get smaller ++ in section 6.2.1
    * Add into 6.2.4 Hardware regarding server being hosted on Doc VM
      etc.
    * 6.4.2 will be updated after the NHM meeting.
    * 6.4.3 Alex will link intro three points and condense paragraph.

## 1. Unit testing frameworks

    * Agreed Google C++ testing may be framework for CUDA stuff.
      Possibly as external library we do not need to test.
    * Python unit test framework to be used for ML Python stuff.
    * Stewart and Simon to compare between Mocha and nodeunit. Mocha is
      current preferred.
    * Ashley wants to use xCodeBuild to do unit testing. Ashley and
      Gerard will discuss further on Unit Testing.

## 2. Logging

    * Investigate logging from Node.js, others to consider how/why?


## 3. Multiple environments

    * Gerard: must have things set up so we can run each component with
```

```
    standard tests. Also start and stop scripts for standard and
    production part of the system.

## 5. Renewed focus for project as a whole

  * Everyone agrees to work backwards from demo, to produce quality
    project.

## 6. Limited Company?

  * Simon will look at app store t+c and process of setting up limited
    company.
```

B.2.9 30/01/2014 15:00

```
Meeting Information
--------------------

*Objective*: Firm up content of Report 1 with Jack
*Date*: 30/01/2014
*Location*: DOC Labs
*Chair*: JK
*Minute Taker*: SL
*Attendees*: All Group Members and Jack Kelly

## 1. Report 1

> JK Suggests we do not make hard claim to recognise something in
    natural environment, great goal but might not prove possible and NHM
     were concerned.

  * Add dates of planned completion date of each sprint and what will
     be delivered (but is this un-Agile).

  * Putting in as a seperate task defining interfaces between app and
     server and server and classifier. Use the word Interface!!!

> JK queried Daily Scrum but GH shot him down.

  * Do not put in report but if we are writing code that is speculative
     (i.e, testing out an idea) then for the sake of time, do not
     bother writing test for that idea at the same time.

  * Insert line that says "the group will only check into DEV if the
     appropriate unit test has been written"

  * Add to the report that the NHM really like us and want to work with
     us.

  * Hammer home that we are really addressing some serious issues.

## 2. Training data and outcome from NHM meeting.

   > JK, PlantCLEF data set is actually quite small.

   * Let's find out exactly what is in imagenet.

   > JK, If we can't find the data to do what we need to do, then it
      would be justifiable to to change direction.

   > JK, What would be bad is if we had to spend a lot of time cleaning
      the training data. We don't want to be dealing with dirt.

   * Make sure we choose data sets that we have to clean the least.

   > JK, NHM think class is really, really specific on what something
      is. Can't we just say "Orchid".

   > John is trying to find decent taxonomy tree that gives us a
      standard naming convention at the level of the tree that we want.

   * Should we come up with set of 40 classes for general public (ie,
      tomato, potato, bla bla).
```

* Could we then consider the NHM orchid issue and show it has application for researchers etc.

* One task is to work backwards from precise names to bucket at whatever level of the tree that we want.

* Take a look at DBPedia to go from Latin names to principled class.

* Potentially scrape flickr to get distrubution of tags for top 100 plant names.

* BIG QUESTION: What level of granularity -> where is the boundary of our system working well and not working well.

B.2.10 31/01/2014 10:00

```
Meeting Information
--------------------

*Objective*: Decide on paramaters for server/app interface.
*Date*: 30/01/2014
*Location*: Library Cafe
*Chair*: SL
*Minute Taker*: SL
*Attendees*: SL, SD, AC

### Server/Graphic Interface

 * Implement request handler on graphic interface that returns sample .
     json object to server POST request.

 * Implement .json responses for:

   * No classification.
   * Error during classification.
   * Success, include payload.
   * Timeout/Unresponsive.
   * Massive delay in response.


 > Ashley was concerned at speed problems using POST request between
     graphic and server interface.


 #### We settled on the following implementation:

 * APP -> SERVER "POST" IMAGE DATA
 * SERVER -> APP RESPOND WITH JOBNO....save job no in DB as ""...upon
     start up, process all jobs that havnt been sent to graphic02 for
     classification
 * SERVER -> GRAPHIC INTERFACE "POST" IMAGE DATA and job no.
 * GRAPHIC INTERFACE -> update db, "received job no", CLASSIFIER "EXEC"
      CLASSIFICATION....later implmentation: re-run failed jobs. later:
     queuing/triggering multiple classifications. Can multiple
     classifications run concurrently???
 * GRAPHIC INTERFACE -> DATABASE ADD JOBNO WITH CLASSIFICATION DATA
 * APP -> SERVER "GET" CLASSIFICATION DATA
 * SERVER -> DATABASE ON GRAPHIC02 GIVE ME ENTRY FOR JOBNO XX
 * DATABASE -> SERVER HERE'S A JSON WITH YOUR FEEDBACK
 * SERVER -> APP, HERE'S A JSON WITH YOUR FEEDBACK....server logic to
     handle un-classified results.

* RESTful connections...between MongoDB and Node
```

B.2.11 17/02/2014 13:00

```
Meeting Information
--------------------

*Objective*: Discuss Sprint 1 outcome, decide goals for Sprint 2.
*Date*: 17/02/2014
*Location*: DOC Meeting Room
*Chair*: GH
*Minute Taker*: SL
*Attendees*: AC, SD, JM, GH, SL

### 1. Progress made on Sprint 1 by individuals.

 > Ger finished the scripting for json/xml.
 > Ash finished the app actually being able to do a full round trip
     query for an image being sent to classification. Improved internal
     queueing.
 > Stewart had the classification script get run off.
 > John finished the classification script actually doing the
     classification and providing our expected input.
 > Alex has 2000 folders of between 1 and 2000 images per folder (
     average about 530) jpg. Alex has to figure out which Synset-ID
     corresponds to plant, so we can extract the correct images.


 > Ger: Mainly machine learning stuff left, ran through the outstanding
      stuff.

 > We reviewed the backlog and which items are due for Sprint 1 still.

 > Sprint 2 -> Focus will be on the unit testing.

 > Ger ran through the backlog that has been assigned for Sprint 2.

 > Stressed point about making notes about issues we encounter and how
     we overcame them.


### 2. Issues encountered.

  * APP -> SERVER POST IMAGE DATA

### 3. Arrange which backlog items will form work for Sprint 2.
```

B.2.12 20/02/2014 15:00

```
Meeting Information
_____________________

*Objective*: Update supervisors on progress on App.
*Date*: 20/02/2014
*Location*: DOC Meeting Room
*Chair*: JK
*Minute Taker*: SL
*Attendees*: All + Will + Jack

### 1. Progress made on Sprint 1 by individuals.

If we can get backend up it will be awesome, otherwise frontend
   templating to guide user.

Some convincing demonstrations.

Actually bring live plants from various angles into the demo.
```

B.2.13 26/02/2014 14:00

Key points:

Simple -> Tutorial or Self-explanatory
Engaging
High Contrast
Large 'Hit' Areas
OFFLINE vs. ONLINE
CLASSIFY vs. BROWSE

APP ICON:

http://www.freevector.com/plant-silhouette-graphics/

Leaf + Question Mark combination - or PLANT TAG. <=====}

We can tie this into the actual running of the app by overlaying the tag if classified.

HOME SCREEN:

Specified by apple to be similar to app home screen.

FLOW OF APP:

No review stage, built into the App as they are taking images of each segment.

Always return to HOME PAGE after submitting the photo.
wa

B.2.14 10/03/2014 13:00

```
Meeting Information
-------------------

*Objective*: set out goals for Report2
*Date*: 10/03/2014
*Location*: H-Bar
*Chair*: GH
*Minute Taker*: SL
*Attendees*: All

Work by section:

1. Abstract: leave to end, deal with it when we come to it.

2. Project Progress

2.1 Gerard Takes this section

2.2 Renamed State -> Current Status - Stewart will rewrite State.

2.2.1-3 The tables:

Frontend:

        Still keep Wikipedia in progress.
        Cancel Comparison image displayed.
        Cancel User feedback of result to server.

        -> Revised goals add to table with a +

Plant Classification:

        Result produced within acceptable time interval -> Complete.
        Neural network capable of processing multiple images for a
            single plant. -> Complete.

Additions:

        Bucketing Algorithm
        Tagging ImageNet data (plant classification).


Server:

Add remark splitting Server into Request Server and Worker Server.

2.3:

        Everyone to comment on their revisions and experiences.

2.3.1:
        Server split into Request Server (for App) and Worker Server

2.3.2 Revised Schedule -> Gerard will give overview.

2.4 Make sure to incorporate areas that caused problems and revisions
    to specification into the revised goals section.

3. Testing
```

Make sure we are hitting points in Reuben's testing presentation.

3.1 Server testing to be looked at again.

B.2.15 12/03/2014 14:00

```
Meeting Information
--------------------

*Objective*: set out goals for Report2
*Date*: 14/03/2014
*Location*: H-Bar
*Chair*: GH
*Minute Taker*: SL
*Attendees*: All

Work by section:

1. Abstract in seperate smaller section perhaps?

*

2.
2.1

Adopt iOS 7 application throughout.

Section 2.3.2.
* Simon to do diagram and Stewart to summarise in one short paragraph (
    multi-queue etc.)


2.3.3
* John to check over rewrite.

3.0
* Gerard will take overview section and hit the necessary Reuben points
    .

3.1

* Ashley to add remarks on Grey box and partition testing.

3.2

* SIMON Add remark on simulating mongo errors a problem.

3.3

* JOHN add explanation on post-processing coverage

3.4

* Ask Jack what the correct means of combination of overall code
    coverage is. - We think the mean

* Check that we have consistent language capitalisation.

4.0

* Everyone add screenshots of their coverage.

* Simon look at hours on VersionOne and try and improve the Burndown.
```

B.2.16 25/03/2014 17:00

Meeting Information

Objective: set out goals for Report2
Date: 25/03/2014
Location: H–Bar
Chair: GH
Minute Taker: SL
Attendees: All

GH:

Following Friday's meetings, lets discuss the positive outcome and where we want to go.
Over Easter we have 5 exams to study for, so what do we think is achievable and what is a priority.
Consider report and what we can do to get the wheels in motion.

Focus on presentation.
Make sure that we can compare between us and google goggles and imagenet.

- Visit Kew next week (Simon, Ashley and Stewart)
- Get a structure of Report 3 together (Ashley will put the barebones up).
- Start adding content to this report as soon as possible then we have a week after exams to finalise and do presentation.
- Presentation to be sorted at the time of the report.
- Shoot some footage at Kew next week.
- Make a shortlist of things that it is good at. John – build script to see how good it is, and do top5 error script.
- Submit but don't release App until the morning of the demonstration. Ashley.
- Make the leaf green. Change the view so it is a bit more sexy. Ashley.
- Stewart fix the completion bug.
- Alex fix data provider bug.
- Train nets for as long as necessary.
- Gerard tell NHM about our progress.
- Keep scrumming once a week.
- Add the description to the stdout.

B.2.17 12/05/2014 10:00

Meeting Information

Objective: set out goals for Report3
Date: 12/05/2014
Location: H–Bar
Chair: GH
Minute Taker: SL
Attendees: All

- Address third party code in the report and the limitations on portability.
- Note the interest from people out in the real world.
- Talk about commercials.
- Slide in presentation re monetising the product.

1. Everyone to do their section's readme.
2. Introduction to be done last.
3. Specification – c/p Jack's original spec, include quotes and facts from Jack's spec, then present out initial table from Report 1. Stakeholders agreement etc. ASHLEY.
4. Seperate Design and Architecture into two different sections for UI design and System Architecture.
5. Gerard to do bucketing pseudo code and the scrum section etc.
6. Stewart to do Architecture section.
7. Ashley to do UI design section.
8. Scalability (issues) – put in in the final product and further development.
9. Scalability (good stuff) – include in design.
10. Each person will do their own section within the implementation.
11. Simon to add to the Testing section and comparison vs spec.
12. John and Alex to look at inter vs intra class variation using his prediction suite.
13. Everyone do bullet points about further development.
14. Everyone to add any sections to the google spreadsheet that is relevant to the project.
15. Everyone kill their dead branches.
16. Gerard to push to master.

B.2.18 13/05/2014 10:00

Meeting Information

Objective: Review early draft of Report3
Date: 13/05/2014
Location: H-Bar
Chair: GH
Minute Taker: SL
Attendees: All

- High level timeline, key milestones.
- Make point that NHM told us not to trust user classification.
- Make point about failure to allow server to send to multiple classification servers.
- Overcoming the issue of identifying individuals and thus using group IDs to identify people.
- Add PUT to diagram.
- GROUPING as challenge under servers
- Write intro to the server thing about choice of stack
- PNG to be fixed by John.
- Everyone individually update testing.
- Gerard will do bucketing pseudo code.
- Git commit history (pretty for log) one way or the other. Pretty things to break up the section.

B.3 Git Commit Logs

```
50340e3 Stewart Douglas 2014-05-15      Completing merge
ecb9548 Stewart Douglas 2014-05-15      Renaming pollv2.js -> poll.js
64cfcd0 Ashley Cutmore  2014-05-14      iOS: README added
42e4a3b Simon Leigh     2014-05-14      tidied README
788ed04 Simon Leigh     2014-05-14      Merge branch 'dev' of gitlab.doc.ic.ac.uk:bjm113/group-project-master into dev
f7216e0 Simon Leigh     2014-05-14      finished readme for servers
83ef21a Ashley Cutmore  2014-05-14      Merge remote-tracking branch 'origin/dev' into dev
ff9f8a8 Ashley Cutmore  2014-05-14      iOS: updated network unit tests
6a59416 Stewart Douglas 2014-05-14      Merging Simon's updates
7d2017c Stewart Douglas 2014-05-14      Additional edits
010e943 Simon Leigh     2014-05-14      tidied test file
f5876f2 Simon Leigh     2014-05-14      testing updates to test logic and db
bc619e1 Simon Leigh     2014-05-14      added a test to increase coverage
d430bac Simon Leigh     2014-05-14      readme, test rationalisation, cleanup of deprecated files
6d10875 Simon Leigh     2014-05-14      Merge branch 'dev' of gitlab.doc.ic.ac.uk:bjm113/group-project-master into dev
e8afef4 Simon Leigh     2014-05-14      updated tests, fixed makefile to allow app_test
0a3f681 Stewart Douglas 2014-05-14      Concluding merge
9ae5615 Ashley Cutmore  2014-05-14      Merge remote-tracking branch 'origin/dev' into dev
f18e60a Ashley Cutmore  2014-05-14      iOS: Updated binary certificates
5b08f62 Stewart Douglas 2014-05-14      pollv2.js now appears to be working correctly
64e11ed John McCormac   2014-05-14      Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master
    into dev
db25af2 John McCormac   2014-05-14      Updated readme with some additional information
67cfe4b Simon Leigh     2014-05-14      Merge branch 'dev' of gitlab.doc.ic.ac.uk:bjm113/group-project-master into dev
8ee53f2 Simon Leigh     2014-05-14      VERY IMPORTANT COMMIT. Froze versions in dependencies to ensure compatability
    with our code.
3e20938 alex    2014-05-14      merging ML/README.md changes with other stuff up there Merge branch 'dev' of https://
    gitlab.doc.ic.ac.uk/bjm113/group-project-master into dev
0debf11 alex    2014-05-14      added to README
1747a47 Simon Leigh     2014-05-14      fixing tests
54e433a Simon Leigh     2014-05-14      fixing config files
5da3e4b John McCormac   2014-05-14      Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master
    into dev
08c14b9 John McCormac   2014-05-14      Updated tests, started README
cf238ee Stewart Douglas 2014-05-13      Implemented promises
d7598cf Stewart Douglas 2014-05-13      Merging
4329dd9 Stewart Douglas 2014-05-13      Additional loop to pollv2.js
36ac9d8 Stewart Douglas 2014-05-12      update runclient.py
```

```
6841bdf Stewart Douglas 2014-05-12      Stewart and Ashs' attempt to fix server
424074e Stewart Douglas 2014-05-12      Commiting changes in order to merge them
8edf612 Stewart Douglas 2014-05-12      Edited pollv2.js
f5bcbfb John McCormac   2014-05-12      Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master
    into dev
3bb5daf John McCormac   2014-05-12      Runclient now exits with sys exit code
8334643 John McCormac   2014-05-12      Modified python server error returns
95289f5 Simon Leigh     2014-05-12      next test update
24d81a2 Simon Leigh     2014-05-12      next test update
cf4d30d Simon Leigh     2014-05-12      next test update
a02f13f Simon Leigh     2014-05-12      updated test to have extant groupId
08feb2f Simon Leigh     2014-05-12      fixed config files
888592f Simon Leigh     2014-05-12      updated configs to use hostnames
ca148ff Stewart Douglas 2014-05-12      New pollv2.js file
76512d3 Stewart Douglas 2014-05-11      New polling file, pollv2.js
0e007db Ashley Cutmore  2014-05-07      Merge remote-tracking branch 'origin/dev' into dev
99fc0ba Ashley Cutmore  2014-05-07      iOS: Deleting an item will stop further attempts to request it s classification
30d3fab Simon Leigh     2014-05-07      updating vm dev conf to new ips
e69b44c Stewart Douglas 2014-05-07      Merging changes - I updated the names of the python run scripts to the latest
    versions
fddf0cb Stewart Douglas 2014-05-07      Updated files with latest python script names
3efbafa Ashley Cutmore  2014-05-06      iOS: camera view - user can swipe to change selected segment
6e0d76c Stewart Douglas 2014-05-02      Merging John's changes
a03551f John McCormac   2014-05-01      Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master
    into dev
cbf04a9 John McCormac   2014-05-01      Client now works for multi-view inputs
f356554 Ashley Cutmore  2014-04-25      iOS: Changed to lighter green tint. Fixed result view image scrolling bug. Trying
    to take a picture when running on a device without a camera no longer causes a crash.
aa43687 Ashley Cutmore  2014-04-19      iOS: Green Tint applied throughout
a65699a John McCormac   2014-04-16      Corrected test_many logical breaks
672b2d2 John McCormac   2014-04-16      Fixed dataprovider bug and printout for new test_many
33886e9 John McCormac   2014-04-15      Fixed runserver config issues
6fa6973 John McCormac   2014-04-15      Added python socket and server to process jobs
eb72dd6 John McCormac   2014-04-15      Multiview testing now in run.py
81fda15 John McCormac   2014-04-15      Predict.py now gives top-1 and top-5 by class, WITH multiview. Test-batch has new
    test-many function. Dataprovider uses random steps
253a931 Ashley Cutmore  2014-04-02      iOS: Sends Completion=False to server if user cancels after uploading an
    image
c4b93c5 Ashley Cutmore  2014-04-02      iOS: camera preview size increased
bebf55b HoldenCaulfieldICL      2014-04-02      modified plantdataproviders to perform 4x4 crops per image
```

```
357511e HoldenCaulfieldICL      2014-04-02      committing before merge
44fc274 HoldenCaulfieldICL      2014-04-02      addingdatasetNoMongo.py
045ca2b alex    2014-04-02      adding layers definition file
b3a4fab HoldenCaulfieldICL      2014-04-01      modified layers file for new, smaller architecture
3e3c27e HoldenCaulfieldICL      2014-04-01      merging with changes made to local laptop for new crop_step param
bdba8b4 alex    2014-04-02      modified options file to accept new crop_step parameter
9d846ef HoldenCaulfieldICL      2014-04-01      merging with changes made to local laptop for new crop_step param
fe43d7a alex    2014-04-02      modified convnet.py to accept new crop_step parameter
f70f800 HoldenCaulfieldICL      2014-04-01      fixed mistake in options_01-04-2014: forgot to change path to params file
faf1ae1 alex    2014-04-02      adding params file for smaller net
63daaeb alex    2014-04-01      added new options.cfg file, implemented crop_step field to data provider
824c5ec alex    2014-04-01      adding test images
a5bc79c alex    2014-03-31      patch_idx now changes after having gone through all batches, i.e. after an epoch
ebfe817 John McCormac   2014-03-30      Added one big net plant classes
cf828cf John McCormac   2014-03-30      MongoClient no longer initialized on helperfunction import
b8d10a1 HoldenCaulfieldICL      2014-03-28      data provider working
93ec452 HoldenCaulfieldICL      2014-03-28      working on data provider bug
62c9a8d HoldenCaulfieldICL      2014-03-27      other ccn-train output file to better understand bug
2119e7e Ashley Cutmore  2014-03-27      iOS: Database schema change (will migrate to this version automatically). Now has
    ability to delete specimen by swiping them on the homepage.
cf956cb Ashley Cutmore  2014-03-24      iOS: Modified camera-view buttons to be more verbose
e787184 Ashley Cutmore  2014-03-24      iOS: 1. web view indicates loading activity  2. webview respects height
    restriction of status bar
e86d5d9 Ashley Cutmore  2014-03-23      Webview added. Clicking a result brings up a google image search for that plant
5897721 Ashley Cutmore  2014-03-22      iOS: Icons Added
d5a1289 Simon Leigh     2014-03-22      Merge branch 'dev' of gitlab.doc.ic.ac.uk:bjm113/group-project-master into dev
3023fd2 Simon Leigh     2014-03-22      added ability to force classification on stuck jobs by resetting their statuses
    in group and segment collections
643ca7b Ashley Cutmore  2014-03-22      iOS: camera view    >    button cycles through segments until the end, then ends
    session
b660072 Ashley Cutmore  2014-03-21      iOS: activity indicator replaces   processing   on home page
1ab0c8c Ashley Cutmore  2014-03-21      Spelling Correction on App
3460e61 Ashley Cutmore  2014-03-21      camera view hints to save image by selecting another segment
e1cdbf2 Ashley Cutmore  2014-03-20      Merge remote-tracking branch 'origin/dev' into dev
8575319 Ashley Cutmore  2014-03-20      1.Displays results in order. 2. UI updates forced to operate on main thread (
    hopefull bug fix)
82d8393 Ashley Cutmore  2014-03-19      Result View scrolls and stays put
2824674 Simon Leigh     2014-03-19      fixing inconsitent group_id cast - now always ObjectID(bla)
3ca335b Ashley Cutmore  2014-03-19      Merge branch 'iOS_resultView' into dev
fb37a6f Ashley Cutmore  2014-03-19      Much Plant!
```

```
e654f57 Stewart Douglas 2014-03-19 Merging in John's change to combine.py
3e43fcc Stewart Douglas 2014-03-19 Modified group_ids
cc11803 John McCormac   2014-03-19 Changed combine json output
065ed45 Ashley Cutmore  2014-03-19   Beleaf Level  added to result view. Results now stored as float.
ca81ed6 Ashley Cutmore  2014-03-18 Branch to work on new result page
6ac38bc Stewart Douglas 2014-03-18 Merging into dev
8cbf1dc Stewart Douglas 2014-03-18 Both individual image analysis and bayesian aggregate now appear to be working
3433dd4 Ashley Cutmore  2014-03-18 Merge remote-tracking branch 'origin/dev' into dev
d124ae3 Ashley Cutmore  2014-03-18 Added more feedback mechanisms to camera view
72b0f34 Simon Leigh     2014-03-18 added new tests and increased robustness of methods to unintended input
847e6b2 Simon Leigh     2014-03-18 pushing fixed PUT method
a8f0d21 Ashley Cutmore  2014-03-18 Merge remote-tracking branch 'origin/dev' into dev
bc80a73 Ashley Cutmore  2014-03-18 Added new portable BuildTarget
569a336 Simon Leigh     2014-03-18 Merge branch 'dev' of gitlab.doc.ic.ac.uk:bjm113/group-project-master into dev
b9ee22d Simon Leigh     2014-03-18 added PUT method and tests
2e07b1e Ashley Cutmore  2014-03-18 Merge remote-tracking branch 'origin/dev' into dev
d350015 Ashley Cutmore  2014-03-18 Renamed completionPUT expected result field
108d6e3 Stewart Douglas 2014-03-18 Corrected combine script
07bfe4b Stewart Douglas 2014-03-18 Added data2 config parser
b74a88b Stewart Douglas 2014-03-18 Merging graphicGroupClassification int dev
12bca16 Ashley Cutmore  2014-03-17 Merge remote-tracking branch 'origin/dev' into dev
2965f41 Ashley Cutmore  2014-03-17 Update cameraView interface
e21f51f John McCormac   2014-03-17 Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master
    into dev
091378b John McCormac   2014-03-17 Added cudaconv binary for server use
244b98a Ashley Cutmore  2014-03-17 Crop-complete
0ee4086 Ashley Cutmore  2014-03-17 cropMethod added
57eece8 Ashley Cutmore  2014-03-17 Now possible to change serverURL in settings without re-compiling
e2952db Stewart Douglas 2014-03-17 Updated graphic_filepath, now exec-ing classification script
76ef397 Stewart Douglas 2014-03-17 Updated routes/index.js so that the absolute filepath to the image is save
70d502e Stewart Douglas 2014-03-16 Added logic to poll.js for combine.py
a1769a3 Stewart Douglas 2014-03-16 Added additional whilst loop to poll.js
5835718 Stewart Douglas 2014-03-16 Incrementing classified count after .pickle file generated
c40f2af Stewart Douglas 2014-03-16 Added exec() to poll.js
bb34493 Stewart Douglas 2014-03-16 Further work on poll.js
3f0be9c Stewart Douglas 2014-03-16 Implemented async.forever
03029c5 Stewart Douglas 2014-03-16 Inluced async.whilst in poll.js
4a94ba9 Stewart Douglas 2014-03-16 Modified poll.js
73f6084 Stewart Douglas 2014-03-16 Graphic can now accept posts while polling
6e23173 Stewart Douglas 2014-03-15 Adding poll.js
```

```
bf4a7d4 Stewart Douglas 2014-03-15      Further changes for group classification
08c7bfa Ashley Cutmore  2014-03-15      Merge branch 'ios_UIdesign2' into dev
e21fd8a Ashley Cutmore  2014-03-15      Completion PUTs not part of automatic update queue processing
ff623a8 Stewart Douglas 2014-03-15      testing group classification
e53cfd3 Ashley Cutmore  2014-03-15      Added notified attribute to DB
ae5baa3 John McCormac   2014-03-14      Adding output from running test training
6da2202 John McCormac   2014-03-14      Corrected typo
92eca1f John McCormac   2014-03-14      Added new dataprovider import to convnet
0642b73 alex    2014-03-14      what's left
515aeb2 alex    2014-03-14      got good coverage
e64c087 alex    2014-03-14      more test coverage
e91ea61 alex    2014-03-14      merging my work on graphic 02 with work on laptop
57096a4 alex    2014-03-14      about to properly branch out for augmentedleaf dataprov
2c1b0c9 HoldenCaulfieldICL      2014-03-14      about to branch out for ccn-train with augmentedleaf
e2ab92e HoldenCaulfieldICL      2014-03-14      merging
f955c6a alex    2014-03-14      forgot to reset test
b3c040a HoldenCaulfieldICL      2014-03-14      merging
a42cee1 alex    2014-03-14      test_dataprovider working depsite change in directory
fa80259 alex    2014-03-14      data_dir moved, hopefully other tests will work fine now
82a970e alex    2014-03-14      using data_dir for flex
a0d73ed alex    2014-03-14      but pca doesn't seem to be doing much
6e9a143 alex    2014-03-14      try saving some images
756f58f HoldenCaulfieldICL      2014-03-14      ready to merge
895f85a alex    2014-03-14      modified run_tests.py for it delete previous results if called again
1e45d8b alex    2014-03-14      pca integrated successfully.
18b3eb3 alex    2014-03-14      pca working well, merging
ee1a7e4 alex    2014-03-14      pca implemented correctly. still need to manually look at a few images to make sure work
    done properly
cc4ec2e alex    2014-03-14      testing merge ours
7cad9e6 alex    2014-03-14      Merge branch 'testours' into dev
d23159d alex    2014-03-14      testing ours again
ee49ca5 alex    2014-03-14      testing merge -X ours
bcf3435 alex    2014-03-14      getting there more
0c78416 alex    2014-03-14      getting there
6f03ac0 HoldenCaulfieldICL      2014-03-14      actually, pca all the time
a65210d alex    2014-03-14      voici un autre test
b3226d7 alex    2014-03-14      will now start committing in french
3876975 alex    2014-03-14      ALLEZ JE SUIS TROP FORT
f9a92ce alex    2014-03-14      test_dataprovider not doing checking crop dimensions properly: added (although a little
    ambiguity, mentioned in comments, remains)
```

```
cb9e539 alex     2014-03-14      double check dimensionality of pickled data
52a500d alex     2014-03-14      yes! now incrementing column-first
a914d65 alex     2014-03-14      yes! now incrementing column-first
48e3fa3 alex     2014-03-14      going to switch from row-first incrementation to column-first incrementation. all working
    well here
095f561 alex     2014-03-14      working well. merge this back into dev. noticed patches are incrementing first down then
   right. want to change that
d3ff327 alex     2014-03-14      yes, this is better
8186e44 alex     2014-03-14      test that data augmentation is 2048-fold: this might kill my RAM
0da9735 alex     2014-03-14      fixed mistake in jpg_to_pickle: was missing numpy vstack
02544e6 alex     2014-03-14      jpg_to_pickle works, have a 1-img dummy batch
c365b70 alex     2014-03-14      created mongo-independent datasetNoMongo.py for testing purposes
d83d371 alex     2014-03-14      fixing typo in dataset
9969bbd alex     2014-03-14      wrote jpg_to_pickle
8f9b2be HoldenCaulfieldICL       2014-03-14      adding image to practice on
26b4113 HoldenCaulfieldICL       2014-03-14      Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-
   master into dev
14e36ec HoldenCaulfieldICL       2014-03-14      adding jpg_to_pickle
fd41eac John McCormac    2014-03-13      Corrected layers number of classes
44a0eb7 John McCormac    2014-03-13      Added ensemble network
6e892f7 John McCormac    2014-03-13      Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master
   into dev
7a05ff3 Stewart Douglas 2014-03-13       Changed db port in graphicConfigTest.js
84ef5c4 Stewart Douglas 2014-03-13       Merging into dev
14d2c79 Stewart Douglas 2014-03-13       Adding further coverage files
13d8992 Stewart Douglas 2014-03-13       Update to code coverage and routes/index.js
3a2c14a HoldenCaulfieldICL       2014-03-13      adding pickle_to_jpg script
c58750e John McCormac    2014-03-13      BUG fix explicitly cast to float for component threshold
e90314d Ashley Cutmore   2014-03-13      Added PUT request constructor method
7d3b657 HoldenCaulfieldICL       2014-03-13      1img batch created
b0d90ba HoldenCaulfieldICL       2014-03-13      merging script to create 1 image dummy batch with remote changes
8810c46 HoldenCaulfieldICL       2014-03-13      1img batch created
a6b13cf Simon Leigh      2014-03-13      removing spurious js test
63e0f1f HoldenCaulfieldICL       2014-03-13      working on creatign single image batch for in depth patch testing
7f008fa HoldenCaulfieldICL       2014-03-13      modified run_tests.py so it can create test_results file (needed to mkdir
   )
a9ef3d5 HoldenCaulfieldICL       2014-03-13      ran 'python run_tests.py -coverage' but error could not create
   test_results file
18079fb alex     2014-03-13      (forgot to add --all) adding data providers in separate file
a235a0a alex     2014-03-13      merging platndataproviders with other work Merge branch 'dev' of https://gitlab.doc.ic.ac
```

```
    .uk/bjm113/group-project-master into dev
2718af8 alex     2014-03-13      adding plantdataproviders
fb4e67b Stewart Douglas 2014-03-13      Merge branch 'graphicGroupClassification' of https://gitlab.doc.ic.ac.uk/bjm113/
    group-project-master into graphicGroupClassification
37224d3 Stewart Douglas 2014-03-13      Minor modifications to graphic.js
046ca9e Stewart Douglas 2014-03-13      Initial unit tests for GraphicServer/routes/index.js
a886dec Stewart Douglas 2014-03-12      add graphicRoutesTest.js
43e9de7 John McCormac   2014-03-12      Added additional tests for combine, refactored combine code
25fe150 John McCormac   2014-03-12      Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master
    into dev
dc59424 John McCormac   2014-03-12      Show net now SAVES images, runtests outputs html
843d3f8 alex     2014-03-12      Merging Alex's changes to dataprovider testing with someone else's changes Merge branch '
    dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master into dev
78a36fd alex     2014-03-12      writing more dataprov tests
365e5f0 Stewart Douglas 2014-03-12      Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master
    into dev
e68aa1a Stewart Douglas 2014-03-12      Now get 100% coverage for graphic config parser
4bf88da alex     2014-03-12      merging with my local changes to ML
afa587a alex     2014-03-12      modified test_dataproviders for them to conform
82faff7 Gerard Haughian 2014-03-11      bucketing issue fixed
324a393 Ashley Cutmore  2014-03-11      BugFix: Previously taken segment can be viewed again when re-selected. Capture
    Session shares Database with UI
b28e3f3 Ashley Cutmore  2014-03-11      Segment Capture Logic moved into separate class and tested. Uploads process in
    the background during capture.
f156f78 John McCormac   2014-03-11      Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master
    into dev
1ed99f5 John McCormac   2014-03-11      Improved mean calculation for run script
ffd43f0 Ashley Cutmore  2014-03-11      Added link to network testing library and podfile for installing. Modified
    database class to disable saving during testing.
e2f38c6 alex     2014-03-11      AugmentLeafDataProvider working
4ff2161 alex     2014-03-11      yes, I think we're good now
b3589ad alex     2014-03-10      realised datadic object used in AugmentLeadDataProvider is for when loading entire
    dataset. Need alternative
6cac5c9 alex     2014-03-10      now need to create dummy batches.meta
37228da alex     2014-03-10      issue when returning list
7ab7a2d alex     2014-03-10      fixed merge conflict in dataset.py: was a typo
595237c alex     2014-03-10      working on data provider
9df9386 Stewart Douglas 2014-03-09      Graphic server now adds uploaded images to their group folder in uploads/<
    database>
1878546 John McCormac   2014-03-09      Improved combination logic
```

```
11c49a6 John McCormac    2014-03-09      Corrected path lookup
daf9451 John McCormac    2014-03-09      Silence output from tests unless -verbose flag added
0454429 John McCormac    2014-03-09      Added additional test_tags test
128d30c John McCormac    2014-03-09      Additional tests
991523b John McCormac    2014-03-09      Added tests for all scripts
38bc710 John McCormac    2014-03-08      Basic tests for run script
92ca2da John McCormac    2014-03-08      Update tagging net
87b6a45 John McCormac    2014-03-08      Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master
    into dev
936e4dd John McCormac    2014-03-08      Added batcher tests
1f71a30 Stewart Douglas 2014-03-08       routes/index.js now creates the relevant group folder and updates mongo's
    submission_state field
7182e71 Simon Leigh      2014-03-08      added more tests
a83cead Stewart Douglas 2014-03-08       Updates to routes/index.js
e887aec Stewart Douglas 2014-03-08       Bug fixes for routes/index.js
c430f07 Simon Leigh      2014-03-08      Merge branch 'appServerS3' into dev
39db7b5 Simon Leigh      2014-03-08      fixed tests and code so everything passes again
e3cacb1 Simon Leigh      2014-03-08      Merge branch 'appServerS3' into dev
68a3fa2 Simon Leigh      2014-03-08      minor fix to graphicserver on app branch to allow testing
7eabac2 Simon Leigh      2014-03-08      implemented changes to collections and grouping of all images
ff443c4 Stewart Douglas 2014-03-08       Modified routes/index.js to accept group_id field from app server
4169f9d John McCormac    2014-03-08      Added tag tests, cleared old results
df94b2e John McCormac    2014-03-08      Added combination labels to dataset.py
4d6f7e9 John McCormac    2014-03-07      Parsed labeled output, gives short version
bacbb69 John McCormac    2014-03-07      Setup combine script
258ce5a John McCormac    2014-03-07      Added combine script
b0612e3 John McCormac    2014-03-07      Small correction 'tag' reference in mongoHelperFunction
8f7a028 Simon Leigh      2014-03-07      added TODO document specifying draft 1 database schema + formatting correction
8f5aec2 Simon Leigh      2014-03-07      added TODO document specifying draft 1 database schema
eaac8f1 Simon Leigh      2014-03-07      actual fix to graphic crash
139c2f4 Simon Leigh      2014-03-07      fixed form parsing variable name to make server work
9fd3d60 Simon Leigh      2014-03-07      added groupID to test file
a7ccaaa Gerard Haughian 2014-03-07       bucketing code coverage now 91%
376a357 Gerard Haughian 2014-03-06       fixed typo in bucketing script...new code coverage report added,
15d18d2 Ashley Cutmore   2014-03-06     Home Screen shows image thumbnail and current status. Camera view flashes white
    when a photo is taken. Network automatically retires after timeout.
f95b99c Gerard Haughian 2014-03-06       bucketing script accidentally had function call commented out
e2f99e7 Gerard Haughian 2014-03-06       added bucketing unit tests
3a29442 Gerard Haughian 2014-03-06       unit tests for bucketing added
6501219 John McCormac    2014-03-06      Added test result directory
```

```
7370e7a John McCormac  2014-03-06  Added sys path test
6167f31 John McCormac  2014-03-06  Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master
    into dev
12c5c62 John McCormac  2014-03-06  Added bucketing test script
2d517eb Simon Leigh    2014-03-06  fixed missing dependency in GraphicServer causing crash
6d64dd0 Simon Leigh    2014-03-06  Merge branch 'appServerTesting' into dev
ba3302f Simon Leigh    2014-03-06  inserting group ids now, so we have groups for each set of images
40aee7a Ashley Cutmore 2014-03-06  Now able to upload multiple images in one session.
8c111a7 Ashley Cutmore 2014-03-06  Networks Up! Update Queuing no longer enters infinite loop, result shown.
    ServerInterface test coverage 80%.
0b6bb67 Gerard Haughian 2014-03-05  native mongo queries not compatible with node/mocha...
d5b2902 Simon Leigh    2014-03-05  Merge branch 'appServerTesting' into dev
631e703 Simon Leigh    2014-03-05  Coverage of AppServer now touches all files with our code
a77e1d1 Simon Leigh    2014-03-05  Merge branch 'appServerTesting' into dev
82cd972 Simon Leigh    2014-03-05  fixed routes for prod
cf65749 John McCormac  2014-03-05  Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master
    into dev
26dd5b7 John McCormac  2014-03-05  Run script now looks towards one big net
1050560 Gerard Haughian 2014-03-05  node version of bucketing algo, unit tests wrote for bucketing algo
9b7e1c3 Gerard Haughian 2014-03-05  added unit testing script for bucketing algo
6557e98 Ashley Cutmore 2014-03-05  After leaving the camera view any images taken will be uploaded to the server.
b8ff098 Gerard Haughian 2014-03-04  titided up bucketing script...i.e removed superfluous code and comments
10026cf John McCormac  2014-03-03  Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master
    into dev
1feb1dc John McCormac  2014-03-03  Corrected backup filler in batch set
8f5ef66 alex    2014-03-03  added AugmentLeafDataProvider
63ec5bb John McCormac  2014-03-03  Fixed batch mean sizing issue
2912fde John McCormac  2014-03-03  Modified data batcher to have backup
a3eca47 Ashley Cutmore 2014-03-02  Restructured Scenes to reflect designV2.0. Images can be save and displayed.
    Needs to be re-hooked up to network interface class.
e780411 John McCormac  2014-03-02  Fix image batching script for efficiency with VERY large batches
372bb9d John McCormac  2014-03-02  Customised big net archtecture for 256 images
b109fee John McCormac  2014-03-02  Added basic image_net copy architecture for bigger network
58681e5 John McCormac  2014-03-02  Mongo Querying available in ccn-make-batches, One Big Net option file included
e5f3a8d Gerard Haughian 2014-03-01  addressed mongo disk flushing issue for bucketing....seems to return expected
    results now
277ddbb Ashley Cutmore 2014-03-01  Database is now non-static. Server Interface reflects new design protocol. Tests
    re-written.
75ad227 Gerard Haughian 2014-03-01  bucketing logic much better...not resets counters etc before each run,
76f616a Simon Leigh    2014-03-01  more POST tests added.
```

```
6ee69ce Simon Leigh      2014-03-01      post request testing valid MongoID in response
3ef2541 Ashley Cutmore   2014-03-01      Network delegate methods send Notifications instead of invoking methods as event
    occur. Weakly coupled.
c41a60a Simon Leigh      2014-02-28      working post test
b10c9f4 Simon Leigh      2014-02-28      removed express parser because it was breaking formidable
9b8d2ac Simon Leigh      2014-02-28      added post test - not yet working
a06adac Simon Leigh      2014-02-28      added more tests for GET function
fa9d483 Simon Leigh      2014-02-28      fixed some indentation
5032f01 John McCormac    2014-02-28      mongoHelper Functions now doesn't run command when importing
7ec6baa John McCormac    2014-02-28      Added base extract buckets script
03cec06 Gerard Haughian  2014-02-27      bucketing algo returns bucketed species name as opposed to the synset
115c031 Simon Leigh      2014-02-27      Merge branch 'appServerTesting' into dev
1d0ae4e Simon Leigh      2014-02-27      Test scripts for all of configParser and getJob routes
bb24678 Gerard Haughian  2014-02-27      fixed exclude logic and now filters by name rather than synset
a438ac6 Simon Leigh      2014-02-27      added placeholder tests to do async testing of HTTP GET request
7ba0fa3 Gerard Haughian  2014-02-27      Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master
    into dev
999e67f Gerard Haughian  2014-02-27      bucketing function can now be run without restricting by component and returns
    the bucket not species
d42fe60 Simon Leigh      2014-02-27      Merge branch 'appServerTesting' into dev
c4c634b Simon Leigh      2014-02-27      Makefile updated to allow pretty commands and no global dependencies
12879f3 Simon Leigh      2014-02-27      added dependency
74cde29 Simon Leigh      2014-02-27      added test commands to readme
d131467 Simon Leigh      2014-02-27      moved json to top directory, added more tests
3ecf1f1 Gerard Haughian  2014-02-27      final result set in bucketing algo restricted to component
998cad4 Gerard Haughian  2014-02-27      Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master
    into dev
c28ec98 Gerard Haughian  2014-02-27      updated help functions to point to qa instance of mongo
05c958e John McCormac    2014-02-27      Merged into dev
4dd29d5 John McCormac    2014-02-27      Added Optimised PCA illumination augment script
6bcc91e Gerard Haughian  2014-02-27      added master insert script that will add all required data into any mongo
    instance we choose
a52d19d Gerard Haughian  2014-02-27      bucketing algo and python help functions complete
186520a Gerard Haughian  2014-02-26      python functions for running buecketing also complete
d62adc8 Simon    2014-02-26      fixed app.js
445819c Simon    2014-02-26      added configParser tests to get to 100% cov
ee645ef Gerard Haughian  2014-02-26      fixed infinite loop bug...whoops
a8806e3 Gerard Haughian  2014-02-26      added more efficient bucket updating query
764459d Gerard Haughian  2014-02-26      added profiling info and fixed bug
e6006c7 Gerard Haughian  2014-02-26      addressed some bucketing issues and added python helper functions for extracting
```

```
    image batches
fb10f5d Ashley Cutmore  2014-02-25      ServerInterface Test Coverage 70%
609bf7d Gerard Haughian 2014-02-25      tweaks to exclude logic
7d3fa7c Gerard Haughian 2014-02-25      added potential extension to taxon tree mongo insert i.e. add Exclude flag
f11ec29 Gerard Haughian 2014-02-25      bucketing algo fairly robust and complete... still some testing needed
a4e5d2b Ashley Cutmore  2014-02-25      Database Unit Tests finished. Server Unit Testing begun : ReFactored Class to
    better support testing.
425f4b5 Gerard Haughian 2014-02-25      bucketing algo near finished
62a5d36 Gerard Haughian 2014-02-25      added first attempt bucketing algo...needs testing
10bf10b Gerard Haughian 2014-02-25      fixed synset ids being returned as nested strings
5718d97 Gerard Haughian 2014-02-25      function for adding taxon tree path to mongodb
4cee356 Gerard Haughian 2014-02-25      tidied up xml to json extraction script...now very efficient and clean
590554b Gerard Haughian 2014-02-25      more efficient implementation of image meta-data extraction
654f872 Gerard Haughian 2014-02-25      xml to json converstion now adds exclude column to MongoDB
4b65624 Gerard Haughian 2014-02-25      added script to insert taxonomy tree into mongodb
ba993a0 Gerard Haughian 2014-02-24      python script to parse xml files to json, add wordnet info and save to mongodb
a582240 Gerard Haughian 2014-02-24      able to extract synsets and parse folders
324e3cd Gerard Haughian 2014-02-24      read synsets at run time
53c3969 Gerard Haughian 2014-02-24      ignoring jpg files in GraphicServer
9237006 Gerard Haughian 2014-02-24      Merge branch 'dev' of gitlab.doc.ic.ac.uk:bjm113/group-project-master into dev
1d18565 Gerard Haughian 2014-02-24      minor tweaks
2367212 John McCormac   2014-02-24      Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master
    into dev
53e19a7 John McCormac   2014-02-24      Added unpickle example
04ddaca Gerard Haughian 2014-02-24      determins synsets at run time
15806c2 John McCormac   2014-02-24      Corrected formatting for test README
9abb271 John McCormac   2014-02-24      Modified README
505e090 John McCormac   2014-02-24      Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master
    into dev
98d49d3 John McCormac   2014-02-24      Built test and coverage suite, added README
0613f19 Gerard Haughian 2014-02-24      tweaking xml to mongo converter to handle new image net xml files
4ed6ca2 Gerard Haughian 2014-02-24      running 'make test-cov' will delete lib-cov robustly before running jscoverage
7e5bf59 Gerard Haughian 2014-02-24      added script which inserts word net data into mongo
58a4539 John McCormac   2014-02-24      Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master
    into dev
bde7d9d John McCormac   2014-02-24      Tagging script now runs on imagenet directory
8d6bb98 John McCormac   2014-02-24      Tests now working, noccn updated accordingly, paths corrected
f57a0ca Stewart Douglas 2014-02-24      Made necessary changes to graphic.js to use formidable
5200127 Stewart Douglas 2014-02-24      Integrated formidable into routes/index.js
237d2e9 Ashley Cutmore  2014-02-23      Unit Tests for Database Interface exist. Coverage files flushed successfully.
```

```
a01e340 John McCormac   2014-02-23      Cleaned up structure, edited build.sh accordingly
0580d76 John McCormac   2014-02-23      Tagging script now tags with component probability
0d47c7c John McCormac   2014-02-23      Improved robustness of image counting for taxon tree
bd850a6 Simon Leigh     2014-02-22      fixed multiple multiparts being sent after each request
ce87487 Simon Leigh     2014-02-22      replaced express bodyparse with formidable
eaa2b49 Simon Leigh     2014-02-22       Merge branch 'nodeCoverage' into 'dev'
a2f044c Simon Leigh     2014-02-22      added pseudo dist that was missing
37b841a Simon Leigh     2014-02-22      merge NodeCoverage
232da34 Simon Leigh     2014-02-22      reverted changes to server relating to test coverage - not needed
131c13c John McCormac   2014-02-22      Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master
    into dev
6c0995c John McCormac   2014-02-21      Added pickled taxonomy tree, cleaned tag script
462e785 John McCormac   2014-02-21      Tagging script now write xmls from scratch
b233d29 John McCormac   2014-02-21      Imagenet taxonomy tree creation script available
4be2344 Simon Leigh     2014-02-21      added appServer code for coverage in plant.js
41901d8 Gerard Haughian 2014-02-21      added pseudo-dist startup option
ee67da7 Simon Leigh     2014-02-21      package.json back in, fixed .gitignore, fixed config parser, updated graphic env
    files, renamed pseudo-dist file
1313f12 Gerard Haughian 2014-02-21      removed non-conforming env file
c53cc14 Gerard Haughian 2014-02-21      unknown changes...seems work that simon did re: jpgs...???
72b194e Simon Leigh     2014-02-18      remove jpg.
34d5b84 Simon Leigh     2014-02-18      jpg
b2240c3 Simon Leigh     2014-02-18      Jpg.
330aaa2 Simon Leigh     2014-02-18      Fixed upload path.
8840198 Gerard Haughian 2014-02-18      took care of removing old liv-cov folder in makefile
e52c3bd Gerard Haughian 2014-02-18      tweaked code coverage logic...all seems to be working fine now
956a54e Stewart Douglas 2014-02-18      Commented out the config_parser in graphic.js
67953d0 Stewart Douglas 2014-02-18      ; . test coverage
01f807f Gerard Haughian 2014-02-18      fixed relative path of node js util script
e169e8a Gerard Haughian 2014-02-18      moved AppServer, GraphicServer and utils folder inside lib
edabc0f Gerard Haughian 2014-02-18      restructured nodejs code base for unit testing and code coverage...updated unit
    test scripts
6761a7f Gerard Haughian 2014-02-18      merging current code snapshot post sprint 1 from dev
524d137 Ashley Cutmore  2014-02-17      Merge branch 'iPhone_serverInterface' into dev
145dd86 Ashley Cutmore  2014-02-17      upload progress bar turns green on classification receipt. Camera view indicates
    a photo was taken by flashing the screen white.
202dad6 John McCormac   2014-02-16      Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master
    into dev
d988c74 Ashley Cutmore  2014-02-16      App will poll uploaded images for result until a classification is returned.
    Classification is saved to DB and UI updated to reflect change. Result is displayed as plain text JSON when user
```

```
    selects a completed item. On network failure the server object stops processing tasks. The network object can be
    invoked to start processing again by a user clicking the refresh button.
f16564a Simon Leigh      2014-02-16      small changes to upload/GET scripts
cf64fb5 John McCormac    2014-02-16      Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master
    into dev
1bb8832 John McCormac    2014-02-16      Bug fix in run.py script, now functions acceptably
e850481 Stewart Douglas 2014-02-16       Merge branch 'graphicServer' into dev
c783518 Stewart Douglas 2014-02-16       Merge branch 'graphicServer' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-
    master into graphicServer
e9480df Stewart Douglas 2014-02-16       routes/index.js now updates mongo on the VM with the correct image it was passed
b3b5e1a Stewart Douglas 2014-02-15       Merge branch 'graphicServer' into dev
10e67b6 Stewart Douglas 2014-02-15       Merge branch 'graphicServer' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-
    master into graphicServer
1e863d2 Stewart Douglas 2014-02-15       Added _id field to testClassify.js
63485c8 Stewart Douglas 2014-02-15       Edited routes/index.js and runtest.py
6421c86 Stewart Douglas 2014-02-15       Edited index.js so that it exec's the python script runtest.py
340c7aa Simon Leigh      2014-02-15      Merge branch 'serverBranch' into dev
bcaf17f Simon Leigh      2014-02-15      Complete interaction between app/server/classifier
b9bcb57 alex     2014-02-15      merging with my changes to runbucketing.py
008d3f6 alex     2014-02-15      ToDo runBucketing.py
976e3fe Simon Leigh      2014-02-15      Merge branch 'dev' of ssh://gitlab.doc.ic.ac.uk/bjm113/group-project-master into
    dev
d628caf Simon Leigh      2014-02-15      fixed mongo connection in graphic object update
b92d29e Gerard Haughian 2014-02-14       completed script to insert bucket info to mongo, updated node util script to
    search for and add bucket info to image meta data as its being converted from xml to json and subsequently inserted
    into a mongodb collection
dccc74f Gerard Haughian 2014-02-14       small logic tweaks
1b8d3b9 Stewart Douglas 2014-02-14       stub classifier code added
b12f1dc John McCormac    2014-02-14      Corrected relative paths for cfg files
a50ed87 John McCormac    2014-02-14      Updated test script output format
daf1277 John McCormac    2014-02-14      Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master
    into dev
84126fe John McCormac    2014-02-14      Added preliminary tests directory and stub test
501cc5a John McCormac    2014-02-14      Added runtest script
f28da3d John McCormac    2014-02-14      Run script output results
1e6830e alex     2014-02-12      Need to merge so I can add my modifications to ML/bucketing
c6c6f0f alex     2014-02-12      now have bucket data structure hopefully easy to interface with mongo. algo code is a bit
    dirty, but working, tests successful
17e2650 alex     2014-02-12      a bit dirty, but working, and tests successful
d41e8e5 alex     2014-02-12      changed parent dict to holding strings not lists, attempting to solve list as key issue
```

```
   by calling via parent dict rather than children dict
4d1984d Gerard Haughian 2014-02-12      added logic to extract buck information from mongo
bafbbe8 alex    2014-02-12      still buggy. just realised though, since class now enforces tree structure, it isn't a
   graph, it's a tree. changing names
48f9d64 John McCormac   2014-02-12      Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master
   into dev
8969648 John McCormac   2014-02-12      Run script added
25b6c1c John McCormac   2014-02-12      Tagging script and new networks added
a0aab2e Gerard Haughian 2014-02-12      added python script that will interface with graph.py extracting bucketing
   information and inserting it into MongoDB
40da883 alex    2014-02-12      compile error because using list as key, must find alternative
2b64185 alex    2014-02-12      added bucket data structure, status data structure, more granular testing - but maybe
   buggy
2fd9ab4 alex    2014-02-12      Merging in order to push changes to graph.py
edfbd18 alex    2014-02-12      modified addEdge() to enforce tree structure to reduce imagePropagation cockup risk
7e601fc Gerard Haughian 2014-02-11      updated graphic server startup command
51b7a56 Stewart Douglas 2014-02-11      Merge branch 'graphicServer' into dev
75cb397 Stewart Douglas 2014-02-11      edited index.js
07cb8e5 Ashley Cutmore  2014-02-11      Progress Bar is now persistent. Add new image button moved down into collection
   view. Force update uploads button now in menubar instead. Server interface grabs job number from DB to GET.
4185b51 Stewart Douglas 2014-02-11      Connect to the Mongo client
b38fbae Stewart Douglas 2014-02-11      Merge branch 'graphicServer' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-
   master into graphicServer
c2efce0 Ashley Cutmore  2014-02-11      Merge branch 'database_class' into dev
d483a8f Ashley Cutmore  2014-02-11      Server queues up requests and processes them in order. Observation Cell now in
   it s own class, fixes progressBar notification dealloc issue.
47102af alex    2014-02-11      merged, adding functioning bucketing
94d2dde Stewart Douglas 2014-02-11      Uncommented POST line
1124264 Gerard Haughian 2014-02-11      cleaned up testing scripts
84e368a Gerard Haughian 2014-02-11      tweaked formatting of unit test emails
7dde0aa Gerard Haughian 2014-02-11      Merge branch 'graphicServer' of gitlab.doc.ic.ac.uk:bjm113/group-project-master
   into graphicServer
f6c27f2 Gerard Haughian 2014-02-11      added concole logging of imcoming http post request from gitlab to CIServer
b4f20c3 Gerard Haughian 2014-02-11      added node code coverage and updated email parsing logic to include coverage
   report
87cb25b Stewart Douglas 2014-02-11      Graphic02 can now receive messages from a remote machine running testClassify.js,
    but still throws a mongo error which needs to be resolved
c2145f8 Gerard Haughian 2014-02-11      Merge branch 'dev' of 146.169.13.187:bjm113/group-project-master into
   graphicServer
274db09 Gerard Haughian 2014-02-11      added comments and removed superfluous comments from code
```

```
f75c8e7 Stewart Douglas 2014-02-11      Modified Mongo update command in routes/index.js
e428e3f Stewart Douglas 2014-02-11      Added files for graphic02 server and updated the config files for graphic02
530b4f7 Gerard Haughian 2014-02-11      added error handling on http POST requests
f2ab905 Simon Leigh     2014-02-10      Implemented get request at host/job/#JOBID#
cbaea47 Simon Leigh     2014-02-10      Fixed some syntax errors.
bc54cd9 Simon Leigh     2014-02-10      Implemented Job GET request.
b4aa778 Simon Leigh     2014-02-10      Further abstraction into config parsing helper.
f8afcef Simon Leigh     2014-02-10      Abstracted Configuration Parsing.
b6fe380 Simon Leigh     2014-02-10      updated Gitignore in root of project
0c1a97a Simon Leigh     2014-02-10      Fixed IP bindings in Mongo config.
3041aa4 Simon Leigh     2014-02-10      Merge branch 'appBranch' into dev
b002933 Simon Leigh     2014-02-10      MongoDB native drive transition complete. Minor differences in syntax corrected.
c170515 Simon Leigh     2014-02-10      Removed references to monk and moved to mongodb native.
58bc792 Simon Leigh     2014-02-10      Merge branch 'appBranch' into dev
d66c09d Simon Leigh     2014-02-10      Added Node.cfg to environment files for the VM.
6f2c03e Gerard Haughian 2014-02-10      ignoring uploaded jpg files
dd865e9 Simon Leigh     2014-02-10      merge appBranch and Dev
fa3d468 Simon Leigh     2014-02-10      updated app.js upload path and fixed mongodb log folders for vm@
84875df Gerard Haughian 2014-02-10      git now ignores node_module files
cb9b4a7 Gerard Haughian 2014-02-10      fixed env config files
35ff20f Simon Leigh     2014-02-09      Uploaded lock files to ensure dir structure consistent.
de56418 Gerard Haughian 2014-02-09      tweaked graphic start script
d2116f1 Gerard Haughian 2014-02-09      added iPhone stub file and finalised some logic in startup script
e326a11 Gerard Haughian 2014-02-08      Merge branch 'dev' of gitlab.doc.ic.ac.uk:bjm113/group-project-master into dev
bc10e57 Gerard Haughian 2014-02-08      tweaked start up logic...almost perfect...changed starting port of graphic mongod
    instances
7970d07 Gerard Haughian 2014-02-08      adding .gitignore to home dir of repo
b011476 Gerard Haughian 2014-02-08      Merge branch 'startup' into dev
45c7dc1 Gerard Haughian 2014-02-08      fixed some typos
ceaa244 Gerard Haughian 2014-02-08      renamed graphic config files
ead3745 Gerard Haughian 2014-02-08      Merge branch 'startup' into dev
d578375 Gerard Haughian 2014-02-08      tidied up startup script
69c318b Ashley Cutmore  2014-02-08      Storyboard now has results view. Server Interface saves returned JOB# to Database
    . CameraView saves segment selection to Database.
703ecbe Simon Leigh     2014-02-08      Had to push required files manually
36d980b Simon Leigh     2014-02-08      fixed env config naming and MERGED AppBranch
3d61289 Simon Leigh     2014-02-08      updated mongo config files
f05827e Simon Leigh     2014-02-08      Pushed new Config Files
37de1c8 Simon Leigh     2014-02-08      Updated App to parse environment config file (shared with MongoDB). Updated
    Environment Config files to contain Node specific initialisation data. Updated Environment Config files to run Mongo
```

```
    on correct ports.
d3fbdfe Gerard Haughian 2014-02-08      added content to startstop_graphic.sh script
2e34131 Gerard Haughian 2014-02-08      Merge branch 'startup' into dev
f2b9610 Gerard Haughian 2014-02-08      minor weaks
2e92efe Gerard Haughian 2014-02-08      added seperate config files for both graphic and vm envs. Added start and stop
    scripts for graphic startup
538e3af Gerard Haughian 2014-02-08      some additional changes
0a7fe1e Simon Leigh     2014-02-07      Updates to allow command line options for environment.
7bae405 Simon Leigh     2014-02-07      actually added the files in the subdirectory.
044c1de Simon Leigh     2014-02-07      moved Appserver to subdirectory of Nodejs
a6a8cd3 Gerard Haughian 2014-02-07      just 'bash'ed one out while I was drunk there...it was gas craic
e649a66 Gerard Haughian 2014-02-06      tweaked email formatting, commentted CIServer code
18e57fe Gerard Haughian 2014-02-06      Merge branch 'dev' of 146.169.13.187:bjm113/group-project-master into startup
5ef2a40 Simon Leigh     2014-02-06      initial commit of server with mongo integration
542e525 Gerard Haughian 2014-02-06      Merge branch 'dev' of 146.169.13.187:bjm113/group-project-master into startup
01aef70 John McCormac   2014-02-06      Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master
    into dev
4038e01 John McCormac   2014-02-06      Modified tagging script to accept options from cfg file
155ecef Gerard Haughian 2014-02-06      add stub directory and some stub files for the purpose of starting up an
    environment with stub processes for testing
7753c32 Gerard Haughian 2014-02-06      script now exits with status 0 if a push is made to a branch other than master/qa
    /dev
50a7c83 Gerard Haughian 2014-02-06      added start timestamp to CIServer
a5c5d5a Gerard Haughian 2014-02-06      added timestamp to log messages
b4410dd Gerard Haughian 2014-02-06      added .gitignore so git ignores any clones the CIServer does which running tests
6893708 Gerard Haughian 2014-02-06      tidied up layout and added comments
add25b8 Gerard Haughian 2014-02-06      Merge branch 'startup' into dev
d7b8294 Gerard Haughian 2014-02-06      added logic to detect valid branch checkins for running tests
42e7bb9 Gerard Haughian 2014-02-06      removed package.json because it is no longer needed: installed packages globally
7cfd189 Gerard Haughian 2014-02-06      added logic to ensure CI tests only run for checkins to dev/qa/master branches
c791d66 Gerard Haughian 2014-02-06      Merge branch 'startup' into dev
ca27b66 Gerard Haughian 2014-02-06      testSuite emails can now be sent to personal/doc accounts. added some tweaks to
    qa and prod test scripts
8fbd5b6 Gerard Haughian 2014-02-06      added some logic to cope with running remote processes
5775ecb Gerard Haughian 2014-02-06      fixed email settingings
6e47245 Gerard Haughian 2014-02-06      renamed start script, added additional logic
3982650 Gerard Haughian 2014-02-06      updated email addr settings, can now email to our doc accounts
8add032 Ashley Cutmore  2014-02-05      Server Interface Class uses a subclass of nsUrlConnection to track individual
    uploads. Custom camera view with specimen segment selection.
714a538 Gerard Haughian 2014-02-04      added starting and stopping of graphic server, added logic to ensure user is
```

```
    running script from a doc machine
dda1ac0 Gerard Haughian 2014-02-04      added more logic for starting processes
a48bcb3 Gerard Haughian 2014-02-04      changed mongodb startup ports for qa and prod envs
a3d0395 Gerard Haughian 2014-02-04      Merge branch 'xml2json' into dev
ed52677 Gerard Haughian 2014-02-04      Merge branch 'master' of 146.169.13.187:bjm113/group-project-master into xml2json
db9ce17 Gerard Haughian 2014-02-04      wrote unit test script for node.js...wrote a util node script for parsing mocha
    results
0ba59b7 Gerard Haughian 2014-02-04      added env and bin dirs with start up scripts and mongo config files
e869707 Gerard Haughian 2014-02-04      can now insert multiple parsed xml files into mongo collection
ad9f86f John McCormac   2014-02-04      Added tagging script
f69c102 John McCormac   2014-02-03      Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master
    into dev
3e92a19 John McCormac   2014-02-03      Added second plant component classifier model
535e976 Gerard Haughian 2014-02-03      added logic to tweak document prior to inserting in DB
3ce199f Gerard Haughian 2014-02-03      added functionality to interate through full image db filesystem and convert to
    json
c7fd685 Gerard Haughian 2014-02-02      added an xml to json parser script. It will then insert the parsed file in a
    mongo collection
f4d62aa Ashley Cutmore  2014-02-02      Database images now referred to as observations. Server class uploading on thread
     with progress being broadcast.
bf9b494 HoldenCaulfieldICL      2014-02-01      modified README with new installation instructions
858f366 HoldenCaulfieldICL      2014-02-01      modified README with new installation instructions
0835a52 HoldenCaulfieldICL      2014-02-01      modified README with new installation instructions
41b3e1c HoldenCaulfieldICL      2014-02-01      modified README with new installation instructions
c022650 HoldenCaulfieldICL      2014-02-01      modified README with new installation instructions
28a4b8a HoldenCaulfieldICL      2014-02-01      modified build.sh to properly install the whole damn thing
8e9e5da John McCormac   2014-02-01      Merge branch 'dev' of https://gitlab.doc.ic.ac.uk/bjm113/group-project-master
    into dev
be302d0 John McCormac   2014-02-01      Convdata.py now includes data provider for 256 sized images
75558a1 alex    2014-02-01      adding python script that checks whether joblib and scikit-learn libraries are installed.
     this for a cool bash script that gets cuda-convnet running like a babe
f929166 Gerard Haughian 2014-01-31      testSuite emails now have a complete subject line
8640a75 Gerard Haughian 2014-01-31      testSuite emails now have a complete subject line
c1e9c02 Gerard Haughian 2014-01-31      testSuite emails now have a complete subject line
81765dd Gerard Haughian 2014-01-31       testSuite now cleans up the repo when its finished running its tests
8f34e1d Gerard Haughian 2014-01-31      testSuite now cleans up the repo when its finished running its tests
da72a66 Gerard Haughian 2014-01-31      Merge branch 'dev' of gitlab.doc.ic.ac.uk:bjm113/group-project-master into dev
39593f4 Gerard Haughian 2014-01-31      Added Continuous Integration Server code
8c5d2ab John McCormac   2014-01-31      Can now exclude plant type from the make batches metadata in cfg files
efbd895 John McCormac   2014-01-31      Component checking network model now available
```

```
3e9b5fa John McCormac  2014-01-31      Corrected model name
b29ef23 Ashley Cutmore 2014-01-30      Network Class has basic network methods for sending data over POST
97e989b alex    2014-01-29      trying new network architecture
794d5ba Ashley Cutmore 2014-01-28      Database scheme rewrote. Database logic abstracted into class.
b4aadee John McCormac  2014-01-27      Corrected the include path for cuda library
bf0dfe3 John McCormac  2014-01-27      Adding basic working leaf data model
d235e4a Ashley Cutmore 2014-01-22      Merged seperate iOS repo into central repo
1cdcb08 Ashley Cutmore 2014-01-22      Added view to display information about each sample s classification
7c645d4 John McCormac  2014-01-22      merged cuda conv into central repo
044c9d2 John McCormac  2014-01-21      Finished tweaking to get a working model
ad0f647 Gerard Haughian 2014-01-18     pulled from prod after release and added an additional comment
8158731 Gerard Haughian 2014-01-18     merged from qa and added a comment
b6dece1 Gerard Haughian 2014-01-18     merged changes from dev and added some additional changes in qa
edb9d02 Gerard Haughian 2014-01-18     testing a commit to dev and propagating to master
a2f9055 John McCormac  2014-01-17      Added batch image parsing script
5ee3bbc John McCormac  2014-01-15      Combined noccn to be part of the convnet repo
a613e2a John McCormac  2014-01-15      Fixed merge conflicts
f3714ba John McCormac  2014-01-15      Added basic data parser
18df803 John McCormac  2014-01-15      Added xml parser to data batcher
be09abb John McCormac  2014-01-11      Modified to work on CUDA 5.5, built test case
2e606ae Ashley Cutmore 2013-12-28      Sample processing moved onto Background thread to free up UI. Image data saved in
    FileSystem not CoreData. Cleaned up tableview cell activityIndicator presentation.
137a566 Ashley Cutmore 2013-12-26      Implemented Upload Queue. Uploads sample to server, queues next one on completion
   . Stops loop on network fail. Loop starts on new sample added, app load or list   pull-down-to-refresh
2b67873 Ashley Cutmore 2013-12-26      Upload methods accept parameters and one file. Tested with a node.js http server.
    FIXED: possible to dismiss image source action sheet. FIXED: Storyboard title name no longer   root view
   controller
322dd81 Ashley Cutmore 2013-12-23      Added the Network delegate protocol to tableView with a functioning test method.
75bb2ac Ashley Cutmore 2013-12-21      Not presented with choice of camera/library if device has no camera
e0eee4a Ashley Cutmore 2013-12-21      Delete cell animates. Cell drawn differently for different sample.status values.
ad1c9d3 Ashley Cutmore 2013-12-20      Add photo to core-data. Shows thumbnail in table. Delete works with no animation.
517c814 Ashley Cutmore 2013-12-20      Add sample view appears with video-capture preview. No other functionality.
b3745a4 Ashley Cutmore 2013-12-20      Core data model setup. Table view loads all samples, displays names.
31ef32d Ashley Cutmore 2013-12-19      Initial Commit
1d41ac3 Daniel Nouri   2013-12-16      docs: Add example that was accidentally left out.
ef91f6d Daniel Nouri   2013-12-16      Follow symlinks in folders.
05a398d Daniel Nouri   2013-12-16      scikit-learn is now a soft dependency.
df9536f Daniel Nouri   2013-12-16      dataset: Remove superfluous input-path option.
cfa1ce8 Daniel Nouri   2013-12-16      Add installation instructions
e7aee02 Ashley Cutmore 2013-12-11      Added link to wiki
```

```
4399708 Ashley Cutmore  2013-12-11      Renamed README -> README.md
a0bec19 Ashley Cutmore  2013-12-11      Trying to get markdown to work
47f0a08 Ashley Cutmore  2013-12-11      Formatted into gitlabMarkdown
477317e Ashley Cutmore  2013-12-11      Moved info to wiki
87a0edf HoldenCaulfieldRye      2013-12-11      fixed numebring mistakes
a7aad4a HoldenCaulfieldRye      2013-12-11      More on ML, data, corwdsourcing
ff05b50 John McCormac   2013-12-11      Initial commit of README
27aa585 Daniel Nouri    2013-10-27      Clean up.
2289f26 Daniel Nouri    2013-10-12      Simplify write-preds operation
079aa80 Daniel Nouri    2013-10-12      Batch creator can now shuffle batch. Also simplified thumbnail creation.
eb763a9 Daniel Nouri    2013-09-14      Improve README
b07e19a Daniel Nouri    2013-09-14      Improvements to 'ccn-make-batches' script
29c256b Daniel Nouri    2013-09-14      Add 'always-save' option to 'ccn-train' script
5a57843 Daniel Nouri    2013-08-14      - Add ccn-make-batches command that creates batches given a directory   with one
    folder per image class.
1bb7caa Daniel Nouri    2013-08-14      Improve reusability
4810ac3 Daniel Nouri    2013-08-14      - In ccn-predict, No longer require the last layer to be called   'logprob'.  (
    This can also be a 'sqdiff' or whatever.)
d97cf37 Daniel Nouri    2013-06-09      Small doc changes
a6755b3 Daniel Nouri    2013-06-08      Merge b36ee97c298fa8ffa58779aa22232342209febb8
70c2f15 Daniel Nouri    2013-06-08      Add a README
b36ee97 Daniel Nouri    2013-06-08      Revert "Allow hooking in more data providers."
481a986 Daniel Nouri    2013-06-08      Reword a little
1633c60 Daniel Nouri    2013-06-08      Rename for Github
662cfdd Daniel Nouri    2013-06-08      Initial commit
733861d Daniel Nouri    2013-05-31      Allow passing in a random seed through an env variable.
f023f22 Daniel Nouri    2013-05-11      Implement dropout
9bfea79 Daniel Nouri    2013-04-25      Only warn about options that cannot be unchanged.
23eb72a Daniel Nouri    2013-02-09      A gitignore
0f640bb Daniel Nouri    2013-02-09      Image that data provider returns may be gray scale.
18db5a6 Daniel Nouri    2013-02-09      Allow hooking in more data providers.
962a724 Daniel Nouri    2013-02-09      Use Ubuntu package standard directories for better defaults.
6627f67 akrizhevsky@gmail.com   2012-07-25      Update note about supported CUDA versions.
dd8fd31 akrizhevsky@gmail.com   2012-07-17      Fixed bug in contrast normalization backpropagation code which caused
    wrong gradients to be computed near image borders.
4660ef5 akrizhevsky     2012-06-17      Fixed bug causing nailbed layer to compute wrong output for batches of size not
    divisible by 128.
4e5cd86 akrizhevsky     2012-06-08      Updated example layer definition/parameter files to conform to new rule that
    response normalization scale/power are given in parameter file.
35d0bc4 akrizhevsky     2012-06-08      Cross-map response layer can now be 'blocked'.
```

```
a2a6af8 akrizhevsky      2012-03-13      Started modifying code to support convolution with non-square images. Added
    response-normalization _across maps_ layer. Added 11pct CIFAR-10 example parameter files.
3878e8f akrizhevsky      2012-03-13      Started modifying code to support convolution with non-square images. Added
    response-normalization _across maps_ layer. Added 11pct CIFAR-10 example parameter files.
38ae981 akrizhevsky      2012-02-10      Most layer types now should work reasonably well with minibatch size 64 or 32.
    Fixed --conserve-mem option so it can be combined with -f (i.e. its value can be changed after a net has been saved).
f97e2a4 akrizhevsky      2012-02-03      Convolution with minibatch size 32 or 64 is now about twice as fast. Added --
    conserve-mem parameter.
c2e4f03 akrizhevsky      2012-01-20      Some color space manipulation routines.
24cfd8a akrizhevsky@gmail.com   2012-01-19      Fixed typo in LinearNeuron initialization which led to wrong computation.
9ea3642 akrizhevsky      2012-01-12      Fixed faulty assert statement found by James Bergstra
bcfc8e9 akrizhevsky      2012-01-11      Image scaling with bilinear filtering.
eb0e6b7 akrizhevsky      2011-12-31      Some notes to help locate paths in build.sh
470cf1d akrizhevsky      2011-12-30      Give initW default value
53749b6 akrizhevsky      2011-12-30      Can initialize biases from outside source too now. Initialization functions can
    take parameters.
343780d akrizhevsky      2011-12-29      Minor cleanup
690293a akrizhevsky      2011-12-29      Weight initialization from python function. Also layers with 0 learning rate no
    longer do any computation in bprop pass.
c9e994f akrizhevsky      2011-12-26      Fixed crash on invalid filter size specification. shownet.py no longer loads data
     just to plot filters/error.
8fca731 akrizhevsky      2011-12-22      Fixed shownet.py script broken a few commits ago.
1ef58cd akrizhevsky      2011-12-22      Fixed crash if input to sum/max layer has differnet dimensionalities.
de2897e akrizhevsky      2011-12-22      Fixed crash if input to sum/max layer has differnet dimensionalities.
c0af3c4 akrizhevsky      2011-12-21      Forgot one more Windows-specific change from Oriol Vinyals
917b53d akrizhevsky      2011-12-21      Added sum-of-squares cost function.
7f28479 akrizhevsky      2011-12-21      Included changes by Oriol Vinyals to allow compilation on Windows systems. Added
    elementwise max layer. Added linear/sqrt/square neurons.
144203a akrizhevsky      2011-12-18      Fixed learning rate in just-uploaded param file -- accidentally left it too low
c027ce3 akrizhevsky      2011-12-18      Fixed header in example file
7345477 akrizhevsky      2011-12-17      Sum layer now takes coefficients. Added gaussian blur and nailbed layer types.
0a72e9a akrizhevsky      2011-12-14      Refactored some of the layer parsing code a bit.
d0139c6 akrizhevsky      2011-12-12      It should now be possible to put a pooling layer right over the data.
e6aeff1 akrizhevsky      2011-12-11      Each layer checks if there are grad consumers below it, so you can now put a
    pooling layer right over a data layer without having to compute a gradient for it.
5d7923d akrizhevsky      2011-12-10      Fixed mistaken error message when input layer has no outputs
90b0e73 akrizhevsky      2011-12-10      Apply neuron after doing max, where applicable.
8520655 akrizhevsky      2011-12-09      Recurrent layer support. Sum layer support. All layers with weights take multiple
     inputs. Decoupled neuron activation functions from fc/conv/local layers.
b668faf akrizhevsky      2011-10-15      Added ability to write convnet's features to disk using shownet script
```

```
c54fce3 akrizhevsky      2011-10-15      Fixed recently-introduced crashing bug on save. Added ability to unshare
    convolutional units.
fbaa0d7 akrizhevsky@gmail.com   2011-10-06      shownet now only takes its own options
3d997c1 akrizhevsky@gmail.com   2011-10-06      shownet.py --show-filters no longer modifies the filters, thereby
    screwing up if --show-preds is passed simultaneously
028e556 akrizhevsky@gmail.com   2011-10-05      Correct label now in red
71ab400 akrizhevsky@gmail.com   2011-10-04      Simplified data indexing in shownet
b8e2724 akrizhevsky@gmail.com   2011-10-04      Data provider now returns plottable data
ea5bae3 akrizhevsky@gmail.com   2011-10-04      --show-preds now takes softmax layer name itself, so it's no longer
    necessary to supply --logreg-name
52028db akrizhevsky@gmail.com   2011-10-04      shownet simplifications, layers-example.cfg bloat comment
11b34e3 akrizhevsky@gmail.com   2011-10-04      Data worker now has dp pointer
0eaba47 akrizhevsky@gmail.com   2011-10-03      Fixed --only-errors in shownet
4a84d31 akrizhevsky@gmail.com   2011-10-03      Minor bug in shownet for global filters
9e5e5b4 akrizhevsky@gmail.com   2011-10-03      Rewrote shownet so that it extends the model class now, making it easy to
     run the model from that script. Added ability to show predictions made by the net.
efe6a3f akrizhevsky@gmail.com   2011-10-02      Don't use temp matrix for local unit gradient computation. Saves memory
    and also faster.
057f63a akrizhevsky@gmail.com   2011-10-02      Better error-checking in local/conv layers
01c9977 akrizhevsky@gmail.com   2011-10-02      Matrix now uses long int to avoid overflow with really big data matrices
e4bdbb6 akrizhevsky@gmail.com   2011-09-30      Added support for random sparse connectivity for local, unshared units
6741783 akrizhevsky@gmail.com   2011-09-27      Updated example file to show local units usage
9dfe8ee akrizhevsky@gmail.com   2011-09-27      Local, unshared unit support added
4580ab4 akrizhevsky@gmail.com   2011-09-24      Updated example file to show random sparse connectivity
c43f12b akrizhevsky@gmail.com   2011-09-24      Added sparse convolutional layers with random connectivity.
4108b63 akrizhevsky@gmail.com   2011-09-20      Neurons that require memory of input for gradient computation no longer
    perform needless copy when computing activation
8788986 akrizhevsky@gmail.com   2011-09-19      Bounded relu unit added
b9f8cbe akrizhevsky@gmail.com   2011-09-18      Fixed variable names in shownet
bc77cb0 akrizhevsky@gmail.com   2011-09-18      Fixed variable names in example files
9438cba akrizhevsky@gmail.com   2011-09-18      CIFAR-10 example files now in svn
9c77174 akrizhevsky@gmail.com   2011-09-18      Renamed numFilters to filters, etc
e0cb096 akrizhevsky@gmail.com   2011-09-18      Groups now multiplies numFilters rather than divides it
0da8c0d akrizhevsky@gmail.com   2011-09-18      Mention groups in printout when net starts
3826e62 akrizhevsky@gmail.com   2011-09-18      Added support for sparse connectivity in convolutional layers. Sped up
    weight gradient computation by 33% in convolutional layers.
e9af5ba akrizhevsky@gmail.com   2011-09-14      Fixed mistaken cost operator return value, not that it's used anywhere
0d70bfb akrizhevsky@gmail.com   2011-09-14      Put back convenience unary operator functions in nvmatrix. Renamed binary
     operator functions to be consistent with unary.
80053bd akrizhevsky@gmail.com   2011-09-11      NVMatrix.apply now just takes operator
```

```
bf7ff4a akrizhevsky@gmail.com   2011-09-10      Misnamed variables
eb969e5 akrizhevsky@gmail.com   2011-09-10      Misnamed function
500e28b akrizhevsky@gmail.com   2011-09-10      Faster row sum in some cases
96e942f akrizhevsky@gmail.com   2011-09-09      Some error checking in shownet
9ab9955 akrizhevsky@gmail.com   2011-09-09      Grads --> Grad
cb21136 akrizhevsky@gmail.com   2011-09-08      And another
033ff63 akrizhevsky@gmail.com   2011-09-07      Another instance of 'error' removed
43de8c1 akrizhevsky@gmail.com   2011-09-07      Code organization
da4add5 akrizhevsky@gmail.com   2011-09-07      Even more cost refactoring
8e2cce1 akrizhevsky@gmail.com   2011-09-06      Remove references to 'error' in favor of consistent use of 'cost'
d3fc8c5 akrizhevsky@gmail.com   2011-09-04      Didn't actually need that new operator
aac5ad3 akrizhevsky@gmail.com   2011-09-03      Faster tanh, also abstanh
e888a3c akrizhevsky@gmail.com   2011-09-02      Soft relu unit support added
81214f4 akrizhevsky@gmail.com   2011-09-02      Undo that ... not consistent with other stuff.
fc32837 akrizhevsky@gmail.com   2011-09-02      Neuron names case insensitive
459ae00 akrizhevsky@gmail.com   2011-09-02      Neuron parameters will now be in square brackets, to help distinguish
    from the actual function call.
dbe6015 akrizhevsky@gmail.com   2011-09-02      Move neuron parsers list closer to layer parsers list
5d229df akrizhevsky@gmail.com   2011-09-02      Tanh unit support added.
2f46ebe akrizhevsky@gmail.com   2011-09-01      Logreg objective now checks that input softmax has correct dimensionality
2ce23f6 akrizhevsky@gmail.com   2011-09-01      Clarified what the data provider is doing a bit. Also renamed convdp.py
    to convdata.py to be consistent with data.py
a322791 akrizhevsky@gmail.com   2011-08-30      DP broken in last commit
7947a8d akrizhevsky@gmail.com   2011-08-30      Do some error checking on data type returned by data provider
87f41cb akrizhevsky@gmail.com   2011-08-30      Linesep screwup
df0e82a akrizhevsky@gmail.com   2011-08-29      Separated pooling layers into distinct classes. scaleTargets now computed
    in Layer class rather than in every bropActs override.
6f7616c akrizhevsky@gmail.com   2011-08-26      Delete sync.h; unused file.
270104d akrizhevsky@gmail.com   2011-08-25      Removed useless variable
e7d77ac akrizhevsky@gmail.com   2011-08-24      Link to shownet documentation
904c434 akrizhevsky@gmail.com   2011-08-24      Check for presence of matplotlib in shownet.py
e05918b akrizhevsky@gmail.com   2011-08-24      Added cost function plotting screenshot.
e8d6c3a akrizhevsky@gmail.com   2011-08-24      Added basic script to look inside saved checkpoints.
537665d akrizhevsky@gmail.com   2011-08-19      Factored out WeightLayer. Added passType variable to fwd/bwd passes to
    allow finer control of operations.
fff8d93 akrizhevsky@gmail.com   2011-08-16      Minor parser simplifications
ac9e6a8 akrizhevsky@gmail.com   2011-08-08      Slightly faster weight gradient computation for conv layers.
7f78bbf akrizhevsky@gmail.com   2011-08-05      More consistent worker parameters
99c91db akrizhevsky@gmail.com   2011-08-05      build.sh no longer messes with LD_LIBRARY_PATH
8720184 akrizhevsky@gmail.com   2011-08-04      Removed old method
```

```
9bad23d akrizhevsky@gmail.com   2011-08-04      Simplified the implementation of the fix in r200 a bit.
9c660f9 akrizhevsky@gmail.com   2011-08-04      Better default initW values for example files
7e17d9c akrizhevsky@gmail.com   2011-08-04      Fixed grad computation for fc layers with multiple inputs. Grad checking
    function now also checks for nans and infs in numerical computation.
df8a946 akrizhevsky@gmail.com   2011-07-28      Wrap BLAS includes in extern C for better compatibility.
b13d18c akrizhevsky@gmail.com   2011-07-25      Correct gradient scale for cost function coefficients unequal to 1 (
    broken in last commit).
27a6f4a akrizhevsky@gmail.com   2011-07-25      Revert GC_SUPPRESS_PASSES default value
8ffde54 akrizhevsky@gmail.com   2011-07-25      More numerically stable and faster computation of logistic regression
    gradients.
999f7dd akrizhevsky@gmail.com   2011-07-25      Contrast normalization can now be applied to data layer too.
9dbc6f7 akrizhevsky@gmail.com   2011-07-24      Fixed some incorrect comments
c3e2253 akrizhevsky@gmail.com   2011-07-24      That was a silly comment.
9affd1e akrizhevsky@gmail.com   2011-07-24      Bring example files up to speed
084f520 akrizhevsky@gmail.com   2011-07-24      Response normalization now takes exponent parameter. Also added proper
    contrast normalization.
ca036ef akrizhevsky@gmail.com   2011-07-23      Faster response normalization, especially for big regions.
0dea753 akrizhevsky@gmail.com   2011-07-23      Stray print statement..
a96834e akrizhevsky@gmail.com   2011-07-23      Disable incomplete avg pooling code
8b7b1c3 akrizhevsky@gmail.com   2011-07-23      BSD license
aafa115 akrizhevsky@gmail.com   2011-07-20      Refactoring.
c0a9f04 akrizhevsky@gmail.com   2011-07-19      License
7598c63 akrizhevsky@gmail.com   2011-07-19      Deleted useless method.
dabfba1 akrizhevsky@gmail.com   2011-07-18      Separated weight gradient computation from input gradient computation.
97af5ce akrizhevsky@gmail.com   2011-07-18      Removed dependency of Layer on ConvNet.
972dfac akrizhevsky@gmail.com   2011-07-15      Removed some unnecessary casts.
848f779 akrizhevsky@gmail.com   2011-07-14      Updated build.sh with descriptions of environment variables.
c1655fa akrizhevsky@gmail.com   2011-07-13      Refactored Cost a bit to make it resemble its parent class more. No
    longer truncate temporary weight gradient matrix in convolutional layers when _saveBwdActs is true. This makes
    gradient computation in convolutional layers slightly faster.
a61c020 akrizhevsky@gmail.com   2011-07-11      Fixed confusing comment.
509faaa akrizhevsky@gmail.com   2011-07-11      Changed default behavior to not delete forward/backward activity matrices
     after they're used. Nets something like a 6% performance gain, so why not take it while the memory's available.
47893e7 akrizhevsky@gmail.com   2011-07-11      Numerically stable backward pass for contrast normalization. Previous
    version was occasionally dividing by zero or near-zero quantities.
7bcf9f5 akrizhevsky@gmail.com   2011-07-11      Updated example files with contrast normalization layer stuff.
c7db104 akrizhevsky@gmail.com   2011-07-11      Faster contrast normalization backward pass for normalization regions >=
    6 pixels.
50a0d7f akrizhevsky@gmail.com   2011-07-11      Renamed subsX to sizeX in example config files.
e274000 akrizhevsky@gmail.com   2011-07-11      Faster contrast normalization forward pass for normalization regions >= 6
```

```
    pixels.
17c6580 akrizhevsky@gmail.com   2011-07-10      Contrast normalization layer support. This is more computationally-
   intensive than I realized. Hopefully faster version coming up soon.
3f692b3 akrizhevsky@gmail.com   2011-07-08      made avg pooling a few times faster
85dfd9c akrizhevsky@gmail.com   2011-07-07      moved some the kernel calling code in layer.cu into layer_kernel.cu. also
    fixed minor cpu memory leak in ErrorResult destructor. a hashmap was being leaked every minibatch.
418bd76 akrizhevsky@gmail.com   2011-07-07      cleaned up bprop a bit. now the individual layers don't have to
   explicitly call bprop on the layers below them.
f7cef8a akrizhevsky@gmail.com   2011-07-06      delete ..unnecessary print
e5b0484 akrizhevsky@gmail.com   2011-07-06      delete commented code
0b0ca59 akrizhevsky@gmail.com   2011-07-06      fixed weightacts bug from last commit
de9f700 akrizhevsky@gmail.com   2011-07-06      got rid of mfi for good. also added data provider that trains on
   translations of cifar10
7b0dd72 akrizhevsky@gmail.com   2011-07-06      moved max_data_on_gpu to util.cuh with the rest of the defines
9137e5a akrizhevsky@gmail.com   2011-07-06      removed restriction that minibatch size must be a multiple of 128. also
   removed mfi output order from cudaconv2, since it was unused.
ad04603 akrizhevsky@gmail.com   2011-07-05      simplified fprop a  bit
c82866c akrizhevsky@gmail.com   2011-07-05      Merged LayerGraph into ConvNet, since they are essentially the same thing
    now that Workers have taken most of ConvNet's jobs away.
fc18808 akrizhevsky@gmail.com   2011-07-05      gradient checking layer files back in vc
0a066c7 akrizhevsky@gmail.com   2011-07-05      add support for shared biases in convolutional layers
7244a1a akrizhevsky@gmail.com   2011-07-05      introduced worker classes so convnet's main loop isn't a jumble of if
   statements. also templated Data, so now it's CPUData and GPUData
4b0bd21 akrizhevsky@gmail.com   2011-07-03      added MAX_DATA_ON_GPU parameter to control how much data can be stored on
    gpu
fcd1acb akrizhevsky@gmail.com   2011-07-03      renamed propgrad
ba632ce akrizhevsky@gmail.com   2011-07-02      fixed slight memory leak in LogregCost, also removed upper bound
   restriction on conv stride size
7267568 akrizhevsky@gmail.com   2011-07-02      forgot the layer files
c14c825 akrizhevsky@gmail.com   2011-07-02      moved layer cfgs to ./example-layers
255a220 akrizhevsky@gmail.com   2011-07-02      report percent error instead of accuracy. also some code consolidation in
    layer.cu
ad6048d akrizhevsky@gmail.com   2011-07-01      print error msg when param files dont exist
7f2a1c7 akrizhevsky@gmail.com   2011-07-01      delete gc files from trunk, they're downloads now
587b16e akrizhevsky@gmail.com   2011-07-01      more variable refactorings and neuron cleanup
b40eb7d akrizhevsky@gmail.com   2011-07-01      variable name refactorings
0db702c akrizhevsky@gmail.com   2011-07-01      nicer output
0262de8 akrizhevsky@gmail.com   2011-06-30      dont use grads matrix for fc layers
61b1e78 akrizhevsky@gmail.com   2011-06-30      got rid of some useless makefile stuff
288a861 akrizhevsky@gmail.com   2011-06-30      added readme
```

```
83430d9 akrizhevsky@gmail.com   2011-06-30      default param files
522676a akrizhevsky@gmail.com   2011-06-30      git-svn-id: http://cuda-convnet.googlecode.com/svn/trunk@27 bc73d74b-de6f
    -cad7-86a4-6ddd3d5d919c
173d333 akrizhevsky@gmail.com   2011-06-30      git-svn-id: http://cuda-convnet.googlecode.com/svn/trunk@13 bc73d74b-de6f
    -cad7-86a4-6ddd3d5d919c
bc809dc akrizhevsky@gmail.com   2011-06-29      link cblas
cb876ce akrizhevsky@gmail.com   2011-06-29      unnecessary file
90753db akrizhevsky@gmail.com   2011-06-29      initial commit
2d90b83 (no author)     2011-06-29      Initial directory structure.
```