Handwritten Chinese Characters Recognition Model
STAT-UN3106 Data Mining Final Project

An, Qi and Xing, Hanwen
Columbia University
Instructor: Wayne T. Lee
May 7, 2022

Table of Contents

Section I: Introduction and Motivation

In the middle of the Spring semester of 2022, the younger brother of one of the team members called him and asked him to check some homework. This particular homework assignment involved handwritten Chinese characters, and the purpose of the homework was for students to correctly identify similar Chinese characters and use them with accurate syntax in word formations. (In Chinese, although one single character could sometimes carry meanings by itself, most of the time, words and sentences are formed and are only meaningful after some combinations of characters.) While checking the homework manually, the team member, with a great passion for data mining and maximizing work efficiency, noticed that manual grading of Chinese characters was not only time-consuming and eye-straining, but most importantly prone to human errors. The grudge to manually check homework for similar Chinese characters gives us the motivation to develop a data mining project about Chinese character recognition.

Our main audience is divided into two groups: Chinese primary and secondary school students and teachers. In 2021 in China, there are an estimated 107 million elementary school students, 20 million students who participate in ZhongKao (High School Entrance Examination), and 10 million students who participate in GaoKao (College Entrance Examination) (Ministry of Education of People's Republic of China, 2021). If we consider testing and homework in Chinese (the subject), more than half of elementary school's work requires recitation of Chinese characters. Although this percentage decreases as students move into higher education, there are some degrees of Chinese character memorization and recitation. Considering the population of Chinese students, manual work of checking for wrong or similar characters is going to be massive. Hence, our project could potentially solve this problem by eliminating manual labor for checking for the wrong character. Additionally, this project could be applied to other East Asian languages such as Japanese or Korean. Most Japanese Kanji resemble traditional Chinese characters, and Korean characters also contain strokes that could be analyzed in the same way as Chinese characters. While some may concern the development of such technology could lead to teachers losing their jobs, it is quite the opposite. With such automated recognition, teachers could spare time to focus more on their students and curriculum.

For this project, we want to train a machine learning model for handwritten Chinese character recognition from images where both the target and wrong characters are identified. Specifically, in this project, we are calling characters that are similar to but not identical to the target character wrong. Additionally, among those characters that are wrongly classified, we want to find out why those characters are classified wrong. We also want to use data mining techniques to analyze how the computer sees characters as similar versus different.

We used three machine learning models to predict and differentiate between handwritten Chinese characters. Specifically, we used random forest, logistic regression, and an attempt on TensorFlow which is a neural network method. Among the three models, accuracy and F1-score from the neural network model are the highest, yet not significantly higher than the other models.

This paper will be divided into eight sections. Section I, the current paragraph talks about project motivation and goals for this project. Section II focuses on data collection and data processing. More specifically, we will discuss grayscaling, size rescale, and applying Gaussian blur to images with handwritten Chinese characters. Section III involves feature selections that produce the desired matrix for model training. Features will be selected by Hough transform and Harris corner detection, both methods isolate and extract 16 features from input images. As a result, a total of 32 features would be selected and put into the desired matrix for model training. Section IV introduces exploratory data analysis of our feature matrix to better understand the distributions of features and to come up with a hypothesis for our model prediction. Section V describes three machine learning models that are used to predict the correct versus wrong characters: logistic regression, random forest, and neural network. In section VI, we introduce a new dataset containing very different-looking characters which test the robustness of our machine learning model. Section VII is about the data mining aspect of this project. Mainly, we will identify and propose reasons that the computer could or could not identify characters. The last section, section VIII concludes this paper by summarizing results, calculating the accuracy of our model, and discussing limitations and future extensions to this current project.

Section II: Data Collection and Processing

Section II.I: Data Collection

For this project, handwritten Chinese characters were collected from Kaggle. The original dataset was created by the Chinese National Laboratory of Pattern Recognition (NLPR) and the Institute of Automation of Chinese Academy of Sciences (CASIA). The whole dataset is produced by 1020 writers between 2007 and 2010, and it contains 7330 handwritten characters in PNG images. All characters are written on a white background with black to gray colored ink. Among those characters, we have picked 9 different characters to complete this project.

Five of these characters have the same radical, the leftmost component to Chinese characters that are considered as the base component, and those are categorized as similar characters for model training and are put into the first dataset. They are: 伍(wu), 伉(kang), 伏(fu), 伊(yi), and 伋(ji).

We are calling 伍(wu) as our target character, or correct character, and the other four characters as wrong. The second dataset contains five characters that look different, i.e. we purposefully choose them to have different radicals. Since we still want to test whether 伍(wu) could be correctly identified among others, 伍(wu) is also included in this second dataset. Specifically, the second dataset contains 伍(wu), 乘(cheng), 个(ge), 上(shang), and 万(wan). Similarly, we are calling 伍(wu) as our correct character while others as incorrect.

For the first dataset, there are around 240 different handwritten versions of each of 伋(ji) and 伉(kang); there are around 550 each of the other three characters, resulting in a total of 2197 pictures of a mix of the five characters. For the second dataset, we want to control the total number of handwritten characters, so discrepancy inaccuracy results from the same machine learning models could be explained more easily in terms of features within data instead of the dimension or number of data. Hence, we have included around 240 handwritten images of 乘(cheng) and 个(ge) and 550 of the other three characters, resulting in a total of 2197 images of characters as well.

Section II.II: Grayscale

With around 2200 images in each dataset, we need to perform image processing to transform human identifiable pictures into machine-readable arrays of numbers for it to pick up important information within certain images. Images are viewed as a combination of pixels where a pixel is the smallest item of information in an image. Cutting the image into a small grid of squares, each square will be a pixel. The variety of colors in a pixel is the result of different intensity combinations of red, green, and blue (RGB). The intensity of a particular color among the three is represented by a number between 0 and 255, where 0 means none of that color and 255 means all of that color. Hence, the color in one pixel could be represented by a three-item tuple, where the first number represents the intensity of red, the second represents the intensity of green, and the third number represents the intensity of blue. For example, if a pixel is purely blue, the tuple would look like (0,0, 255). And by combining those colorful pixels, the original image will appear.

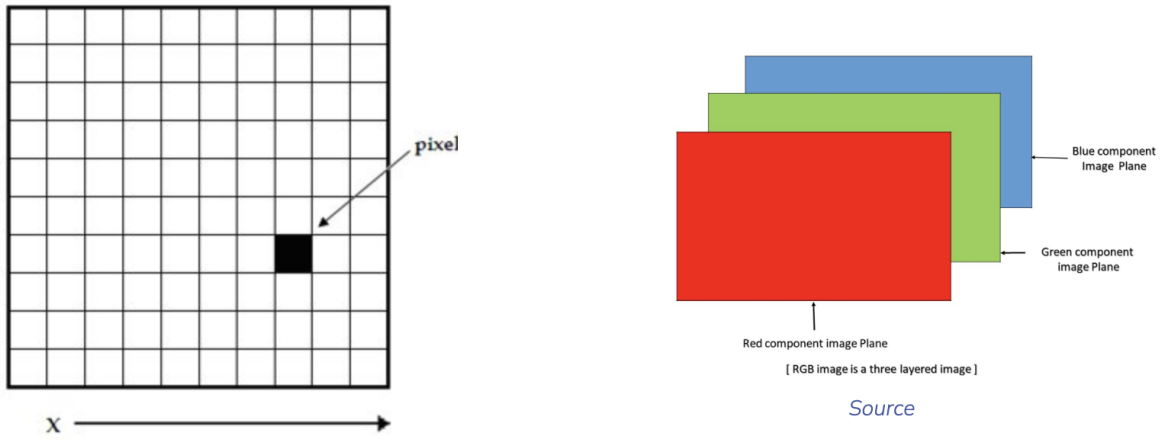


Figure 1: Pixel within a picture and RGB Components of a Image Plane
(Image source: Neptune.ai)

In order for the machine to extract the correct information from images to better serve the purpose of character recognition, one of the first steps is grayscaling. grayscale is a method to change the original color image into an image that only contains “black, white, and gray, in which gray colors have multiple levels” (Liu, 2020).

Grayscale is important in image processing because it “simplifies the algorithm and reduces computational power” (Kunchala, 2021). It was mentioned earlier that each pixel contains information about the combination of red, green, and blue. Keeping those colors requires more computational power and complicates the model. However, from the perspective of character recognition, we do not need the machine to differentiate between and keep information about different colors since our images are mainly black and white.

Before the grayscale transformation, each pixel stores a three-item tuple; however, after the grayscale transformation, each pixel only stores one number representing the level of grayness. The resulting array size is original pixel width times original pixel length, but only one number in the color channel.

Figure 2: An Example of Grayscale Image. (A) original image (B) grayscale image
(Image Source: Patil, Mrunmayee C. and Kagalkar, Ramesh M. "An Automatic Approach for Translating Simple Images into Text Descriptions and Speech for Visually Impaired People")

Section II.III: Resize

In addition to grayscaling, we resize the image to 300-pixel width times 300-pixel length with the character in the center position. This size transformation is performed in two steps. First, given a non-square image, we want to change it to a square. For this image, we pick the larger dimension between width and length, and we create a white square image using the larger dimension to act as the background, to which we paste the original image at the center to get a squared image containing this character. By the end of this step, all images should be square. Second, we will resize all those images to 300-pixel width times 300-pixel length.

Section II.IV: Gaussian Blur

Our final step in terms of data processing is Gaussian Blur. Gaussian Blur is the process of blurring an image using a Gaussian function. While the term Gaussian or Gaussian distribution might be unfamiliar, most of us all have some familiarity with or have seen a Gaussian distribution. It is often referred to as a normal distribution or a bell curve in one dimension where the distribution has only one peak, and the peak is at the mean of the distribution. The Gaussian distribution in two dimensions is depicted below.

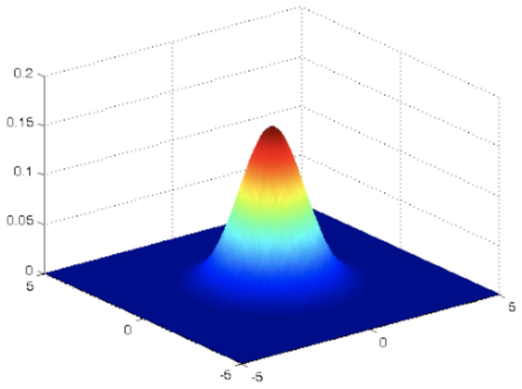


Figure 3: Gaussian distribution in two dimensions
(Image Source: Sharda, Aryaman "Image Filters: Gaussian Blur")

The idea of Gaussian blur is to take a filter of numbers, called the convolution kernel, created from the 2-D Gaussian Distribution and overlay this filter to each section of the input image to output new pixels to replace the original pixels at their original positions. Since we could consider the height to the Gaussian distribution as weights, pixels at the center of the overlaid portion plays a more important role in outputting the new pixel, which is the essence of how Gaussian blur blurs input images. An example of overlaying filters and an example of comparing results before and after Gaussian blur is included below. The importance of Gaussian blur to character recognition is to reduce image noise, which is commonly used in image processing.

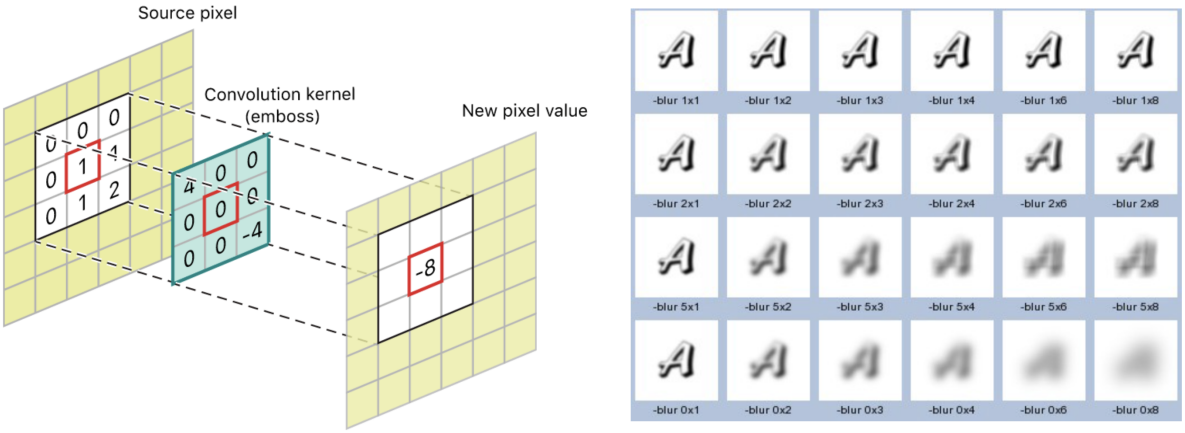


Figure 4: Example of how to overlap filter to output new pixel values (left) and real image example of different degree of Gaussian blur (right)
(Image Source: Sharda, Aryaman "Image Filters: Gaussian Blur")

From data collection to grayscaling, to resizing, and to Gaussian blur, we have successfully transformed our image data to a uniform array where the image of each handwritten character is 300 pixel-width times 300-pixel length and one value in each entry to represent the intensity of gray levels for each pixel. The final data preparation step is to combine the arrays together. We concatenate the values one row after the other so that one character will be in one row with 90000 (300*300) columns. Since in each dataset, we have 2197 characters, the resulting matrix from data preparation will be a matrix of 2197 rows with 90000 columns.

Section III: Feature Selection

Due to limited computing power on our laptops, we will perform feature selection on the 90000 columns to get our final data matrix that will be used in training machine learning models. In this section, Hough transform and Harris Corner Detection are introduced as methods to reduce original data dimension by feature selection.

Section III.I: Hough Transform

Hough transform is used in image processing to detect lines and simple shapes like circles in an image. An example can be seen below where an input image generates edges, and when the edges are passed into Hough transform, the algorithm detects lines.

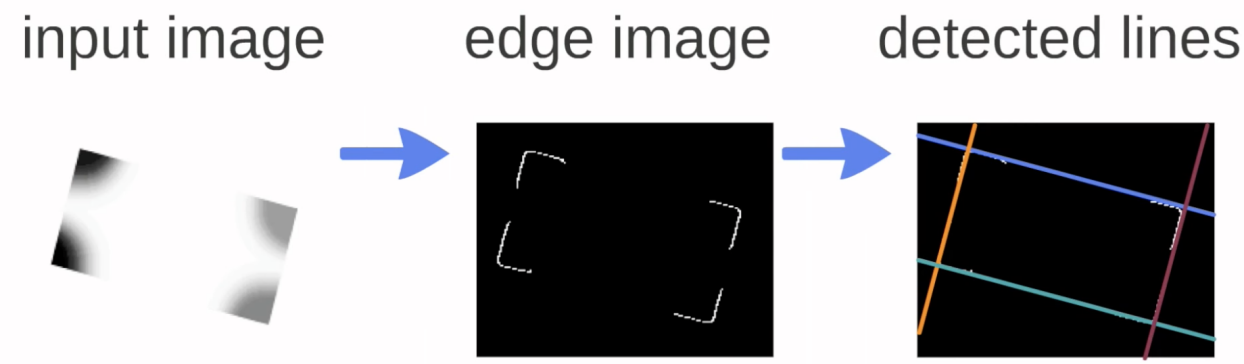


Figure 5: Example of Hough Transform

From knowledge of linear equations in high school, we know that in the x-y plane, a given line can be represented by $y = ax+b$ where a is the slope and b is the intercept. However, the same line could also be represented in polar coordinates in the θ and ρ plane where ρ represents the vertical distance between the origin and the line, and θ is the angle between the x-axis and the vertical distance line. The conversion equation is shown in the picture below.

Despite the fact that this transformation is making our life less intuitive and involving more math, there are benefits to it. While the slope could take any value between negative and positive infinity, θ is bounded between 0 and π. Since the Hough transform requires the parameters to be bounded, switching to polar coordinates allows us to perform this particular transformation.

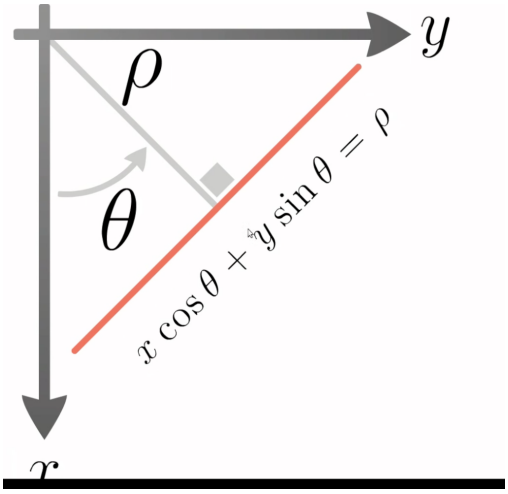


Figure 6: Two Ways of Representing a Linear Line

Another interesting aspect of changing from Cartesian x-y coordinates to polar coordinates is that one line in the x-y coordinates corresponds to a point in polar coordinates, and one point in the x-y coordinates corresponds to a curve in polar coordinates.

An intuitive way to understand the transform is that when an image is initially passed in, an algorithm detects the edge image and finds all points on the edges. Once the algorithm touches a point on the edge, Hough transforms points into polar coordinates in the θ and ρ planes, in which original straight lines will become curves. The algorithm continues to iterate through all the possible points in the edge image and draws a curve in the θ and ρ plane whenever it touches the edge. By the end of the iteration, a Hough transforms intensity plot will be drawn, from which you could count the number of lines within the input image.

An example is shown below with four lines from the edge image (right) and four intensity points circled in red (left). In the input image, the edges are white, and the lines from the Hough transform are yellow. We know there are four lines because there are four intensity points in the θ and ρ planes.

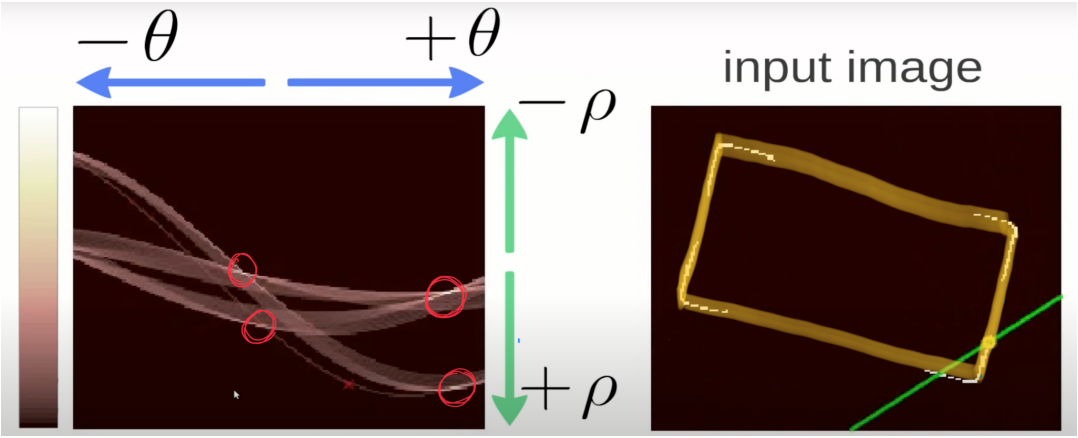


Figure 7: Hough Transform intensity plot and Input Edge Image

The first set of features for our handwritten character recognition model is generated through Hough transform. We divided our input image into 4 by 4 regions (a total of 16 regions). In each region, we calculate the number of lines detected in the region and output a number. Hence, for each character there are 16 features selected from the original image.

Section III.II: Harris Corner Detection

Harris Corner Detection is most commonly used in computer vision algorithms to extract corner information about an image. A corner is defined as "a point whose local neighborhood stands in two dominant and different edge directions" (Tyagi, 2019), meaning a corner exists at a point where two edges from different directions join each other. An example is demonstrated below.

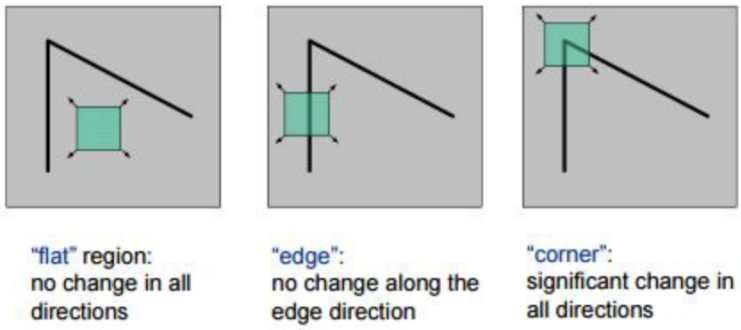
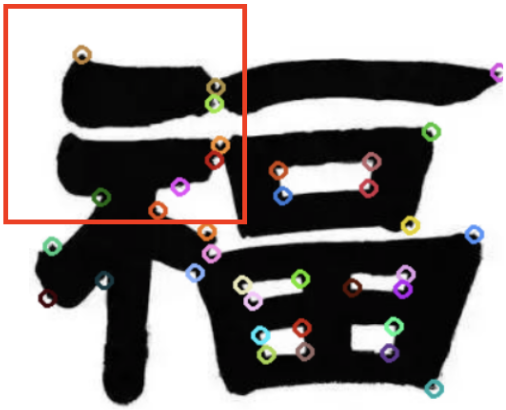


Figure 8: Example of Flat, Edge, and Corner
(Image Source: Tyagi, Deepanshu "Introduction to Harris Corner Detector")

Harris Corner could be crucial in terms of Chinese characters recognition since the direction of strokes usually determine what the character looks like and what the character is. An example of Harris Corner is used on the Chinese character 福(fu) below. After the detection of Harris corners, we could calculate how many corners are there in each particular region. In the fu example case, there are 8 corners identified in the red-box region.



We performed the same thing onto our 2197 characters. Similar to Hough transform, for each character, we divided the image into a 4 by 4 region (a total of 16 regions). In each region, we apply Harris corner detection algorithm and calculate the number of corners in each region. Hence for each character, there will be 16 different numbers outputted as features for this character.

To conclude this section, our feature matrix has 32 total columns - the first 16 features are selected from Hough transform, and the next 16 features are selected from Harris Corner detection. Given the total number of handwritten Chinese characters, our final feature matrix is a 2197 by 32 matrix.

Section IV: Exploratory Data Analysis

Using our feature matrix, we will first perform exploratory data analysis to understand the distribution of features across the same character and make hypotheses of our prediction result. To analyze the distribution of features, we take the average number of features for each column for the same character, after which we could plot a bar plot to visualize the distribution. Each bar represents different features, and the height of the bar is the average value of a particular feature across different images of the same character.

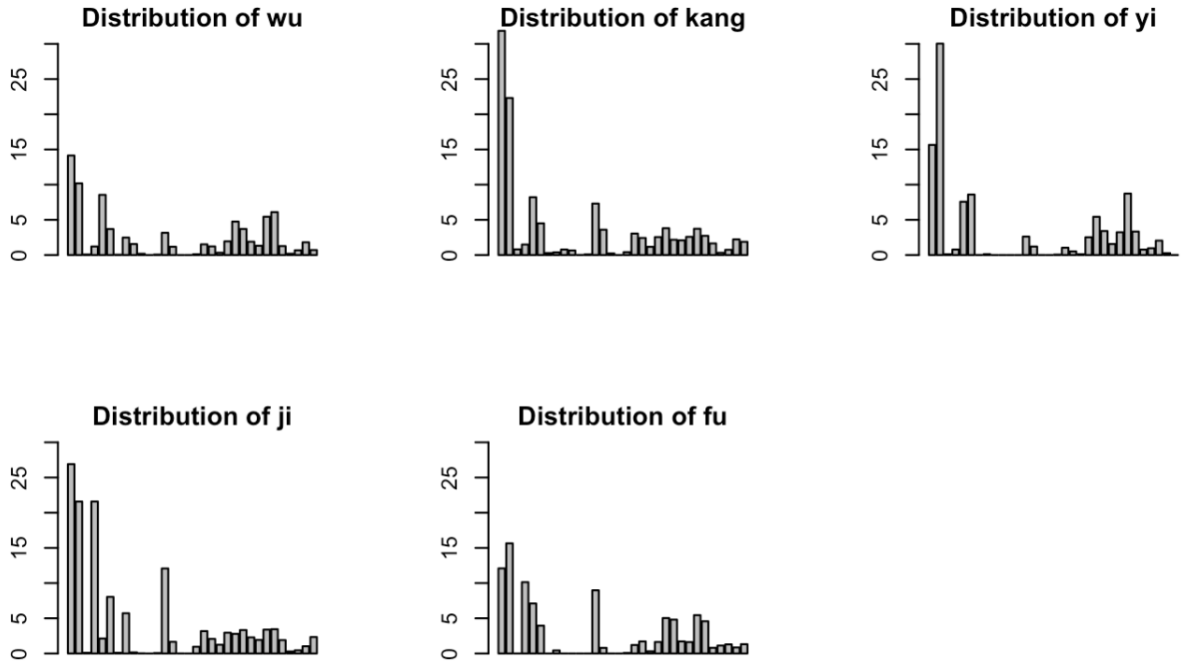


Figure 9: Distribution of Features for Each Character.

From the distribution, we hypothesize that it would be difficult for the machine to separate between 伍(wu) and 伏(fu) since the distribution of features are extremely alike. Before seeing the distribution, we though 伍(wu) and 伉(kang) would be similar from the human eye, yet the distribution shows otherwise - distribution for 伍(wu) and 伉(kang) actually looks quite different. It is interesting to see how machine's perception of similar characters is different from humans' perceptions.

Quantitatively, we could also calculate the Euclidean distance between features for different characters. The result is included below. Similarly, the Euclidean distance between fu and wu is significantly smaller than other pairwise distances.

	wu	kang	yi	ji
kang	22.73132			
yi	21.43682	20.46352		
ji	29.77178	23.25593	28.97927	
fu	12.78375	23.42653	19.79136	22.13507

Figure 10: Pairwise Euclidean Distance

Next, we analyze the data quality issue of our dataset. However, since our original dataset is all images (or arrays generated from reading those images), our data quality analysis looks different

from traditional approaches which usually focus on missing values and non-sense data entries in the raw data file. Instead, we will focus on analyzing the feature matrix data frame generated from feature selection.

We have created a correlation heatmap to compare the pairwise correlation among the 32 features. From the heatmap below, we can see that features in general do not have a high correlation between each other. This makes intuitive sense since our features correspond to different parts of the image (or character). This is also significant in telling us that since there is a low probability of multicollinearity, linear machine learning models should work well for this particular feature matrix. It is interesting to note that row number 15 and column number 15 are completely white. After cross-checking with the original feature matrix, this is might be our potential data quality issue which happens due to Hough transform which transforms this entire column to 0 across all characters.

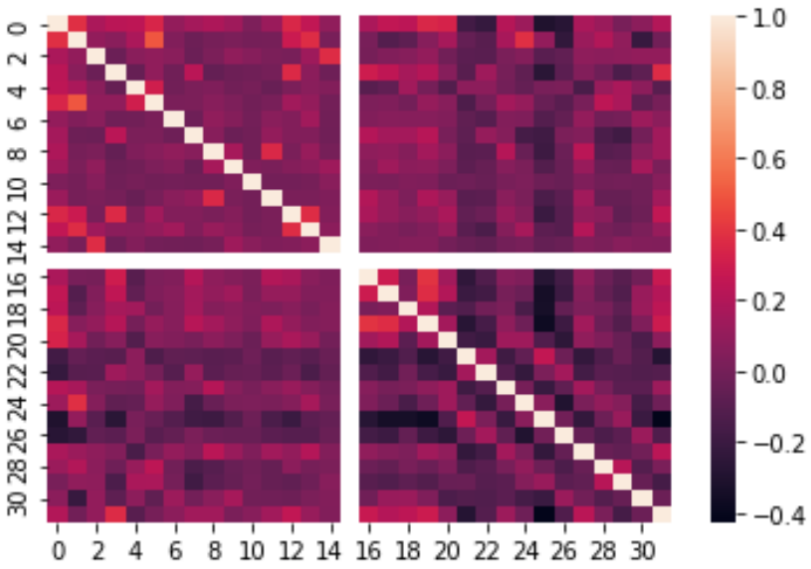


Figure 11: Pairwise Correlation Heatmap for the First Dataset

Section V: Machine Learning Models

In this section, we have fitted one baseline random prediction algorithm along with three machine learning algorithms: logistic regression, random forest, and neural network. Dataset is divided into training and testing sets. Since each character has a different number of images, training and testing sets are created first by each character, where 80% of the randomly selected images are in the training set while the 20% left are the testing sets. This process ensures that each character has the same proportion, or weights, in each set. The final training and testing sets are created by combining corresponding sets of each character.

Furthermore, model accuracy is calculated from F1-score whose equation is displayed below. The higher the F1-score, the more accurate the model is.

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

Precision = True Positives
Recall = True Positives / (True Positive + False Negatives)

		True Classification	
Model Classification		Correct	Incorrect
	Correct	True Positive	False Positive
	Incorrect	False Negative	True Negative

Section V.I: Baseline Random Algorithm

The random prediction algorithm randomly predicts correct versus incorrect based on their proportions as weights in the input dataset. The F1-score for this model is 0.3, which is really low. Hence, this algorithm serves as a baseline to the following models.

```
confusion matrix:
 [[235  87]
 [ 83  36]]
confusion matrix:
 [[0.73 0.27]
 [0.7  0.3 ]]
              precision    recall  f1-score   support

           0       0.74      0.73      0.73       322
           1       0.29      0.30      0.30       119

    accuracy                           0.61       441
   macro avg       0.52      0.52      0.52       441
weighted avg       0.62      0.61      0.62       441
```

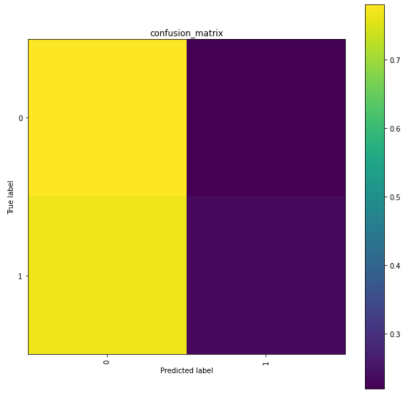


Figure 12: Random Prediction Accuracy Results(Left) and Confusion Matrix(Right)

Section V.II : Logistics Regression

Logistics Regression is the go-to method when predicting binary outputs. The logistic regression function is $Y = \frac{exp(\beta_0 + \beta_1 x_1 + \ldots\ldots + \beta_{32} x_{32})}{1 + exp(\beta_0 x_0 + \beta_1 x_1 + \ldots\ldots + \beta_{32} x_{32})}$ or

$logit(Y) = \beta_0 X_0 + \beta_1 X_1 + ... + \beta_{32} X_{32}$. In these equations, Y is the response binary variable that outputs either 0 or 1. And the X_i's are different independent variables representing features. After using the training set to estimate different β values, we could use those values to predict the binary response variable Y in the test set.

From F1-score, we can see that the logistic regression model has improved significantly from the random prediction - the F1 score increases from 0.3 to 0.81.

```
confusion matrix:
 [[287  35]
 [ 14 105]]
confusion matrix:
 [[0.89 0.11]
 [0.12 0.88]]
              precision    recall  f1-score   support

           0       0.95      0.89      0.92       322
           1       0.75      0.88      0.81       119

    accuracy                           0.89       441
   macro avg       0.85      0.89      0.87       441
weighted avg       0.90      0.89      0.89       441
```

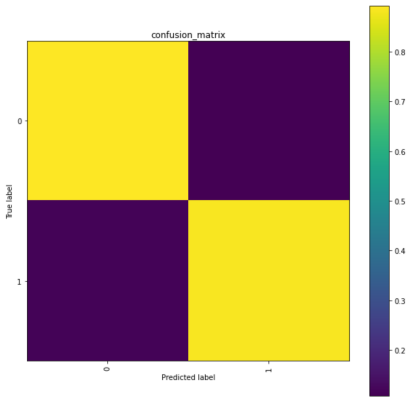


Figure 13: Logistics Regression Prediction Accuracy Results(Left) and Confusion Matrix(Right)

Section V.III: Random Forest

Random Forest is another machine learning algorithm that is used for classification. It uses the bagging ensemble technique meaning that the original data is subsetted with replacement into different subsets. The total size of the subsets will be the same as the original data. For each subset, a decision tree is trained to produce a decision about whether to output 1 for the correct character or 0 for the wrong character. Finally, the class that gets the highest number of votes will be output to be the result. An illustration is shown below.

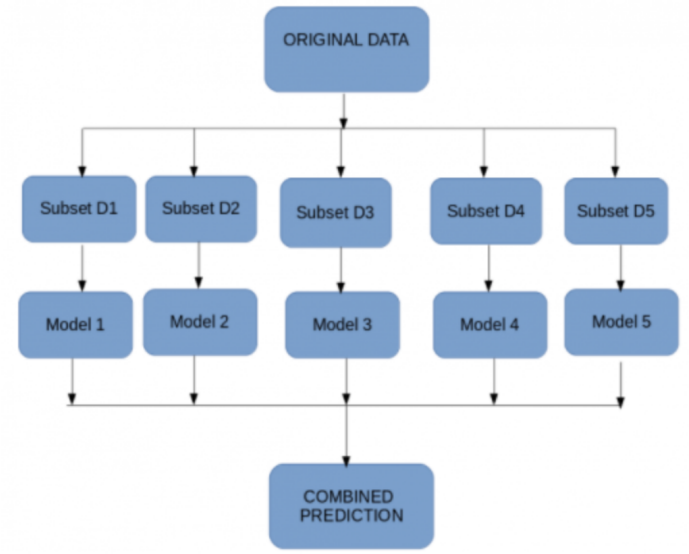


Figure 13: Random Forest Intuition
(Image Source: Saini, Anshul "An Introduction to Random Forest for beginners")

Fitting the random forest model on the feature matrix, gives us the F1 score of 0.83, which is slightly higher than that of the logistic regression model.

```
confusion matrix:
 [[315   7]
 [ 29  90]]
confusion matrix:
 [[0.98 0.02]
 [0.24 0.76]]
              precision    recall  f1-score   support

           0       0.92      0.98      0.95       322
           1       0.93      0.76      0.83       119

    accuracy                           0.92       441
   macro avg       0.92      0.87      0.89       441
weighted avg       0.92      0.92      0.92       441
```

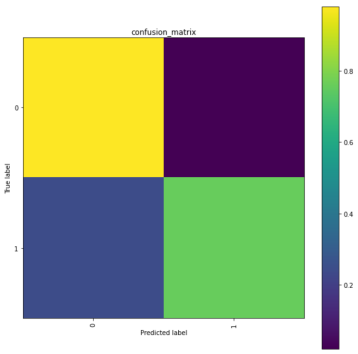


Figure 14: Random Forest Model Accuracy Result(Left) and Confusion Matrix(Right)

Section V.IV: Neural Network Attempt with Tensorflow

Neural networks attempt to take in input data, learn the relationship between them, and produce output data. Inside the neural network, there are different interconnected layers of processing elements, called nodes. The nodes are put into different layers, and different layers are divided into three main types: input layer, hidden layer, and output layer. The input layer takes in data; the hidden layer performs hidden mathematical operations, and the output layer predicts categories. After each layer finishes performing its task, it will push the value forward to the next

layer. In particular, we used the Stochastic Gradient Descent algorithm in tensorflow to classify characters. The F1-scores, accuracy results, and the confusion matrix is attached below.

```
confusion matrix:
 [[0.97 0.03]
 [0.17 0.83]]
              precision    recall  f1-score   support

           0       0.94      0.97      0.95       322
           1       0.91      0.83      0.87       119

    accuracy                           0.93       441
   macro avg       0.92      0.90      0.91       441
weighted avg       0.93      0.93      0.93       441
```

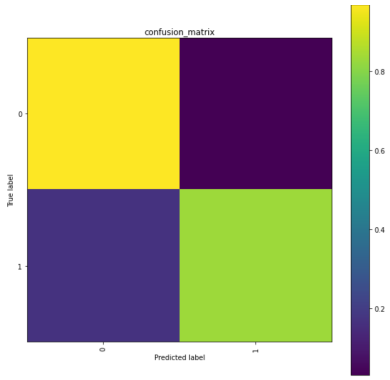


Figure 15: Tensorflow Model Accuracy Result(Left) and Confusion Matrix(Right)

Section VI: Applying Second Data Set

In this section, we introduce a second dataset to test whether our trained models would produce similar, better, or worse results. In this particular dataset, we have included 伍(wu), the character in previous sections that we defined as correct, and four other characters that look different and have different radicals from 伍(wu). The other four characters are: 乘(cheng), 个(ge), 上(shang), and 万(wan). In this case, 伍(wu) is also perceived as correct, and the prediction goal is to separate 伍(wu) from the rest. This data set is significant since the characters are different, and it also serves the purpose of iteration for our model - from accuracy outputs, we can draw conclusions about whether the model could be further applied to other dataset or should be further improved.

Section VI.I: Exploratory Data Analysis on Second Data Set

Similar to the first data set, we first analyze the distribution of average features across different images of characters. Almost all of the plots demonstrate a bimodal distribution for Harris Corner detection features, yet the distributions for Hough transform are quite different. Compared to the first dataset, we hypothesize that this difference in distribution will result in relatively higher F1 scores for the three models.

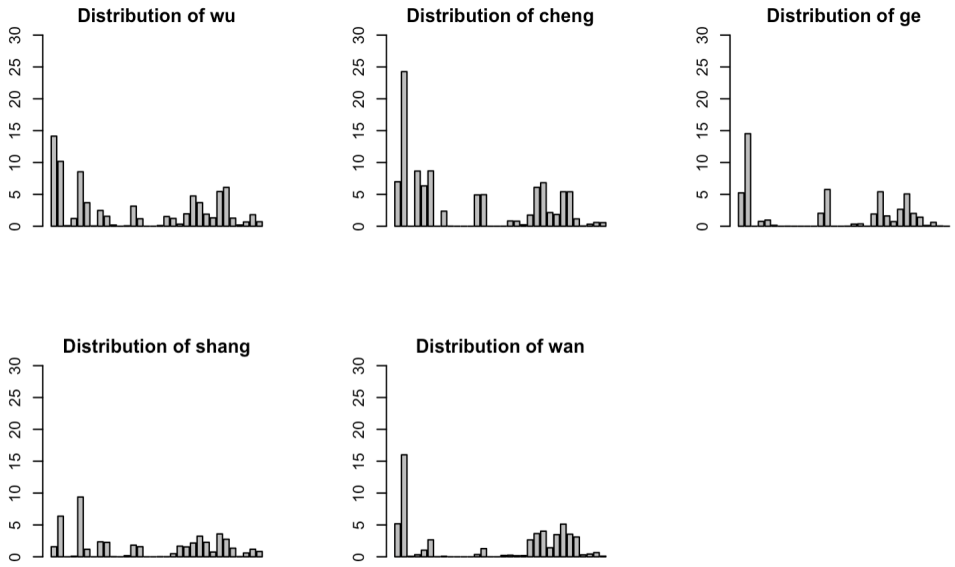


Figure 16: Distribution of Features for Different Characters in Second Dataset

Quantitatively, we could also calculate the pairwise Euclidean distance between each character. Since we mainly want to differentiate 伍(wu) among others, we could look at the first column of the distance matrix. It could be seen that no significant difference in distance could be seen in pairwise distances.

	wu	cheng	ge	shang
cheng	19.223245			
ge	15.167668	17.882111		
shang	14.497211	23.618485	14.399131	
wan	14.481257	16.461285	6.933585	14.555339

Figure 17: Pairwise Euclidean Distance Matrix

Additionally, we analyze the data quality issue of our dataset. Similar to our approach for the first dataset, we focus on analyzing the feature matrix data frame generated from feature selection to spot any potential issues within this feature matrix.

A correlation heatmap is created between pairwise features among all 32 features. Like the previous correlation heatmap, our potential data issue is that three columns (and rows) with completely white fills, columns (and rows) 10,14, 15. Similarly, this is because of transformation in Hough transform where all three features are transformed to 0 across all input characters. Moreover, more zero columns could cause issues when using linear regression models like the logistic regression; either causing prediction accuracy and F1-score to fall or to increase not as much as other models.

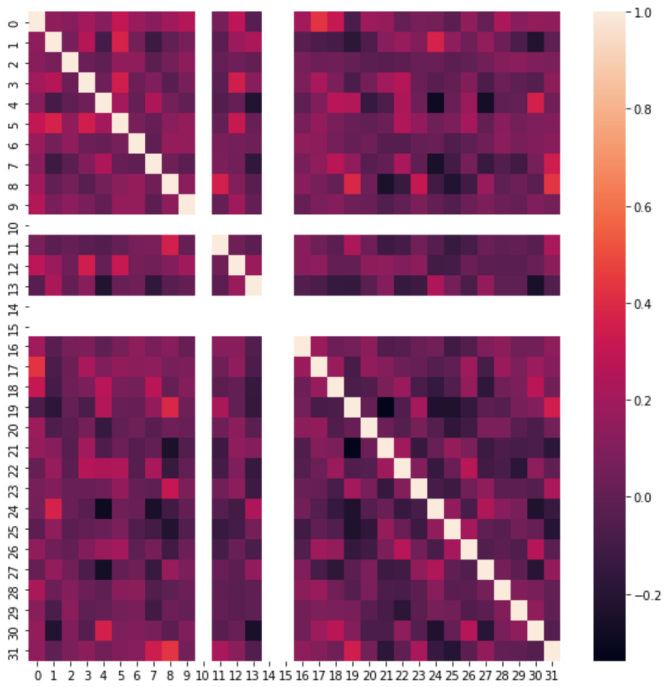


Figure 18: Pairwise Correlation Heatmap of 32 Features in Second Dataset

Section VI.II: Apply Machine Learning Models

We will apply the logistic regression model, random forest algorithm, and neural network to the second dataset and compare F1 score results to corresponding scores for the first dataset. The F1-score for logistic regression model has increased from 0.81 to 0.84. The F1-score for random forest classification from 0.83 to 0.91 where this magnitude of increase is greater than that or the logistic regression. Lastly, the F-1 score for tensorflow increases from 0.87 to 0.88. While this increase is not quite significant, there are improvements when images of different characters are introduced. The detailed accuracy results and confusion matrix are illustrated below.

```
confusion matrix:
 [[296  26]
 [ 14 105]]
confusion matrix:
 [[0.92 0.08]
 [0.12 0.88]]
              precision    recall  f1-score   support

           0       0.95      0.92      0.94       322
           1       0.80      0.88      0.84       119

    accuracy                           0.91       441
   macro avg       0.88      0.90      0.89       441
weighted avg       0.91      0.91      0.91       441
```

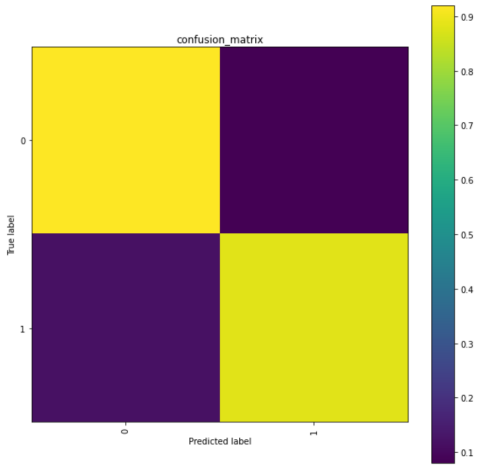


Figure 19: Logistics Regression Prediction Accuracy Results(Left) and Confusion Matrix(Right)

```
confusion matrix:
 [[316   6]
 [ 15 104]]
confusion matrix:
 [[0.98 0.02]
 [0.13 0.87]]
              precision    recall  f1-score   support

           0       0.95      0.98      0.97       322
           1       0.95      0.87      0.91       119

    accuracy                           0.95       441
   macro avg       0.95      0.93      0.94       441
weighted avg       0.95      0.95      0.95       441
```

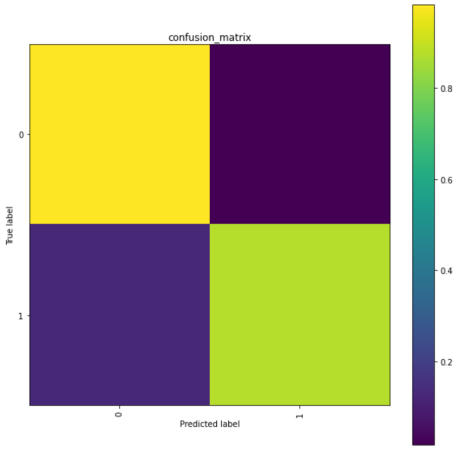


Figure 20: Random Forest Prediction Accuracy Results(Left) and Confusion Matrix (Right)

```
confusion matrix:
 [[0.97 0.03]
 [0.14 0.86]]
              precision    recall  f1-score   support

           0       0.95      0.97      0.96       322
           1       0.90      0.86      0.88       119

    accuracy                           0.94       441
   macro avg       0.93      0.91      0.92       441
weighted avg       0.94      0.94      0.94       441
```

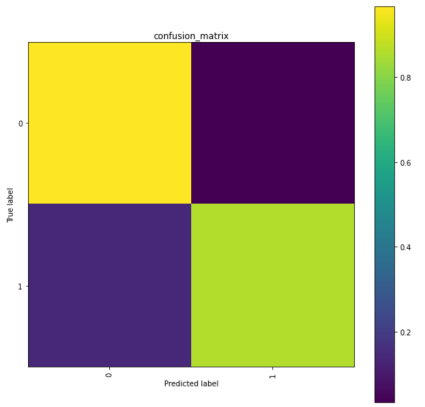


Figure 21: Tensorflow Prediction Accuracy Results(Left) and Confusion Matrix(Right)

Section VI.III: Robustness of Model

Even though all three models display a higher F1 score, this does not mean that our model is robust enough. One characteristic of the second dataset is that all of the five characters look different not only to the human eye but also to the machine after feature extractions. Hence, accuracy could be due to the data instead of the model. Thus, we could not conclude that our model is robust at this point.

Section VII: Data Mining Aspect of Project

This section addresses our data mining goal: why some characters are classified wrong in different machine learning models. From the exploratory data analysis section, we have shown that the distribution of 伍(wu) and 伏(fu) are fairly similar. Calculating the Euclidean distance between those two characters also shows a significantly smaller distance between those two.

In addition, we will apply clustering algorithms to the first dataset. We applied clustering with 5 centers, a table of cluster size and a two-way table of clusters versus characters are shown below. For the first table output, the third cluster has more data points than any single character. (The maximum number of images for one character is around 600.) Further analysis shows that a lot of images of 伍(wu) and 伏(fu) are clustered together in cluster No.3. This follows our initial hypothesis that the machine will have a hard time identifying 伍(wu) and 伏(fu) since their feature distributions are similar. This is also evident in the two-way table, most of the images containing 伍(wu) are clustered in No. 3 as well as most images containing 伏(fu).

```
> table(km_out$cluster)

   1    2    3    4    5
  62  289 1144  451  251
> table(km_out$cluster, characters)
   characters
     fu  ji kang  wu  yi
  1   8  14   11   3  26
  2  31  34  122  62  40
  3 348  48   73 487 188
  4 106  11   33  38 263
  5 109 133    0   5   4
```

Figure 22: Cluster Size (top) and Two-way table(bottom)

Quantitatively, when we calculate the distance between and within each cluster. Distance within each cluster is higher than that of between. We visualized this clustering by first applying Principal Component Analysis (PCA) to change the 32 dimensional feature matrix to 2

dimensions, which is easier to plot and interpret. In addition to graphing with only 5 clusters, we tried clustering with 2,3,4 clusters as well. However, even with only 2 clusters, this algorithm could not identify between 伍(wu) and 伏(fu). 90% of images containing 伍(wu) or 伏(fu) are clustered together. Hence, our hypothesis for incorrect model prediction is probably due to the similar distribution of features between 伍(wu) or 伏(fu).

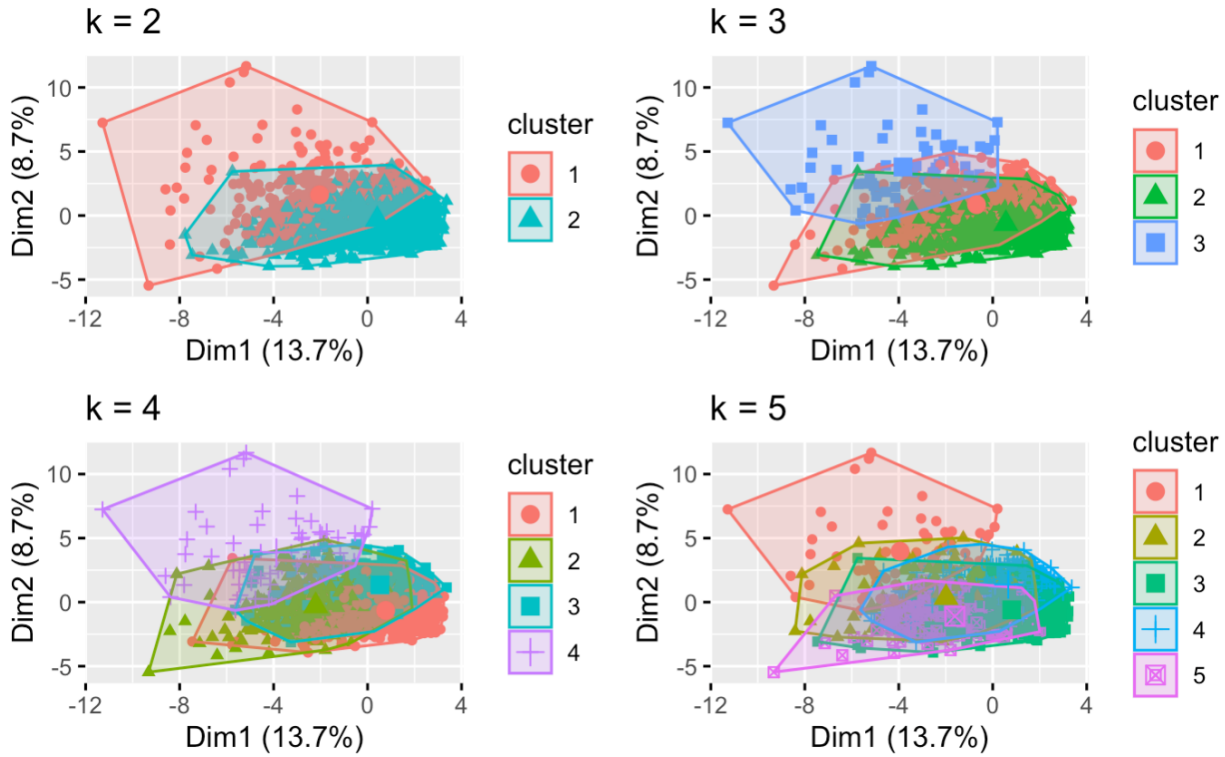


Figure 23: Illustration of Clustering with PCA-reduced dimensions with cluster size = 2,3,4,5

Section VIII: Conclusion

In this project, we used three machine learning models to predict and differentiate between handwritten Chinese characters. Specifically, we used logistic regression, random forest, and an attempt on the neural network TensorFlow. Among the three models, the neural network produces the highest F1-score for both data sets. Yet, this result slightly contradicts our original expectation, since we believed that TensorFlow should produce a much significantly better prediction due to its prevalence in image and character recognition. However, a possible explanation of the discrepancy between expectation and result is that we should not pass in a pre-selected feature matrix into the network. Instead, we should pass in the original 90000-dimensional data, and let the algorithm develop its own feature selection process.

In addition, we also investigated the reasons behind incorrect classifications. For the first dataset, due to the similar distribution of features between 伍(wu) and 伏(fu), the machine perceives those two characters as extremely similar, causing 90% of the images containing those

two characters to be clustered together. It is interesting to note that from the human eye, 伍(wu) is very similar to 伉(kang); however, from our analysis, the machine could separate those two characters. This illustrates the discrepancy in nuance between our chosen feature selection algorithm and the human eye.

Other iterations of our project include an attempt to use Python code to generate characters ourselves, instead of using the NLPR and CASIA dataset mentioned previously. However, due to the lack of enough font styles to provide us with a dataset of sufficient number (this method can only provide us with at most about 50 versions of the same character) and the fact that there are discrepancies between computer generated characters and handwritten characters, we have to give up this method.

A potential data snooping in our project is to train the model with different-looking characters. From the second data set example, we can see that the accuracy and F1-score for different-looking characters are higher than those of the models trained from similar characters. An intuitive choice is to pick the model with higher accuracy or F1-score However, although the result could be statistically significant, it is misleading. Since the final goal of our project is to identify similar characters, we would be passing in test sets or datasets from real-world problems containing similar characters. With models trained from different-looking characters, our prediction and classification for real-world applications would be poor.

A potential extension to this issue is to include more characters in the original training set. Not only should the number of unique characters go up, but the variety should also increase, for example including characters with different radicals, characters without radicals, etc. In the future, we would like to take time to modify our input data, test other feature selection algorithms, or apply neural networks on bigger, more informative datasets to develop a more mature version of our model. For this model, we eventually want to extend to other East Asian languages as well.

Acknowledgements

We would like to acknowledge Wayne T. Lee for project topic selection and recommendations for methodologies and data mining techniques. We would also like to express our greatest appreciation for Yunhan Liu who took the time to proofread our report and provided editing suggestions. Lastly, we would like to thank Ohad Klopman's group for project ideas and critique.

Reference

[1]
Textor, C.. "Number of students at elementary schools in China between 2010 and 2020" *Statista.* 29 November 2021. Access Date 7 May 2022 https://www.statista.com/statistics/227034/number-of-students-at-elementary-schools-in-china/#:~:text=Number%20of%20students%20at%20elementary%20schools%20in%20China%202010%2D2020&text=In%202020%2C%20around%20107.25%20million%20students%20attended%20elementary%20school.

[2]
“2021年全国高考报名人数1078万”. *Ministry of Education of the People's Republic of China.* 03 June 2021. Access Date 7 May 2022. http://www.moe.gov.cn/jyb_xwfb/xw_zt/moe_357/2021/2021_zt12/meiti/202106/t20210603_535277.html#:~:text=%E6%9C%AC%E6%8A%A5%E5%8C%97%E4%BA%AC6%E6%9C%88,%E6%AF%94%E5%8E%BB%E5%B9%B4%E5%A2%9E%E5%8A%A07%E4%B8%87%E3%80%82 (Chinese source)

[3]
Khandelwal, Neetika. "Image Processing in Python: Algorithms, Tools, and Methods You Should Know”. *Neptuneblog*. 17 January 2022. Access Date 7 May 2022. https://neptune.ai/blog/image-processing-python#:~:text=Image%20processing%20allows%20us%20to,programming%20languages%20for%20this%20purpose

[4]
Liu, Hui. *Rail transit collaborative robot systems*. Chapter 3.3.1.2.6 Grayscale and Gray Image. Elsevier, 2020. Access Date 7 May 2022. https://www.sciencedirect.com/book/9780128229682/robot-systems-for-rail-transit-applications#book-description

[5]
Kunchala, Pavan. "Why Grayscale the images in computer vision?” *Medium*. 7 January 2021. Access Date 7 May 2022. https://medium.com/analytics-vidhya/why-grayscale-the-images-in-computer-vision-936091b734d5

[6]
Sharda, Aryaman. "Understanding Gaussian Blur Filters". *Medium*. 24 January 2021. Access Date 7 May 2022. https://aryamansharda.medium.com/image-filters-gaussian-blur-eb36db6781b1

[7]
Korting, Thales Sehn. “How Hough Transform Works”. *YouTube*. 16 April 2016. Access Date 7 May 2022.
https://www.youtube.com/watch?v=4zHbI-fFIlI

[8]
Saini, Anshul. "An Introduction to Random Forest Algorithm for Beginners". *Analytics Vidhya*. 19 Oct. 2021. Access Date 7 May 2022.
https://www.analyticsvidhya.com/blog/2021/10/an-introduction-to-random-forest-algorithm-for-beginners/

[9]
Tyagi, Deepanshu. "Introduction to Harris Corner Detector". *Medium*. 16 March 2019. Access Date 7 May 2022.
https://medium.com/data-breach/introduction-to-harris-corner-detector-32a88850b3f6

[10]
"K-means Cluster Analysis”. *UC Business Analytics R Programming Guide*. Access Date 7 May 2022. https://uc-r.github.io/kmeans_clustering.

[11]
"Chapter 4 Exploratory Data Analysis with Unsupervised Machine Learning | Computational Genomics with R." https://compgenomr.github.io/book/unsupervisedLearning.html