**Name:** Jake Taylor **Student Number:** 101147594

# Using the Principal Components of Images of Chinese Characters for Classification

**Abstract**

In this report, I applied the "Eigenfaces" algorithm developed by Turk and Pentland to classify a set of images of 15 unique handwritten Chinese characters. By applying a threshold of 20 to the image pixels, approximately centering the character pixels in the image and using 50 principal components to represent the images in a lower-dimensional space, I achieved an optimal test set accuracy of 0.966.

Using the optimal 50 principal components without applying the additional preprocessing steps, the standard "Eigenfaces" algorithm achieved a test set accuracy of 0.759. Therefore, the additional preprocessing of the images proved to be a critical aspect of achieving high accuracy in classifying the character images in the test set.

When using principal component analysis, we also always applied PCA whitening, which improved the performance of the model both with and without additional preprocessing.

**Introduction**

In recent years, we commonly see the use of a convolutional neural network to classify a set of labeled images, such as the "Chinese MNIST in CSV – Digit Recognizer" dataset used in this report. In my own experience working on image classification problems both in school and recreationally, I have only used neural networks to solve them. Thus, when I discovered this "Eigenfaces" algorithm, it presented a unique way to solve these problems that made it interesting.

Although it was only mentioned in the context of face images in the literature, I wanted to see how well it would perform on a novel dataset and how I could improve the performance of the algorithm by applying additional preprocessing to the input images.

The key challenges that presented themselves were figuring out exactly how many principal components would be optimal to use to represent the images and determining the methods of preprocessing that could be used to further improve the accuracy of the model. These preprocessing methods would prove to also have hyperparameters that had to be optimized, further increasing the difficulty of the problem.

**Background**

The mathematical model used in this report was proposed in the paper from 1991 titled "Face Recognition Using Eigenfaces" by Matthew A. Turk and Alex P. Pentland. The main concept discussed in the paper involves extracting the principal components from the covariance matrix that corresponds to a set of training face images of known individuals and using them to classify unseen face images of the same individuals.

They begin with a set of $M$ face images of size $N \times N$ that correspond to $K$ unique individuals. In the paper they explain that they first flatten each image, $I_i$, $1 \leq i \leq M$, into a vector of length $N^2$ and place them all into a matrix of shape $N^2 \times M$, which we will call $I$. Then, from each of the columns of the matrix they subtract the mean image vector, which we will call $\mu = \frac{1}{M}\sum_{n=1}^{M} I_n$, to create the matrix of centered image vectors $\phi$. Once they have computed $\phi$, they then compute the covariance matrix $C = \phi\phi^T$ of size $N^2 \times N^2$. In the next step, they compute each of the eigenvalues, $\lambda_k$, of the matrix, $C$, and

each of their corresponding eigenvectors, $u_k$. It should be noted that in the case that $N^2$ is much larger than $M$, the paper cites another paper that details a method to compute the eigenvalues and eigenvectors using a matrix of size $M \times M$ instead. The cited paper was ignored since the Chinese MNSIT dataset used in this report has $N^2 = 4096 < M = 15000$ images. The eigenvectors of the matrix, $C$, that correspond to the largest eigenvalues explain the most variance in the training set of face images. These eigenvectors are referred to as "eigenfaces" for the simple fact that they are the eigenvectors of the covariance matrix, $C$, of the training set of face images and they resemble faces themselves. They then choose $M' < M$ "eigenfaces" that correspond to the $M'$ largest eigenvalues of the matrix, $C$, to be their final set of "eigenfaces". They place these $M'$ "eigenfaces" in a matrix, $U$. The final step before classification of new images is projecting each of the $M$ original training set images onto the $M'$-dimensional space spanned by the "eigenfaces", which they refer to in the paper as projecting them onto "face space", accomplished by $\Omega = U^T \phi$.

The paper explains they can classify a new face image by first converting it into a vector of size $N^2$, subtracting the mean image μ from it and projecting it onto "face space". Then, they compute the Euclidian distance of the resulting vector to each of the projected face images in the matrix $\Omega$. For the minimum such distance, $\epsilon$, the new face image is classified as being of the same person as the projected face image in the corresponding column of Ω if $\epsilon < \theta$, for a threshold $\theta$. Otherwise, if $\epsilon \geq \theta$, they classify it as unknown.

**Methodology**

The dataset that was used in this report was the "Chinese MNIST in CSV – Digit Recognizer" dataset uploaded to Kaggle.com by the user "Fedesoriano". The dataset consists of 15, 000 images of handwritten Chinese characters in black and white. Each of the 15,000 image instances in the dataset consists of 4098 columns, where the first 4096 columns represent the pixel values of the character image, the 4097th column represents the numerical value of the character, which is a digit, and the last column represents the character in plain text. In order to have a single label for each of the images in the dataset, each of the 15 unique plain text character labels was encoded as a unique integer in the interval [0, 14]. A summary of the label encoding is provided below.

{0: '一', 1: '七', 2: '万', 3: '三', 4: '九', 5: '二', 6: '五', 7: '亿', 8: '八', 9: '六', 10: '十', 11: '千', 12: '四', 13: '百', 14: '零'}

**Figure 1:** A summary of the encoding of the labels for the 15 unique characters that make up the "Chinese MNIST in CSV – Digit Recognizer" dataset. Every image in the dataset is a handwritten drawing of one of these characters. Each entry in this dictionary has the form <Encoded Label>:<Plaintext Character>. The <Encoded Label> will simply be referred to as the "label" from now on.

The distribution of the complete dataset according to the labels of the images in the dataset is shown below.

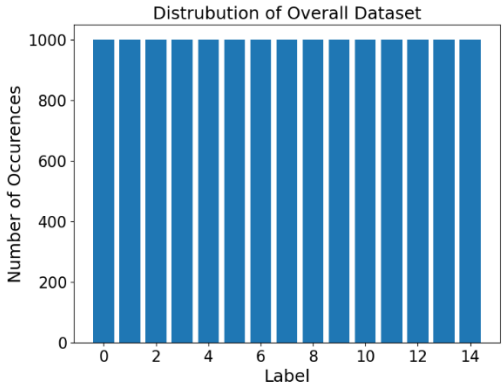


**Figure 2:** The distribution of the images in the "Chinese MNIST in CSV – Digit Recognizer" dataset. As we can see from the figure, there are 1000 images for each of the 15 unique characters in the dataset.

The method we used for classifying the Chinese characters in the "Chinese MNIST in CSV – Digit Recognizer" dataset was the same as the method used in the paper "Face Recognition Using Eigenfaces" to classify face images. In order to evaluate the performance of this method for classification, we split the matrix of character images into a training set of 12, 000 instances and a test set of 3,000 instances. We only used the images in the training set to construct the "eigenchars", which are the eigenvectors of the covariance matrix of Chinese character images in the training set. Here are the 2 "eigenchars" from one iteration of the implementation that explained the most variance in the training set:

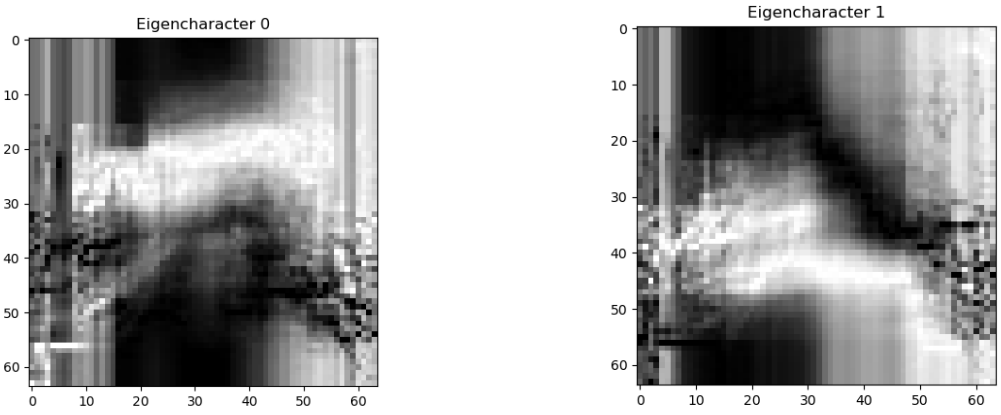


**Figure 3:** Images of the "eigenchars" that explain the most variance (left) and the second most variance (right) of the training set of images. These are 2 of the "eigenchars" for the training dataset before any preprocessing has been applied.

Each of the test set images was classified in the same way as a new face image was classified in the paper, with the exception that we did not use a threshold $\theta$. Instead, we classified the test image with the same label as the image in the matrix of projected training set Chinese characters that the projected test set image was closest to, according to Euclidian distance. Since all of the images in "Chinese MNIST in CSV – Digit Recognizer" have labels, the images in the training and test sets will also both have labels. Using the labels, we know a prediction is correct if the label of the non-projected training set image to

which the projected image the projected test image was closest to corresponds is equal to the label of the test set image.

After splitting the data into a training and test set, for a given run of the implementation, here are plots of the distribution of Chinese character images according to their label in the training and test sets, respectively:

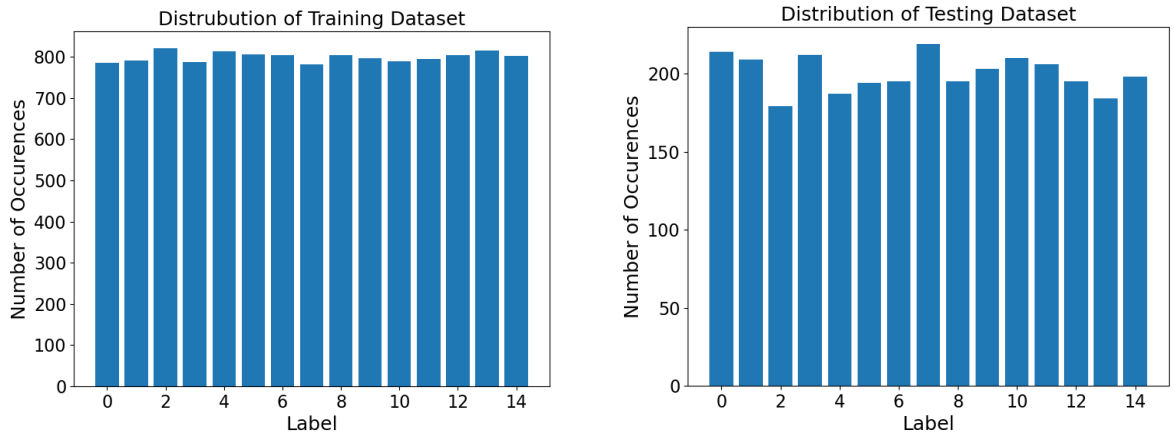


**Figure 4:** Distributions of the training and test sets of Chinese character images according to the labels associated with the images in each dataset. The distributions of images according to label in both sets were always approximately evenly distributed among the labels.

As we can see in *Figure 4*, by randomly splitting the dataset into a training and test set, the distribution of images in each set was fairly evenly spread out among all of the labels. This pattern was observed every time the dataset was split during different runs of the implementation.

In order to obtain the best results possible, a number of additional steps had to be taken such as tuning the number of principal components ("eigenchars") to use from the image covariance matrix and applying additional preprocessing. Furthermore, when applying PCA to the dataset, we always applied PCA whitening since it reduces the amount of correlation between pixel values when applied to images and in all cases that will be discussed in "Results", using it led to a greater accuracy on the test set.

**Results**

Before applying any preprocessing to the image data and using 50 "eigenchars", we obtained an accuracy of 0.759 on the test set. Although this was a decent accuracy, the goal for this project was to obtain an accuracy of at least 90%. In order to try and improve the accuracy on the test set, I applied additional preprocessing to the images.

When viewing the images in the dataset, one of the most notable things I found was that the character images were not consistently centered in the frame. Instead, some of the characters were closer to the corners of the frame while others were closer to the center. Therefore, in an attempt to improve the accuracy of the model on the test set, I first tried centering the characters in the frame as a preprocessing step. The method I used zooms in upon the region of the image where there is handwriting by determining a bounding box of nonzero pixel points around the handwritten digit. It then removes all points outside of this box from the image and finishes by resizing all of the images back

to a common shape of (64, 64) using bilinear interpolation. Although it does not exactly center every character, it works fairly well as is shown in the images below.

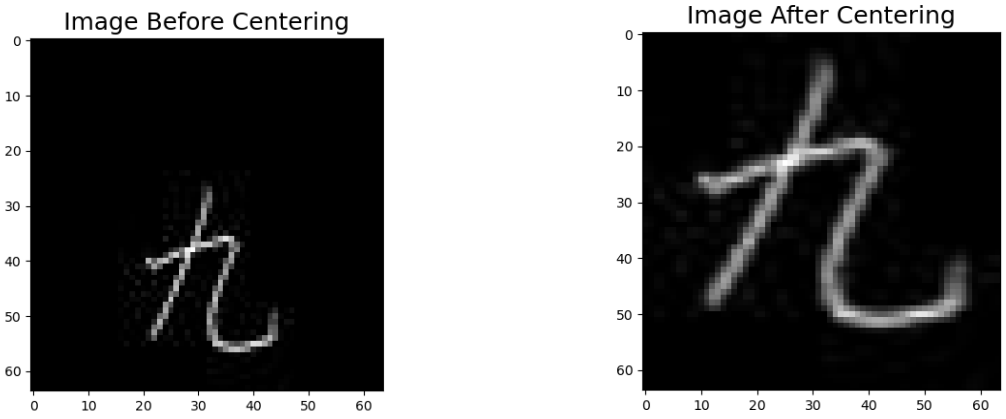


**Figure 5:** For a given image in the dataset, the left shows the image before the centering algorithm was applied and the right shows the image after the centering algorithm was applied.

In order to evaluate the effectiveness of the centering algorithm on improving accuracy, we held everything else constant in the model, continuing to use 50 principal components ("eigenchars"). After preprocessing all of the images in both the training and test sets using the centering algorithm described above, the accuracy of the model increased from 0.759 without centering to 0.8887 with centering. Therefore, I kept this preprocessing step in the final model.

The next step I performed to try to improve the performance of the model was trying to tune the number of principal components, or in this case "eigenchars", to keep after applying PCA. In order to determine the optimal number of principal components, I held everything else in the model constant at the best model I had found so far, which included using the centering algorithm mentioned above for image preprocessing, and I varied the number of principal components. With 50 principal components having achieved an accuracy of 0.8887, we would only change the number of principal components if it led to an accuracy greater than this value. We can plot the accuracy the model achieved on the test set as a function of the number of principal components, or "eigenchars", that we used as follows:

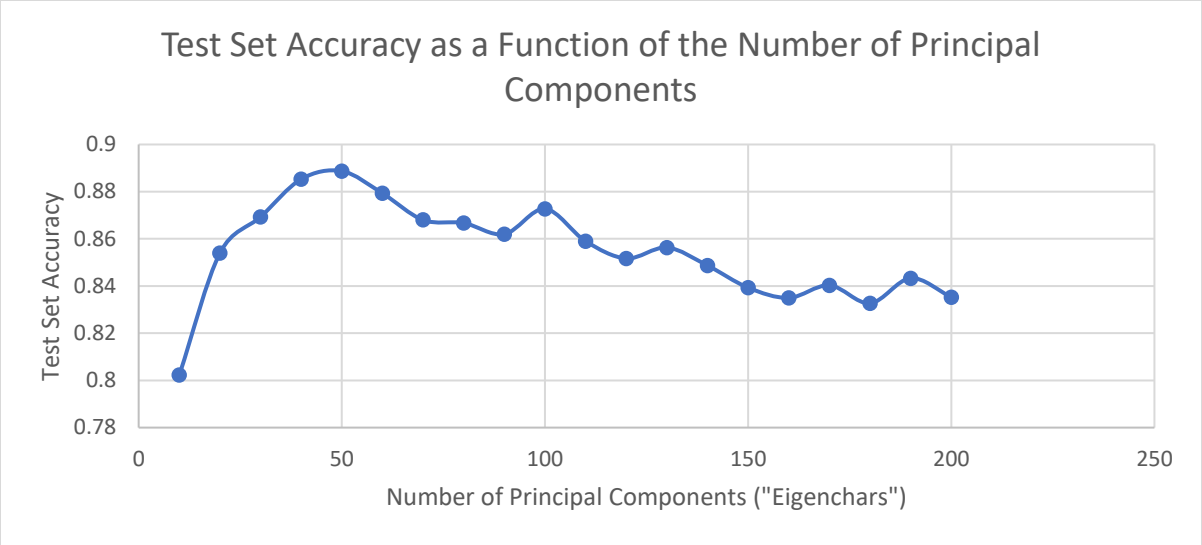


**Figure 6:** Plot of the accuracy obtained by the Chinese handwritten character classifier as a function of the number of principal components used to represent the original handwritten characters.

Therefore, as is shown in *Figure 6*, the optimal number of principal components to use was 50, achieving an accuracy of 0.8887. Hence, I kept the number of principal components at 50.

The final attempt I made at increasing the accuracy of the model was to threshold the pixel values in the original image before centering the image, such that all pixels with values less than the threshold were set to 0 and all pixels with values greater than or equal to the threshold were set to 255. The motivation to do so came from observing the original images and noticing that surrounding the main outline of the characters in each image, which was whiter in colour, there were always many blurry gray pixels. In addition, when looking at the centered images, the pixels causing the character to still be shifted slightly off-centre tended to be these gray pixels. In order to choose the best threshold, I experimented with thresholds in the interval [0, 100] that are multiples of 10. The model used the same centering algorithm as mentioned above, with the exception of applying the threshold before centering, and used 50 principal components. I compared the accuracy using the best threshold in the interval with the previous best accuracy without using a threshold of 0.8887 to see if using a threshold would be worthwhile. I plotted the accuracy of the model on the test set as a function of the threshold as follows:

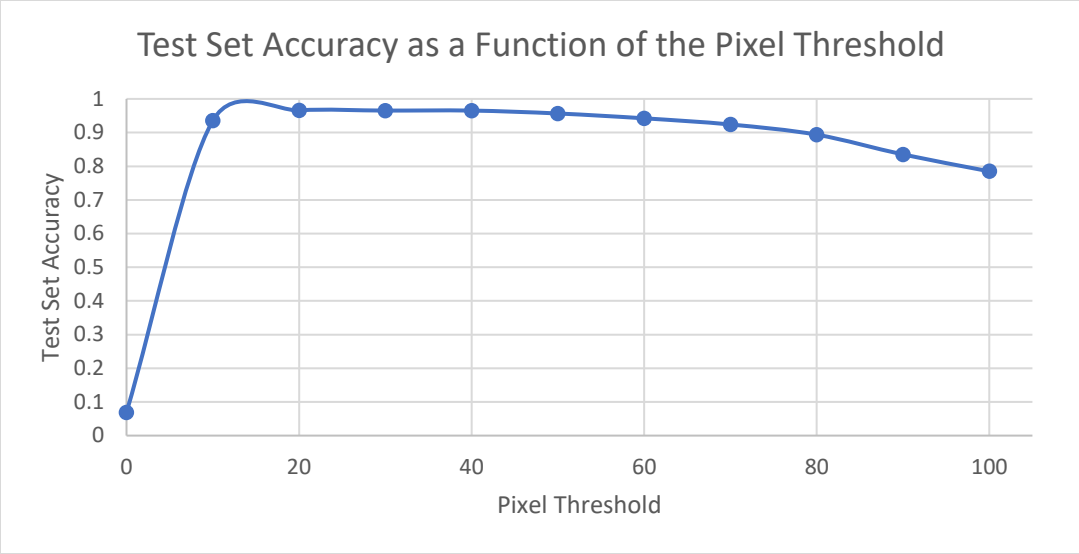


**Figure 7:** Plotting the performance of the model as a function of the pixel value threshold applied to the input images before they were centered in the frame.

Therefore, as we can see in *Figure 7,* the optimal threshold value is 20, leading to an accuracy on the test set of 0.966.

**Discussion**

The relevance of my findings is that I have shown how to improve the accuracy of the "Eigenfaces" algorithm proposed by Turk and Pentland. By first thresholding the pixel values to reduce the amount of blur surrounding each of the characters and then centering the character images in the frame, I improved the performance of my Chinese character classifier. Having achieved a final test set accuracy of 0.966 shows that the classifier was effective when applied to a novel dataset and that the basic "Eigenfaces" algorithm can be slightly tweaked to achieve high accuracy using images of things other than faces.

The main limitation of my findings is that the dataset is fairly simple, consisting only of written Chinese characters. A model similar to the one developed in this report may struggle to achieve high accuracy when trying to classify a more complex and diverse set of images. In addition, the model would likely be unable to classify other Chinese characters that were not seen in the dataset.

**References**

1. Turk, M. A. & Pentland A. P. (1991). Face Recognition Using Eigenfaces. https://sites.cs.ucsb.edu/~mturk/Papers/mturk-CVPR91.pdf