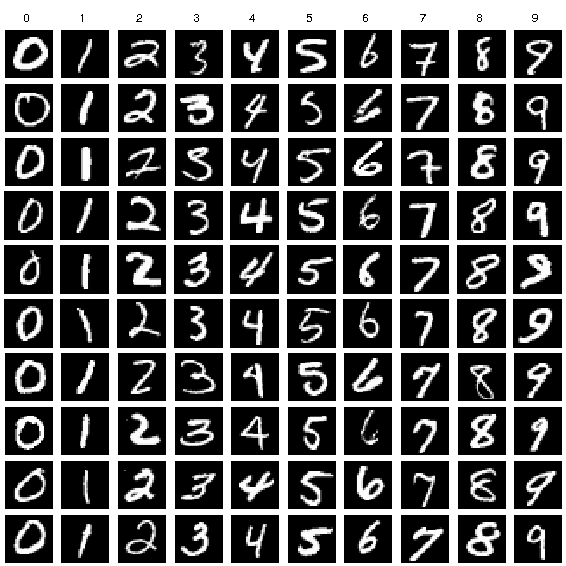
| **Ex No: 3a**  **Date: 22 Oct** | **Handwritten Digit Recognition using CNN** |
| --- | --- |

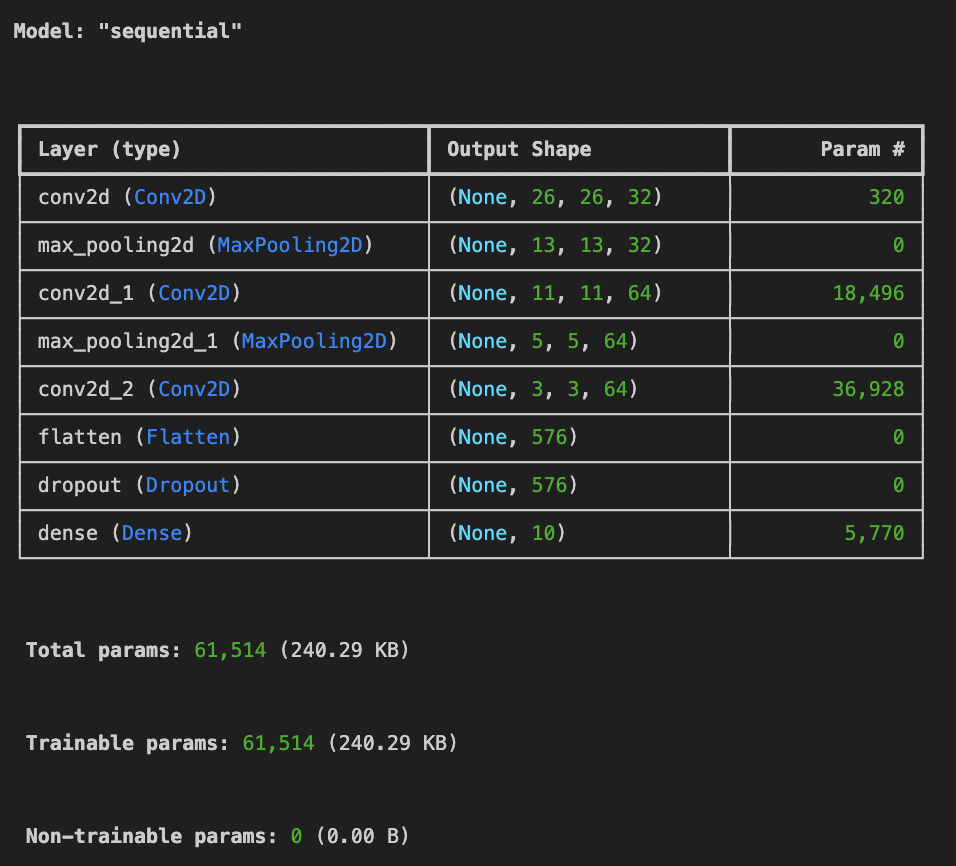
**Objective:**The objective is to build and evaluate a convolution neural network model using TensorFlow and Keras for multi-class classification(10 classes) of handwritten digits (0-9) using MNIST dataset. It focuses on training, validation, and testing while analysing the model's performance by plotting loss over epochs.

**Description:**

Handwritten Digit Recognition Using CNN is a deep learning approach to identifying and classifying handwritten digits from images, typically using datasets like MNIST. The task involves building a Convolutional Neural Network (CNN) that can automatically recognize digits (0-9) from grayscale images of fixed size (28x28 pixels). The CNN architecture extracts spatial features through convolutional layers, applying filters to detect patterns, followed by pooling layers that downsample the feature maps. After flattening these feature maps, fully connected layers are used to make predictions. This method is highly effective for digit recognition, achieving high accuracy due to CNN’s ability to capture local patterns in the images. Applications include digit classification for automated systems, such as postal code recognition or digit-based authentication systems.



**Model Summary:**

****

**Fig1**: A sequential CNN model for Handwritten digit recognition.

**Building the parts of the algorithm**

Here are the steps involved in building each part of the algorithm:

1. **Data Preprocessing**: The dataset is loaded and preprocessed into training and testing sets. The class vectors of the input data is converted to binary class matrices.
2. **Model Architecture**:
   * The neural network model is built using Keras. It contains multiple layers, including input, hidden (dense and dropout), and output layers.
   * ReLU activation functions are used in hidden layers, while softmax is used in the output layer for multi-class classification.
3. **Compilation**:
   * The model is compiled using a loss function (categorical cross-entropy) and an optimizer (Adadelta), with metrics set to monitor accuracy.
4. **Training**:
   * The model is trained over multiple epochs, with validation accuracy and loss tracked during each epoch.
5. **Evaluation**:
   * The final performance is evaluated on a test set which gives the accuracy and loss.
   * Two plots are generated to visualize training and validation loss over time.
   * The model is tested with some test images to understand the prediction and its probability.

**Inference**

The model achieved an impressive accuracy of 99.14% on the handwritten digit recognition task. This high level of accuracy indicates that the CNN effectively learned to recognize and classify the handwritten digits with minimal errors. The use of the Adam optimizer helped improve convergence speed and performance during training. Additionally, categorical cross-entropy was utilized as the loss function, which is suitable for multi-class classification problems, allowing the model to measure the distance between predicted probabilities and actual class labels effectively.

**Conclusion**

In conclusion, the **Handwritten Digit Recognition CNN** model demonstrates exceptional performance with a classification accuracy of **99.14%**. This result highlights the model’s ability to generalize well to unseen data, making it a reliable solution for digit recognition tasks. The architecture effectively captures relevant features through convolutional layers, while the dropout layer aids in preventing overfitting. Overall, this model not only validates the effectiveness of CNNs for image classification but also showcases their applicability in real-world scenarios, such as automated digit recognition in postal services and bank checks. Future improvements could involve experimenting with more complex architectures or transfer learning techniques to further enhance performance.

**GitHub Link:** [**https://github.com/Kashishvarmaa/DL-CS3232/blob/main/Lab\_3/mnist.ipynb**](https://github.com/Kashishvarmaa/DL-CS3232/blob/main/Lab_3/mnist.ipynb)