Classification of MNIST Dataset using Feature Extraction and kNN

Blake Diamond
Electrical and Computer Engineering
UC Santa Barbara
Goleta, California, United States
bdiamond@engineering.ucsb.edu

Himangshu Chowdhury
Electrical and Computer Engineering
UC Santa Barbara
Goleta, California, United States
himangshuchowdhury@ucsb.edu

***Abstract*—This document details the process of creating an image classifier specifically for the well-known MNIST dataset classifying a handwritten image into one of the digits from 0 to 9. In order to achieve our goal, we implemented data visualization and preprocessing for feature extraction and proved its incapability before performing a k-nearest neighbors algorithm.**

I. Introduction

The task being performed in this challenge is to create an image classifier that is able to classify different handwritten digits in the MNIST dataset ranging from 0 to 9. The implementation required and preprocessing of the raw image data for extracting temporal and spectral features from the input signal, which will be detailed in the next section. The approach then summarizes the features, which are utilized to compare different digits using data visualization, and then proves the inability of these features to distinguish between digits, which leads to using the k-nearest neighbors (kNN) method used on the raw data for the classification.

II. Methods

A. Pre-Processing

The training and test sets, along with their respective labels were loaded into our program using helper functions from an outside source. In previous assignments, we have taken use of an overlapping block processing scheme in order to perform signal processing; in this assignment, we are largely choosing to perform pixel-based processing methods, which requires relatively little pre-processing. Many of the features that were extracted in the following section are based on the spectrum of the image, which is obtained utilizing a two-dimensional fast Fourier algorithm.

B. Feature Extraction and Data Visualization

Extending the work from the last assignment, many of the features that were extracted are fundamentally the same, but instead applied to images rather than audio signals. Previously, we had used both time based and frequency based features in order to characterize signals, but due to the nature of digital images, we are mostly leveraging the spectrum of the images. The only non-spectrum-based features that we are collecting are the mean and standard deviation of the image pixels. Using histograms in order to visualize this data, seen in figure 2 below, these two summary features describe the ratio of black pixels to white pixels in the context of the MNIST dataset; however, this information does not capture in any way the spatial distribution of the image.

The frequency spectrum of the image perhaps contains the most information that can be used to extract meaningful features of the images. Multiple features were calculated in order to provide a holistic feature summary of a particular image. First, energy and the logarithm of energy were computed, as described in equation 1 and equation 2, respectively. This feature describes the total energy of the image, which highlights the content of black or white pixels in the image.

$$E[i,j] = \frac{1}{N}\sum_{n} x[i,j]^2 \tag{1}$$

$$LE[i,j] = \frac{1}{N}\sum_{n} \log\left(x[i,j]^2\right) \tag{2}$$

Cepstral coefficients, described by equation 3, were calculated because they provide a useful way to capture frequency domain information and summarize it in a way that is palatable to our kNN algorithm. Spectral Centroid was also calculated in order to characterize the result of the spectrum obtained in pre-processing; it provides for a meaningful way to locate the principle component of the spectrum. Spectral flatness and spectral crest, which is described in equation 4, were also computed because they measure how noise-like and the ratio of the peak values to nominal values of the signal, respectively.

$$c[k] = DFT^{-1}\log(|STFT\{x[i,j]\}|) \tag{3}$$

$$C_{dB} = 20 \times 10y_{10}\left(\frac{|x_{peak}|}{x_{rms}}\right) \tag{4}$$

C. K-Nearest Neighbors

To classify the input sound signal as one of the five environmental sounds, we used a machine learning approach, where we divided each of the sounds' files into 3 sets of files – training, validation and testing. We checked the k-nearest neighbors to each of the validation files and assigned a label to it based on a voting system; the labels being 0-4 based on which of the environmental sounds it corresponded to. To implement this algorithm, we took each validation file and found its distance from each of the training files that we had gathered using different methods discussed below:

a) Euclidian: Calculates the distance by evaluating the RMS difference between each element of two vectors.

$$d_{st} = \sqrt{(x_s - x_t)(x_s - x_t)'}$$

b) Squared Euclidian: Calculates the distance by squaring the RMS difference between each element of two vectors.

$$d_{st} = (x_s - x_t)(x_s - x_t)'$$

c) Manhattan: Calculates the distance by evaluating the sum of the difference between each element of two vectors.

$$d_{st} = \sum_{j=1}^{n} |x_{sj} - x_{tj}|$$

d) Minkowski: Calculates the distance by evaluating the p^{th} root of the (difference between each element of two vectors) raised to the power of p.

$$d_{st} = \sqrt[p]{\sum_{j=1}^{n} |x_{sj} - x_{tj}|^p}$$

e) Mahalanobis: Calculates the distance by evaluating the correlation between the difference between each element of two vectors and the correlation matrix.

$$d_{st} = \sqrt{(x_s - x_t)C^{-1}(x_s - x_t)'}$$

After finding the distances and storing them in an array along with their respective indices, we sorted them ascendingly. Depending on the value of k, the first k elements in the array are used their respective indices to determine the k-nearest neighbors. Using a majority voting system, we would then classify the validation file, assigning it a label. After measuring the accuracy of the algorithm by testing all the validation files and checking the predicted labels against their assigned labels, we implemented a looping algorithm that determined the best possible value of k that would yield the highest accuracy. Using this value of k for our test files, we then confirmed the validity of our algorithm.

III. RESULTS

This section explains how feature extraction was unfeasible for image classification. This section also compares the accuracies of the kNN algorithm with and without feature extraction.

Data visualization was an important tool in realizing that feature extraction for the MNIST dataset was not a "good" input for the kNN algorithm. Summarizing the features for different digits furthered this claim.

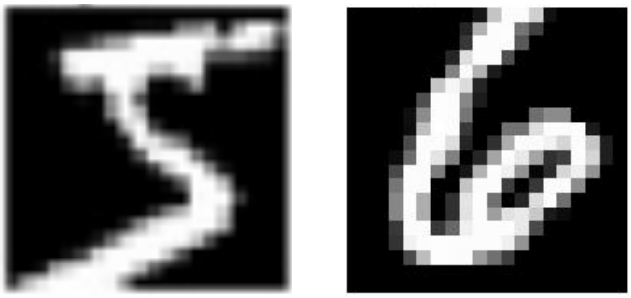

Figure 1: Test Images From Dataset

Two sample cases representing '5' and '6', shown in Figure 1, were chosen and the similarities and differences were computed using feature extraction as well as data visualization. The image histogram of both the images shown in Figure 2 reveals the grave similarities in the amount of whitespace in each image. Summarizing the amount of whitespace by taking a mean of the image histogram coefficients proved that to be the case as the mean for '5' was 0.2579, while for '6' it was 0.2672. The standard deviations for the same were 0.3765 and 0.3659 respectively.

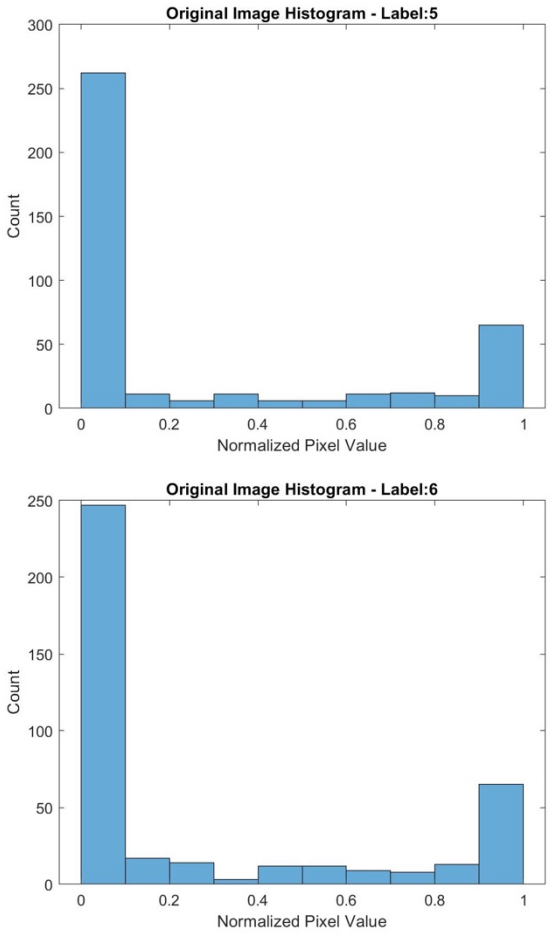

Figure 2: Coefficient Histograms of Test Images

The spectral magnitudes of the two different digits, shown in Figure 3, and spectral energy of the digits, shown

in Figure 4, did however have slight differences when visualized. Nevertheless, the summarization of these features resulted in yielding means 0.2562 and 0.2662, and standard deviations 0.3897 and 0.3887 for spectral magnitude, while giving means 0.2059 and 0.2337, and standard deviations 0.2769 and 0.2945 for spectral energy respectively for the two digits.

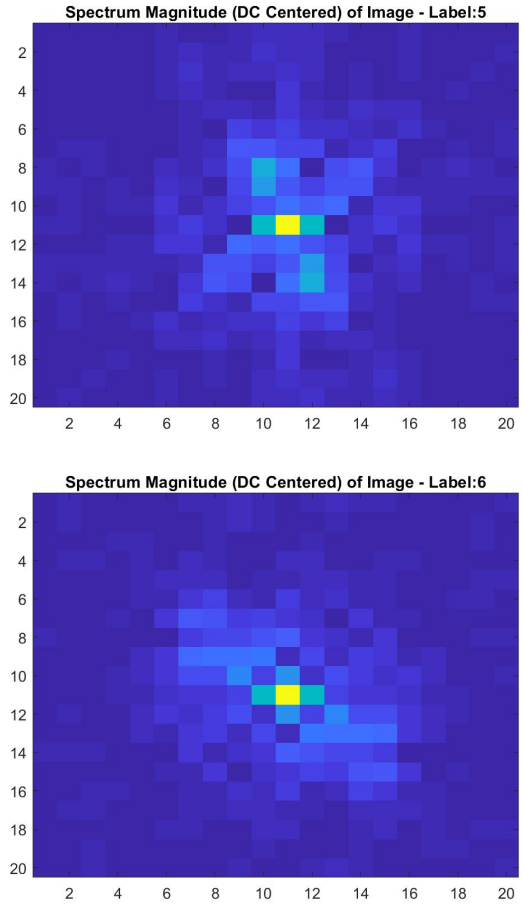

Figure 3: Spectral Magnitude Plots of Test Images

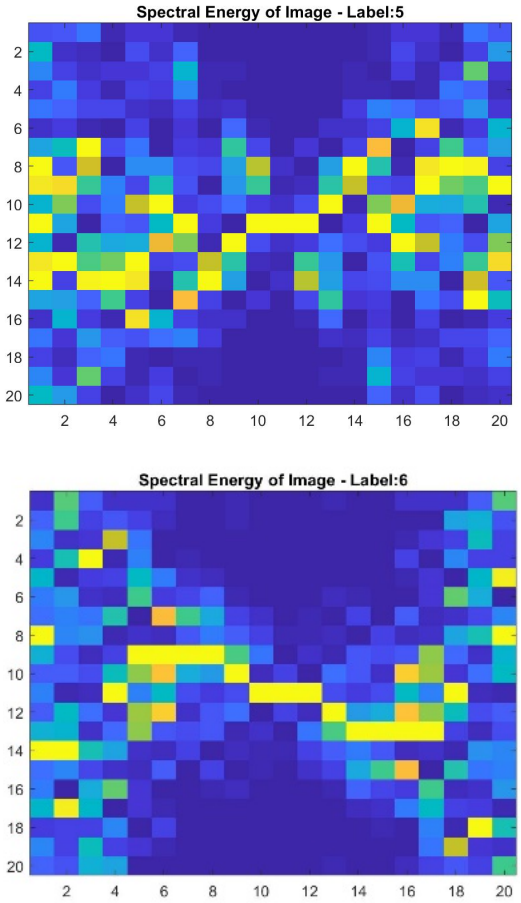

Figure 4: Spectral Energy Plots of Test Images

Visualizing and summarizing the remaining features led into a discovery that features extracted from images in the MNIST dataset, irrespective of whether they looked similar or not, had surprisingly comparable summarizations. Table 1 summarizes the results of the different feature summaries gathered for this assignment for the digits ‘5’ and ‘6’.

Features	Summaries '5'	Summaries '6'
Image Histogram	*Mean*: 0.2579 *Std Dev*: 0.3765	*Mean*: 0.2672 *Std Dev*: 0.3659
Spectral Magnitude	*Mean*: 0.2562 *Std Dev*: 0.3897	*Mean*: 0.2662 *Std Dev*: 0.3887
Spectral Energy	*Mean*: 0.2059 *Std Dev*: 0.2769	*Mean*: 0.2337 *Std Dev*: 0.2945
Logarithmic Energy	*Mean*: 0.3374 *Std Dev*: 0.4774	*Mean*: 0.3232 *Std Dev*: 0.4473
Cepstral Coefficients	*cc1*: 0.5091 *cc2*: 0.0328 *cc3*: 0.0888 *cc4*: 0.0350	*cc1*: 0.8310 *cc2*: -0.0520 *cc3*: 0.2129 *cc4*: 0.1361
Spectral Centroid	181.4367	178.7367
Spectral Flatness	0.6342	0.6284
Spectral Crest	4.1257	3.8381

Table 1: Summary of Feature Extraction Summarizations of Test Images

Analyzing the values obtained in Table 1, one can safely assume that the images representing '5' and '6' are very similar in terms of feature summarizations. Cepstral coefficients are the only feature that seem to be different for the two test images. To answer that, we ran the kNN algorithm to help us determine which features were important in classifying the images in the MNIST dataset. We found that the accuracy with all the aforementioned features included was only ~19%. To test which features were the most important to the classification, it was necessary to run several tests with different sets of features as the input to the kNN algorithm. We also found some features that did not affect the performance of the classification, namely spectral centroid, spectral crest, spectral flatness and entropy. Contrary to intuition, features like cepstral coefficients, which we originally thought could have been important, turned out to be only slightly helpful to improving the accuracy to approximately 27% with only itself as the input to kNN. Out of all the tests done with features as input to kNN, the highest accuracy obtained was approximately 32% with k = 7. Running the same kNN algorithm with only the raw image data as the input, yielded a very high accuracy of 97.05% with k = 3.

Additionally, one may also wonder that if there are other digits for which these features have differences. From our results obtained from feature extraction for several different digits and test cases, we made our case stronger because some other digits had even higher similarities than the cases we presented in this report.

IV. DISCUSSION AND CONCLUSIONS

In completing this assignment, we learned quite a few things about feature extraction, feature summarization, and kNN method in the context of image classification. Similar to previous assignments, setting up the skeleton of the solution did not take that much of a time contribution, but rather it was fine-tuning the features and other signal processing aspects of the project.

At the onset of this project, our goal was to use feature extraction, summarize those features, and then use that summarized vector as an input to our kNN algorithm. Based on the poor accuracy results that we saw when we attempted this solution in a similar way to audio signals, we realized the shortcomings of our approach in the context of the MNIST dataset. The dataset that we are using in our assignment is a grayscale image, meaning that there is not separate RGB color planes. Perhaps the approach of feature extraction and summarization before introduction to the kNN algorithm would have worked much better if there were richer test images, such as a dataset with color images.

Another shortcoming of our original approach was that we attempted to summarize information without taking into account the spatial distribution of pixel values. Without making use of that information, you lose much of the information that makes the image unique and therefore suitable for a kNN algorithm. Additionally, we could have not summarized the features and instead used the unabridged features to feed into the kNN algorithm. This way some features like image corner-ness could have been helpful in differentiating between digits as corner-ness takes spatial aspects of an image into consideration.

V. AUTHOR CONTRIBUTIONS

The work for this project was very evenly distributed between Blake and Himangshu. The work for this assignment was split but was at the same time coherent. Since we already had the kNN algorithm from last quarter, most of the time this assignment was spent in testing different sets of features, done by Himangshu. Blake mostly worked on collecting summarizations of features for different test cases and finding similarities and differences in the features. Blake also eventually worked on proving the unfeasibility of the features to the kNN in classifying images in the MNIST dataset. Both had equal contributions on the report.