Project - Flower Image Classification

A. Introduction and Problem Statement

For the Flower Classification project, the aim is to build a model that can accurately classify different species of flowers based on their images. This problem has applications in several fields, including agriculture, botany, and ecology, where identifying plant species is crucial for research and conservation efforts.

We found several challenges during our development, one of which is determining the appropriate hyperparameters for different models. This includes selecting the batch size, regularization, and optimizer while considering different learning rates [4]. Additionally, the algorithm's training on a dataset consisting of only one flower breed per image can result in specialization, leading to reduced accuracy when testing with images containing multiple flower breeds, causing confusion in the algorithm. We performed the data augmentation to overcome this problem.

The primary objective is to utilize PyTorch in creating deep learning models that can classify images of flowers into different categories. We plan to evaluate the model's performance using metrics such as accuracy, precision, recall, and F1 score.

B. Proposed Methodologies

B.1. Dataset

Dataset I: A dataset of flower images from Utkarsh Saxena's "Flower Classification dataset" on kaggle is broken up into 5 categories: Rose, Daisy, Sunflower, Lavender, and Lily.The images are of varying resolutions, with a majority of them having a size of 225x225 pixels. The images are in RGB format and have been preprocessed to remove any noise and resizing. There are around 5000 images(jpeg) for training in total, with about the same amounts in each class. For tasks involving flower classification, the dataset is diverse and difficult due to the images' various aspect ratios and resolutions. [1].

Dataset II: A collection of 11200 images(jpg) of the pixel size 178x256 to 648x500 of flowers from 7 different classes make up Nadyana's "Flowers" dataset on Kaggle which is a large enough number of images for training a deep learning model. Additionally, the images are annotated with labels indicating the species of each flower, providing clear and well-defined targets for the model to predict. This can make it easier to train a robust and accurate model for flower image classification [2].

Dataset III: This dataset available on Kaggle is the most versatile dataset of our selection which includes 15740 total images(jpg) of size 256x256 with 16 different classes. The images are labeled with their corresponding flower type, allowing for supervised machine learning algorithms to be applied. The images were taken in various settings and under different lighting conditions [3].

Name	Total Images	Classes
Flowers Dataset I	5k	5
Flowers Dataset II	11.2k	7
Flowers Dataset III	15.7k	16

B.2. Steps for Pre-processing

1. The Resize transformation : Resizing the images can also help to reduce the computational complexity of the model by reducing the number of pixels in each image. here, it resizes the input image to 256x256 pixels and improves its ability to generalize to new data.

2. The CenterCrop transformation: It is a transformation technique that crops the center part of a resized image with a square size of 224x224 pixels. By cropping the center of the image, the most important features of the object or scene can be retained while eliminating any extraneous background or noise.

3. ToTensor(): It is a transformation technique that converts an image from its original format into a PyTorch tensor. This is a necessary step when working with deep learning models as they require inputs in the form of tensors. Additionally, this function also scales the pixel values of the image to a range of 0 to 1, which standardizes the input data and can enhance the performance of the model.

4. The Normalize transformation : It is a transformation technique that normalizes the input tensor by subtracting the mean values and dividing by the standard deviation values. This helps to ensure that the model's inputs have a similar scale and range, which can improve training performance and accuracy.

B.3. CNN Models

ResNet18 is a potent model due to the combination of key elements. The convolutional layers, which number 18 to 152 layers and more, extract features from the input image, while residual connections enable efficient gradient propagation during training. The use of batch normalization layers reduces internal covariate shifts, improving accuracy and convergence speed [5]. However, ResNet18 may underperform if some flower classes lack sufficient image variety for detecting complex patterns.

MobileNetV2 utilizes depthwise separable convolution layers and inverted residual blocks for flower image classification. The former includes a depthwise convolution and

pointwise convolution, optimizing computation and parameter count without sacrificing accuracy. The latter leverages shortcut connections to maintain spatial resolution and increase representational power [6].

VGGNet19 employs deep convolutional layers (16) and small 3x3 filters to excel at flower image classification. The deep layers extract high-level features, while small filters capture fine-grained details and edges [8]. However, VGGNet19 may overperform when learning complex features, and its performance may be better with larger datasets.

B.4. Assessment of the Model

To assess the effectiveness of our flower classification model, we employ a range of metrics, including Accuracy, Precision, Recall, and F1 Score. We also utilize the confusion matrix to gain insight into the model's performance. In addition, we leverage the ROC curve to evaluate the binary classifier and flower image classification performance, using TPR and FPR values obtained from predictions and actual labels at different thresholds. The resulting AUC measures the model's effectiveness, with a score closer to 1 indicating better performance. To ensure robustness, we divide our labeled dataset into training, validation and testing sets.

C. Attempts at solving the problem

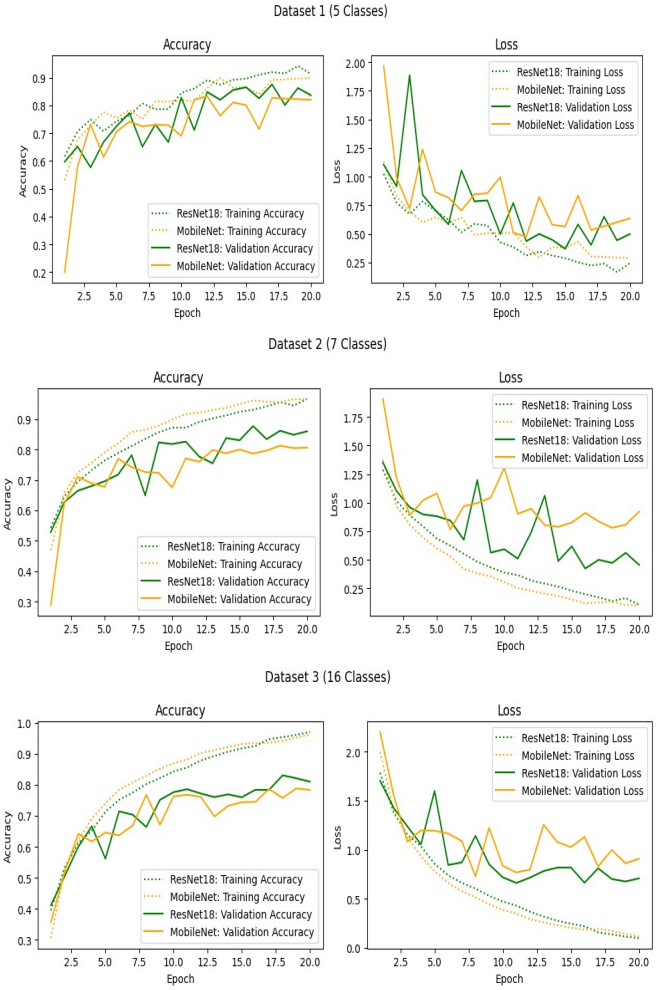
As of now, we have implemented 8 models. In which we have trained and tested all the datasets with ResNet18 and MobileNetV2. We were able to train and test two of the datasets for VGG19, but we still need to fine-tune the hyperparameters.For ResNet and MobileNet we have successfully fine tuned different Loss function and Optimizer.

Model Parameters				
Epoch	Train-Validation-Test	Loss fun.	Optim.	Batch
20	0.8/0.1/0.1	Cross-Entropy	Adam	128

We are able to achieve a good result for all the datasets with ResNet and MobileNet. Unfortunately, for VGG19 we obtained lower numbers in terms of accuracy, recall, precision and F1 score. One of the reason for this could be 19 layers inside the VGG. These many layer could lead to overfit the data easily. Another reason could be the batch size, right now we are using the batch size of 128. To handle this problem we will reduce the batch size and also try to dropout the layers of VGG19.

	ResNet-18			MobileNetV2			VGG-19		
	D1	D2	D3	D1	D2	D3	D1	D2	D3
Accuracy (%)	81	85	78	74	78	77	-	57	43
Precision (%)	81	86	81	77	80	78	-	58	44
Recall (%)	81	85	78	74	75	77	-	57	43
F1 score	81	85	78	73	77	77	-	56	41

Out of all the datasets we have used, RestNet18 is consistently outperforming the others. We have generated confusion matrices for every class in all of the datasets.



D. Future Improvements

Our current approach for a multi-label classification model on a dataset involves using a default threshold. However, we plan to improve the model's accuracy by finding optimal thresholds for each class based on the data distribution. Currently, we are utilizing 20 epochs for all of our datasets, but we plan to increase the number of epochs in the future. We will work on hyper-parameter optimization techniques to enhance the model's performance and compare the outcomes especially for VGG19. While we have made progress in visualizing the data with graphs, we aim to leverage some tools (TSNE) [7] for better analysis and more effective visualization. At the end, we will develop transfer learning models that require fewer epochs and less time to train, resulting in more efficient model.

References

[1] Flower dataset i. https://www.kaggle.com/datasets/utkarshsaxenadn/flower-classification-5-classes-roselilyetc.

[2] Flower dataset ii. https://www.kaggle.com/datasets/nadyana/flowers.

[3] Flower dataset iii. https://www.kaggle.com/datasets/l3llff/flowers.

[4] Dengsheng Lu and Qihao Weng. A survey of image classification methods and techniques for improving classification performance. *International journal of Remote sensing*, 28(5):823–870, 2007.

[5] Bashar Alathari Ruaa A. Al-Falluji, Zainab Dalaf Katheeth. Automatic detection of covid-19 using chest x-ray images and modified resnet18-based convolution neural networks. *Computers, Materials & Continua*, 66(2):1301–1313, 2021.

[6] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[7] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[8] Long Wen, X. Li, Xinyu Li, and Liang Gao. A new transfer learning based on vgg-19 network for fault diagnosis. In *2019 IEEE 23rd International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 205–209, 2019.

E. Appendix

D1: Dataset I
D2: Dataset II
D3: Dataset III
ROC: Receiver Operating Characteristic Curve
AUC: Area Under the ROC Curve