Programming Assignment:

Flower Classification using Transfer Learning & Data Augmentation

Manually download the 5-class Flower dataset, keep it **in folders (one subfolder per class)**, and then **write code to create balanced subsets** (e.g., 50, 100, 200, 400 images per class) from disk for experiments.

Link: http://download.tensorflow.org/example_images/flower_photos.tgz

Dataset Structure:

```
flowers/
 ├── daisy/
 ├── dandelion/
 ├── rose/
 ├── sunflower/
 └── tulip/
```

You can use any other dataset, but make sure to use a subset of the dataset (if it is large) to do the experiments of TFL and Data augmentation.

Part A — Dataset Handling from Disk

1. Write a simple Python script to create class-balanced subsets (200) per class of your 5-class flower dataset from disk.
2. Count and print the number of images available for each flower category by traversing the folders.
3. Display one random image from each class using matplotlib.pyplot.
4. Write a function make_subset(src_dir, dst_dir, per_class=100, seed=42) that creates new folders containing K images per class randomly copied from the original dataset.
5. Print the total number of images and classes inside the subset folder to confirm successful subset generation.

Part B — Data Loading and Augmentation

6. Load the subset (e.g., subset_100) from disk using image_dataset_from_directory() and split it into train (80%) and validation (20%) using validation_split.

7. Create a data augmentation pipeline using tf.keras.Sequential() that includes RandomFlip, RandomRotation, and RandomZoom. Visualize 5 augmented versions of a single flower image.

Part C — Model Development

8. Build a simple CNN from scratch with 3 convolutional blocks, Flatten, Dense(128, ReLU), and Dense(5, softmax). Print the model summary.
9. Compile the model using optimizer='adam', loss='categorical_crossentropy', and metrics=['accuracy'].
10. Train the model for 20-40 epochs and plot training vs validation accuracy.

Part D — Transfer Learning

11. Use a pretrained MobileNetV2/ResNet50 as a frozen base (weights='imagenet', include_top=False) and train only the new classification head.
12. Fine-tune the top 20 layers of the MobileNetV2 backbone and compare accuracy with the frozen version.
13. Create another subset (e.g., 50 or 200 images per class) and repeat the experiment to observe how dataset size affects performance.

Part E — Evaluation & Reporting

14. For each experiment (subset size × method), record Training Accuracy, Validation Accuracy, and Model Size (in MB).
15. Plot Validation Accuracy vs. Subset Size for Scratch, Transfer (frozen), and Transfer (fine-tuned) models.
16. Generate a confusion matrix and classification report for the best-performing model.
17. Save the best model as best_flower_model.keras and write a small program to predict the class of a new flower image loaded from disk.