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.)

2.1 Visual Classification Apps

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

2.2 Flower Classification

Look over previous papers on flower classification

2.3 Convolutional Neural Networks

CNNs, where they came from, who invented them, imagenet challenge

Description of LibLinear

Chapter 3

Classification

3.1 Overview of classification pipeline

A photograph of a flower supplied to the classification software. A convolutional neural network is used to produce a feature vector, which describes the content of the photograph. A number of support vector machine classifiers compare the feature vector to previously trained models, and predict the species of flower.

3.2 Convolutional neural network (CNN) inspired feature extraction

The pipeline makes use of the first seven layers of the fast convolutional neural network CNN-F introduced in 'Return of the Devil in the Details: Delving Deep into Convolutional Nets' [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. 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 for quick, large scale classification. 1 vs the rest classification is used. n SVM models are trained for n flower categories, producing n weight vectors, $\vec{w_1}, \vec{w_2}, ..., \vec{w_n}$. These weight vectors are stacked into a weight matrix $\vec{W}$:

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

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

$$\vec{x}.\vec{W} = \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{pmatrix}$$

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 models must be trained, to produce the weight matrix W. 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.

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. The similarity between a new flower and those in the category can be found by comparing the new flower's feature vector and the category's weight vector.

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.

3.3.3 C Parameter

Adjusting the C Parameter of a SVM changes 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.

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 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 j 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.

3.4.2 Improving SVM accuracy

Mirroring of training set images

As most flowers have vertical symmetry, the number of images in a training set can be doubled by vertically mirroring each image in the set. As a new flower image may more closely resemble one of the training set image's mirrors, this results in a higher classification accuracy without needing to expand the training set with new images.

Sampling training set images

The number of images in the training set can be further expanded by sampling several smaller images from each training image. As a new flower image may more closely resemble one of the training set image's samples, this again results in a higher classification accuracy without needing to expand the training set with new images.

Mirroring and sampling test images

For the same reasons as laid out above, mirroring and sampling the test image can improve classification accuracy. As, for each image taken by the user, several images are now being classified, the results from the different classifications can be averaged, or the maximum taken

3.4.3 Results

	17 flower dataset	**102 flower dataset**
Standard		85.1%
Mirroring		85.5%
Sampling		85.4%
Mirroring and Sampling		86.0%

Table 3.1: Mean classification accuracy 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.

	17 flower dataset	**102 flower dataset**
Standard		0.0059%
Mirroring		0.0068%
Sampling		85.4%
Mirroring and Sampling		86.0%

Table 3.2: Standard deviation of classification accuraies for the Oxford 17 and 102 flower datasets while using the standard, mirrored and subsampled training image sets

Analysis of results

Standard Deviation low, indicating good something

Mirror and Sampling have little effect on accuracy - these effects could already be looked at by the CNN

Confusion Matrices allow deeper analysis of results. Certain categories perform badly. This explain why (similar looking flowers? - give photos)

Chapter 4

Client and Server Architecture

(3 pages long)

4.1 Client architecture

Android application allows the user to take the photo and upload it to the server, which performs the classification before sending the results back to the application for display.

4.1.1 Uploading the photo

HTTP connection, Using an Async Task to keep the slow data connection off the UI thread

4.1.2 Receiving results

Description of data sent (name, species, link to image, wikipedia excerpt, links to google and wikipedia) Data sent as JSON string,

4.2 Server architecture

Flask server receives HTTP request, saves photo to a folder. Passes filename to the backend server which runs the classification script, returns a JSON Array of info to the Flask server, which sends it back to the Android client.

4.2.1 Flask server

How server accepts the photo

Error catching (safe filenames, only certain filetypes allowed

4.2.2 Connection between servers

4.2.3 Backend server

Description of how server works

mlwrap

Chapter 5

Application Design

"...describing the design of the app, showing screen shots of the interface as it goes through an example, plust a gallery of results."

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, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, 2014.