**Assignment 3**

**Problem Statement:** Implement Image classification using convolutional neural networks (CNNs) for multiclass classification.

**Library:**

* TensorFlow/Keras: TensorFlow and its high-level API, Keras, are used to build, train, and evaluate CNN models for image classification.
* Matplotlib: Used for visualizing images and plotting training performance metrics (accuracy and loss).
* CIFAR-10 Dataset: This is a standard dataset used for training machine learning and deep learning models, particularly for image classification tasks.

**Theory:**

Convolutional Neural Networks (CNNs) are a class of deep neural networks primarily used for processing grid-like data, such as images. CNNs are composed of layers like convolutional layers, pooling layers, and fully connected layers, which work together to automatically extract features from images.

* Convolutional Layers extract spatial features using filters that learn from the data.
* Pooling Layers reduce the dimensionality of the data, preventing overfitting and speeding up computations.
* Fully Connected Layers map the high-level features extracted by previous layers to the output labels.

For multiclass classification, softmax activation is used in the final layer to output the probability distribution across multiple classes**.**

**Methodology:**

1. Data Loading and Preprocessing:
   * The CIFAR-10 dataset is loaded using Keras datasets and split into training and testing sets.
   * Images are normalized to bring pixel values between 0 and 1 for faster convergence during training.
2. **Model Architecture**:

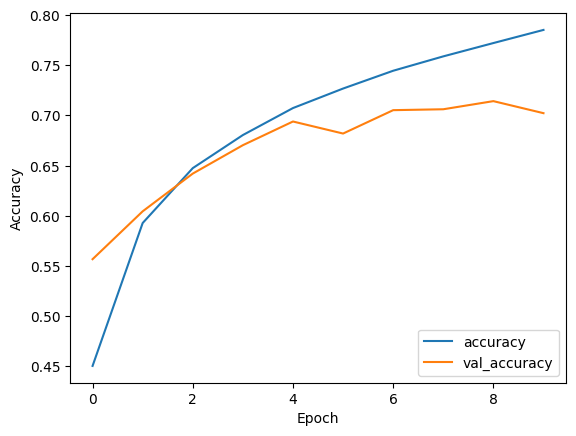
* A Sequential model is created, which allows us to stack layers linearly.
* Convolutional layers followed by max-pooling layers are added to extract hierarchical features.
* Dropout layers are used to prevent overfitting.
* The final layer is a Dense (fully connected) layer with softmax activation for multiclass classification.

1. **Model Compilation**:

* The model is compiled with a loss function suited for multiclass classification (categorical\_crossentropy), an optimizer (adam), and accuracy as a metric.

1. **Model Training**:

* The model is trained on the training dataset using mini-batches for a fixed number of epochs.



1. **Evaluation**:

* The trained model is evaluated on the test dataset to determine its performance.
* The accuracy and loss during training and validation are visualized to monitor the learning process.



**Advantages:**

1. **Automatic Feature Extraction**: CNNs automatically learn to detect relevant features from images (e.g., edges, textures, shapes), reducing the need for manual feature engineering.
2. **Efficient in Handling High Dimensional Data**: CNNs use filters and pooling layers, which reduce the dimensionality of images while retaining important information.
3. **Scalability**: CNNs can be extended to more complex and larger datasets with higher resolution images.
4. **Generalization**: CNNs can generalize well to new, unseen data, as they learn hierarchical feature representations.

**Disadvantages:**

1. **Computationally Expensive**: Training CNNs can be computationally intensive, requiring powerful GPUs for faster processing.
2. **Large Data Requirement**: CNNs typically require large amounts of labeled data to achieve high accuracy, making them difficult to apply in domains where labeled data is scarce.
3. **Risk of Overfitting**: If the model is too complex or trained for too many epochs, it can overfit the training data, leading to poor generalization on unseen data.

**Conclusion:**

CNNs are a powerful tool for image classification tasks, especially when dealing with large and complex datasets such as CIFAR-10. By leveraging layers of convolutions and pooling operations, CNNs can extract rich features from images and achieve high classification accuracy. While they require significant computational resources and data, CNNs remain one of the most effective approaches for solving image recognition problems. In this implementation, we achieved notable accuracy using a relatively simple CNN architecture, showing that even small models can perform well on moderately complex tasks like multiclass classification. ​