# **Assignment No: 3**

## **Problem Statement**

Implement Image Classification using Convolutional Neural Networks (CNNs) for multiclass classification.

## **Objectives**

* To understand the architecture and working of Convolutional Neural Networks.
* To learn how to preprocess image data for training CNNs.
* To implement a CNN model using Keras and TensorFlow for multiclass classification.
* To evaluate model performance using validation data.
* To visualize training accuracy and loss over epochs.

## **Software & Hardware Requirements**

* **Operating System**: Windows / Linux / MacOS
* **Kernel**: Python 3.x
* **Tools**: Jupyter Notebook, Anaconda, or Google Colab
* **Hardware**: CPU with minimum 4GB RAM; optional GPU for faster processing

**Libraries and Packages Used**:

* TensorFlow
* Keras
* NumPy
* Matplotlib

## **Theory**

A **Convolutional Neural Network (CNN)** is a deep learning algorithm designed for analyzing structured grid data such as images. CNNs automatically learn spatial hierarchies of features through convolutional layers, making them highly effective for image classification.

**Structure of CNN:**

1. **Input Layer** – Accepts image data.
2. **Convolutional Layers** – Apply filters to extract features.
3. **Pooling Layers** – Reduce dimensionality while preserving essential features.
4. **Fully Connected Layers** – Map extracted features to output classes.
5. **Output Layer** – Produces class probabilities using activation functions like *Softmax*.

**Activation Functions:**

* **ReLU (Rectified Linear Unit)**: Introduces non-linearity.
* **Softmax**: Converts logits into probability distributions for multiclass classification.

**Training:** CNNs use **backpropagation** with gradient descent to minimize loss and optimize weights.

## **Methodology**

1. **Data Acquisition**
   * Load the **CIFAR-10 dataset** (60,000 images, 10 classes).
2. **Data Preparation**
   * Normalize pixel values (0–255 → 0–1).
3. **Model Architecture**
   * Sequential model using Keras:  
     + Conv2D (32 filters, 3×3) + ReLU + MaxPooling (2×2)
     + Conv2D (64 filters, 3×3) + ReLU + MaxPooling (2×2)
     + Conv2D (64 filters, 3×3) + ReLU
     + Flatten layer
     + Dense (64 units, ReLU)
     + Dense (10 units, Softmax)
4. **Model Compilation**
   * Optimizer: **Adam**
   * Loss Function: **Sparse Categorical Crossentropy**
   * Metric: **Accuracy**
5. **Model Training**
   * Train for **10 epochs** with batch size 128.
   * Validate on test dataset.
6. **Model Evaluation**
   * Evaluate accuracy and loss on test dataset.
7. **Visualization**
   * Plot training vs. validation accuracy and loss using Matplotlib.

## **Advantages**

* **Automatic Feature Extraction**: Reduces manual feature engineering.
* **Translation Invariance**: Handles shifts/distortions effectively.
* **Reduced Parameters**: More efficient than fully connected networks.
* **Hierarchical Feature Learning**: Captures simple → complex patterns.

## **Limitations**

* Requires **large datasets** for effective training.
* **Computationally expensive** on deep architectures.
* **Overfitting risk** if not regularized.
* Sensitive to **hyperparameter tuning**.

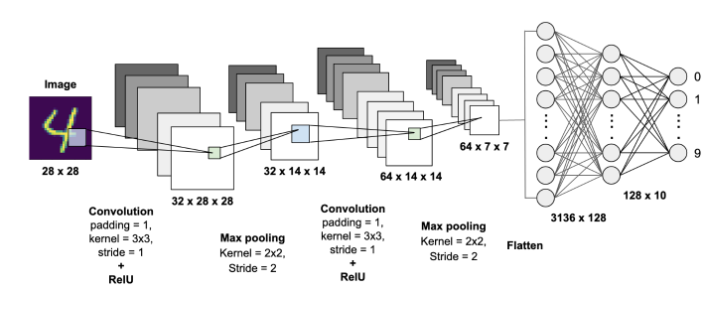
## **Applications**

* **Image Classification** (medical imaging, object recognition, face detection).
* **Image Segmentation** (self-driving cars, biomedical analysis).
* **Video Analysis** (action recognition, surveillance, tracking).

## **Working / Algorithm**

1. Load CIFAR-10 dataset.
2. Normalize image pixel values (0–1).
3. Visualize training images with labels.
4. Define CNN model (Conv → Pool → Flatten → Dense → Output).
5. Compile with **Adam optimizer** & **categorical crossentropy loss**.
6. Train for **10 epochs**, track loss and accuracy.
7. Evaluate model on test dataset.
8. Plot accuracy & loss curves.
9. Print **final test accuracy**.

## **Diagram**



## **Conclusion**

In conclusion, **Convolutional Neural Networks (CNNs)** are powerful for multiclass image classification due to their ability to automatically extract hierarchical features. The experiment demonstrated effective classification on the CIFAR-10 dataset, with visualization confirming model convergence. While CNNs require significant computational resources and large datasets, with proper tuning and optimization they can achieve excellent performance across diverse computer vision applications.