ATTEMPT TO FIND NUMBER OF PETALS OF FLOWER IMAGES AND CLASSIFY FLOWERS USING COMPUTER VISION

[1]INDRANIL DATTA

[1]Master of Science
[1]Machine Learning and Artificial Intelligence,
[1]Liverpool John Moores University, UK.

Abstract : It is imperative for researchers, botanists having inquisitive mind for flower species of given unknown flower. Petals of flowers detected through an automated process will be an exciting advantage for them. There were many studies conducted to classify flowers based on hand-crafted methodologies and recently machine learning and deep learning techniques through computers. In this work, few novel ideas are attempted to find out the petals from the flower images using machine learning methods irrespective of its colour, shape or orientation. This information is augmented as a description during flower classification by using deep learning models like ResNet50 and LSTM. The models are trained using images collected from Oxford University's flower image database. Among many different attempts to perceive flower petal counts like k Means algorithm, region based, edge based, watershed transformation and hue based image segmentation, the last one has better accuracy in finding the correct numbers of petals. A robust convolution neural network classifier is built to classify the flowers thereafter. The predicted result includes petals count information derived by the above said method.

Keywords: petals count, classification of flowers, Resnet50, LSTM.

I. INTRODUCTION

Identification of petals count and classification of flowers together are difficult tasks considering wide range of colour differences, similarities and dissimilarities in shape, number of petals, sepals, edge differences and background. Different categories of flowers can share similar features, which are very hard to segregate. Images of flowers contain natural background of many different objects, which cause impediment to categorize the flower. Number of petals identification from a flower image is another complex work with respect to resemblance, accumulation, orientation, shape, colour, texture of petals within the flower. Sometimes petals are almost identical among many different categories of flowers. There is more than 250K of known flower species classified into 350 distinct families using taxonomy [1, 20]. These biological classification methods for flower recognition are time consuming and performed by engaged enriched number of botanists. An attempt to identify number of petals along with the flower automatically and effortlessly using computer vision techniques can bring rich, detailed interesting information about a flower without much domain knowledge in botany. Additionally providing petals information for a flower can reduce the pain of the length and breadth search for the flower name. That might bring extensive interest among general people, researchers and botanists about flowers.

This influences researchers to cater various flower classification techniques using many distinguished machine learning methods. Recently computer vision techniques as in various convolution neural networks have paved the ways to detect flowers due to their abilities to extract relevant features from any image. Also many researchers have demonstrated to identify objects within an image. In fact flowers are also identified by masking techniques. In this work, to proceed beyond these studies, an attempt has been made to find count of petals for specific categories of flowers from the flower images by using machine-learning methodologies.

Hundred and two categories of flower image database having 8,189 images from Oxford University have been chosen for this work. The following chapters will describe the details of the attempted algorithms for petal count from the flower image as well as flower classification. The classification of flowers is experimented with two different ways. First, a LSTM model has been created to correctly identify flower using transfer learning by a ResNet model and next, a ResNet50 model has been trained to identify the flower with petals information calculated earlier.

II. LITERATURE REVIEW

In this section, various research works have been reviewed to distinguish previously performed processes for flower image classifications using handcrafted machine learning and deep learning methods with this research work. Previous methodologies include feature extractions using histogram of Oriented gradients [2], SIFT [3, 7], Gaussian Blur [4], Canny [5], contours detections [6] of the flower images, whereas in deep learning vanilla CNN, RNN, faster RNN, Inception v3 and various custom neural networks are used to classify flower images.

Flower classification using non neural networks

Object detection from the images was very popular from ages. Navneet Dalal and Bill Triggs has presented their work in 2005 [2] for object identification using histograms of oriented gradients (HOG) descriptors to separate the object from the background of the image. In another work, clusters belong to the same object are identified and used for identify the object from the images by DG Lowe in 2004 [3]. Estevao et al. proposed that for segmenting images large variance Gaussian function is needed [4]. Self adaptive edge detection algorithm to detect the edges of the object in the image carried out by Geng Hao et al [5]. Recently another way to detect object is through contours. Single pixel wide contour on every side of the object in a binary image has been detected in 2D images by Leventic et al [6]. At the same time research work has been started on the flower classification as well. Earlier in 2017 research paper based on the colour and variance distribution, image segmentation has been carried out by Avishikta Lodh and Ranjan Parekh [12] on the flower images. Once an image is captured, it has been preprocessed to extract features and converted into training and testing images. Training features (colour and GIST) are first sent through SVM classifier to obtain a predictive model and later verified through test features to predict flowers. These images are further sent through SVM for predicting the flower categories. A machine learning algorithm, t-distributed Stochastic Neighbour Embedding (t-SNE) is used for dimensionality reduction. From this research it has been found that combined features helped to differentiate between interclass similarities and intra class dissimilarities. It has been observed that out of KNN (using Euclidean distance), KNN (city block distance), Logistic regression and SVM, SVM's accuracy is best. Even before that, in 2012, Hong, Soon-Won & Choi, Lynn [17] used colour and edge based contour shape and colour groups with history matching to detect flowers. Flower image classification has been tried by using content-based structural recognition also. Similarly by using SIFT like feature descriptors and feature context flower [18], shape characteristics of flowers and dividing flower images into different feature ring regions [19] classification have been pursued. Also, a nearest neighbour classifier architecture is developed and optimized and evaluated on challenging flower image database. A visual vocabulary was developed to represent various aspects (colour, shape, texture). The novelty lies in the vocabulary used for each aspect, and how these vocabularies are combined into a final classifier. The various stages of the classifier (vocabulary selection and combination) are each optimized on a validation set.

Flower classification using deep convolutional neural networks

But object detection and for the purpose of this reference, flower categorization from images started to be a part of studies of neural networks in the statistical world. Generally in CNN, features were extracted from each of the neurons in the initial layers and the most important features are accumulated through max pooling layers. These important features were recognized through the activation layers for identification of categories of the flowers. In 2018, Hazem Hiary et al. [10] proposed a CNN model to create a feature map. In this architecture, filters/kernel size is learnt through the training process, padding is set to half of the kernel size, and stride is defined to kernel movement over the image. And ReLU is used over the resulted feature maps to increase detection process. In the model some layers are dropped for downsampling the feature maps. With the resulted feature map another fully connected CNN is used with larger receptive fields. This is created for segmenting each flower image to identify the minimum bounding box for each of the flowers. Flower regions have been identified using Fast CNN, Faster CNN and YOLO. Later they used the detected cropped images and these were sent through another customized VGG16 model which is further optimized by SGD to minimize loss. This research is far advanced from the paper presented by Viraj et al. [33] earlier. Viraj et al. used hue based image segmentation for cropping the flower images in order to identify the bonding box of the flower image and later they used a CNN model to classify the flower images. In another study, by using transfer learning with CNN models VGG-16, VGG-19, Inception-v3 and ResNet50, Yong Wu et al. [11] proposed a model to classify flower images which has solved local optimal and overfitting problems. The convolution layer fetched low-level features and thereafter more abstract and complex semantic features which are passed to a pooling layer. Few network parameters are dropped and finally one or more fully connected layers are added to the network to get different class information. The last layer is connected to a output layer and the output is converted to probability distribution through Softmax layer. VGG16, VGG19, Resnet50, Inception-v3 CNN models are used in this regard. These 4 models have pre-trainied the model on ImageNet dataset and transferred the model parameters to oxford flower dataset. The weights of these models have been transferred to the original data. Even Inception v3 model created and transfer learning technology is used to retain flower category dataset, that improves greatly the accuracy of flower classification [13]. In fact based on low level features like color and texture [14] are used to define and describe the image content. Colour features are extracted using normalized colour histogram and texture features are extracted using gray-level co-occurrence matrix. The finding from the study reveals that the number of images generated to represent each type of flower influences the classification accuracy. Another the deep learning network [15] is constructed by repeating a number of building blocks of neurons and aggregates a set of transformations with the same topology. This design results in a simple homogeneous architecture that has few hyper parameters to set. In 2015 NIPS paper [35] FAIR researchers came up with deep masking approach. In this study, they demonstrated image segmentation by deep learning based instance segmentation. There, from the input image patch features are extraction by VGGNet. The fully connected layers and the last max pooling layer in VGGNet are removed. The output is connected to two paths after VGGNet. The first path is to predict the segmentation mask which is class agnostic. And the second path is to assign a score on the likeliness of the presence of the patch contained in the image.

Motivation for the work

In the reviews as described earlier, there are many object detection techniques proposed. Flower regions from the flower can be detected by using these techniques. More details about a particular flower are yet to be identified by these machine learning methods. For the continuation of the research momentum on the flower images, petals count identification has been challenged in this work. With the invent of different CNN models, and due to availability of high speed computers, object detection has been conducted using plenty of deep neural networks. Flower image classification has been also proposed by applying some of these methods to extract flower region from the actual image. And thereafter a deep neural network is applied to detect the flower. It can be noted from the above reviews that earlier handcrafted approaches are used for training the model. Later when deep learning approaches become popular features are extracted by well-crafted neural networks. In most of the deep learning methodologies, two steps approaches have been adopted. These are extraction of features and training a specialized model for training and evaluating the flowers. In this research work, petal count from the flowers has been calculated in order to provide more information about the identified flower. Also, instead of two step approach, a pre trained ResNet50 model can be projected for this purpose. In this work, it is evaluated if ResNet50 model can be taken for the classification of flower images alone without extracting the features beforehand. Also this study is also vigilant about the number of epochs need to train and evaluate the model. So, the actual image of the flower is not masked to get the features before training the model. In this study, how the flower along with its petals information can be presented has been taken as the main focus and that is where it is an extended part from researches where only flower image recognition was a priority. It can also be noted that this study is also focused to perform prediction of flower images with minimum number of epochs. This detail information is an attempt to provide quick benefit to interested people, paleontologists, and botanists in their work and encourage people to get inclined towards knowing flowers in no time.

III. RESEARCH METHODOLOGY

In this chapter, the approaches of the research are explained elaborately on the calculation of petal counts and classification of flower images thereafter.

Calculation of count of flower petals

Counting petals from a flower image depends on the flower orientation, background, size and shape of the flower petals along with the flower as a whole. Following approaches attempted to calculate petals from a flower image.

Using K Means Algorithm for binary segmentation

One of the approaches is that, segment the image by segregating the background from the flower. K-Means is one of the easiest methods to perform this. This algorithm is used to cluster the image into two segments. Internally the algorithm uses mechanism by minimizing a criteria known as inertia or within-cluster sum-of-square. Within cluster sum of square is the measurement of the variability of the pixel values within each image. The pixels are divided into K disjoint clusters C. Each of the pixels is described by the mean µi which is one of the cluster centroids. Once K-means is applied to the images, from its cluster centers the image data of the labels has been taken out. This image is saved as masked image for further processing.

Image Thresholding Using Otsu's algorithm and find specific regional area

After applying K-Means the masked images still have noise. To reduce the noise thresholding is applied on the image. Otsu's algorithm is used to perform automatic image thresholding. The algorithm segregates image into two different classes based on a threshold value. Once Otsu's thresholding is applied to the gray image, the holes with in the image are filled by binary dilations. The connected regions are labelled from the array of the image pixels. The areas of the labelled image regions are measured. If the area of the region is less than a specific value (limiting area) then consider it a petal area. This limiting area is decided based on multiple experiments. The number of petals in the flower is decided to be the areas which match this criterion. Figure shown below demonstrates number of petals count calculated using above method.

Using Sobel filter and Watershed transformation morphology

It can be observed that the above method 3.1.2 could not able to solve the problem correctly as the image is not completely getting rid of the background. Hence another method using edge detection technique is tried. In this method first Sobel filter is applied on the flower image and later the image is morphed by region based watershed transformation in order to calculate the petals count.

Using filtered images segmented based on Hue

As the above two attempted methods do not give satisfactory results. Hence another approach has been taken to calculate petals count on the segmented flower image. In this approach, image segmentation has been performed based on hue as proposed by Viraj et al. [34]. And cropped gray image has been created. Based on hierarchy contours with simple chains criteria all the contours are found out from that cropped image. The bounding rectangular area has been found out for those contours. The contour whose width of the bonding box is lesser than the image size, but bigger than any other contours has been considered as the actual flower contour. Contour area of that contour has been calculated. Extreme left, right, bottom contour points are also calculated from the contour. The center of the contour is also found out. Two lines have been drawn using left, right and top, down points. And another two lines are drawn from the center to left and top. The area of the triangle has been formed by considering three points i.e. the left, center and top extreme contour points. Number of petals is calculated based on the following formula
The extreme left, top, right and bottom points are calculated as follows.
extLeft = $C[0]_{Argmin}$, extRight = $C[0]_{Argmax}$
extTop = $C[1]_{Argmin}$, extBot = $C[1]_{Argmax}$
where C is the contour of the bounding box of the flower as shown below by red line.
A triangle is formed using contour center and extreme left and extreme top points. The sides of this triangle are calculated as follows

side1 = $\sqrt{(extLeft[0] - extTop[0])^2 + (extLeft[0] - extTop[0])^2}$
side2 = $\sqrt{(extLeft[0] - centerX[0])^2 + (extLeft[0] - centerY[0])^2}$
side3 = $\sqrt{(extTop[0] - centerX[0])^2 + (extTop[0] - centerY[0])^2}$
semi_perimeter = (side1 + side2 + side3)/2
triangle_area= $\sqrt{semi_{perimeter} * (semi_{perimeter} - side1) * (semi_{perimeter} - side2) * (semi_{perimeter} * side3)}$
num_of_petals = $\lceil area_{contour}/area_{triangle} \rceil$ as the contour area divided by the area formed by the triangle area. The petals information is stored in a csv file for further processing. Below are the few images of detected contour and petals count database created from this method.

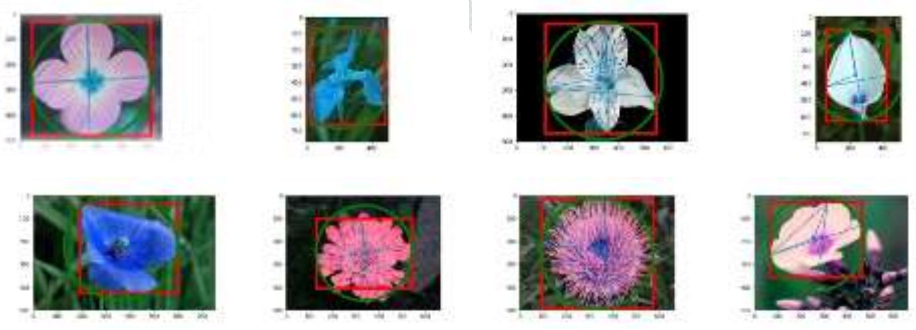


Figure 0.1 Detected contours by red circular region and connected extreme points

Deep Learning methods for identifying flower images

The challenging factors of identifying any object in any image is its background, orientation, contrast, shape, colour and size. It can be very difficult to extract such high-level, abstract features from raw pixels in an image. The mapping of the function from the pixels of the image to create an identity of an object is very complex. This mapping cannot be learnt directly easily. Hence, using only machine learning techniques, it is very difficult and complex to discover the mapping from such kinds of representation to output the features or the variations that can identify the object.

Deep learning solves this problem by separating the required mapping into series of nested simple mappings. Each of these mappings is described by a new layer of the model [25]. The term "deep learning" is to emphasize that researchers can able to train deeper neural networks.

Basics of Convolutional Neural Network

CNN is introduced by LeCun et al. [26] and it was enhanced in the paper [27]. Their multi layered artificial neural network can classify handwritten number and it can be trained with the backpropagation algorithm [27].

In paper [26], the following mathematical equations are presented.

$Y^P = F(Z^P, W)$ where Z^P is the pth input pattern, W is the collection of adjustable parameters in the system, Y^P is the output class level as a score or output probability of the input Z^P

A l**oss** function $E^P = Ɗ(D^P, F(Z^P, W))$ calculates the difference between correct output for the pattern D^P and output produced from the system. $E_{Train}(W)$ is the average of the errors over the training set $\{(Z^1, D^1),.....(Z^P, D^P)\}$.

The training problem is to find the W that minimizes $E_{Train}(W)$. The performance is estimated by another set of samples outside of the training set. That set is called test set. The difference between the expected error on the test set E_{Test} and the training set E_{Train} is approximately $E_{Test} - E_{Train} = \kappa (h/P)^{\alpha}$ where P is the number of training samples, κ is a constant, *h* is measure of capacity or complexity of the machine, $0.5 < \alpha < 1.0$.The structural risk minimization is implemented by reducing $E_{Train} + \beta\, H(W)$where H(W) is the regularization term and β is a constant.

Minimizing H(W) will limit the accessibility of the subset of the parameters space and so it will control the trade off between minimizing the training error and minimizing the expected difference between training error and test error.

Forward Propagation

The loss function can be minimized by estimating the delta variations of the parameter values on the loss function. This is measured by gradient of loss function. In the gradient descent algorithm W is iteratively adjusted by

$W_\kappa = W_{\kappa-1} - \varepsilon\, \partial E(W)/\partial W$ where ε is the scalar constant.

In case of stochastic gradient algorithm, $W_\kappa = W_{\kappa-1} - \varepsilon\, \partial E^{P(k)}(W)/\partial W$ considering an approximated average gradient. With this calculation the parameter vector may converge faster than regular gradient descent. The reason for faster convergence is as follows. In case of first gradient descent algorithm, it is a batch gradient. In this case, gradients are accumulated over the entire training set and thereafter the parameters are updated. In case of stochastic gradient descent gradient is evaluated on the basis of one single training sample or a small number of samples and the parameters are updated using the approximated gradient. But due to presence of redundant data, the difference or the gradient does not get updated and hence faster convergence.

Back propagation

It is the computation of gradients from output to input. This is introduced due to following reasons. First, the presence of local minima in the loss function is not a major problem, back propagation can solve complex learning tasks and computing gradient for a non-linear system needs to be simple.In a system of repeated modules, each of which implements the function

$X_n = F_n(W_n - X_{n-1})$ where X_n is the output vector of the module, W_n is the weight vector and X_{n-1} is the input vector. If the partial derivative E_P with respect to W_n is known then using backward reference the following derivatives can be calculated.

$$\frac{\partial E^p}{\partial W_n} = \frac{\partial F}{\partial W}(W_n, X_{n-1})\frac{\partial E^p}{\partial X_n}$$
$$\frac{\partial E^p}{\partial X_{n-1}} = \frac{\partial F}{\partial X}(W_n, X_{n-1})\frac{\partial E^p}{\partial X_n}$$

Where $(\partial F/\partial W)(W_n, X_{n-1})$ is the function F with respect to weight W calculated for the point(Wn, X_{n-1}). The first equation calculates gradients of $E^P(W)$ and the second equation creates a backward propagation for the neural network.

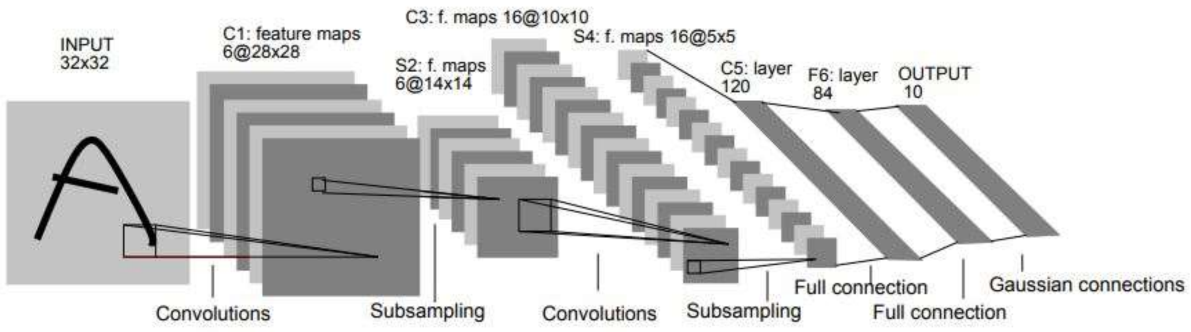


Figure 0.2 Simple Convolutional Neural Network

Deep Learning Models

To gain highest accuracy in a multi-class classification problem raises several problems. Model averaging or ensemble of DNNs increase amount of computation time. The reported accuracy is biased on a particular sampling technique and ensemble size. It also affects power consumption, latency and resource utilization. The below figures are depicting accuracies on different ImageNet models and operations. The below image net model accuracies has been referenced from the research work of Alfredo et al. [29].

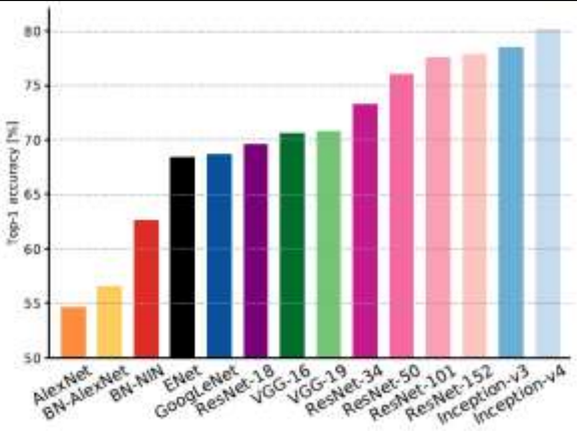


Figure 0.3 ImageNet Model Accuracies

Chris & Koss et al. [36] has demonstrated the accuracies of different models starting from AlexNet to ENet.

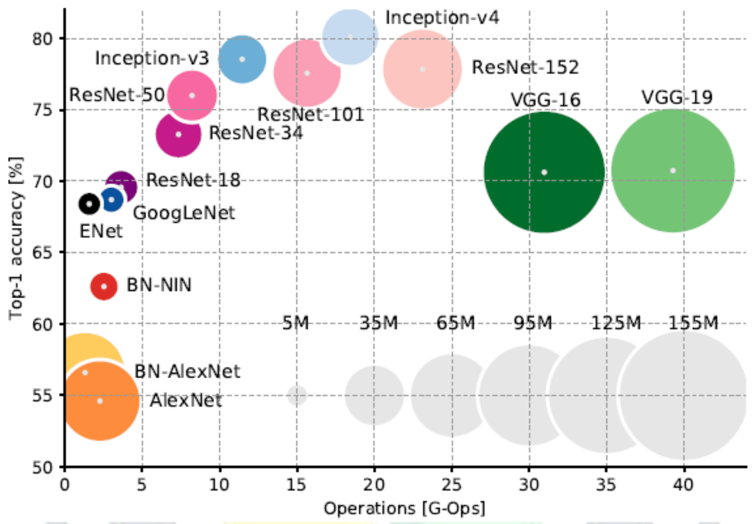


Figure 0.4 Accuracy based on operations of DNN models

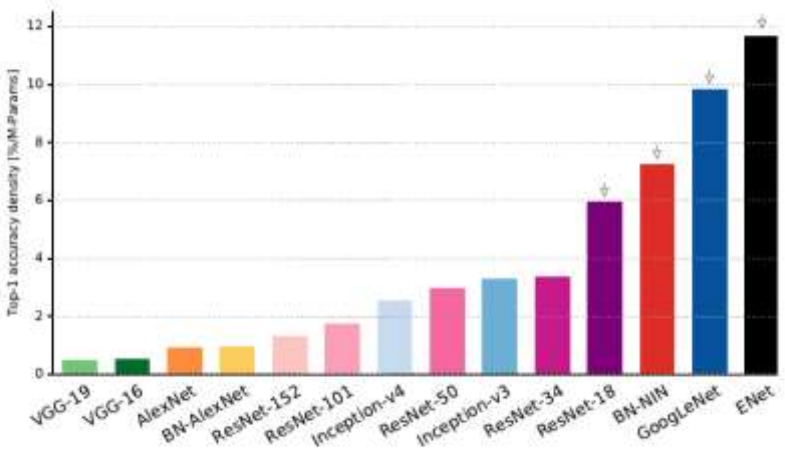


Figure 0.5 Accuracy of models in terms of Parameter reduction

VGG is most expensive architecture in terms of computational requirement and number of parameters. ENet is the best architecture considering parameters space utilization and reducing up to 13 X more information [29] compared to others. But in this research due to constraint on the system configuration, ResNet50 has been considered.

Universal Approximation Theorem and activation functions

According to Universal Approximation Theorem [30] a feed forward network which uses a single hidden layer that connects a finite number of neurons can approximately make a continuous function on subset of Euclidean space under few assumptions on the activation function. So a simple neural network may have wide range of functions, given appropriate parameters.
Hence the approximation function is
$F(x) = \sum_{i=1}^{N} \vartheta_i \varphi(\omega^T_i x + b_i)$ where φ is the activation function, ϑ_i, b_i are real constants, ω_i are real vectors.
The non linear activation function φ are Sigmoid or Logistics Activation function.

Sigmoid function is used when required output is in the range of (0, 1).
Softmax Activation Function

$\Phi = \frac{e^{zi}}{\sum_{j=1}^{K} e^{zj}}$ where K is the real input number, z is the input vector

In case of multi class classification, softmax activation function is used. In the last layer before output layer i.e. (n – 1)th layer where n is the total number of layers in the network, the values get multiplied by weights(ω_i) passed through this activation function and aggregated into a vector where each output class has a value. In fact softmax is a arg max function which returns position of the largest values.
In this case the output ranges from -1 to 1 and this function can be used for classification between two classes.
It is defined as y = max (0, x) where x is the input.
This means y is zero if x < 0 and y = x if x >= 0.
In this function all negative inputs becomes zero that causes problems to train the model properly. Leaky ReLU is y = 0.01x when x < 0 and parametric ReLU is y = ax where x < 0 and a is the parameter which neural network decides by itself.
In case of ReLU there is a dying ReLU problem due to being zero for all negative values. It happens when learning rate is too high or there is a large negative bias. This problem is solved by ELU. In case of large negative values it saturates and allows them to be inactive.
SELU is a modified version of ELU can be defined by

$$f(\alpha, x) = \lambda \begin{cases} \alpha(e^{x} - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

where α is the constant and when λ > 1 the gradient also becomes greater than 1 and the activation function can increase the variance.
Concatenated ReLU In this ReLU activation function, ReLU and negative ReLU are concatenated together i.e. for positive *x* output is *[x, 0]* and for negative *x*, output becomes *[0, x]*. Therefore it makes the output dimention double.

Regularization Terms for deep neural networks

Generalization is the ability of the algorithms to perform well on new, previously unseen inputs.
Regularization is any modification made to a learning algorithm that is intended to reduce its generalization error but not to its training error. Training error is the computation of some error during training of the algorithm with the training dataset.
Generalization error or test error is defined as the expected value error on a new input. Underfitting happens when the model is unable to calculate a sufficiently low error value on the training set. Overfitting happens when the difference between the training error and test error is too high. Occam's razor principle states that we should choose the "simplest" one from competing hypotheses that bring known observations same. The theory of Vapnik-Chervonenkis dimension or VC dimension measures the capacity of a binary classifier. It is defined as being the largest possible value of m for which there exists a training set of m different x points that the classifier can label arbitrarily. According to no free lunch theorem, every classification algorithm has the same error rate when classifying previously unobserved points on an average for all possible data-generating distributions. Therefore there is no best machine learning algorithm and no best form of regularization. Hyperparameters are settings that we can use to control the algorithm's behavior. Hyperparameters control model capacity. A point estimator or statistic is any function of the data that provides best prediction.
$\bar{\theta}_m = g(x(1), \ldots, x(m))$ where x(1), …, x (m) be a set of m independent and identically distributed (IID) data points.
Function estimation is the estimation of the relationship between input and target variables.
Bias of an estimator is the expectation is over the data. **bias($\bar{\theta}_m$) = E($\bar{\theta}_m$) – θ** where θ is the true value of data generating estimation.
In deep neural networks regularized objective function by
$\hat{J}(\theta; X, y) = J(\theta; X, y) + \alpha\, \Omega(\theta)$ where α ϵ [0, ∞ is a hyperparameter.
To control L2 parameter norm penalty or weight decay,ridge regression or Tikhonov regularization is applied as follows

$$L_2 = (wx + b - y)^2 + \lambda w^2$$

where w is the weight matrix, b is the bias vector, y is the output and λ is the regularization. Another way to penalize the size of the model parameter is L1 regularization or Lasso is

$$L_1 = (wx + b - y)^2 + \lambda|w|$$

Early Stopping is used to determine the best amount of time to train. This is required when training large models which has sufficient representational capacity to overfit the task, the training error goes down steadily over time, but validation error starts to rise again.Bagging is a technique for reducing generalized error by combining several models. Dropout is the inexpensive approximation to training and evaluating a bagged ensemble of exponentially many neural network.

Optimization Terms and algorithms for deep neural networks

Optimization is a task of either minimizing or maximizing some function f(x) by changing x. This function is called as objective function. For minimizing the function, it is called cost function, loss function or error function. Gradient descent is reducing a function in small steps by changing the input with opposite sign of the derivative. A local minima is a point where the input function is lower than at all neighboring points so that it is not possible further to decrease the function. A local maxima is a point where the function is higher than at all neighboring points so it is not possible further to increase the function. The critical points which are neither maxima or minima are known as saddle points. The point for which the function has absolute value is called a global minima. To minimize a function, the following the directional derivative can be used to find the direction in which the function is decreasing:

$$\min_{u,u^{T}u=1} u^{T}\nabla_{x} f(x) = \min_{u,u^{T}u=1} \|u\|_2 \|\nabla_x f(x)\|_2 \cos\theta$$

where $\boxed{u}$ a unit vector, $\|u\|_2$ is the L² form of unit vector, $\nabla_x f(x)||_2$ is the L² form of gradient of f(x) with respect to x.
This means gradient points directly uphill and the negative gradient points directly downhill.

Method of Gradient decent

$x` = x - \ell r \nabla_x f(x)$ where ℓr is the learning rate

Batch Gradient Descent

When we add up over all gradient descent on each iteration when performing updates to the parameters, it is called Batch Gradient Descent. $w = w - \alpha \nabla_w J(w)$ In this case it is guranteed to converge to global minimum if the loss function is convex and to a local minimum if the loss function is not convex.

Minibatch Gradient descent

Based on the batch size if gradient descent is added up instead of adding up all of them

$$w = w - \alpha \nabla_w J(x^{[i:i+b]}, y^{[i:i+b]}; w)$$

In this case batch size is chosen usually a power of 2 i.e. 32, 64, 128, 256 considering GPU runtime efficiency.

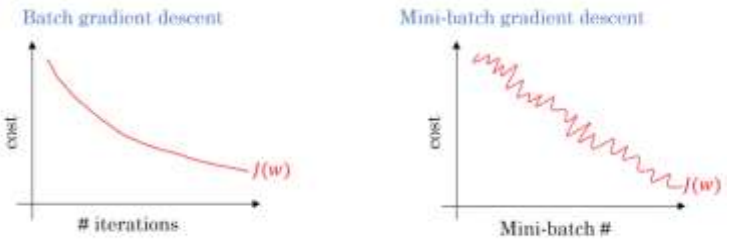


Figure 0.6 Gradient Descent for Batch Size and Minibatch size

Convolutional Neural Networks

Convolutional neural networks are a special class of neural network architecture which is used to handle data with some spatial topology. Unlike vanilla neural network, it uses specific local connectivity and weight sharing. Convolutional network consists of two special type of layers known as convolution layer and pooling layer. Convolution layer is used to extract features, where pooling layer is used to bring shift and distortion invariance. The non-linearity is incorporated in the network by applying activation functions such as ReLU, tanh, sigmoid etc. after the convolution layer and finally dense layer is applied at the end of the network for classification and regression task. The layers used in ConvNets are described below.

Input Layer: Input in CNN is a multi-dimensional array (i.e. a tensor). For example, a 256 * 256 colour image is a 256 * 256 * 3 tensor (for 3 colour channels red, green, blue).

Convolutional Layer: In this layer input is convolved with filters, results in activation map. High value regions in activation map indicates, receptive field at the input corresponding to those positions, matches with the filter. So each filter extracts a particular type of features from the input. Filter dimension depends on the type of data the CNN is handling. Filters are the weights of the network which we learn through backpropagation. While in forward pass we convolve the filter with input, at backward pass we do cross-correlation instead of convolution.

For a convolution layer, four hyper-parameters are there. These are filters K, the spatial extent of the filter F, the stride S and the zero padding amount P.

An input volume of size W1 x H1 x D1 produces a volume of size W2 x H2 x D2, where $W2 = (W1 - F + P)/S + 1$, $H2 = (H1 - F + P)/S + 1$, $D2 = K$

Pooling Layer:

In pooling layer, a region at input is replaced by a single pixel value at output. When the representative single pixel value at output is maximum of the input region its call max pooling. Similarly average pooling or L2 norm pooling can be achieved. The hyperparameters of this layer are their spatial extent F and the stride S. The resulting operation produces a volume of size W2 x H2 x D2 where $W2 = (W1 - F)/S + 1$, $H2 = (H1 - F)/S + 1$, $D2 = D1$

Fully-connected layer:

Neurons in a fully connected layer is connected to all neurons in previous layer though weights, as seen in regular neural networks. The backward pass for FC layer can be done similar to CONV layer, where the spatial extent of the filter is the same as that of input volume.

Activation layer:

This layer is used to introduce non-linearity to the function approximation, which is operated element-wise. Some of the popularly used activation function for ConvNet is RELU, tanh and sigmoid.

Recurrent Neural Networks

RNN is a family of neural network for processing sequential data. In many practical problems input and output spaces contain sequences. For example video can be considered as a sequence of images, sentence can be considered as sequence of words each of which is encoded in one-hot vector form. A recurrent neural network(RNN) has a cyclic connection that processes a sequence of vectors $\{x1, \dots \dots \dots, xT\}$ g using a recurrence formula of the form $ht = f\theta(ht - 1, xt)$, where f is a function we willdescribe later in more details and θ is parameters shared over all time steps. The hidden vector ht can be considered as a memory which contains the summary of all the past inputs. A common practice is to use initial hidden state $h0 = 0$ or to consider them as parameters and learn them. The precise mathematical form of the recurrence $(ht - 1, xt) \rightarrow ht$ varies from model to model and we describe these details next.

Vanilla Recurrent Neural Network (RNN):

Uses a recurrence of the form $ht = tanh\,(W_{xh}x_t + W_{hh}h_{t-1})$ where the previous hidden memory$h_{t-1}$and current inputx_t is concatenated and linearly transformed using matrixW_{xh} and W_{hh}. If inputx_t is of D dimension, hidden memory ht is of H dimension then matrix W_{hh} and W_{xh} are of size H x H and H x D respectively. Common practice for nonlinearity is to use tanh or ReLU. For regression tasks (e.g. time series prediction) hidden layer (ht) is transformto output layer followed by linear activation function. In case of classification problems (e.g. language modelling) hidden layer (ht) is transformed to output layer followed by softmax activation function.

Forward Pass: Consider a length T input sequence x presented to an RNN with d input units, H hidden units, and K output units. Letx_i^tbe the valueof input i at time t, and leta_j^t and b_j^tbe respectively the network input to unit jat time t and the activation of unit j at time t. Let f be the activation function.

$$a_h^t = \sum_{i=1}^{D} w_{ih} x_i^t + \sum_{h'=1}^{H} w_{h'h}\, b_{h'}^{t-1}$$

$$b_h^t = f\,(a_h^t) \Rightarrow \frac{\partial b_h^t}{\partial a_h^t} = f'(a_h^t)$$

where $W_1 = \{w_{ih}\}$is the input-hidden weight matrix and $W_2 = \{w_{h\prime h}\}$is thehidden-hidden weight matrix and $W_3 = \{w_{hk}\}$is the hidden-output matrix.

Training RNN

Since the network is unfolded in time space, gradients has to flow backwards through time as well as layers. BPTT is a technique to find the gradients in RNN, where Back Propagation is applied on the unrolled network. Total loss/error on a given task is the sum of per-time loss

$$L(z, y) = \sum_{t=1}^{T} L_t = \sum_{t=1}^{T} L(z_{t,} y_{t)}$$

where, Lt is the error at time step t.

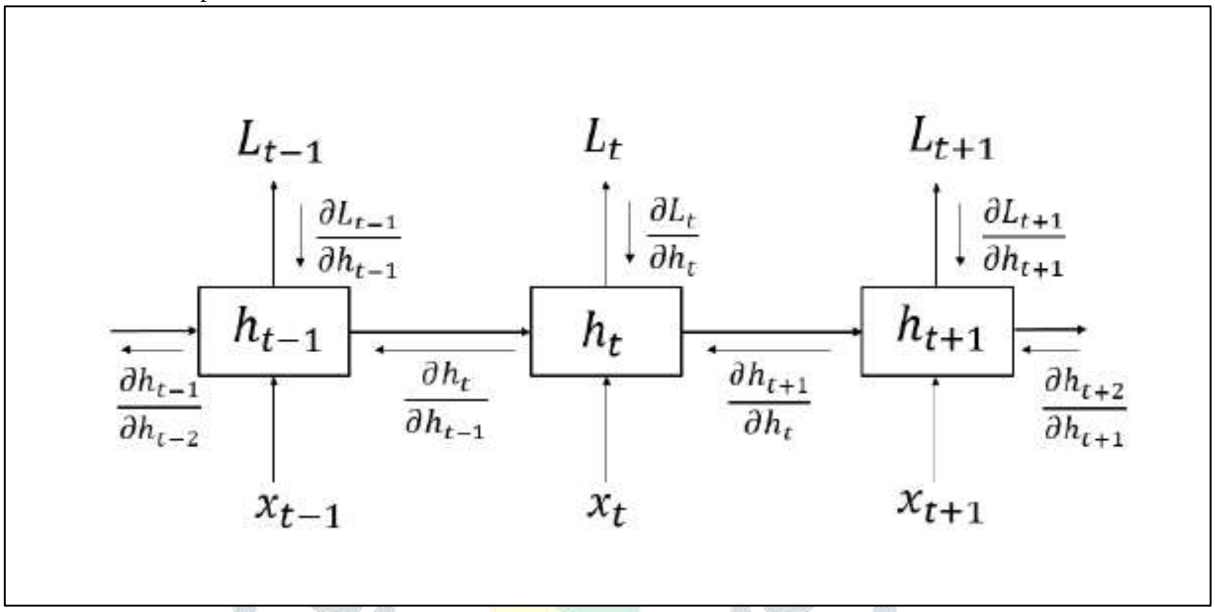

L_{t-1}
L_t
L_{t+1}
$\frac{\partial L_{t-1}}{\partial h_{t-1}}$
$\frac{\partial L_t}{\partial h_t}$
$\frac{\partial L_{t+1}}{\partial h_{t+1}}$
h_{t-1}
h_t
h_{t+1}
$\frac{\partial h_{t-1}}{\partial h_{t-2}}$
$\frac{\partial h_t}{\partial h_{t-1}}$
$\frac{\partial h_{t+1}}{\partial h_t}$
$\frac{\partial h_{t+2}}{\partial h_{t+1}}$
x_{t-1}
x_t
x_{t+1}

Figure 0.7 Illustration how gradients back propagate through time as well as layers

Consider

$$\begin{aligned} \partial_h^t &= \frac{\partial L}{\partial a_h^t} \\ &= \frac{\partial L}{\partial b_h^t}\frac{\partial b_h^t}{\partial a_h^t} \\ &= \frac{\partial L}{\partial b_h^t} f'(a_h^t) \end{aligned}$$

Output of the neuron at time t, b_h^t not only affects the output layer, but also the hidden layer at the next time step t + 1. Hence

$$\begin{aligned} \frac{\partial L}{\partial b_h^t} &= \sum_{k=1}^{K} \frac{\partial L}{\partial a_k^t}\frac{\partial a_k}{\partial b_h^t} + \sum_{h\prime=1}^{H} \frac{\partial L}{\partial b_h^{t+1}}\frac{\partial b_h^{t+1}}{\partial b_h^t} \\ &= \sum_{k=1}^{K} \partial_k^t\, w_{hk} + \sum_{h\prime=1}^{H} \frac{\partial L}{\partial b_h^{t+1}}\frac{\partial b_h^{t+1}}{\partial a_h^{t+1}}\frac{\partial a_h^{t+1}}{\partial b_h^t} \\ &= \sum_{k=1}^{K} \partial_k^t\, w_{hk} + \sum_{h\prime=1}^{H} \frac{\partial L}{\partial b_h^{t+1}} f'(a_h^{t+1}) w_{h'h} \\ &= \sum_{k=1}^{K} \partial_k^t\, w_{hk} + \sum_{h\prime=1}^{H} \partial_{h\prime}^{t+1} w_{h'h} \end{aligned}$$

By substituting the above equations

$$\partial_h^t = f'(a_h^t)\,(\sum_{k=1}^{K} \partial_k^t w_{hk} + \sum_{h\prime=1}^{H} \partial_{h\prime}^{t+1} w_{h'h})$$

In general, when we have more than one hidden layer, for the unit h in the l^{th}layer H_l. This can be thought of as applying the back propagation applied to a FFNN in which target values are specified for every layer not just the last. Since the network weights are reused at every time step, take the average or sum of weights to get the final derivatives.

$$\frac{\partial L}{\partial w_{ij}} = \sum_{i=1}^{T} \frac{\partial L}{\partial a_j^t} \frac{\partial a_j^t}{\partial w_{ij}} + \sum_{i=1}^{T} \partial_j^t \, b_j^t$$

The terms can be calculated by starting at t = T and by applying the above equation repeatedly decrementing t at each step. $\delta_j^{T+1} =$ 0, ∇j, since after time Tthere is no error.

Two Issues of Vanilla RNN: Training RNNs is difficult because of the problem known as 'Vanishing and exploding gradients' mentioned in Bengio et al. This problem makes it hard to learn and tune the parameters of the model, especially when using the gradient-based methods. Due to vanishing gradient problem the vanilla recurrent neural network is unable to capture the long term dependency.

Long Short-Term Memory: The LSTM recurrence is designed to address the limitations of the vanilla RNN. To capture long term dependency it uses a special kind of memory ct known as cell memory, which is controlled through specially designed structures call gates. These gates can be used optionally to pass the information through, which is made of sigmoid neural net layer and point wise multiplication operation. LSTM use three gates namely forget gate, input gate and output gate. So information can flow though cell memory ct unchanged and which makes LSTM network able to capture long-term-dependency at least for 1000 time steps as author claims.

Though LSTM solved vanishing gradient problem to some extent, but exploding gradient problem is still there and for that gradient clipping can be used.

ResNet Architecture

ResNet architecture [31] has very deep network and it's depth can go upto 152 layers. In this architecture, residual representation functions are used instead of directly learning input data. Before ResNet, AlexNet having 8 layer (5 convolutional, 3 connected), VGG having 19 layers and GoogleNet having 22 layers were representative of stacked up layers. In these networks, the vanishing gradient problem eats up the performance. ResNet introduces skip connection or shortcut connection to fit the input from the previous layer to the next layer without modifying the input. He, Kaiming & Zhang et al. [32] has described the connection of residual block, skip connection layers and deeper residual functions as follows.

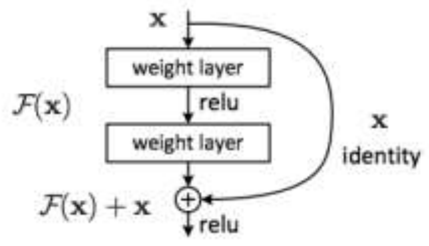

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 0.8 Skip Connection Residual Block

The skip connection identity mapping does not add any parameters and connects output from the previous layer to the layer ahead. In the figure above x is the input vector and F(x) is the activation function. There can be situations when x and F(x) will not have same dimension. A convolution operation shrinks spatial resolution of an image for example a 3x3 kernel on 32x32 image results in 30x30 image. A linear projection W is used to multiply with the identity mapping to increase the channels of shortcut to identify the residual. This allows for the input x and F(x) to be combined as input to the next layer.

$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}.$ A 34 layer VGG layer is compared in the below figure against skip connected layers in ResNet. The VGG-19 [2] (bottom) is a state-of-the-art approach in ILSVRC 2014. 34-layer middle network is considered as the deeper network of VGG-19, i.e. more convolution layers. 34-layer top residual network (ResNet) is the vanila one with addition of skip / shortcut connection. 3 types of skip / shortcut connections are there for ResNet. They appear when the input dimensions are smaller than the output dimensions. First, shortcut performs identity mapping, with extra zero padding for increasing dimensions. Thus no extra parameters are present. Second, the projection shortcut is used for increasing dimensions only, the other shortcuts are identity. Extra parameters are needed. Third, all shortcuts are projections. Extra parameters are more than that of second.

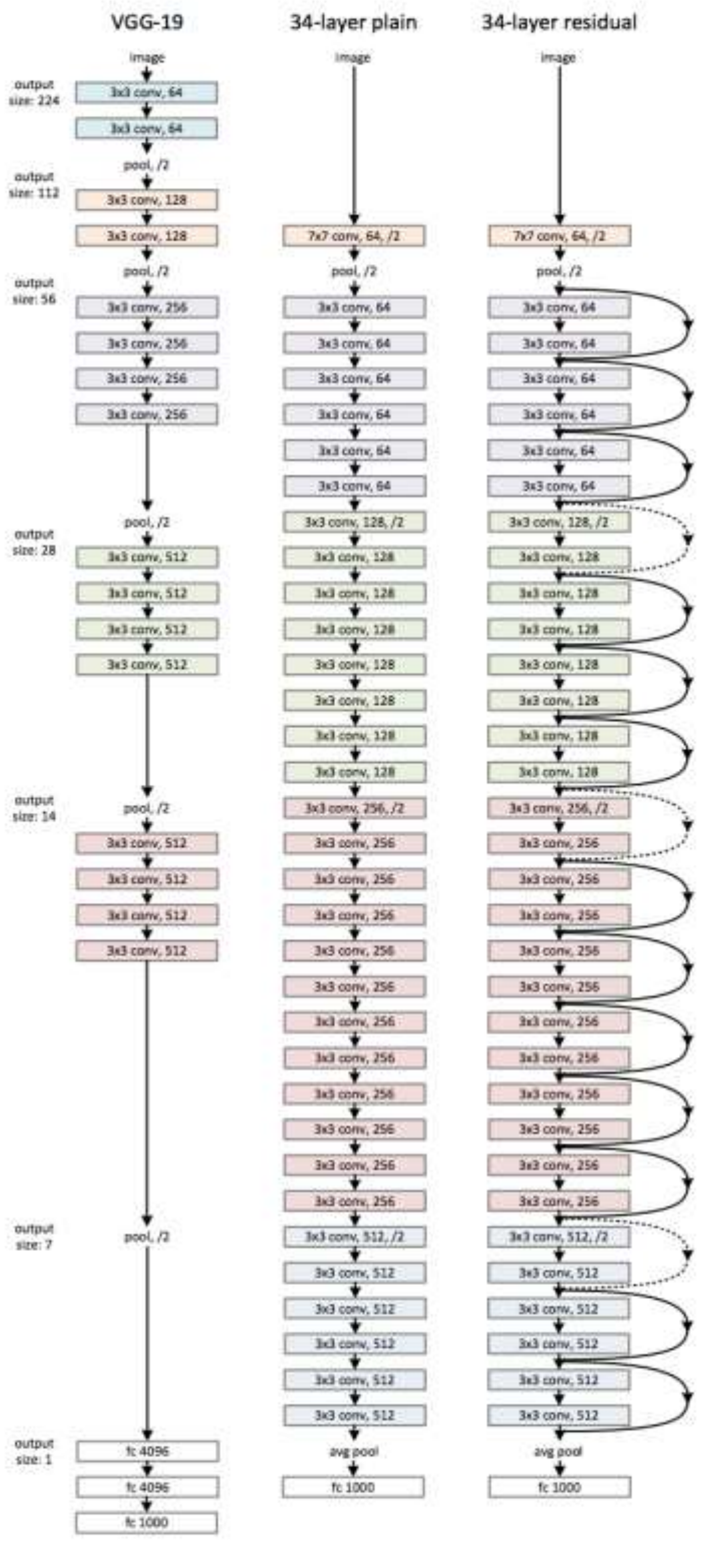


Figure 0.9 34 layer VGGNET verses Skip Connection Layers

Since the network has been developed very deep, the time complexity is high as well. A bottleneck design is used to reduce the complexity as follows.

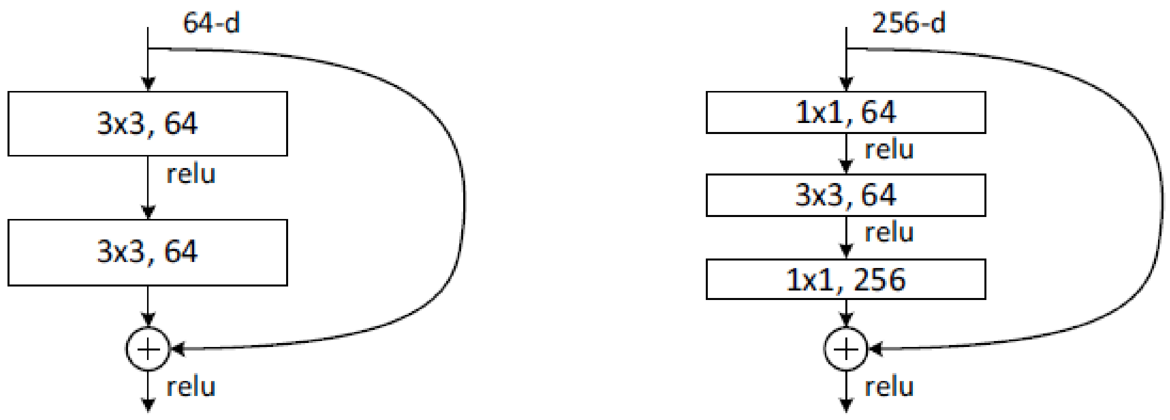
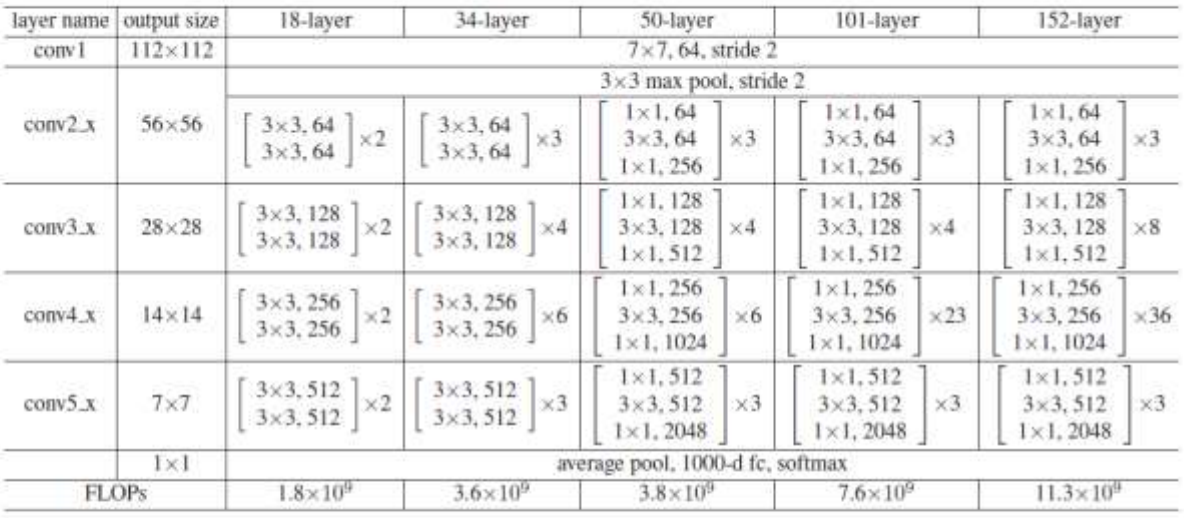


Figure 0.10 Deeper Residual Function

The 1×1 convolution layers are added to the start and end of network as in the figure (right). This is a suggested technique in Network In Network and GoogLeNet (Inception-v1). It turns out that 1×1 convolution can reduce the number of connections (parameters) while not degrading the performance of the network so much. 34-layer ResNet become 50-layer ResNet with the bottleneck design. This is ResNet50 pretrained model which is used in this work. Other than that, there are deeper network with the bottleneck design: ResNet-101 and ResNet-152. The complete architecture for all network is as below:



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		[3×3, 64 3×3, 64] ×2	[3×3, 64 3×3, 64] ×3	[1×1, 64 3×3, 64 1×1, 256] ×3	[1×1, 64 3×3, 64 1×1, 256] ×3	[1×1, 64 3×3, 64 1×1, 256] ×3
conv3_x	28×28	[3×3, 128 3×3, 128] ×2	[3×3, 128 3×3, 128] ×4	[1×1, 128 3×3, 128 1×1, 512] ×4	[1×1, 128 3×3, 128 1×1, 512] ×4	[1×1, 128 3×3, 128 1×1, 512] ×8
conv4_x	14×14	[3×3, 256 3×3, 256] ×2	[3×3, 256 3×3, 256] ×6	[1×1, 256 3×3, 256 1×1, 1024] ×6	[1×1, 256 3×3, 256 1×1, 1024] ×23	[1×1, 256 3×3, 256 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512 3×3, 512] ×2	[3×3, 512 3×3, 512] ×3	[1×1, 512 3×3, 512 1×1, 2048] ×3	[1×1, 512 3×3, 512 1×1, 2048] ×3	[1×1, 512 3×3, 512 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 0.11 ResNet Architecture

IV. Analysis

Dataset

To carry out the proposed method to determine petals number along with the flower classification the following dataset is used. Oxford-102 dataset contains 8,189 images consisting of 102 flower categories. The flowers are commonly occurring in the United Kingdom. Each category has between 40 and 258 images.

Data Understanding

A MATLAB file imagelabels.mat contains labels of the images. From this file the number of images for each label have been found out. The Label list comprises label id and number of images for that label.
oxford102_flower_dataset_labels.txt contains flower names against the label ids. A dataset has been generated from this file for extracting flower names or its label.
The following functions have been created to understand the relationship between labels, label name, image name and their index so that images can be fetched easily.

Function Name	Description
find_flower_label_from_flower_image_name	To find the flower image name from the label id. Also a generic method has been created to find index from the flower label.
find_unique_label	Find the index of the flower position from the label.
find_flower_category_name	Find flower category name by passing file name in the argument.
get_flower_id	Get flower id from the flower file path.
get_image_dictionary	Create an image dictionary of the flowers by passing flower images path.
get_train_test_catalog	Create train test catalog dictionary which contains train and test label dataset. Each flower label vs number of flowers to be trained or tested using 80 : 20 rule

generate_train_test_split	Generate training and testing image ids randomly during training and testing purpose.
train_test_img_nums	Extract Image ids for training and testing
load_training_image_data	converting the image_database into train and test datasets using 80:20 principles for each category the images will be separated in that r atio for traning and test purposes.

Table 0.1 Descriptions of the functions created for loading the flower images

Dataset usage

There are two ways oxford 102 flower dataset is used. In the beginning, all the flower images have been kept under a single folder named "jpg". Each flower image has a unique name. This dataset is used to perform petals calculation for the methods in the section 3.1.2 and 3.1.3.

Train, Test Images

In the second way, the oxford dataset is divided into three parts, train, test and valid. In the train folder total 6552 images are kept according to different classes as subfolder names. In the test folder total 818 images are kept for validation purpose.

Flowers for Petals calculation

There is one more folder created to keep distinguished images specifically to be used for petals count as described in section 3.1.4.

Data Cleaning

The description for each image from the image id associated to the physical image has been extracted. These image ids used to find the label of that image. Training and testing image dictionary have been created where images are loaded by their ids. There is no specific cleaning has not been performed. But for the calculation of petals count in the hue based image segmentation method, flower images have been chosen carefully. From each of the categories flower images which contain one single significantly visible flower has been considered for the petals count calculation.

Petal count calculation using K-Means, thresholding and by edge detection

All 8189 images were segmented into two clusters by using K-Means algorithm. In this process the images are filtered with pixel values between 0 and 1. Each image is reshaped in an array. These clustered images are saved as the masked image dataset dictionary. These images are further used to find the petal count by the methods described in section 3.1.2 and 3.1.3.

Petals count by using Otsu's algorithm

Each of these masked images is reshaped to 256x256. Otsu algorithm is applied on the image to get thresholded or binary image. The nearest holes of the image that is assumed to be the filled with the same coloured pixel (either 0 or 1) have been filled in the binary image. The final image after filling the holes in the image can be seen as follows. The final petals count is considered by the area of the drops region greater than 40. This value has been considered by trial and error process.

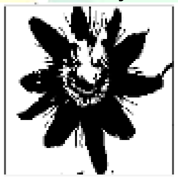


Figure 0.1 Image after applying Otsu algorithm

The connected components of this binary image are labelled. Out of these labelled image regions, the region which has area greater than 40 has been considered as a petal in the image. This limiting number has been adopted by trial and error methods.

Petals count by using Sobel's algorithm Similarly on the masked images Sobel algorithm has been applied. And thereafter the holes formed during the edge detection are filled by using morphology technique. These images are marked between 90 and 140. These values are taken by considering trial and error process. The final petals count is considered again by taking the drops regions greater than 40 as described above.

Petals count by hue based image segmentation Petals count has been also performed using another method in section 3.1.4. In this method specific flowers have been chosen from all the categories. In this case the contour has been calculated from the threshold image produced from hue based image segmentation. Using this contour, center of the contour as well as four extreme points has been calculated. A triangle has been formed from the extreme left, top and center points. The petals count has been calculated by dividing contour area by the area of the triangle. It has been observed that extreme points are the points of the flower petals extreme points. Considering this fact the triangle has been formed by two petal's extreme point and the center. This triangle contains the area of a single petal. Thus whole contour area has been divided by this triangular area to find the petals.

ResNet50Model Architecture

Resnet50 model has been chosen to classify the flowers from the training image database. The size of the input image is chosen as 224x224. The final model of ResNet50 contains the initial layers as follows.

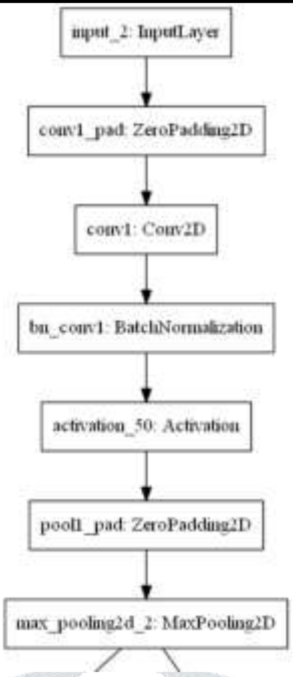

input_2: InputLayer
conv1_pad: ZeroPadding2D
conv1: Conv2D
bn_conv1: BatchNormalization
activation_50: Activation
pool1_pad: ZeroPadding2D
max_pooling2d_2: MaxPooling2D

Figure 0.2 Reset50 with 90 percent accuracy model architecture I

The input layer has gone through ZeroPadding2D, Conv2D, BatchNormalization and Activation. Thereafter it has gone through one ZeroPadding2D and a MaxPooling2D layer.

Layer(type)	Output Shape	Param #	Connected to
input_2 (InputLayer)	(None, None, None, 3	0	
conv1_pad (ZeroPadding2D)	(None, None, None, 3	0	input_2[0][0]
conv1 (Conv2D)	(None, None, None, 6	9472	conv1_pad[0][0]
bn_conv1 (BatchNormalization)	(None, None, None, 6	256	conv1[0][0]
activation_50 (Activation)	(None, None, None, 6	0	bn_conv1[0][0]
pool1_pad (ZeroPadding2D)	(None, None, None, 6	0	activation_50[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, None, None, 6	0	pool1_pad[0][0]

Table 0.2 ResNet50 Initial layers

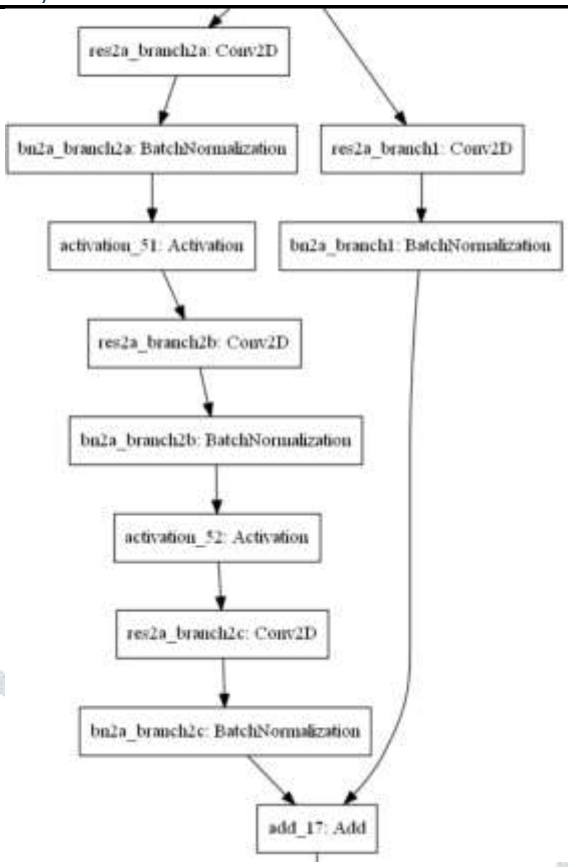

res2a_branch2a: Conv2D
bn2a_branch2a: BatchNormalization
res2a_branch1: Conv2D
activation_51: Activation
bn2a_branch1: BatchNormalization
res2a_branch2b: Conv2D
bn2a_branch2b: BatchNormalization
activation_52: Activation
res2a_branch2c: Conv2D
bn2a_branch2c: BatchNormalization
add_17: Add

Figure 0.3 ResNet 50 with 90 percent accuracy model architecture II

This maxPooling2D layer is input to the 8 layers of skip connection. Hence Conv2D, BatchNormalization, has been applied in the direct layer thrice and Activation has been applied for the first 2 layers of Conv2D and BatchNormalization. The Activation has been done after adding the skipped layers with the BatchNormalized layer.

Layer(type)	Output Shape	Param #	Connected to
res2a_branch2a (Conv2D)	(None, None, None, 6	4160	max_pooling2d_2[0][0]
bn2a_branch2a (BatchNormalization)	(None, None, None, 6	256	res2a_branch2a[0][0]
activation_51 (Activation)	(None, None, None, 6	0	bn2a_branch2a[0][0]
res2a_branch2b (Conv2D)	(None, None, None, 6	36928	activation_51[0][0]
bn2a_branch2b (BatchNormalization)	(None, None, None, 6	256	res2a_branch2b[0][0]
activation_52 (Activation)	(None, None, None, 6	0	bn2a_branch2b[0][0]
res2a_branch2c (Conv2D)	(None, None, None, 2	16640	activation_52[0][0]
res2a_branch1 (Conv2D)	(None, None, None, 2	16640	max_pooling2d_2[0][0]
bn2a_branch2c (BatchNormalization)	(None, None, None, 2	1024	res2a_branch2c[0][0]
bn2a_branch1 (BatchNormalization)	(None, None, None, 2	1024	res2a_branch1[0][0]

Table 0.3 ResNet50 Layers before skip connection layers

There are 15 layers of skip connection has been added one after another after this architecture as shown below.

Layer(type)	Output Shape	Param #	Connected to
activation_53 (Activation)	(None, None, None, 2	0	add_17[0][0]
res2b_branch2a (Conv2D)	(None, None, None, 6	16448	activation_53[0][0]
bn2b_branch2a (BatchNormalization)	(None, None, None, 6	256	res2b_branch2a[0][0]

activation_54 (Activation)	(None, None, None, 6	0	bn2b_branch2a[0][0]
res2b_branch2b (Conv2D)	(None, None, None, 6	36928	activation_54[0][0]
bn2b_branch2b (BatchNormalization)	(None, None, None, 6	256	res2b_branch2b[0][0]
activation_55 (Activation)	(None, None, None, 6	0	bn2b_branch2b[0][0]
res2b_branch2c (Conv2D)	(None, None, None, 2	16640	activation_55[0][0]
bn2b_branch2c (BatchNormalization)	(None, None, None, 2	1024	res2b_branch2c[0][0]

Table 0.4 Repeated Skip Connection Layer for 15 Layers in ResNet50

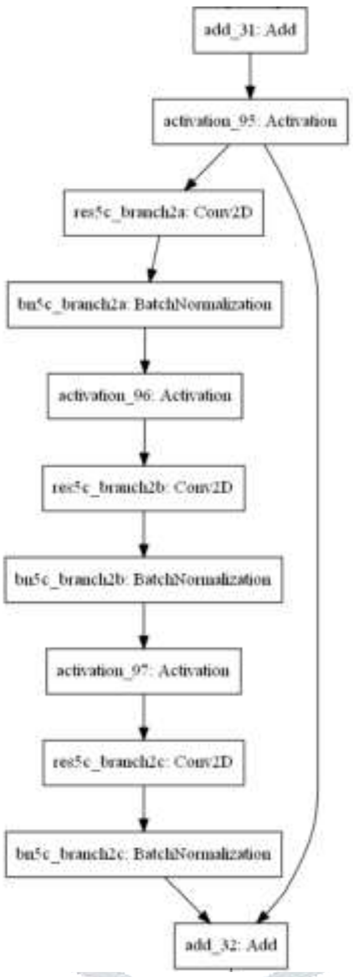

add_31: Add
activation_95: Activation
res5c_branch2a: Conv2D
bn5c_branch2a: BatchNormalization
activation_96: Activation
res5c_branch2b: Conv2D
bn5c_branch2b: BatchNormalization
activation_97: Activation
res5c_branch2c: Conv2D
bn5c_branch2c: BatchNormalization
add_32: Add

Figure 0.4 ResNet50 with 90 percent accuracy model architecture III

Each of these skip connections has 3 layers of Conv2D and BatchNormalization and Activation thereafter as explained above. Output of the last skip connection has gone through a GlobalMaxPooling2D and two Dense layers as shown below.

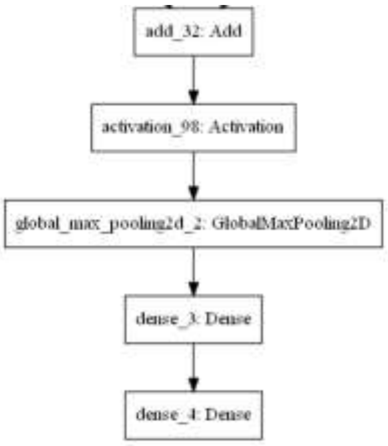

add_32: Add
activation_98: Activation
global_max_pooling2d_2: GlobalMaxPooling2D
dense_3: Dense
dense_4: Dense

Figure 0.5 ResNet50 with 90 percent accuracy model architecture IV

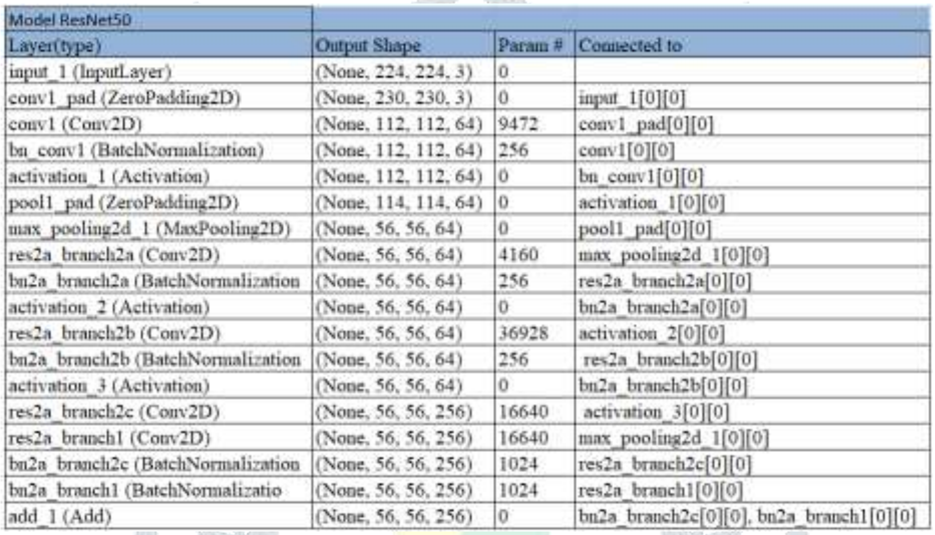
Layer(type)	Output Shape	Param #	Connected to
add_32 (Add)	(None, None, None, 2	0	bn5c_branch2c[0][0], activation_95[0][0]
activation_98 (Activation)	(None, None, None, 2	0	add_32[0][0]
global_max_pooling2d_2	(GlobalM (None, 2048)	0	activation_98[0][0]
dense_3 (Dense)	(None, 1024)	2098176	global_max_pooling2d_2[0][0]
dense_4 (Dense)	(None, 102)	104550	dense_3[0][0]

Table 0.5 ResNet50 Dense Layers at the end

This model is used on the training as well as feature extraction during flower image classification.

Feature Extraction from ResNet50 and trained by LSTM

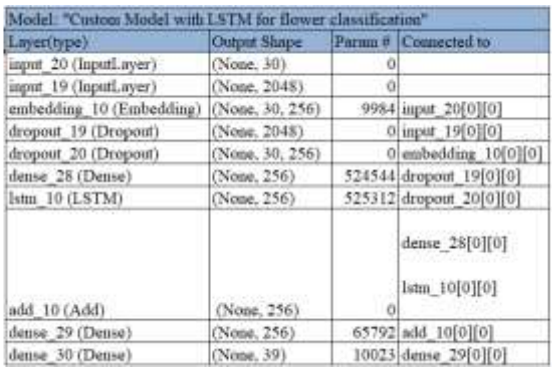
Resnet50 model has been chosen to extract features from the training image database. The size of the input image is chosen as 224x224. Instead of taking the output as is, the output of the model has been modified till avg pooling in order to get the flattened image size as 2048 as output of the model. This model is used for the extraction of flower image features.



Model ResNet50			
Layer(type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 224, 224, 3)	0	
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	0	input_1[0][0]
conv1 (Conv2D)	(None, 112, 112, 64)	9472	conv1_pad[0][0]
bn_conv1 (BatchNormalization)	(None, 112, 112, 64)	256	conv1[0][0]
activation_1 (Activation)	(None, 112, 112, 64)	0	bn_conv1[0][0]
pool1_pad (ZeroPadding2D)	(None, 114, 114, 64)	0	activation_1[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 64)	0	pool1_pad[0][0]
res2a_branch2a (Conv2D)	(None, 56, 56, 64)	4160	max_pooling2d_1[0][0]
bn2a_branch2a (BatchNormalization	(None, 56, 56, 64)	256	res2a_branch2a[0][0]
activation_2 (Activation)	(None, 56, 56, 64)	0	bn2a_branch2a[0][0]
res2a_branch2b (Conv2D)	(None, 56, 56, 64)	36928	activation_2[0][0]
bn2a_branch2b (BatchNormalization	(None, 56, 56, 64)	256	res2a_branch2b[0][0]
activation_3 (Activation)	(None, 56, 56, 64)	0	bn2a_branch2b[0][0]
res2a_branch2c (Conv2D)	(None, 56, 56, 256)	16640	activation_3[0][0]
res2a_branch1 (Conv2D)	(None, 56, 56, 256)	16640	max_pooling2d_1[0][0]
bn2a_branch2c (BatchNormalization	(None, 56, 56, 256)	1024	res2a_branch2c[0][0]
bn2a_branch1 (BatchNormalizatio	(None, 56, 56, 256)	1024	res2a_branch1[0][0]
add_1 (Add)	(None, 56, 56, 256)	0	bn2a_branch2c[0][0], bn2a_branch1[0][0]

Table 0.6 Customized ResNet50 Model

This model is used on the training image dataset and the extracted features were added to a dictionary for reuse during flower image classification. For flower image classification a LSTM model has been defined. This model will be used to train on the extracted features earlier. The input shape of this model is 2048 which is the flattened image size. There are 50 % dropout layers applied thrice to this model to prevent Overfitting during training time and to achieve 256 input data points. After each drop out layer there is a dense layer with "relu" activation applied. These layers act like a feature extractor layer from the extracted features from previous input. A sequence layer having input shape of the maximum length of the word embeddings have been created. This layer is embedded with the vocabulary created earlier. Vocabulary creation will be explained later in this chapter. Then 50% drop out has been applied to this layer. Thereafter LSTM model is applied to this sequence layer. A decoder layer is applied to add these two layers i.e. the extractor layer and the sequence layer. Then a dense layer with "relu" activation is applied, thereafter another dense layer with "softmax" activation layer has been applied. Below is the summary of the classification model.



Model: "Custom Model with LSTM for flower classification"			
Layer(type)	Output Shape	Param #	Connected to
input_20 (InputLayer)	(None, 30)	0	
input_19 (InputLayer)	(None, 2048)	0	
embedding_10 (Embedding)	(None, 30, 256)	9984	input_20[0][0]
dropout_19 (Dropout)	(None, 2048)	0	input_19[0][0]
dropout_20 (Dropout)	(None, 30, 256)	0	embedding_10[0][0]
dense_28 (Dense)	(None, 256)	524544	dropout_19[0][0]
lstm_10 (LSTM)	(None, 256)	525312	dropout_20[0][0]
add_10 (Add)	(None, 256)	0	dense_28[0][0] lstm_10[0][0]
dense_29 (Dense)	(None, 256)	65792	add_10[0][0]
dense_30 (Dense)	(None, 39)	10023	dense_29[0][0]

Table 0.7 Deep Learning Classification Model

From the image database and training image numbers a word vocabulary dictionary has been created. This dictionary will be presented to the flower classification model during training of the flower images. A tokenizer has been created in this context which converts the texts to binary objects. A data generator is created which is used to train the model batch by batch by an instance of sequence. The sequence is created by word vocabulary, flower image feature dictionary with the help of tokenizer and the maximum length of all image's description. For each description from the image description created from the vocabulary, the sequence is extracted from the tokenizer. Input arrays formed by the flattened image matrix, generated input image description and expected output description of the image through this function are passed to the data generator. The data generator tries to fetch each input sequence through this function from the training set during training of the model for flower classification.

Training with solo ResNet50 model for flower image classification

A total 6552 images are trained. The images are placed in the train folder. Each subfolder name represents category id of individual flower. Each subfolder contains flower images of respective type. An instance of ImageDataGenerator has been created. The training images are made ready to be augmented via a number of random transformations so that the model will not see twice the exact same picture. This helps in preventing overfitting and makes the model more generalized. The following transformations are used in this process. rotation_range is 30 degrees. In this range the training images are rotated randomly.
width_shift_range and height_shift_range as 0.2. The pictures are randomly translated vertically or horizontally within this range.
horizontal_flip is used for flipping the images randomly.
Below are the sample images of a category 1 or "pink primrose" flowers after applying augmentation transformation.

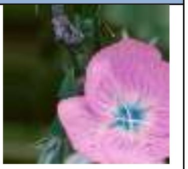
Transformed Training images for pink primrose						
						

Table 0.8 Augmented Flowers during training

A train generator has been created by the above process which contains batches of augmented images. The input shape of the images is 224x224 and as these are colored images, each image has 3 channels. The hyperparameters tuning has been added as follows. This model is compiled with the "adam" optimizer. Using this optimizer computes and improves adaptive learning rate for each parameter. The loss function categorical_crossentropy is used in order to classify the flower images in 102 categories. Accuracy has been chosen as metrics to understand how model is performing during training and validation. Totally 256 epochs have been executed for training the images. And the number of batches per epochs is 32.
Fine tuned Model This model is fine tuned thereafter as validation accuracy was below 60 percent. The initial 45 layers have been freezed to improve the model performance. The decision of freezing this specific number of layers is made by using trial and error method. Later the model was compiled with SGD optimizer where initial learning rate is 0.00001 and momentum is 0.9 and other hyperparameters are kept as of earlier. This tuned model is compiled and trained with one epoch as it received 90 percent validation accuracy and due to limitation of the system capacity.

Callbacks for the solo ResNet50 Model

There are two callbacks used in this model in order to save the best model after a certain number of epochs and when the model stops improving. In this case a ModelCheckPoint has been defined to save the best models after every 100 epochs when validation loss is growing after reaching a lower number. An EarlyStopping has been defined to stop the training when validation accuracy is not improving after going through 100 epochs.

Summary of training of deep learning classification model

Below is the summary of the Reset50 classification model.

Total Parameters	Trainable Params	Non-trainable params
2,57,90,438	2,53,29,510	4,60,928

Table 0.9 Summary of Deep Learning Classification Model

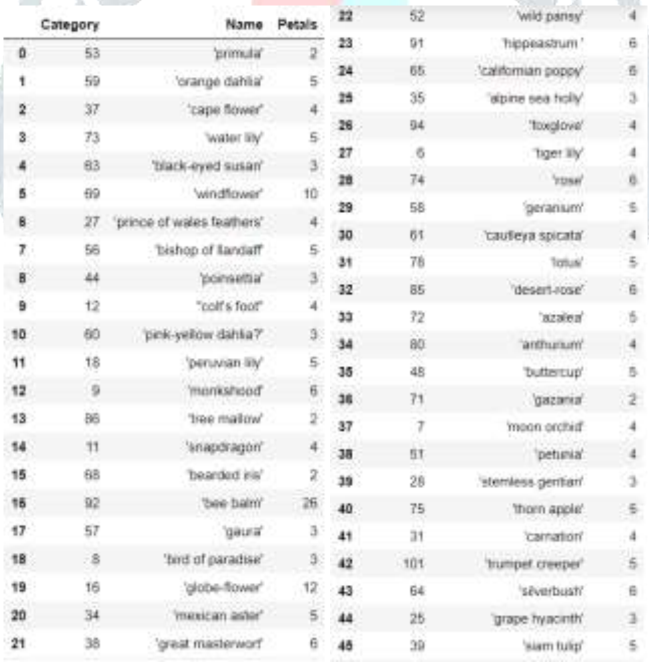
The trainable parameters from the original ResNet50 model which is inherited from "imagenet" have been discarded. The model is compiled with the optimizers, loss function and metrics information as stated above. The below information excerpts description of the training results for last few epochs of first 64 epochs.

Epochs	loss	accuracy	validation loss	validation accuracy
60/64	0.0677	0.9787	3.5348	0.6043
61/64	0.066	0.9779	2.6251	0.6043
62/64	0.0869	0.9756	4.0541	0.5941
63/64	0.0791	0.979	2.7289	0.5115
64/64	0.0651	0.981	1.7341	0.5827

Table 0.10 Excerpts details from training of Reset50 Model

Petals count for the flower images

In order to finally add the number of petals count in the outcome of the flower classification, there is a petals count database created. A separate flowers_for_petals image database has been created to count petals count for each category of the flower images. Using this image database hue based image segmentation has been followed as per the paper [34]. Each of these images is converted to the threshold image based on the hue data. Later as mentioned by the section 3.1.4 petal count for each of these categories has been calculated. The petals are calculated by assuming the contour circle area as the bonding box of the actual flower. The extreme points of the two opposite petals have been joined to create a triangle where the other point is the center of the circle of the contour selected. Considering the area of the triangle is the area between the two petals and dividing the contour area by this triangular area, number of petals has been found out. This information has been saved in a csv file named flowers_with_petals_info.csv. This is the petals count database which has been used to create predicted flower description later.

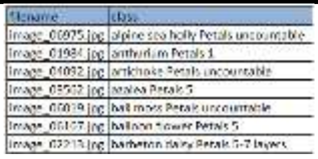


	Category	Name	Petals
0	53	'primula'	2
1	59	'orange dahlia'	5
2	37	'cape flower'	4
3	73	'water lily'	5
4	63	'black-eyed susan'	3
5	69	'windflower'	10
6	27	'prince of wales feathers'	4
7	56	'bishop of llandaff'	5
8	44	'poinsettia'	3
9	12	"colt's foot"	4
10	60	'pink-yellow dahlia?'	3
11	18	'peruvian lily'	5
12	9	'monkshood'	6
13	86	'tree mallow'	2
14	11	'snapdragon'	4
15	68	'bearded iris'	2
16	92	'bee balm'	26
17	57	'gaura'	3
18	8	'bird of paradise'	3
19	16	'globe-flower'	12
20	34	'mexican aster'	5
21	38	'great masterwort'	6
22	52	'wild pansy'	4
23	91	'hippeastrum '	6
24	65	'californian poppy'	6
25	35	'alpine sea holly'	3
26	94	'foxglove'	4
27	6	'tiger lily'	4
28	74	'rose'	6
29	58	'geranium'	5
30	61	'cautleya spicata'	4
31	78	'lotus'	5
32	85	'desert-rose'	6
33	72	'azalea'	5
34	80	'anthurium'	4
35	48	'buttercup'	5
36	71	'gazania'	2
37	7	'moon orchid'	4
38	51	'petunia'	4
39	28	'stemless gentian'	3
40	75	'thorn apple'	5
41	31	'carnation'	4
42	101	'trumpet creeper'	5
43	64	'silverbush'	6
44	25	'grape hyacinth'	3
45	39	'siam tulip'	5

Table 0.11 Computed Petals count Database

Testing the flower classification model

A test image data generator is created to validate 873 test images. The keras API predict_generator is used to classify test images. The number of batch size is same as of train image generator. Number of steps calculated is the number of test images divide by the batch size. To calculate the model accuracy, true labels of the test images have been found out. The oxford's 102 image database has imagelabels.mat file. This is a MATLAB file. All the labels are found out from that file. A function has been made to find the actual flower category of the test images from that label information. A true label information has been generated manually for all the test images to help in the evaluation process later part of which can be seen as follows.



filename	class
image_00975.jpg	alpine sea holly Petals uncountable
image_01984.jpg	anthurium Petals 1
image_04092.jpg	artichoke Petals uncountable
image_03502.jpg	azalea Petals 5
image_06019.jpg	ball moss Petals uncountable
image_06167.jpg	balloon flower Petals 5
image_02213.jpg	barbeton daisy Petals 5-7 layers

Table 0.12 Flower Category Truth Table for Test images

IV. DEVELOPMENT ENVIRONMENT

In this section explanation is provided about the development environment used for executing flower classification methods. Python open source libraries have been extensively used to calculate, measure and evaluate the metrics for this research work.

System Information

This research work has been conducted on the system as per the below configuration.

Item	Configuration
OS	Windows 10
System Type	x64
Processor	Intel® Core™ i5-8300 Hz CPU @ 2.3 GHz, 4 Cores
Physical Memory	8 GB
GPU	NVIDIA GeForce GTX 1050 Ti, 4 GB Graphics memory
Development IDE	Jupyter Notebook

Table 0.13 System configuration

Calculate the Petal count

The following python libraries have been used for the petals count calculations.

Sr. No	Libraries
1	Numpy
2	Pandas
3	cv2
4	Matplotlib
5	Csv
6	Math
7	Os

Table 0.14 Libraries for calculating petals count

Detection of the flower images

The following libraries have been used for flower image detection.

Sr. No	Libraries
1	Numpy
2	Pandas
3	keras.applications.resnet50
4	keras.models
5	keras.layers
6	keras.preprocessing.image
7	keras.optimizers
8	keras.callbacks.callbacks
9	matplotlib.pyplot

10	cv2
11	Scipy
12	scipy.io
13	Sklearn

Table 0.1 Python Libraries used for flower image detection

V. RESULTS AND DISCUSSIONS

This section covers the evaluation metrics and the outcome of the effectiveness of the thesis work that has been done on the flower image classification model.

Evaluation Metrics

In this section different evaluation metrics are presented which are generated during applying petals count methodologies and the flower image classification techniques. BELU is used as metric for capturing the score when LSTM based model is used. Experiment with ResNet50 and LSTM Model. In this case feature extraction is done by using pre-defined ResNet50 from keras application library and customized LSTM model. The result is as follows after manually running 3 epochs.

Actual	Predicted
startseq 0 4 1 2 5 a r t i c h o k e endseq	startseq 0 3 0 3 1 b e e b a l m endseq
startseq 0 1 4 5 0 p e t u n i a endseq	startseq 0 5 4 6 2 a z a l e a endseq

Table 0.1 Actual verses Predicted in Transfer Learning with ResNet50 and LSTM

Train Score	Test Score
74.85%	74.9%

Table 0.2 Train verses Test Scores in Transfer Learning with ResNet50 and LSTM

Experiment with only ResNet50 Model and fine tuning

While training with ResNet50 model using transfer learning after 140 epochs the EarlyStopping is invoked as the model was stopped learning any more. Below are the model accuracy and model loss information captured. It is clear from the below plots that there is very high training accuracy as compared to validation accuracy. On the other hand the training loss was low and almost flat where as the validation loss is randomly increasing and decreasing and does not going down. From this information, it can be concluded that the model is overfitting and needs fine tuning.

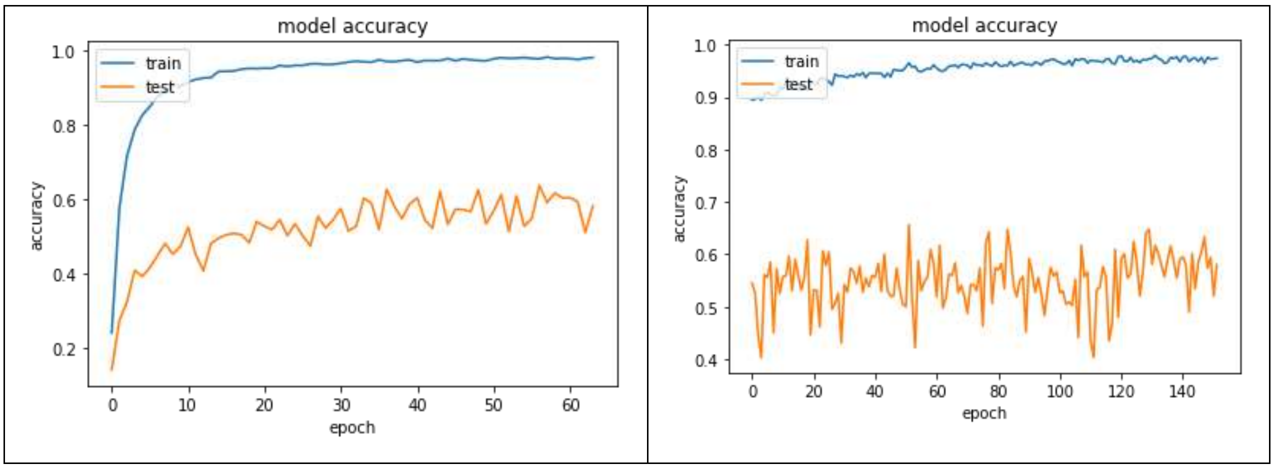


Table 0.3 Resnet50 Model Accuracy

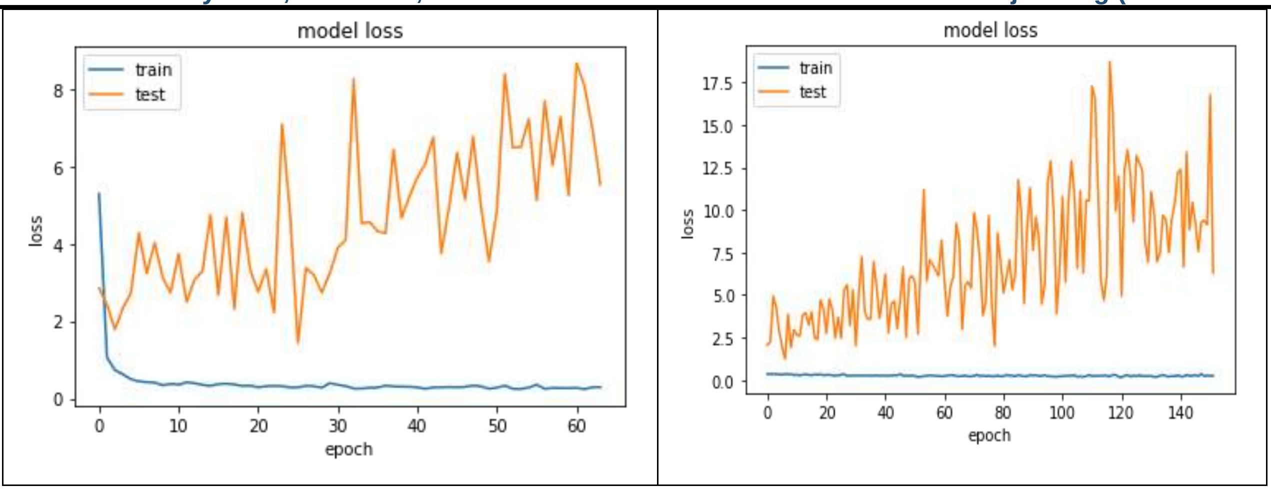


Table 0.4 ResNet50 Model loss

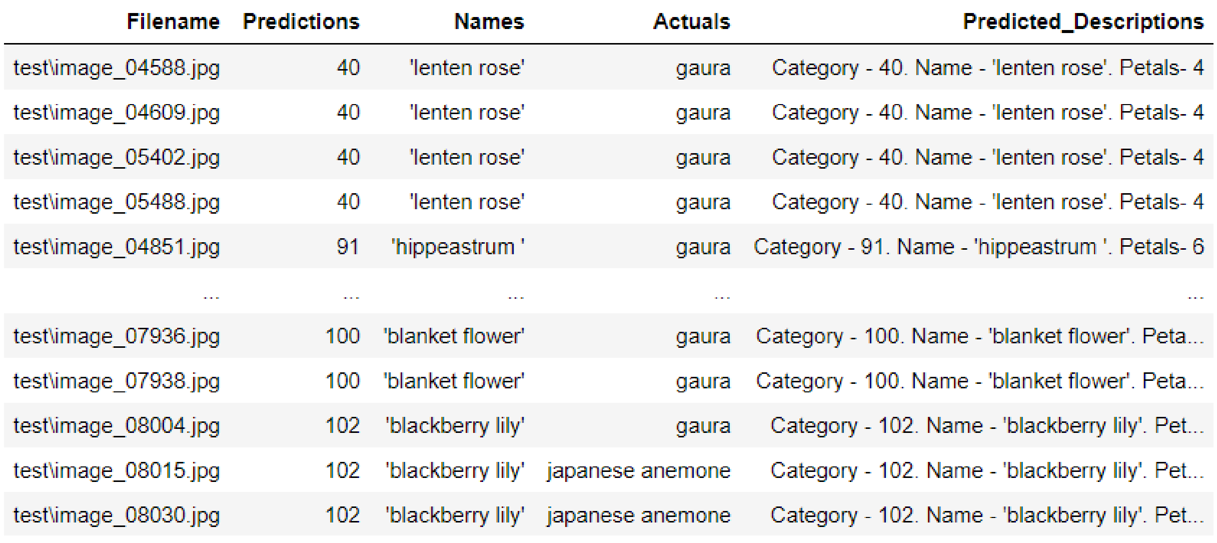
After using fine tuning the ResNet50 model the validation accuracy increased to 90 percent where as training accuracy reached to the level of 99 percent. Therefore overfitting has been taken care by fine tuning the model. The training details can be found from the below table.

Parameter	Metrics
Training Loss	0.1241
Training Accuracy	0.99
Validation Loss	0.2345
Validation Accuracy	0.9

Table 0.5 Training details of Fine tuned ResNet50 Model

This model further has been considered for prediction of the flower images.

The final result of the petals count along with the flower category detection has been captured below. In the below table it can be observed that the last column "Predicted_Decription" has been provided in order to visually identify the correctness of the model's performance. This column has been formatted with the text description as "Category <Name of the flower category predicted>. Name <Name of the flower predicted by the flower category>. Petals- <This petals information has been calculated by the hue based image segmentation methods>"



Filename	Predictions	Names	Actuals	Predicted_Descriptions
test\image_04588.jpg	40	'lenten rose'	gaura	Category - 40. Name - 'lenten rose'. Petals- 4
test\image_04609.jpg	40	'lenten rose'	gaura	Category - 40. Name - 'lenten rose'. Petals- 4
test\image_05402.jpg	40	'lenten rose'	gaura	Category - 40. Name - 'lenten rose'. Petals- 4
test\image_05488.jpg	40	'lenten rose'	gaura	Category - 40. Name - 'lenten rose'. Petals- 4
test\image_04851.jpg	91	'hippeastrum '	gaura	Category - 91. Name - 'hippeastrum '. Petals- 6
...	...	...	...	...
test\image_07936.jpg	100	'blanket flower'	gaura	Category - 100. Name - 'blanket flower'. Peta...
test\image_07938.jpg	100	'blanket flower'	gaura	Category - 100. Name - 'blanket flower'. Peta...
test\image_08004.jpg	102	'blackberry lily'	gaura	Category - 102. Name - 'blackberry lily'. Pet...
test\image_08015.jpg	102	'blackberry lily'	japanese anemone	Category - 102. Name - 'blackberry lily'. Pet...
test\image_08030.jpg	102	'blackberry lily'	japanese anemone	Category - 102. Name - 'blackberry lily'. Pet...

Figure 0.1 Final predicted verses actual using ResNet50 model

Experiment with Petals counts algorithms

There are three approaches attempted to find number of petals in this research work.

Using Hue based Image segmentation

The process of this algorithm has been explained in the section 3.1.4. Below is the metrics of flower petals calculation.

Parameter	Metrics
Total Matched	16
Total Unmatched	60
Total Unmatched Uncountable	24

Table 0.6 Metrics of categories of flowers matched through algorithm for petals

It can be seen that the petals calculation algorithm has been successfully evaluated for only 16 categories. The details of the specific flower categories have been listed below. In this regard there is an item called unmatched uncountable. This is referring 24 categories of flowers whose petals could not be able to determine due to its very close appearances in very small area within the flower.

Details of Matched Flower categories	
Category	Petals
mexican petunia	5
Azalea	5
black-eyed susan	3
Geranium	5
Buttercup	5
sweet William	5
canna lily	5
trumpet creeper	5
Hippeastrum	6
thorn apple	5
canterbury bells	6
barbeton daisy	5
Pelargonium	5
Bougainvillea	5
english marigold	5
Hibiscus	5

Table 0.7 Categories of correctly matched flower petals

Using K-Means and Image thresholding

After applying K-Means on the training images the masked images are saved in a pickle file for further processing. Following images are sample of these transformed images.

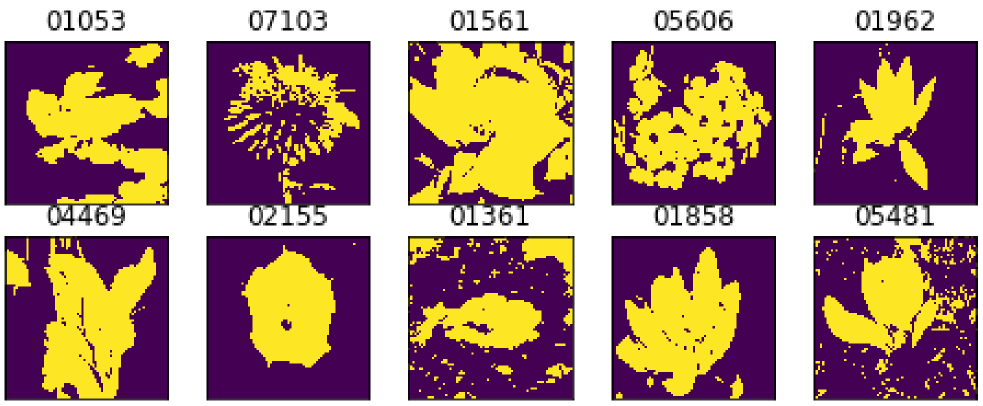


Figure 0.2 Transformed images after K-Means

These filtered images are not noise free. So, image thresholding is applied on them and holes in the respective regions are filled on the images. The following images are sample of the otsu's thresholded image with petals count information after hollow regions are filled by pixels.

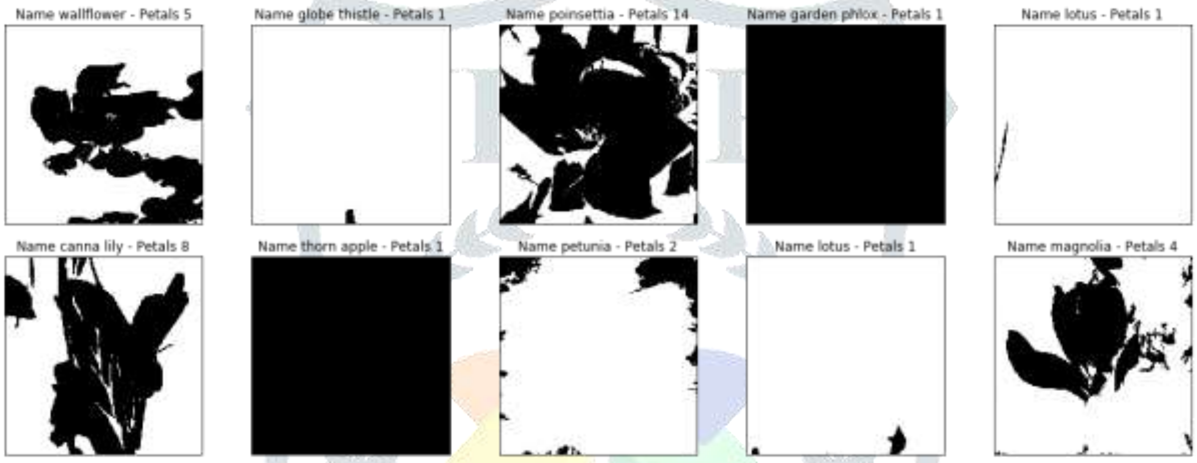

Name wallflower - Petals 5
Name globe thistle - Petals 1
Name poinsettia - Petals 14
Name garden phlox - Petals 1
Name lotus - Petals 1
Name canna lily - Petals 8
Name thorn apple - Petals 1
Name petunia - Petals 2
Name lotus - Petals 1
Name magnolia - Petals 4

Figure 0.3 Transformed images with petals information after thresholding

Using K-Means and Edge detection on the Images

One more attempt has been made to calculate petals after applying K-Means. Instead of using image thresholding technique, Sobel's edge detection algorithm is applied using open cv2 on the filtered images and then watershed morphology is used to fill the regions of the elevation map starting from the markers determined earlier. These background markers are based on the extreme parts of the histograms of gray values. Below are the sample images for elevation map, markers and the resulted flower with petals counted.

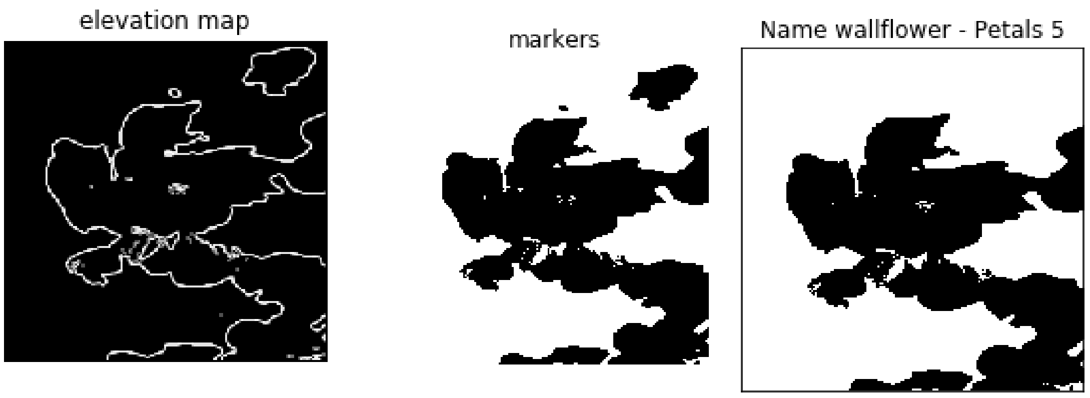


Figure 0.4 Petals count using Sobel with Watershed Transformation

The following flower images are displayed after applying Sobel's edge detection algorithm and the petals information.

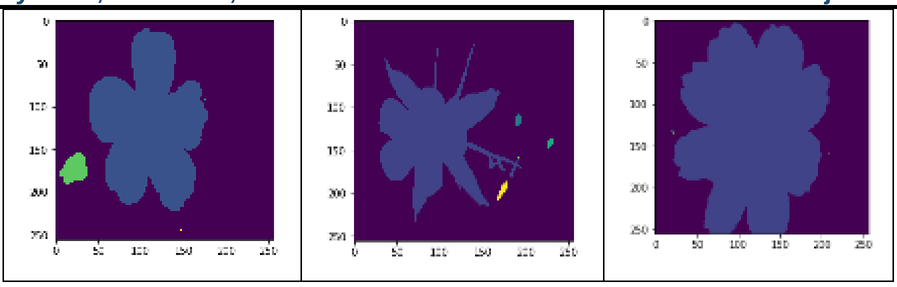


Table 0.8 Flower images after applying Sobel's Edge detection

File Name	Flower Category	Petals
image_06863.jpg	prince of wales feathers	10
image_01333.jpg	Petunia	0
image_01530.jpg	Poinsettia	0
image_04931.jpg	giant white arum lily	1
image_05868.jpg	black-eyed susan	0
image_02032.jpg	Anthurium	0
image_02490.jpg	morning glory	0
image_04666.jpg	Buttercup	1
image_07191.jpg	tiger lily	7
image_04944.jpg	giant white arum lily	3
image_08100.jpg	moon orchid	1
image_03659.jpg	Primula	0
image_05092.jpg	hard-leaved pocket orchid	17
image_07931.jpg	blanket flower	0
image_02241.jpg	barbeton daisy	2
image_05656.jpg	sweet pea	0

Table 0.9 Petals count from the edge detection or image thresholding technique

VI. CONCLUSION & FUTURE WORKS

This section captures the conclusion of the above research work and future scope of work to extend further improvement on flower image classification with petals count information in the field of image classification.

Conclusion

Petal count calculation from a flower is very challenging. There are flowers which do not have definite number of petals. There are flower images from which it is very difficult to find the petals count. The reasons are their orientation in the image, shadow from other petals or background, overlapped section with other petals, low hue and variability of shapes from another flower or within the same flower. Besides, in some flowers due to natural disturbances the flower lost its original number of petals when captured as an image. K-Means algorithm could not able to help to remove noise from all the images using same number of clusters. There are very few matches in a set of parameters when image thresholding or edge detection algorithms applied. Each individual image needs special attention for their individual petals. Due to these constraints on the petals, the algorithm worked only on 16 categories out of 102.

On the other hand, for flower category classification the feature extraction of flowers with ResNet50 model and embedding flower description using LSTM model on that feature map has achieved only approximately 75 percent accuracy. This accuracy has been further improved by using ResNet50 model solely to 90 percent by using transfer learning and fine tuning by removing few layers and applying SGD optimization.

Future Scope of work

There are lots of opportunities for the future scope of work in this kind of research. First of all, there are flowers which have sepals look like petals and this research leaves that opportunity for further study and research into that area. Deep masking technique for instance segmentation within an image could be a choice for further exploring to mask the individual petals within a flower image. Secondly the training accuracy could have been improved by increasing number of epochs and it is restricted due to system limitation. More over there are scope of many other metrics that could have been added to this research also. For example, callbacks could have been used to on internal states and statistics of the model during training for better clarity about model performance. EarlyStopping could have been used to prevent degrading of performance of the training weights. Alternative model for ResNet50 can be experimented to achieve for better performance.

References

[1] Hazem Hiary ; Heba Saadeh ; Maha Saadeh ; Mohammad Yaqub "Flower classification using deep convolutional neural networks" Volumn 12 Issue 6 pp. 855-862 2018.

[2] Navneet Dalal and Bill Triggs, "Histograms of oriented gradients for human detection", Proceedings of the IEEE CVPR 2005.

[3] DG Lowe, "Distinctive image features from scale-invariant keypoints", International Journal of Computer Vision, 2004.

[4] Gedraite, Estevao & Hadad, M.. (2011). Investigation on the effect of a Gaussian Blur in image filtering and segmentation. 393-396.

[5] Hao, Geng & Min, Luo & Feng, Hu. (2013). Improved Self-Adaptive Edge Detection Method Based on Canny. 527-530. 10.1109/IHMSC.2013.273.

[6] Leventic, Hrvoje & Keser, Tomislav & Vdovjak, Kresimir. (2018). A Fast One-Pixel Wide Contour Detection Method for Shapes Contour Traversal in Binary Images. 11-14. 10.1109/SST.2018.8564595.

[7] Lowe, David. (2004). Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision. 60. 91-110. 10.1023/B%3AVISI.0000029664.99615.94.

[8] Liu, Yuanyuan & Tang, Fan & Dengwen, Zhou & Meng, Yiping & Dong, Weiming. (2016). Flower classification via convolutional neural network. 110-116. 10.1109/FSPMA.2016.7818296.

[9] Kachouri, Rostom & Soua, Mahmoud & Akil, Mohamed. (2016). Unsupervised image segmentation based on local pixel clustering and low-level region merging. 177-182. 10.1109/ATSIP.2016.7523091.

[10] Hiary, Hazem & Saadeh, Heba & Saadeh, Maha & Yaqub, Mohammad. (2018). Flower Classification using Deep Convolutional Neural Networks. IET Computer Vision. 12. 10.1049/iet-cvi.2017.0155.

[11] Wu, Yong & Qin, Xiao & Pan, Yonghua & Yuan, Changan. (2018). Convolution Neural Network based Transfer Learning for Classification of Flowers. 562-566. 10.1109/SIPROCESS.2018.8600536.

[12] Lodh, Avishikta & Parekh, Ranjan. (2017). 2017: Flower Recognition System based on Color and GIST Features. 10.1109/DEVIC.2017.8074061.

[13] Xia, Xiaoling & Xu, Cui & Nan, Bing. (2017). Inception-v3 for flower classification. 783-787. 10.1109/ICIVC.2017.7984661.

[14] Siraj, Fadzilah & Mohd Ekhsan, Hawa & Zulkifli, Abdul Nasir. (2015). Flower image classification modeling using neural network. 81-86. 10.1109/IC3INA.2014.7042605.

[15] Xie, Saining & Girshick, Ross & Dollar, Piotr & Tu, Z. & He, Kaiming. (2017). Aggregated Residual Transformations for Deep Neural Networks. 5987-5995. 10.1109/CVPR.2017.634.

[16] Hong, Soon-Won & Choi, Lynn. (2012). Automatic recognition of flowers through color and edge based contour detection. 2012 3rd International Conference on Image Processing Theory, Tools and Applications, IPTA 2012. 141-146. 10.1109/IPTA.2012.6469535.

[17] Cho, David. (2012). Content-based structural recognition for flower image classification. Proceedings of the 2012 7th IEEE Conference on Industrial Electronics and Applications, ICIEA 2012. 541-546. 10.1109/ICIEA.2012.6360787.

[18] Qi, Wenjing & Liu, Xue & Zhao, Jing. (2012). Flower classification based on local and spatial visual cues. 670-674. 10.1109/CSAE.2012.6273040.

[19] Pei, Yong & Cao, Weiqun. (2010). A method for regional feature extraction of flower images. Proceedings of 2010 International Conference on Intelligent Control and Information Processing, ICICIP 2010. 10.1109/ICICIP.2010.5565245.

[20] Kenrick, Paul. (1999). The family tree flowers. Nature. 402. 358-9. 10.1038/46437.

[21] Otsu, Nobuyuki. (1979). A Threshold Selection Method from Gray-Level Histograms. Systems, Man and Cybernetics, IEEE Transactions on. 9. 62-66.

[22] Sobel, Irwin. (2014). An Isotropic 3x3 Image Gradient Operator. Presentation at Stanford A.I. Project 1968.

[23] Soille, Pierre & Ansoult, Marc. (1990). Automated basin delineation from digital elevation models using mathematical morphology. Signal Processing. 20. 171-182. 10.1016/0165-1684(90)90127-K.

[24] Roerdink, Jos & Meijster, A.. (2003). The Watershed Transform: Definitions, Algorithms and Parallelization Strategies. Fundam Inf. 41.

[25] Ian Goodfwllow, Yoshua Bengio, & Aaron Courville (2016). Deep Learning

[26] Y Lecun, L Bottou, Y Bengio et al., "Gradient-based learning applied to document recognition[J]", Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.

[27] Y L Cun, B Boser, J S Denker et al., Handwritten digit recognition with a back-propagation network[C] // Advances in Neural Information Processing Systems, Morgan Kaufmann Publishers Inc, pp. 465, 1990.

[28] R. Hecht-Nielsen, Theory of the backpropagation neural network[M] // Neural networks for perception (Vol. 2), Harcourt Brace & Co., vol. 1, pp. 593-605, 1992.

[29] Canziani, Alfredo & Paszke, Adam & Culurciello, Eugenio. (2016). An Analysis of Deep Neural Network Models for Practical Applications.

[30] Balázs Csanád Csáji (2001) Approximation with Artificial Neural Networks; Faculty of Sciences; Eötvös Loránd University, Hungary

[31] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. Deep Learning. The MIT Press.

[32] He, Kaiming & Zhang, Xiangyu & Ren, Shaoqing & Sun, Jian. (2016). Deep Residual Learning for Image Recognition. 770-778. 10.1109/CVPR.2016.90.

[33] Gurnani, Ayesha & Mavani, Viraj. (2017). Flower Categorization using Deep Convolutional Neural Networks.

[34] Mavani, Viraj & Gurnani, Ayesha & Shah, Jhanvi. (2017). A Novel Approach for Image Segmentation based on Histograms computed from Hue-data.

[35] Pinheiro, Pedro & Collobert, Ronan & Dollar, Piotr. (2015). Learning to Segment Object Candidates.

[36] Kawatsu, Chris & Koss, Frank & Gillies, Andy & Zhao, Aaron & Crossman, Jacob & Purman, Benjamin & Stone, Dave & Dahn, Dawn. (2017). Gesture Recognition for Robotic Control Using Deep Learning.