Classification of Pipe Weld Images with Deep Neural Networks
— Literature Survey —

Dalyac Alexandre
ad6813@ic.ac.uk

Supervisors: Prof Murray Shanahan and Mr Jack Kelly
Course: CO541, Imperial College London

May 28, 2014

Abstract

Automatic image classification experienced a breakthrough in 2012 with the advent of GPU implementations of deep neural networks. Since then, state-of-the-art has centred around improving these deep neural networks. The following is a literature survey of papers relevant to the task of learning to automatically multi-tag images of pipe welds, from a restrictive number of training cases, and with high-level knowledge of some abstract features. It is therefore divided into 5 sections: foundations of machine learning with neural networks, deep convolutional neural networks (including deep belief networks), multi-tag learning, learning with few training examples, and incorporating knowledge of the structure of the data into the network architecture to optimise learning.

Include a separate section on progress that describes: the activities and accomplishments of the project to date; any problems or obstacles that might have cropped up and how those problems or obstacles are being dealt with; and plans for the next phases of the project.

Contents

1 Introduction

Automatic classification of pipe weld images has strong academic and economic motivations: the classification task is highly stochastic, to the extent that until 2012 technology was incapable of providing the means to automate it. That is to say, the distribution of each class to detect is of such high variance that it has it is rarely possible for computer vision programmers to hand-specify sufficiently abstract feature detectors that attain high classification accuracy. Deep learning distinguishes itself from other forms of machine learning by learning features; given sufficiently many training examples and computational power, it can learn a high number of features at many different levels of abstractness. In this particular case, the number of training cases may be insufficient to learn good features with deep learning algorithms. One way to remedy this would be make use of knowledge of the high-level features used by humans for this particular task. However, little to no research exists in this field.

The economic motivation is industrial: ControlPoint currently resorts to humans to classify images, which results in high costs and a much slow turnover than automatic classification could offer.

Include a separate section on progress that describes: the activities and accomplishments of the project to date; any problems or obstacles that might have cropped up and how those problems or obstacles are being dealt with; and plans for the next phases of the project.

1.1 Motivation and Objectives

1.2 Contributions

2 Background

2.1 Deep Neural Networks

2.2 Deep Convolutional Neural Networks

2.2.1 Discriminative Models

2.2.2 Generative Models

2.3 Multi-Tagging

2.4 Transfer Learning: learning with a restricted training set

3 Design

4 Implementation

5 Experimentation

6 Conclusions and Future Work

this is just [1] a sentence [2] showing [3] how citations/references work.

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.

7 From Plant Report - useful looking stuff

- **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.

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

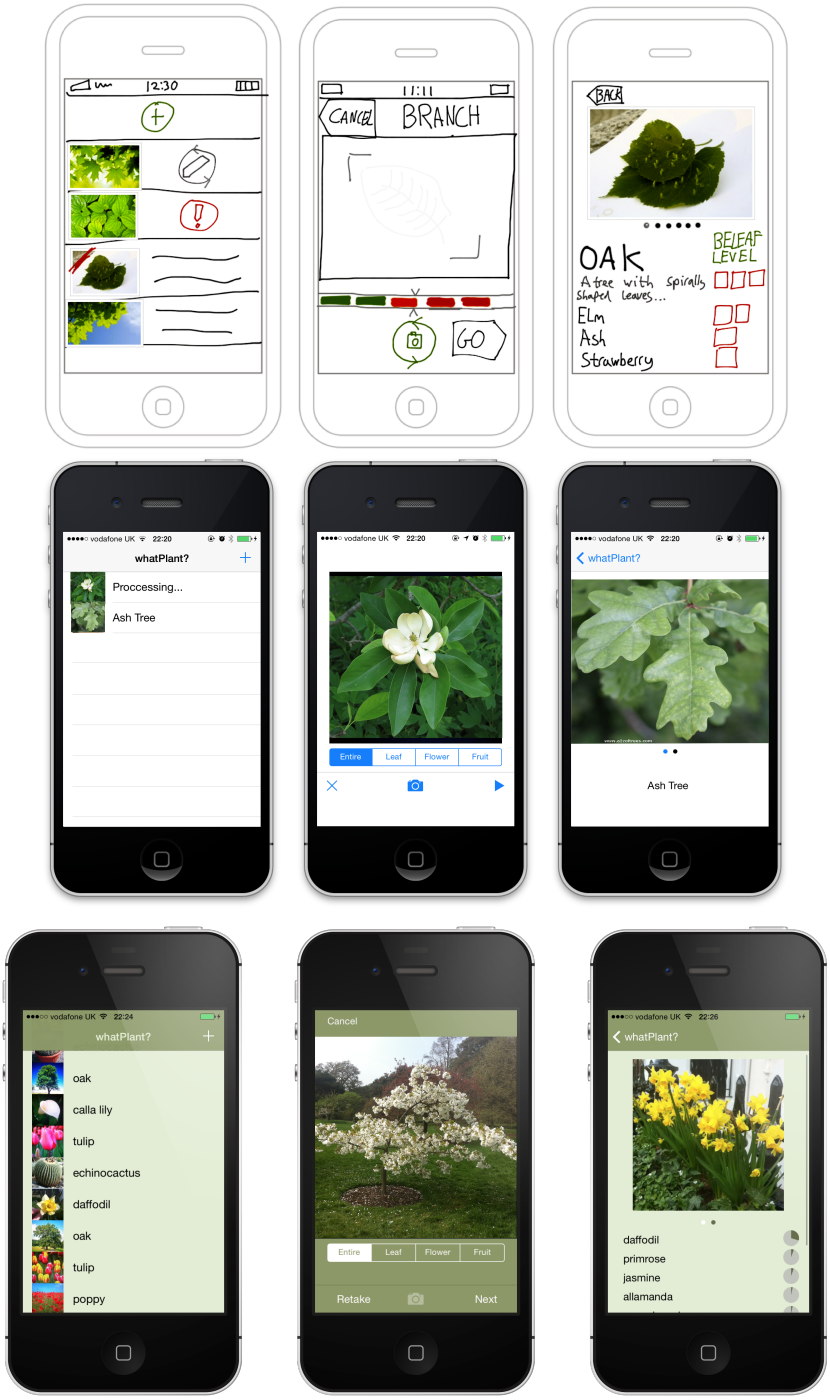

Figure 1: Evolution of the App

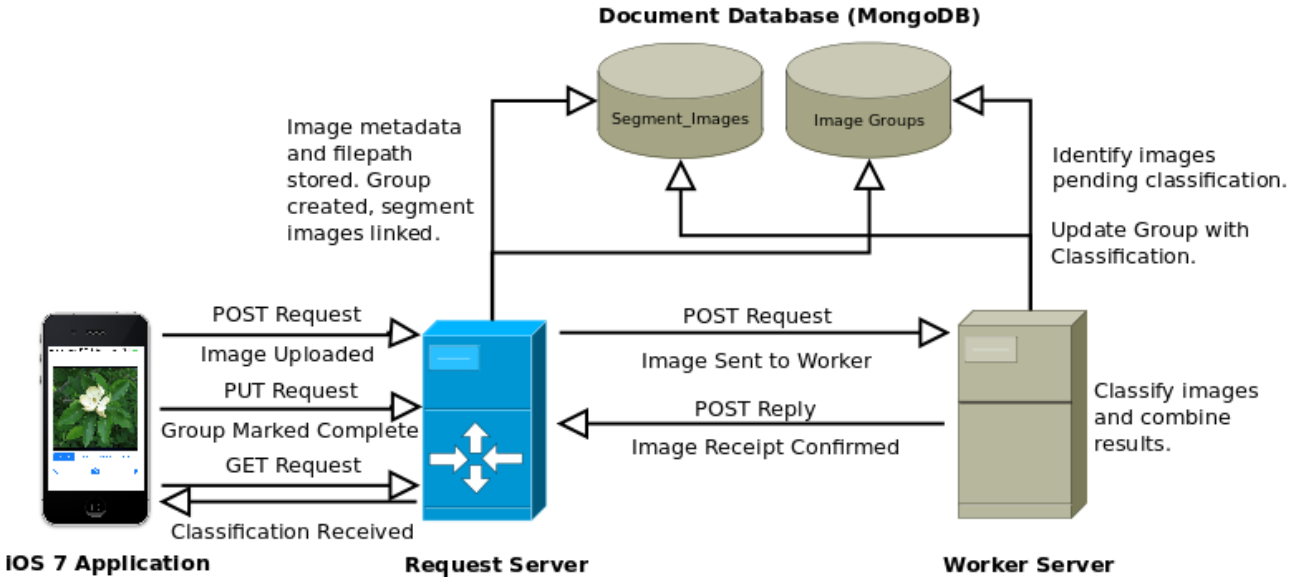

Figure 2: System Context Diagram

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 1: Code Coverage for Request Server

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 [**?**]. 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 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
```

A appendix part 1

A.1 appendix part 1.1

A.2 appendix part 1.2

B appendix part 2

B.1 appendix part 2.1

B.2 appendix part 2.2