Image Classification using ResNet

Last Updated : 13 Feb, 2025

This article will walk you through the steps to implement it for image classification using Python and TensorFlow/Keras.

Image classification classifies an image into one of several predefined categories. ResNet (Residual Networks), which introduced the concept of **residual connections** to address the vanishing gradient problem in very deep neural networks.

Here are the key reasons to use **ResNet** for image classification:

- **Enables Deeper Networks**: ResNet makes it possible to train networks with hundreds or even thousands of layers without performance degradation.
- **Improved Performance**: By using residual learning, ResNet achieves better accuracy in tasks like image classification.
- **Better Generalization**: The architecture helps avoid overfitting, improving model performance on unseen data.

Image Classification Using ResNet on CIFAR-10

Here's a step-by-step guide to implement image classification using the CIFAR-10 dataset and ResNet50 in TensorFlow:

1. Import Libraries

We begin by importing the necessary libraries from TensorFlow and Keras:

```
import tensorflow as tf
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, GlobalAveragePooli
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
```

2. Load and Preprocess the CIFAR-10 Dataset

We load the CIFAR-10 dataset using **tensorflow.keras.datasets.cifar10**. Then, we normalize the pixel values of the images (by dividing by 255) to scale them to a range of 0 to 1. Lastly, we one-hot encode the labels to match the output format for categorical classification.

```
# Load CIFAR-10 dataset
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Preprocess the data
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0

# One-hot encode the labels
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

3. Load ResNet50 Pre-trained on ImageNet

We use ResNet50, pre-trained on the ImageNet dataset. The **include_top=False** parameter ensures that the fully connected layers (the classification head) are not included, so we can add our custom layers.

```
base_model = ResNet50(weights='imagenet',
                      include_top=False,
                      input_shape=(32, 32, 3))

# Freeze the base model
base_model.trainable = False
```

4. Build the Classification Model

We now build the model using the pre-trained **ResNet50** as a base. We add