Mohammad Idrees Bhat

Tech Skills Trainer | AI/ML Consultant

Handwritten Digit Classification

Building and evaluating a deep learning model for classifying handwritten digits using the MNIST dataset

PROJECT DESCRIPTION

Handwritten Digit Classification using MNIST Project Description: This project aims to guide students in building a neural network to classify handwritten digits (0-9) using the MNIST dataset. The dataset consists of 60,000 training images and 10,000 test images of grayscale handwritten digits. By the end of the project, students will have hands-on experience in data preprocessing, model creation, training, and evaluation while exploring strategies to enhance performance.

PROJECT DELIVERABLES

1. Trained Model: Saved CNN model file with architecture and weights (e.g., a .h5 file or TensorFlow model).
2. Jupyter notebook with code and documentation
3. Evaluation Report: Final accuracy, confusion matrix, and performance metrics with training insights.

FURTHER INSTRUCTIONS

1. Introduction

Familiarize yourself with the MNIST dataset and its significance in image classification tasks.

2. Data Preprocessing

- Load the MNIST dataset using TensorFlow or Keras.
- Normalize pixel values to a range between 0 and 1 for faster computation.

- Reshape the data to match the input requirements of a CNN.
- Split the dataset into training and testing sets.

3. Model Building

- Import libraries like TensorFlow and Keras.
- Build a Sequential CNN architecture with convolutional and pooling layers.
- Use ReLU as the activation function in hidden layers and Softmax for the output layer.

4. Model Training

- Compile the model using a suitable optimizer (e.g., Adam) and loss function (e.g., categorical cross-entropy).
- Train the model using the `fit` function and set the number of epochs and batch size.
- Monitor training and validation accuracy during the process.

5. Model Evaluation

- Evaluate the trained model on the test set to measure accuracy.
- Create a confusion matrix to visualize classification results.
- Plot accuracy and loss curves to analyze the training process.

6. Challenges and Enhancements

- Modify the CNN architecture to see its impact on performance.
- Experiment with optimizers (e.g., RMSprop, SGD) and learning rates.
- Implement dropout layers to reduce overfitting.
- Use data augmentation techniques to enhance model generalization.

7. Conclusion

- Reflect on the project results and document the learning experience.
- Suggest further improvements and discuss potential applications of the model.

BEST OF LUCK ... :)

THANK YOU!

Mohammad Idrees Bhat

Tech Skills Trainer | AI/ML Consultant