|  |  |
| --- | --- |
| **Ex No: 3**  **Date: 30-08-2024** | **Small Image Classification Using Convolutional**  **Neural Network (CNN)** |

**Objective:**

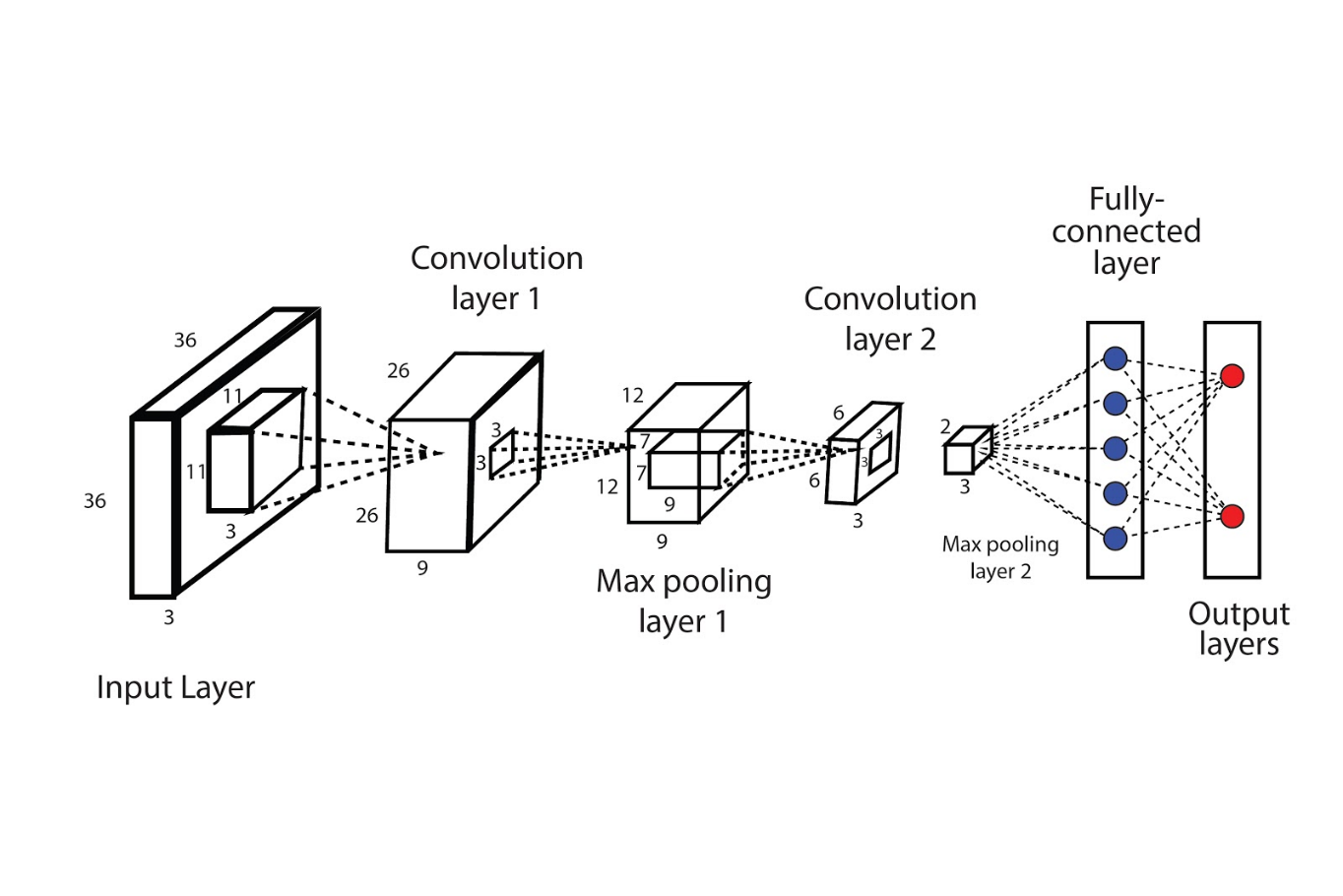
The objective of this Assignment is to classify small images from the CIFAR-10 dataset using a Convolutional Neural Network (CNN). The CIFAR-10 dataset consists of 60,000 32x32 color images belonging to 10 different classes. Our goal is to build a model that can accurately distinguish between these classes, such as airplanes, cars, and animals. The model will be trained using labeled data, and its performance will be evaluated on unseen test data. By optimizing the network, we aim to achieve high classification accuracy.

**Descriptions:**

This project uses the CIFAR-10 dataset, which contains small 32x32 color images across 10 different categories. These categories include airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. Convolutional Neural Networks (CNNs) are well-suited for image classification tasks due to their ability to automatically detect important features like edges, shapes, and textures in images.

The model is built using TensorFlow and Keras, leveraging multiple convolutional layers followed by pooling layers to reduce the dimensionality of feature maps. After training, the model will be evaluated on unseen test data to assess its performance. The key challenge here is to ensure the model generalizes well across different image classes while avoiding overfitting.

**Model:**



 **Input Layer**: The model takes input images of shape 32x32 with 3 colour channels (RGB).

 **Convolutional Layers**: These layers apply filters to the input image, detecting spatial features like edges, patterns, and textures. We use two convolutional layers with ReLU activation.

 **Pooling Layers**: Max-Pooling layers are used after convolution to reduce the spatial size of the feature maps and control overfitting by summarizing the presence of features in patches of the feature maps.

 **Dense Layers**: After flattening the output from the convolutional layers, we apply fully connected (dense) layers for classification. The last layer uses softmax activation for multi-class classification.

 **Optimizer**: We use the Adam optimizer, which adapts the learning rate during training and improves convergence speed.

**Evaluation**: The model's performance is measured by accuracy and evaluated using a test set. Metrics like confusion matrix and classification report are used to assess performance across each class.

**Building the parts of algorithm**

1. **Data Preprocessing**:
   * Normalize the images by scaling pixel values to the range [0, 1].
   * Reshape the labels from a 2D format (one-hot encoding) to a 1D format.
2. **CNN Architecture**:
   * **Convolutional Layer**: Extract features from the image using convolution filters.
   * **Activation Function (ReLU)**: Introduces non-linearity after each convolution.
   * **Max-Pooling Layer**: Reduces dimensionality by keeping the most prominent features.
   * **Flatten Layer**: Converts the 2D feature maps into a 1D vector for the fully connected layer.
   * **Dense Layer**: Fully connected layers for classification.
   * **Soft-max Output Layer**: Provides probabilities for each of the 10 classes.
3. **Model Compilation**:
   * Use the Adam optimizer and sparse categorical cross-entropy as the loss function.
   * Track accuracy as the performance metric.
4. **Training**:
   * Train the model for a fixed number of epochs (e.g., 10 epochs), using the training data.
   * Validate the model performance on a holdout validation set.
5. **Evaluation**:

* **Evaluate the model on the test data to measure performance:** After training, the model’s performance is tested on unseen data (the test set). This final evaluation tells us how well the model generalizes to new data and whether it can make accurate predictions outside of the training set.
* **Generate classification reports and confusion matrices to assess performance across each class:** A classification report provides detailed metrics like precision, recall, and F1-score for each class. The confusion matrix shows how often predictions were correct or misclassified for each class.
* **Use visualization techniques to display sample images and predictions:** Visualizing sample images along with their predicted labels and actual labels can help in understanding how the model is performing. It’s also useful for debugging purposes, allowing you to see which types of images the model may be misclassifying.

**GitHub Link:**