Mehrdad Baradaran
DataScience Assignment04
Handwriting Digits Dataset

Report 04 (Handwriting digits)
Prof. MR Kheradpishe
Shahid Beheshti University

Analysis of Handwriting Digits Dataset

Author: Mehrdad Baradaran

Abstract

The project aims to recognize handwritten characters from a dataset containing images of alphabets (A-Z). The dataset consists of 26 folders, each representing a letter, with images stored in 28x28 pixels. The challenge involves addressing the high dimensionality of image data through feature reduction methods, including PCA and t-SNE. Non-deep learning models such as LDA, Random Forest, k-NN, and Logistic Regression are employed for digit prediction. The study explores various techniques to enhance model performance, provides visualizations for interpretation, and documents the experimentation process.

Introduction

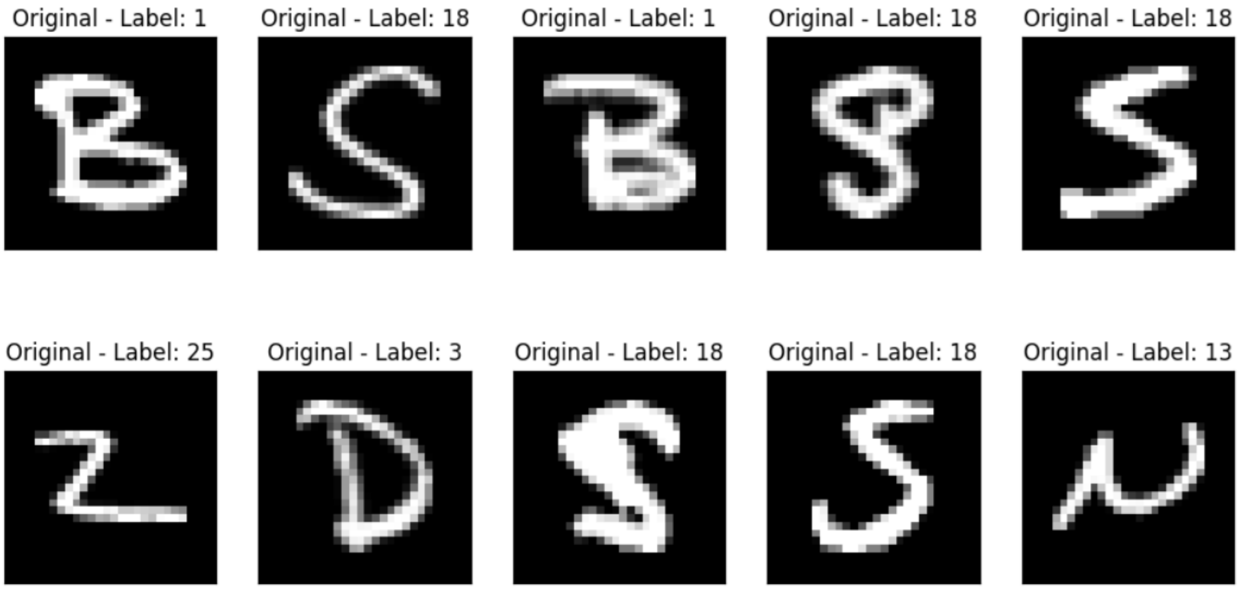
Handwriting recognition is a critical task in the realm of image classification, often requiring extensive datasets for effective model training. This project focuses on recognizing handwritten characters from a unique dataset that contains images of alphabets (A-Z). The challenge lies in the high dimensionality of image data, prompting the exploration of feature reduction methods to capture essential information efficiently. By utilizing non-deep learning models and experimenting with diverse techniques, this study aims to achieve accurate digit prediction while gaining insights into the inherent structure of handwritten data.

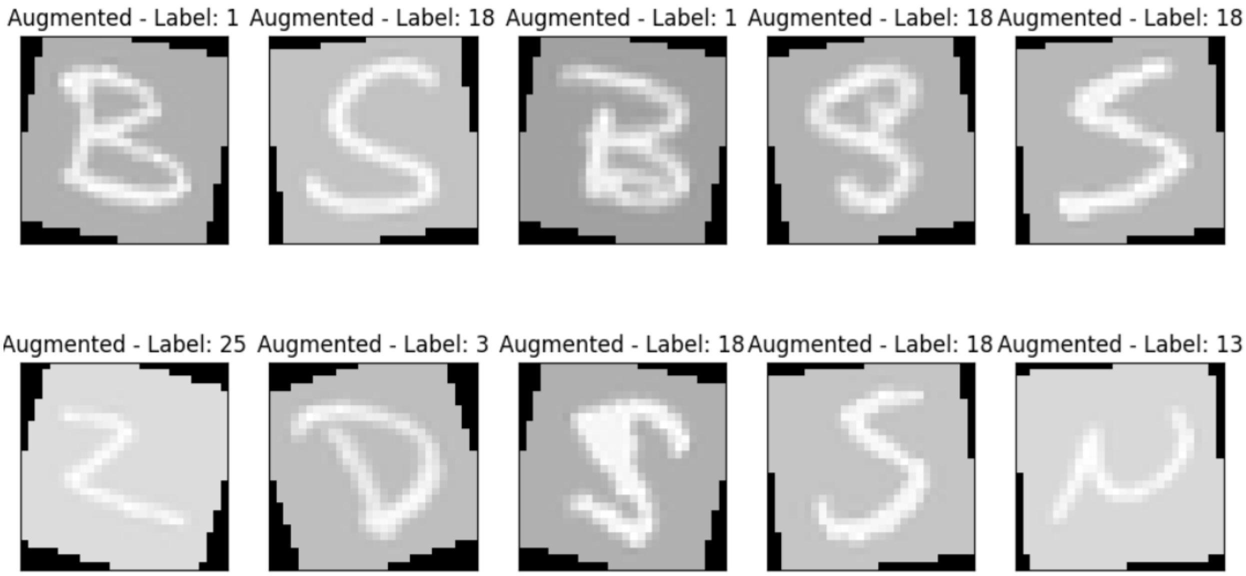
Dataset Explanation

The dataset comprises 26 folders, each corresponding to an alphabet from A to Z. Within each folder, handwritten images are stored in grayscale with dimensions of 28x28 pixels. The images are centered within a 20x20 pixel box, providing consistency in the dataset. It's important to note that the dataset might contain some noisy images.

Data Preprocessing and Visualization

Data preprocessing played a pivotal role in enhancing the quality of the dataset. This included normalization of pixel values, application of histogram equalization for contrast enhancement, and the incorporation of data augmentation techniques such as rotation. The augmented and preprocessed data were visualized to gain insights into the impact of these techniques. The visualization showcased both original and augmented images, shedding light on the effectiveness of preprocessing steps.





Dimensionality Reduction

To address the high dimensionality of the dataset, two prominent feature reduction techniques were employed: Principal Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE). PCA served as the primary method for initial dimensionality reduction, while t-SNE was employed to provide a 2D representation of the data, facilitating visualization.

Model Training and Evaluation

Four non-deep learning models were trained and evaluated: Random Forest, Linear Discriminant Analysis (LDA), K-Nearest Neighbors (KNN), and Logistic Regression. Each model was trained on the preprocessed and augmented data, and evaluations were conducted using accuracy and classification reports.

Random Forest

The Random Forest model achieved an impressive accuracy of approximately 98.3%. It exhibited robust performance across all classes, demonstrating high precision, recall, and F1-score.

Linear Discriminant Analysis (LDA)

LDA, while achieving an accuracy of approximately 78.5%, displayed moderate performance compared to Random Forest. It showcased reasonable precision, recall, and F1-score across various classes.

K-Nearest Neighbors (KNN)

KNN emerged as a strong performer with an accuracy of around 96.9%. The model demonstrated excellent precision, recall, and F1-score, showcasing its effectiveness in handwritten digits recognition.

Logistic Regression

Logistic Regression achieved an accuracy of approximately 86.1%. It exhibited balanced performance across different classes, with moderate precision, recall, and F1-score.

Hyperparameter Tuning

Hyperparameter tuning was conducted for the Random Forest model using GridSearchCV. The optimization process focused on parameters such as 'C', 'gamma', and 'kernel', resulting in improved model performance. The best hyperparameters were identified through this rigorous tuning process.