Available online at sjuoz.uoz.edu.krd

Science Journal of University of Zakho

Vol. 8, No. 2, pp. 66– 71, June-2020

p-ISSN: 2663-628X
e-ISSN: 2663-6298

FONT RECOGNITION OF ENGLISH LETTERS BASED ON DISTANCE PROFILE FEATURES

Aveen J. Mohammed [a], *, Hasan S.M. Al-Khaffaf [a]

[a] Dept. of Computer Science. of Science, University of Duhok, Duhok, Kurdistan Region, Iraq – (avin.kovli@gmail.com; hasan.salim@uod.ac)

Received: Jan., 2020 / Accepted: Apr., 2020 / Published: Jun.,2020 ***https://doi.org/10.25271/sjuoz.2020.8.2.694***

ABSTRACT:
This paper presents a system for recognizing English fonts from character images. The distance profile is the feature of choice used in this paper. The system extracts a vector of 106 features and feeds it into a support vector machine (SVM) classifier with a radial basis function (RBF) kernel. The experiment is divided into three phases. In the first phase, the system trains the SVM with different Gamma and C parameters. In the second phase, the validation phase, we validate and select the pair of Gamma and C values that yield the best recognition rates. In the final phase, the testing phase, the images are tested and the recognition rate is reported. Experimental results based on 27,620 characters glyph images from three English fonts show a 94.82% overall recognition rate.

KEYWORDS: distance profile features, support vector machines, English font recognition, character font classification, optical font recognition.

1. INTRODUCTION

Optical character recognition (OCR) is used to convert text image into a text document suitable for searching and editing. However, OCR is not able to retrieve typographical properties (i.e. font information) and at the same time, its job is more difficult because of the variation of typographical attributes of printed texts. A font is a graphical representation of text. Optical font recognition (OFR) is one of the primary functions in document recognition and analysis which aims at recovering typographical attributes of printed text (Doermann, & Tombre, 2014). When OFR used as a pre-processing step before OCR, it has a substantial impact on the optical character recognition accuracy rate. OFR can also be used as a post-processing step to support creating a document that looks like the original document in terms of text shape and font. For document reconstruction, font information can be used. An OCR and OFR systems together will help in recreating digital documents that have search ability while visually resemble the original.
In general, font recognition systems can be of two types in terms of data entry levels: single-level and multiple levels. Where in single-level the image will be in four kinds: (text block, text line, word or character) level; only one of the four levels is used in the system. Text block-level can be called in other names like text page, or multiple lines of text. On the other hand, in multiple levels the image could be in two or more levels; such as a block of text with word level. Knowledge of the font may be useful for identifying its logical label, such as chapter title, section title, and paragraph (Al-Khaffaf, & Musa, 2018).
In this work, support vector machines (SVMs) and distance profile features (DP) are going to be used for character level font recognition of the English language. This work differs from others in three areas. First, we used a larger feature vector to grasp more detail of each font compared to the study of (Bharath, & Rani, 2017). Second, we performed a validation operation to reveal the proper model parameters' values to be used in the recognition phase. Third, we performed normalization of data. As will be shown in the next sections, the first and third points mentioned above raise the recognition rate of the proposed system while the second point raises confidence about the results. Figure 1 shows a diagram of the system. Solid-line shapes are only used during the training phase. Dashed- and solid-line boxes are used during validation and testing.
In the next section, related works are presented. In section 3, the algorithms that are used in the methodology are presented. The dataset that has been used in this paper is explained in section 4. In section 5, the classification algorithm is illustrated. In section 6, the results of applying SVM are shown. Finally, conclusions are presented in section 7.

2. RELATED WORKS

Many studies have been published on font recognition. In (Hajiannezhad, & Mozaffari, 2012) the authors presented a method based on a directional fractal dimension, for recognizing font of three languages (Farsi, Arabic, and English) text images. It considers the extracted features are independent of document content. In the feature extraction step, the Variogram method was used which is a directional fractal dimension method that is an area of fractal geometry. Each sample is expressed by a 6D feature vector. The system can recognize 10 different fonts for each of Arabic and Farsi, and 8 fonts for the English language. For Arabic and English, ALPH-REGIM datasets were used and for Farsi, they created their own dataset. The RBF and k-nearest neighborhood

* Corresponding author

classifier (KNN) are used to classify the font. The average recognition rates using RBF, and KNN classifiers are respectively 95.5%, 96% for Farsi fonts, and 96.9%, 98.84% for Arabic fonts, and 98.21%, 99.6% for English fonts.

The system developed by Bharath and Rani in (Bharath, & Rani, 2017) has three steps; in the first step, the system reads a character image and pre-processes it. In the second step, distance profile features are computed. In total, 74 features per character image are generated. In the last step, a support vector machine (SVM) and *k* nearest neighbors (KNN) are used for classification. Five different font styles are tested, and the system achieves approximately 80% average accuracy using SVM and 75% using KNN.

In 2017, Jaiem, Slimane, and Kherallah presented a font recognition system based on steerable pyramids called Arabic font recognition steerable pyramids (AFR/SP) (Jaiem, 2017). The system uses three levels of text entity analysis: word, line and text block. In feature extraction, steerable pyramids and two statistical variables (standard deviation and mean) are used. This study used two databases, Arabic Printed Text Image Database / Multi-Font (APTID/MF) and Arabic Printed Text Image (APTI) with different resolutions. In the classification phase, the BPANN is used. The experimental results for high-resolution text block samples show high recognition rates of approximately 99% and 93% for low-resolution; for text line and word levels in low-resolution, the recognition rates are 90.67% and 78.98% respectively.

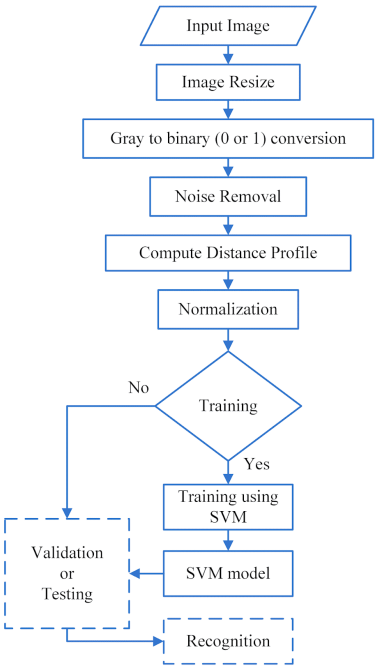

Figure 1. Diagram of the proposed system.

Tensmeyer, Saunders, and Martinez presented a system that classifies font of two levels: text line and text page by using Convolutional Neural Networks (CNNs) framework (Tensmeyer, 2017). For line level, King Fahd University Arabic Font Database (KAFD) datasets used to recognize 40 Arabic fonts; while for page level, two datasets are used KAFD and Latin Medival Manuscripts (CLaMM) database that classifies 12 English scripts. The author splits the dataset into three sets: training, validation and testing set; which helps to choose the most accurate model. For comparison purposes, the authors used two CNN architectures: AlexNet architecture that has five convolution layers, and the state-of-the-art ResNet-50. In the training, two models for each of ResNets and AlexNet created from KAFD database on line-level and the page level; while in CLaMM database one model for each of both architectures created. In the validation, ResNets architecture had better performance for both datasets. In the testing, the recognition rate for line-level was better than page level of 98.8%, while page-level obtained 86.6%.

(Bui, 2015) developed a system that extracts Gradient orientation features for boundaries of the character image to recognize its font. The authors obtained a bounding box of character images via a series of morphological filtering and thresholding operations. They started by scaling the image. From the edges of the character, a chain code sets are created. Next, a histogram of oriented gradient (HOG) descriptors set are extracted of size 3×3. By using 8 bins quantization level, frequency histograms of gradient orientation are calculated and normalized to get 72-dimensional HOG descriptor. For each character, about 30-40 descriptors are obtained. Hierarchical k-means clustering on a random sub-sample of 10% of the HOG descriptors which yields quantizing centers of k cluster performed. Then, a hard-assignment Bag of Visual Words (BoVW) is applied. KNN used to assign each descriptor to k codewords (cluster canter). For each character in training, a frequency histogram is calculated by computing the codewords. Finally, Stop-words are specified. A dataset of 1000 Latin fonts, that consists of 5 samples for each of the capital letters (A-Z) only. The authors collected 4 million HOG descriptors. In the classification logistic regression (LR) with multi-class is used and a one-vs-all combination strategy. The accuracy of recognition 93.4% achieved.

3. METHODOLOGY

3.1 PREPROCESSING AND FEATURE EXTRACTION

During the pre-processing, the image is prepared for the feature extraction step. The pre-processing step is very important because it reduces data and removes noise. All samples of the dataset are degraded PNG images of the size (102×102). Each image is shrunk by 50% in both directions to (51×51) to minimize processing time. The shrink ratio is selected as 50% to keep the image dimension linearly proportional to the original in order to reduce further degradation of image quality. Next, pixel values are converted to binary with a value of 0 or 255, where black pixels are represented as a value of 0 and white background pixels are represented as a value of 255. Then, noise is removed using a median filter with a mask of size 5*5. Figure 2 shows an example result of pre-processing operations.

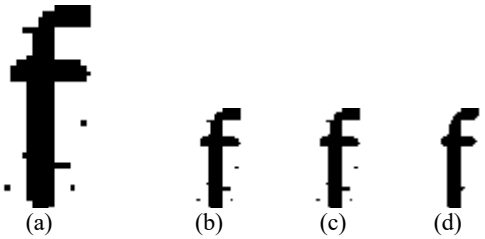

Figure 2. Results of pre-processing step. (a) Sample image. (b) Shrinked image. (c) Conversion to binary. (d) Noise removed.

After the pre-processing step, the image is fed to the next step, which is feature extraction, where 106 distance profile features are extracted; these features consist of the left distance profile (LDP) of 51 features, right distance profile (RDP) of 51 features, top left corner (TLD), top right corner (TRD), bottom left corner (BLD), and bottom right corner (BRD). Then, these features are normalized before the training phase.

3.1.1 Left Distance Profile Features (LDPs) and Right Distance Profile Features (RDPs)

The left distance profile feature is computed by finding the distance between the first column of the input character image and the first outer border pixel (Bharath, & Rani, 2017). For each row in the image, the number of white pixels is computed until the first black pixel appears; then, it goes to the next row, and the same process is performed on each row from left to right. Equation 1 shows how to compute the left distance profile features (see Figure 3(a)). The feature vector is of size 51 because each image has 51 rows.

$$\text{LDP(i)} = \sum_{j=0}^{n-1} P(i,j) \quad (1)$$

where LDP(i) is the i*th* left distance profile feature, P(i,j) is the background pixel intensity, while i, j are row and column number respectively, and n is the value of the first foreground pixel.

For the right distance profile, a similar process is performed, but this time, it moves from the right to the left outer border pixel. Equation 2 shows how to compute the left distance profile features (see Figure 3(b)).

$$\text{RDP(i)} = \sum_{j=m}^{n-1} P(i,j) \quad (2)$$

where RDP(i) is the i*th* right distance profile feature, P(i,j) is the background pixel intensity, while i, j are row and column numbers respectively, m is the value of the last column where the value of m is decremented until it reaches (n-1), and n is the value of the first foreground pixel.

(a) (b)

Figure 3. Distance profile features. (a) left distance profile features and (b) right distance profile features.

3.1.2 Diagonal Distance Profile (DDF) Features

As their name indicates (see Figure 4), DDF features counts white pixels in the main- and anti-diagonals. In the main diagonal the top left corner and bottom right corner are calculated. For the top left corner; $p(i,j), p(i+1,j+1), p(i+2,j+2) \ldots p(i+k,j+k)$ are summed (refer to Eq. 3). For the bottom right corner; the sum of $p(m,n)$, $p(m-1,n-1), p(m-2,n-2), \ldots\ldots, p(m-k,n-k)$ is used in (4). In the anti-diagonal, the top right corner and bottom left corner are calculated in (5) and (6), respectively. For the top right corner; $p(i,n)$, $p(i+1,n-1), p(i+2,n-2) \ldots\ldots p(i+k,n-k)$ are summed. For the bottom left corner; $p(m,j)$, $p(m-1,j+1), p(m-2,j+2), \ldots, p(m-k,j+k)$ are summed.

$$\text{TLD} = \sum_{i=0}^{m} P(i,i) \quad (3)$$

$$\text{BRD} = \sum_{i=m}^{0} P(i,i) \quad (4)$$

$$\text{TRD} = \sum_{i=0}^{n} P(i,n-i) \quad (5)$$

$$\text{BLD} = \sum_{j=0}^{m} P(m-j,j) \quad (6)$$

where TLD is the top-left main diagonal feature, TRD is the top right secondary diagonal feature, BLD is the bottom left secondary diagonal feature, BRD is the bottom right main diagonal feature, and as mentioned before i, j are the row and column value respectively.

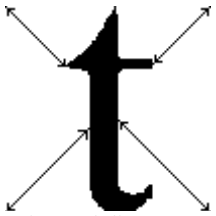

Figure 4. Diagonal distance profile features.

4. DATASET

The dataset that has been used in this paper is taken from (Al-Khaffaf, 2012), an electronic book composed of 60 pages. In the next statements, we explain how the dataset was generated by (Al-Khaffaf, 2012). The book was available in three fonts: Comic Sans MS (Comic), DejaVu Sans Condensed (DejaVu), and Times New Roman (Times). Page images were feed to the OCRopus and Decapod open-source software to segment the images into character images of PNG file formats. In the second step, the entire image samples are degraded by Kanungo et al. algorithm (Kanungo, 2000). The output of that process was an isolated degraded character image of size (102×102) pixel with 8bpp. Only the first three pages and the last three pages of the book are used in the experiments of this paper. Figure 5 shows samples of letters a, b, c, d, e, f, and g of the three fonts.

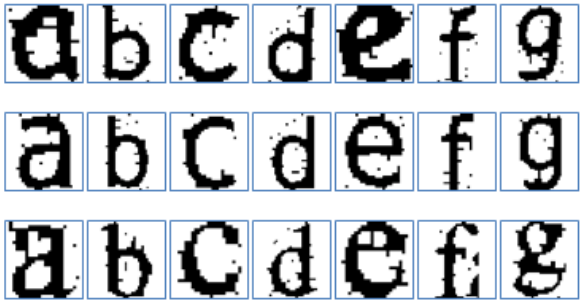

Figure 5. Samples of degraded image of letters (a, b, c, d, e, f, and g). Top – Comic, Middle – DejaVu, Bottom – Times.

5. THE SVM CLASSIFICATION METHOD

For classification, SVM, a supervised machine learning algorithm is used in classification. The SVM was originally a binary (2-class) classification method developed in 1995 by Vapnik and Cortes (Cortes, 1995), see Figure 6. An SVM can classify both linear and nonlinear data for finding the best hyper-plane that has the maximum margin between support points. In the system SVM with RBF kernel is used to identify the font of the English isolated character image. Because our data are nonlinearly separable we used a feature transformation (kernels), the basic idea is to transform the data from a nonlinear space into a linear space (Xu, 2006) by mapping input vectors $\overline{x} \in R^n$ into vectors $\Phi(\overline{x})$:

$$\overline{x} \in R^n \rightarrow \Phi(\overline{x}) = [\Phi(\overline{x}_1), \Phi(\overline{x}_2) \ldots \Phi(\overline{x}_n)] \in R^f$$

In the linearly separable case, we have Wolfe dual Lagrangian function:

$$L(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \overline{x}_i . \overline{x}_j \quad (7)$$

where m is the number of training samples, $\overline{x}_i$ is the i*th* training vector, y_i is the class of that training vector, and α_i is the positive Lagrange multipliers (Kowalczyk, 2017), the value of

a training example $\overline{x}$ is not used; only the dot product $(\overline{x}_i.\overline{x}_j)$ between two training examples are used. Therefore, this product is replaced with the scalar product $K(\overline{x}_i,\overline{x}_j)$ of the respective kernel functions and it will return the same results as in linear space. The kernel function computes the inner product between two projected vectors (Scholkopf, 1997):

$$K(\overline{x}_i,\overline{x}_j) = \{\Phi(\overline{x}_i),\Phi(\overline{x}_j)\}$$

$$L(\alpha)=\sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j K(\overline{x}_i,\overline{x}_j) \quad (8)$$

The soft-margin dual problem can be written:

$$\max_{\alpha} \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j K(\overline{x}_i,\overline{x}_j)$$
$$\text{s.t. } 0 \le \alpha_i \le C,\ \forall i \text{ and } \sum_{i=1}^{m}\alpha_i y_i = 0$$

where C is the penalty parameter (Kecman, 2005). A high C value tries to classify every training data point correctly; on the other hand, a low value of C ensures a smooth decision surface.

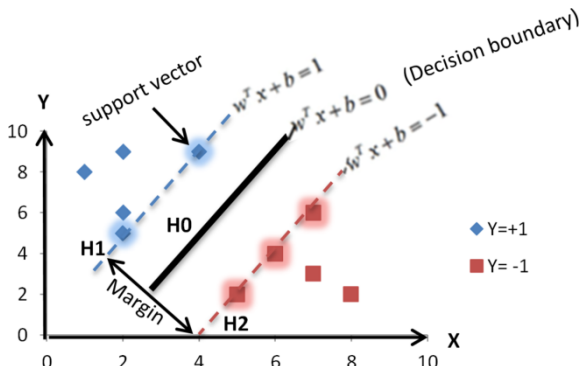

Figure 6.SVM classification. Taken from (Katiyar, 2017).

The hypothesis function for predicting the class of the test image is as follows:

$$y(\overline{x}_i) = \text{sign}\left(\sum_{j=1}^{S}\alpha_j y_j K(\overline{x}_i,\overline{x}_j) + b\right) \quad (9)$$

The RBF kernel depends only on the radial distance (Bishop, 2006) which is the Euclidean distance ($||\overline{x}_i - \overline{x}_j||$):

$$K(\overline{x}_i,\overline{x}_j) = exp^{\frac{-||\overline{x}_i-\overline{x}_j||^2}{2\sigma^2}}$$

The Gamma ($\frac{1}{2\sigma^2}$) parameter is the distance that a single training example can reach and it is always σ>0, with small values meaning that the system can reach 'far' data, and large values meaning that 'close' data can be reached.

Our system has three classes; therefore, we need to use one-versus-one (ovo) method instead of one-versus-all (ova). (ovo) constructs $\frac{k(k-1)}{2}$ hyper-planes (Liu, 2007). The (ovo) strategy is more memory efficient than the (ova) (Bouchut, 2018).

6. EXPERIMENTAL RESULTS AND DISCUSSION

6.1 Experimental setup

A system is developed using the Python programming language on a computer equipped with a Core i7 5500U processor, 2.40 GHz and 8 GB RAM. The experiment is divided into three phases: training, validation, and testing. A third-party dataset of 27,620 sample character glyph images that was described in section 4 is used in our experiment.

The experiment starts by splitting the database into two portions, 80% of images for training and 20% of images for testing. Then training portion is further split into 80% for actual training of SVM and 20% for validation. Table 1 shows the distribution of data of our experiment. It is important to mention that the data that has been used in training differ from validation and testing; the same thing for validation and testing. There is no universal rule that governs the distribution of data between the three phases. In many studies, the dataset was divided into three parts as we did, but in different distributions. Senobari et al. split their dataset into 72% training, 5% validation and 23% testing (Senobari, 2012); while (Tensmeyer, 2017) distribute their data with unspecified ratios. Regardless of the experiment phase, a feature vector of size 106 is obtained and normalized for each image.

Table 1. Samples distribution among three phases of the experiment

Stage	#Samples	Percentage
Training	17,677	64%
Validation	4,419	16%
Testing	5,524	20%
Total	27,620	100%

6.2 Training phase

Distance profile features for 17,677 characters images were extracted for the training phase. To train the system, the feature vector of the training character images and their classes were fed to the SVM with an RBF kernel classifier and 75 models are created by using different values of C and Gamma parameters (Table 2). Because we have more than two classes (i.e. three fonts) one-versus-one technique has been used in the SVM classifier.

6.3 Validation Phase

In the validation phase (Figure 7), seventy-five experiments for various trials with different values of the C and Gamma were carried out to reveal a pair of C and Gamma values that are most suitable for recognition. The first data row of Table 2 shows the parameter values used in the validation phase of our main experiment. There is no specific limitation for choosing the value of the C and Gamma parameters (SKlearn Library, 2020). Therefore, we arbitrarily selected three values for the Gamma parameter (1/10, 1/26, and 1/106). We only used odd numbers for the C parameter to avoid excessive computation time. In total, 4,419 sample images were used in the validation phase. When looking at Figure 7, it can be shown that the Gamma value of 1/10 yields the highest recognition rate. The highest recognition accuracy of 96.51% is produced when the C parameter has one of the values 5, 7, 9, 11, or 13 with a corresponding Gamma value of 1/10. Therefore, the Gamma and C pair of (1/10, 7) was selected as an optimal parameter for the SVM, to be used in the testing phase.

6.4 Testing Phase

To find the font class of the test images, a database of 5,524 images is used. The tests are performed with the Gamma and C value pair (7, 1/10) chosen in the validation phase. The proposed system gets an overall recognition rate of 94.82%.

In the study by Bharath and Rani (Bharath, & Rani, 2017) using distance profile features and an SVM as a classifier, the authors obtained 80% recognition accuracy. Our system uses degraded images and the result of font recognition is about 14% better than the result of (Bharath, & Rani, 2017) where they used synthetic images.

Another comparison with (Al-Khaffaf, & Musa, 2018) study, where they used the same database but with a smaller number of samples (6144 images), and obtained recognition rate of

97%. The reasons for our system getting less recognition rate is that we used a number of samples that are 4 times much more than their samples and we obtained 94.82%. To show the effect of using a smaller dataset on getting a higher recognition rate, we performed a side experiment (second data row of Table 2) of image data twice the size of (Al-Khaffaf, & Musa, 2018) with 6,618 training samples and 6,884 testing samples. We obtained a recognition rate of 96.33% using (1/10, 21) Gamma and C value pair which is very close to (Al-Khaffaf, & Musa, 2018). Another reason for having a lower recognition rate is that (Al-Khaffaf, & Musa, 2018) used PCA to reduce the number of features to only prominent ones while we did not use this optimization step in this work.

Table 2. Gamma and C parameter values used in our experiments.

Experiments	DB Size	Gamma	C parameter	Accuracy
Main experiment	27,620	1/10, 1/26, 1/106	1, 3, 5, 7, ..., 47, 49	94.82%
Side experiment	13,502	1/10, 1/26, 1/106	1, 2, 3, 4, ..., 99, 100	96.33%

7. CONCLUSIONS

A font recognition system was presented in this paper that recognizes English fonts using an SVM with an RBF kernel. While the use of SVM in font recognition is not new, however, in this paper we tried to shed a light on the parameters of SVM represented by Gamma and C. Choosing the proper parameter values plays key role in getting high recognition rate in font recognition and shall not be selected randomly. The validation phase is important aspect of the experiment that will show the proper SVM parameters. However, we also note that our experiment is limited to only three fonts. In the future, we plan to run an experiment by using an extended dataset with more fonts, inspecting different feature extraction methods and using PCA to reduce number of features to only prominent ones.

8. ACKNOWLEDGEMENTS

The authors would like to thank the two anonymous reviewers for their valuable comments that helped in improving the quality of this manuscript.

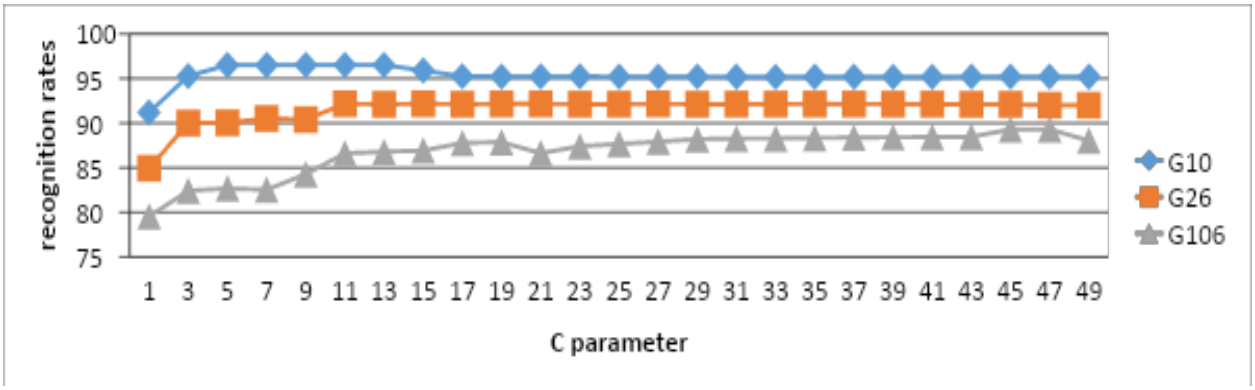

Figure 7. Validation phase. Recognition rate for a set of Gamma and C values. Best parameters for high recognition rate are when C=5, 7, 9, 11, or 13 and G=1/10.

REFERENCES

Al-Khaffaf, H. S., & Musa, N. A. (2018). Optical english font recognition in document images using eigenfaces. Revista Innovaciencia, 6(1), 1-11.

Al-Khaffaf, H. S., Shafait, F., Cutter, M. P., & Breuel, T. M. (2012, November). On the performance of Decapod's digital font reconstruction. In Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012) (pp. 649-652). IEEE.

Bharath, V., & Rani, N. S. (2017, June). A font style classification system for English OCR. In 2017 International Conference on Intelligent Computing and Control (I2C2) (pp. 1-5). IEEE.

Bishop, C. M. (2006). Pattern recognition and machine learning. springer.

Bouchut, Q., Appiah, K., Lotfi, A., & Dickinson, P. (2018, June). Ensemble One-vs-One SVM Classifier for Smartphone Accelerometer Activity Recognition. In 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS) (pp. 1110-1115). IEEE.

Bui, T., & Collomosse, J. (2015, September). Font finder: Visual recognition of typeface in printed documents. In 2015 IEEE International Conference on Image Processing (ICIP) (pp. 3926-3930). IEEE.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297.

Hajiannezhad, A., & Mozaffari, S. (2012, May). Font recognition using variogram fractal dimension. In 20th Iranian Conference on Electrical Engineering (ICEE2012) (pp. 634-639). IEEE.

Jaiem, F. K., Slimane, F., & Kherallah, M. (2017, February). Arabic font recognition system applied to different text entity level analysis. In 2017 International Conference on Smart, Monitored and Controlled Cities (SM2C) (pp. 36-40). IEEE.

Kanungo, T., Haralick, R. M., Baird, H. S., Stuezle, W., & Madigan, D. (2000). A statistical, nonparametric methodology for document degradation model validation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(11), 1209-1223.

Katiyar, G., Katiyar, A., & Mehfuz, S. (2017). Off-line handwritten character recognition system using support vector machine. American Journal of Neural Networks and Applications, 3(2), 22-28.

Kecman, V. (2005). Support vector machines–an introduction. In Support vector machines: theory and applications (pp. 1-47). Springer, Berlin, Heidelberg.

Kowalczyk, A. (2017). Support vector machines succinctly. Syncfusion Inc.

Liu, Y., Wang, R., & Zeng, Y. S. (2007, August). An improvement of one-against-one method for multi-class support vector machine. In 2007 International Conference on Machine Learning and Cybernetics (Vol. 5, pp. 2915-2920). IEEE.

Scholkopf, B., Sung, K. K., Burges, C. J., Girosi, F., Niyogi, P., Poggio, T., & Vapnik, V. (1997). Comparing support vector machines with Gaussian kernels to radial basis function classifiers. IEEE transactions on Signal Processing, 45(11), 2758-2765.

Senobari, E. M., & Khosravi, H. (2012, October). Farsi font recognition based on combination of wavelet transform and sobel-robert operator features. In 2012 2nd International eConference on Computer and Knowledge Engineering (ICCKE) (pp. 29-33). IEEE.

SKLearn Library (2020, February 21). https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html. Accessed on Sep 2019.

Tensmeyer, C., Saunders, D., & Martinez, T. (2017, November). Convolutional neural networks for font classification. In 2017 14th IAPR international conference on document analysis and recognition (ICDAR) (Vol. 1, pp. 985-990). IEEE.

Xu, Y., Zomer, S., & Brereton, R. G. (2006). Support vector machines: a recent method for classification in chemometrics. Critical Reviews in Analytical Chemistry, 36(3-4), 177-188.