Name: Aziz Sayyad
Roll No: 381069
Batch: A3

Assignment No: - 3

Problem Statement:
Implement image classification using Convolutional Neural Networks (CNNs) in Python with Keras and TensorFlow for multiclass classification tasks.

Objective:

- To understand the architecture and working of Convolutional Neural Networks.
- To preprocess image data for deep learning models.
- To implement CNNs for multiclass image classification.
- To evaluate classification performance using validation/test datasets.
- To visualize model learning through accuracy and loss curves.

S/W Packages and H/W apparatus used:

- **Operating System:** Windows/Linux/MacOS
- **Kernel:** Python 3.x
- **Tools:** Jupyter Notebook, Anaconda, or Google Colab
- **Hardware:** CPU with minimum 4GB RAM; GPU recommended for faster training
- **Libraries and packages used:** TensorFlow, Keras, NumPy, Pandas, Matplotlib, Scikit-learn, OpenCV (optional for preprocessing)

Theory:
A Convolutional Neural Network (CNN) is a deep learning architecture designed for image classification and computer vision tasks. It automatically extracts spatial features from images using convolutional filters, reducing the need for manual feature engineering.

Structure:

- **Input Layer:** Accepts raw image data (e.g., 64×64 or 128×128 pixels with RGB channels).
- **Convolutional Layers:** Extract spatial features using learnable filters.
- **Pooling Layers:** Down-sample feature maps to reduce dimensionality.
- **Flatten Layer:** Converts 2D feature maps into a 1D vector.
- **Fully Connected Layers:** Learn non-linear combinations of features.
- **Output Layer:** Softmax activation produces probability distribution across multiple classes.
- **Activation Functions:** ReLU in hidden layers, Softmax in output.
- **Loss Function:** Categorical Crossentropy for multiclass classification.
- **Backpropagation:** Optimizes weights using gradient descent (Adam optimizer commonly used).

Methodology:

1. **Data Acquisition:** Collect a dataset with multiple image categories (e.g., CIFAR-10, MNIST, custom dataset).
2. **Data Preparation:**
   - Resize images to a fixed size.
   - Normalize pixel values between 0 and 1.
   - Apply one-hot encoding to labels.
   - Split into training, validation, and test sets.
3. **Model Architecture:**
   - Conv2D(32 filters, ReLU) → MaxPooling2D.
   - Conv2D(64 filters, ReLU) → MaxPooling2D.
   - Conv2D(128 filters, ReLU) → MaxPooling2D.
   - Flatten layer.
   - Dense(128, ReLU).
   - Dense(number of classes, Softmax).
4. **Model Compilation:** Adam optimizer, categorical crossentropy loss, accuracy metric.
5. **Model Training:** Train CNN on training data and validate using validation data.
6. **Model Evaluation:** Evaluate model performance on unseen test data using accuracy, confusion matrix, and classification report.
7. **Visualization:** Plot training and validation accuracy/loss curves.

Advantages:

- Automatically extracts spatial features from images.
- High accuracy in image-related tasks compared to traditional ML methods.
- Scalable to large and complex datasets.
- Supports transfer learning with pre-trained models (VGG, ResNet, EfficientNet).
- Reduces manual feature engineering effort.

Limitations:

- Requires large labeled datasets for effective training.
- Computationally expensive and time-consuming without GPUs.
- Sensitive to overfitting on small datasets.
- Limited interpretability ("black-box" nature).
- Performance depends on hyperparameter tuning and data quality.

Applications:

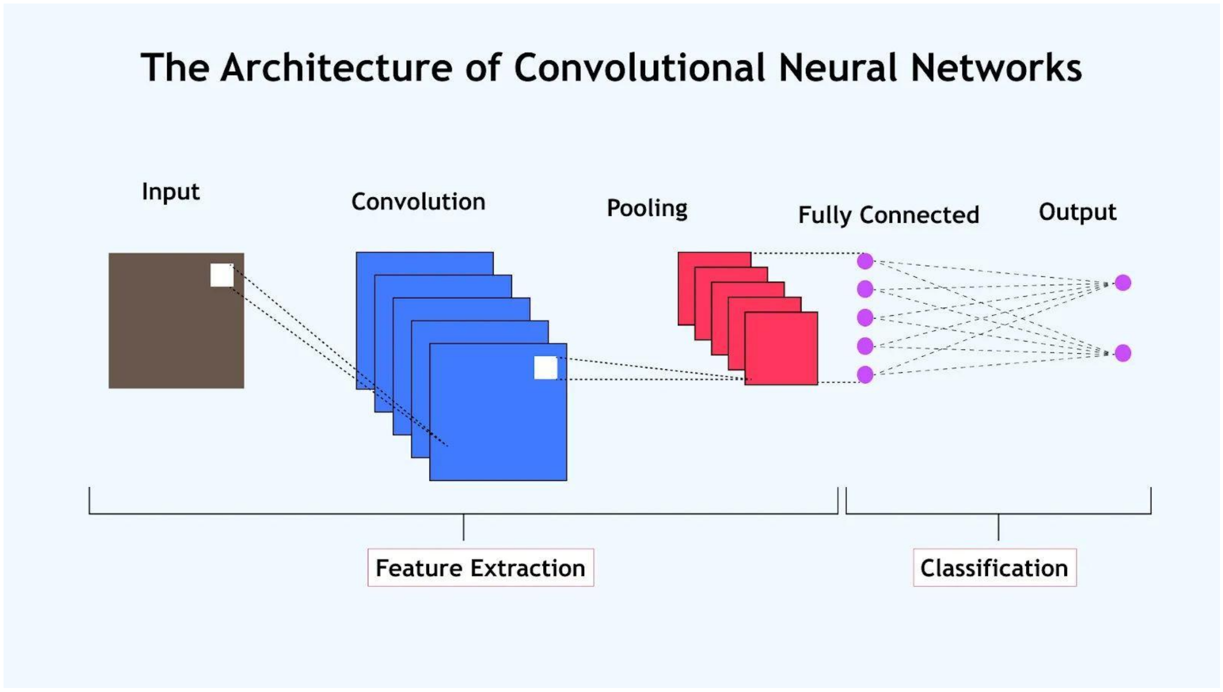
- Object recognition and detection (e.g., cars, animals, products).
- Medical imaging (e.g., cancer detection, X-ray classification).
- Face recognition and verification.
- Autonomous vehicles (road sign and pedestrian recognition).
- Industrial defect detection.

Working / Algorithm:

1. Import required libraries (TensorFlow, Keras, NumPy, Matplotlib, etc.).
2. Load dataset (e.g., CIFAR-10).

3. Preprocess images (resize, normalize).
4. Apply one-hot encoding to class labels.
5. Build CNN model with Conv2D, MaxPooling, Flatten, Dense, and Softmax layers.
6. Compile the model using Adam optimizer and categorical crossentropy loss.
7. Train the model with training data and validate with validation set.
8. Evaluate the model on test data using accuracy and confusion matrix.
9. Visualize accuracy and loss curves across epochs.
10. Deploy the trained model for real-world image classification.

Diagram:



Conclusion:
Convolutional Neural Networks (CNNs) provide a powerful and efficient method for multiclass image classification. By automatically learning spatial hierarchies of features, CNNs achieve high performance across diverse applications. Despite their computational cost and dependency on large datasets, CNNs remain the state-of-the-art approach in computer vision tasks and are widely adopted in real-world systems.