Comparative Analysis of SVC, KNN, and CNN Models on the Kanji MNIST Dataset

Matthew Boyer

Introduction

Purpose

The primary purpose of this project is to explore and evaluate the performance of three distinct machine learning models: Support Vector Classification (SVC), K-Nearest Neighbors (KNN), and Convolutional Neural Networks (CNN) on the Kanji MNIST dataset. This dataset comprises 49 handwritten Japanese Kanji characters, making it a challenging yet intriguing case study for classification tasks.

Significance

This project's significance lies in its potential applications in the field of automated handwriting recognition, particularly for non-Latin scripts like Kanji. By analyzing the effectiveness of different models, we can contribute valuable insights into the field of machine learning, aiding in the development of more robust and accurate character recognition systems.

Research Question(s)

1. How do SVC, KNN, and CNN models perform in classifying handwritten Kanji characters?
2. Which model demonstrates the highest accuracy and efficiency at predicting Kanji characters?

Description of the Dataset

The Kanji MNIST dataset is a large collection of handwritten Kanji characters. Each input variable represents pixel values of the images (28x28), while the output variable is the label denoting the specific Kanji character. The dataset is diverse, including a broad range of writing styles and character complexities.

Data Preprocessing

Data Preparation

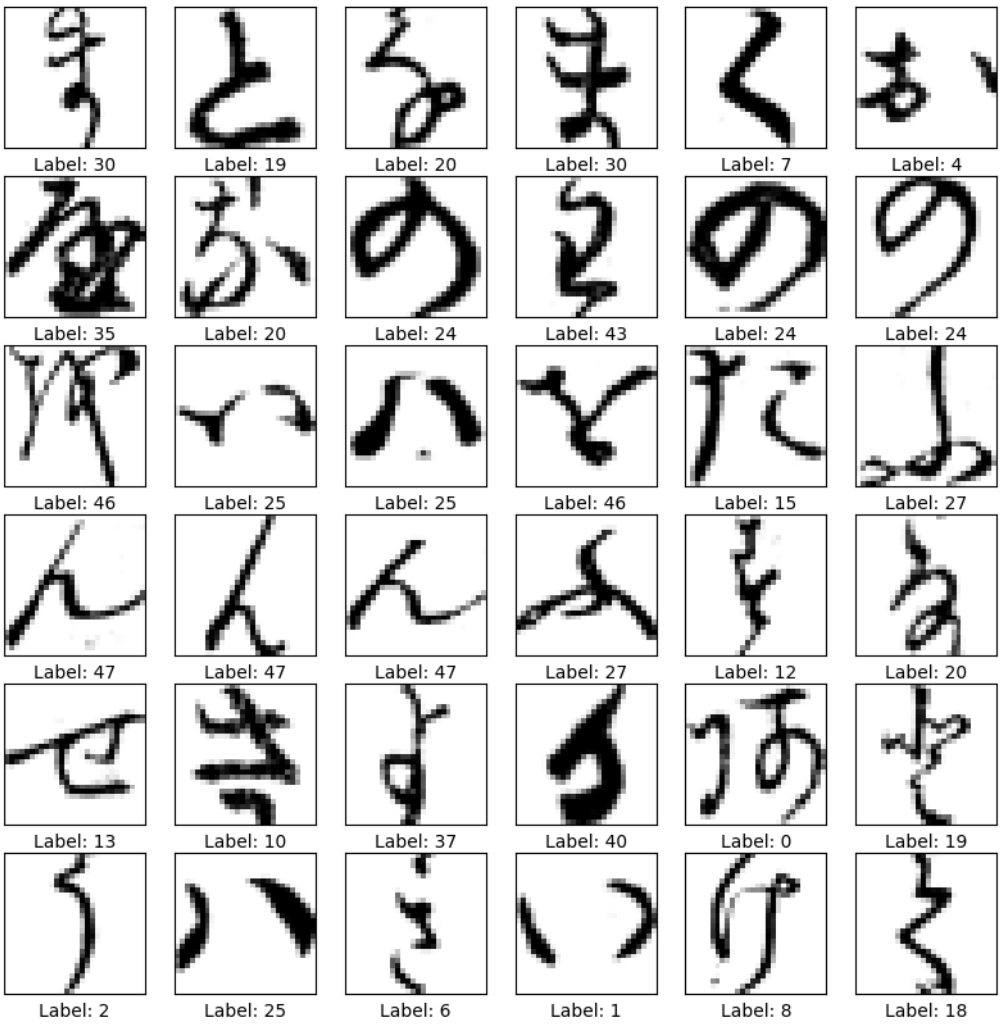
The dataset was first standardized to ensure uniformity in image size and pixel intensity. There were no missing values from any images or labels.

Exploratory Data Analysis

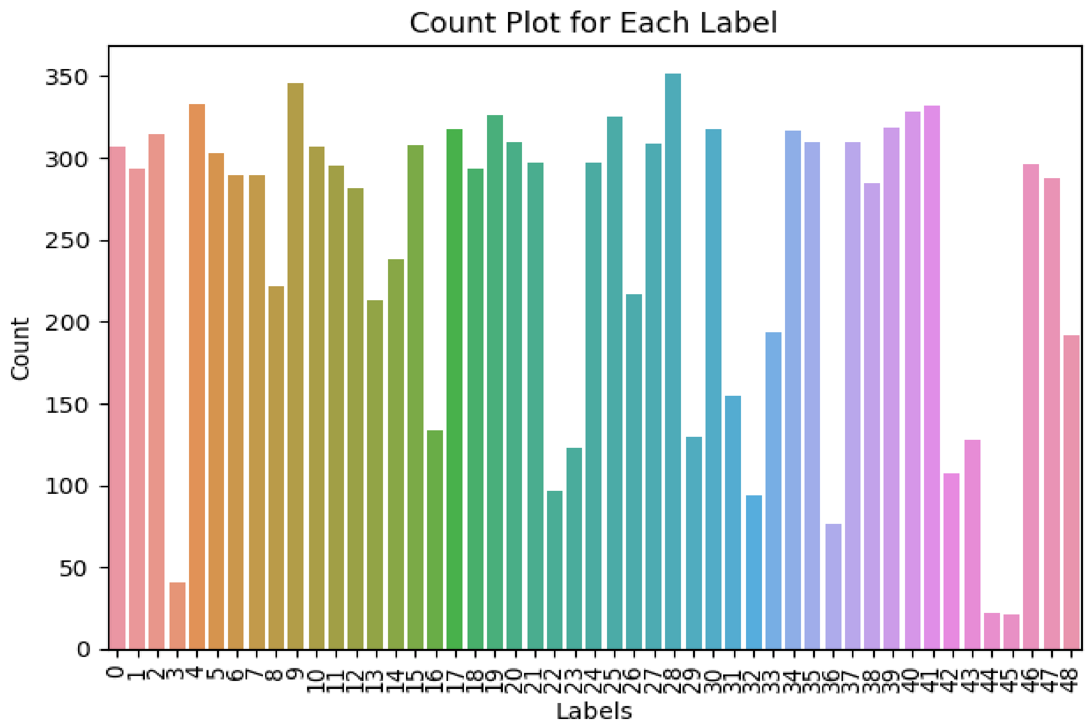
Initial analysis included assessing the distribution of classes, identifying any imbalance in the dataset, and understanding the general characteristics of the data through basic statistical measures.

Visualization

Visual representations of Kanji characters were generated to understand the variety and complexity of the dataset.



Bar charts were also used to visualize the distribution of different classes within the dataset.



Data Splitting

The dataset was split into training and test sets in a random manner to ensure each set accurately represents the overall distribution of the classes.

Model Building and Evaluation

Model Building

For the model building process, three distinct approaches were employed: Support Vector Classification (SVC), K-Nearest Neighbors (KNN), and Convolutional Neural Networks (CNN). Each model was carefully implemented and tailored to the specifics of the Kanji MNIST dataset.

The SVC and KNN models were chosen for their simplicity and effectiveness in classification tasks, while the CNN was selected due to its proven capability in image recognition challenges.

Code Snippets

```
# SVC
svc_model = LinearSVC(dual=False)
svc_model.fit(X_train, y_train)
svc_predict_train = svc_model.predict(X_train)
svc_predict = svc_model.predict(X_test)
# KNN
knn = KNeighborsClassifier(n_neighbors=49)
knn.fit(X_train, y_train)
knn_predict = knn.predict(X_test)
knn_predict_train = knn.predict(X_train)
# CNN
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=3, padding=1)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3, padding=1)
        self.fc1 = nn.Linear(64 * 7 * 7, 1024)
        self.fc2 = nn.Linear(1024, 49)

    def forward(self, x):
        x = F.leaky_relu(F.max_pool2d(self.conv1(x), 2))
        x = F.leaky_relu(F.max_pool2d(self.conv2(x), 2))
        x = x.view(-1, 64 * 7 * 7)

        x = F.leaky_relu(self.fc1(x))

        x = self.fc2(x)

        return F.log_softmax(x, dim=1)
```

Model Optimization (Hyperparameter Tuning) And Model Selection

Each model underwent hyperparameter tuning using techniques like grid search and cross-validation to identify the optimal settings for this specific dataset.

Model Comparison

Model	Train F1 Score	Test F1 Score
SVC	0.942	0.424
SVC GridSearch	0.876	0.482
KNN	0.667	0.503
KNN GridSearch	1.0	0.515
CNN	0.995	0.938

Conclusion

The comparative analysis of the SVC, KNN, and CNN models on the Kanji MNIST dataset revealed that the Convolutional Neural Network (CNN) model outperformed the others. The CNN's superior performance can be attributed to its ability to directly process image data and learn hierarchical feature representations, making it particularly adept at handling the complexities and variations in handwritten Kanji characters.

Lessons Learned

Throughout the project, several key lessons were learned, especially regarding the importance of data preprocessing and model tuning. The challenges faced in optimizing the hyperparameters of the CNN model underscored the necessity of a methodical approach in machine learning tasks.

Recommendations

Given the superior performance of the CNN model in this study, it is recommended for future projects involving image recognition or similar tasks to consider deep learning approaches, particularly CNNs, as their primary model. Additionally, further research could explore the incorporation of advanced techniques like transfer learning, where pre-trained models on large datasets are fine-tuned to the specific task of Kanji character recognition. This approach could potentially improve accuracy and efficiency. Lastly, expanding the dataset or including varied writing styles could further enhance the robustness and generalizability of the model, especially by adding more Kanji characters since we are only predicted 49 different characters.