Classification model for handwritten digit recognition

Classification of handwritten digits into their respective labels using PCA-KNN-SVM techniques

Anujay Ghosh, Shefali Mahadik
Department of Computer Engineering
Atharva College of Engineering
Malad(W), Mumbai, India
anujay.ghosh@gmail.com

Abstract— **In this work, an amalgamation of two classification techniques to recognize the handwritten digit input which is Support Vector Machine and K – Nearest Neighbors have been used to classify handwritten digits into their respective labels. Handwritten Digit Recognition is the competence of a machine to receive, calculate and decipher a human handwritten input from sources such as handwritten manuscripts and other paperwork, especially created before the advent of a digital revolution and images.**

Keywords—handwritten digit recognition; support vector machine; k nearest neighbors; principal component analysis; classification; supervised learning; machine learning

I. Introduction

Handwritten digit recognition is the proficiency of a machine to receive handwritten digit images as input from sources such as handwritten physical documents, images as well as other sources. There are many devices now such as smartphones and tablets that can take handwriting as an input to a touch screen via a finger or using an electronic stylus. This is useful as it allows the user to quickly write down numbers and text to the devices, especially for the selective audience which isn't well versed with input devices such as keyboards and hence type slowly but can write words with their hands way faster. There are many applications for handwriting recognition available in effect today. There are numerous techniques that have been developed to recognize handwritten digits. This work provides a rudimentary yet unique solution for the task.

A. Need of the System

Handwritten Digit Recognition system is developed to improve on the accuracy of the existing solutions to achieve higher accuracy and reliable performance. Identifying handwritten digits have various real world applications. There are various practical problems where this recognition system is very beneficial such analysis of handwritten documents, address interpretation on letters and mails, bank cheque processing, zip code detection.

B. Scope of the System

The rise of a digital new world, although a revolution in its own field, is not a bubble that excludes everything else in the world. The digital world's advent began a mere century or two ago, but scriptures and books after books have been handwritten by human scholars from the beginning of mankind. Accepting the digital world first begins with the task of integrating the scripts that came into existence before the rise of computers and technology. Thus, this conversion and integration must begin with the most common values in the world that transcend different languages as well - numbers. This work of Handwritten Digit Recognition can help us with the first step of this process and can later be expanded to include letters of English as well as other diverse languages. It can also be used in different authentication systems where verification of the manually written digits can be processed into digital digits. This system can be useful in various pattern recognition tasks as well.

C. Applications

Handwritten Digit Recognition system can be used as a part of bigger projects in various fields such as Handwritten to Digital Character Conversion, Meaning Translation, Content Based Image Retrieval, Keyword Spotting, Pattern Recognition, Signboard Translation, Text-to-Speech Conversion, Scene Image Analysis, etc.

II. Problem statement

The problem was to classify handwritten digits with accuracy as high as possible. The goal was to take an image of a handwritten digit, and correctly determine what that digit was. The digits range from zero (0) to nine (9). In this work, we looked into the Support Vector Machines(SVMs), Principle Component Analysis(PCA) Transformation and K-Nearest Neighbour (KNN) techniques, combining them all to form a unique method to solve the problem. The problem used a labelled dataset which is available on Kaggle, taken from the famous MNIST (Modified National Institute of Standards and Technology) dataset. So according to this data set the input is given and outputs the likelihood that the image belongs to different classes (i.e. the machine-encoded digits, 0-9).

III. Requirement analysis

A. Hardware Requirements

- Intel i-3 or later Processor: The system we designed will be used by people having minimum of Intel i-3 processor. This is because strong cores and processors are essential for processing large data and affect the performance of the model.

- Minimum 2 GB RAM: Minimum of 2 GB RAM is required for user to access the system smoothly.

- Minimum 2 GB Graphics Processing Unit: This will help with processing of large dataset containing thousands of digits in image or pixel data format.

B. Software Requirements

- Operating System (Windows 7 and above): System will be applicable for the Windows operating system 7 and above like Windows 8, Windows 8.1, Windows 10 or its Linux equivalent.

- Anaconda Python: this software is open source and is the easiest way to perform Python and machine learning on a Windows system. It also enables to develop and train machine learning and deep learning models with scikit-learn, analyze data with scalability and mathematical functions with NumPy, import the dataset using pandas and visualize results with MatPlotLib.

- Python 3.7: This version of Python is used to run the program. It is the base on top of which Anaconda performs its tasks.

IV. Design of the model

A. Control Flow

Control flow describes the sequence in which individual statements, instructions or function calls of our model were executed or evaluated. An ellipse denotes beginning or end for the model, a rounded rectangle denotes a set of instructions/program code/function code encapsulated within a single process and a diamond denotes a decision/conditional statement with two or more outcomes. The Figure 4.1 alongside shows the control flow diagram that our model uses.

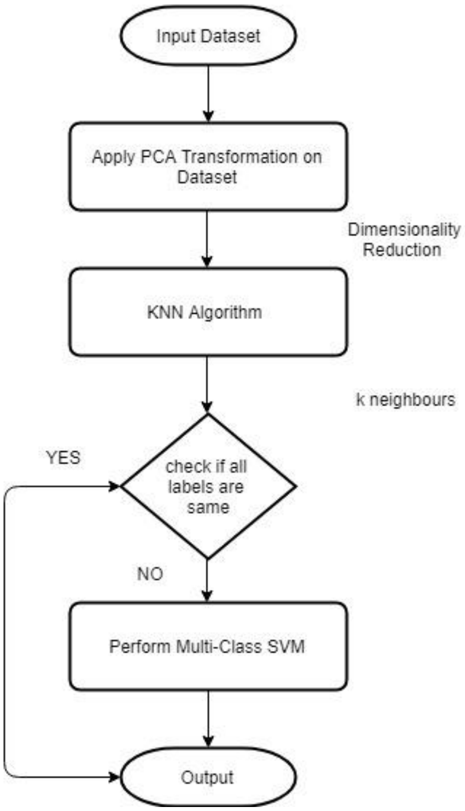


Figure 4.1: Control Flow Diagram

B. Sequence Diagram

A Sequence diagram is an interaction diagram that shows how processes operate with one another and in what order with respect to time flow. The Figure 4.2 below shows object interactions arranged in time sequence and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. It gives us a pictorial representation of how the end user may interact with the program and the sequence of actions that will take place along with the level of the machine it will occur in.

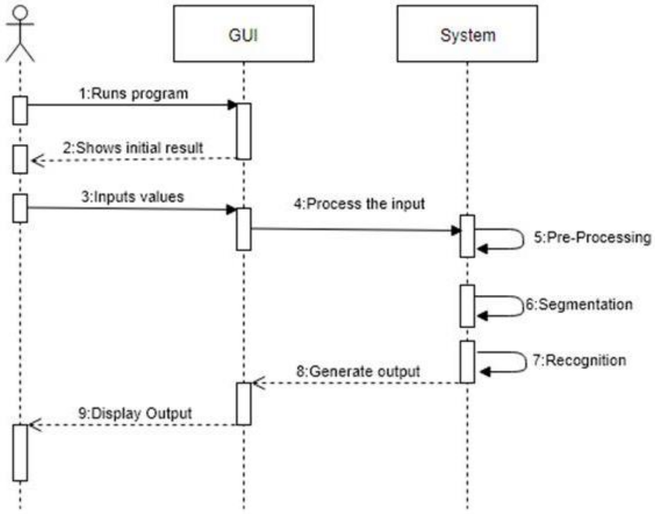


Figure 4.2: Sequence Diagram

V. Implementation

A. *MNIST Dataset Details*

This system is based on Python and sklearn to run classification and read the dataset. It uses MNIST dataset for training and evaluation for classification. MNIST problem is a dataset for evaluating machine learning models on the handwritten digit classification problem. The dataset was constructed from a number of scanned document dataset available from the National Institute of Standards and Technology (NIST). Each image in this dataset is a 28 by 28 pixel square (748 pixels total). There are close to 70000 images in the combined dataset that can be used for training and evaluate the system. In this proposed system two (2) classification algorithm are used for the recognition which is Support Vector Machine (SVM) and K-Nearest Neighbor (KNN).

B. *Model Implementation*

The implementation of our work is as follows:

We first import all the necessary libraries namely NumPy, MatPlotLib, Time and packages in the sklearn library such as model_selection, SVM, KNeighborsClassifier, PCA and other necessary packages as and when required.

We then load the dataset and separate the feature columns with target columns. Since the original dimension size of the dataset is quite large (784 input features), the dimensionality reduction becomes necessary. First step is to extract the principal components from the original data.

We do this by performing PCA Transformation for dimensionality reduction to extract out the principal components in the initial large dimensional features of the data. As shown in Figure 5.1, the first 314 principal components can interpret approximately 95% of total information (in terms of total variance retained), which suffice to be representative of the information in the original dataset. Thus implementing the first 314 principal components as the extracted features

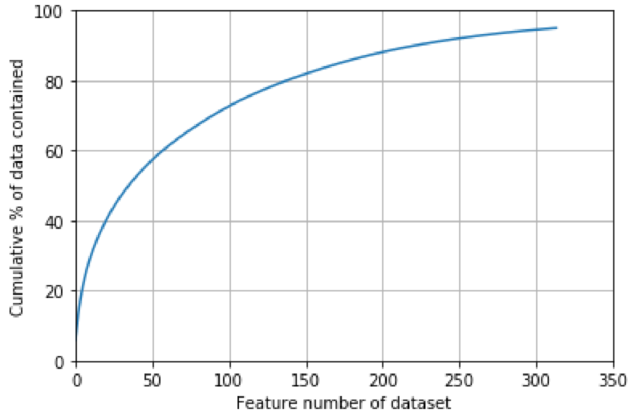


Figure 5.1: The amount of data v/s component number

The data is then split into training and testing sets using StratifiedShuffleSplit function.

The KNN model is then created and fit to the training set values which trains the KNN classifier.

For a query, we then compute the Euclidean distances of the query to all the training samples and pick the K-nearest neighbors. The general value of Euclidean distance(d) is calculated using (5.1)

$$d(p,q) = d(q,p) = \sqrt{(q_1-p_1)^2+(q_2-p_2)^2+..+(q_n-p_n)^2} = \sqrt{\sum_{i=1}^{n}(q_i-p_i)^2} \qquad (5.1)$$

where,

p = first data point

q= second data point

n = number of dimensions in a data point.

If the K-neighbors (excluding the query) all have the same labels, the query is labeled the same as its neighbors and exit.

Otherwise, it computes the pairwise distances between the K neighbors, converts the distance matrix into a kernel matrix and a multiclass Support Vector Machine is applied to it, to finally label the query using the classifier.

In the initial implementation, we extract 314 principal components and use parameters values of k=2 for KNN and C=0.5 for SVM. This resulted in an accuracy score of 0.964.

TABLE I. Initial Test

k	C	Prediction Time (test)	Accuracy (train)	Accuracy (test)
2	0.5	51.002	1.0	0.964

Then a trial and error approach is used to tune the k parameter by changing its value while keeping the other parameter values the same and observing the results. We perform it for 20 distinct values of k (number of neighbors), keeping the c (penalty parameter) value constant.

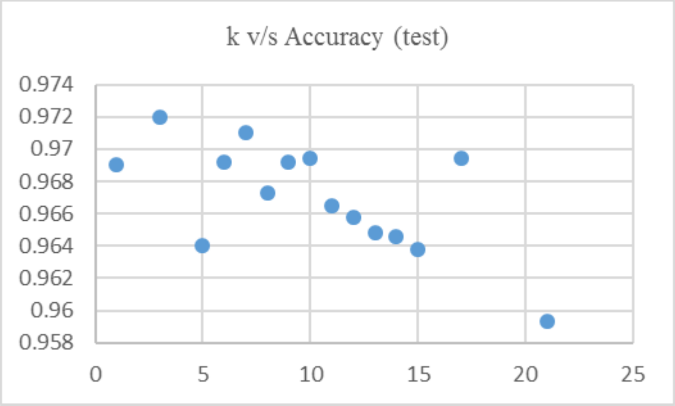
TABLE II. Changing k values with C=0.5 to observe accuracy

k	C	Prediction Time (test)	Accuracy (train)	Accuracy (test)
5	0.5	42.3s	1.0	0.9640
3	**0.5**	**54.9s**	**1.0**	**0.9720**
1	0.5	55.6s	1.0	0.9690
7	0.5	54.8s	1.0	0.9710
8	0.5	65.5s	1.0	0.9673

9	0.5	66.5s	**1.0**	0.9692
10	0.5	62.2s	1.0	0.9694
11	0.5	69.9s	1.0	0.9665
12	0.5	68.8s	1.0	0.9658
13	0.5	70.1s	**1.0**	0.9648
14	0.5	70.6s	1.0	0.9646
15	0.5	71.6s	1.0	0.9638
6	0.5	61.8s	1.0	0.9692
17	0.5	68.6s	**1.0**	0.9694
21	0.5	76.9s	1.0	0.9593

The Graph 5.1 below plots accuracy of test set with respect to changing k values taken from Table II above.

Graph 5.1: k v/s Accuracy(test)



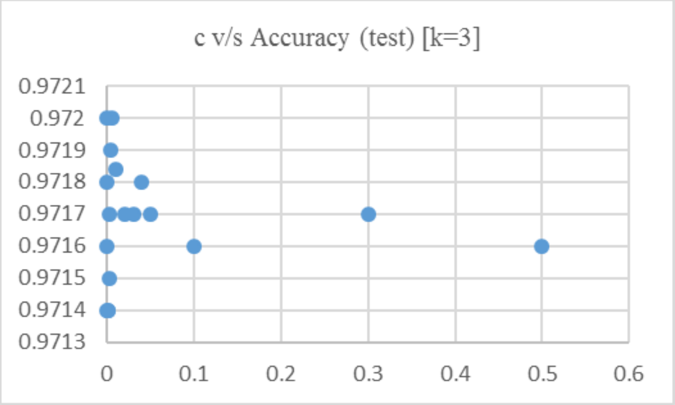
We clearly see that k=3 gives the highest accuracy, hence we keep k=3 constant and change values of c to understand variation in accuracy with change in c as follows:

TABLE III. CHANGING C VALUES WITH K=3 TO OBSERVE ACCURACY

k	C	Prediction Time (test)	Accuracy (train)	Accuracy (test)
3	0.01	54.9	1.0	0.9718
3	0.02	55.2	1.0	0.9717
3	0.03	55.1	1.0	0.9717
3	0.04	55.3	1.0	0.9718
3	0.05	55.2	1.0	0.9717
3	0.001	55.0	1.0	0.9714
3	0.002	61.9	1.0	0.9717
3	0.003	54.9	1.0	0.9715
3	0.004	54.9	1.0	0.9719
3	**0.005**	**54.58**	**1.0**	**0.9720**
3	0.1	55.14	1.0	0.9716
3	0.3	54.89	1.0	0.9717
3	0.5	55.28	1.0	0.9716
3	0.0001	55.35	1.0	0.9714
3	0.0002	55.95	1.0	0.9718

The Graph 5.2 below plots accuracy of test set with respect to changing C values taken from Table III above.

Graph 5.2: Accuracy(test) with respect to C



We understand that changing values of see provides negligible change in accuracy.

Thus we conclude that the best value of k is k=3. However, changes in the C value do not impact the final accuracy score. This result is quite unusual because the input space to the SVM is very small (size 3) and the SVM algorithm can classify the dataset pretty quickly hence changing the parameters does not have much effect on the accuracy. The final solution then uses k=3, C=0.005, and yields and accuracy score of 0.9720 meaning that the model can accurately recognize close to 97% of the handwritten digits.

VI. RESULT AND ANALYSIS

A. Dataset Analysis

Digits dataset has 70000 image samples (42000 training set and 28000 testing set samples with 784 features each)

Figure 6.1 represents the number of occurrences of all the digits (i.e. 0-9) present in the training dataset of 42000 samples.

```
Out[4]: Text(0, 0.5, 'number of occurrences')
```

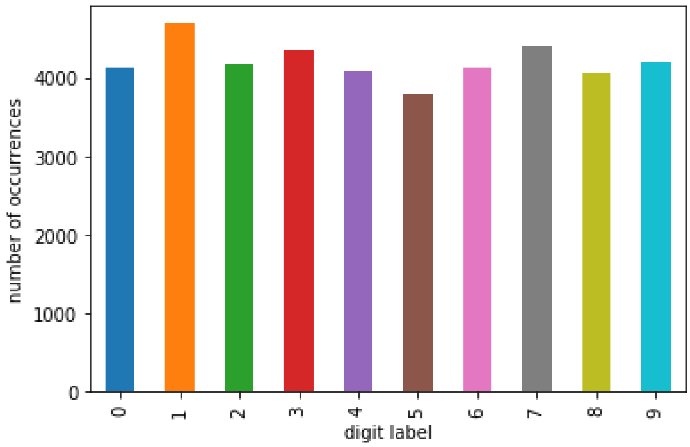
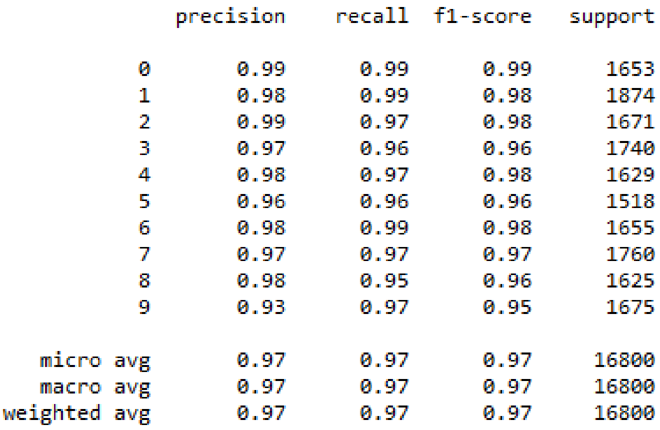


Figure 6.1: Occurrence of each digit in the training set

B. *Classification Report*

Figure 6.2 below displays the extensive classification report containing details about the precision of the model, recall, f1-score and support, hence overall denoting the accuracy of the predicted values for every digit. It also displays the micro, macro and weighted average of all four result outcomes obtained by the model while predictions.



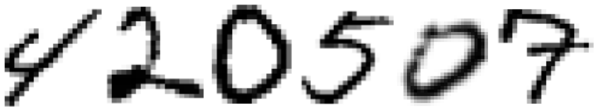
```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1653
           1       0.98      0.99      0.98      1874
           2       0.99      0.97      0.98      1671
           3       0.97      0.96      0.96      1740
           4       0.98      0.97      0.98      1629
           5       0.96      0.96      0.96      1518
           6       0.98      0.99      0.98      1655
           7       0.97      0.97      0.97      1760
           8       0.98      0.95      0.96      1625
           9       0.93      0.97      0.95      1675

   micro avg       0.97      0.97      0.97     16800
   macro avg       0.97      0.97      0.97     16800
weighted avg       0.97      0.97      0.97     16800
```

Figure 6.2: Classification Report

C. *Manual Result Testing*

By manually taking out digits from the data set, plotting their 28px by 28px square image using imshow function in matplotlib, and comparing the results with the predicted outcome, we get the following:

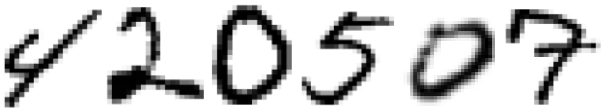
The actual images and their labels are as show in Figure 6.3



4 2 0 5 0 7

Figure 6.3: Actual images with their true labels

The same images were fed to the model and the model's prediction was as show in Figure 6.4 below.



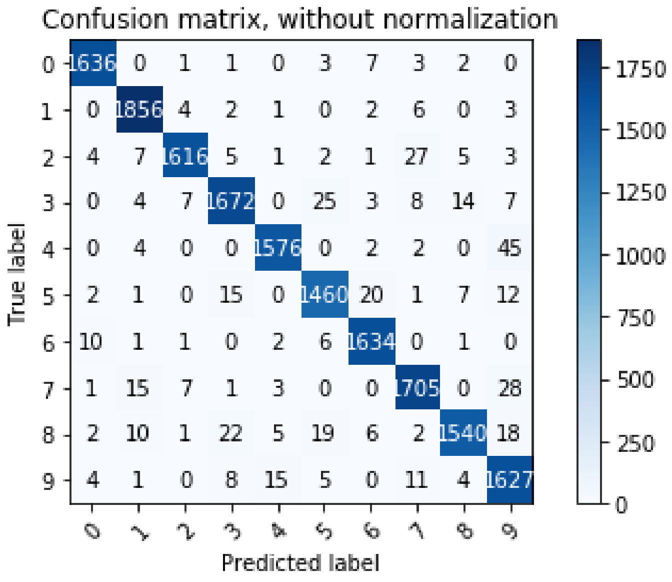
4 2 0 5 9 7

Figure 6.4: Images with their predicted labels

This model incorrectly labels the fourth image and identifies it as a 0, but the correct label is 9. This is interesting because the shape of the top part of number 9 is similar to the shape of 0 which could be the cause of confusion for this model to be able to distinguish the two.

D. *Confusion Matrix*

Figure 6.3 below visualizes the confusion matrix graphically for us to understand it better. It plots the predicted values versus actual values where the actual labels are represented on Y-axis and predicted values are represented on X-axis. This model has been applied to the testing dataset. The model predicted the label to be 0 correctly 1636 times. We must also note that the model predicted true label 0 to be predicted 6, ten times. It might be because 0 and 6 are very similar looking digits when written by hand and so on and so forth.



	0	1	2	3	4	5	6	7	8	9
0	1636	0	1	1	0	3	7	3	2	0
1	0	1856	4	2	1	0	2	6	0	3
2	4	7	1616	5	1	2	1	27	5	3
3	0	4	7	1672	0	25	3	8	14	7
4	0	4	0	0	1576	0	2	2	0	45
5	2	1	0	15	0	1460	20	1	7	12
6	10	1	1	0	2	6	1634	0	1	0
7	1	15	7	1	3	0	0	1705	0	28
8	2	10	1	22	5	19	6	2	1540	18
9	4	1	0	8	15	5	0	11	4	1627

Figure 6.3: Confusion matrix, without normalization

VII. Conclusion

Thus, we conclude that the model is acceptable for providing a solution for classifying handwritten digits into their respective labels in the MNIST dataset as it is able to accurately categorize well with accuracy quite close to humans using a combination of classification techniques Support Vector Machine and K-Nearest Neighbors.

However, the model is still in its rudimentary stages, useful in a limited domain. Some changes would have to be made to solve a bigger problem of recognizing multiple digits in an image or recognizing arbitrary multi-digit text in unconstrained natural images.

Acknowledgment

It gives us great pleasure in presenting this work titled: *'Classification model for handwritten digit recognition'*.

On this momentous occasion, we wish to express our immense gratitude to the range of people who provided invaluable support in the completion of this system. Their guidance and encouragement has helped in making this system a great success.

We express our gratitude to our guide Prof. Aruna Pavate, who provided us with all the guidance and encouragement and making the lab available to us at any time.
We would also like to acknowledge our classmates Gauri Shenoy, Vidit Gholam and Mahendra Patel for their support with the project work.
We would like to deeply express our sincere gratitude to our respected Principal Dr. Srikanth Kallurkar and the management of Atharva College of Engineering for providing such an ideal atmosphere with well-equipped libraries with all the utmost necessary reference materials and up-to-date computer laboratories.
We are extremely thankful to all staff and the management of the college for providing us all the facilities and resources required.

References

The template will number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first ..."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

[1] G. Eason, B. Noble, and I.N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529-551, April 1955. (*references*)

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[3] I.S. Jacobs and C.P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G.T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271-350.

[4] K. Elissa, "Title of paper if known," unpublished.

[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740-741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.