# Transfer Learning with ResNet50 on TF-Flowers

## Introduction:

ResNet50 is a powerful image classification model that can be trained on large datasets and achieve state-of-the-art results. One of its key innovations is the use of residual connections, which allow the network to learn a set of residual functions that map the input to the desired output. These residual connections enable the network to learn much deeper architectures than was previously possible, without suffering from the problem of vanishing gradients.

The architecture of ResNet50 is divided into four main parts: the convolutional layers, the identity block, the convolutional block, and the fully connected layers.

## 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 ResNet50 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 ResNet50 backbone remained frozen. This phase used the SGD optimizer with a learning rate of 0.45 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.

The best-performing unfreezing strategy for each model is based on unfreezing *individual layers*, which is more stable than unfreezing by *blocks*.  
When I unfreeze too many layers or entire blocks, the training process takes significantly more time and often results in poorer model performance.  
However, unfreezing only a few layers or a small portion of a block yields better results, both in terms of stability and accuracy.

## Results:

After fine-tuning, the ResNet50 architecture achieved approximately 90% accuracy on both the training and testing datasets.  
However, when I applied a block-wise unfreezing strategy, the training process became unstable and the test accuracy dropped to 88%, with clear signs of overfitting.  
In contrast, using a layer-wise unfreezing strategy resulted in stable training and maintained a 90% test accuracy without overfitting.

## Discussion:

ResNet50 demonstrated strong adaptability to the TF-Flowers dataset, with transfer learning significantly improving both training speed and accuracy. The reuse of ImageNet features contributed to faster convergence and reduced overfitting. Layer-wise fine-tuning provided more stable performance compared to block-wise tuning, which often led to instability and signs of overfitting. While block-based strategies yielded slightly lower test accuracy, the layer-based approach consistently reached around 90% without overfitting. Overall, ResNet50 proved to be a more robust choice for this fine-grained classification task, although further tuning may still enhance performance.

## Conclusion:

Transfer learning with ResNet50 on the TF-Flowers dataset achieved around 90% test accuracy, demonstrating strong effectiveness in handling fine-grained image classification. Leveraging a frozen pre-trained backbone offered a robust foundation, and selective fine-tuning—especially at the layer level—yielded stable training and further performance gains. Although deeper and heavier than lightweight models, ResNet50 showed better generalization and resilience to overfitting. While not as fast as MobileNetV2 in terms of iteration speed, its reliability and accuracy make it a strong candidate for production use with further optimization.