



Increasing the effectiveness of handwritten Manipuri Meetei-Mayek character recognition using multiple-HOG-feature descriptors

KISHORJIT NONGMEIKAPAM¹, WAHENGBAM KANAN KUMAR^{2,*},
OINAM NICKSON MEETEI³ and THEMRIKON TUITHUNG³

¹Department of Computer Science and Engineering, Indian Institute of Information Technology Manipur, Imphal, India

²Department of Electronics and Communication Engineering, North Eastern Regional Institute of Science and Technology, Nirjuli, India

³Department of Computer Science and Engineering, National Institute of Technology, Nagaland, Dimapur, India

e-mail: kishorjit@iitmanipur.ac.in; wahengbam.kanankumar@gmail.com; nickson.oinam@gmail.com; t_tuithung@yahoo.com

MS received 28 August 2017; revised 22 February 2018; accepted 3 January 2019; published online 5 April 2019

Abstract. Detection and reading of the text from natural images is a difficult computer vision task, which is essential in a variety of emerging applications. Document character recognition is one such problem, which has been widely studied and documented by many machine learning and computer vision researchers, which is practically used for solving applications like recognizing handwritten digits. In this paper, a new approach for efficiently extracting cognition out of a total of 56 different classes of Handwritten Manipuri Meetei-Mayek (HMMM) (an Indian language) is described. Although character recognition algorithms have been researched and developed for other Indian scripts, no research work has been reported so far for recognizing all the characters of the Manipuri Meetei-Mayek (MMM). The work begins with a thorough analysis of the recognition task using a single hidden layer type Multilayer Perceptron Feedforward Artificial Neural Network with Histogram of Oriented Gradient (HOG) feature descriptors. After reviewing the level of accuracy and time it takes to train the network, the limitations are experimentally removed using multiple-sized cell grids using HOG descriptors. HOG, being a gradient-based descriptor, is very efficient in data discrimination and very stable with illumination variation. For efficient classification of the HOG features of the MMM, a linear multiclass support vector machine (SVM) classifier has been proposed for classifying the different offline characters because of its simplicity and speed. The classification based on linear multiclass SVM yielded a very high overall accuracy of 96.928%

Keywords. Manipuri Meetei-Mayek (script); multilayer perceptron; feedforward artificial neural network; histogram of oriented gradient (HOG); linear multiclass support vector machine (SVM).

1. Introduction

Handwritten character recognition is increasingly gaining momentum owing to its application areas for significantly reducing time in applications such as data entry, filling forms, banking, automation in postal services, etc. However, developing a more dependable approach or more technically 'a system' for recognizing handwritten characters of such regional scripts still poses a challenge to researchers. The main reason behind the problem is the variation in the shapes of the characters that may otherwise depend upon certain factors like acquisition device,

ink colour, the width of the pen and many other factors. Moreover, handwritten Meetei-Mayek characters tend to be much more complex in comparison with common English characters due to the presence of modifiers, shape and structure. These factors demand a sophisticated pattern recognition algorithm that will be able to efficiently handle the challenging task of classifying these characters.

In this paper, the design of a Handwritten Manipuri Meetei-Mayek (HMMM) character recognition system is being discussed. The history and origin of Meetei-Mayek can be found in detail in the works of literature by Wangkhemcha [1], Mangang [2] and Hodson [3]. Manipuri or Meeteilon is one of the scheduled languages of India and also the official language of Manipur, which is one of

*For correspondence

the states located in the North-Eastern part of India. The script contains a total of 56 characters, which can be classified into 5 different categories: Iyek Ipee/Mapung Iyek, which consists of 27 alphabets, Cheitek Iyek (8 symbols), Lonsum Iyek (8 letters), Khudam Iyek (3 symbols) and Cheishing Iyek, which consists of 10 numeral figures. Apart from the three Khudam Iyek, all other characters are internationally accepted symbols. The basic characters or the Iyek Ipee appear only as the main character of a word, which may be modified by adding one of the extended symbols or Vowel modifiers to produce the required pronunciation. All the original characters of the Manipuri Meetei-Mayek (MMM) alphabets are drawn, winded and wreathed based on the features of the human anatomy. Accordingly, the alphabets are named after different parts of the human body [2]. The Meetei-Mayek characters for which recognitions are performed in the

current work is shown in figure 1(a) along with the meaning against their names.

The paper is organized in the following order. Section 2 highlights some works related to handwritten character recognition. Section 3 presents the system design of the proposed handwritten character recognition method. Section 4 presents the experimental results, error analysis and evaluations of the proposed character recognition approaches described in section 3. Section 5 charts a comparison between the two approaches for determining size of the feature descriptor, and also to other existing research works for MMM character recognition. Section 6 concludes the paper.

2. Related works

Introduction of MMM OCR is in an infant stage whereas many research works have already been carried out on other Indian scripts of different languages. Sections 2.1 and 2.2 highlight the research works carried out on popular Indian languages and MMM, respectively.

2.1 Research works on other Indian languages

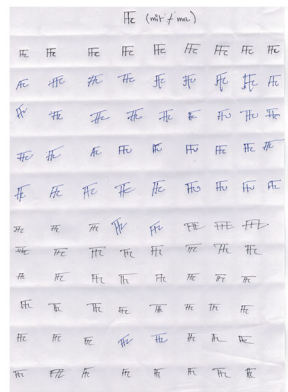
Rani *et al* [4] focused on the problem of recognition related to Gurumukhi script; they used different techniques for extracting features such as projection histogram, background directional distribution (BDD) and zone-based diagonal features. These features extraction techniques were classified using a support vector machine (SVM) classifier with 5-fold cross-validation and RBF (radial basis function) kernel. They achieved a very high accuracy of 99.4% using a combination of BDD and diagonal features with SVM classifier. Arora *et al* [5] discussed the characteristics of some classification methods that have been successfully applied to handwritten Devanagari character recognition. The results showed that good classification of Devanagari can be achieved with SVM. Sinha [6] presented an overview of the historical development of the modern Indian scripts' writing system, their mechanization and adaptation to computing and examined how it facilitated the development of Indian language processing. He concentrated primarily on the Devanagari script and also discussed those features found in current language usage; he explained how the unifying characteristics of the scripts and languages have been exploited for all Indian scripts and languages. Pal *et al* [7] proposed a system for recognizing offline Bengali handwritten compound characters using Modified Quadratic Discriminant Function (MQDF). Using a 5-fold cross-validation technique they were able to obtain an accuracy of 85.90% from a dataset of Bengali compound characters containing 20,543 samples. Sharma *et al* [8] proposed a scheme for unconstrained offline handwritten Devanagari numeral and character recognition using a

Eyek Epee (Original Letters)	
kok (ko) head	nam (na) hair
lai (la) forehead	mit (ma) eye
pa (pa) eyelash	na (na) ear
chil (cha) lip	sil (sa) saliva
khon (kha) saliva	ngon (nga) pharynx
thou (tha) throat	wai (wa) voice
yang (ya) throat	huk (ha) throat
hok (ho) throat	ee (ea) blood
pham (pha) throat	stiya (sa) throat
pham (pha) throat	stiya (sa) throat

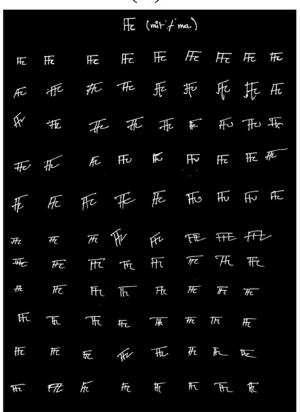
Lom Eyek (Additional Letters)	Cheising Eyek (Digits)
gok (ga) pham (pha)	0 1 2 3 4
ru (ra)	5 6 7 8 9
ba (ba) sil (sa)	
di (da)	
phon (pha) thon (tha) bhan (bha)	

Lonsum Eyek (Letters with short ending)	Cheitap Eyek (Vowel Signs)
kok (k) lai (l) mit (m) pa (p)	atap (aa) inap (i)
na (n) ni (r) ngon (ng) ee (ee)	yetap (e) utap (o) cheitap (ei)
	scounap (ou) mung (ug)

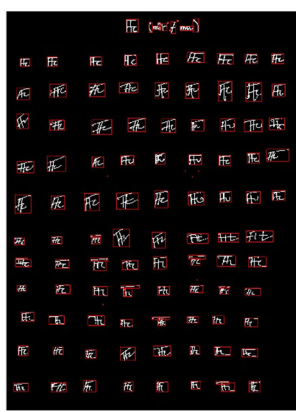
Khudam Eyek (Punctuation Marks)
question mark comma fullstop



(b)



(c)



(d)

Figure 1. (a) Meetei-Mayek script. (b) A sample of the handwritten character ‘MIT(Ma)’. (c) Pre-processed image. (d) All elements are detected and then encapsulated prior to extraction of each one of them.

quadratic classifier based on feature obtained from chain code histogram. They were able to achieve an average accuracy of 98.86% for Devanagari numerals and 80.36% for Devanagari characters. Basu *et al* [9] presented a recognition system for handwritten Bengali alphabets using a 76-element feature set, which included 24 shadow features, 16 centroid features and 36 longest-run features. The recognition performances achieved for training and test sets were 84.46% and 75.05%, respectively. Plamondon and Srihari [10] presented a comprehensive survey for online and offline handwriting recognition; they described the nature of handwritten language and the basic concepts behind written language recognition algorithms. They also indicated in their literature the algorithms for pre-processing, character word recognition and performance with practical systems. Other fields of application like signature verification, writer authentication and handwriting learning tools were also considered.

2.2 Research works on MMM

Maring and Dhir [11] described the recognition of Meetei-Mayek numerals for both handwritten as well as printed. A Gabor filter was used for feature extraction and classification was carried out using SVM. The experiment was carried out using 14×10 pixel images and overall accuracy of 89.58% and 98.45% was achieved for handwritten and printed, respectively. Romesh *et al* [12] described the design of an OCR system for handwritten text in Meitei-Mayek alphabets using artificial neural network (ANN). The database contained 1000 samples, from which 500 samples were considered as a training database and the remaining samples were kept for testing and validation purpose. They observed that the success of the system depended on the feature used to represent the character as well as on the segmentation stage of the test image. Chandan and Sanjib [13] in their literature presented an SVM-based handwritten numeral recognition system for Manipuri script or Meetei-Mayek. They used various techniques for extracting features such as BDD, zone-based diagonal, projection histograms and Histogram Oriented features, which were then classified using SVM as 5-fold cross-validation with RBF kernel. They were able to achieve maximum accuracy of 95%. Romesh *et al* [14] described a way for simulating and modelling handwritten Meitei-Mayek digit using back-propagation (BP) neural network approach. They were able to achieve an overall performance of 85%. Thokchom *et al* [15] proposed methods for training BP network with probabilistic features, fuzzy features and a combination of both features for recognizing handwritten Meetei-Mayek characters. They were able to achieve an accuracy of 90.3% for the proposed 27-class classifier neural network with a combination of probabilistic and fuzzy features.

3. System design

The motivation of this paper is to propose a robust method for classifying offline HMM characters. In the current work, a comparison of Multilayer Perceptron Neural Network (MLPNN)-based classification with Histogram of Oriented Gradient (HOG) descriptors and multiple-feature-size HOG descriptor with linear kernel SVM classifier is presented (figures 2 and 3). The work began with a thorough literature survey of the existing works in MMM script. It was realized that so far no literature exists that can successfully or efficiently classify handwritten Meetei-Mayek alphabets and numerals, which is due to the complex nature of the script. However, previous works reported on numerals alone were quite successful as reported in section 2.2 under the heading ‘Research works on MMM’.

In order to fully comprehend the nature of problem affecting the recognition accuracy of such handwritten characters, two different approaches are being studied in detail. To begin with, all the acquired sample images are pre-processed to remove noise as well as for extracting them individually. The pre-processing steps are discussed in section 3.1. The pre-processed image samples are passed to HOG descriptor system having their required cell sizes for the purpose of extracting feature vectors. As a first approach, the HOG feature vectors were trained and recognized using MLPNN. The classification task using this approach as described in section 3.3 under the

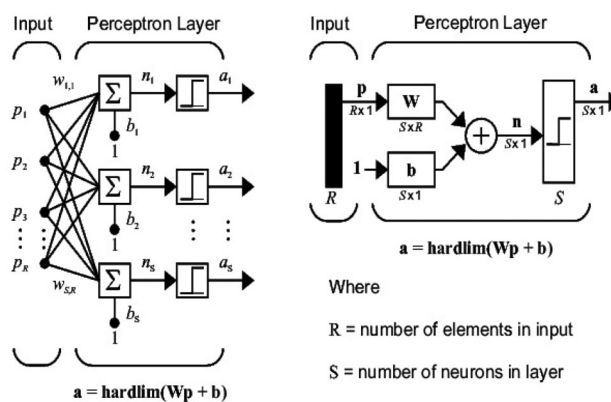


Figure 2. Perceptron neural network.

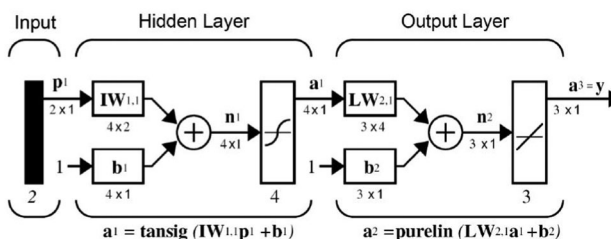


Figure 3. Feedforward neural network.

heading ‘MMM recognition using MLPNN with HOG descriptors’ was done once, i.e. using 128×128 input image sizes. It was learnt that an increase in the input image size resulted in just a meagre increase in recognition result, which did not justify the accuracy, however small the time taken to train the network was. It was also realized from the three confusion matrices that most of the characters like ‘ᱚ (2)’, ‘ᱜ (4)’, ‘ᱠ (LAI)’, ‘ᱡ (MIT)’, ‘ᱢ (PA)’, ‘ᱣ (NA)’, ‘ᱤ (CHEEN)’, ‘ᱥ TIL’, ‘ᱦ (KHOU)’, ‘ᱧ (WAI)’, ‘ᱨ (GOK)’, ‘ᱩ (RAAI)’, ‘ᱪ (GHOU)’, ‘ᱫ (DHOUSUM)’, ‘ᱬ (BHAM)’, ‘ᱭ (KOK-LONSUM)’, ‘ᱮ (MIT-LONSUM)’, ‘ᱯ (NGOU-LONSUM)’, ‘ᱰ (EE-LONSUM)’, ‘ᱱ (INAP)’, ‘ᱲ (UNAP)’, ‘ᱳ (SOUNAP)’, ‘ᱴ (CHEINAP)’ and ‘ᱵ (NUNG)’ were consistently misclassified or confused amongst themselves despite the increase in the number of image size. However, other characters showed a slight improvement in their individual accuracies.

Therefore, based on the work by Dalal and Triggs [16], section 3.3 describes a procedure for efficiently recognizing HMM using HOG feature descriptors and linear SVM classifier. Their feature extractor worked by dividing up an image into small spatial regions or cells; each of these regions accumulated a local 1-D edge orientation over pixels of the cell, and the combined histogram entries formed the representation. In this work, multiple cell sizes for extracting HOG features have been considered in order to determine which size yielded better results for our current classification problem. The extracted feature vectors were used as training data for the linear kernel SVM classifier. Thus, we were able to obtain a significant increase in overall or average accuracy along with a tremendous decrease in training time as compared with the former.

3.1 Processing the handwritten image

In this section, the stages prior to recognition stage are being described.

3.1a Image acquisition: In this stage, raw data are created and collected. A total of 5600 handwritten samples were collected from people having different handwriting styles. Secondly, the image samples were scanned using a scanner and saved as a jpeg file. A sample of the acquired handwritten image for the letter ‘ᱡ (MIT)’ is shown in figure 1(b).

3.1b Pre-processing: In order to make the image suitable for further processing the acquired images must be pre-processed. The term pre-processing refers to the removal of any form of noise that is corrupting the useful data so that efficiency as a result of it is not decreased. For character recognition tasks, a binary image is sufficient to work with, so the input grey image is suitably transformed using thresholding. Morphological erosion is performed so as

to close the discontinuities between some letters; a square-shaped structural element having a size equal to 2 is selected for the purpose. Morphological erosion is a simple operator in mathematical morphology that is usually performed in binary images or greyscale images. The purpose of the operation is to erode or decay the boundaries of regions of the foreground pixels (i.e. white pixels), and therefore the areas of foreground pixels shrink in size, and holes within those regions become larger. The morphologically eroded image is finally converted into a binary image [17]. Figure 1(c) shows the final image after pre-processing.

3.1c Extracting individual elements: Prior to extracting each element from the binary image so obtained in the previous step, each of them must be labelled so that automatic extraction from them is possible. For this purpose, each of the elements is bounded by rectangular boxes. It can be seen from figure 1(d) that the size of each of the boxes differs due to the fact that some characters are bigger than others and vice-versa. The bounding box property for each object is an array having 4 elements, which are formatted as $[x, y, w, h]$, where (x, y) represents the row-column coordinates of the upper left corner of the box; w and h are, respectively, the width and height of the box. The next step is creating a 4-column matrix that encapsulates all of these bounding box properties together, where each row denotes a single bounding box. It is necessary to define a good illustration of these bounding boxes, and thus a red box is drawn around each character that is detected. Now, the final task is to extract all of the characters and place them into a cell array because the character sizes are uneven, so putting this into a cell array will accommodate for the different sizes. A cell array is a type of container used for indexing data called cells; each cell may contain any type of data. Commonly they may contain combinations of text and numbers, or list of strings or numeric arrays of varying sizes. Now simply looping over every bounding boxes that we have and then extracting the pixels within each of them will result in a character that can be placed in a cell array. Thereafter, using a loop function, each of the characters in the cell array is written into the directory for further usage.

3.2 Feature extraction from Handwritten Meetei-Mayek script using HOG descriptors

Detecting features in Meetei-Mayek script is a complicated task due to similarity complex of each character. The very first requirement is a robust feature detector that conforms to the shape or structure of the input image so that characters can be discriminated clearly. The current study inclines on the issues of feature set extraction from Handwritten Meetei-Mayek script using the HOG descriptors. The features extracted by multiple-cell-sized HOG features

are used as training data for multiple classifiers; the detailed implementation is explained in sections 3.3 and 3.4.

HOG is a standard image feature used, among others, in object detection and deformation object detection. The method evaluates normalized histograms of gradient orientation of images in a dense grid. The most simple explanation is because the shapes and appearance of an object can be characterized easily by distributing the edge detection even without exact knowledge of the corresponding edge positions. It is implemented by dividing up the image window into ‘cells’, which are small spatial regions. Each cell will accumulate a local 1-D histogram of gradient directions over the cell, and the combined histogram entries form the notation. It is also useful to properly equalize the contrast for improved invariance to shadowing or illumination effects before putting them to use. This feature is achieved by accumulating a measure of ‘energy’ of the local histogram over somewhat larger spatial ‘blocks’ or region and then normalizing all of the cells in the block. This is also referred to as HOG descriptor.

HOG divides the input image into square cells of cell size, fitting as many cells as possible, filling the image domain from the upper-left corner down to the right one. For each row and column, the last cell is at least half contained in the image. More precisely, the number of cells obtained in this manner is

$$width_hog = (width + cellsize/2)/cellsize, \quad (1)$$

$$height_hog = (height + cellsize/2)/cellsize. \quad (2)$$

Later, the image gradient $\delta l(x, y)$ is computed using central difference, which is then assigned to one of the $2 \times \text{number_of_orientations}$ orientation in the range $[0, 2\pi]$. The contributions are then accumulated using bilinear interpolation to four neighbouring cells, which results in a histogram of dimension $2 \times \text{number_of_orientations}$, called directed orientations since it accounts for the direction as well as the orientation of the gradient.

Implementation: The implementation of the HOG feature descriptors for Meetei-Mayek script is based on the research work by Dalal and Triggs [16]. The detector has been tested in our Meetei-Mayek database, which roughly comprises 56 different classes multiplied by 100 samples each.

The training images comprise roughly 56 different classes times 75 samples each. The pre-processing procedure detailed in section 3.1 is used to segment each of the character samples and finally, the images are resized to 50×50 pixels. For testing, the remaining 25 samples for each of the character/class are used to validate how well the classifier performs on data that are different from the training data. Although this is not the most representative dataset, there are enough data to train and test a classifier, and show the feasibility of the approach.

The data that are used for training the classifier are the HOG feature vectors extracted from the input training images. Hence, it is important that the feature vector encodes a sufficient amount of information about the object. With the variation in cell size parameter, the amount of information encoded by each feature vectors can be observed. Each of the pixels in the image calculates a weighted vote for an edge orientation histogram channel. The weighted vote, which is based on the orientation of the gradient element, is accumulated into bins over local regions, which are termed as cells. The orientation bins are specified as a logical scalar and they are evenly spaced from 0 to 180 degree. In this case, a scalar of value less than 0 is placed into the +180 degree value bin. The dark to light versus light to dark transitions contained within some areas of an image can be differentiated using signed orientation. The bilinear interpolation of votes between the neighbouring bin centres can reduce aliasing for orientation as well as position. Increasing cell size can be used for capturing large-scale spatial information. It may be noted that cell size is specified as a 2-element vectored form in pixels. The suppression of changes in local illumination may be reduced with increasing cell size, i.e., losing minute details as a result of averaging. Therefore, a reduction in the size of blocks will help in capturing the significance of local pixels. However, in actual practice, the gradient parameters must be varied by repeatedly training and testing for identifying the optimal parameter settings.

For instance, in the current work, the optimal block size of HOG feature that must be maintained for efficiently recognizing Meetei-Mayek characters is explored by considering the cell sizes, viz., 6×6 , 7×7 and 8×8 . Figure 4 shows the features extracted using HOG descriptors for the Meetei-Mayek numeral ‘ᱥᱟ (9)’.

The extracted HOG features are returned as a $1 \times N$ vector (the features encode local shape information from regions or from point locations within an image) where N is HOG feature length and is based on the image size and the function parameter values. Let us suppose that B_{image} is the number of blocks per image, C is the cell size, N_b is the number of bins, B_o is the block overlap, B_{size} is the block size and $size_{image}$ is the size of the image. The following equations are used for appropriately deducing the value of N :

$$N = B_{image} B_{size} N_b \quad (3)$$

where

$$B_{image} = \frac{\left(\frac{size_{image}}{C} - B_{size}\right)}{B_{size} - B_o} + 1. \quad (4)$$

Table 1 highlights the number of detected features on MMM for different cell sizes. It is important to deduce the dimension of cell size that gives us the best recognition performance when combined with classifiers.

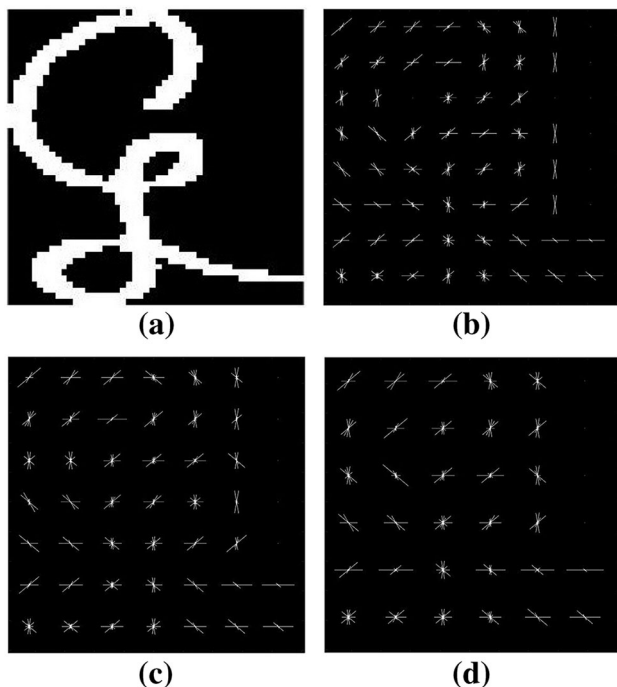


Figure 4. (a) Sample of the pre-processed handwritten Meetei-Mayek numeral '9'. (b) HOG feature of 6×6 size cell with length 1764. (c) HOG feature of 7×7 size cell with length 1296. (d) HOG feature of 8×8 size cell with length 900.

Table 1. Cell size versus HOG feature length.

Cell size	Length
6×6	1764
7×7	1296
8×8	900

3.3 MMM recognition using MLPNN with HOG descriptors

ANNs can be aptly stated as one of the popular techniques for task related to recognition and classification because of their learning and generalization abilities. They are composed of multiple layers, within which large processing elements or more specifically neurons are interconnected to one another and also work in unison for solving problems. They can be tuned for solving specific applications like MMM recognition using MLPNN data classification or pattern recognition via a learning approach. The multilayer perceptron is a network of fully connected neurons having an input layer, hidden layer(s) and an output layer. The neurons in a layer are connected to each and every neuron in the next layer by a weighted link through which the state of the neuron is transmitted. Each layer has a different activation function, with different neurons in it.

3.3a Perceptron: The perceptron neural network consists of a single layer of S neurons connected to R inputs through

a set of weights w_{ij} as shown in figure 2. The indices i and j indicate that w_{ij} is the strength of the connection from the j^{th} input to the i^{th} neuron.

3.3b Feedforward neural network: Feedforward network usually consists of one or more hidden layers of sigmoid neurons followed by linear output neurons. More than one hidden layer with a non-linear type transfer function will facilitate the network to learn non-linear as well as the linear relationship between the input and output vectors. The linear output layer produces values in the range -1 to $+1$.

On the other hand, a sigmoid-type transfer function should be used if the outputs of the network are required to be constrained (such as between 0 and 1). The superscript on the weight matrices is determined by the number of layers in case of multiple-layered network structure, which can also be noted from the neuron model and network architectures. As seen in figure 3, a two-layered tansig/purelin network is shown; it can be used as a general approximation function to suitably approximate any type of function with a finite number of discontinuities, provided the number of neurons in the hidden layer is sufficient.

3.3c BP algorithm: The BP algorithm is the most popular method for neural networks training and it has been used to solve numerous real-life problems. BP is a multilayer feedforward neural network that performs iterative minimization of a cost function by making weight connection adjustments according to the error between the computed and the desired output values. Figure 2 shows a general three-layer network. The error or cost function is the mean squared sum of differences between the output values and the desired target values of the desired network. The following formula is used for this error:

$$E = \frac{1}{2} \sum_p \left(\sum_k (t_{pk} - o_{pk})^2 \right). \quad (5)$$

In the current work, a general three-layer BP network is being used. When w_{ik} changes, it affects the error only on one output unit k . When w_{ij} changes, it affects the error on all the output units. Here p in the subscript represents a pattern and k represents the output units. Thus, t_{pk} is the target value of output unit k for pattern p and o_{kp} is the actual output value of the output layer unit k for pattern p . This error function is commonly used; however, other types of error function can also be applied. During the training process, a set of pattern examples is used, each example consisting of a pair with the input and corresponding target output. The patterns are presented to the network sequentially in an iterative manner. Appropriate weight corrections are performed during the process to adapt the network to the desired behaviour. The iterative procedure continues until the correction weight values allow the network to perform the required mapping; each

iterative presentation of the whole pattern set is named an epoch. The minimization of the error function is carried out using a gradient descent technique. The necessary corrections to the weights of the network for each iteration n are obtained by calculating the partial derivative of the error function in relation to each weight w_{jk} , which gives a direction of steepest descent. A gradient vector representing the steepest slope in direction of weight space is obtained. The weight update value δw_{jk} uses the negative of a gradient vector to perform a minimization. Based on the gradient direction, the delta rule will determine the required amount of weight update along with step size:

$$\delta w_{jk} = -\eta \frac{\delta E}{\delta w_{jk}} \quad (6)$$

The parameter η represents the step size and is called the learning rate.

We have the weight change in the hidden layer equal to

$$\delta w_{ij} = \eta \delta_j O_i \quad (7)$$

The δ_k for the output units can be calculated using directly available values since the error measure is based on the difference between the desired t_k and the actual o_k values. However, this measure is not available for the hidden neurons. The solution is to back-propagate the δ_k values layer by layer through the network so that finally the weights are updated. A momentum term was introduced in the BP algorithm by Rummelhart. The idea consists in incorporating in the present weight update some influence of the past iterations. The delta rule thus becomes

$$\delta w_{ij}(n) = -\eta \frac{\delta E}{\delta w_{ij}} + \alpha \delta w_{ij}(n-1) \quad (8)$$

where α is the momentum parameter and it determines the amount of influence from the previous iteration on the present one. It introduces a ‘damping’ effect on the search procedure, thus avoiding oscillations in irregular areas of the error surface by averaging gradient components with opposite sign and accelerating the convergence in long flat areas. In some situations, it possibly avoids the search procedure from being stopped in local minima. It may be considered as an approximation to a second-order method as it uses information from the previous iterations. In some applications, it has been shown to improve the convergence of the BP algorithm [18, 19].

3.3d Implemented neural network architecture: The neural network architecture consists of three layers: the input layer, the hidden layer and the output layer. The input layer consists of 16384 neurons, which is because the originally pre-processed samples are converted to Glyph images of size 128×128 . The output layer is composed of only 6 neurons which represent only 0 or 1, i.e., binary representation; ‘tansig’ and ‘logsig’ transfer functions are used

because their output range is 0–1 and perfect for learning to give output as Boolean values. The matched character is in the form of binary digits, which can be decoded from values stored in 6-digit binary numbers. The 6 output neurons have the capability for representing a total of 63 characters according to binary calculations.

A hidden layer of 100 neurons was finally selected after testing on different layer sizes for its optimum results because many numbers of neurons will increase the chances of overfitting. Characters were resized, normalized and formed into vectors to feed-in the network for training. Figure 5 presents the neural network architecture. In our current work, the same neural network architecture is used in three different ways: i.e., the training samples are used with three different HOG cell sizes (6×6 , 7×7 and 8×8) to study the effects on accuracy.

Training set: A total of 56 different characters/classes with 100 handwritten samples each were collected, out of which 75 samples for each of the characters were used for training the neural network. Out of the 75 samples for each of the classes, 80% are used for training, 10% are used for validation and remaining 10% are used to test during the regression. The 57 different classes of Meetei-Mayek characters considered in the current work comprises 43 alphabets and 10 digits as shown in figure 1(a).

As described earlier, training of the neural network was done in two passes during each iteration – forward pass and backward pass. In the forward pass, the input signals are propagated from the neural network input layer to the output layer. In the reverse pass, error signals generated at the output layer are propagated backward through the network for adjusting the weights of the neurons. The training of the neural network is done using a training function that updates weight and bias values according to gradient descent momentum and an adaptive learning rate.

The performance goal was set to 0.0005. Due to constant optimization during the regression phase, the neural network with HOG cell size 6×6 converges to its maximum accuracy in just 123 epochs with a training time of 2 min and 34 s. Secondly, the neural network that was trained using Glyph images of HOG cell size 7×7 took 190 iterations or epochs, in 5 min and 17 s to converge to its maximum accuracy. Lastly, for the HOG cell size 8×8 the solutions converged to their maximum accuracy in 117

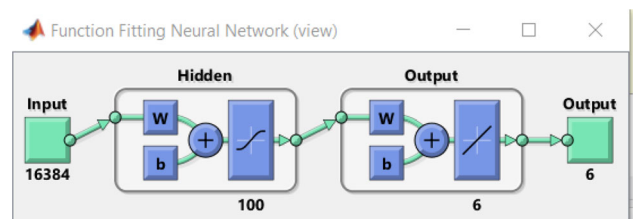


Figure 5. Neural network for 20×20 pixel feature size.

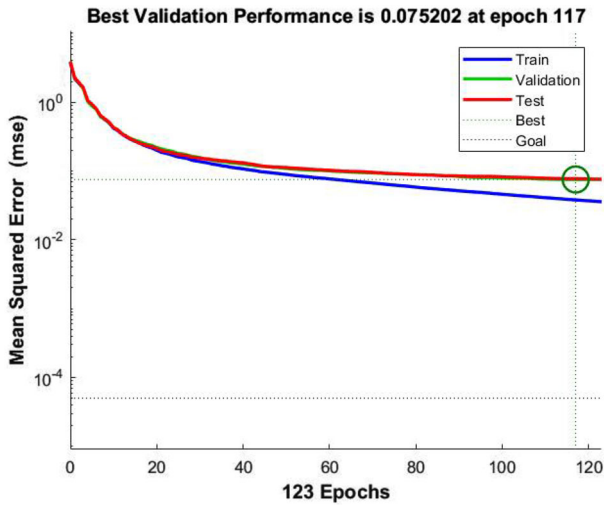


Figure 6. Mean squared error (mse) versus epochs for HOG cell size 6×6 with MLPNN.

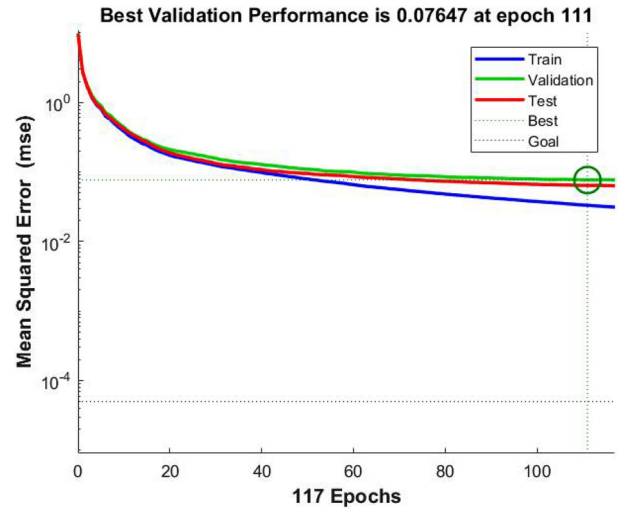


Figure 8. Mean squared error (mse) versus epochs for HOG cell size 8×8 with MLPNN.

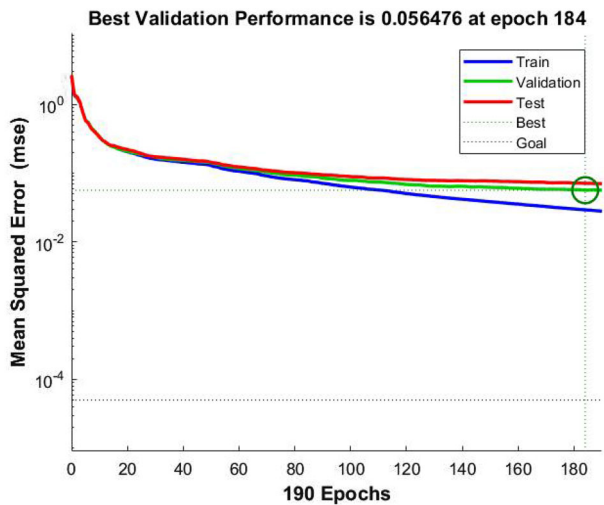


Figure 7. Mean squared error (mse) versus epochs for HOG cell size 7×7 with MLPNN.

epochs in just 1 min and 8 s. The mean squared error versus epochs for the networks trained with HOG cell sizes 6×6 , 7×7 and 8×8 is shown in figures 6, 7 and 8, respectively.

3.4 MMM recognition using linear SVM classifier with HOG feature descriptors

The current section provides a deep analysis of how features can be extracted from HMMM using multiple cell sizes HOG descriptors and then using them to train a classifier for efficient recognition. A linear kernel type SVM classifier has been used in the current training tasks because of its speed and reliability. The following subsections explain the basic terminologies involved, implementation and performance of the proposed approach.

SVM is a classifier separating classes in feature space; it is used to identify a set of linearly separable hyperplanes, which are linear functions of the feature space. Among the separable hyperplanes, only one hyperplane is chosen and placed such that the distance between the classes is maximum. SVM has a very high accuracy rate for two-class problems but it can be also modified to classify multiclass problems. If a classifier works with a large number of adjustable parameters and therefore large capacity, it probably learns the training set without error. The effective number of parameters is adjusted automatically to match the complexity of the problem [20]. The equation $w^t x + b = 0$ is a hyperplane separating two classes. Let us consider (X_i, Y_i) for $i = 1, 2, 3, \dots, N$ denoting the training dataset, where Y_i is the training data of X_i . There may be numerous hyperplanes that can separate the two classes, but the aim of SVM is to find the one that gives equal and maximum margin from both the classes. Mathematically, the aim of SVM is to maximize the objective function $L(\alpha)$ given by

$$L(\alpha) = \sum \alpha_i - \frac{1}{2} \sum_{i=1}^N \alpha_i \alpha_j N_j N_i = Y_i Y_j \phi(X_i)(X_j) \quad (9)$$

subject to the constraint

$$\sum_{k=1}^N \alpha_i Y_j = 0, \quad 0 \leq \alpha_i \leq C \quad \forall i \quad (10)$$

where C is the cost parameter that determines the cost caused by constraint violation, α_i is the hyper-parameter and $\phi(\cdot)$ is the feature mapping function. Asking for the maximum-margin linear separator in Eq. (17) leads to standard quadratic programming (QP) problems. With the mentioned constraints, the QP solution leads to the following classification function for SVMs:

$$Y = \text{sgn}(W\phi(Z) + b) \quad (11)$$

$$Y = \text{sgn}\left(\sum_{i=1}^q \alpha_i Y_i (X_i Z + b)\right) \quad (12)$$

where α_i is the Lagrange multiplier assigned to each training data, whose value depends on the role of training the data in the classifier system. The non-zero values of α_i correspond to the support vectors that are used to construct the classifier in (20); ‘ q ’ denotes the number of support vectors. If the feature functions $\phi(\cdot)$ are chosen with care one can calculate the scalar products without actually computing all features, thereby greatly reducing the computational complexity. In SVM the learning algorithms require only dot products between the vectors in the original input space, and the mapping is chosen such that these high-dimensional dot products can be computed within the original space means of a kernel function, also called ‘kernel trick’ [20]:

$$K(x, x_i) = \phi(x) \cdot (x_i). \quad (13)$$

4. Experimental results and evaluation

The current section highlights the experimental results and evaluation in detail for the two implementation strategies mentioned earlier. Finally, the comparison subsection also verifies the selection of a suitable type of feature extractor and classifier for efficiently recognizing HMMM characters.

4.1 Experimental results and error analysis for MLPNN with HOG feature descriptors

The experimental results described in the current section are a follow-up of the procedure explained in section 3.3. As described earlier, the network was tested against each of the remaining 25 samples from each of the 56 classes of the script for three different HOG cell sizes, i.e. 6×6 , 7×7 and 8×8 . Table 2 shows the success percentage against each of the characters.

Table 2. Tabulation of the resulting accuracy (%) and time taken for training the classifier in seconds.

Cell size	Accuracy (%)	Training time (s)
6×6	68.61	154
7×7	75.57	317
8×8	66.79	68

The bold values indicate the best values in comparison to the remaining ones.

Using HOG cell size 6×6 : It can be seen from tables 4 and 5 that most of the characters have very low accuracy, i.e. as low as 4% for characters like ‘**o** (0)’ and ‘**᳚**(KHOU)’, whereas characters that have only about 30–50% accuracy rates are ‘**᳚** (SAM)’, ‘**᳚** (MIT)’, ‘**᳚** (CHEEN)’, ‘**᳚** (THOU)’, ‘**᳚** (EE)’, ‘**᳚** (DIL)’ and ‘**᳚** (4)’. Other characters whose accuracy are low and in the range 50–80% include ‘**᳚** (1)’, ‘**᳚** (2)’, ‘**᳚** (3)’, ‘**᳚** (4)’, ‘**᳚** (6)’, ‘**᳚** (8)’, ‘**᳚** (KOK)’, ‘**᳚** (LAI)’, ‘**᳚** (PA)’, ‘**᳚** (NGOU)’, ‘**᳚** (WAI)’, ‘**᳚** (PHAM)’, ‘**᳚** (ATIYA)’, ‘**᳚** (RAAD)’, ‘**᳚** (BAA)’, ‘**᳚** (GHOU)’, ‘**᳚** (DHOU)’, ‘**᳚** (KOKLONSUM)’, ‘**᳚** (LAILONSUM)’, ‘**᳚** (MITLONSUM)’, ‘**᳚** (PALONSUM)’, ‘**᳚** (NGOULONSUM)’, ‘**᳚** (EELONSUM)’, ‘**᳚** (INAP)’, ‘**᳚** (SOUNAP)’, ‘**᳚** (YETNAP)’, ‘**᳚** (OTNAP)’, ‘**᳚** (CHEINAP)’, ‘**᳚** (NUNG)’, ‘**᳚** (QUESTION MARK)’, ‘**᳚** (COMMA)’ and ‘**᳚** (FULLSTOP)’, whereas the remaining characters showed fair level to good level of accuracy. The overall accuracy achieved is 68.61%.

Using HOG cell size 7×7 : Tables 4 and 5 show that the use of HOG cell size 7×7 results in increased accuracy in comparison with its 6×6 counterpart. It can be seen that there are more characters for which the accuracy has crossed 80%. However, there are three characters for which the accuracy is below 50%: ‘**o** (0)’, ‘**᳚** (MIT)’ and ‘**᳚** (CHEEN)’. The characters for which the accuracy lies between 51% and 80% are: ‘**᳚**(1)’, ‘**᳚**(2)’, ‘**᳚**(5)’, ‘**᳚**(8)’, ‘**᳚**(KOK)’, ‘**᳚**(SAM)’, ‘**᳚**(LAI)’, ‘**᳚**(MIT)’, ‘**᳚**(PA)’, ‘**᳚**(NA)’, ‘**᳚**(CHEEN)’, ‘**᳚**(KHOU)’, ‘**᳚**(THOU)’, ‘**᳚**(UN)’, ‘**᳚**(EE)’, ‘**᳚**(PHAM)’, ‘**᳚**(ATIYA)’, ‘**᳚**(RAAD)’, ‘**᳚**(DIL)’, ‘**᳚**(GHOU)’, ‘**᳚**(LAILONSUM)’, ‘**᳚**(PALONSUM)’, ‘**᳚**(NALONSUM)’, ‘**᳚**(NGOULONSUM)’, ‘**᳚**(EELONSUM)’, ‘**᳚**(INAP)’, ‘**᳚**(SOUNAP)’, ‘**᳚**(YETNAP)’, ‘**᳚**(OTNAP)’, ‘**᳚**(CHEINAP)’, ‘**᳚**(NUNG)’ and ‘**᳚**(FULLSTOP)’. The overall accuracy achieved is 75.57%.

Using HOG cell size 8×8 : Tables 4 and 5 show that when HOG cell size 8×8 is used it results in a slight decrease in individual accuracy as compared with that of 7×7 . It can be seen that there are more characters for which the accuracy has crossed 80%. However, there are some characters for which the accuracy is below 50%: ‘**o** (0)’, ‘**᳚** (SAM)’, ‘**᳚** (MIT)’, ‘**᳚** (NA)’, ‘**᳚** (CHEEN)’, ‘**᳚** (KHOU)’, ‘**᳚** (THOU)’, ‘**᳚** (EE)’ and ‘**᳚** (YETNAP)’. The characters for which the accuracy lies between 51% and 80% are ‘**᳚** (1)’, ‘**᳚** (2)’, ‘**᳚** (3)’, ‘**᳚** (4)’, ‘**᳚** (9)’, ‘**᳚** (KOK)’, ‘**᳚** (LAI)’, ‘**᳚** (PA)’, ‘**᳚** (TIL)’, ‘**᳚** (NGOU)’, ‘**᳚** (WAI)’, ‘**᳚** (UN)’, ‘**᳚** (DIL)’, ‘**᳚** (GHOU)’, ‘**᳚** (DHOU)’, ‘**᳚** (LAILONSUM)’, ‘**᳚** (PALONSUM)’, ‘**᳚** (NALONSUM)’, ‘**᳚** (TILLONSUM)’, ‘**᳚** (NGOULONSUM)’, ‘**᳚** (EELONSUM)’, ‘**᳚** (SOUNAP)’, ‘**᳚** (OTNAP)’, ‘**᳚** (CHEINAP)’, ‘**᳚** (NUNG)’ and ‘**᳚** (FULLSTOP)’. The overall accuracy achieved is 66.78%.

4.1a Evaluation: The current work has demonstrated the application of MLP networks with HOG descriptors for HMMM character recognition problem. It can be seen that the accuracies are greatly affected by the use of HOG cell sizes; choosing a suitable value of HOG cell size is a must. Maximum accuracy can be seen when HOG cell is set to 7×7 . The total time it took to train the network was 5 min and 17 s, and the solution converged in 190 iterations as highlighted earlier. The lower percentage of successful character recognition is because of the roundedness, starting and finishing style of the Meetei-Mayek characters. Finally, it can be learnt from the current application that training a model from a very complex dataset using neural network is too computationally intensive, which means that it will be slow on low-end PCs, or machines without math co-processors. However, processing speed alone is not the only factor in performance and neural networks do not require the time programming and debugging or testing assumptions that other analytical approaches do. Therefore, there is a need to improve the performance of our system using a more robust feature and pattern recognition system like SVM in the classification phase. The limitations of the current character recognition tasks such as speed and accuracy are alleviated with the help of HOG descriptors combined with linear SVM.

4.2 Experimental results and evaluation using multiple-HOG-feature vector with multiclass linear kernel SVM classifier

The current section describes the experimental results of the HMMM character recognition operation using multiple-HOG-feature vector with multiclass linear kernel SVM classifier as described in section 3.4. The use of linear SVM classifier returned a fully trained multiclass, error-correcting output codes (ECOC) model using the training features or HOG descriptors and the class labels in the HOG feature. The *One-versus-one coding scheme* was employed. In this scheme, for each binary learner, one class is positive, another is negative and the software ignores the rest. This design exhausts all combinations of class pair assignments. The number of binary learners is $K(K-1)/2$, where K is the number of unique classes of labels [21, 22]. In the current study, a handwritten character recognition for Meetei-Mayek script based on HOG feature descriptors and trained by SVM linear kernel is successfully implemented. Three different values of cell sizes have been considered, which were examined for accuracy by training the linear SVM classifier individually. Table 3 shows the time taken to train the linear SVM classifiers and the accuracy achieved in each case. Testing of the classifier was performed using the remaining 20 samples from each of the 56 classes of the script; the individual performance or the success rates are recorded in tables 4 and 5. Some of the characters that are marked

Table 3. Tabulation of the resulting accuracy (%) and time taken for training the classifier in seconds.

Cell size	Accuracy (%)	Training time (s)
6×6	96.928	67.91
7×7	94.339	58.86
8×8	93.714	50.99

The bold values indicate the best values in comparison to the remaining ones.

with an asterisk (*), viz. ‘𑜃𑜂𑜆𑜫 (PA)’, ‘𑜃𑜂𑜆𑜫 (KHOU)’ and ‘𑜃𑜂𑜆𑜫 (WAI)’ in table 4, have very low accuracy in comparison with other characters. For the ‘𑜃𑜂𑜆𑜫 (PA)’ character the accuracy increased drastically from 40% to 96%, which is promising. However, the worst recognition rate is achieved in case of ‘Khou’, in which the accuracy starts from just 20% and ends at a maximum of 52%. While most of the characters need to be worked on for better efficiency, some other characters like ‘LAI’ and ‘THOU’ also need an increase in accuracy. Despite the low accuracy readings mentioned earlier, there are also 14 cases where the 100% accuracy holds for all cell sizes, viz. - ‘𑜃𑜂𑜆𑜫 (7)’, ‘𑜃𑜂𑜆𑜫 (8)’, ‘𑜃𑜂𑜆𑜫 (SAM)’, ‘𑜃𑜂𑜆𑜫 (PHAM)’, ‘𑜃𑜂𑜆𑜫 (GOK)’, ‘𑜃𑜂𑜆𑜫 (RAAD)’, ‘𑜃𑜂𑜆𑜫 (BAA)’, ‘𑜃𑜂𑜆𑜫 (ATAP)’, ‘𑜃𑜂𑜆𑜫 (UNAP)’, ‘𑜃𑜂𑜆𑜫 (YETNAP)’, ‘𑜃𑜂𑜆𑜫 (NUNG)’, ‘= (QUESTION MARK)’, ‘| (COMMA)’ and ‘|| (FULLSTOP)’. Table 5 also shows a huge variation in training time extending from 50.99 up to 67.91 s, which is due to the varying length of HOG feature sizes for each cell size. Thus, it can be aptly stated that offline handwritten Meetei-Mayek can be most suitably used with HOG feature vector of cell size 6×6 to achieve an accuracy of about 96.928%.

5. Comparison

This section is broken up into two major portions. In subsection 5.1 the current study, i.e. HOG features with SVM classifier, is being compared against the MLPNN with HOG descriptors. Subsection 5.2 shows a comparison of our results against some of the top performing character recognition works in MMM.

5.1 Comparison between the MLPNN with HOG descriptors and linear SVM with HOG descriptors classifiers

In this study, we propose a novel feature-extraction-cum-classification approach for recognizing HMMM by establishing that HOG features can play a crucial role in increasing the accuracy of classifiers. This objective is accomplished by

Table 4. Comparison of accuracy for each class of the MMM between MLPNN with HOG feature and linear SVM with HOG-feature-based classifiers.

Sl. no.	Class		Accuracy (%)					
			MLPNN with HOG			Linear SVM with HOG		
			HOG cell size			HOG cell size		
Mayek	Notation	6 × 6	7 × 7	8 × 8	6 × 6	7 × 7	8 × 8	
<i>Cheising Mayek (digits)</i>								
1	0	0	4	44	40	100	100	96
2	1	1	68	76	52	96	100	92
3	2	2	56	60	68	80	92	76
4	3	3	80	84	52	100	92	92
5	4	4	64	96	76	100	100	100
6	5	5	86	80	60	100	88	96
7	6	6	64	88	84	92	88	88
8	7	7	100	96	92	100	96	100
9	8	8	80	80	100	100	100	100
10	9	9	92	96	76	100	100	100
<i>Eeyek Eepie (main alphabets)</i>								
11	KOK	KOK	80	76	76	96	96	96
12	SAM	SAM	44	60	36	100	100	100
13	LAI	LAI	56	56	60	88	92	92
14	MIT	MIT	40	44	24	100	100	100
15	PA	PA	52	60	60	88	92	96
16	NA	NA	48	64	44	100	100	100
17	CHEEN	CHEEN	36	36	32	92	88	96
18	TIL	TIL	60	92	68	92	96	100
19	KHOU	KHOU	4	44	28	52	44	40
20	NGOU	NGOU	88	88	80	96	100	100
21	THOU	THOU	36	52	40	76	88	84
22	WAI	WAI	64	96	60	72	92	80
23	YANG	YANG	96	84	92	100	100	100
24	HUK	HUK	96	96	96	96	96	96
25	UN	UN	96	60	56	96	96	96
26	EE	EE	36	68	48	76	80	76
27	PHAM	PHAM	80	72	96	100	100	100
28	ATIYA	ATIYA	76	64	92	96	96	96





























comparing two different models as mentioned earlier: in the first model, HOG feature vectors are trained using MLPNN, while in the second model, HOG features are used for training a fast and reliable SVM classifier. The implementation strategies and the discussion for the two approaches are explained in detail in section 3.

The results obtained by training and testing our Meetei-Mayek database are well tabulated in tables 4 and 5 for comparison. A total of 56 different classes are being put up for comparison. The value recorded in the tables gives the accuracy achieved against each character, which is calculated by tabulating the number of recognition times for each character in multiple confusion matrices. A total of 6

confusion matrices were recorded for the tasks, 3 for the first method and 3 for the second method. For both the techniques, three different HOG cell sizes are used: 6 × 6, 7 × 7 and 8 × 8.

The average recognition rate for each of the classifiers as recorded in table 6 shows that the when HOG feature vectors extracted from MMM are simply trained by MLPNN, a maximum accuracy of about 75.57% can be realized using a cell size of 7 × 7. However, if these HOG vectors are trained using SVM classifier, a maximum accuracy of about 96.928% can be realized using a cell size of 6 × 6, which is greatly increased as compared with the MLPNN with HOG descriptor method. The level of

Table 5. Comparison of accuracy for each class of the MMM between MLPNN with HOG feature and linear SVM with HOG-feature-based classifiers.

Sl. no.	Class		Accuracy (%)					
			MLPNN with HOG			Linear SVM with HOG		
			HOG cell size			HOG cell size		
Mayek	Notation	6 × 6	7 × 7	8 × 8	6 × 6	7 × 7	8 × 8	
<i>Lom Eeyek (addl. alphabets)</i>								
29		GOK	88	96	92	100	100	100
30		JHAM	72	80	84	92	100	96
31		RAAI	76	76	80	100	100	100
32		BAA	80	88	88	100	100	100
33		JIL	84	96	96	96	92	96
34		DIL	40	52	60	84	84	84
35		GHOU	68	76	64	96	96	92
36		DHOU	76	88	52	92	84	76
37		BHAM	88	88	88	100	100	100
<i>Lonsum Eeyek (alphabets with short ending)</i>								
38		KOK-L	60	96	100	72	84	88
39		LAI-L	64	76	80	84	84	84
40		MIT-L	76	88	96	92	92	92
41		PA-L	72	80	68	92	96	92
42		NA-L	92	56	72	96	96	100
43		TIL-L	92	92	80	88	100	92
44		NGOU-L	60	56	72	92	96	88
45		EE-L	72	76	72	88	92	88
<i>Cheising Eeyek (vowel signs)</i>								
46		ATAP	92	96	100	100	100	100
47		INAP	76	76	88	96	92	100
48		UNAP	96	84	84	100	100	100
49		SOUNAP	64	68	52	96	96	96
50		YETNAP	64	56	48	100	100	100
51		OTNAP	64	88	80	100	100	100
52		CHEINAP	56	68	80	92	88	96
53		NUNG	80	84	80	100	100	100
<i>Khutam Eeyek (punctuation marks)</i>								
54		QUESTION MARK	72	80	88	100	100	100
55		COMMA	72	88	84	100	100	100
56		FULLSTOP	64	72	72	100	100	100

accuracy can still be improved using more number of training samples per class.

5.2 Comparison to other works on MMM

The current techniques, i.e. HOG features with SVM and the linear SVM with HOG features, are compared to four other research works on MMM. It can be seen that, previously, most of the work was performed either on the 10 numerals or on the 27-character decimals. However, for

developing a complete OCR platform, there is a need to classify all the classes of the script by a single baseline classifier and that too with a high level of accuracy. In this sense, we are the first to implement a recognition system that is able to classify all the 56 different classes of the script with a high level of accuracy. Table 7 shows a comparison of the proposed approach with some previous works in MMM character classification or recognition.

Numeral classification: Romesh *et al* [14] used around 700 and 300 training and testing samples, respectively, for classifying the 10 different numeral classes of the script.

Table 6. A comparison of accuracy and training time between MLPNN with HOG descriptors and HOG descriptors with SVM classifiers.

Algorithm	Feature size	Accuracy (%)	Training time (s)
MLPNN	6×6	68.61	154
MLPNN	7×7	75.57	317
MLPNN	8×8	66.79	68
HOG with SVM	6×6	96.928	67.91
HOG with SVM	7×7	94.339	58.86
HOG with SVM	8×8	93.714	50.99

The bold value indicates the best value in comparison to the remaining ones.

Table 7. Comparison of other works on MMM with the current work.

Method	Dataset		Samples for each class		Accuracy (%)
	No. of classes	Class type	Training	Testing	
Romesh <i>et al</i> [14]	10	Numerals	70	30	85
Maring and Dhir [11]	10	Numerals	600	120	89.58
Thokchom <i>et al</i> [15]	27	Characters	17	5	90.3
HOG with SVM	56	All classes	75	25	96.93
HOG with MLPNN	56	All classes	75	25	75.57

They achieved an overall accuracy of 85%. Maring and Dhir [11], on the other hand, proposed a different architecture and used around 6000 and 1200 training and testing samples, respectively, for classifying the same 10 different classes and achieved an accuracy of around 89.58%.

Character classification: Thokchom *et al* [15] developed a technique to categorize all the 27 different main alphabets of the script using around 459 and 135 training and testing samples, respectively. They achieved an overall accuracy of 90.3%.

Numeral, character, punctuation mark and additional letters: In the current paper, the use of HOG features with SVM classifier is studied in detail using different cell sizes and also tuned to classify all the 56 classes of the script. Seven different HOG cell sizes were used (i.e., 6×6 , 7×7 and 8×8). For each of them the same numbers of training and testing samples were used, i.e. 4200 and 1400, respectively. The overall accuracy of 96.93% with a cell size of 6×6 was achieved.

6. Conclusion

In this work, a novel approach for efficiently recognizing HMM characters is presented by means of comparison between the traditional BP-learning-based ANN with HOG descriptors and the multiple-cell-sized HOG descriptors with linear SVM classifier. About 5600 handwritten samples of the 56 different classes of the MMM were collected from a group of different people. The samples were then pre-processed to remove the noise in and around the letters followed by extraction of each letter from the group. The training and testing phase used three different HOG descriptors sizes: 6×6 , 7×7 and 8×8 . The maximum accuracy that we were able to achieve was 96.928% with a minimum training time of just 50.99 s.

Therefore, it can be stated that the complex Meeitei-Mayek characters can be efficiently recognized using the 6×6 cell-sized HOG descriptors with multiclass linear kernel SVM classifier.

References

- [1] Wangkhemcha C 2007 *A Short History of Kangleipak (Manipur) Part-II*. Sagolband Thangjam Leirak, Imphal, India: Kangleipak Historical and Cultural Research Centre
- [2] Mangang N K 2003 *Revival of a Closed Account, a Brief History of Kanglei Script and the Birth of Phoon (Zero) in the World of Arithmetic and Astrology*. Lamshang, Imphal: Sanamahi Laining Amasung Punshiron Khupham (Salai Punshipham)
- [3] Hodson T C 1908 *The Meitheis*. Delhi: Low Price Publications
- [4] Rani A, Rani R and Dhir R 2012 Combination of different feature sets and SVM classifier for handwritten Gurumukhi numeral recognition. *Int. J. Comput. Appl.* 47(18): 28–33
- [5] Arora S, Bhattacharjee D, Nasipuri M, Malik L, Kundu M and Basu D K 2010 Performance comparison of SVM and ANN for handwritten Devanagari character recognition. *Int. J. Comput. Sci. Issues* 7(3): 01–10
- [6] Sinha R M K 2009 A journey from Indian scripts processing to Indian Language processing. *IEEE Ann. Hist. Comput.* 31(1): 8–31
- [7] Pal U, Wakabayashi T and Kimura F 2007 Handwritten Bangla compound character recognition using gradient feature. In: *Proceedings of the 10th International Conference on Information Technology*, pp. 208–213
- [8] Sharma N, Pal U, Kimura F and Pal S 2007 Recognition of offline handwritten Devanagari characters using quadratic classifier. In: *Proceedings of the Indian Conference of Computer Vision, Graphics and Image Processing*, pp. 808–816
- [9] Basu S, Das N, Sarkar R, Kundu M, Nasipuri M and Basu D K 2005 Handwritten Bangla alphabet recognition using MLP based classifier. In: *Proceedings of the 2nd National Conference on Computer Processing of Bangla*, pp. 285–291

- [10] Plamondon R and Srihari S 2000 Online and offline handwriting recognition: a comprehensive survey. *IEEE Trans. Pattern Anal. Mach. Intell.* 22(1): 63–84
- [11] Maring K A and Dhir R 2014 Recognition of Chising Iyek/Eeyek-Manipuri digits using support vector machines. *Int. J. Comput. Sci. Inf. Technol.* 1(2): 1–6
- [12] Romesh L, Singh P B, Singh T S D, Anilkumar S and Singh A U 2014 A neural network based handwritten Meetei Mayek alphabet optical character recognition system. In: *Proceedings of the IEEE International Conference on Computational Intelligence and Computing Research*, pp. 1–5
- [13] Chandan J K and Sanjib K K 2013 Recognition of handwritten numerals of Manipuri script. *Int. J. Comput. Appl.* 84(17): 1–5
- [14] Romesh L, Singh A U, Singh N C, Singh A S and James H 2012 Simulation and modelling of handwritten Meitei Mayek digits using neural network approach. In: *Proceedings of the International Conference on Advances in Electronics, Electrical and Computer Science Engineering—EEC 2012*, pp. 147–150
- [15] Thokchom T, Bansal P K, Vig R and Bawa S 2010 Recognition of handwritten character of Manipuri script. *J. Comput.* 5(10): 1570–1574
- [16] Dalal N and Triggs B 2005 Histograms of oriented gradients for human detection. In: *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, pp. 1–8
- [17] Kresch R and Malah D 1998 Skeleton-based morphological coding of binary images. *IEEE Trans. Image Process.* 7(10): 1387–1399
- [18] LeCun Y, Boser B, Denker J S, Henderson D, Howard R E, Hubbard W and Jackel L D 1990 Handwritten digit recognition with a back-propagation network. In: Touretzky D (Ed.) *Proceedings of Advances in Neural Information Processing Systems 2 (NIPS*89)*. Denver, CO: Morgan Kaufmann, pp. 396–404
- [19] Blais A and Mertz D 2001 *An Introduction to Neural Networks Pattern Learning with Backpropagation Algorithm*. Gnosis Software, Inc.
- [20] Christianini N and Shawe-Taylor J C 2000 *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge, UK: Cambridge University Press
- [21] Escalera S, Pujol O and Radeva P 2009 Separability of ternary codes for sparse designs of error-correcting output codes. *Pattern Recognit. Lett.* 30(3): 285–297
- [22] Escalera S, Pujol O and Radeva P 2010 On the decoding process in ternary error-correcting output codes. *IEEE Trans. Pattern Anal. Mach. Intell.* 32(7): 120–134