Flower Classification

A. Introduction and Problem Statement

The Flower Classification project aims to develop a model that accurately identifies different flower species from their images. This task is significant in various fields such as agriculture, botany, and ecology, where recognizing plant species plays a vital role in research and conservation efforts.

Throughout the development process, we faced multiple challenges, particularly in optimizing the hyperparameters for different models. This includes selecting the right batch size, regularization methods, and optimizers while adjusting various learning rates [4]. Moreover, training the algorithm on a dataset with only one flower species per image can lead to overfitting, which reduces accuracy when tested with images containing multiple species, resulting in confusion for the model. To mitigate this issue, we employed data augmentation techniques.

The main goal is to utilize PyTorch to build deep learning models that can classify images of flowers into various categories. We plan to evaluate the model's performance using several metrics, including accuracy, precision, recall, and F1 score.

B. Proposed Methodology

B.1. Dataset

Dataset I: features flower images from Utkarsh Saxena's "Flower Classification dataset" on Kaggle, organized into five categories: Rose, Daisy, Sunflower, Lavender, and Lily. The images exhibit varying resolutions, with the majority measuring 225x225 pixels. All images are in RGB format and have been preprocessed to eliminate noise and resize appropriately. The dataset includes approximately 5,000 JPEG images for training, with a relatively balanced distribution across each class. The range of aspect ratios and resolutions presents a significant challenge for flower classification tasks [8].

Dataset II: consists of 11,200 flower images (in JPEG format) with pixel dimensions ranging from 178x256 to 648x500, representing seven distinct classes from Nadyana's "Flowers" dataset on Kaggle. This large collection provides sufficient data for training a deep learning model. Each image is annotated with labels that identify the species, creating clear targets for the model to predict. This organized labeling enhances the ability to train a robust and precise flower classification model [6].

Dataset III:, accessible on Kaggle, is the most versatile of the datasets selected, comprising 13,716 images (in JPEG format) sized at 256x256 pixels, spanning 14 different classes. The images were captured under various conditions and lighting scenarios. This dataset supports the identification of multiple flower types, including carnations, irises, bluebells, golden English flowers, roses, fallen nephews, tulips, marigolds, dandelions, chrysanthemums, black-eyed daisies, water lilies, sunflowers, and daisies [5].

Name	Total Images	Classes
Flowers Dataset I	5k	5
Flowers Dataset II	11.2k	7
Flowers Dataset III	13.7k	14

B.2. Pre-processing Steps

- **Resize Transformation**: Resizing images reduces the computational complexity of the model by lowering the number of pixels in each image. In this step, images are adjusted to 256x256 pixels, which helps improve the model's ability to generalize to new data.
- **CenterCrop Transformation**: This method involves cropping the central region of a resized image to a square dimension of 224x224 pixels. By concentrating on the center, this technique preserves the most significant features of the object or scene while removing extraneous background elements or noise.
- **ToTensor() Transformation**: This transformation changes an image from its original format into a PyTorch tensor, which is crucial for deep learning models that require tensor inputs. Furthermore, this function rescales the pixel values to a range of 0 to 1, standardizing the input data and potentially enhancing the model's performance.
- **Normalize Transformation**: This technique normalizes the input tensor by subtracting the mean and dividing by the standard deviation of the values. This normalization ensures that the inputs to the model are on a similar scale and range, which can lead to improved training performance and accuracy.

B.3. CNN Models

ResNet18 is a robust model characterized by its combination of critical components. Featuring between 18 and over 152 convolutional layers, it effectively extracts features from input images, while residual connections allow for efficient gradient flow during the training process. The inclusion of batch normalization layers reduces internal covariate shifts, enhancing both accuracy and convergence speed [1]. Nevertheless, ResNet18 may perform suboptimally if certain flower classes do not possess enough image variety to recognize complex patterns.

MobileNetV2 utilizes depthwise separable convolution layers and inverted residual blocks for flower image classification. This structure incorporates depthwise and pointwise convolutions, which streamline computations and reduce the parameter count without sacrificing accuracy. The inverted residual blocks use shortcut connections to maintain spatial resolution and bolster the model's representational capability [7].

DenseNet121 is a convolutional neural network (CNN) architecture notable for its unique dense connectivity, which connects each layer to all preceding layers. This design enhances gradient flow and preserves information effectively throughout the network. Comprising a total of 121 layers, including convolutional and pooling layers, DenseNet121 achieves high accuracy in image classification tasks while maintaining a lower parameter count than traditional CNNs. This efficiency makes it a powerful and effective choice for a wide range of computer vision applications [3].

B.4. Assessment of the Model

To evaluate the performance of our flower classification model, we use a variety of metrics, such as Accuracy, Precision, Recall, and F1 Score. The confusion matrix also helps us gain deeper insights into the model's effectiveness. Furthermore, we analyze the ROC curve to assess the performance of both the binary classifier and flower image classification by examining True Positive Rate (TPR) and False Positive Rate (FPR) values based on predictions and actual labels across different thresholds. The Area Under the Curve (AUC) indicates the model's performance, with higher scores, closer to 1, reflecting superior effectiveness. To ensure robustness, we partition our labeled dataset into training, validation, and testing sets.

C. Attempts at Solving the problem

Based on our experimentation, we have developed and evaluated six models, including ResNet18 and MobileNetV2, across three datasets. Through these trials, we have fine-tuned various loss functions and optimizers specifically for ResNet18 and MobileNetV2 to maximize performance.

Model Parameters				
Epoch	Train-Validation-Test	Loss fun.	Optim.	Batch
30	0.8/0.1/0.1	Cross-Entropy	Adam	32

Our findings demonstrate that both ResNet18 and MobileNetV2 performed effectively across all datasets, despite facing challenges with class imbalance and noisy data. To address these issues, we applied techniques such as data augmentation, class weighting, and early stopping, which helped mitigate overfitting and ensured that the model captured important patterns in underrepresented classes. MobileNetV2 achieved the highest metrics, with an accuracy of up to 92%, precision of 94%, recall of 92%, and F1 score of 92% on Dataset D3. In comparison, ResNet18 also performed well, achieving a peak accuracy of 85% and a recall of 87% on Dataset D2.

In our next phase, we plan to train DenseNet121 on all datasets to further expand our model comparisons. Given DenseNet121's higher layer count, which increases the risk of overfitting, we will incorporate techniques such as reducing the batch size, adding dropout layers, and using batch normalization to improve performance and generalization.

	ResNet-18			**MobileNetV2**		
	D1	D2	D3	D1	D2	D3
Accuracy (%)	83	85	82	87	87	92
Precision (%)	82	87	85	88	87	94
Recall (%)	87	85	82	88	87	92
F1 score	83	85	81	88	86	92

Overall, ResNet18 exhibited robust performance across all datasets, consistently achieving metrics in the 80–85% range. Additionally, we generated confusion matrices for each class in all datasets, providing deeper insights into each model's classification effectiveness and further guiding our tuning efforts.

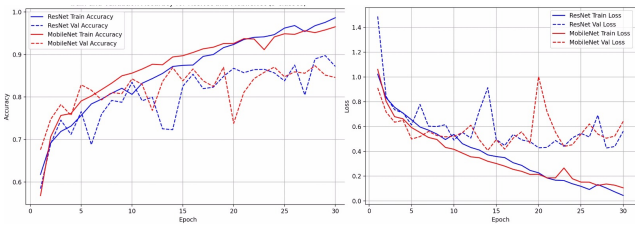

Figure 1. Dataset 1(5 Classes)

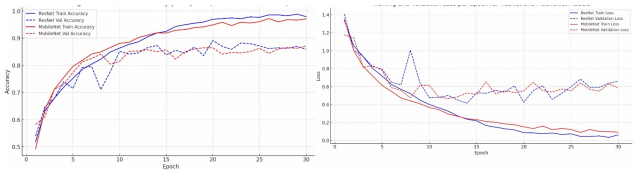

Figure 2. Dataset 2(7 Classes)

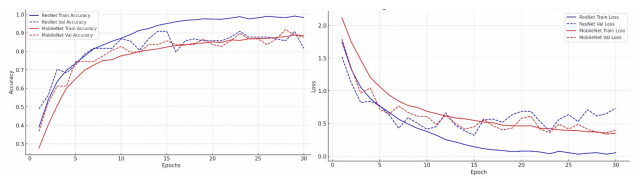

Figure 3. Dataset 3(14 Classes)

D. Future Improvements

Currently, we use a default threshold for class predictions in our multi-label classification model. To improve accuracy, we plan to adjust this by finding the best threshold for each class based on the data, which will help make the classification more accurate and specific. And, we train across all datasets for 30 epochs, but we intend to increase the number of epochs in future experiments to capture more complex patterns. Additionally, we will employ hyper-parameter optimization techniques to further enhance model performance, with a particular focus on comparing results for the Densenet121 architecture.

In terms of data visualization, we have made progress through graph-based insights and aim to incorporate advanced tools like t-SNE [2] for deeper analysis and more effective visualization, which will help in understanding the data structure and class separability in a high-dimensional space. In the final stages of development, we plan to implement 2 transfer learning models that require fewer epochs and shorter training times, resulting in greater model efficiency.

Furthermore, we will also train a DenseNet model on all three datasets, allowing us to comprehensively evaluate its performance alongside other architectures. This approach will provide us with insights into how DenseNet compares with other models in terms of accuracy, training time, and resource efficiency across the datasets.

References

[1] Bashar Alathari, Ruaa A. Al-Falluji, and 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. 1

[2] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(11), 2008. 3

[3] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4700–4708, 2017. 2

[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. 1

[5] Marquis03. Flower classification : 14 types of flower image classification. Kaggle, 2024. Available at: https://www.kaggle.com/datasets/marquis03/flower-classification. 1

[6] Nadyana. Flowers dataset. Kaggle, 2024. Available at: https://www.kaggle.com/datasets/nadyana/flowers. 1

[7] 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. 2

[8] Utkarsh Saxena. Flower classification — 10 classes. Kaggle, 2024. Available at: https://www.kaggle.com/datasets/utkarshsaxenadn/flower-classification-5-classes-roselilyetc. 1

E. Appendix

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