Name: Joselyn Riana Manoj
USN: 1RVU22CSE071

Ex No: 4 Date: 28-08-2024	Handwritten digit recognition using CNN

Objective:

To design, implement, and evaluate a Convolutional Neural Network (CNN) for recognizing handwritten digits from the MNIST dataset. The primary objective is to develop a model that can classify images of handwritten digits (0-9) with high accuracy and create a user interface to interact with the model for digit prediction.

Descriptions:

In this lab, we will develop a deep neural network that classifies images into two categories: cat or non-cat. The dataset comprises images labeled as either cat (1) or non-cat (0), making this a binary classification problem. The task is further complicated by the inclusion of multiple hidden layers within the neural network.

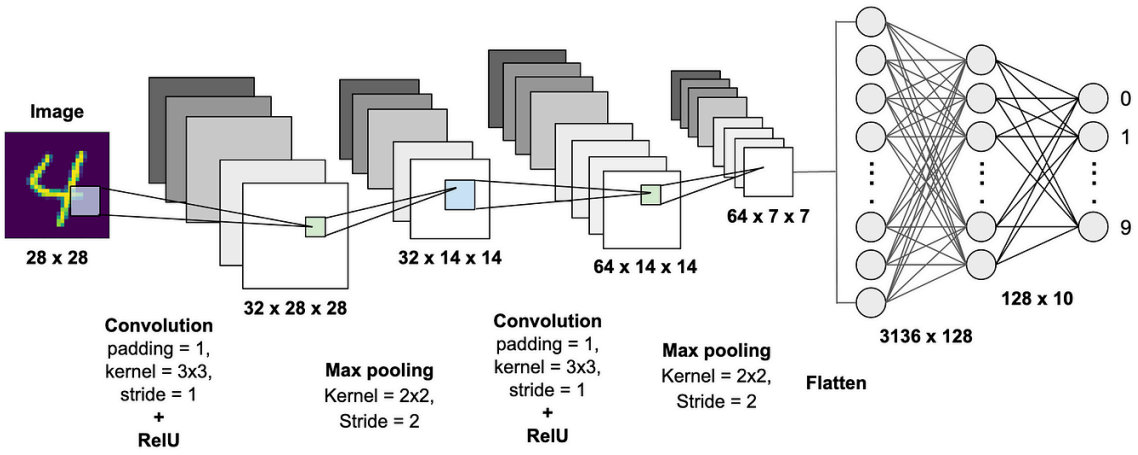
The neural network is structured with both linear and non-linear layers, organized as follows:

- **Input Layer:** Accepts the image data (64x64x3) and flattens it into a single vector.
- **Hidden Layers:** Utilize ReLU activation functions to introduce non-linearity, which enhances the model's ability to detect complex patterns.
- **Output Layer:** Employs a sigmoid activation function to produce a binary output, indicating whether the image is a cat or non-cat.

The model is trained using a cross-entropy loss function, which quantifies the difference between the predicted and actual labels. This loss function guides the model in minimizing errors over multiple training iterations. The parameters of the model will be optimized using backpropagation.

Name: Joselyn Riana Manoj
USN: 1RVU22CSE071

Model:



Building the parts of algorithm:

Initialize Parameters:

1. **Define Input and Output Dimensions:**
   - **Input Layer:** Set to handle input images, e.g., `(28, 28, 1)` for MNIST.
   - **Hidden Layers:** Define the number and size of filters (e.g., 32 and 64 filters).
   - **Output Layer:** Number of units equal to the number of classes (e.g., 10 for MNIST).

Forward Propagation:

1. **Convolutional Layers:**
   - Apply 2D convolutions to extract features from the input images.
   - Use activation functions like ReLU to introduce non-linearity.
2. **Pooling Layer:**
   - Apply max pooling to reduce the spatial dimensions of the feature maps.
3. **Flatten Layer:**
   - Flatten the 2D feature maps into a 1D vector to feed into fully connected layers.
4. **Fully Connected Layers:**
   - Apply dense layers to compute class scores from the flattened vector.

- Use activation functions like ReLU for hidden layers and softmax for the output layer.

5. **Predicted Outputs:**
   - Compute the final probabilities for each class using the softmax activation.

Compute Cost:

1. **Loss Function:**
   - Calculate the cross-entropy loss between the predicted class probabilities and the true labels.

Backward Propagation:

1. **Gradient Calculation:**
   - Use the chain rule to compute gradients of the loss function with respect to each parameter.
2. **Parameter Update:**
   - Update the model parameters (weights and biases) using the computed gradients to minimize the loss function.

Train the Model:

1. **Epochs:**
   - Repeat the forward and backward propagation steps for a specified number of epochs.
2. **Optimization:**
   - Use an optimizer (e.g., Adadelta) to adjust the learning rates and improve convergence.

GitHub Link:
https://github.com/joselynrianaaa/DeepLearning_Labs