Classifying Styles of Handwriting with Neural Networks

Patrick Ancel

**Abstract**

The intent of this project is to capture some degree of stylistic data from images of handwritten text. Rather than asking what is written, it essentially asks what “font” or style the text is written in. No prior knowledge of the classes is assumed ahead of time, so they are created by converting a list of images into vectors and performing K-means clustering on them. Each cluster is arbitrarily labelled by an integer from zero to (K - 1). In order to generalize the results of this clustering to more samples of text, an artificial neural network is employed. Four-fifths of the images (along with their corresponding class labels) are randomly selected to be used as training data for the neural network, and the remaining fifth are used to test the trained model. The code for the project is written in the Python programming language so that it can utilize powerful third-party packages – in particular, Pillow, NumPy, and TensorFlow. Pillow is used to read and process the input images, NumPy to perform calculations on arrays, and TensorFlow to build, train, and evaluate artificial neural networks. The code was tested five times on 403 images of the number eight. On each run, the images were clustered into three classes, and the results were used to train and test nine different neural networks. The results of the clustering were unreliable, being too dependent on the vectors selected to be the initial means. The neural networks, however, performed generally well, with some having mean accuracies exceeding 90%.

**Introduction**

While this tool could be used to determine the identity of an unknown symbol (i.e., finding out *what* is written), the interest of this project is in capturing stylistic data that would otherwise be lost when converting handwritten text into pure ASCII or Unicode characters. In other words, the goal isn’t to determine what character is presented, but rather to determine the font that best represents it. Ultimately, this model could be used alongside another model to convert pages of handwritten text into documents in a two-layered system. The first layer reads a portion of the image data and determines which character it sees. Then, it passes the image along with the character identification to the second layer which decides the font type to represent the character as. The second layer is the subject of this project.

**Related Works**

In [1], handwriting recognition was improved by classifying the writing style before determining what was written. Multiple Hidden-Markov models were used, each specific to a particular writing style to improve the ability of the system to recognize written text. The overall method involved two steps: classifying the style first and reading the text second. The project in this paper proposes a conceptually similar two-step system, but with the layers in the reverse order of [1]. Here, a single character is held fixed, assumed to be already known, and the style classification is performed for the individual symbol.

The research in [2] attempted to create a model that could classify handwritten text as legible, illegible, or somewhere in between. While this project is not concerned with those specific classes, what is of interest are the methods used to extract 36 features from image contours. In future work, such features may be more useful than the arrays of lightness values that are used in this project.

**Methodology**

To apply the K-means clustering algorithm to the list of images, it is necessary to convert the data to a form in which the distance between two images can be calculated. For this project, the images are represented as very high-dimensional vectors, and they are clustered according to their Euclidean distances. Each image is divided into an array of equally sized squares, and each square represents one coordinate of the vector. The value of a coordinate is calculated by taking the average lightness of every pixel contained within its square region; this value is then scaled so that it ranges between 0 and 1. The intuition behind using the average lightness is that it represents how much ink is present or absent from that square section of the paper. The larger the square regions, the more information is lost from the original image. Figure 1 shows a visual representation of this compression on three square images.

A picture containing crossword, clock

Description automatically generatedThe remaining specification is the method of flattening the array of lightness values into a one-dimensional vector. This is done by arranging the rows one after the other, starting from the top and ending at the bottom. As long as the method of ordering the coordinates is applied consistently to every image, the specifics of doing so are arbitrary, due to the fact that each coordinate describes a position along its own dimension. An unfortunate result of this is that the positions of the square sections relative to each other cannot affect the outcome of the clustering algorithm. One would likely assume that squares which are positioned near each other in the image should be related in some way, but as far as the calculations are concerned, it doesn’t matter whether two squares are adjacent to each other or on opposite sides of the image, since they are represented as magnitudes along orthogonal dimensions.

Figure 1 This is the visual result of compressing 150x150 images using blocks of size 15x15.

The training and evaluation images were taken from a dataset on Kaggle [3]. From the dataset, 403 images of the number eight were taken, all of which had the dimensions of 155 by 135 pixels. These images were processed by the code five times with varying results. The square sections were 5 pixels wide, and the number of clusters was set to 3. On each of the five runs, after the clustering was performed, the image vectors, along with their class labels, were split into training data and evaluation data. In all rounds, 322 vectors were used for training and 81 were used for evaluation, meaning about 20.0993% of the data was used to test the neural network. For each round, nine different neural networks were trained and tested. The results are displayed in table 2.

**Results**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 |
| Cluster 0 Size | 161 | 176 | 1 | 193 | 37 |
| Cluster 1 Size | 145 | 126 | 149 | 153 | 158 |
| Cluster 2 Size | 97 | 101 | 253 | 57 | 208 |
| Number of Iterations | 16 | 13 | 11 | 18 | 16 |

Table 1 Results of Performing K-Means Clustering on 403 Image Vectors.

The results of the clustering algorithm, summarized in table 1, are sensitive to the vectors that are initially selected to be cluster means. This is most apparent from round 3, the only round in which any cluster had only a single element. This could be a fault in the clustering algorithm, the method by which images are converted into vectors, or the sample size. The cluster labels have no meaning between different runs, making it difficult to measure the consistency of the results across different runs. Nonetheless, it appears that the vectors are not partitioned in the same way reliably across different attempts.

|  |  |  |  |
| --- | --- | --- | --- |
| Hidden Layers | Nodes per Layer | Mean Accuracy | Standard Deviation |
| 1 | **64** | 0.923456788 | 0.03425786 |
| 2 | **64** | 0.925925922 | 0.048604981 |
| 3 | **64** | 0.839506185 | 0.120330802 |
| 1 | **128** | 0.901234567 | 0.023096637 |
| 2 | **128** | 0.913580251 | 0.035993529 |
| 3 | **128** | 0.930864191 | 0.020658283 |
| 1 | **256** | 0.829629636 | 0.113955376 |
| 2 | **256** | 0.888888884 | 0.041866227 |
| 3 | **256** | 0.896296299 | 0.087992726 |

Table 2 Results of the Neural Networks Across all Five Runs.

Evaluating the overall performance of this model is a tricky task, as there are several components of the process that can be examined, some of which are subjective. The simplest measure is the accuracy of the best performing type of neural network. The configuration with three hidden layers and 128 neurons per layer has the highest mean accuracy, which is approximately 93%. However, this only provides a measure of the ability for the neural network to predict the classes defined by the K-means algorithm. It does not indicate whether the K-means algorithm did a good job clustering the data in the first place. The effectiveness of the clustering is questionable due to the inconsistencies from run to run, but even if it had yielded consistent results, determining its usefulness is difficult. The K-means algorithm was chosen for the very reason that no prior knowledge of the nature of the samples (including their similarities and differences from each other) was assumed. Therefore, no such prior knowledge can be utilized to evaluate the quality of the clusters.

**Conclusion**

In future work. critiquing and refining the vector conversions and clustering methods will likely take two approaches. One of them is by subjective visual inspection. The developer should visually examine the clusters and determine if they are satisfactory. The second approach is theoretical. As discussed earlier, when an image is converted to a vector, the relative positions of the square sections to each other are completely disregarded. In fact, a better method would capture such relationships between squares as their distance and direction from one another. Even if this insufficiency causes no apparent issues when clustering the training and testing data, it could result in performance issues when the final neural network is applied to other data. This would affect users down the line, despite the clusters passing visual inspection by the developer. Once these issues are resolved at a theoretical level, the developer can have more confidence that the algorithm captures the essence of the input samples, and he or she can relax and trust that the program is doing its job according to the objective data.

**References**

1. G.A. Fink and T. Plötz, “Unsupervised Estimation of Writing Style Models for Improved Unconstrained Off-line Handwriting Recognition.” in *Tenth International Workshop on Frontiers in Handwriting Recognition,* Université de Rennes,La Baule, France, Oct. 1, 2006. Available: https://hal.inria.fr/inria-00103797v2
2. M.E. Dehkordi, N. Sherkat, and T. Allen, “Handwriting style classification,” *IJDAR*, 6, pp. 55-74, June 2003, doi: https://doi.org/10.1007/s10032-003-0101-4.
3. C. Zhao, “Handwritten math symbol and digit dataset.” Distributed by Kaggle. Available: https://www.kaggle.com/clarencezhao/handwritten-math-symbol-dataset (accessed Aug. 5th, 2020).

Source code for the project available at: https://github.com/PatrickAncel/predictingHandwritingStyle