
University of Oxford

Fourth year project

---

# A Flower Classification Application

---

*Author:*

James Routley

Trinity College

*Supervisors:*

Professor Andrew Zisserman

Yuning Chai

May 2015

# Contents

# Chapter 1

# Introduction

## 1.1 Definition of the problem

## 1.2 Why problem is worth solving, challenges which occur during this type of classification

## 1.3 Description of data

This project makes use of the Oxford 17 category flower dataset and the Oxford 102 category flower dataset.

'Both contain sets of images of flowers which commonly occur in the United Kingdom. The images have large scale pose and light variations. In addition, there are categories that have large variations within the category and several very similar categories.' - ?

The 102 dataset are split into training, validation and test sets.

The more manageable 17 dataset was used at first to get an overview of the classification process.

Once an initial classification pipeline was developed, the more comprehensive 102 dataset was used.

## 1.4 Achievements

# Chapter 2

# Literature review

(3-5 pages long. Gives reader knowledge of what people have done before on this topic.)

This application builds on

## 2.1 Visual classification apps

Similar apps: Leafsnap - iPhone application which uses computer vision techiniques to identify trees based on leaf images. Covers the 185 tree species in Northeastern US. Uses image segmentation (`http://neerajkumar.org/base/papers/nk_eccv2012_leafsnap.pdf`).

Dog classification: `http://www.umiacs.umd.edu/~kanazawa/papers/eccv2012_dog_final.pdf`

Google googles (very broad - ): `https://play.google.com/store/apps/details?id=com.google.android.apps.unveil&hl=en_GB`

(there are for dogs, leaves) - look up these papers and compare them.

## 2.2 Flower classification apps

Whilst these do exist, most either provide a database of plant knowledge which the user then manually searches through:

(https://play.google.com/store/apps/details?id=org.pottssoftware.agps21, https://play.google.com/store/apps/details?id=com.luontoportti, https://itunes.apple.com/us/app/ipflanzen/id416983587?mt=8),

or use a large network of users to crowdsource a classification.

(https://play.google.com/store/apps/details?id=cz.thran.flowerchecker, https://play.google.com/store/apps/details?id=org.plantnet&hl=en)

## 2.3 Flower Classification

Look over previous papers on flower classification (http://www.robots.ox.ac.uk/~vgg/research/flowers_demo/index.html )

## 2.4 Convolutional Neural Networks

CNNs, where they came from, who invented them, imagenet challenge (http://www.image-net.org/), image depicting CNN architecture.

Description of LibLinear?

# Chapter 3

# Classification

## 3.1 Overview of classification pipeline

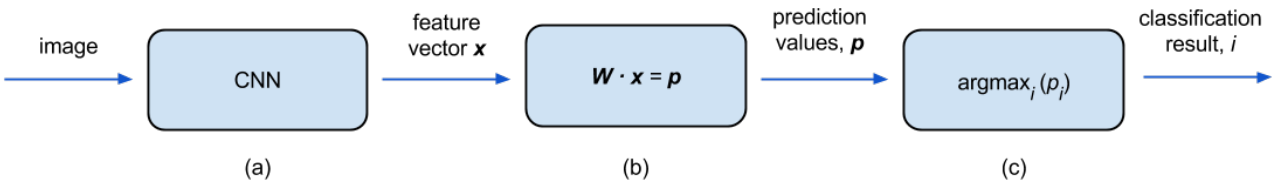



Figure 3.1: (a): Convolutional neural network inspired feature extraction. An image is supplied to the convolutional neural network which generates its feature vector $x$, which describes the content of the photograph. (b): Support vector machine image classification. The vector is compared to the previously found weight vectors $W$, producting a vector of prediction values, $p$. (c) The index $i$ of the maximum value in $p$ is found, which gives the classification result $i$.

## 3.2 Convolutional neural network (CNN) inspired feature extraction

The pipeline makes use of the first six layers of the fast convolutional neural network CNN-F introduced in “Chatfield et al. [1]. This algorithm takes a photo as its input and produces a vector which describes the features of the photo. The photo is cropped to 224x224 pixels and normalised before use and the outputted feature vector is also normalised.

The CNN is trained on the ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012) dataset [2]. Details of the training can be found in [1]. The feature extraction process itself is treated

as a black box in this project.

## 3.3 Support vector machine (SVM) image classification

### 3.3.1 How SVMs are used in the classification pipeline

The classification pipieline uses the open source support vector machine (SVM) library LibLinear [3] for quick, large scale classification. One-vs-the-rest classification is used. $n$ SVM models are trained for $N$ flower categories, producing $N$ weight vectors, $\boldsymbol{w_1}, \boldsymbol{w_2}, ..., \boldsymbol{w_N}$. These weight vectors are stacked into a weight matrix $\boldsymbol{W}$:

$$\boldsymbol{W} = \begin{pmatrix} \boldsymbol{w_1} \\ \boldsymbol{w_2} \\ \vdots \\ \boldsymbol{w_n} \end{pmatrix}$$

During classification, the scalar product of the unseen flower's feature vector, $\boldsymbol{x}$ and each of the weight vectors is calculated, to produce a vector of prediction values, $p_1, p_2 ... p_n$:

$$\boldsymbol{W} \cdot \boldsymbol{x} = \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{pmatrix}$$

The value of $p_i$ indicates how closely the unseen flower's features matched those of category $i$. The higher the product, the more closely they match. The unseen flower's category is therefore predicted to be that which produced the greatest prediction value.

### 3.3.2 Training and testing SVMs

Before use, the SVM model $\boldsymbol{W}$ must be trained. They are trained using a set of images which are known to be of certain categories. These models are then tested, using a set of images also known to be of certain categories, to check they are sufficiently accurate.

The C parameter of the SVM must be adjusted. The C Parameter allows for a solution with a larger

margin, in return for the violation of some constraints. To find the optimal C Parameter, SVM models are trained using the training set images before testing against the validation set images. The change in classification accuracy is recorded as the C Parameter is changed.

#### Training

SVM models are trained using LibLinear's training function. To train a model to recognise a flower category $i$, all photos in the training and validation sets are passed to the training function, with the photos of category $i$ labeled as in the category, and all other flowers labeled as not in the category. The flowers are labeled as in or not in the category by passing a 1 or -1 respectively to the training function, alongside that flower's feature vector.

The training function produces a weight vector of order 4096, which describes the characteristics of the flowers in that category.

#### Testing

Testing allows the quality of each SVM model to be checked. Testing uses the test subset of the flower datasets. Each flower in the test set is run against each of the SVM models generated during training. The dot product of the flower's feature vector and each of the model's weight vectors is found. The greater the result of the dot product, the greater the correlation between the tested flower and that category. Thus the category which produces the greatest dot product result is predicted to be the category of the flower.

By comparing the actual categories of the flowers in the test set to the predicted categories, an overall percentage accuracy can be calculated.

## 3.4 Experiments

(half the content. 7 pages, half should be experiments)

### 3.4.1 SVM accuracy

#### How we define accuracy

Accuracy is defined as the percentage number of correct classifications, where under correct classification, the category of the model which most closely identifies with the flower is indeed the category of that flower:

$$Accuracy = \frac{Number\ of\ correctly\ classified\ flowers}{Total\ number\ of\ flowers} \times 100$$

Accuracy is found by classifying all images in the test sets, and checking the classified categories against the actual categories.

#### Rank accuracy

Rank accuracy measures the percentage accuracy of the correct category being in the top $K$ classified categories. As a user will be receiving results on their phone, a selection of likely flower categories can presented to that user, who can make the final decision. How rank accuracy changes as we consider more ranks therefore can give us an indication as to how many results to present the user. The more results are presented, the more likely the correct classification will be included, but the more cluttered the presentation is.

$$Rank\ accuracy = \frac{Number\ of\ correctly\ classified\ flowers\ considering\ the\ top\ K\ results}{Total\ number\ of\ flowers} \times 100$$

### 3.4.2 Improving SVM accuracy

#### Mirroring and sampling training set images

Taking mirrors and samples of the images in the test set (Figures 3.2 and 3.3) allows the test set to be expanded. The objective of this is to get the most out of the training set images. Both techniques generate new images from which new feature vectors can be calculated. These can be used to improve the quality of the SVM model.

Only the vertical mirror can be used, as shown in Figure 3.2, as flowers only exhibit vertical symmetry.

Figure 3.2: An example flower shown with its vertical mirror

Figure 3.3: An example flower shown with the images sampled from it

**Mirroring and sampling test images**

For the same reasons as laid out above, mirroring and sampling the test image can improve classification accuracy.

For each image, several image vectors $\boldsymbol{x_i}$ are now being produced. Each one of these is run against the weight matrix to produce several vectors of prediction values $vectp_i$. To classify the flower, these vectors of prediction values msut be combined. This can be done by finding the mean of each corresponding value, or by taking the maximum of each corresponding value:

$$\boldsymbol{x} = \sum_{n=0}^{n} \boldsymbol{x}_i$$

(Add analysis when completed)

### 3.4.3 Results

| | 17 flower dataset | 102 flower dataset |
|---|---|---|
| Baseline | 89.8 $\pm$ 0.68% | 85.1 $\pm$ 0.59% |
| Mirroring | 89.4 $\pm$ 0.82% | 85.5 $\pm$ 0.68% |
| Sampling | 89.7 $\pm$ 0.75% | 84.3 $\pm$ 0.31% |
| Mirroring and Sampling | 90.2 $\pm$ 0.70% | 85.2 $\pm$ 0.41% |

Table 3.1: Mean classification accuracy and standard devaition for the Oxford 17 and 102 flower datasets while using the standard, mirrored and subsampled training image sets

If the images in the test set happen to closely match those in the training set, the classification accuracy would be higher than if the opposite were true. To account for these idiosyncrasies, the script which calculates classification accuracy is run five times. Each time it uses 20 randomly selected images as the training set and the rest of the images as the test set. The mean accuracy and the standard deviation of the results are found. 20 images are used as this is the number of images used in the training set defined in the dataset.

### 3.4.4 Analysis of results

The standard deviations of the results are low, indicating that the accuracy figures are reliable, and the accuracy rates are high. These accuracies are obtained using models trained on subset of the total images available. The final models used for the application are trained on all the images, and should have higher accuracy rates.

The following sections make use of tools for further analysis of results.

#### Confusion matrices

Confusion matrices give a more detailed view of the classification. It presents the classification results as a series of rows. Each row represents a category being classified and the numbers in that row show what percentage of the test images were classified as that category.

For example, for the images tested from category 1 of the 17 flower dataset. 53% were correctly classified as category 1, 3% were classified as category 5, 10% as 8 and 35% as 14.

(add further analysis - talk about how some flowers look like others (e.g. 14 in 17dataset is particularly strong). Also category 10 in the 17 database has 100% accuracy without any other species being misclassified to it. Probably distinctive. Add images of the flowers in question.)

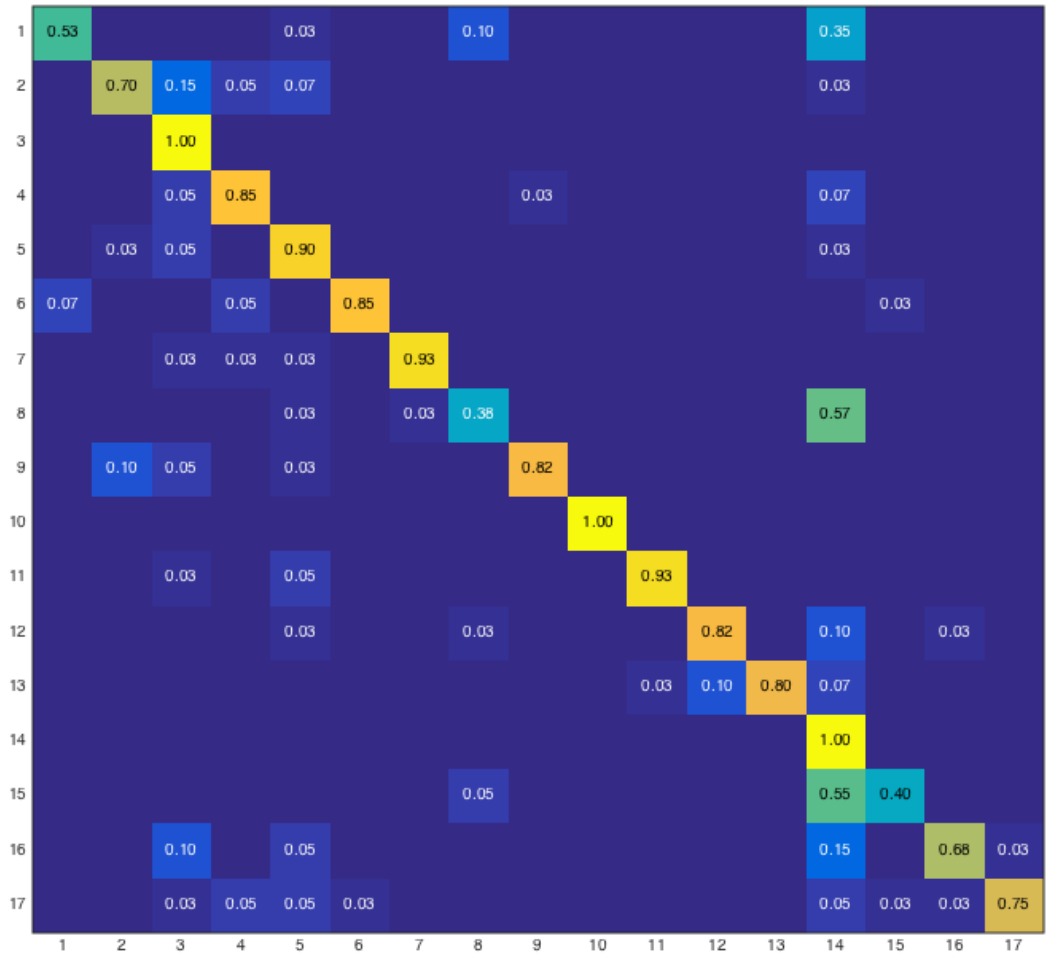



Figure 3.4: Confusion matrix for the 17 flower dataset, obtained using mirroring and sampling

**Rank accuracy**

The rank accuracy graph shows that classification accuracy quickly increases as more ranks are considered. This backs up the presentation of several choices to the user, who makes the final classification choice.

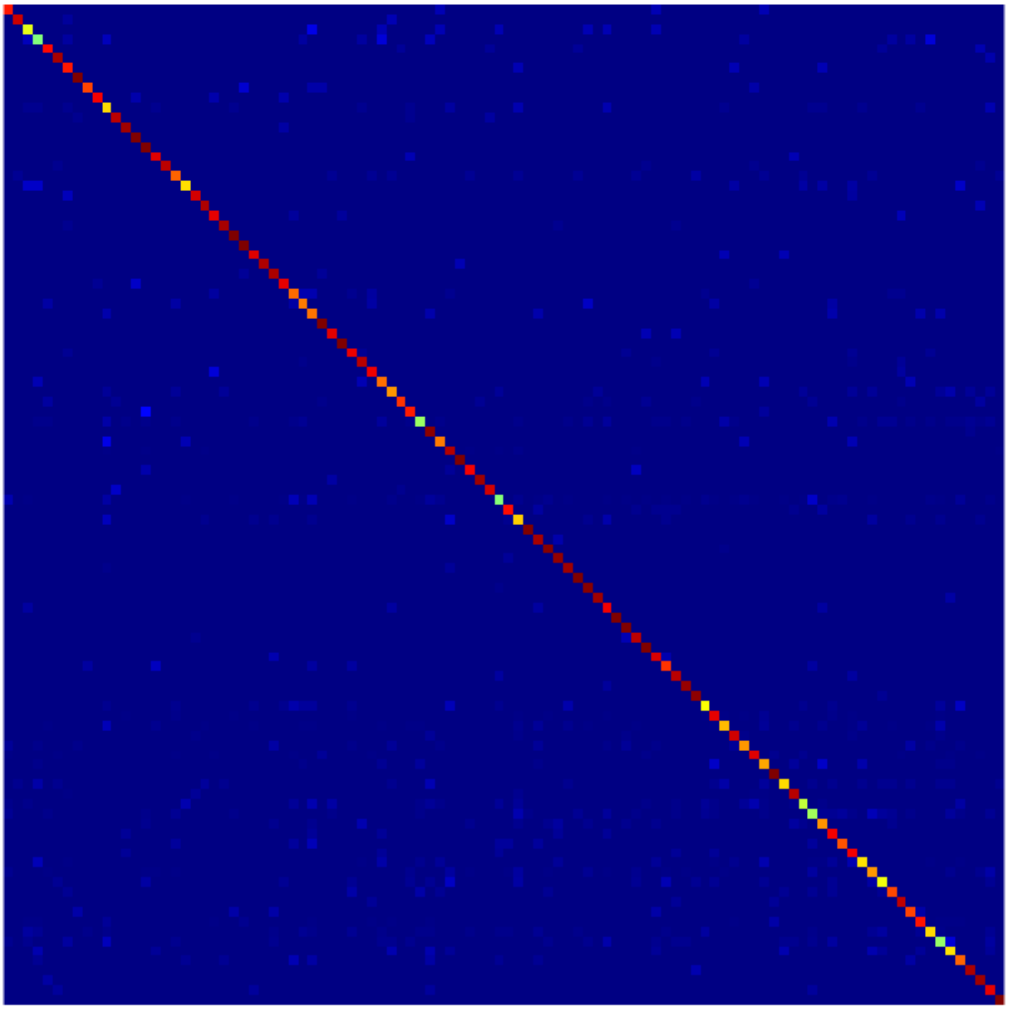


Figure 3.5: Confusion matrix for the 102 flower dataset, obtained using mirroring and sampling

(add rank accuracy for 17 flowers. Add table showing rank accuracy when considering 4-10 ranks, decide on number of ranks to show the user - the more there are the higher the accuracy rate, but the more complex the app experience is.)

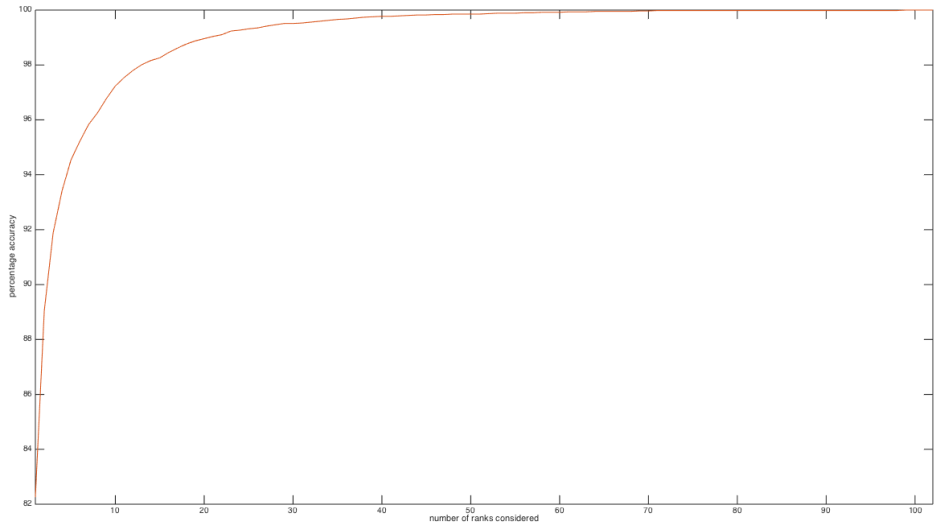



Figure 3.6: Rank accuracy for 102 flower dataset

# Chapter 4

# Client and server architecture

## 4.1 Overview of client and server architecture

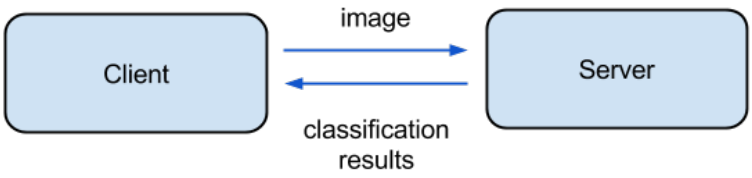



Figure 4.1: The classification system is made up of two parts, a client and a server. The client is an application running on a user's Android phone. The application allows the user to upload a photo to the server. The server performs the classification, before returning the results to the application, where they are displayed.

## 4.2 Client architecture

The client is an Android application which is made up of three activities [4]. Each activity is an object which allows the user to perform a specific task, and has a screen which which the user can interact with it. The functions of the activities are shown in Figure 4.2. Screenshots of the activities are shown in Figure 5.1

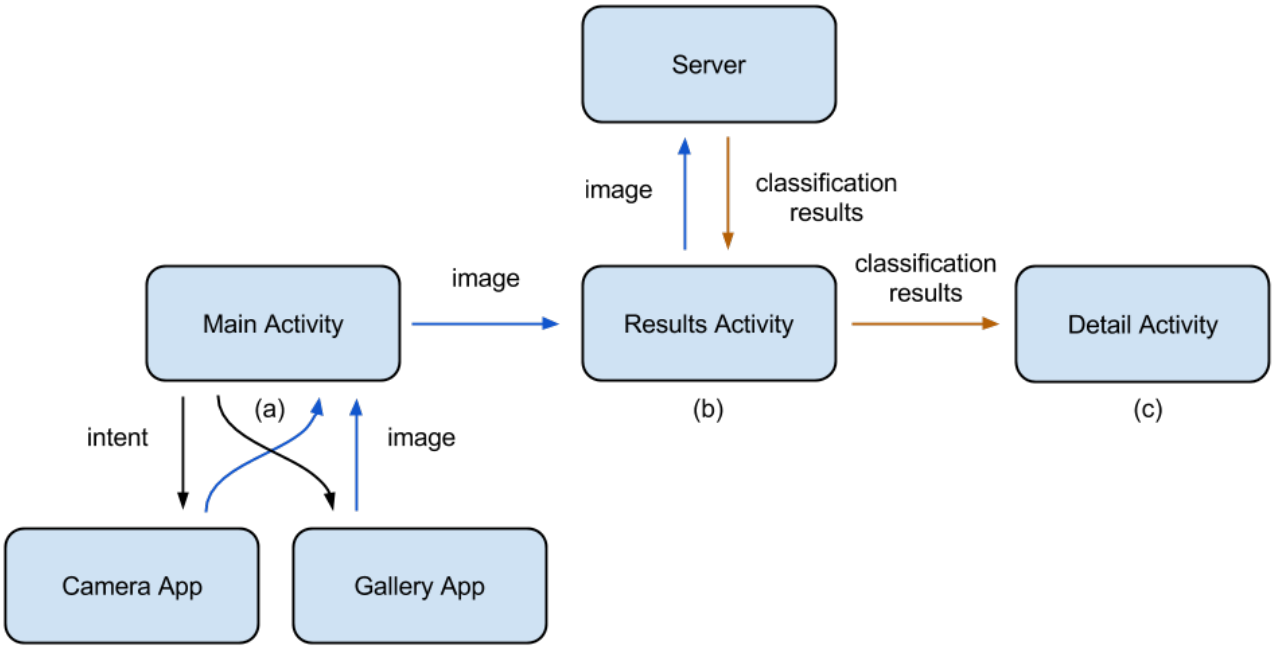



Figure 4.2: Diagram showing the information flow between different sections of the client

### 4.2.1 Taking the photo

After the application is launched, the Main Activity of the application (Figure 4.2 (a)) presents the user with two buttons, which allow the user to take a photo, or upload a photo previously taken from a gallery. Both of these buttons trigger intent calls. Intents are objects native to the Android system, which allow an application to request an action from another app component. The intent call starts the built-in camera or the gallery application, and allows the user to take or pick a photo without having to write that code into the application.

Once a photo is taken or chosen, the second activity of the application (Figure 4.2 (c)) is called, and the filename of the photo is passed to it.

### 4.2.2 Uploading the photo

As the Results Activity (Figure 4.2 (b)) is called, a HTTP connection is opened, connecting to the URL of the server. A HTTP POST request is started, and the photo is uploaded to the server. The filename is appended to the POST request header as the value in a key-value pair. The key is predetermined and is known by the client and server and is used by the server to extract the image from the body of the request.

The object which contains the code to upload the image extends a native Android object called an AsyncTask [5]. The AsyncTask starts a new thread, and runs the upload code on it. This is done because the photo upload can take several seconds. If this were to take place on the user interface thread, user interface features such as scrolling and use of the navigation buttons could not take place until the uploading had done. This makes the application seem like it has frozen and is bad user experience design.

### 4.2.3 Receiving results

Once the photo has been uploaded to the server, and the classification has been carried out, the server responds with a JSON array containing the classification results and information about those flowers.

The array consists of a number of flower objects, each of which contains the flower name, species, the URL of an example image, a snippet of information about the flower from Wikipedia and a links to that flower's Wikipedia page and google search.

The results are placed in a list, shown in Figure 5.1 (b), and the images are downloaded from their URLs using an image downloading and caching library named Picasso [6]. Picasso allows for the downloading, display and cacheing of an image with a single line of code. If an the user scrolls down the list and an image scrolled off the screen, it is destroyed in order to save memory space. If the user scrolls up again, the cacheing implemented by Picasso allows that same image to be loaded from memory, rather than be downloaded again. This improves performance as loading from memory is essentially instantaneous while downloading the image can take a few seconds.

If an image in the list is clicked, a detailed view of that flower is shown (Figure 5.1 (c)). This shows the user extra information about the flower, and presents them with buttons which allow them to Google search the flower or go to the flower's Wikipedia page.

## 4.3 Server architecture

The server side consists of two servers. The first (front end) receives the image from the client, saves it to a folder and passes the filename to the second (back end) server. The second server calls a

Matlab instance, and runs the Matlab classification script on the image. It constructs a JSON array of results and passes it to the first server which returns the results to the client.

This two-server architecture improves security as file input and output is kept separate the classification logic.

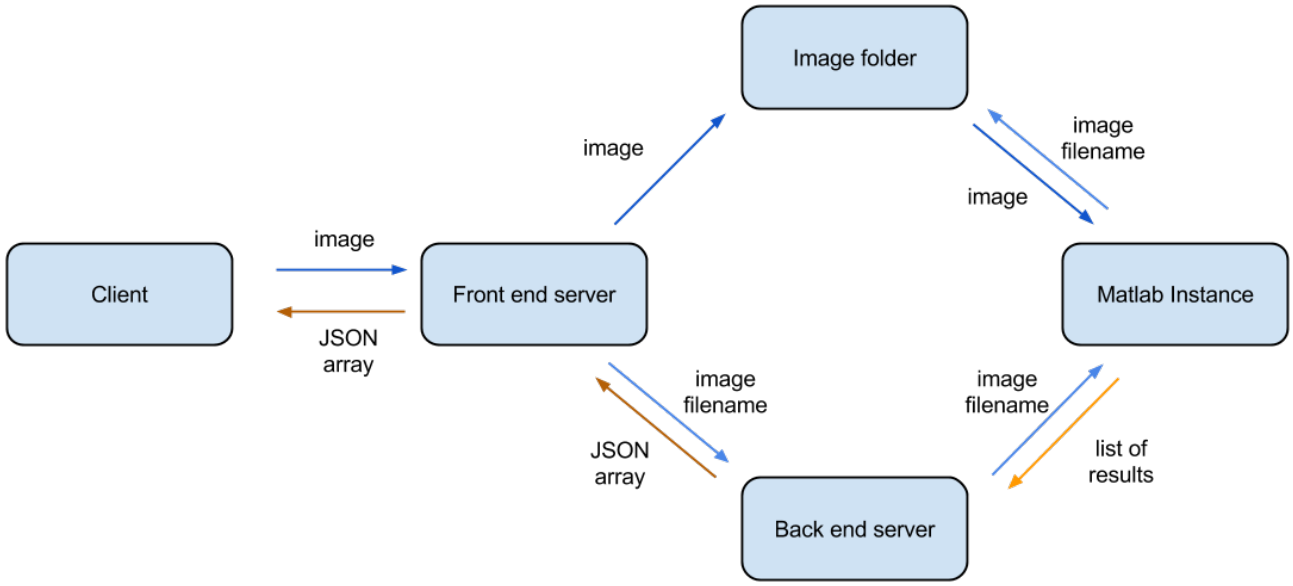



Figure 4.3: Diagram showing information flow between the different sections of the server side

### 4.3.1 Flask server

The front end server runs the Flask server framework [7]. The server sets up a URL and checks it for incoming POST requests. When a POST request is detected, it checks the request header, using a predetermined key, for the filename. It then uses the filename to download the image and save it to the image folder.

Error catching is implemented as the incoming POST requests cannot be assumed to necessarily be benevolent. The filetype is checked and only image files are downloaded. Flask's `secure_filename` function is used to generate filenames for the images before they are saved. The secure filename function stops filenames which could negatively affect computer behaviour, and prevents name clashes.

### 4.3.2 Connection between servers

### 4.3.3 Backend server

The backend server acts as a python wrapper around the Matlab classification code. It uses the python to Matlab bridge Mlwrap [8], which allows Matlab functions as if they were python functions.

Upon starting, the server opens an instance of Matlab, loads the convolutional neural network and the weight matrix $W$. Each of these actions take several seconds but only need to be done once. By completing them upon start, they don't need to be run when a classification request is received, speeding up the classification time.

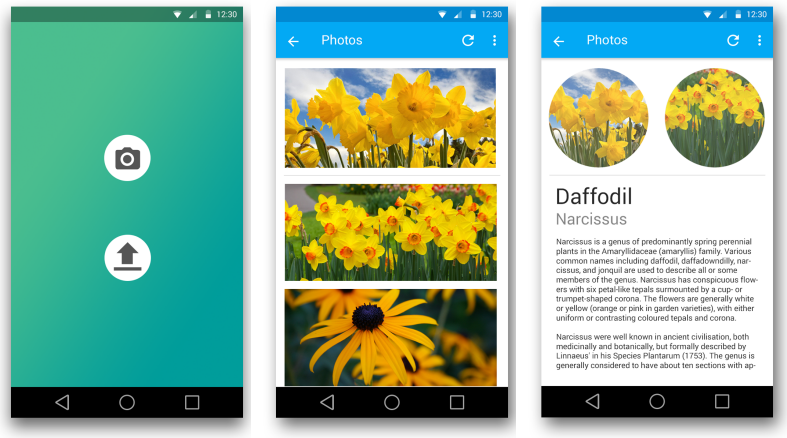
It receives the filename from the front end server, and passes the filename and CNN to the Matlab classification function. This Matlab function returns a list of the category numbers of the eight top classification results.

The backend server then constructs the JSON Array. Lists of flower names, species, image URL, Wikipedia text and Wikipedia and Google search URLs are stored. They are ordered by flower category such that the information for flower $i$ is stored at the $i^{th}$ for of the files. As the flower categories are numbered starting at $1$, the lists are indexed starting at $1$. The server uses the flower category numbers from the Matlab script to look up the appropriate information. This JSON array is passed back to the front end server which passes it to the client.

# Chapter 5

# Application design

## 5.1 User interface design




(a) (b) (c)

Figure 5.1: Screenshots of the screens seen by user. (a): The Main Activity allows the user to choose whether to take or upload a photo. (b): The Results Activity displays the classification results. (c): Detail Activity. If one of the images in (b) is clicked, a detail view of that flower is shown.

The user interface has been optimised by following Android's design guidelines [9]. This results in an Application which looks and behaves like applications the user has seen before, making it familiar and intuitive to use.

### 5.1.1 Material theme

The application implements Android's recently released Material theme [10]. While the full Material theme can only run on devices running the latest version of Android, version 5.0 (API level 21), the look of the theme can be replicated on older devices by using Android's `AppCompat` theme.

### 5.1.2 List and detail views

The results of the classification are presented in a list (Figure 5.1 (b)), showing an image of the flower and the flower's common name. Above the list the photo the user has taken is shown, allowing for easy comparison. The images are cropped such that they are large enough to show the flower without taking up too much screen space. Each item on the list is clickable, and clicking reveals a page which details more information about that flower (Figure 5.1 (c)). This is a common way of presenting data and will be familiar to most users.

### 5.1.3 Button and icon design

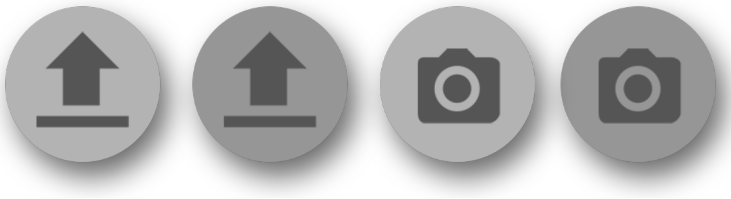


Figure 5.2: The unpressed and pressed versions of the Take Photo and Upload Photo buttons

The take photo and upload photo buttons are custom designed (Figure 5.2), and use icons rather than words for simplicity. The icons used are adapted from the official Android icon pack [11], and will be familiar to Android users. The buttons use flat colour, keeping with the Material theme [10]. Both pressed and unpressed icons have been made, and the change in colour gives visual feedback to the user that the button has been pressed.

(insert icon design when done)

## 5.2 User experience design

User experience has been optimised by focusing on making the classification as fast as possible and the application as simple to use as possible.

### 5.2.1 Simplification

By keeping the application simple, it is easier to use. The application presents the user with a single choice; whether to take a photo or to upload a photo which has been previously taken. Uploading and downloading happens automatically, and no menu options are provided except the ability to share the result of a classification.

### 5.2.2 Flower images

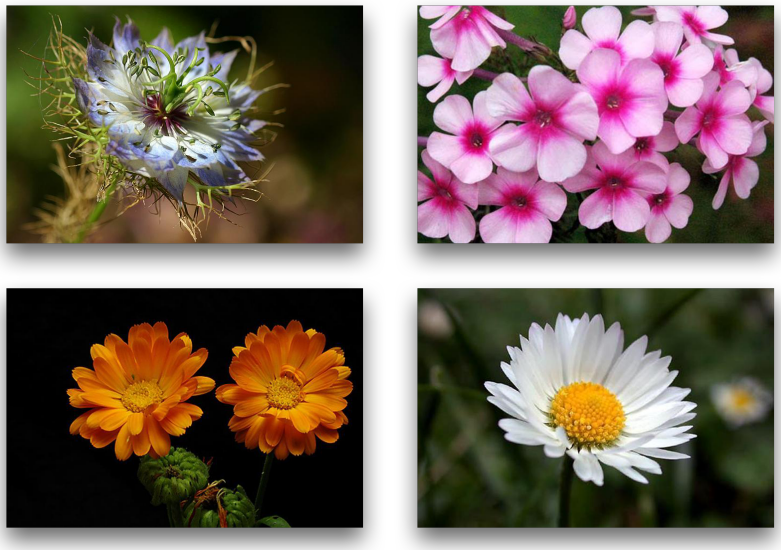


Figure 5.3: A selection of the hand-picked flower images shown to the user

When the results are returned to the client, the images which are downloaded have been hand-picked for their quality. They have been chosen with the criteria of clearly display the flower in question, minimising any confusing background, and beauty. The images are cropped to the dimensions they will be displayed at, minimising the amount of data which needs to be transferred and speeding up the downloading time.

### 5.2.3 No good classification option

It would appear unpolished if the application gave normal classification results for an image which is not a flower. If, after classification, the image does not closely match any of the categories ($p_i < -x \forall i = 1...n$), the user is notified, but the classification results are still provided.

(TODO: run tests to find suitable x)

# Chapter 6

# Conclusions and future work

Mirror of intro. Intro discusses challenges, conclusions describe how challenges were minimised. Were goals achieved

## 6.1 Conclusions

## 6.2 Porting algorithm to Android

# Bibliography


[1] K. Chatfield et al. ``Return of the Devil in the Details: Delving Deep into Convolutional Nets". In: *British Machine Vision Conference*. 2014.

[2] .

[3] .

[4] .

[5] *AsyncTask | Android Developers*. 2015. url: `http://developer.android.com/reference/android/os/AsyncTask.html`.

[6] 2015. url: `http://square.github.io/picasso/`.

[7] .

[8] .

[9] .

[10] .

[11] .