Machine Learning Based Flower Recognition System

Utkarsh Tiwari, Rohit Kumar Singh, Rohan Vijay Wargia, P Uttareshwar Vikashrao

Abstract: ***Automatic flower plucking systems for smart agriculture are being studied for many years to support flower harvesting. Such systems require flower recognition task to be integrated as part of the system. This paper presents an approach for classification of flowers using a machine learning algorithm. The method categorizes flowers into different species with the help of convolutional neural networks and deep learning techniques. The system uses a pre-trained CNN model to improve the accuracy rate. Concepts such as Feedforward, back-propagation and transfer learning are used to create the neural network model. Different hyper-parameter values have been tested on the model which provides maximum accuracy of 85.0 percentage on the testing dataset. The result is visualized in the form of bar-plots which provides the top 5 predictions of flower species for the given input image of a flower.***

Index Terms**:** ***Image Recognition, Machine learning, CNN, Feedforward.***

I. INTRODUCTION

Traditional flower recognition tasks are done by botanists. However, they face difficulties to manually recognize the flowers when they have limited knowledge and experience in the field. Because, there are more than 200000 species of flowers in the world, and botanists cannot remember all the species. Ordinary people who want to recognize these flowers have to perform a tedious task of searching through books or even searching through the internet with the help of different search engines. These situations demands for an automated system that detects and recognizes flower species. The importance of building automated flower recognition method stands out in many benefits such as providing fast recognition for harvesting robots. The automated flower recognition gives the people, with limited experience in flower species, the ability to recognize the species of a flower, with the advantages of saving time and effort. Also, recently there has been a demand for flower floriculture which has made it an important commercial trades in agriculture. This system could help in the field of floriculture in a commercial way.

Revised Manuscript Received on April 25, 2019.
Utkarsh Tiwari, REVA University, India.
Rohit Kumar Singh, REVA University, India.
Rohan Vijay Wargia, REVA University, India.
P Uttareshwar Vikashrao REVA University, India.

Image recognition is one of the important concepts in the field of computer vision. As the recognition helps eventually with the classification, we have a lot of classification algorithms such as SVM's, vector machines, artificial neural networks, etc. One of these methods can help to classify flowers into their respective categories by calculating the class label probabilities. There are more than 200000 species of flowers in the world on which proper research and analysis could be performed. In this paper, a machine learning based method is employed for classification of flowers into categories. The method categorizes flowers into different species with the help of convolutional neural networks and deep learning techniques. The system uses a pre-trained CNN model to improve the accuracy rate. Concepts such as Feedforward, back-propagation and transfer learning are used to create the neural network model. Different hyper-parameter values have been tested on the model which provides maximum accuracy of 85.0 percentage on the testing dataset. The result is visualized in the form of bar-plots which provides the top 5 predictions of flower species for the given input image of a flower.

The rest of the paper is organized into 4 sections. Section II describes related work. The machine learning model for flower classification is described in section III. Results and discussions are presented in section IV. Section V concludes the work.

II. RELATED WORK

The following are the research works related to recognition systems.

Oxford17 flower dataset was introduced in [1]. Three types of features were studied independently with Nearest Neighbour classifier. The highest recognition rate for their consist viewpoints is 75%, which was achieved based on the shape feature of the Scale Invariant Feature Transform (SIFT) [2]. The worst recognition rate was achieved based on the Hue Saturation Value (HSV) color feature, it is only 49%. Combing features were also studied in their approach; the results showed that shape and color are the best for flower recognition. Texture feature of convolving the image with Maximum Response Filters (MRF8) of [3] was excluded. The recognition accuracy of this approach on the full oxford17 is 71.76% [4]. Features and testing methodology of [1] were applied by [5]. The

results show that shape is the best feature while color is as good as texture. Accuracy of combining all features was 80.49% with one-versus-one SVM, while it was 82.55% with one-versus-all SVM. The recognition accuracy of using one-versus-all SVM exceeded the recognition accuracy that was achieved by Nearest Neighbour classifier, which was used in [1]. [6] in his paper proposed an improved system for [1,7,8,9]. Combined features from Lab color space and various descriptors for SIFT feature were extracted from foreground only. Linear SVM that takes kernelized feature vectors was used for classification. They provide ground truth segmentation and unsupervised learning algorithm named as Bi-Level Co-Segmentation (BiCoS). The system was tested on Oxford17 dataset, the mean value of the precisions of all species was used as a measure for recognition accuracy. Recognition accuracy for BiCoS approach was 90.4%, and 91.1% for Bi-Level CoSegmentation Multi-Task (BiCoS-MT) approach, which outperforms all previous works. In this paper, a new dataset includes Jordanian flowers, which is not addressed even in the well-known dataset, will be an addition to the automated flower species.

III. METHODOLOGY

The proposed methodology for flower species recognition comprises of various steps namely: Data transformation, Training, and Classification. The block diagram of the proposed model is given in Fig 1. The detailed description of the proposed method is as follows.

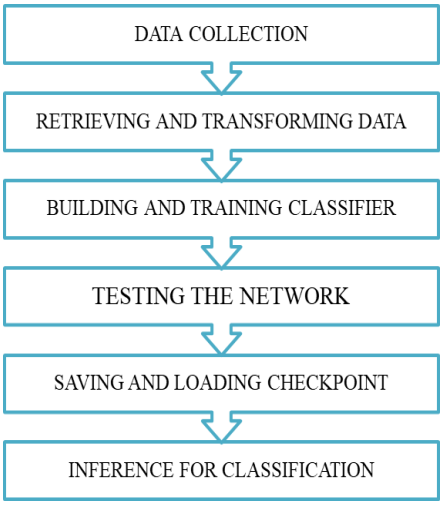

Fig 1: Block Diagram of Flower Recognition System

A. Data collection

Many successful real-world image recognition systems require hundreds or thousands of training examples. Either developing better learning algorithms or increasing training set size has significant potential for improving object classification performance, and they may have an equal impact on practical applications.

When building image recognition systems, data collection usually proceeds by first taking pictures of object instances in their natural environment. This involves either searching for the object in its environment or gathering the object instances beforehand and placing them, in a natural way, in the environment. The quality of the dataset is also heavily dependent on human collection and annotation performance. In our approach, we start by collecting different flower images from a lot of data sources such as Kaggle, etc. or capturing the images manually in different lighting conditions and angles.

B. Retrieving and transforming data

The data obtained after the enhancement of images is split into 3 parts – training, test and validation data. Here **'torch vision'** is used to load the data. For the training, we'll apply transformations such as random scaling, cropping, and flipping. This will help the network generalize leading to better performance. We'll also make sure the input data is resized to 224x224 pixels as required by the pre-trained networks.

The validation and testing sets are used to measure the model's performance on data it hasn't seen yet. For this, any scaling or rotation transformations are not needed, but we'll need to resize then crop the images to the appropriate size.

The pre-trained networks we'll use were trained on the ImageNet dataset where each color channel was normalized separately. For all three sets, we'll normalize the means and standard deviations of the images to what the network expects. For the mean the values [0.485, 0.456, 0.406] and for the standard deviations [0.229, 0.224, 0.225] are calculated from the ImageNet images. These values will shift each color channel to be centered at 0 and range from -1 to 1.

C. Building and training classifier

The working of an artificial neural network is based on two methods:

1. Feedforward
2. Backpropagation

To train the network the backpropagation method is used. The main function of backpropagation is to calculate the error in the model in its current state and to update the weights based on the error equation (1). The error term in backpropagation is defined by the following formula:

$$\delta_j = \sum [w_{jk}\delta_k] f'(h_j) \quad (1)$$

Where-

w_{jk} - defined as the weight between the hidden layers.

∂_k - defined as the error term for the output layer.

∂_j - defined as the error term for the current hidden layer.

Now the data is ready, we'll build and train the classifier. We have used one of the pre-trained models from torchvision models to get the image features. The procedure for training feed-forward classifier is described below.

- Load a pre-trained network (For example, the VGG networks work great and are straightforward to use).
- Define a new, untrained feed-forward

network as a classifier, using ReLU activations and dropout.
- Train the classifier layers using backpropagation using the pre-trained network to get the features.
- Track the loss and accuracy on the validation set to determine the best hyperparameters.

D. Testing the network

We'll test our trained network on test data images, the network has never seen either in training or validation. This will give a good estimate for the model's performance on completely new images. Run the test images through the network and measure the accuracy, the same way to be done for validation. A minimum of 70% accuracy on the test set is to be gained if the model has been trained well.

E. Saving and loading checkpoint

After the network is trained, save the model so we can load it later for making predictions. We will also save the mapping of classes to indices which you get from one of the image datasets: **image_datasets['train'].class_to_idx**. We can attach this to the model as an attribute which makes inference easier later on. We have to completely rebuild the model later so to use it for inference. We have made sure to include any information needed in the checkpoint. If we want to load the model and keep training, we have to save the number of epochs as well as the optimizer state, **optimizer.state_dict**. We can use this trained model in the future for further use. After this, we write a function that can load a checkpoint and rebuild the model. That way we can return to the project and keep working on it without having to retrain the network.

F. Inference for classifier

Here, we have defined a function which uses a trained network for inference. That is, we'll pass an image into the network and predict the class of the flower in the image. The function called predict will take an image and a model, they will return the top K most likely classes along with the probabilities.

But before that, we have to handle processing the input image such that it can be used in your network. We'll use **PIL** to load the image. This function preprocesses the image so that it can be used as input for the model. This function should process the images in the same manner used for training. First, resize the images where the shortest side is 256 pixels, keeping the aspect ratio. This can be done with the thumbnail or 'resize' methods. After that, we'll crop out the center 224x224 portion of the image. Color channels of images are typically encoded as integers 0-255, but the model expected floats 0-1. We'll convert the values, It's easier with a **Numpy** array, which we can get from a PIL image like this:

np_image = np.array(pil_image)

As before, the network expects the images to be normalized in a specific way. For the means, the values are [0.485, 0.456, 0.406] and for the standard deviations the values are [0.229, 0.224, 0.225]. We'll subtract the means from each color channel, then divide by the standard deviation.

IV. RESULTS AND PREDICTIONS

Once we get images in the correct format, we define a function for making predictions with the trained model. A common practice is to predict the top 5 or so (usually called top- K) most probable classes. We'll calculate the class probabilities then find the K largest values. To get the top-K largest values in a tensor use **x.topk(k)**. This method returns both the highest k probabilities and the indices of those probabilities corresponding to the classes. Then we'll convert from these indices to the actual class labels using **class_to_idx** which we have added to the model used to load the data. We'll invert the dictionary to get a mapping from index to class as well.

Now onwards the trained model is ready for making predictions, we'll check to get some good results. We'll use matplotlib to plot the probabilities for the top 5 classes as a bar graph, along with the input image. We'll convert from the class integer encoding to actual flower names with the **cat_to_name.json** file. To show a PyTorch tensor as an image, we'll use the **imshow** function.

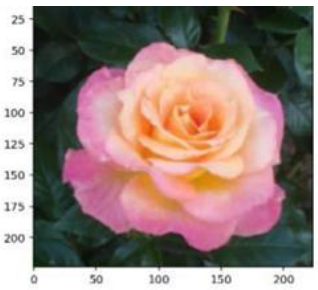

Fig 2.1: Sample flower image 1

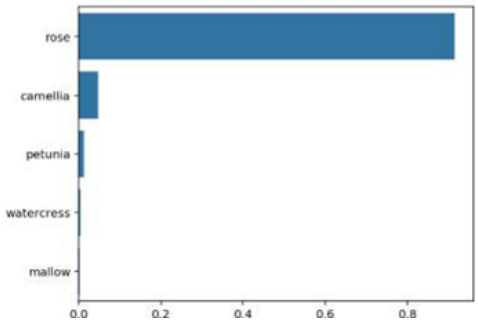

Fig 2.2: Probabilities of flower classes (Flower 1)

Fig 3.1: Sample flower image 2

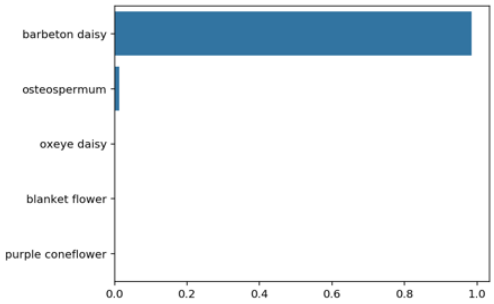

Fig 3.2: Probabilities of flower classes (Flower 2)

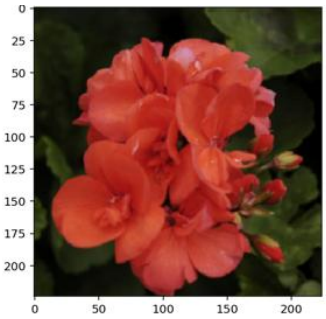

Fig 4.1: Sample flower image 3

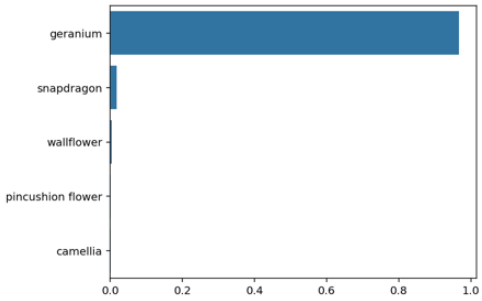

Fig 4.2: Probabilities of flower classes (Flower 3)

Fig 5.1: Sample flower image 4

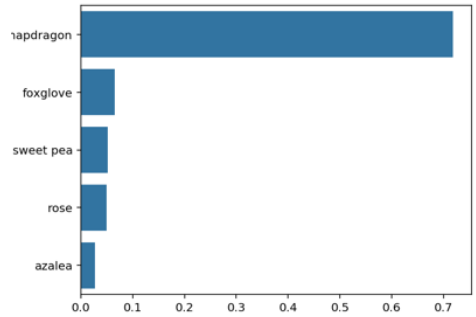

Fig 5.2: Probabilities of flower classes (Flower 4)

V. CONCLUSION

The aim of work is to collect a large amount of raw data from different sources such as Kaggle or GoogleAI which can provide different flower images. Also, we are going to use some of the real-time data collected by our team practically. Our system uses the image processing techniques to synthesize this raw data and provide some enhancement and clarity to the acquired pictures which will be very efficient during neural network model training. So basically, image processing will help our system recognize flower species quite easily.

REFERENCES

1. Nilsback, M. E., & Zisserman, A. (2006). A visual vocabulary for flower classification. Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference, Vol. 2, 1447-1454. IEEE.
2. Lowe, D. G. (2004). Distinctive image features from scale invariant keypoints. International journal of computer vision, 60(2), 91-110. Springer IJCSNS International Journal of Computer Science and Network Security, VOL.17 No.4, April 2017 151.
3. Varma, M., & Zisserman, A. (2002). Classifying images of materials: Achieving viewpoint and illumination independence. Computer Vision—ECCV 2002, 255-271. Springer Berlin Heidelberg.
4. Guru, D. S., Kumar, Y. S., & Manjunath, S. (2011). Textural features in flower classification. Mathematical and Computer Modelling, 54(3), 1030-1036. Elsevier.
5. Varma, M., & Ray, D. (2007, October). Learning the discriminative power-invariance trade-off. Computer Vision, ICCV 2007. IEEE 11th International Conference, 1-8. IEEE.
6. Chai, Y., Lempitsky, V., & Zisserman, A. (2011). BiCoS: A bi-level co-segmentation method for image classification. IEEE. Retrieved January 16, 2016, from http://www.yuningchai.com/docs/iccv2011.pdf.
7. Nilsback, M. E., & Zisserman, A. (2008, December). Automated flower classification over a large number of classes. Computer Vision, Graphics & Image Processing, ICVGIP 08. Sixth Indian Conference, 722-729. IEEE.
8. Nilsback, M. E., & Zisserman, A. (2009). An automatic visual flora: segmentation and classification of flower images. Doctoral dissertation, University of Oxford.
9. Nilsback, M. E., & Zisserman, A. (2010). Delving deeper into the whorl of flower segmentation. Image and Vision Computing, 28(6), 1049-1062. Elsevier.

AUTHORS PROFILE

Utkarsh Tiwari
B.Tech. Final year(Computer Science)
REVA University
Bangalore,India

Rohit Kumar Singh
B.Tech. Final year(Computer Science)
REVA University
Bangalore,India

Rohan Vijay Wargia
B.Tech. Final year(Computer Science)
REVA University
Bangalore,India

P Uttareshwar Vikashrao
B.Tech. Final year(Computer Science)
REVA University
Bangalore,India