# **Transfer Learning with MobileNetV2 on TF-Flowers**

## **Introduction**

Exploring transfer learning using MobileNetV2, a lightweight convolutional neural network (CNN) designed for mobile and embedded vision applications. The experiment focuses on fine-tuning MobileNetV2 for flower classification using the TF-Flowers dataset, which consists of 3,670 labeled images spanning five classes: daisy, dandelion, roses, sunflowers, and tulips.

The goal is to leverage the representational power of a pre-trained MobileNetV2 model and adapt it to the flower classification task with minimal training time and retaining high accuracy.

## **Methodology**

Three main stages were followed: data preparation, model setup, and training stage.

**Data Setup**: The dataset was loaded and prepared with the following configurations: the input images were resized to 224×224. The data was split into 75% for training, 15% for testing, and 10% for validation. All images were converted to NumPy arrays, normalized, and one-hot encoded. Class names were dynamically extracted from the dataset metadata.

**Model Setup**: The model architecture was built using a customizable OOP based system configured with ImageNet weights. The input shape was set to (224, 224, 3), and all backbone layers were frozen during the initial training phase. A custom head was added, consisting of global average pooling, configurable dense hidden layers, optional dropout and batch normalization, and a final output layer with softmax activation for five classes.

**Training**: Training was carried out in two phases. In Phase 1, only the classification head was trained while the MobileNetV2 backbone remained frozen. This phase used the SGD optimizer with a learning rate of 0.01 and momentum of 0.9, categorical cross-entropy loss, and tracked accuracy, recall, precision, and F1-score over a configurable number of epochs typically 3.

In Phase 2, fine-tuning was performed by partially unfreezing the backbone using a configurable strategy: either by blocks default, unfreezing the top 2 inverted residual blocks or by layers e.g., unfreezing the top 56 layers. Additionally, the top 2 layers were always unfrozen.

## **Results**

After fine-tuning, the MobileNetV2 architecture achieved approximately 78% training accuracy and 70% testing accuracy. However, performance was relatively unstable, with fluctuations in loss and inconsistent precision and recall across classes. While some classes showed reasonable F1-scores, the overall model struggled with generalization, indicating limited effectiveness in this setup. Most errors occur between visually similar classes, such as daisy and dandelion.

## **Discussion**

MobileNetV2 showed moderate adaptability to the TF-Flowers dataset, with transfer learning offering some benefits despite limited generalization. While ImageNet feature reuse helped speed up training and slightly reduce overfitting, performance remained unstable. Partial fine-tuning by layers gave slightly better results than block-based tuning, but the gains were minimal. Although the modular pipeline allowed flexible experimentation, MobileNetV2 may not be the best fit for this fine-grained task without further tuning or architectural changes.

## **Conclusion**

Transfer learning with MobileNetV2 on the TF-Flowers dataset achieved around 70% test accuracy, showing only modest effectiveness. While using a frozen pre-trained backbone provided a solid starting point, selective fine-tuning led to only slight improvements and overall unstable performance. Lightweight models like MobileNetV2 can be efficient, but in this case, struggled to generalize well on a fine-grained dataset with limited data. Although suitable for experimentation and fast iteration, this setup requires further optimization before being considered for edge deployment or production use.