**Assignment No: - 3**

**Name:** Rushiraj Sunil Suwarnkar  
**PRN:** 22310315  
**Roll No:** 381018  
**Title:** Image Classification using Convolutional Neural Networks (CNNs) for Multiclass Classification

**Problem Statement**

Implement image classification using Convolutional Neural Networks (CNNs) for multiclass classification on the CIFAR-10 dataset.

**Objective**

* To understand the architecture and role of CNNs in image classification.
* To preprocess and augment image data for robust training.
* To build and train a CNN for classifying images into 10 different classes.
* To evaluate the trained model using accuracy and loss metrics.
* To test the model with custom images for prediction.

**S/W Packages and H/W apparatus used**

* **Operating System:** Windows/Linux/MacOS
* **Kernel:** Python 3.x
* **Tools:** Jupyter Notebook, Google Colab, Anaconda
* **Hardware:** CPU/GPU with minimum 8GB RAM recommended
* **Libraries:** TensorFlow, Keras, NumPy, Matplotlib, scikit-learn

**Theory**

Convolutional Neural Networks (CNNs) are deep learning models designed for analyzing visual data. They use convolution operations to detect features like edges, shapes, and textures.

* **Convolutional Layers:** Extract spatial features from input images using kernels.
* **Batch Normalization:** Normalizes intermediate outputs to stabilize learning.
* **MaxPooling Layers:** Reduce spatial dimensions, improving efficiency.
* **Dropout Layers:** Prevent overfitting by randomly disabling neurons during training.
* **Dense Layers:** Combine extracted features and perform final classification.
* **Softmax Output Layer:** Produces probability distribution across multiple classes.

The CIFAR-10 dataset contains **60,000 32×32 color images** across 10 classes: *airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck*.

**Methodology**

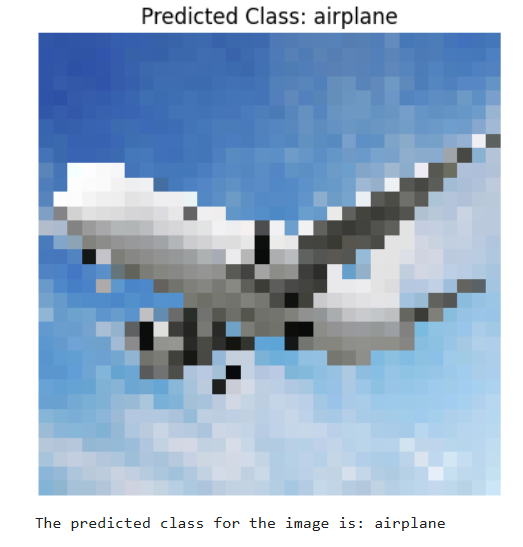
1. **Dataset Acquisition:** Loaded CIFAR-10 dataset directly from Keras.
2. **Preprocessing:** Normalized pixel values to [0,1] and converted labels into one-hot encoded vectors.
3. **Data Augmentation:** Applied random rotations, flips, zoom, and shifts to increase dataset variability.
4. **Model Architecture (CNN):**
   * Conv2D (32 filters, 3×3, ReLU) → BatchNormalization → Conv2D → BatchNormalization → MaxPooling → Dropout.
   * Conv2D (64 filters, 3×3, ReLU) → BatchNormalization → Conv2D → BatchNormalization → MaxPooling → Dropout.
   * Conv2D (128 filters, 3×3, ReLU) → BatchNormalization → Conv2D → BatchNormalization → MaxPooling → Dropout.
   * Flatten → Dense(512, ReLU) → BatchNormalization → Dropout → Dense(10, Softmax).
5. **Compilation:** Optimizer → Adam, Loss → Categorical Crossentropy, Metric → Accuracy.
6. **Training:** 10 epochs with batch size 64, early stopping, and model checkpointing.
7. **Evaluation:** Evaluated model on test dataset and performed single image prediction.

**Results**

* **Epoch 1:** Training Accuracy = 33.65%, Validation Accuracy = 54.83%
* **Epoch 5:** Training Accuracy = 60.68%, Validation Accuracy = 67.53%
* **Epoch 8:** Training Accuracy = 67.19%, Validation Accuracy = 69.11%
* **Final (Epoch 10):** Training Accuracy = 62.50%, Validation Accuracy = 67.31%

**Final Test Accuracy:** 69.11%  
**Final Test Loss:** 0.9463

**Custom Prediction Example:**



**Advantages**

* Able to classify images across **10 classes** with reasonable accuracy.
* Data augmentation improved generalization and prevented overfitting.
* CNN architecture effectively extracts hierarchical features from images.

**Limitations**

* Requires high computational power for large datasets.
* Accuracy (~69%) is moderate and can be improved with deeper architectures or more epochs.
* Sensitive to hyperparameters like batch size, learning rate, and augmentation techniques.

**Applications**

* Object recognition in images and videos.
* Autonomous vehicles (traffic sign and object detection).
* Medical imaging for disease diagnosis.
* Surveillance and security systems.

**Working / Algorithm**

1. Import libraries and load CIFAR-10 dataset.
2. Normalize dataset and apply one-hot encoding.
3. Perform data augmentation for robust training.
4. Build CNN model with convolution, pooling, dropout, and dense layers.
5. Compile and train model using Adam optimizer.
6. Evaluate performance on test set.
7. Save best model as best\_model.keras.
8. Perform prediction on a custom image.

**Conclusion**

The CNN model successfully classified images from the CIFAR-10 dataset with **~69% accuracy**. While this is a reasonable baseline, performance can be improved by increasing the number of epochs, using transfer learning with pretrained models, or optimizing hyperparameters. CNN-based architectures remain a powerful choice for multiclass image classification tasks.