EasyChair Preprint

№ 7672

Variable Length Digit Recognition for Gujarati Language

Shrey Malvi, Nirmal Patel and Pratik Prajapati

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

March 29, 2022

Variable Length Digit Recognition for Gujarati Language

Shrey Malvi
Data Science Department
Playpower Labs
Gandhinagar, India
shrey.malvi@playpowerlabs.com

Nirmal Patel
Data Science Department
Playpower Labs
Gandhinagar, India
nirmal@playpowerlabs.com

Pratik Prajapati
Data Science Department
Playpower Labs
Gandhinagar, India
pratik.prajapati@playpowerlabs.com

***Abstract*—In this paper, we describe a method to perform handwritten digit recognition for Gujarati - a regional Indian language. Our method can handle variable-length inputs, meaning that there are no limitations around the digit length for the input image. To our knowledge, this is the first attempt to do variable length digit classification for the Gujarati language numerals. We outline a two-step method to classify handwritten Gujarati numerals. The first step identifies connected components of the input image and predicts the numeric length of each connected component. The second step predicts the actual number that is contained within each connected component. The final result is a concatenation of individual predictions. Our Convolutional Neural Networks (CNN) architecture for this task has a low number of output classes (e.g. 30 classes for 3 digit classifier). Our method achieves 83.8% test set accuracy for 1 to 4 digit Gujarati numerals. On NIST19 dataset, our method achieves 96.1% test set accuracy for 2 to 6 digit English numerals.**

***Index Terms*—Handwritten Digit Recognition, Gujarati Language, Convolutional Networks, Image Processing**

I. Introduction

Handwriting recognition Machine Learning (ML) models are transforming many real-world processes. From automatic processing of bank cheques to paper forms, many manual data entry processes are becoming highly efficient through the use of more and more accurate handwriting recognition models. Research in this field has largely remained focused on the English language. There are multiple publicly available data set for the English numerals and alphabet (e.g. MNIST and Extended MNIST), and many research studies have proposed various methods to get over 95% accuracy [1].

To increase the impact of ML research in non-English speaking communities, we need to develop recognition models for other languages. The complexity of doing so depends on the language we are targeting. Our study focuses on the Gujarati language, which is spoken in the state of Gujarat in India. The population of the state is approximately 60 million. Gujarati numerals are similar to Hindi numerals, and at a high level, recognition of Gujarati numerals can be done similarly to the English numerals, except for the digit 9 which needs specialized treatment. You can see in Figure 1 that digit 9 is made up of two distinct parts of the ink. When it comes to the alphabet, the challenge becomes quite different. The Gujarati language has over 10 modifiers for all of its vowels and consonants, which makes the possible number of characters over 300. This makes it very difficult to do word recognition by using character-based models. Here, we may find word-level models more interesting.

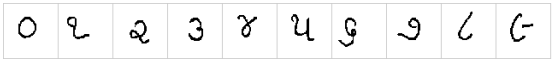

Fig. 1. Gujarati numerals from 0 to 9

To recognize the handwritten Gujarati numerals of any possible length, we designed a two-step approach. The first step divides the input image into connected components (different digits that are joined with each other). The second step recognizes the digits in each of the components separately. Our novel neural net architecture to recognize the digits in step 2 contains a smaller number of output nodes for detecting longer connected components. For example, a two-digit classifier will have 20 classes instead of 100, and a three-digit classifier will have 30 classes instead of 1000. In practice, it has been observed that it is extremely rare to have more than 4 digits connected [2]. This means that if we build a model to support up to 4 connected components, it will work out in most real-world scenarios. Our motivation to build the Gujarati handwriting recognition system revolved around building an automated math fluency worksheet checking system that teachers could use in the classrooms. Full or partial auto-evaluation can potentially save teacher time. The formative data from paper can also help teachers give personalized learning materials to the students (by identifying who needs help in which area). If such solutions can be deployed in large-scale public school settings such as in India, actions can also be taken at a system or government level. India and many other countries have large student populations in rural areas where children leave school before they can acquire basic numeracy skills. Application of computer vision to identify such students can help us give them needed interventions, and make learning more equitable.

The rest of the paper is structured as follows. Section 2 outlines the previous literature in Gujarati handwritten digit recognition and variable-length digit recognition. Section 3 describes our data sample. Section 4 details our proposed approach, and section 5 shows the results of our experiments. Lastly, Section 6 concludes the paper.

II. Literature Survey

A lot of research has been carried out in the field of variable length digit recognition for English numerals. To our knowledge, there has been no work has been done for variable-length digit recognition in the Gujarati language. First, we will mention prior work on variable-length digit recognition in the English language. Then, we will review the work on handwritten number recognition in the Gujarati language which is only limited to single digits.

Prior studies on variable-length number recognition were mainly based on segmentation-based and heuristics-based approaches. Segmentation-based approaches focus mainly on pre-processing digit strings and segregating them into a combination of isolated digits. These methods only require a 10 class classifier at the recognition phase. Ribas et al. proposed a segmentation-based method that relied on finding optimal segmentation-cut or a cut-point between two connected digits [3]. It is possible that while doing the segmentation, we end up with more segments than actual digits. This is called over-segmentation. Vellasques et al. proposed a method to filter out the components from over-segmentation using SVM classifiers [4]. Segmentation is not always perfect as there are many variations in handwriting. As the number length increases, the overall error rate of the segmentation process may increase because of some difficult to identify cut points. Apart from this, the segmentation-based methods put much of the overhead on the pre-processing and only rely on a single-digit recognizer in the end.

In recent years, several segmentation-free approaches have been introduced to overcome the drawbacks of segmentation-based methods. Hochuli et al. proposed a method where method they trained a separate classifier for each different length of the digit [2]. The method they proposed has a 100-class classifier and a 1000-class classifier for 2 and 3 length digits respectively. This method is difficult to scale to longer numbers. The models fall short of training a 4 or more length classifier (requires a 10000 class classifier). Later, several object detection-based methods such as YOLO were applied to the task of digit recognition task [5], [6]. In these methods, the individual digits are identified as objects, and the final locations of the detected objects help us construct the final output. These methods performed well on individually detecting and localizing the digits irrespective of the digit string length. But these methods don't perform well when the input data have overlapping digits. Also, for training these object detection-based models, we require annotated bounding boxes. Hochuli et al. adapted CRNN based scene-text detection methods [7] for variable-length digit recognition. Scene-text detection methods are designed to detect variable-length text from the photographs, and when Hochuli et al. [6] used this method for digit recognition, their model achieved excellent performance improvement over previously proposed methods. This method makes use of Recurrent Neural Networks (RNN) which is good at capturing the contextual information among the characters/words. For digit strings, there is no contextual information between two isolated digits [8], hence we eliminated the use of RNN in the architecture and proposed a purely CNN-based variable-length digit string recognition.

Fig. 2. Data samples collected from students

Various methods have been proposed for Gujarati handwritten numeral recognition, for both offline and online inputs [9], [10], [11], [12]. All of these methods have focused exclusively on single digits. To our knowledge, there is no major work that has explored and demonstrated the possibility and challenges of doing variable-length digit recognition. Thus, we have presented a novel work for handling variable-length strings in the Gujarati language by proposing a segmentation-free approach.

III. Data

We collected Gujarati handwritten numerals data from 400 students across 1st to 8th-grade students. The data collection forms were a grid full of numbers, showing students which numbers to write in which box. The data consisted of both pen-written and pencil-written samples ranging from 0 to 99. Figure 2 shows some examples of the handwritten digits. We collected 400 samples of each class and hence there was a wide range of variability in terms of handwriting, orientation, and positioning of the digits. The total data set consisted of 40000 samples. The samples which were too small to be considered as a blob or too large to be readable were removed. The data samples were resized to 64x64 pixel images and converted to gray-scale. Till now, the data set that was discussed for the Gujarati language consisted of only single-digit numerals. We have collected the handwritten numerals for up to two length digit strings and proposed the system to support variable-length strings.

We only collected 1 and 2 length handwritten digits and generated 3 and 4 digit numerals synthetically. For doing the synthetic data generation, we used single-digit numerals from the same author to generate new samples which are similar to real-world cases. First, we applied the image pre-processing step (discussed in a later section) on single-digit samples and then cropped the region containing digits. Then for concatenation, we randomly inserted gaps between two adjacent digit regions and generated the images. Figure 3 shows some examples of synthetically generated digits of more than 2 lengths. As the data contained the handwriting of many different people, there were strokes of digits with variable heights, widths, and angles, and concatenating those samples resulted in very robust and diversified synthetic data. We had also segmented the two-length digit samples into one-length digits to generate more samples for single-length digits. Table I shows a summary of total samples for each of the digit lengths in our data set.

TABLE I
Distribution of multi-length digits data

Length	samples per class		Total no. of digits/classes	Total samples
	collected	generated		
1-digit string	4000	-	10	40000
2-digit string	400	1000	100	140000
3-digit string	-	100	1000	100000
4-digit string	-	10	10000	100000

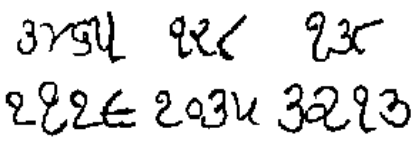

Fig. 3. Data samples collected from students

IV. Method

We have proposed a two-step digit recognition approach based on the work done by Hochuli et al. [2]. The first step involved finding the connected components of the input image and predicting the digit length of each of the components. We applied vertical projection for segmenting the connected components. In the second step, based on the length of the connected component, we applied the length-specific classification model. We made four separate digit classifier for 1, 2, 3 and 4 length digits ([0-9], [00-99], [000-999], [0000-9999]). Then the final result was generated by an aggregating predicted digit(s) of each connected component. So, we improved the method of Hochuli et al. [2] by adding a 4-length digit classifier and reducing the output neurons for each network.

A. Step 1: Length classification of connected components

The data consists of the digits written on white paper so we will interpret the foreground pixels as black and background pixels as white. As the data consists of only handwritten numerals, we do not need the color channel for the classification task. Hence, we converted every sample into a one-channel image by applying OTSU thresholding by keeping the default parameters. Apart from this, we also cleaned the samples by removing the noisy black patches around the boundaries. Figure 4 shows some samples after cleaning and binarization.

We have made a system to support a string of up to four connected components. To handle variable-length strings, we first found all the connected components in the input image and then processed each component separately. In the end, we aggregated the result of each component and generated the

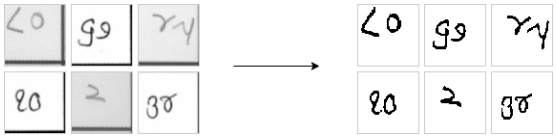

Fig. 4. cleaned data samples

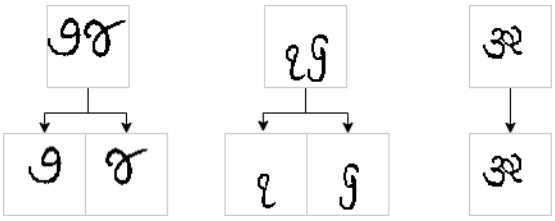

Fig. 5. Results of under segmentation using vertical projection

final result. For finding the connected components, we used vertical projection which is an under-segmentation technique. In vertical projection, the image gets segregated into different regions when there is visible white space between them. In this technique, we count the foreground pixel in every column. The column with zero foreground pixel will be considered as a separation point between two adjacent components. The reason behind using this segmentation technique was to handle the case of digit '9' represented as '૯' in Gujarati which consists of two disconnected components('C' and '-'). Figure 5 represents the results of applying vertical projection on two length strings. For the first two images in the Figure 5, we have at least one white column between two digits and hence they are divided into two separate components. For the last image, there is no white column in between both the digits, hence they are considered as one component. Hence, for higher length digit strings, we will apply vertical projection as a pre-processing step before feeding to the length classifier.

After dividing the image into connected components, we find the length of each component. We have created a length classifier, similar to [2] (which was based on Le-Net5), that can classify the input image between 1 to 4 length strings. We padded the collected samples to 64x96 (HxW) to accommodate up to 4 digits in a string. Figure 6 represents the block diagram of the proposed length classifier. The description of block L feature extractor is given in Table II.

For training the length classifier, we took 36,000 samples from every string of length 1 to 4. The data for 3 and 4 lengths were synthetically generated and these samples were created in such a way that there was no gap between adjacent digits. Hence, the total data used for training consisted of 1,44,000 samples. For better generalization, we applied data augmentation techniques. We created a custom data generator to apply width shift for 1 length string. For string lengths of 2 to 4, we applied shear and horizontal shifts. The model was trained by keeping a batch size of 64 and using Adam optimizer. Table IV shows the summary of training of length classifier.

Fig. 6. Block diagram for multi length classifiers

TABLE II
Feature block description for length and digit classifiers

Feature Block	layer 1	layer 2	layer3	layer 4	layer 5	Output
Block L	Conv2D 3x3 @64 + MaxPool 2x2	Conv2D 5x5 @128 + MaxPool 2x2	Conv2D 5x5 @256	GAP	FC @128	4
Block D			Conv2D 5*5 @128			10

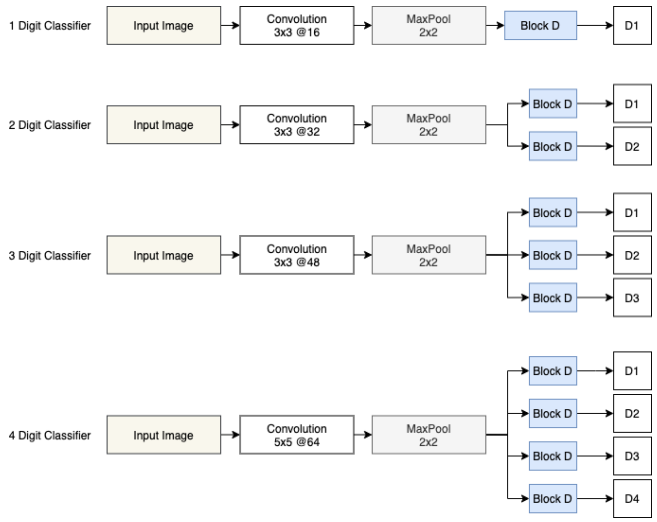

Fig. 7. Block diagram for multi length classifiers

B. Step 2: Digit classification of connected components

Step 1 gave us the predicted length of the connected component strings. Based upon the string length, we feed the image into the respective length string classifier. We developed four separate digit classifiers for string lengths 1, 2, 3, and 4.

For a one-length string, we created a 10 class classifier that has similar architecture to the length classifier discussed earlier. We started by creating a generic feature extractor block for one length classifier. The architecture of this generic Digit Classifier block is described in Table 2. This block was reused in multi-length string classifiers as well. Hochuli et al. [2] created a 100 class classifier for two lengths and a 1000 class classifier for three lengths. We have developed a novel system that reduces the number of predicted classes. We developed a novel multi-output CNN architecture that gives more than 1 output for the same input. Using the multi-output architecture, we created a 20 class classifier for recognizing the two-digit strings. The idea behind the multi-output CNN came by interpreting the multi-length number image as a combination of single digits. To exemplify, considering a two-length string sample '45', method by Hochuli et al. [2] classifies '45' among 100 classes whereas our multi-output network classifies '4' and '5' separately in their respective 10-class softmax classifier. Hence, by dividing a single task into two different tasks, we can perform the same functionality of a multi-length classifier by reducing the number of output units. Similarly, a 30 class and 40 class classifier were developed for recognizing three and four length strings. Figure 8 represents the architecture of two-length classifier.

TABLE III
Data distribution for training multi-digit classifiers

Length	Samples				Classes
	Train	Validation	Test	Total	
1-digit	27717	6930	1000	35647	10
2-digit	93429	23356	1000	117785	20
3-digit	80000	20000	1000	101000	30
4-digit	80000	20000	1000	101000	40

TABLE IV
Summary of training classifiers

Classifier	Parameters (x1000)	No. of Epochs	Training time (minutes)	Validation Acc(%)
Length	1,077	20	10	98.84
1-digit	278	30	15	98.31
2-digit	1,302	10	3	99.32
3-digit	1,980	10	3	99.47
4-digit	2,679	10	5	98.85

[a]CPU: Intel i7-8700k GPU: Nvidia GeForce RTX 3080

We developed multi-output CNN by replicating the feature extractor block D horizontally and divided the task of classifying strings at digit level. As shown in Figure 8, for two length classifier, we replicated the block D feature extractor twice and created two separate 10 neurons softmax classifiers for each digit. As shown in the architecture, we placed a single Convolutional layer (Conv+pool) before dividing the network. This layer is responsible for learning the position of the digits. We used the same architecture for 3 and 4 length strings as well by replicating block D respective number of times. Figure 7 shows the block diagram of multi-length digit string classifiers.

For training a one-length classifier, we leveraged the collected samples from [0-9] and also used the samples after isolating the collected two-length samples from [10-99]. The data for two lengths consisted of samples from 10-99. So, we synthetically created data samples from 00-09 to support 100-classes. For 3 and 4-length, all samples were synthetically generated. The data distribution for digit classifiers is mentioned in Table III. The data for training and validation consists of the samples we collected and generated whereas the testing data is based on the data provided by Goswami et. al [13]. We applied data augmentation techniques such as zooming, vertical shift, horizontal shift, and shear range for 1 length classifier. Whereas, for 2, 3, and 4 lengths, we used only width and height shift augmentation. For training, we used the same set of parameters as the length classifier model. Table IV shows the summary of training all the classifiers. We used the categorical-cross-entropy loss function for all classifiers. For higher length strings classifiers, we used n-loss functions for the n-output network. The loss weight was kept uniform for all the models.

V. Experiments

For assessing the performance of the proposed approach, we evaluated our model on handwritten numerals in Gujarati and English language.

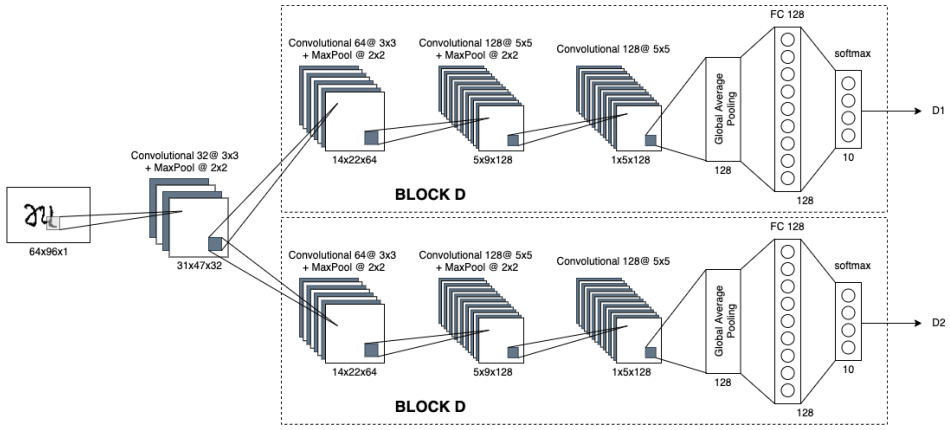

Fig. 8. Architecture for two-length classifier

A. Gujarati numerals dataset

To our knowledge, no research has been carried out in the field of variable length numeral string in the Gujarati language, and hence, we assessed the performance of our proposed model on synthetically generated samples using the data set provided by Goswami et. al [13]. This data set consists of 14000 samples of handwritten Gujarati numerals collected from 140 distinct subjects having different ages, educational backgrounds, and work cultures. So, this data contains a wide range of variability in terms of written strokes and handwriting. Each sample was binarized using OTSU thresholding. We created a new test based on this data and generated 1000 samples of each of 1 to 4 length strings using synthetic data generation algorithm. The performance of our length classifier has been described in Table V.

We can observe that our length classifier performed well for 1 and 2 length digit strings as there are few misclassifications. For digit strings 3 and 4, we observed a comparatively higher error rate than digit strings of lengths 1 and 2. We also noticed that the digit strings are mostly confused with their subsequent higher-length string. The overall accuracy of the length classifier is 91.8 %.

Table VI describes the evaluation for the test set at both the digit and string level. The overall accuracy is calculated while taking both the length and digit error rate into account. We can observe the error rate is increasing with increase in string length. This is because of the increase in complexity of the connected numerals for higher length strings. From the table, we can note that the error rate for intermediate digits is relatively higher than the digit at the start and end of the strings for 3 and 4 length digit strings. We got average overall accuracy of 83.8% for the test-set of 4000 samples. The overall confidence of the predicted image is calculated by multiplying the probability of length classifier and probabilities from the digit classifiers.

Figure 9 represents the predictions with corresponding probabilities. We have observed that the majority of the misclassifications are due to confusion between numerals because, unlike the English language, Gujarati numerals have very similar

TABLE V
Confusion matrix for length classifier

Digit string length	Predicted length				Accuracy (%)
	1	2	3	4	
1	936	64	0	0	93.6
2	0	944	55	1	94.4
3	0	5	889	104	88.9
4	0	0	96	904	90.4

TABLE VI
Evaluation of length and digit classifiers on generated test set

Digit string length	Step-1 error (length)(%)	Error at digit level(%)				Step-2 error (digit)(%)	Overall accuracy(%)
		C1	C2	C3	C4		
1	6.4	1.3	-	-	-	1.3	92.3
2	5.6	2.8	2.8	-	-	5.5	88.9
3	11.1	2.0	4.1	2.1	-	7.8	81.1
4	9.6	3.6	5.4	5.6	4.5	17.5	72.9
Avg.	**8.2**	-				**8.0**	**83.8**

shapes. In the Gujarati language, there are multiple sets of confusing pairs such as '૧' and '૨', '૮' and '૯', '૬' and '૭'. We also noted that the proposed networks fall short when the stroke of digits is stretched. For example, in the above fig, the image with ground truth '૫૯' has been wrongly predicted as a three-digit string as '૧૮૯' as the digit '૫' was stretched which looks like a combination of connected numerals '૧' and '૮' . Also, for string '૧૬', it was confused with '૧૭' because

Fig. 9. Architecture for two-length classifier

TABLE VII
Comparison of recognition rates for NIST19 database

Length	No. Samples	Britto et al. [14]	Oliveria et al. [15]	Sadri et al. [16]	Gattal et al. [17]	Houchuli et al. (2018)[2]	S.Aly et al. [18]	Our method	Houchuli et al. (2020)[5]
2	2370	94.8	96.8	95.5	99	97.6	98.8	98.39	98.6
3	2385	91.6	95.3	91.4	97.3	96.2	96.4	96.77	97.6
4	2345	91.3	93.3	91	96.5	94.6	95	95.44	97.1
5	2316	88.3	92.4	88	95.9	94.1	95.4	95.41	96.5
6	2169	89	93.1	88.6	96.6	93.3	95	94.32	95.8
Average	-	91	94.2	90.9	97.1	95.2	96.1	**96.1**	97.1

of similarity of digit '૬' and '૭'. After carrying out several experiments on the newly generated test-set, we have defined the minimum threshold of 85% for detection.

B. English numerals dataset

The proposed models might seem biased towards learning the type of connected components in the Gujarati test set because the same algorithm was used for generating both train and test data. Hence, to validate the generalization over real and unseen connected components, we evaluated our models on the English language real data set NIST19. For this evaluation, we kept the input size of 64x64 to compare with existing approaches. We used the same train/test distribution of NIST19 and pre-processing steps as given by Hochuli et al. [2] for training our length and digit classifiers. The test set consists of 2, 3, 4, 5, and 6-length strings from the hsf-7 series of the NIST19 database. The numeral strings contain several discrepancies in terms of touching components and broken fragments. We obtained an overall accuracy of 96.1% for all five categories of numeral strings in the NIST19 database. Table VII shows the comparison of the results with existing approaches.

Some methods shown in the table VII did not do well for more than 3 digits, while others performed well. Our method also performed well for 4,5 and 6-length digits. By attaining near state-of-the-art results on real data, we validated that our proposed method is not biased in learning fixed type of connected components, and it generalizes well over unseen data as well.

VI. Conclusion and Future Work

We have proposed a segmentation-free approach to handle variable-length strings in the Gujarati language which requires fewer output neurons for classification. We have also demonstrated the performance of our model on the new Gujarati digit strings test-set generated by our data generation algorithm. Moreover, we also evaluated real-world data of the English numerals to validate the generalization of our method. We thank Mayur Jaguwala for helping with the data collection. For future work, we will be looking for the method to create one single classifier which can handle both the task of length classification and digit classifications.

References

[1] A. Baldominos, Y. Saez, and P. Isasi, "A survey of handwritten character recognition with mnist and emnist," *Applied Sciences*, vol. 9, no. 15, p. 3169, 2019.

[2] A. G. Hochuli, L. S. Oliveira, A. Britto Jr, and R. Sabourin, "Handwritten digit segmentation: Is it still necessary?" *Pattern Recognition*, vol. 78, pp. 1–11, 2018.

[3] F. C. Ribas, L. Oliveira, A. Britto, and R. Sabourin, "Handwritten digit segmentation: a comparative study," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 16, no. 2, pp. 127–137, 2013.

[4] E. Vellasques, L. S. Oliveira, A. Britto Jr, A. L. Koerich, and R. Sabourin, "Filtering segmentation cuts for digit string recognition," *Pattern Recognition*, vol. 41, no. 10, pp. 3044–3053, 2008.

[5] A. G. Hochuli, A. S. Britto, J. P. Barddal, R. Sabourin, and L. E. Oliveira, "An end-to-end approach for recognition of modern and historical handwritten numeral strings," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[6] A. G. Hochuli, A. S. Britto Jr, D. A. Saji, J. M. Saavedra, R. Sabourin, and L. S. Oliveira, "A comprehensive comparison of end-to-end approaches for handwritten digit string recognition," *Expert Systems with Applications*, vol. 165, p. 114196, 2021.

[7] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 11, pp. 2298–2304, 2016.

[8] H. Zhan, P. N. Chowdhury, U. Pal, and Y. Lu, "Handwritten digit string recognition for indian scripts," in *Asian Conference on Pattern Recognition*. Springer, Cham, 2019, pp. 262–273.

[9] J. R. Prasad, U. V. Kulkarni, and R. S. Prasad, "Template matching algorithm for gujrati character recognition," in *2009 Second International Conference on Emerging Trends in Engineering & Technology*. IEEE, 2009, pp. 263–268.

[10] A. A. Desai, "Gujarati handwritten numeral optical character reorganization through neural network," *Pattern recognition*, vol. 43, no. 7, pp. 2582–2589, 2010.

[11] M. Maloo and K. Kale, "Support vector machine based gujarati numeral recognition," *International Journal on Computer Science and Engineering*, vol. 3, no. 7, pp. 2595–2600, 2011.

[12] A. A. Desai, "Support vector machine for identification of handwritten gujarati alphabets using hybrid feature space," *CSI transactions on ICT*, vol. 2, no. 4, pp. 235–241, 2015.

[13] M. M. Goswami and S. K. Mitra, "Offline handwritten gujarati numeral recognition using low-level strokes," *International Journal of Applied Pattern Recognition*, vol. 2, no. 4, pp. 353–379, 2015.

[14] A. d. S. Britto Jr, R. Sabourin, F. Bortolozzi, and C. Y. Suen, "The recognition of handwritten numeral strings using a two-stage hmm-based method," *International Journal on Document Analysis and Recognition*, vol. 5, no. 2-3, pp. 102–117, 2003.

[15] L. S. Oliveira, R. Sabourin, F. Bortolozzi, and C. Y. Suen, "Automatic recognition of handwritten numerical strings: A recognition and verification strategy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 11, pp. 1438–1454, 2002.

[16] J. Sadri, C. Y. Suen, and T. D. Bui, "A genetic framework using contextual knowledge for segmentation and recognition of handwritten numeral strings," *Pattern Recognition*, vol. 40, no. 3, pp. 898–919, 2007.

[17] A. Gattal, Y. Chibani, and B. Hadjadji, "Segmentation and recognition system for unknown-length handwritten digit strings," *Pattern Analysis and Applications*, vol. 20, no. 2, pp. 307–323, 2017.

[18] S. Aly and A. Mohamed, "Unknown-length handwritten numeral string recognition using cascade of pca-svmnet classifiers," *IEEE Access*, vol. 7, pp. 52 024–52 034, 2019.