|  |  |
| --- | --- |
| **Ex No: 4**  **Date: 28/08/2024** | **Lab Record: Handwritten Digit Recognition using CNN** |

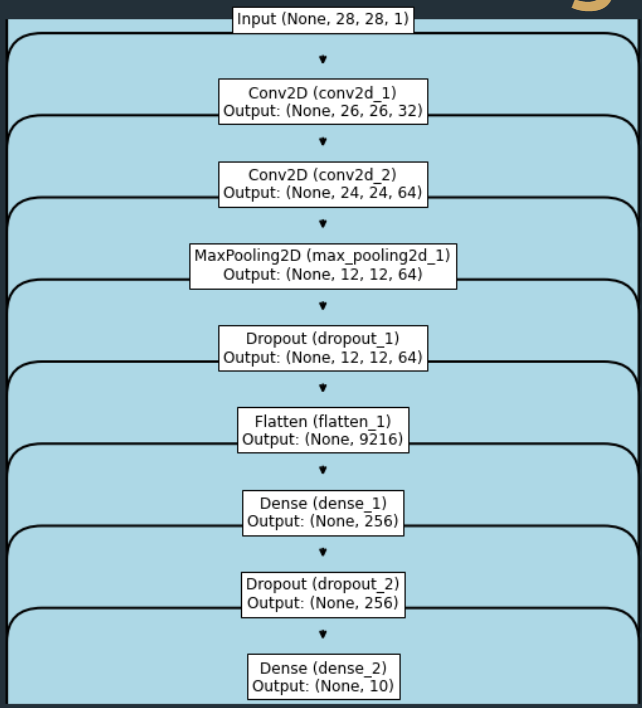
**Objective:**

To design and implement a Convolutional Neural Network (CNN) for recognizing handwritten digits from the MNIST dataset, and evaluate its performance using metrics like accuracy and loss.

**Description:**

The objective of this lab is to explore CNNs as an efficient deep learning model for image classification tasks. We focus on recognizing handwritten digits from the MNIST dataset, which contains 28x28 grayscale images of digits (0-9). Each digit belongs to one of the 10 classes. The CNN will be trained on this dataset and evaluated on unseen test data to determine its accuracy in classifying digits.

**Model:**



**Building Blocks of the Algorithm:**

1. **Dataset Preprocessing:**

The MNIST dataset is loaded, consisting of 60,000 training samples and 10,000 testing samples. Each image has dimensions of 28x28 pixels. The dataset is reshaped into the required format for the CNN model, where each image is represented as a 3D tensor: width, height, and channels (grayscale, so 1 channel). Additionally, the pixel values are normalized to the range `[0,1]` by dividing by 255.

The labels for each image are converted from categorical format into a one-hot encoding to be compatible with the categorical cross-entropy loss function.

**2. CNN Architecture:**

A sequential model is used to define the layers in the CNN architecture:

- Convolutional Layers: The first two layers apply 32 and 64 filters, respectively, each with a 3x3 kernel size. These layers extract important features such as edges, textures, and shapes from the input images. The activation function used is ReLU, which introduces non-linearity and speeds up convergence.

- MaxPooling Layer: After the convolutional layers, a max-pooling operation reduces the spatial dimensions (height and width) by taking the maximum value in 2x2 regions. This reduces computational complexity and captures the most significant features.

- Dropout: To prevent overfitting, a dropout layer randomly sets 25% of neurons to zero during training. This helps the model generalize better to unseen data.

- Flatten Layer: The feature maps produced by the convolutional and pooling layers are flattened into a 1D vector to be used by fully connected (dense) layers.

- Fully Connected Layers: A dense layer with 256 units and ReLU activation further processes the flattened feature vector. A dropout layer with a rate of 50% is added to reduce overfitting.

- Output Layer: The final dense layer consists of 10 neurons, corresponding to the 10 digit classes (0-9). A softmax activation function is applied to produce a probability distribution over the 10 classes.

3. Model Compilation:

- Loss Function: The categorical cross-entropy loss is used since this is a multi-class classification problem.

- Optimizer: The Adadelta optimizer is chosen for adaptive learning rate management, which adjusts the learning rate based on the gradients during training.

- Metrics: Accuracy is used to track the model’s performance during training and testing.

4. Model Training:

The model is trained over 10 epochs with a batch size of 128, meaning the model processes 128 samples in one batch before updating the weights. The training data is split into training and validation sets, and the model learns through multiple forward and backward passes over the data. As training progresses, the loss function is minimized, and the accuracy improves.

5. Model Evaluation:

Once training is complete, the model is evaluated on the test set to compute its generalization performance. The test accuracy measures how well the model performs on unseen data, and the test loss quantifies the error made by the model in its predictions.

6. Saving the Model:

After the training and evaluation, the trained model is saved to a file (`mnist.h5`) for future use. This allows the model to be loaded and used for inference without retraining.

7. Summary:

The model summary provides details of each layer, including the output shapes and the number of parameters (weights and biases). This is important for understanding the model's complexity and memory requirements.

This lab demonstrates the powerful feature extraction capabilities of CNNs in image classification tasks. By training a CNN on the MNIST dataset, we successfully built a model that can recognize handwritten digits with high accuracy. The architecture leverages convolutional layers for automatic feature extraction, pooling layers for dimensionality reduction, and fully connected layers for classification.

**GitHub Link:** https://github.com/dyuthiramesh/Deep\_Learning\_Elective/blob/main/Sem5/Lab4/Handwritten\_Digit\_CNN\_distri.ipynb