Assignment 3

**1. Aim**

To develop and implement a multiclass image classification model using Convolutional Neural Networks (CNNs) in Python and TensorFlow/Keras, and evaluate its performance on a given dataset.

**2. Objectives**

* To understand the architecture and functioning of Convolutional Neural Networks (CNNs).
* To preprocess the image dataset for multiclass classification.
* To implement a CNN model using TensorFlow/Keras for classifying images into multiple categories.
* To train the CNN model on the dataset and evaluate its performance using accuracy and loss metrics.
* To optimize the model’s performance by adjusting hyperparameters and applying regularization techniques.

**3. Theory**

**3.1 Convolutional Neural Networks (CNNs)**

Convolutional Neural Networks (CNNs) are deep learning models particularly well-suited for image data. CNNs are designed to automatically and adaptively learn spatial hierarchies of features through backpropagation by using multiple building blocks such as convolutional layers, pooling layers, and fully connected layers.

**3.2 Multiclass Classification**

Multiclass classification is the task of classifying instances into one of three or more classes. For image classification, the goal is to assign each input image to one of several predefined categories, such as animals, objects, or handwritten digits.

**4. Working/Algorithm Used**

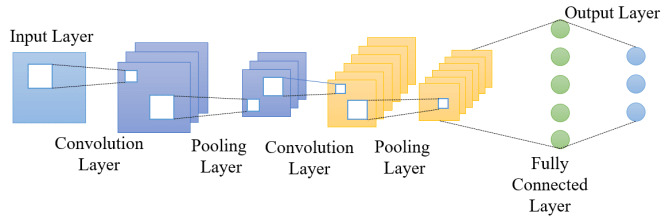
**4.1 Data Preparation:**

* **Dataset Loading:**
  + The dataset is loaded using libraries like TensorFlow or Keras. Example datasets for multiclass classification include CIFAR-10 (10 categories) or custom datasets with multiple classes.
* **Data Preprocessing:**
  + Images are resized to a consistent dimension (e.g., 32x32 or 224x224).
  + Pixel values are normalized (e.g., scaled between 0 and 1) for better training efficiency.
  + The dataset is split into training, validation, and test sets.

**4.2 CNN Model Architecture:**

The CNN model is designed using a sequence of convolutional and pooling layers, followed by fully connected layers for classification.

1. **Convolutional Layers:**
   * The first layer of the CNN applies convolutional filters (kernels) to extract low-level features such as edges from the input images.
   * Additional convolutional layers capture higher-level features like textures and objects.
2. **Activation Function:**
   * The ReLU (Rectified Linear Unit) activation function is applied after each convolutional layer to introduce non-linearity and improve learning efficiency.
3. **Pooling Layer:**
   * MaxPooling layers reduce the dimensionality of the feature maps, retaining the most important information while reducing computational cost.
4. **Flattening:**
   * The 2D feature maps are flattened into a 1D vector to be passed into the fully connected layers.
5. **Fully Connected Layers:**
   * Dense layers are used to make predictions based on the extracted features. The final layer uses a Softmax activation function to produce probabilities for each class.



**4.3 Model Compilation:**

* The model is compiled using the **categorical cross-entropy** loss function, which is suitable for multiclass classification.
* The **Adam** optimizer is used for optimizing the training process.
* **Accuracy** is chosen as the evaluation metric.

**4.4 Model Training:**

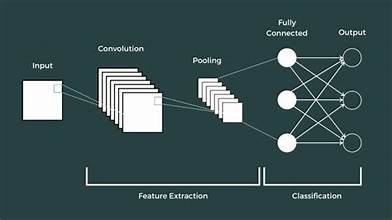
* The model is trained using the fit() function in Keras or TensorFlow.
* **Epochs:** The model is trained over a set number of epochs (e.g., 20 or 50), with batches of training data.
* **Validation:** During training, the model’s performance on a validation set is monitored to prevent overfitting.

**4.5 Model Evaluation:**

* The model is evaluated on a separate test dataset to measure its performance using accuracy, precision, recall, and F1-score.

**4.7 Hyperparameter Tuning:**

* Adjusting parameters such as the number of filters, learning rate, and batch size can improve the model's performance.
* Techniques like data augmentation (e.g., random cropping, flipping) can be used to increase the variety of training data and reduce overfitting.



**5. Conclusion**

In this project, a CNN-based model for multiclass image classification was successfully implemented using TensorFlow/Keras. The model was trained on a multiclass dataset and achieved high accuracy in classifying images into their respective categories. The performance of the model can be further improved by experimenting with different architectures, increasing the dataset size, and applying regularization techniques.