**Flower Classification using CNN and Transfer Learning**

**Flower Classification using CNN and Transfer Learning**

**1. Project Aim**

The objective of this project is to develop a high-accuracy deep learning model capable of classifying flower images into five distinct categories: **Daisy, Dandelion, Rose, Sunflower, and Tulip**. We employ both **custom Convolutional Neural Networks (CNNs)** and **Transfer Learning** (specifically using **ResNet50**) to explore and compare performance across different architectures.

**2. Dataset Description**

We used the **"5-flowers" dataset**, which contains labeled images for five flower categories. The dataset was structured in a directory format with subfolders for each class.

* Total Images: ~4500
* Classes: Daisy, Dandelion, Rose, Sunflower, Tulip

The images were:

* Loaded using TensorFlow's image\_dataset\_from\_directory API
* Resized to **(224, 224)** to match the input size expected by pre-trained models
* Split into **training**, **validation**, and **test** sets

**3. Data Augmentation**

To reduce overfitting and increase model robustness, strong data augmentation techniques were applied using Keras preprocessing layers:

* RandomFlip
* RandomRotation
* RandomZoom
* RandomContrast
* RandomTranslation

These augmentations helped the models generalize better to unseen images by simulating various real-world image conditions.

**4. Models & Experiments**

**4.1 Basic CNN Model**

* Architecture: Conv2D → MaxPooling → Dropout → Dense
* Performance: ~**86%** validation accuracy
* Observations: Simple architecture, limited capacity for complex features

**4.2 VGG16-based Model**

* Improvements: Added BatchNormalization and Dropout layers
* Performance: ~**89%** validation accuracy
* Observations: Better convergence and stability, but still limited by VGG16's depth and parameter count

**4.3 Transfer Learning with EfficientNet-Bo**

* Approach: Used **EfficientNet-Bo** with ImageNet pre-trained weights (excluding top layers)
* Custom Classification Head:
  + GlobalAveragePooling2D
  + Dense → BatchNormalization → Dropout → Softmax
* Performance: **93.28%** validation accuracy
* Observations: Significant performance boost from transfer learning

**4.4 Fine-tuned ResNet50**

* Strategy: Unfroze the last 50 layers of ResNet50 for fine-tuning
* Combined with advanced augmentation
* Performance: **93.63%** validation accuracy
* Observations: Marginal improvement, but more computationally intensive

**📊 5. Model Evaluation**

Each model was evaluated using validation and test datasets. Accuracy and loss curves were tracked across training epochs. Below is a performance summary:

| **Model** | **Validation Accuracy** |
| --- | --- |
| Basic CNN | ~86% |
| VGG16-based CNN | ~89% |
| EfficientNet-Bo | 93.28% |
| ResNet50 | 93.63% |

**6. Testing on External Images**

External flower images (downloaded from the web) were resized, normalized, and fed into the final model. Predictions were displayed alongside the images.

* The model performed well overall
* Misclassifications occurred when flowers had similar visual features (e.g., Daisies vs. Dandelions)

**7. Challenges Faced**

* **TPU unavailability**: Cloud resources like TPUs were not always available, requiring fallback to GPUs or CPUs.

**8. Conclusion**

This project demonstrated the power of **transfer learning** and **fine-tuning** in image classification tasks. While a basic CNN gave reasonable results, leveraging a pre-trained ResNet50 model significantly improved performance. Careful use of augmentation and layer unfreezing pushed the model to achieve **over 93% validation accuracy**, making it suitable for real-world flower recognition applications.

**🔮 Future Work**

* Deploying the model as a web app or mobile app using TensorFlow Lite or Flask
* Training on a larger, more diverse flower dataset
* Experimenting with other architectures like EfficientNet or MobileNetV2