Classifying Handwritten Characters via Multilayer Perceptron and K-Nearest Neighbor Machine Learning Algorithms

University of Florida

Ding-Shin Kuo, Carlos Salinas, Jenny Tong, Morgan Urdaneta

***Abstract*—Written language revolutionized human society by providing a substrate for transmission and preservation of knowledge. One of the most quintessential written formats is handwriting. Its practicality, uniqueness, and ease of use makes this format, one of the most widely used written schemes. Hence, even in the modern world, handwriting is still prevalent in a plethora of documents such as postal service labels, bank checks and old archives. Notwithstanding, in some cases, handwritten text has several pitfalls, including: illegibility, physical source damage, and difficulties transporting and editing the information. One potential way to overcome this issue is to transcribe handwritten language into digital format. Nevertheless, manually transcribing handwritten to digital is a daunting task. Here, we apply a machine-learning classification algorithm to specific features of a handwritten dataset to determine the corresponding label of a handwritten character. In order to test the efficacy of our training algorithms, we converted the input data into training and testing datasets via cross-validation. Using this approach, we systematically tested the components of our machine learning pipeline: preprocessing tools, extracted features (HOG and Hu moments) and ML algorithms (KNN and MLP). After several iterations of this process, our algorithm reliably predicted slightly above 95% accuracy the labels of the class data set.**

Keywords—KNN, MLP, ML, HOG features, Hu moments, sklearn.

I. Introduction

With the advent of written language approximately five millennia ago, a paradigm shift in the way knowledge was transferred across generations occurred [1]. Prior to the invention of written language, the only way to convey information was through spoken language, a non-efficient, transient, and subjective communication method [Chafe et al, 1987].

The invention of the printing press and in the 16th century and the recent massification of digital media contributed to a continuous shift of handwritten language as the main form of written language [2]. This transition allowed humankind to effectively transmit knowledge to the masses and has been linked to an exponential increase in scientific discoveries. Digital written language offers the advantages such as space-saving, legibility, and accessibility [*Gee et al, 2011*].

Even though digitization has reduced the amount of handwritten text used nowadays, some important documents remain in this handwritten format, such as postal service labels and bank checks. Being able to transcribe handwritten text into digital can potentially improve legibility, decrease human error, and increase productivity [3].

Nevertheless, manually transcribing handwritten characters could be a very time-consuming task. In order to overcome this issue, machine learning and pattern recognition algorithms could be applied [4]. Machine learning (ML) uses statistical methods to allow computers to progressively predict outcomes without being explicitly programmed [Nasrabadi, 2007].

In order to train the ML algorithm for handwritten character recognition, features from a previously-preprocessed character must be extracted, once these features are extracted, a classification algorithm is implemented. Two common classification algorithms are K-Nearest Neighbor (KNN) [5] and Multilayer Perceptron (MLP) [6]. Here, in order to classify handwritten characters from a class dataset. Individual handwritten characters were pre-processed (flattened and resized) and specific features were posteriorly extracted. Namely, occurrences of gradient orientation per character site through a histogram of gradient outcome (HOG) [7] and weighted invariant moments (average) of the character [Hu, 1962].

These features were then used to systematically train two KNN and MLP algorithms via cross-validation. After defining the set of parameters that provided the most accurate results, namely, KNN for characters ‘a’ and ‘b’ and MLP for all . we proceeded to optimize specific parameters such as number of neighbors, number of hidden layers, activations functions. Our methodology and ML implementation allowed reliably predict handwritten characters with high success.

II. Implementation

The implementation of our machine learning algorithm to classify handwritten characters from a dataset consisted on systematically cross-validating different pre-processing and feature extraction parameters with a simple ML algorithm. Namely, K-Nearest neighbor. In order to achieve this goal, we extracted the input data (*ClassData*) with its respective labels (*ClassLabels*) (Fig. 1A). Posteriorly, we preprocessed each character by flattening (more on this) and resizing to equally-sized matrices of 50 by 50 pixels. Two different approaches were used to do the resizing of the images according to their initial dimensions. If the image was smaller than 50 by 50 pixels, we padded zeros at the end vertically and horizontally. If the any of the images had one or both dimensions larger than 50 pixels, the image was scaled down instead of being cropped. Cropping might get rid of important features that we could have needed to train our ML model.

Although scaling down the images reduced their quality, it still allowed us to extract features (Fig. 1B). Once each input character had the same dimensions, the histogram of gradient outcomes, as well as, the Hu moments were extracted (Fig. 1C). In order to test the performance from the extracted features we split the data into training and testing datasets. We compared the accuracy of KNN classifier and MLP classifier of each feature independently. From these results, we focused on the features that gave us better accuracy.

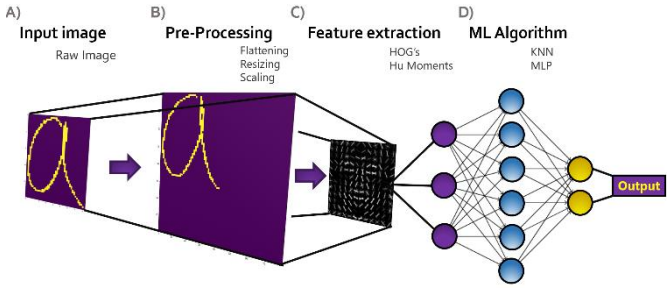

Figure 1 Implementation Pipeline of the handwritten character machine learning classification algorithm. A) The input data consisted of raw images of the class dataset. B) Each individual character from the input dataset was pre-processed to optimize feature extraction. C) Occurrences of gradient orientation per character site through a histogram of gradient outcome (HOG) and weighted invariant Hu moments of each character were extracted as our features. D) K-nearest neighbor and Multilayer perceptron were used throughout multiple parameter optimizations to determine the most accurate classifier via cross-validation.

A. *Features*

a) Hu Moments

One of the challenges in pattern analysis is the recognition of objects or characters regardless of their position, size and orientation in an image. A possible solution to deal with this problem is the utilization of moments -weighted average of pixels that hold statistical properties about the pixels. Normalized moments, Hu's moments, have the advantage of being invariant to object scale, position and orientation and can be calculated using algebraic invariants. [8]. In our implementation, the Hu's moments were computed using *Open-CV*. From there, we wrote our own functions to compute the Hu's moments. The equations we used in the computation of Hu's seven moments are:

$$
\begin{aligned}
M_1 &= (\eta_{20} + \eta_{02}) \\
M_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11} \\
M_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - 3\eta_{03})^2 \\
M_4 &= (\eta_{30} + 3\eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
M_5 &= (\eta_{30} + 3\eta_{12})\,(\eta_{30} + \eta_{12})[(\eta_{30} + 3\eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\
&\quad +(3\eta_{21} - \eta_{03})\,(\eta_{21} + \eta_{03})\,[3(\eta_{30} + 3\eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
M_6 &= (\eta_{20} - \eta_{02})\,[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}\,(\eta_{30} + \eta_{12})\,(\eta_{21} + \eta_{03}) \\
M_7 &= (3\eta_{21} + \eta_{03})\,(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\
&\quad -(\eta_{30} + 3\eta_{12})\,(\eta_{21} + \eta_{03})\,[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]
\end{aligned}
$$

where η_{pq} is the normalized central moment of order (p+q).

b) HOG Features

Another solution to the position, size, and orientation-variance problem is using the histogram of oriented gradients (HOG). The basic idea is that local object appearance and shape can often be characterized rather well by the distribution of local intensity gradients or edge directions. [9]

In the first stage of feature extraction using HOG, first order image gradients are computed to capture contour information by binning each pixel in the image into cells. Each pixel in the cell contributes a weighted vote on the orientation of the gradient and a histogram channel is generated per cell. Then, cells are grouped into larger, spatially connected blocks to locally normalize the gradient strengths, accounting for illumination and contrast. Block normalization can be done in four different ways:

L2-norm: $f = \frac{v}{\sqrt{/|v|_2^2 + e^2}}$

L2-hys: L2-norm followed by clipping of v

L1-norm: $f = \frac{v}{(||v||_1 + e)}$

L1-sqrt: $f = \sqrt{\frac{v}{(||v||_1 + e)}}$

The HOG features are extracted using the *hog* function provided in the *skimage.feature* package with L1-norm block normalization.

B. *Models*

a) K-Nearest Neighbor Classifier

The K-Nearest Neighbor (KNN) classifier is used to predict or formulate ideas by comparing how similar new points in the data space are to the data points known by the KNN model. Even though this model is computationally costly, we decided that KNN would a viable model due to ease of use. We implemented the KNN classifier to calculate the distance between our feature vectors. The classifier predicts the unknown data by locating the most common label among the *k* closest neighbors. Due to the large computation time of using a self-made KNN classifier and predicter, we utilized the KNN classifier from the *sklearn.neighbors* package.

b) Multilayer Perceptor Classifier

The architecture of MLP networks is similar to that of a brain where cells or neurons communicate to one another back and forth to convey information and obtain a result. In our project, the preceptors communicate to back and forth, forward and back propagation, among them to see to what characters the input features correspond to. The extracted features from the images are several so implementing our own model from scratch given our time constraints wasn't feasible; our model needed over 200 neurons in the input layer and 8 neurons in the output layer, and with at least one hidden layer, the creation of such network would have been be a project of its own. We opted to use the MLP classifier from the *sklearn.nerual_network* package. This classifier, at each step, calculates the partial derivative of the loss function with respect to the model parameters to update the weights and achieve better accuracy. We

also used a regularization term to avoid overfitting. The model used minibatches for optimization as well. It also does one thousand iterations. Since the model takes some time to finish training, the *early_stopping* condition was set so that the model finish when there is no error improvement margin greater than 0.001 after 10 iterations

III. Experiments

In order to label handwritten characters from the class dataset, we applied a systematic methodology that consisted on a series of validation experiments for each step throughout our implementation pipeline (Fig 1). Incidentally, the methodology applied herein (pre-processing, feature extraction, and classification) coincides with a commonly used experimental approach in the field of ML [4]. The set *ClassData* contains a total of 6192 samples of the following characters: 'a', 'b', 'c', 'd', 'h', 'i', 'j' and 'k'. *ClassLabels* contains numbers 1 to 8 corresponding to each of the characters in *ClassData*.

A. Training and Testing K-Nearest Neighbor Classifier

After pre-processing the images, we obtained HOG features (225 features per image). To determine the best HOG parameters to use, we altered the pixels per cell (PPC) and cells per block (CPB) as shown in Figure 2A and Figure 2B respectively. We varied the PPC value and the CPB value and compared the resulting predicted labels to the target values to calculate the accuracy. We determined the best combination of PPC and CPB parameter is (10, 10) and (5, 5), respectively. In testing the value of PPC and CPB, the values were jointly varied, keeping the product of the two to be less than 50 (the maximum number of pixels per side of the image). K was kept constant at 1.

Utilizing the *train_test_split* function from *sklearn.model_selection* package, we split the features accompanied with their target values into 66% training data and 33% validation data. We then fit the KNN model using the training HOG features as training data and the respective labels as our target values. To test, we also varied the *K* nearest neighbor value and compared the resulting accuracy as shown in Figure 2C. The best *K* parameter after training and validation is found to be 1. In testing, *K* was varied from 1 to 14 and PPC and CPB were kept constant at (10, 10) and (5, 5), respectively. We found with increasing *K*, the accuracy decreased since the excess neighbors that were casting a vote skewed the vote towards the wrong class.

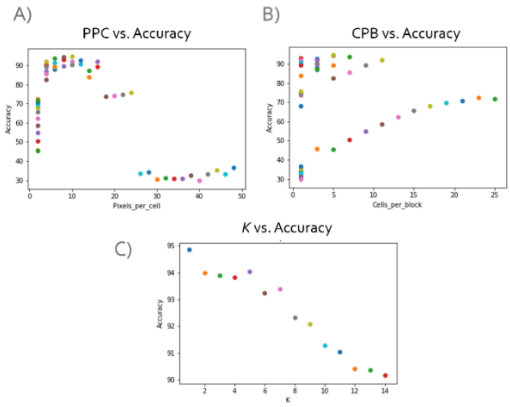

Figure 2 Comparison of KNN classifier accuracy A) with varying PPC. B) with varying CPB. C) with varying *K*.

B. Traning and Testing the Multi-Layer Perceptron Classifier

From the pre-processing of the images, we obtained HOG features (225 features per image) and Hu moments (7 features per image). We ran these features through a simple MLP model of four hidden layers with the HOG features as inputs and an MLP model of 2 hidden layers with the Hu moments as inputs. The number of hidden layers was chosen based on the following rule

$$\#\,h\_layers(\#\,features) = \begin{cases} 2, & features \leq 100 \\ 3, & features \leq 150 \\ 4, & features \leq 200 \\ 5, & anything\ else \end{cases}$$

The number of neurons in each hidden layer matched the number of features we were working with. i.e. 225 and 7 respectively for each MLP model. We compared the accuracy of both models. Given that the accuracy obtained from the HOG features was much higher than that of the Hu moments (Fig. 3A), we opted to optimize the model for HOG features. *ClassData* and *ClassLabels* were split into a training data set ,again, with 66% of the data from *ClassData* and *ClassLabels* and a testing data set with 33% of the data.

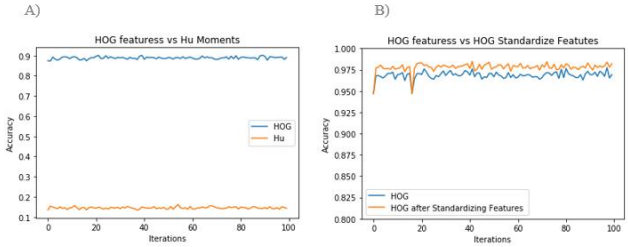

Figure 3 A) comparison of results using HOG features and Hu moments. B) Increase of performance of MLP model after standardizing the HOG features

The first optimization we did was to standardize the HOG features by removing the mean and scaling to unit variance. The results of the optimization are shown in Fig. 3B.

We implemented distinct activation functions (Fig 4A), number of iterations (Fig 4B), numbers of hidden layers (Fig 4C), and numbers of neurons per hidden layer (Fig 4D) to

determine optimal MLP parameters to classify the handwritten dataset. A confusion matrix with the highest performance MLP parameter set was plotted to visually illustrate the performance of our algorithm in classifying the full dataset (Fig 4E).

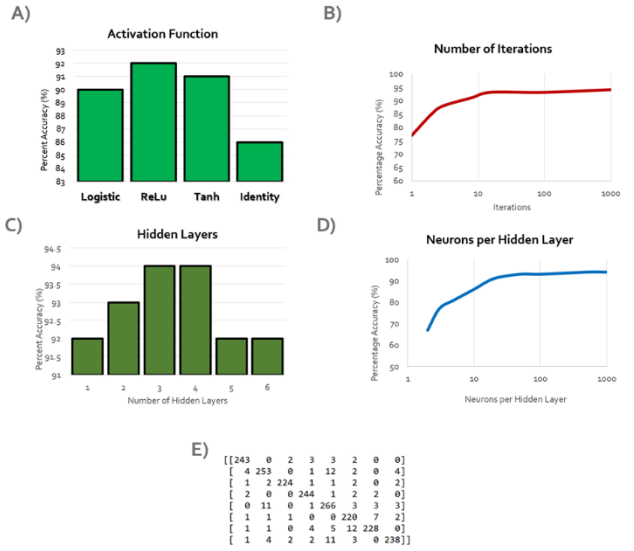

Figure 4. Determining optimal parameters for the multilayer perceptron classifier. A) classification accuracy of distinct activation functions. B) numbers of iterations; C) number of hidden layers; D) Numbers of neurons per hidden layer. E) Confusion matrix with the highest performance MLP parameters illustrating the performance of our algorithm in classifying the full dataset.

It is imperative to clarify that only one parameter was changed at time while keeping the others constant.

IV. Conclusions

When extracting the HOG features, several parameters can be varied, including the number of pixels per cell (PPC) and cells per block (CPB). Through experimentation, we found that a PPC of (10, 10) and CPB of (5, 5) returned the optimal number of features without resulting in over-complexity of the model since increasing the number of features increases the number of inputs into the model.

HOG features turn out to be one a strong feature extraction technique to classify characters, with accuracy scores about 90% for all *ClassData* set.

With the KNN model and a *K* of 1, an accuracy score of 96.6% is achieved on the *ClassData* dataset when classifying only a and b. The accuracy decreases when scored against the full dataset with all letters (a, b, c, d, h, i, j, and k) due to having more variance in the data.

For the MLP model, although the Hu moments are invariant to position, size and orientation, they still lack in information to accurately classify characters. The accuracy of our model to identify the characters is as low as 13%. It is clear that more features are needed to obtain better results.

Increasing the number of neurons in each hidden layer, as well as the number of hidden layers increased the overall accuracy to 95%. However, having more than a hundred neurons in a hidden layer didn't increase the performance significantly. Also increasing too much the number of hidden layers reduced the overall accuracy

The best score accuracy was given with four hidden layers in the MLP model. However, our final model implemented five layers because it gave us a better accuracy identifying and classifying letters 'a, and 'b'.

The overall accuracy scores between the two models were similar; however, the MLP model performed a little bit better with a score of 97 % when classifying and labeling characters 'a' and 'b'.

References

[1] W. Chafe, J. Danielewicz, *"Properties of spoken and written language".* Academic Press, 1987.

[2] J. P. Gee and E. Hayes, "*Language and learning in the digital age". Routledge*.London, 2011.

[3] A. Toselli, V. Romero, L. Rodriguez and E. Vidal, *"Computer Assisted Transcription of Handwritten Text Images,"* Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), Parana, 2007, pp. 944-948.

[4] N.M. Nasrabadi, "Pattern recognition and machine learning," Journal of Electronic Imaging, vol. 16, pp. 049901, 2007

[5] *K.* Beyer, J. Goldstein, R. Ramakrishnan and U. Shaft, "*When is "nearest neighbor" meaningful?"* in International conference on database theory, pp. 217-235, 1999

[6] Ruck, Dennis W., et al. *"The multilayer perceptron as an approximation to a Bayes optimal discriminant function."* IEEE Transactions on Neural Networks 1.4 (1990): 296-298.

[7] Rodriguez, Jose A., and Florent Perronnin. *"Local gradient histogram features for word spotting in unconstrained handwritten documents." Proc. 1st ICFHR* (2008): 7-12.

[8] Ming Kuei Hu, "*Visual Pattern Recognition by Moment Invariants"* , IRETransactions on Information Theory, Vol. 8, issue 2, pp. 179 –187, 1962

[9] N. Dalal and B. Triggs, "*Histograms of oriented gradients for human detection*," 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 2005, pp. 886-893 vol. 1. doi: 10.1109/CVPR.2005.177