CSCI 367 Final project.

Saul E. McCoy
Department of Computer Science
Queens College
saul.mccoy11@qmail.cuny.edu

Abstract

For this paper I want to present an algorithm that would recognize images of handwritten digits without the use of Convolutional Neural Networks (CNNs). CNNs is also applicable for object detection, image segmentation and video analysis, this system leverages traditional machine learning algorithms and feature extraction techniques to classify handwritten digits based on the dataset that you would give the system. However with applications that are in various domains that use different algorithms such as Multi-Layer Perceptron(MLP), Support Vector Machines (SVM), and K-Nearest Neighbors(KNN) can also accurately recognize images of handwritten digits in conjunction with feature extraction methods tailored for the task we want to complete, and it this context it is character recognition. With project I want to prove that even though CNN is not allowed we can still find suable accuracy results we looking through the dataset that was given to us.

1 Introduction

With the study of image recognition, CNNs have become very popular for having different types of models that are well-suited for image recognition and processing tasks which for this case is handwriting recognition which is often given good results when testing for accuracy. This deep learning model is the standard for machine learning that offer valuable insights on the task that you would want to train your model for, especially when a deeper understanding of feature[2]. However there are other image classification models that can be performed through traditional machine learning methods which would require manual classification of the features which I will discus in more detail at a later time.

With this paper I want to go over three models in detail and how they differ from each other and then implement a model that I believe would give me the best accuracy result when testing and training the data that was given to me. There is also the challenges of achieving high accuracy and reliability in handwritten digit recognition due to the variety of human handwriting styles that I would have to classify and train the model to understand what each letter should be based on a general shape that each printed letter should have regardless of the font type.

Overall what this paper is going to convey is that for designing and implementing a handwritten digit recognition system the most popular option would be CNN, because with deep learning the feature extraction is automated. But with the other models it allows us to have a better understanding of how traditional machine learning works, the methodology adopted, the experimental results obtained and the final results along with concluding remarks regarding what I've learned and problems I ran into along the way.

Preprint. Under review.

2 Brief Model Overview

The first model I would like to discus is a **Multilayer Perceptron (MLP)**, an MLP is a type of artificial neural network consisting of multiple layers of neurons. The neurons in the MLP typically use nonlinear activation functions, allowing the network to learn complex patterns in data. MLPs are significant in machine learning because they can learn nonlinear relationships in data, making them powerful models for tasks such as classification, regression, and pattern recognition[4]. An MLP is a type of feedforward neural network consisting of fully connected neurons with a nonlinear kind of activation function. It is widely used to distinguish data that is not linearly separable which is the type of data I was given to work with. MLPs are considered deep neural networks and have previously successfully applied to image classification problems, with one study reported an overall accuracy of 90.37% [1].

The next model I will talk about is **K-Nearest Neighbors(KNN)**. The k-nearest neighbors (KNN) algorithm is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. It is one of the popular and simplest classification and regression classifiers used in machine learning today. While the KNN algorithm can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another[5]. For image classification, KNN can be applied by representing each image as a feature vector, often using pixel values directly or after some dimensionality reduction. Studies have shown KNN achieving competitive results on handwritten digit recognition, with error rates as low as 1.58% reported on the MNIST dataset using structural and statistical features[7]. Similar to the model above this model is nonlinear which is why this is another valuable model to select for testing the accuracy of the dataset, but the next model I will talk about is the one that I enviably selected to use as the actual model.

The final model I will discuss is **Support Vector Machines (SVMs)**. SVM works by mapping data to a high-dimensional feature space so that data points can be categorized, even when the data are not otherwise linearly separable. A separator between the categories is found, then the data are transformed in such a way that the separator could be drawn as a hyperplane. Following this, characteristics of new data can be used to predict the group to which a new record should belong[8].Traditionally, SVMs rely on manual feature extraction to transform the input images into feature vectors suitable for classification. While more recent research has explored combining SVMs with CNNs for even higher accuracy , SVMs have also demonstrated strong performance when used with handcrafted features, achieving error rates as low as 1.24% on the MNIST dataset with appropriate feature engineering[1].

In summary, I wanted to introduce some algorithms that I believe are viable using a traditional machine learning method that would give a good accuracy result, and have also been successfully applied to handwritten digit recognition. This paper will dive into more detail about SVM models and how I implemented feature extraction techniques used to represent the digit images and aiming to achieve an optimal recognition performance without the used of CNNs.

3 Support Vector Machine (Methodology)

The main reason behind the selection of SVMs is the proven effectiveness in image classification tasks, its strong generalization capabilities, and its ability to handle high-dimensional data, which is often the case when dealing with image features. While SVMs can be used for both linear and non-linear classification similar to kNN, I wanted to explore the use of the non-linear kernel, specifically the Radial Basis Function (RBF) kernel [9], to account for the potentially complex and non-linearly separable nature of handwritten digit variations.In SVM, the kernel function plays a vital role in the classification of data. The kernel function maps the original input data into a higher dimensional feature space where it becomes easier to classify the data into different classes using a linear decision boundary[9,15], I will go into more detail about this later. For feature extraction, we will utilize the Histogram of Oriented Gradients (HOG). HOG is a relatively widely used feature descriptor in computer vision for object detection and image classification, I will dive into farther detail of how HOG works and how it is integrated with SVM at a later time.

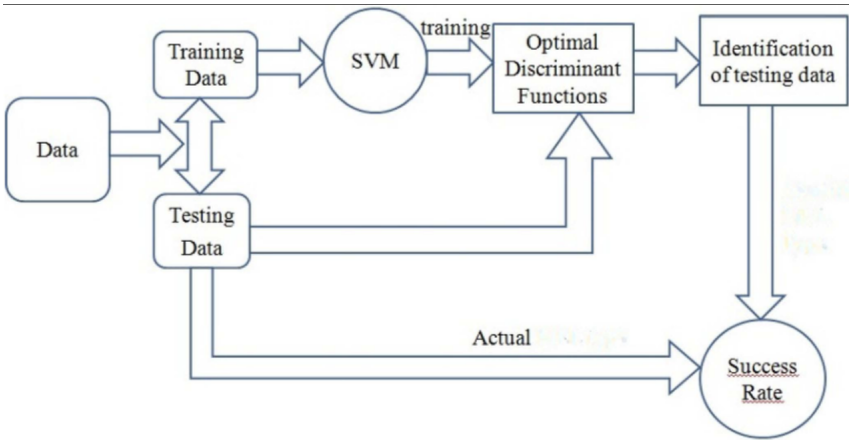

Figure 1: SMV framework

3.1 Radial Basis Function kernal

Radial Basis Function Kernel is a very powerful kernel used in SVM. Unlike linear or polynomial kernels, RBF is more complex and efficient at the same time that it can combine multiple polynomial kernels multiple times of different degrees to project the non-linearly separable data into higher dimensional space so that it can be separable using a hyperplane.The RBF kernel works by mapping the data into a high-dimensional space by finding the dot products and squares of all the features in the dataset and then performing the classification using the basic idea of Linear SVM. For projecting the data into a higher dimensional space, the RBF kernel uses the so-called radial basis function which can be written as: $K(X_1, X_2) = exp(\frac{||X_1 - X_2||^2}{2\sigma^2})$
$K(x, xi) = exp(-\gamma * ||x - x_i||^2)$[14,15] where x and xi are input vectors, ||x - xi||2 is the squared Euclidean distance between the two vectors, and gamma is the kernel parameter[10,15]. It's important to note that while I was testing he RBF kernel function can be sensitive to the choice of gamma. If gamma is too small, the model may underfit the data, while if gamma is too large, the model may overfit the data.

3.2 Histogram of Oriented Gradients

Histogram of Oriented Gradients (HOG) features were first introduced by Dalal and Triggs in 2005 as a robust feature extraction method for pedestrian detection. The core idea behind HOG is to capture the distribution of gradient orientations in an image, which can be used to describe the shape and appearance of objects. HOG features are computed by dividing an image into small cells, calculating the gradient orientations within each cell, and then aggregating these orientations into a histogram. This histogram represents the distribution of gradient orientations, which can be used as a feature vector for object detection[11]. Some HOG features that persuaded me into selecting this type of model for implementing this dataset was the feature to handle partial images with it still able to provide accurate object detection even when objects are partially hidden, and the flexibility to be used with various classification algorithms like random Random Forests and of course SVM just to name some of the few algorithms it can integrate with. Other features are robustness to lighting changes and and computational efficiency for real-time object detection, this doesn't really help in the context of what I am looking for but I still feel like this is a nice feature to not for future reference. In conclusion HOG feature offer a powerful tool for object detection that provides a robust and efficient way to represent images. By visualizing HOG features using Python and skimage, we can gain a deeper understanding of how these features capture the essence of an image, enabling accurate object detection in various scenarios[11,12]. Even though in this context we are just looking for the accuracy of the number this feature can be used for facial recognition, or object tracking, and because of this I believe HOG features are an essential component of any computer vision pipeline.

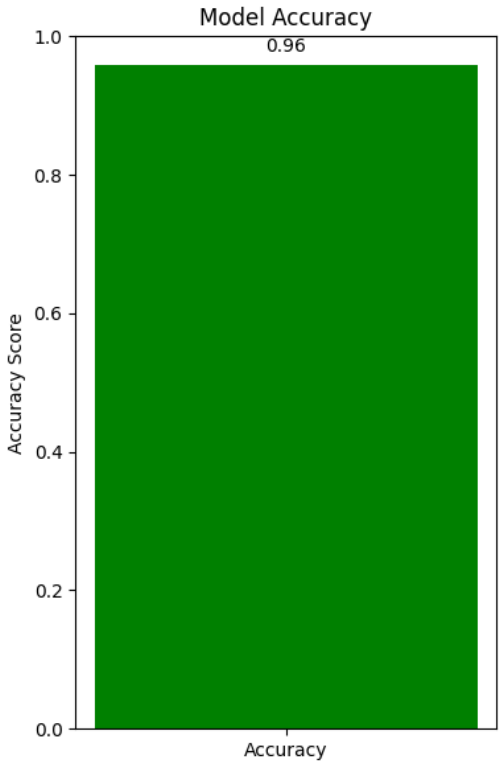

Figure 2: Model accuracy during training and evaluation.

4 Implementation

To piece together how everything in the end worked together with the primary goal was to build a machine learning pipeline to classify images of handwritten digits (0-9) using the Histogram of Oriented Gradients (HOG) for feature extraction and a Support Vector Machine (SVM) with an RBF kernel for classification. `Scikit-learn` is one of the core libraries used for implementing SVM; Support Vector Classifier (SVC) was used for implementing the SVM classifier along with the accuracy score, classification, and confusion matrix. **Scikit-image** is used for HOG and as talked about before it extracts features from the digit images. For the feature extraction to properly use HOG I had to reshape each 256-pixel array is reshaped into a 16x16 2D image, **pixels_per_cell=(4, 4):**The 16x16 image is divided into 4x4 cells. For RBF kernel within the SVM model I had to select **kernel='rbf'** allowing the model to learn non-linear decision boundaries and **gamma='scale'** is used, which is a common and often effective default for the RBF kernel's gamma parameter. The **C** regularization parameter uses its default value (typically 1.0). Finally for the training model I used the **svm_model.fit(X_train, y_train)** command which trains the SVM classifier using the HOG features of the training images and their corresponding true labels.

4.1 Confusion Matrix

With the accuracy above showing 96% the confusion matrix shows which numbers numbers are properly mapped to the actual number compared to the predicted number. Along with the incorrect mapped number I with this model I can show that for the most part this performs very well across all of the digit categories.

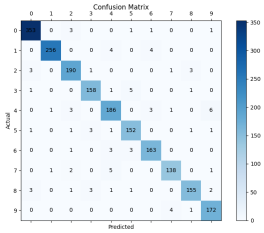

Figure 3: Confusion matrix showing classification performance across classes.

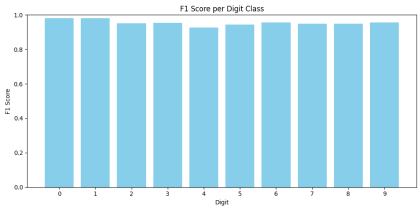

Figure 4: F1 scores for each digit class.

4.2 F1 Scores

The F1 score is a performance metric used to evaluate classification models especially when classes are imbalanced or when both precision and recall matter. The reason I want to show this is because even though I have a high accuracy rating the model is still struggling with classifying the numbers 4 and 9 the most. It shows the strength of the model but is also shows how some work can be done with identifying certain types of numbers.

5 Conclusion & Final Result

In conclusion, this project successfully implemented a handwritten digit recognition system using a traditional machine learning approach, specifically SVM with HOG features, achieving an overall accuracy of roughly 96% on the provided dataset. This demonstrates the continued viability of non-CNN techniques for this task and underscores the importance of careful algorithm selection and feature engineering. Future work could focus on further optimizing the HOG parameters for a deeper understanding of the topic and learn what other models can be implemented with this feature, experimenting with other feature extraction methods like Local Binary Patterns (LBP), or exploring different classification algorithms such as Multi-Layer Perceptron (MLP) or K-Nearest Neighbors (KNN) to potentially achieve even higher recognition rates.

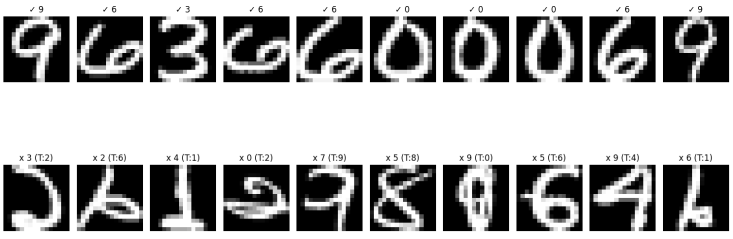

Figure 5: A final output for correct and incorrect mappings

References

[1] JATIT. (2020). "Image classification using support vector machines." https://www.jatit.org/volumes/Vol102No5/37Vol102No5.pdf

[2] DigitalOcean. "Image Classification Without Neural Networks." https://www.digitalocean.com/community/tutorials/image-classification-without-neural-networks

[3] GeeksforGeeks. "Convolutional Neural Network (CNN) in Machine Learning." https://www.geeksforgeeks.org/convolutional-neural-network-cnn-in-machine-learning/

[4] DataCamp. "Multilayer Perceptrons in Machine Learning." https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning

[5] IBM. "What is KNN?" https://www.ibm.com/think/topics/knn

[6] GeeksforGeeks. "Support Vector Machine Algorithm." https://www.geeksforgeeks.org/support-vector-machine-algorithm/

[7] Arize AI. "KNN Algorithm: K-Nearest Neighbor Explained." https://arize.com/blog-course/knn-algorithm-k-nearest-neighbor/

[8] IBM. "How SVM Works." https://www.ibm.com/docs/en/spss-modeler/saas?topic=models-how-svm-works

[9] GeeksforGeeks. "Linear vs Non-Linear Classification Using the Kernel Trick." https://www.geeksforgeeks.org/linear-vs-non-linear-classification-analyzing-differences-using-the-kernel-trick/

[10] Scikit-learn. "SVM Documentation." https://scikit-learn.org/stable/modules/svm.html

[11] GeeksforGeeks. "HOG Feature Visualization in Python Using skimage." https://www.geeksforgeeks.org/hog-feature-visualization-in-python-using-skimage/

[12] PyImageSearch. "Histogram of Oriented Gradients and Car Logo Recognition." https://customers.pyimagesearch.com/lesson-sample-histogram-of-oriented-gradients-and-car-logo-recognition/

[13] ResearchGate. "The multi-class one-against-all SVM classifier framework." https://www.researchgate.net/figure/The-multi-class-one-against-all-SVM-classifier-framework_fig2_261309488

[14] QuarkML. "The RBF Kernel in SVM: A Complete Guide." https://www.quarkml.com/2022/10/the-rbf-kernel-in-svm-complete-guide.html

[15] GeeksforGeeks. "RBF SVM Parameters in Scikit-learn." https://www.geeksforgeeks.org/rbf-svm-parameters-in-scikit-learn/