2024

HAND WRITTEN DIGIT PREDICTION

PRESENTED BY:
VAISHNAV KRISHNA P - 20221COM0040
MOHD TUFAIL - 20221COM0032
NAVNEETH PANDEY - 20221COM0009
SN DHANUSH - 20221COM0030

INSTRUCTOR:
MR.JAMIL AHMED

ABSTRACTION

This report explores the application of the Keras library in handwritten digit prediction through convolutional neural networks (CNNs). Leveraging Keras' intuitive interface and built-in optimization algorithms, deep learning models are trained on the MNIST dataset to recognize patterns and features in handwritten digits. The report highlights Keras' role in democratizing deep learning applications, particularly in computer vision tasks, by offering a user-friendly interface and powerful functionality for accurate digit classification. Through an analysis of model development and evaluation, this report showcases the effectiveness of Keras in facilitating handwritten digit prediction.

Table of CONTENTS

01 **Introduction**
Obtaining handwritten digit images from MNIST dataset

02 **Data collection**
Obtaining handwritten digit images from MNIST dataset

03 **Data Processing**
Preparing the cleaned data for analysis and modeling.

04 **Data Visualisation**
Matplotlib's imshow function facilitates the visualization of MNIST dataset images

05 **Neural Network Architecture**
The neural network model for MNIST digit classification consists of an input layer with 784 neurons, representing flattened images..........

06 **Model training**
Training convolutional neural networks on processed data.

07 **Model Evaluation**
Assessing the performance of trained models.

08 **Conclusion**
Summarizing findings and implications of the model's performance in handwritten digit prediction.

Introduction

The MNIST dataset comprises 28x28 pixel grayscale images of handwritten digits (0-9). It serves as a benchmark for machine learning tasks, with 60,000 training images and 10,000 test images. The objective is to accurately classify the digits, enabling foundational exploration in pattern recognition and deep learning algorithms.

Data collection

Keras, an open-source neural network library written in Python, provides convenient access to popular datasets, including the MNIST dataset, through its built-in datasets module. Users can easily load datasets into their Python environment using simple commands, eliminating the need for manual data collection and preprocessing.

Data Preprocessing

1.Flatten images from 28x28 to 1D arrays of 784 pixels, simplifying input.
2.Normalize pixel values to [0,1] range by dividing by 255.

3.Split dataset into training and testing sets for model evaluation.

Data Visualisation

Matplotlib's imshow function facilitates the visualization of MNIST dataset images. By reshaping flattened arrays into their original 28x28 grid format and applying the 'gray' colormap, grayscale images can be displayed. This simple code snippet enables easy inspection of individual digit images, aiding in data exploration and comprehension.

Neural Network Architecture

The neural network model for MNIST digit classification consists of an input layer with 784 neurons, representing flattened images. A hidden layer comprising 100 neurons utilizes the rectified linear unit (ReLU) activation function to introduce non-linearity. The output layer consists of 10 neurons, each representing a digit from 0 to 9, with a sigmoid activation function for classification probabilities.

Compiling the Model

- The model is compiled using the Adam optimizer, which is well- suited for training deep neural networks.
- Sparse categorical cross-entropy is chosen as the loss function for multi-class classification tasks like MNIST.
- Model performance is evaluated based on accuracy, a commonly used metric for classification tasks.

Model Training

The model is trained on the training dataset for 10 epochs, with a validation split of 0.1 (10% of the training data used for validation). This process allows monitoring the model's performance and adjusting hyperparameters accordingly

Model Evaluation

Utilizing a confusion matrix for model evaluation provides a comprehensive view of classification performance. It displays true positive, true negative, false positive, and false negative predictions, aiding in understanding model strengths and weaknesses across all classes. This visualization is essential for fine-tuning and improving model accuracy.

Conclusion

- Achieved an impressive training accuracy of 99.4% on the MNIST dataset.
- Demonstrates the model's ability to learn intricate patterns and relationships within the training data.
- Essential to validate generalization ability on a separate test dataset for real-world applicability.
- Further exploration for potential overfitting and fine-tuning of hyperparameters may enhance model reliability and effectiveness.