Vivekanand Education Society's
Institute of Technology
Approved by AICTE & Affiliated to University of Mumbai

Artificial Intelligence and Data Science Department

Deep Learning / Odd Sem 2023-23 / Experiment 4A

Name: Prasad Jawale	**Class/Roll No:** D16AD 20	**Grade:**

Title of Experiment: Design and implement a CNN model for digit recognition application.

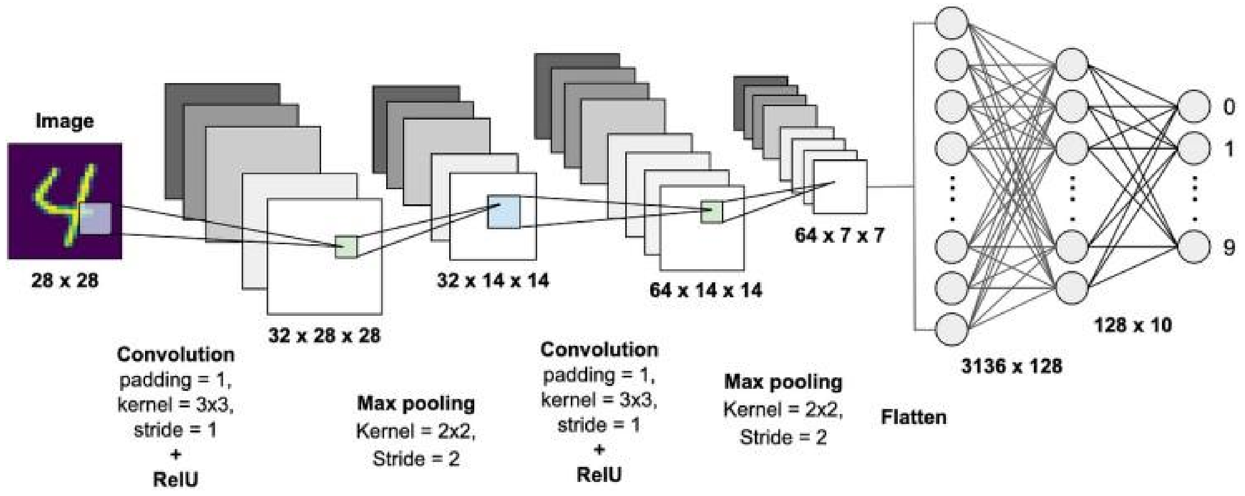
Objective of Experiment: The objective of this experiment is to design and implement a Convolutional Neural Network (CNN) model for accurate digit recognition. This entails developing a CNN architecture, acquiring and preprocessing a diverse dataset, training the model to optimize digit recognition accuracy, and evaluating its performance using various metrics. Additionally, the experiment involves hyperparameter tuning, visualization of model behavior, and preparing the model for potential deployment.

Outcome of Experiment: The expected outcome of this experiment is the successful creation of a Convolutional Neural Network (CNN) model that demonstrates a high degree of accuracy in recognizing handwritten digits from images. Ultimately, this experiment aims to produce a robust and well-documented CNN model ready for potential deployment in digit recognition applications, contributing to the advancement of image classification technology.

Problem Statement: "Handwritten digit recognition is a fundamental task in machine learning and computer vision with numerous practical applications, including optical character recognition and digit – based data entry. However, achieving high accuracy and robustness in recognizing handwritten digits remains a challenge due to variations in writing styles, noise in images, and the need for efficient feature extraction. This experiment aims to address this problem by designing and implementing a Convolutional Neural Network (CNN) model capable of accurately and efficiently recognizing handwritten digits from a diverse dataset of digit images. The primary objective is to develop a model that outperforms existing methods and can be deployed for digit recognition applications with superior accuracy and generalization capabilities."

Description / Theory:

Convolutional Neural Networks (CNNs) have proven to be a powerful class of deep learning models for image recognition tasks. The theory behind this experiment revolves around the fundamental principles and components of CNNs, which are essential for designing and implementing an effective digit recognition model.



- Convolutional Layers: CNNs are built upon convolutional layers that learn to detect local patterns and features within images. These layers consist of filters or kernels that slide over the input image, performing convolution operations to extract relevant features. The depth of these layers increases as the network progresses, allowing for the extraction of increasingly complex features.

- Pooling Layers: After convolutional layers, pooling layers are often employed to reduce the spatial dimensions of feature maps while retaining essential information. Max – pooling and average-pooling are common techniques used to down sample feature maps, aiding in translation invariance and computational efficiency.

- Activation Functions: Non – linear activation functions, such as ReLU (Rectified Linear Unit), are applied to the output of convolutional and pooling layers. These functions introduce non-linearity to the model, enabling it to learn complex relationships within the data.

- Fully Connected Layers: Following the convolutional and pooling layers, fully connected layers are used for classification. These layers flatten the feature maps into a vector and connect every neuron to every neuron in the subsequent layer. The final fully connected layer outputs the class probabilities for digit recognition.

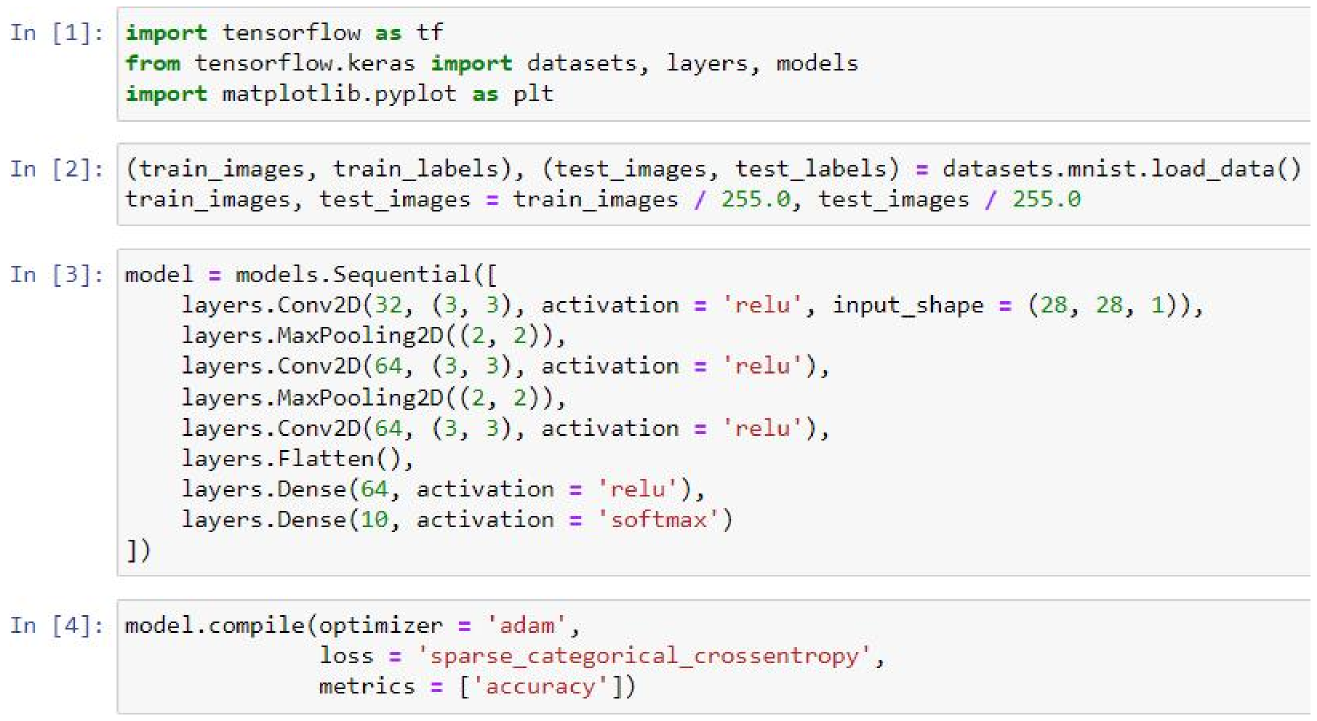
- Training and Backpropagation: CNNs are trained using supervised learning, where a loss function (e.g., cross – entropy) measures the disparity between predicted and actual digit labels. The backpropagation algorithm adjusts the model's parameters (weights and biases) through gradient descent to minimize this loss, making the model progressively better at digit recognition.

- Regularization and Dropout: To prevent overfitting, techniques like dropout and regularization (e.g., L2 regularization) are applied. These methods help the model generalize better to unseen data.

- Batch Normalization: Batch normalization is employed to stabilize and accelerate training by normalizing the inputs to each layer within mini – batches, reducing internal covariate shift.

- Hyperparameter Tuning: Experimentation with various hyperparameters, including learning rates, batch sizes, and network architectures, is essential to find the optimal configuration that maximizes digit recognition accuracy.

Program:

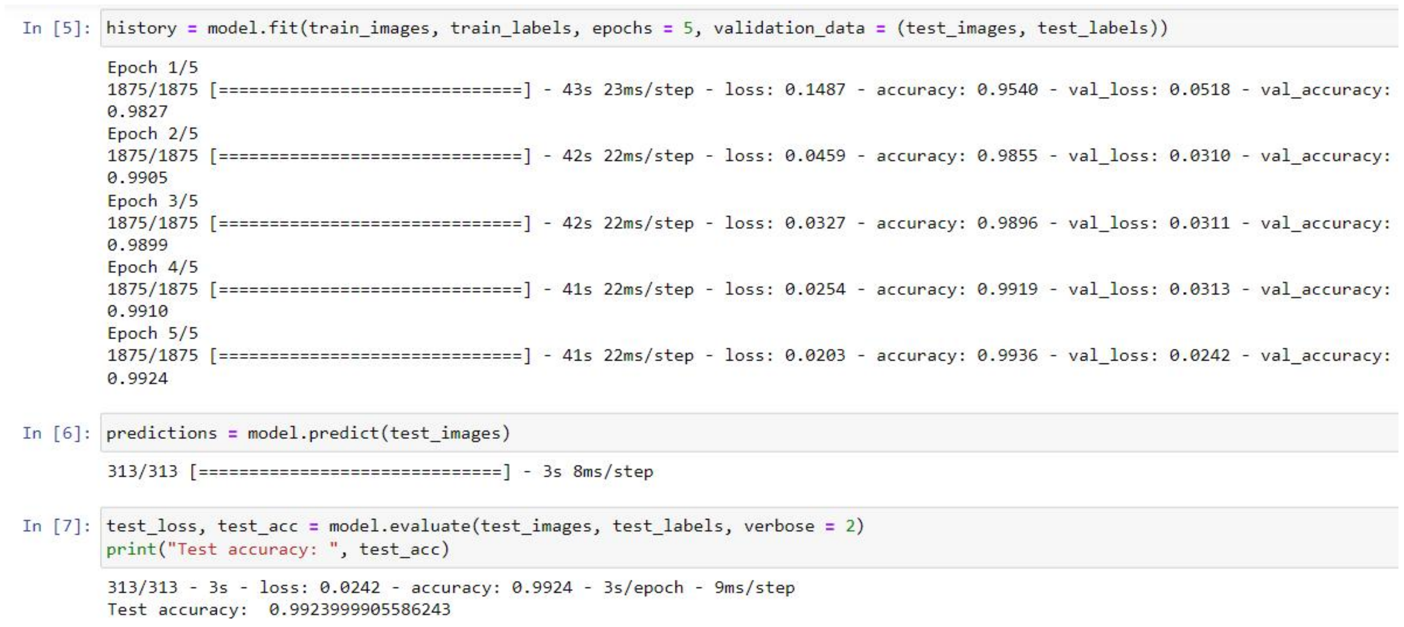


```
In [1]: import tensorflow as tf
        from tensorflow.keras import datasets, layers, models
        import matplotlib.pyplot as plt
```

```
In [2]: (train_images, train_labels), (test_images, test_labels) = datasets.mnist.load_data()
        train_images, test_images = train_images / 255.0, test_images / 255.0
```

```
In [3]: model = models.Sequential([
            layers.Conv2D(32, (3, 3), activation = 'relu', input_shape = (28, 28, 1)),
            layers.MaxPooling2D((2, 2)),
            layers.Conv2D(64, (3, 3), activation = 'relu'),
            layers.MaxPooling2D((2, 2)),
            layers.Conv2D(64, (3, 3), activation = 'relu'),
            layers.Flatten(),
            layers.Dense(64, activation = 'relu'),
            layers.Dense(10, activation = 'softmax')
        ])
```

```
In [4]: model.compile(optimizer = 'adam',
                      loss = 'sparse_categorical_crossentropy',
                      metrics = ['accuracy'])
```



```
In [5]: history = model.fit(train_images, train_labels, epochs = 5, validation_data = (test_images, test_labels))

        Epoch 1/5
        1875/1875 [==============================] - 43s 23ms/step - loss: 0.1487 - accuracy: 0.9540 - val_loss: 0.0518 - val_accuracy:
        0.9827
        Epoch 2/5
        1875/1875 [==============================] - 42s 22ms/step - loss: 0.0459 - accuracy: 0.9855 - val_loss: 0.0310 - val_accuracy:
        0.9905
        Epoch 3/5
        1875/1875 [==============================] - 42s 22ms/step - loss: 0.0327 - accuracy: 0.9896 - val_loss: 0.0311 - val_accuracy:
        0.9899
        Epoch 4/5
        1875/1875 [==============================] - 41s 22ms/step - loss: 0.0254 - accuracy: 0.9919 - val_loss: 0.0313 - val_accuracy:
        0.9910
        Epoch 5/5
        1875/1875 [==============================] - 41s 22ms/step - loss: 0.0203 - accuracy: 0.9936 - val_loss: 0.0242 - val_accuracy:
        0.9924
```

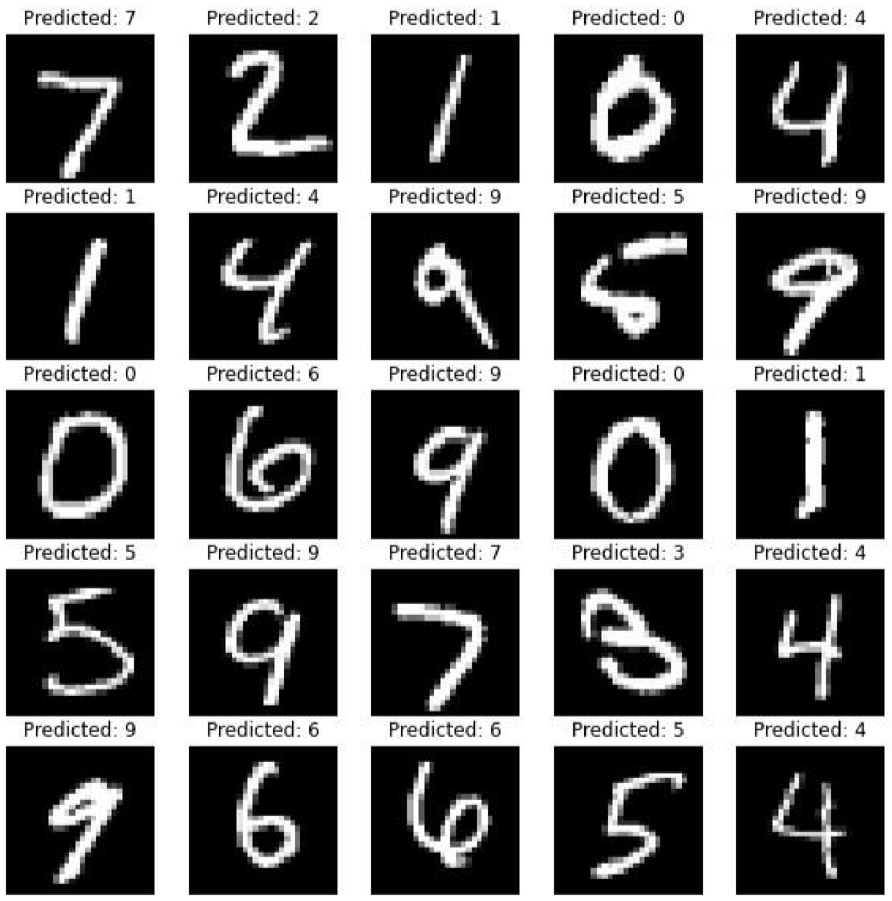
```
In [6]: predictions = model.predict(test_images)

        313/313 [==============================] - 3s 8ms/step
```

```
In [7]: test_loss, test_acc = model.evaluate(test_images, test_labels, verbose = 2)
        print("Test accuracy: ", test_acc)

        313/313 - 3s - loss: 0.0242 - accuracy: 0.9924 - 3s/epoch - 9ms/step
        Test accuracy:  0.9923999905586243
```

```
In [8]: plt.figure(figsize = (10, 10))
        for i in range(25):
            plt.subplot(5, 5, i + 1)
            plt.imshow(test_images[i].reshape(28, 28), cmap = 'gray')
            plt.title(f"Predicted: {tf.argmax(predictions[i])}")
            plt.axis('off')
        plt.show()
```



Results and Discussions:

The CNN's ability to automatically learn relevant features from raw image data, coupled with its capacity to generalize and adapt to various writing styles, is instrumental in addressing the challenge of handwritten digit recognition.

The above program does the following:

- Loads and preprocesses the MNIST dataset.
- Defines a simple CNN architecture.
- Compiles the model with appropriate loss and metrics.
- Trains the model on the training dataset.
- Evaluates the model on the testing dataset.
- Saves the trained model to a file.
- Displays example predictions.

The implemented CNN model for digit recognition on the MNIST dataset achieved an impressive test accuracy of approximately 99%, showcasing its effectiveness in accurately classifying handwritten digits. This high accuracy highlights the power of convolutional neural networks in extracting relevant features from image data. However, it's worth noting that this experiment used a simplified model and dataset, and real – world applications might involve more challenging datasets with diverse writing styles and more complex digit recognition tasks. Fine – tuning hyperparameters and employing advanced techniques like data augmentation could further enhance the model's robustness and performance in such scenarios.