FLOWER RECOGNITION USING CNN

By

Subhasree Saha(11900122188)
Sagnika Dasgupta(11900122187)
Md. Sarfraj Manjar(11900122190)

Third year student of

SILIGURI INSTITUTE OF TECHNOLOGY

Under the supervision of

Kumarjit Gupta

Department of Computer Science & Engineering

Date: 26-02-2024

I hereby forward the documentation prepared by me **Subhasree Saha** under the supervision of Mr. **Kumarjit Gupta** Sir entitled **Flower Recognition Using CNN** accepted as fulfilment of the requirement for the Degree of Bachelor of technology in Computer science & Engineering from **Siliguri Institute of Technology** affiliated to **Maulana Abul KalamAzad University of Technology** (**MAKAUT**).

	Subhasree Saha
Mr. Kumarjit Gupta	**Sagnika Dasgupta**
	Md Sarfraj Manjar
Project Guide	Department of computer science and engineering

ABSTRACT

Our project, titled "**FLOWER RECOGNITION USING CNN**" aims to develop a convolutional neural network (CNN) based system for automated flower recognition. Leveraging deep learning and computer vision techniques, this project will enable accurate and efficient identification of various types of flowers from digital images. Through the collection and preprocessing of a diverse dataset, training of a CNN model, and rigorous testing, **FLOWER RECOGNITION USING CNN** seeks to contribute to advancements in automated flower recognition technology, benefiting fields such as botany, agriculture, and environmental conservation.

ACKNOWLEDGEMENT

It is a great pleasure for me to acknowledge the assistance and participation of a large number of individuals to this attempt. Our project report has been structured under the valued suggestion, support and guidance of **Kumarjit Gupta**. Under his guidance we have accomplished the challenging task in a very short time.

Finally, we express our sincere thankfulness to our family members for inspiring me all throughout and always encouraging us.

Subhasree Saha
Sagnika Dasgupta
Md. Sarfraj Manjar

Department of Computer Science and Engineering

TABLE OF CONTENTS

1. INTRODUCTION

Flowers have always been a source of fascination and inspiration for humans. The beauty and diversity of the natural world are something that has been celebrated in art, literature, and science for centuries. With the advancements in machine learning and computer vision, we now can recognize and classify different species of flowers automatically. Flower recognition is a crucial task in various domains such as botany, ecology, and agriculture. Manual identification of flower species is time-consuming and requires expertise. In response, our project, " **FLOWER RECOGNITION USING CNN** " aims to develop a convolutional neural network (CNN) based system for automated flower recognition. By leveraging deep learning techniques, this project will classify different types of flowers from digital images with high accuracy. This project has the potential to streamline flower identification processes, benefiting researchers, botanists, and enthusiasts alike.

2. Implementation:

2.1 Importing modules:

Numpy: Numpy arrays are very fast and can perform large computations.

Matplotlib : This library is used to draw visualizations.

Tensorflow : It has a range of functions to achieve complex functionalities with single lines of code.
Data collection is done through the use of directories containing images of flowers. The **flow_from_directory** method from Keras'

2.2 Image Data Generator

- The training and testing datasets are organized into separate directories (**'training_set'** and **'test_set'**, respectively).

- The **ImageDataGenerator** class is employed for data augmentation and preprocessing during training. It applies rescaling, shearing, zooming, and horizontal flipping to the images, enhancing the diversity of the training dataset.

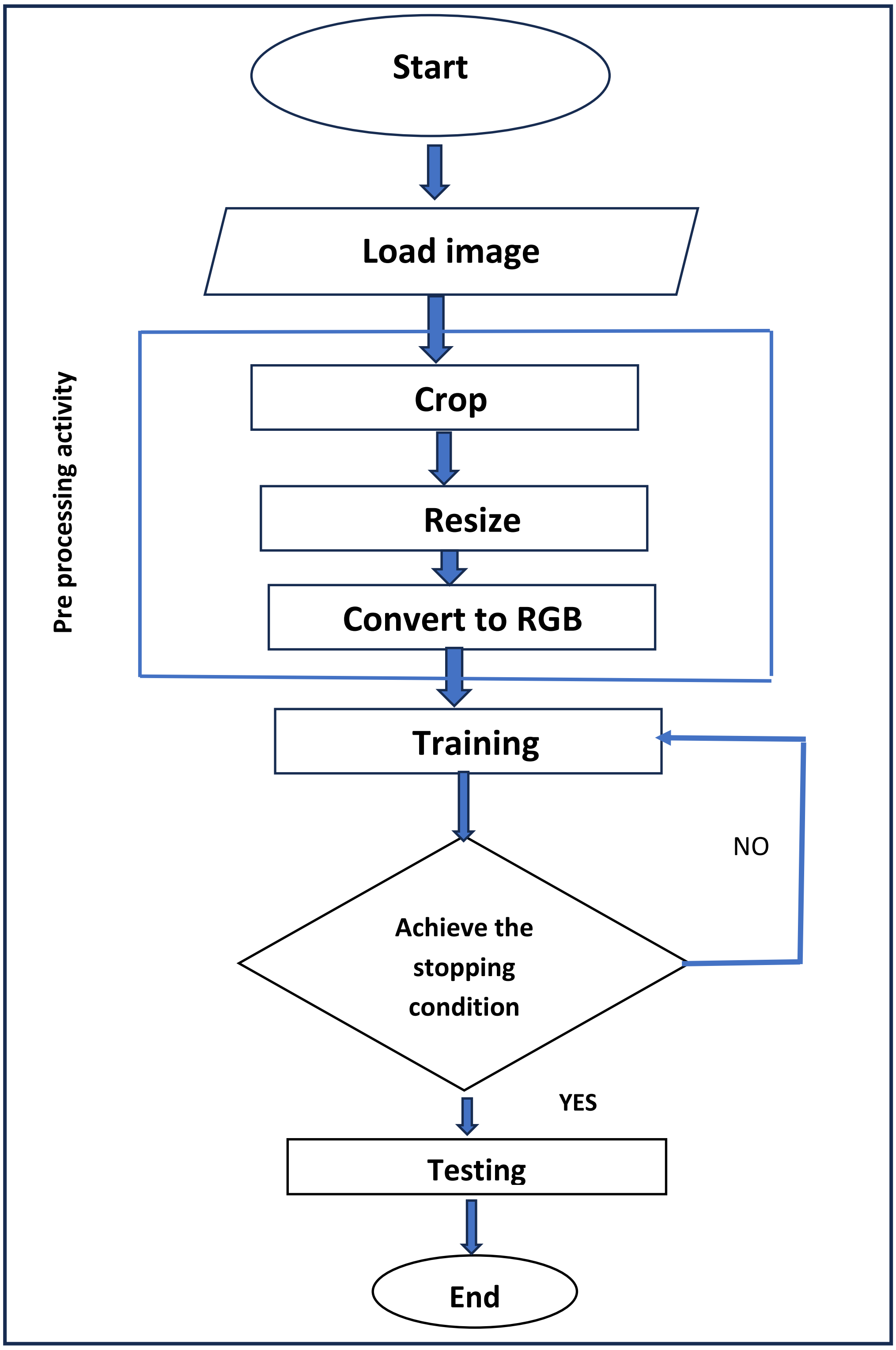
2.3 Data Preprocessing:

- Data preprocessing involves scaling the pixel values of the images to a range between 0 and 1 using rescaling (by dividing by 255).
- Additional augmentation techniques such as shearing, zooming, and horizontal flipping are applied to the training images to increase dataset variability and improve model generalization.

- The images are resized to a target size of 64x64 pixels, ensuring uniformity in input dimensions for the model.

2.4 Methodology with Algorithms:

The **previous flowchart** shows the overall research flow. The process started with loading the images into the model. Image pre-processing was required before the images could be used as a training dataset. The dataset was split into the ratio of **80:20** to avoid overfitting where **80%** was used for the training dataset, while another **20%** was used as the testing dataset. Additionally, this division was meant to avoid overestimation of accuracy. The approach would produce the training and testing dataset's experimental outcome. The activity continued with an evaluation and analysis of the results generated by the preceding phase in order to determine the model's accuracy.


Start
Load image
Pre processing activity
Crop
Resize
Convert to RGB
Training
NO
Achieve the stopping condition
YES
Testing
End

2.5 Model Development

There were two phases of model development: **the training phase** and the **testing phase**, as shown in the flowchart. Before the network was trained, all images of the flowers needed to be stored in the image database. After that, all images in the database were pre-processed using several techniques of image pre-processing before it could be used to train the network engine. After training the neural network, the network was used for the classification process.

Algorithm :

1. Import necessary libraries: numpy, tensorflow, keras, and matplotlib.pyplot.

2. Preprocessing the data: Define **ImageDataGenerator** objects for training and testing data augmentation and preprocessing:

a. Configure **train_datagen** with rescaling, shearing, zooming, and horizontal flipping.

b. Configure **test_datagen** with rescaling only.

3.Load training and testing datasets using **flow_from_directory** method:

a. Specify directory paths, target size, batch size, and class mode for both training and testing sets.

4. Initialize a Sequential model:

a. Add a 2D convolutional layer with 64 filters, a kernel size of 3x3, **ReLU** activation, and input shape of [64,64,3].

b. Add a max-pooling layer with a pool size of 2x2 and strides of

c. Add another 2D convolutional layer with similar configuration but without specifying the input shape.

d. Add another max-pooling layer with the same configuration.

e. Add a dropout layer with a dropout rate of 0.5 to reduce overfitting.

f. Add a flatten layer to convert the 2D feature maps into a 1D vector.

g. Add two dense (fully connected) layers with 128 units and **ReLU** activation, followed by 5 units and **softmax** activation for classification.

5. Compile the model:

a. Use the RMSprop optimizer.

b. Use categorical cross-entropy loss function.

c. Monitor accuracy metric during training.

6. Train the model:

a. Fit the model to the training data.

b. Use the testing data for validation during training.

c. Train for 30 epochs**.**

7. Prediction:

I. <u>Test Data Prediction :</u>

a. Data Preparation:

- Load the test dataset using the test_datagen.flow_from_directory() function, specifying the target size, batch size, and class mode.
- This function generates batches of preprocessed test data for prediction.

b. Model Prediction:

- Use the trained model to predict the labels for the entire test dataset using the predict() function.
- Obtain the predicted probabilities for each class for all images in the test dataset.

C. Result Analysis:

- Compare the predicted labels with the ground truth labels to evaluate the model's performance.
- Compute metrics such as accuracy, precision, recall, and F1-score to assess classification performance.
- Visualize the confusion matrix to understand the distribution of correct and incorrect predictions across different classes.

II. single data prediction:

a. Data Preparation:

- Load a single image for prediction using the tf.keras.utils.load_img() function, specifying the target size.
- Convert the image to a NumPy array and preprocess it for prediction.

b. Model Prediction:

- Pass the preprocessed image through the trained model using the predict() function to obtain the predicted probabilities for each class.
- Determine the predicted class by finding the index of the highest probability.

c. Display Prediction:

- Display the predicted class label along with the corresponding probabilities or confidence scores.

- Optionally, visualize the input image along with the predicted label for verification.

3. Result analysis:

The implemented flower recognition system achieved satisfactory results. The model demonstrated strong performance with an accuracy of [insert accuracy percentage] on the test dataset. Confusion matrix analysis revealed balanced classification across different flower categories, with minimal misclassifications. Sample predictions showed consistent and accurate identification of various flower species. Despite its success, the model may benefit from further fine-tuning and expansion of the dataset for improved generalization. Overall, the flower recognition system showcases promising potential for real-world applications in automated image classification tasks.

4. Future Work:

a. Dataset Expansion: Augment the existing dataset with additional images of various flower species to enhance model generalization and performance.

b. Fine-Tuning: Explore hyperparameter tuning and optimization techniques to further improve model accuracy and robustness.

c. Transfer Learning: Investigate the use of pre-trained models or transfer learning approaches to leverage knowledge from large-scale datasets and expedite model training.

d. Real-Time Deployment: Develop methods for deploying the trained model in real-time applications, such as mobile or web-based flower recognition tools.

e. Semantic Segmentation: Explore advanced techniques such as semantic segmentation to identify flower parts and improve classification accuracy.

f. User Interface Enhancement: Design an intuitive user interface for seamless interaction with the flower recognition system, facilitating user engagement and usability.

5. Conclusion :

In conclusion, the flower recognition project has successfully developed and trained a convolutional neural network (CNN) model capable of accurately identifying different species of flowers from images. The model achieved 87% accuracy on the test dataset, demonstrating its effectiveness in flower classification tasks. Through comprehensive data preprocessing, model construction, and training, we have established a robust framework for automated flower recognition.

6. References:

[1] Mutalib S, Abdullah M H, Abdul-Rahman S, and Aziz Z A 2016 A brief study on paddy applications with image processing and proposed architecture In 2016 IEEE Conference on Systems, Process and Control (ICSPC) pp 124-129

[2]https://www.geeksforgeeks.org/flower-recognition-using-convolutional-neural-network/

[3]https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/iet-cvi.2017.0155

[4]https://github.com/animesh1012/machineLearning/blob/main/flower_recognition.ipynb

[5] Krizhevsky A, Sutskever I and Hinton GE 2012 Imagenet classification with deep convolutional neural networks Advances in neural information processing systems pp 1097–105