Received September 7, 2021, accepted September 20, 2021, date of publication September 22, 2021, date of current version October 1, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3114799

A Thresholded Gabor-CNN Based Writer Identification System for Indic Scripts

M. F. MRIDHA[1], (Senior Member, IEEE), ABU QUWSAR OHI[1], JUNGPIL SHIN[2], (Senior Member, IEEE), MUHAMMAD MOHSIN KABIR[1], MUHAMMAD MOSTAFA MONOWAR[3], (Member, IEEE), AND MD. ABDUL HAMID[3]
[1]Department of Computer Science and Engineering, Bangladesh University of Business and Technology, Dhaka 1216, Bangladesh
[2]School of Computer Science and Engineering, The University of Aizu, Aizuwakamatsu 965-8580, Japan
[3]Department of Information Technology, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia

Corresponding authors: M. F. Mridha (firoz@bubt.edu.bd) and Jungpil Shin (jpshin@u-aizu.ac.jp)

ABSTRACT Writer identification is the procedure of identifying individuals from handwriting. Writer identification is a common interest in biometric authentication and verification systems, and numerous studies are available for English, Chinese, Arabic, and Persian specific handwriting. This paper introduces a supervised offline Indic script writer identification system that can identify individuals using less than a single page of handwriting. A lightweight Convolutional Neural Network (CNN) architecture fused with non-trainable Gabor filters is used as an identification model that can recognize writers from scarce data. For the experiment, we used BanglaWriting dataset, which is openly available for Bengali writing and writer recognition. Further, we added Devanagari and Telugu datasets for evaluation. The overall evaluation shows that the proposed thresholded Gabor-based CNN architecture performs superior to numerous deep CNN architectures for Indic writer recognition.

INDEX TERMS Image processing, convolutional neural network, Gabor filter, writer identification, Indic script.

I. INTRODUCTION

Handwriting biometrics is a division of biological phenomena that allows a person to be identified and authenticated based on a set of recognizable and verifiable features that are unique and specific among individuals [1]. Writer identification is primarily classified into two scenarios, offline and online [2]. Offline writer identification systems only receive handwritten images as features, whereas online writer identification systems receive more robust features, writing strokes, pen pressure, writing coordinates, among others. Although the usage of an online writer identification system is limited to electronic writing devices, offline writer identification can be applied to both electronic and paper-based handwriting.

Writer identification systems are also segmented into two categories based on the criterion of recognition procedure: text-dependent and text-independent recognition [3]. Text-dependent recognition systems can only recognize writers from a specific written word phrase which the system has previously seen as an example from the writer (or stored in the database). Signature verification systems are a variant of text-dependent writer recognition. In such a case, writers provide specific signatures which they have to write again for validation purposes. In contrast, text-independent systems can recognize any writing from the writers, which the system may not have previously observed by the writer. The advantage of text-independent systems is that the writers can be recognized from any written words or characters. In this paper, by writer recognition, we term the text-independent writer recognition systems.

Before the advent of machine learning systems, most writer identification and forensic techniques targeted various facets, such as the physics of writer's ink, the paper on which writing is written, the size of each letter, character-level spacing, word-level spacing, and so forth [4]. After the advancement of machine learning and deep learning methods, it is possible to identify writers using image processing techniques [2]. Although recent machine learning and deep learning methods perform excellently on writer identification, they bring out some new challenges: a) insufficient amount of training data,

The associate editor coordinating the review of this manuscript and approving it for publication was Larbi Boubchir.

b) noise in the image of writing, c) existing similar pattern among writers, d) risk of overfitting, e) risk of underfitting, f) occurrence of unknown writing patterns or strokes, and so on.

Some studies are found in the literature that developed language-specific writer identification systems for major languages, including English [5], Chinese [6], [7], Arabic [8], and Persian [9]. Indic scripts cover most scripts used in India, Bangladesh, Nepal, and Pakistan continental, such as Bengali, Devanagari, Gujarati, Gurmukhi, Telugu. In comparison, non-Indic scripts are English, Chinese, Persian, Thai, Arabic, Farsi, Latin, Roman, among others.

The complexity of writer identification and verification varies among different languages due to the individual shapes, strokes, number of characters belonging to the language, and the similarity between characters. To introduce the properties of the Indic languages, the following concepts of Bengali language writing should be acknowledged:

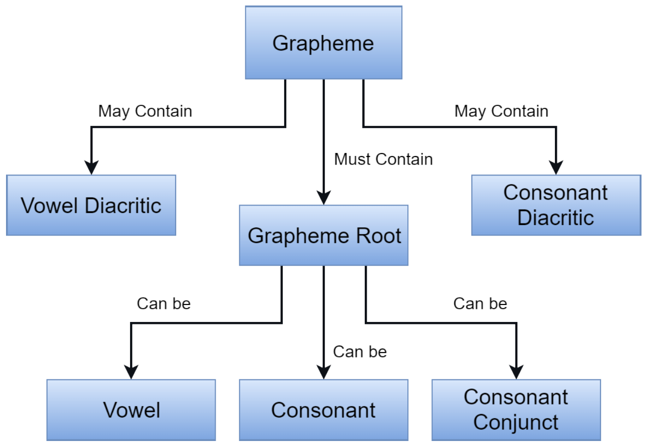


FIGURE 1. The structure of grapheme (the smallest unit of writing) in the Bengali language.

A. VOWEL

The letters or characters which represent the speech sound that can be uttered only with an open vocal tract are considered as vowels. The Bengali language has 11 vowels.

B. CONSONANT

The letters or characters that represent the speech sound that can be uttered with a complete or partial open vocal tract are considered as consonants. There are 39 consonants in the Bengali language.

C. DIACRITIC

The signs that are written above or below a character to define a different pronunciation of the same letter or character. In the Bengali language, there are two types of diacritics, vowel diacritic and consonant diacritic. There are 11 vowel diacritics and seven consonant diacritics.

D. CONSONANT CONJUNCTS

Consonant conjuncts are letters or characters that contain two or more joined consonants. In the Bengali language, 118 consonant conjuncts are primarily used.

E. GRAPHEME

Grapheme is the smallest unit of any writing system. In terms of the Bengali language, a grapheme must contain a grapheme root. A grapheme root is a letter that can be a vowel, a consonant or, a consonant conjunct. A grapheme can also contain diacritics, yet they are optional. A grapheme may contain at most one vowel diacritic and at most one consonant diacritic. Figure 1 exhibits the structure of Bengali grapheme. Figure 2 shows an example of a Bengali word construction.

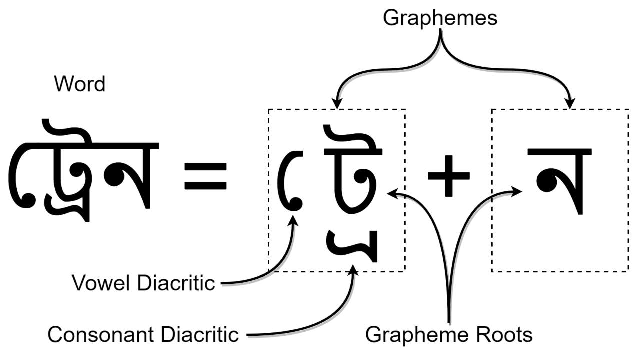


FIGURE 2. An example of a Bengali word structure. A Bengali word is constructed with one or more graphemes. Graphemes must contain a grapheme root, whereas containing diacritic is optional.

Due to the comprehensive collection of vowels, consonants, consonant conjuncts, and diacritics, the Bengali language has around 13,000 different grapheme variations [10]. These grapheme variations also introduce various combinations of character patterns. Compared to English, Bengali writing patterns are more complex due to the increasing number of character patterns, making Bengali writer identification a challenging task. As a result, image processing-based writer identification systems unaware of the overall patterns may produce less accuracy in the Bengali writer identification than English. Indic scripts are generated from ancient Bhramic script [11]. Therefore, like Bengali, Indic scripts, such as Tamil, Devanagari, Gujarati, among others, are packed with complex writing patterns, grapheme variations, complex constructions, and an approximately similar number of characters. The visual appearance of Devanagari, Gurumukhi, and Gujarati is nearly identical. Nevertheless, Devanagari is more circular, Gurumukhi includes diverse half-length vertical lines, and Gujarati contains extra loops. Furthermore, the visible appearance of Malayalam, Kannada, Tamil, and Telugu are also alike. Tamil and Kannada scripts are similar, while Malayalam scripts have a round shape. Concavities of most of the Telugu and Kannada characters are upwards while downwards for Malayalam and Tamil. Besides, the Telugu and Kannada characters have head marks on the above.

Nevertheless, the writing style and characters of Indic scripts are equivalent, with minor changes resulting in similarities in writing strokes and patterns.

This paper contributes to a Gabor filter-based Convolutional Neural Network (CNN) architecture as a Bengali writer identification system that can recognize writers from a small amount (less than a single page writing) of training data. The overall architecture is named Thresholded Gabor CNN (TGCNN), and it is trained with word-level black and white image data. Each word image is first passed through a convolution of thresholded Gabor filters (TGC). Further, the outputs are passed through a comparatively light CNN architecture that learns the writing patterns from the TGC extracted linear handwritten features.

The overall contribution of the paper includes:

- We introduce a Deep Convolutional Neural Network (DCNN) based writer recognition system, specifically for Indic script writer recognition systems.
- We demonstrate the overall construction of the Indic scripts based on the Bengali writing system at the word level. Moreover, we address the complexity of the Indic writer identification system based on the Bengali language.
- We exploit a non-trainable, thresholded Gabor filter fused with a CNN, which is used as a writer identification system.
- We carried out experiments implementing other CNN-based state-of-the-art methods on our dataset and validated that our approach performs better in most cases.

The rest of the paper is segmented as follows: Section II introduces a subset of recent approaches that are available in the field of offline writer identification. Section III motivates towards using Gabor filters for the writer recognition tasks. Section IV provides a detailed description of the proposed TGCNN architecture. Section V presents the performance evaluation of the proposed architecture. Finally, Section VI concludes the paper.

II. RELATED WORKS

In the phase of the offline writer identification system, most works relate to specific methods, which are but not limited to, the Hidden Markov Model (HMM), Gaussian Mixture Model (GMM), Gabor Filters, Deep Neural Networks (DNN), and CNN. Due to the robustness of deep learning approaches, a more significant share of modern systems is structured based on deep learning strategies [12].

Before the advancement of neural network architectures, Gabor filters were exploited for writer identification. Gabor filters are linear filters, which are used to obtain linear features from an image. Similar to Gabor filters, XGabor filters were also used to extract feature data. After the extraction of feature information from image data, most works performed wavelets [13], graph relations [9], statistical analysis [6], [14] and so on [15]–[18]. HMM-based methods have also been introduced in writer identification systems, where expert identifiers are built for individual writers [19]. Although these methods performed better, they are less accurate compared to the neural network methods, which are observed to work better for writer recognition pipeline [2].

Following the advancement of neural network architectures, new robust methods have been introduced in the field of writer identification. Chu and Srihari presented a word-based identification system that performs word-level feature learning by identifying similarities between two words, and the learned knowledge was further applied to identify writers [20]. They also collected annual writings and performed experiments on them. The limitation of the approach includes its reliance on DNN instead of CNN, which is preferable for image identification and image pattern recognition. Fiel and Sablatnig used a CNN to extract features vector [21]. The pre-trained CNN model on a known writers database is used as a feature vector. The last softmax classification layer of the CNN model is cut off after the pre-training phase and is used as a feature vector for further identification. Christlein *et al.* introduced a unique method that uses both CNN and GMM supervectors [22]. The research work defined CNN activations as a local descriptor and formed a global descriptor produced by GMM supervector encodings. Xing and Qiao introduced a multi-stream CNN method called DeepWriter along with a patch scanning technique [5]. Yang *et al.* developed a DCNN architecture as a writer identification system. They concluded that DCNN methods require a great amount of data to achieve better performance and introduced a writing augmentation technique named DeepStroke [23]. Chen *et al.* introduced a semi-supervised feature learning process that was mainly implemented using CNN. The proposed semi-supervised system employed a mixture of labeled and unlabeled data in the training phase and acquired better accuracy [24]. Christlein *et al.* introduced a CNN feature learning strategy that is fully unsupervised and uses ResNet as a baseline, and performs better on available datasets [25]. From the most recent works, it can be concluded that CNN is the most suitable state-of-the-art writer identification system.

Compared to non-Indic scripts, writer identification targeting Indic scripts has been less explored [26]. In addition, the datasets concerning writer recognition are mostly closed sources in previous research endeavors. Therefore, the scarcity of openly available datasets limits the scope of the research work in this domain. Strategies targeting basic deep learning-based pipelines are mainly investigated for Indic scripts [27], and more significant contribution is required.

Among the different Indic scripts, Bengali writer identification methodologies have been primarily investigated [26]. In [26], a few Bengali writer identification systems were observed and reviewed. However, previous research endeavors are conducted in closed-source datasets, resulting in a scarcity of Bengali writer recognition datasets [28]. In Bengali writer identification, efforts have been given on identifying isolated characters [29], [30]. Further, efforts have been investigated in word-level writer identification [31], as well as document-level writer recognition [28]. Biswas and Das used radon transform [32] for word-level writer identification system, evaluated on a group of 55 people. In contrast, Halder *et al.* introduced a document-level approach of writer recognition, one of the most promising writer recognition pipelines specifically developed for the Bengali language. The approach first converts images into binary black and white binary pixels and segments the handwritten portions of the datasets using image pixel statistics. After segmentation, the following statistical features extracted from the document: a) word-spacing, b) stroke-thickness, c) angular distance of word center, e) circularity and rectangularity ratio of words, f) height and width of words, g) fractal dimension analysis, and h) four diagonal (horizontal, vertical, left, right) stroke directions. These features were fed into a classifier for the final classification. However, most research strategies [28]–[30] involved text-dependent writer recognition systems, as both training and testing sets contain the same isolated characters and words. Therefore, the strategies introduced in the research endeavors are still outdated based on the current implementation of the computer vision system.

Apart from the Bengali language, efforts have been observed in Gurmukhi [33]–[35], Tamil [36], Telugu [37], and Devanagari [38], [39] scripts. The investigation related to the specific scripts targets statistical exploration of handwritten patterns and depends on machine learning analysis. Efforts have also been made to recognize writers based on Indic scripts. Reddy *et al.* [40] implemented a regressor for extracting stroke vectors in a vector space to generate a cluster space for writer stroke patterns. Kumar *et al.* [27] introduced a general CNN architecture for the Indic writer identification system, evaluated on a close-sourced Devanagari dataset. The writer recognition systems targeting Indic scripts are still advancing, requiring an extensive investigation. Hence there are opportunities to improve both architectural and text-independent writer identification systems.

This paper introduces a writer identification scheme that explicitly targets Indic handwriting scripts. The proposed method uses CNN with Gabor filters as an identification system. The usage of Gabor filters with CNN boosts the accuracy of standard CNN architecture in classic image identification systems [41]. Although CNN fused Gabor filters are trainable, it is mostly suitable for high-end image classification tasks. However, for writer identification, instead of learning the Gabor hyperparameters through the backpropagation technique, the paper introduces non-trainable Gabor filters to find linear orientations. The non-trainable Gabor filters extract specific linear textures from handwritings and pass the handwriting image textures to CNN that performs feature detection. Although the end-to-end architecture of the proposed system is supervised, the system requires less handwriting data from traditional CNN-based methods. The challenges include attaining higher accuracy with a limited amount of data. In addition, CNN architectures often tend to overfit while being trained on a limited amount of training data. Again, with the difficulty of overfitting and minimal training data, the dataset also contains a large number of unique word images. As a result, a generalized architecture is to be implemented to attain satisfactory accuracy on the Indic script datasets.

III. MOTIVATION

Gabor filters are named after Dennis Gabor, which is used for numerous image processing tasks, including edge-detection [42], image pre-processing, texture analysis [43], feature extraction, feature modification. The procedure of Gabor filters is similar to the human vision system and excellent for texture analysis tasks [44]. Gabor filters can separate texture information from images, which can also eliminate some specific texture information. Gabor filters are mutated and used for numerous image processing purposes, including iris-recognition [45], face recognition [46], speech recognition [47], vein recognition [48].

In general, a CNN learns the filter representation used to determine the availability of certain features in a given input image. CNN filters can recognize complex and higher-dimensional features from images. However, they suffer from overfitting issues and hinder generalization capability if the number of training samples is relatively low [49]. Therefore, training a DCNN architecture on handwriting data which includes less than a single page writing of an individual, causes overfitting (reported in Section V: Tables 2, 3, and 4).

The overfitting on small training datasets can be solved using numerous strategies, among which data-augmentation [50], regularization [49], and transfer learning [51] are well-recognized. These strategies increase generalization by achieving reducing overfitting issues. Data-augmentation and transfer learning are data-dependent approaches, whereas regularization is a model-dependent approach. Transfer learning consists of numerous strategies, including feature-based approach, parameter-based approach, instance-based approach, and so on [51]. In the most straightforward transfer strategy, the model is trained on a different dataset, and the learned experience (model weights) is further used to learn representation from a different dataset. Further, the preceding layers of the DCNN models are frozen (not updated) in the transfer learning approach. The approach is similar to non-trainable convolution, as they are never trained in the final training procedure of transfer learning, and they often produce better results.

Gabor filters are excellent for obtaining texture features. Hence in this study, they are implemented to recognize writers from word-level images. Instead of using trainable CNN in the initial layer of the DCNN, a non-trainable Gabor filter is implemented. This property is similar to the non-trainable weights of transfer learning approaches. Implementing non-trainable Gabor filters for writer recognition systems results in the following advantages:

- Achieves better accuracy while trained on small datasets.

- Avoids overfitting and provides better generalization on unseen data.
- The requirement of the number of CNN layers is comparatively low based on the current implementations of deeper CNN architectures.

Gabor filters merged with CNN layers achieved superior performance in the case of Indic scripts. Indic scripts mostly contain non-linear stroke patterns. However, in English, which is a non-Indic script, the stroke patterns are mostly linear. Therefore, Gabor filters only activate the subsequent CNN layers to learn the primary linear strokes, which results in an imbalance of accuracy benchmark compared to Indic scripts. In the following section, the overall methodology for Indic scriptwriter identification is introduced.

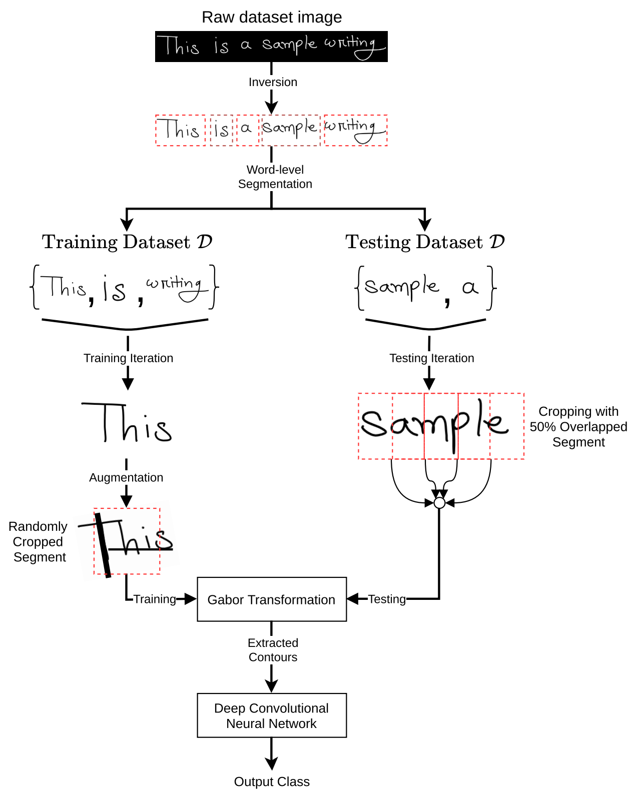


FIGURE 3. The overall pipeline of the proposed TGCNN architecture. The processes include image pre-processing, augmentation, random image cropping, Gabor transformation through thresholded Gabor convolution, and deep convolutional neural network.

IV. METHODOLOGY

The writer identification system uses a traditional CNN architecture that is fused with a modified Gabor filter. This section presents a thorough walkthrough of the proposed Indic writer identification system. Figure 3 provides an overview of the complete workflow of the introduced approach. The system firstly inputs normalized images, discussed in Section IV-A. Consequently, the inputs are passed to the first layer of the network, performing Thresholded Gabor Convolution (TGC), discussed in Section IV-B. The TGC-processed outputs are passed to a CNN architecture discussed in Section IV-C. As the architecture receives fixed-size images, the training and testing of random size word-level images require modifications, which are finally discussed in Section IV-D.

A. IMAGE PRE-PROCESSING

The architecture is inputted 110×110 sized cropped word-level images. The images are converted into black and white, where white portions of the images are presented by zero in the image matrix. Throughout the research investigation, it is examined that if the datasets are scaled in $[-1, 1]$ range, the Gabor kernel fails to adapt input features. The proposed Gabor convolutional layer only contains positive values. Therefore convolving a positive weight matrix with a negative scale input causes the resulting image to be negative. Hence, often the output image does not pass through the activation function. In contrast, scaling input image between $[0, 1]$ provides a better passthrough the activation functions and results better accuracy.

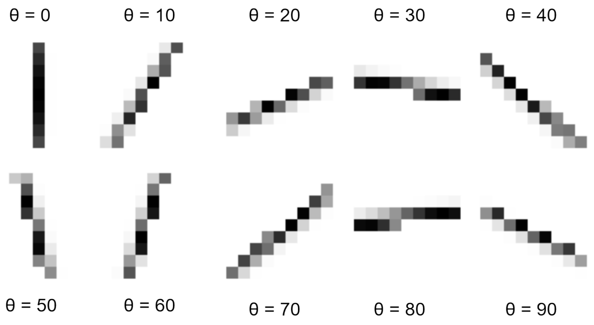


FIGURE 4. The 9 × 9 kernels of the non-trainable Gabor filters, where $\theta = \{0, 10, 20, 30, 40, 50, 60, 70, 80, 90\}$ respectively.

B. THRESHOLDED GABOR CONVOLUTION

Gabor filters are two-dimensional matrices that extract specific directional frequency content from an image. A Gabor filter is a complex Gaussian sinusoid modulation consisting of real and imaginary values, represented as follows,

$$\mathbf{R}eal : g(x, y, \lambda, \theta, \psi, \sigma, \gamma) = exp(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}) \times cos(2\pi\frac{x'}{\lambda} + \psi) \quad (1)$$

$$\mathbf{I}maginary : g(x, y, \lambda, \theta, \psi, \sigma, \gamma) = exp(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}) \times sin(2\pi\frac{x'}{\lambda} + \psi) \quad (2)$$

where,

$x' = xcos\theta + ysin\theta$
$y' = -xsin\theta + ycos\theta$
$\lambda =$ Wavelength of the sinusoidal factor
$\theta =$ Orientation of the parallel stripes
$\psi =$ Phase offset
$\sigma =$ Standard deviation of Gaussian envelope
$\gamma =$ Spatial aspect ratio

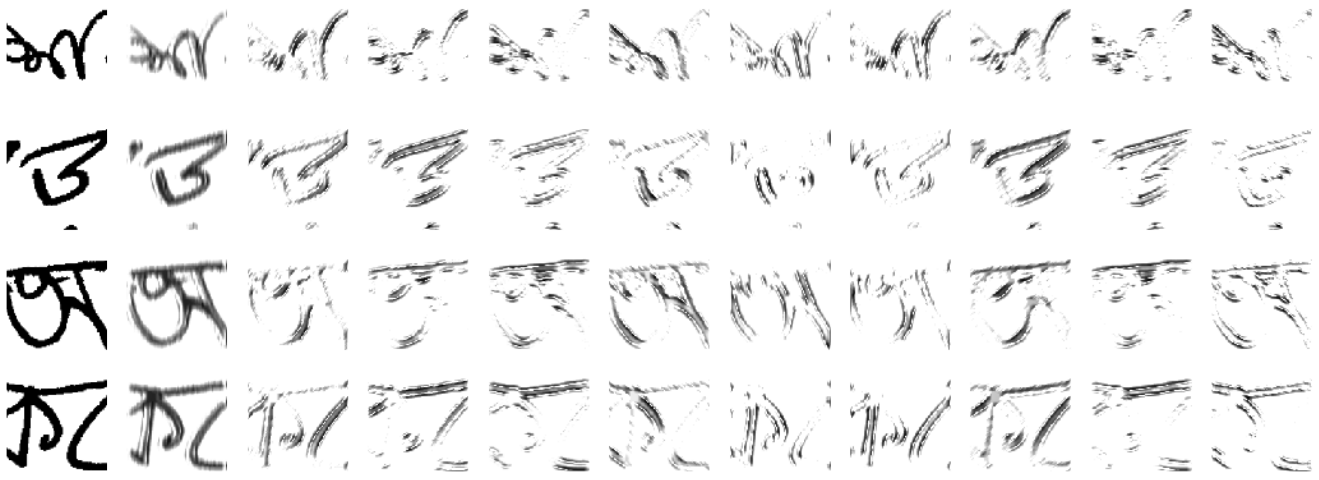


FIGURE 5. The left column is the input image. The rest of the columns are the outputs of Gabor kernel convolution (explained in Section IV-B) along with $\theta = \{0, 10, 20, 30, 40, 50, 60, 70, 80, 90\}$ respectively.

The Gabor filters are used as convolution filters in the CNN layer. The convolution operation of CNN is most suitable for calculating real integer values. Also, current processing units and applications do not adapt complex-number calculations. For the sake of avoiding complex-number calculations from CNN, instead of using the imaginary part, the real part of Gabor filters is used, as stated in Equation 1. The general Gabor filters still encounter noises (negative values), excluding features from input images through the convolution operation. Hence, the negative values of Gabor filters are eliminated by passing the Gabor filter matrix through a threshold function stated as,

$$Threshold(x_{i,j}) = \begin{cases} x_{i,j} & \text{if } x_{i,j} \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The thresholded Gabor filters are used as the kernels of convolutional operations. The Gabor filter is initialized by setting $\lambda = 102$, $\psi = \frac{\pi}{2}$, $\sigma = \frac{1}{2}$, and $\gamma = \frac{1}{10}$. As Gabor filters are isomorphic, the value of θ is set in range of [0, 90], which determines is the angel of stroke of Gabor filter. Figure 4 illustrates thresholded Gabor filters in different orientation w.r.t θ. In the proposed TGCNN baseline, a 9×9 size of convolution filters are used. 46 filters are used in the Gabor filter convolution where each filter represent different value of θ, in the pattern $\{0, 2, 4, \ldots, 88, 90\}$.

The thresholded Gabor convolution exploits the contours of handwritten strokes in different orientations. Figure 5 illustrates a scenario after particular handwriting is convolved using thresholded Gabor filters at different angles. The Gabor convolution extracts the outlines of writing in different orientations, which helps the lower-end of the architecture to recognize stroke patterns instead of directly identifying individual character-based features.

C. BASELINE STRUCTURE

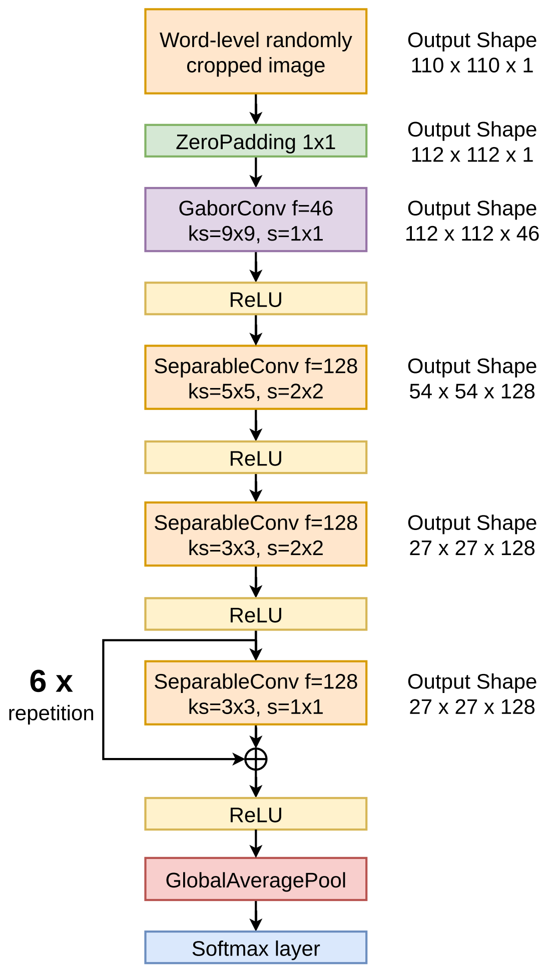
Handwriting images can be recognized using black and white images, whereas general image recognition architectures consider color features. CNN architectures can recognize geometric shapes and patterns from images [52]. Hence, CNN architectures are also suitable for identifying geometric features of handwriting.

Although the Gabor filter convolution can be used on any specific DCNN baseline, a lightweight residual CNN architecture is introduced. The overall architecture is presented in Figure 6. The architecture is initially trained using randomly cropped images. As the cropped images contain unfurnished pen-strokes, padding the input images with zero paddings adds extra furnish to the input image.

After applying TGC, the baseline implements general convolution on the TGC outputted images. For performing convolutions, separable convolutional operations are chosen. A separable convolution includes a depthwise convolution, followed by a pointwise convolution, which reduces trainable parameters with faster processing time [53]. However, for this specific case, the pointwise convolution is performed before the depthwise convolution, which is introduced in Xception architecture [54]. In the first two separable convolution operations, a stride of 2 is used to downsample the resolution of the input features.

Deeper CNN architectures tend to work better for generalization from training dataset [55]. In addition, residuals [56] are excellent for generalization, and identity mapping [57] of deeper CNN architectures. Hence, this paper introduces residuals with a custom depth CNN network. After applying TGC and two Separable convolutions, the rest of the CNN layers are connected through skip connections (residuals). The number of residual layers is kept variable (may change based on the complexity of the dataset), and the default number of residual layers is chosen to be six.

As an activation function, ReLU is implemented after every convolution operation. BatchNormalization [58] is attached to every convolution block to reduce the covariance shift, which gives better results in general residual



Word-level randomly cropped image
Output Shape 110 x 110 x 1
ZeroPadding 1x1
Output Shape 112 x 112 x 1
GaborConv f=46 ks=9x9, s=1x1
Output Shape 112 x 112 x 46
ReLU
SeparableConv f=128 ks=5x5, s=2x2
Output Shape 54 x 54 x 128
ReLU
SeparableConv f=128 ks=3x3, s=2x2
Output Shape 27 x 27 x 128
ReLU
6 x repetition
SeparableConv f=128 ks=3x3, s=1x1
Output Shape 27 x 27 x 128
ReLU
GlobalAveragePool
Softmax layer

FIGURE 6. The workflow of TGCNN architecture. The randomly cropped images are zero-padded with a border of one horizontally and vertically, and then Gabor filters are convolved. Two separable convolutions perform downsampling of the image. Further, a block of separable convolution with residual is repeated six times, finally passing the output to a global average pool, followed by a softmax layer. A batch normalization layer is added after each convolution operation. The abbreviations of the graph are: *ks* = kernel size, *s* = stride, *f* = number of filters.

architectures [56]. The pattern Convolution → Activation → BatchNormalization is followed for each convolution segment. A general cross-entropy loss is implemented to train the overall architecture.

D. TRAINING-TESTING STRATEGY

The architecture is trained using fixed-shaped word-level handwriting images. However, as word-level handwriting can be of different widths, a modified training-testing strategy is implemented. These techniques are discussed below. The input words are kept at the height of 110 while keeping the aspect ratio of the width unchanged.

1) RANDOM IMAGE CROPPING

A randomly cropped image of 110 × 110 is used from every word-level image to train the TGCNN architecture. Random image cropping is also a data augmentation process proven to improve the overall accuracy and reduce the overfitting of

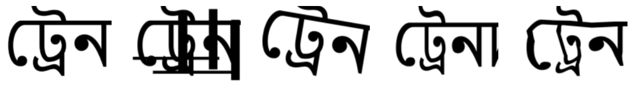


FIGURE 7. An example of the augmentation processes applied to the training dataset.

CNN architecture [59]. The random positions are generated for each training data on every iteration/epoch.

2) HANDWRITING AUGMENTATION

Apart from randomized cropping of the word-level training data, custom augmentation processes are also implemented. The augmentation includes various basic geometric techniques, including horizontal and vertical cutouts from the image, rotation, grid distortion, and affine transformation. Figure 7 illustrates an example of the augmented images.

3) WORD-LEVEL IDENTIFICATION

The architecture is trained over randomly cropped image frames. Finally, to classify word-level images, each variable-width-sized word is segmented into 110 × 110 shaped image frames while maintaining a 50% overlap between each image frame. Let, p_{ij} be the probability distribution of i^{th} word-image frame of being j^{th} class, n is the number of frames segmented from a word-image and m is the number of classes. The final classification for each word is calculated as,

$$wordClass = \underset{j}{argmax}\, f(j)$$
$$f(j) = p'_{0\leq j<m}$$
$$p'_j = \frac{\sum_{i=1}^{n} p_{ij}}{n} \tag{4}$$

Although the training is conducted in randomly cropped pieces from word-level images, the testing and validation are conducted on the actual word-level images.

TABLE 1. The statistics of the datasets used for evaluation.

Dataset	Classes	Words per-class		
		Mean	Median	Deviation
Bengali [60]	260	69.4	69.0	26.7
Devanagari [61]	12	60.0	64.5	22.8
Telugu [62]	11	40.0	43.0	22.7

V. EVALUATION

In this section, the performance of the proposed architecture is evaluated and compared with the state-of-the-art schemes. In the following sections, the datasets and training strategy are investigated and a brief on the experimental results is provided.

A. DATASETS

To validate the writer recognition architecture, BanglaWriting [60] dataset was used. However, owing to the scarcity of openly available writer recognition datasets for Indic languages, two small similar domain datasets of Devanagari [61],

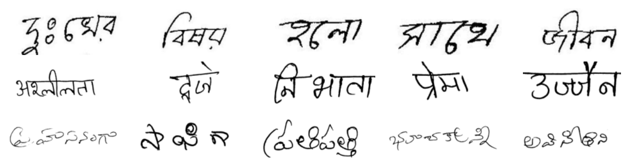


FIGURE 8. Different word-level images of the three datasets. The first, second, and third rows illustrate the word-level images of Bengali, Devanagari, and Telugu writings, respectively.

and Telugu [62] are added to the experimental benchmark. Figure 8 illustrates the word-level representations of the datasets. In the figure, it can be observed that Bengali and Devanagari contain horizontal lines above each character (also known as "matra"). In contrast, the Telugu script does not contain any horizontal line. Bengali and Devanagari is a mixture of vertical and circular writing patterns. In comparison to Bengali and Devanagari, Telugu scripts contain mostly circular patterns.

The Devanagari and Telugu datasets are pre-processed using similar methods to generate the BanglaWriting dataset script [63] to remove the variance shifts of the input data. Each dataset is split into train, test, and validation sets with a ratio of 40%-30%-30%, respectively. Table 1 explains the number of classes and per-class word-level data statistics. The word per-class is tuned to a minimum to keep the recognition position challenging, resulting in less training data than full-page writing.

The world-level data is fetched from datasets for training. The words were manually segmented, which were already present in the datasets. The writer detection models are trained using heavy augmentation procedures, including random cropping, rotation, cutout, and distortions (explained in Section IV-D2). Therefore minor errors in word-level segmentation do not affect the inference results of the models.

B. COMPETITIVE ARCHITECTURES

Apart from the TGCNN baseline architecture (illustrated in Figure 6) the thresholded Gabor is also implemented in four different DCNN baselines: ResNet [56], MobileNet [53], Xception [54], and DenseNet [64]. For each backbone (ResNet + TGCNN, MobileNet + TGCNN, Xception + TGCNN, and DenseNet + TGCNN), the architectures directly receive the inputs from the thresholded Gabor filter. Architectures specially designed for writer recognition (DeepWriter [65] and FragNet [5]) are trained using word-level images. Every architecture is pre-trained using a small subset of ImageNet [66] dataset.

C. TRAINING STRATEGY

All architectures are trained using a learning rate (lr) of 0.001 is implemented with *Adam* [67] optimizer. Furthermore, a default learning rate scheduler is used to train each of the deep learning models. The initial value of lr is decayed using the formula $lr = max(0.9 \times lr, 10^{-5})$. The decay is applied if the loss value on the test dataset does not reduce within five epochs. Training iteration is halted if the loss on testing data does not improve within the last 50 epochs. The benchmarks presented in the paper are a result of the mean of $k \times 3$, where k is the number of splits in k-fold cross-validation. For every dataset, the value of k is set to five. The overall training process is continued three times with augmentation ratios of 0, 0.1, and 0.2 for all architectures.

D. EVALUATION METRIC

Accuracy is a metric that is used to evaluate a classification model. In a biometric identification system, accuracy is one of the commonly used measurements to evaluate how a system accurately classifies any individual from a set of people. This metric is based on a "confusion-matrix" that contains four measures, True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). The significance of the four measures can vary depending on the variety of accuracy metrics. For evaluation, the fraction of the accurate predictions over all the predictions performed is measured and computed as,

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

Precision metric is used to compute the relevancy of the positive prediction, calculated as,

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

Recall metric is used to compute the positive predictions over the ground positive predictions computed as,

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

Finally, F1-score is used to represent the balance of precision and recall of the predicted result, which is computed as follows,

$$F1_{score} = 2 \times \frac{precision \times recall}{precision + recall} \tag{8}$$

Floating point operations (FLOPs) are used to measure the computational cost of the neural network architectures. FLOPs calculate the number of computations required in one second of calculation for a model. A lower FLOPs score describes a less-computationally complex model.

Furthermore, the receiver operating characteristic curve (ROC) is used to measure the classification performance of the models at different thresholds. Also, the area under the ROC curve (AUC) is used to present the aggregated purity of classification at different thresholds.

The architectures are evaluated based on the metrics tested for word-level classification. Word-level evaluation defines the accuracy metrics based on the overall prediction of word-level images, calculated using the methods explained in Section IV-D3.

TABLE 2. The comparison of different architectures in the Bengali dataset. The identification methods are sorted in ascending order based on the performance of the validation dataset. The best results based on the F1 score are highlighted in bold.

Model	Train				Test				Validation			
	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.
DeepWriter [5]	67.2	45.5	54.3	54.0	65.0	40.5	49.9	52.2	64.7	41.2	50.3	51.2
DenseNet121 + TGC [64]	88.1	66.3	75.7	79.3	86.2	65.9	74.7	78.0	83.9	63.9	72.5	73.8
ResNet + TGC [56]	90.7	73.6	81.3	83.1	88.5	71.0	78.8	78.8	88.8	69.0	77.7	77.9
MobileNet + TGC [53]	93.6	84.3	88.7	89.1	89.6	73.0	80.5	81.7	88.2	72.8	79.8	81.1
ResNet [56]	95.6	76.3	84.9	85.1	93.5	72.8	81.9	84.6	94.5	72.1	81.8	83.5
FragNet [65]	91.4	82.3	86.6	88.4	87.7	80.2	83.8	85.2	87.7	80.2	83.8	85.2
MobileNet [53]	97.9	88.3	92.9	94.9	96.2	87.6	91.7	92.6	95.4	85.5	90.2	90.7
DenseNet121 [64]	97.9	90.1	93.7	94.1	96.7	88.8	92.5	93.3	95.9	87.9	91.7	91.6
Xception [54]	96.8	91.6	94.1	94.8	96.9	90.4	93.5	93.7	96.1	90.1	93.0	93.6
Xception + TGC [54]	98.0	94.1	96.0	96.5	97.6	94.0	95.7	95.1	97.8	92.6	95.1	95.7
TGCNN	**99.1**	**98.2**	**98.7**	**98.8**	**98.4**	**96.6**	**97.5**	**97.5**	**98.7**	**95.4**	**97.0**	**97.4**

TABLE 3. The comparison of different architectures in the Devanagari dataset. The identification methods are sorted in ascending order based on the performance of the validation dataset. The best results based on the F1 score are highlighted in bold.

Model	Train				Test				Validation			
	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.
DeepWriter [5]	70.4	60.2	64.9	71.9	60.8	58.7	59.7	60.2	59.6	54.2	56.8	57.0
DenseNet121 + TGC [64]	93.7	61.9	74.6	77.4	89.2	54.2	67.4	72.0	87.1	53.6	66.4	72.0
FragNet [65]	88.7	65.2	75.2	79.3	87.4	61.9	72.5	72.9	87.1	59.5	70.7	71.4
ResNet + TGC [56]	89.1	70.9	80.0	84.7	87.3	65.5	74.8	81.0	86.8	67.9	76.2	79.8
DenseNet121 [64]	91.2	76.4	83.1	85.3	87.3	73.8	80.0	81.5	87.1	73.4	79.7	81.2
MobileNet [53]	92.8	77.2	84.3	86.6	89.4	75.6	81.9	82.7	88.1	75.8	81.5	82.6
MobileNet + TGC [53]	95.0	79.0	86.3	88.9	93.0	78.6	85.2	85.9	92.6	76.4	83.7	83.8
ResNet [56]	96.8	79.2	87.1	89.1	94.1	78.4	85.5	86.3	93.5	77.4	84.7	84.5
Xception [54]	97.8	80.4	88.3	91.6	96.9	78.7	86.9	89.4	96.7	78.2	86.5	88.2
Xception + TGC [54]	**98.1**	93.1	95.3	95.5	**97.5**	92.3	**94.8**	**94.2**	**97.4**	88.1	92.5	93.5
TGCNN	98.0	**95.3**	**96.6**	**96.8**	95.7	**93.5**	94.6	94.1	95.2	**92.1**	**93.6**	**94.1**

TABLE 4. The comparison of different architectures in the Telugu dataset. The identification methods are sorted in ascending order based on the performance of the validation dataset. The best results based on the F1 score are highlighted in bold.

Model	Train				Test				Validation			
	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.
DeepWriter [5]	22.4	21.9	22.1	23.6	19.3	16.3	17.8	17.1	19.1	15.7	17.2	17.0
DenseNet121 + TGC [64]	29.2	27.1	28.1	28.3	26.4	24.5	25.4	24.6	23.3	24.1	23.7	24.1
MobileNet + TGC [53]	31.2	27.7	29.3	31.8	27.5	25.0	26.2	25.0	23.3	24.6	23.9	24.3
ResNet + TGC [56]	30.8	29.5	30.1	29.9	29.5	29.5	29.5	29.5	25.3	25.3	25.3	25.6
ResNet [56]	80.4	64.0	71.3	73.2	75.0	57.7	65.2	65.9	75.0	55.2	64.3	65.5
FragNet [65]	82.6	75.1	78.7	80.7	76.4	69.1	72.6	74.3	75.9	68.3	71.9	72.8
MobileNet [53]	96.8	77.3	86.0	88.2	93.7	72.6	81.8	83.9	93.9	71.3	81.1	82.2
DenseNet121 [64]	98.7	81.7	89.4	86.3	96.1	75.1	84.3	85.1	92.0	74.3	82.2	84.9
Xception [54]	99.0	88.1	93.2	95.1	97.2	80.6	88.1	91.2	94.3	79.7	86.4	89.1
TGCNN	**99.2**	92.4	95.7	96.0	98.7	87.5	92.7	94.1	**94.9**	85.1	89.7	93.1
Xception + TGC [54]	**99.2**	**96.4**	**97.8**	**99.1**	**98.8**	**93.2**	**95.9**	**96.0**	94.3	**86.2**	**90.1**	**93.6**

E. EXPERIMENTAL SETUP

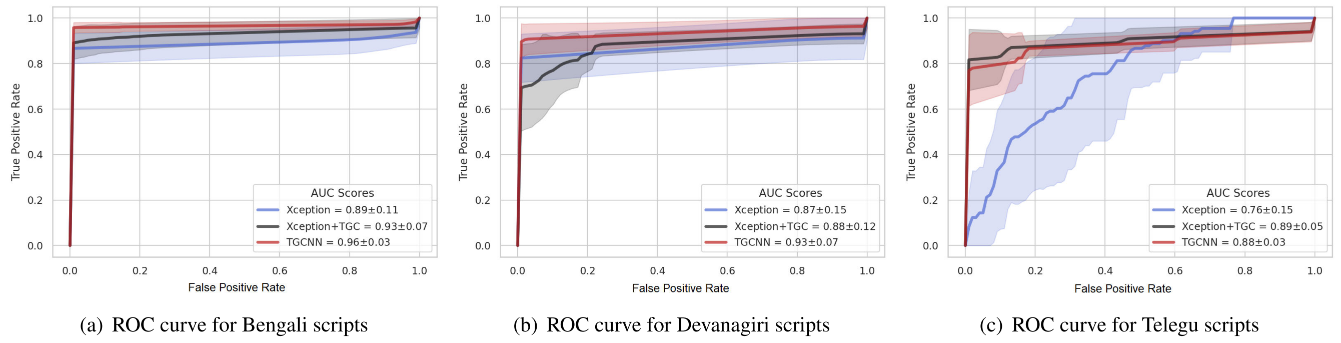
The overall data collection processings, experiments, and evaluations are performed using Python. Deep learning methods are implemented using Keras [68]. Numpy [69] and OpenCV [70] are used to perform mathematical operations and image processing on an individual basis.

F. EXPERIMENTS AND COMPARISONS

The comparison is split into three subsections. Firstly the computation speed for each architecture is observed. Then the performance of the architectures is discussed based on the four datasets. Finally, explanations related to the comparisons are investigated.

Table 5 explains the comparison of the computational complexity and number of parameters present in the architectures. The FragNet architecture consumes the highest computational complexity for input processing. In contrast, the proposed TGCNN architecture acquires the minimum number of parameters while being the second fastest architecture presented in the comparison. In the comparison, the TGC layer is also fused with different classification baselines: Xception, ResNet, DenseNet, and MobileNet architectures. It can be observed that adding a TGC layer with different baselines increases computational complexity, which is almost four times higher than the existing complexities of the baselines.

Table 2 illustrates a comparison of the architectures on the Bengali dataset. Existing writer recognition systems, FragNet, perform marginally in the Bengali dataset, and DeepWriter achieves poor performance. In contrast, among the TGC architectures, TGCNN achieves the best result



(a) ROC curve for Bengali scripts (b) ROC curve for Devanagiri scripts (c) ROC curve for Telegu scripts

FIGURE 9. The ROC curve of TGCNN, Xception + TGC, and, Xception baselines in Bengali, Devanagari, and Telugu scripts. The AUC scores are also reported in the figures.

TABLE 5. A comparison of the number of parameters and time-performance measure, FLOPs of DCNN architectures. Architectures are sorted from high to low time complexity.

Model	Parameters (in million)	FLOPs (in million)
FragNet [65]	78.91	6038.89
Xception + TGC [54]	21.86	4504.82
ResNet + TGC [56]	24.58	3756.61
DenseNet121 + TGC [64]	7.40	2538.49
Xception [54]	21.19	1121.14
ResNet [56]	23.92	1024.33
DenseNet121 [64]	7.13	704.71
DeepWriter [5]	5.43	591.46
MobileNet + TGC [53]	3.60	503.40
TGCNN	0.20	171.00
MobileNet [53]	3.32	114.38

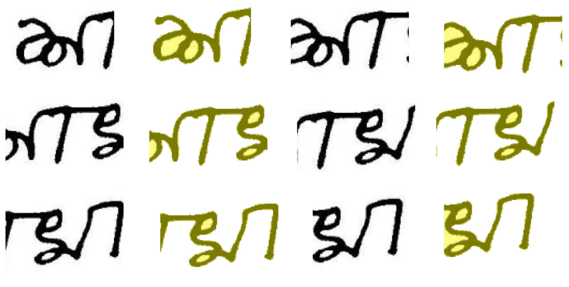
in the Bengali dataset. Furthermore, TCG mutated with Xception architecture (Xception + TGC) performs closely to TGCNN architecture. Compared to Xception + TGC, the Xception architecture closely falls behind compared to accuracy and F1 score, proving that TGC results in a slight performance improvement in the Xception architecture. Hence, TGC architectures are better compatible with Xception architecture. Comparatively, TGC mutated ResNet, DenseNet, and MobileNet architectures fall behind the base architectures' performance. The reason for the poor performance of any other baseline prevails in the architectural construction of Xception. Xception architecture implements separable convolution, which provides better non-linearity recognition capability than any other architecture. Hence, apart from Xception, any other baseline fails to learn the stroke patterns extracted through TGC. The benchmark clarifies a practical intuition for selecting separable convolution in the proposed TGCNN architecture.

Table 3 illustrates a comparison of the architectures on the Devanagari dataset. The performance of the existing writer recognition architectures is marginal. For this dataset, Xception + TGC leads to the Xception baseline. In comparison, the TGCNN architecture achieves superior accuracy. However, in the comparison of the Telugu dataset, enlisted in Table 4 Xception + TGC achieves superior performance. While the TGCNN architecture slightly lags behind the superior architecture. For the Telugu dataset, DeepWriter architecture results in poor performance. In contrast, FragNet architecture maintains a marginal performance for the three Indic script datasets.

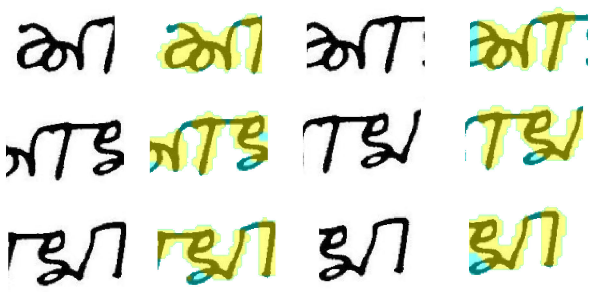
The TGCNN architecture follows almost the same strategy of performing pointwise convolution before depthwise convolution, which is introduced in Xception architecture. However, the Xception architecture contains a skip connection (or residual) with a pointwise convolution, which is avoided in the TGCNN architecture. With the similar implementation of Xception, the TGCNN architecture mainly performs similarly to Xception architecture in the Indic script-based writer identification task. The TGCNN is lightweight, with lesser parameters, and achieves better performance.

Finally, the three best-performing architectures, TGCNN, Xception + TGC, and Xception, are compared based on their classification purity. Sequentially, Figure 9 represented the ROC curve for Xception, Xception + TGC, and TGCNN classification models on Bengali, Devanagari, and Telugu language datasets. In addition, the graphs highlight the AUC scores achieved by the models. The graphs illustrate that both Xception and Xception + TGC architectures break down Telugu and Devanagari scripts, respectively. In contrast, the TGCNN model performs better in Bengali and Devanagari scripts. Nevertheless, a slight decrease in the AUC score is noticed for Telugu scripts. However, the TGCN model performs best when the probability threshold is increased in all cases.

Figure 10 shows a Grad-CAM [71] inference example of TGCNN and Xception architectures on Bengali dataset. Grad-CAM is a visual representation for a class-specific heatmap of a deep learning model. Hence, it is excellent for visualizing spatial attention of deep learning models. By inspecting the Grad-CAM heatmap, it can be perceived that the Xception architecture gives a similar priority to the handwriting strokes. In contrast, the TGCNN architecture gives a distinctive priority to some of the specific stroke patterns. The general perspective of Gabor convolution is to prioritize specific stroke patterns. Hence, the Grad-CAM visualization validates the approach of performing thresholded Gabor convolution.



(a) Inference of Xception architecture



(b) Inference of TGCNN architecture

FIGURE 10. The Grad-CAM [71] inference visualization conducted on the image patches of the Bengali dataset. The first column of every block illustrates the input, and the following column shows the priority heatmap for classification. Yellow color defines higher priority, while cyan indicates average priority.

VI. CONCLUSION

The paper introduces a deep learning based writer identification system specifically for Indic scripts, such as Bengali, Telugu, and Hindi. The paper provides examples of the basic construction of Indic scripts based on the Bengali language to explicate the structural differences in writing. Further, the paper derives thresholded Gabor filters, which filter the contours of the writing of specific stroke orientations when convolved in the writing image. The application is further extended into a shallow CNN architecture that improves the accuracy of the model, specifically for Indic writer recognition tasks. The architecture is tested over three Indic datasets, including Bengali, Devanagari, and Telugu. The proposed architecture performs better on Indic datasets than numerous DCNN architectures and writer-recognition methods while training with less than a single page of handwriting data. The overall research contribution would inspire researchers to explore the diversity of performance, the variance of handwritten patterns, and recognition methodologies related to both language-dependent and language-independent writer identification systems.

ACKNOWLEDGMENT

The authors would like to thank the Advanced Machine Learning (AML) lab for resource sharing and precious opinions.

REFERENCES

[1] Y. Zhu, T. Tan, and Y. Wang, "Biometric personal identification based on handwriting," in *Proc. 15th Int. Conf. Pattern Recognit. (ICPR)*, vol. 2, 2000, pp. 797–800.

[2] A. Rehman, S. Naz, and M. I. Razzak, "Writer identification using machine learning approaches: A comprehensive review," *Multimedia Tools Appl.*, vol. 78, no. 8, pp. 10889–10931, Apr. 2019.

[3] A. A. Ahmed and G. Sulong, "Arabic writer identification: A review of literature," *J. Theor. Appl. Inf. Technol.*, vol. 69, no. 3, pp. 1–11, 2014.

[4] C. Adak, B. B. Chaudhuri, and M. Blumenstein, "An empirical study on writer identification and verification from intra-variable individual handwriting," *IEEE Access*, vol. 7, pp. 24738–24758, 2019.

[5] L. Xing and Y. Qiao, "DeepWriter: A multi-stream deep CNN for text-independent writer identification," *CoRR*, vol. abs/1606.06472, pp. 584–589, Aug. 2016. [Online]. Available: http://arxiv.org/abs/1606.06472

[6] Z. Y. He and Y. Y. Tang, "Chinese handwriting-based writer identification by texture analysis," in *Proc. Int. Conf. Mach. Learn. Cybern.*, vol. 6, Aug. 2004, pp. 3488–3491.

[7] W. Y. Leng and S. M. Shamsuddin, "Writer identification for Chinese handwriting," *Int. J. Adv. Soft Comput. Appl.*, vol. 2, no. 2, pp. 142–173, 2010.

[8] S. A. Maadeed, W. Ayouby, A. Hassaïne, and J. M. Aljaam, "QUWI: An Arabic and English handwriting dataset for offline writer identification," in *Proc. Int. Conf. Frontiers Handwriting Recognit.*, Sep. 2012, pp. 746–751.

[9] B. Helli and M. E. Moghaddam, "A text-independent Persian writer identification based on feature relation graph (FRG)," *Pattern Recognit.*, vol. 43, no. 6, pp. 2199–2209, Jun. 2010.

[10] S. M. Reza, M. A. M. Bhuiyan, and N. Tasnim, "A convolution neural network with encoder-decoder applied to the study of Bengali letters classification," *Big Data Inf. Anal.*, vol. 6, pp. 41–55, Jan. 2021.

[11] T. R. Trautmann, *Languages and Nations: The Dravidian Proof in Colonial Madras*. New Delhi, India: Yoda Press, 2006.

[12] C. Janiesch, P. Zschech, and K. Heinrich, "Machine learning and deep learning," *Electron. Markets*, pp. 1–11, 2021.

[13] Z. He, B. Fang, J. Du, Y. Y. Tang, and X. You, "A novel method for offline handwriting-based writer identification," in *Proc. 8th Int. Conf. Document Anal. Recognit. (ICDAR)*, vol. 1, Aug. 2005, pp. 242–246.

[14] Y. Zhu, T. Tan, and Y. Wang, "Biometric personal identification based on handwriting," in *Proc. 15th Int. Conf. Pattern Recognit. (ICPR)*, vol. 2, 2000, pp. 797–800.

[15] B. Zhang, S. N. Srihari, and S. Lee, "Individuality of handwritten characters," in *Proc. 7th Int. Conf. Document Anal. Recognit.*, vol. 3, 2003, p. 1086.

[16] A. Schlapbach, M. Liwicki, and H. Bunke, "A writer identification system for on-line whiteboard data," *Pattern Recognit.*, vol. 41, no. 7, pp. 2381–2397, 2008.

[17] P. Thumwarin and T. Matsuura, "On-line writer recognition for Thai based on velocity of barycenter of pen-point movement," in *Proc. Int. Conf. Image Process. (ICIP)*, vol. 2, 2004, pp. 889–892.

[18] A. Bensefia, T. Paquet, and L. Heutte, "A writer identification and verification system," *Pattern Recognit. Lett.*, vol. 26, no. 13, pp. 2080–2092, 2005.

[19] A. Schlapbach and H. Bunke, "A writer identification and verification system using HMM based recognizers," *Pattern Anal. Appl.*, vol. 10, no. 1, pp. 33–43, Jan. 2007.

[20] J. Chu and S. Srihari, "Writer identification using a deep neural network," in *Proc. Indian Conf. Comput. Vis. Graph. Image Process. (ICVGIP)*. New York, NY, USA: Association for Computing Machinery, Dec. 2014, pp. 1–7, doi: 10.1145/2683483.2683514.

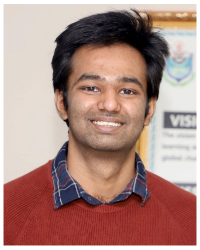
[21] S. Fiel and R. Sablatnig, "Writer identification and retrieval using a convolutional neural network," in *Computer Analysis of Images and Patterns*, G. Azzopardi and N. Petkov, Eds. Cham, Switzerland: Springer, 2015, pp. 26–37.

[22] V. Christlein, D. Bernecker, A. Maier, and E. Angelopoulou, "Offline writer identification using convolutional neural network activation features," in *Pattern Recognition*, J. Gall, P. Gehler, and B. Leibe, Eds. Cham, Switzerland: Springer, 2015, pp. 540–552.

[23] W. Yang, L. Jin, and M. Liu, "Character-level Chinese writer identification using path signature feature, dropstroke and deep CNN," 2015, *arXiv:1505.04922*. [Online]. Available: https://arxiv.org/abs/1505.04922
[24] S. Chen, Y. Wang, C.-T. Lin, W. Ding, and Z. Cao, "Semi-supervised feature learning for improving writer identification," *Inf. Sci.*, vol. 482, pp. 156–170, May 2019.
[25] V. Christlein, M. Gropp, S. Fiel, and A. Maier, "Unsupervised feature learning for writer identification and writer retrieval," 2017, *arXiv:1705.09369*. [Online]. Available: https://arxiv.org/abs/1705.09369
[26] S. Dargan and M. Kumar, "Writer identification system for indic and non-indic scripts: State-of-the-art survey," *Arch. Comput. Methods Eng.*, vol. 26, no. 4, pp. 1283–1311, Sep. 2019.
[27] B. Kumar, P. Kumar, and A. Sharma, "RWIL: Robust writer identification for indic language," in *Proc. 2nd Int. Conf. Intell. Comput. Control Syst. (ICICCS)*, Jun. 2018, pp. 695–700.
[28] C. Halder, S. M. Obaidullah, K. C. Santosh, and K. Roy, "Content independent writer identification on Bangla script: A document level approach," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 32, no. 9, Sep. 2018, Art. no. 1856011.
[29] C. Halder and K. Roy, "Individuality of isolated Bangla numerals," *J. Netw. Innov. Comput.*, vol. 1, pp. 33–42, Jan. 2013.
[30] C. Adak and B. B. Chaudhuri, "Writer identification from offline isolated Bangla characters and numerals," in *Proc. 13th Int. Conf. Document Anal. Recognit. (ICDAR)*, Aug. 2015, pp. 486–490.
[31] S. Biswas and A. K. Das, "Writer identification of Bangla handwritings by radon transform projection profile," in *Proc. 10th IAPR Int. Workshop Document Anal. Syst.*, 2012, pp. 215–219.
[32] S. Helgason and S. Helgason, *The Radon Transform* (Progress in Mathematics), vol. 2. Springer, 1980.
[33] M. Kumar, R. K. Sharma, and M. K. Jindal, "Offline handwritten Gurmukhi character recognition: Study of different feature-classifier combinations," in *Proc. Workshop Document Anal. Recognit. (DAR)*, 2012, pp. 94–99.
[34] K. Kalra and S. Rani, "Writer identification from offline isolated handwritten Gurumukhi characters," *Adv. Comput. Sci. Technol.*, vol. 10, no. 5, pp. 903–914, 2017.
[35] S. Dargan and M. Kumar, "Writer identification system based on offline handwritten text in Gurumukhi script," in *Proc. 6th Int. Conf. Parallel, Distrib. Grid Comput. (PDGC)*, Nov. 2020, pp. 544–549.
[36] T. Thendral, M. Vijaya, and S. Karpagavalli, "Prediction of writer using Tamil handwritten document image based on pooled features," *Int. J. Comput. Inf. Eng.*, vol. 9, no. 6, pp. 1572–1578, 2015.
[37] P. Purkait, R. Kumar, and B. Chanda, "Writer identification for handwritten Telugu documents using directional morphological features," in *Proc. 12th Int. Conf. Frontiers Handwriting Recognit.*, Nov. 2010, pp. 658–663.
[38] C. Halder, K. Thakur, S. Phadikar, and K. Roy, "Writer identification from handwritten Devanagari script," in *Information Systems Design and Intelligent Applications*. New Delhi, India: Springer, 2015, pp. 497–505.
[39] S. Dargan, M. Kumar, A. Garg, and K. Thakur, "Writer identification system for pre-segmented offline handwritten Devanagari characters using k-NN and SVM," *Soft Comput.*, vol. 24, pp. 1–12, Nov. 2019.
[40] S. Reddy, C. Andrew, U. Pal, A. Alaei, and V. Pulabaigari, "Writer identification in indic scripts: A stroke distribution based approach," in *Proc. 4th IAPR Asian Conf. Pattern Recognit. (ACPR)*, Nov. 2017, pp. 947–952.
[41] A. Alekseev and A. Bobe, "GaborNet: Gabor filters with learnable parameters in deep convolutional neural networks," 2019, *arXiv:1904.13204*. [Online]. Available: http://arxiv.org/abs/1904.13204
[42] R. Mehrotra, K. R. Namuduri, and N. Ranganathan, "Gabor filter-based edge detection," *Pattern Recognit.*, vol. 25, no. 12, pp. 1479–1494, Dec. 1992.
[43] T. P. Weldon, W. E. Higgins, and D. F. Dunn, "Efficient Gabor filter design for texture segmentation," *Pattern Recognit.*, vol. 29, no. 12, pp. 2005–2015, 1996.
[44] I. N. Bankman, T. S. Spisz, and S. Pavlopoulos, "Two-dimensional shape and texture quantification," in *Handbook of Medical Image Processing and Analysis*, I. N. Bankman, Ed., 2nd ed. Burlington, Vermont: Academic, 2009, ch. 15, pp. 261–277. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780123739049500246
[45] H. Li, Q. Zhang, and Z. Sun, "Iris recognition on mobile devices using near-infrared images," in *Human Recognition in Unconstrained Environments*, M. De Marsico, M. Nappi, and H. Proença, Eds. New York, NY, USA: Academic, 2017, ch. 5, pp. 103–117. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780081007051000051
[46] T. M. Abhishree, J. Latha, K. Manikantan, and S. Ramachandran, "Face recognition using Gabor filter based feature extraction with anisotropic diffusion as a pre-processing technique," *Proc. Comput. Sci.*, vol. 45, pp. 312–321, Jan. 2015.
[47] S.-Y. Chang and N. Morgan, "Robust CNN-based speech recognition with Gabor filter kernels," in *Proc. 15th Annu. Conf. Int. Speech Commun. Assoc.*, Sep. 2014, pp. 1–5.
[48] Y. Zhang, W. Li, L. Zhang, X. Ning, L. Sun, and Y. Lu, "Adaptive learning Gabor filter for finger-vein recognition," *IEEE Access*, vol. 7, pp. 159821–159830, 2019.
[49] Q. Xu, M. Zhang, Z. Gu, and G. Pan, "Overfitting remedy by sparsifying regularization on fully-connected layers of CNNs," *Neurocomputing*, vol. 328, pp. 69–74, Feb. 2019.
[50] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, pp. 1–48, Dec. 2019.
[51] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proc. IEEE*, vol. 109, no. 1, pp. 43–76, Jan. 2021.
[52] I. Rocco, R. Arandjelovic, and J. Sivic, "Convolutional neural network architecture for geometric matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6148–6157.
[53] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*. [Online]. Available: http://arxiv.org/abs/1704.04861
[54] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1251–1258.
[55] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556
[56] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[57] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 630–645.
[58] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
[59] R. Takahashi, T. Matsubara, and K. Uehara, "Data augmentation using random image cropping and patching for deep CNNs," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 9, pp. 2917–2931, 2019.
[60] M. F. Mridha, A. Q. Ohi, M. A. Ali, M. I. Emon, and M. M. Kabir, "BanglaWriting: A multi-purpose offline Bangla handwriting dataset," *Data Brief*, vol. 34, Feb. 2021, Art. no. 106633.
[61] K. Dutta, P. Krishnan, M. Mathew, and C. V. Jawahar, "Offline handwriting recognition on Devanagari using a new benchmark dataset," in *Proc. DAS*, 2018, pp. 25–30.
[62] K. Dutta, P. Krishnan, M. Mathew, and C. V. Jawahar, "Towards spotting and recognition of handwritten words in indic scripts," in *Proc. ICFHR*, 2018, pp. 32–37.
[63] A. Q. Ohi. (2020). *BanglaWriting: A Multi-Purpose Offline Bangla Handwriting Dataset (Script)*. [Online]. Available: https://github.com/QuwsarOhi/BanglaWriting
[64] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.
[65] S. He and L. Schomaker, "FragNet: Writer identification using deep fragment networks," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3013–3022, 2020.
[66] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[67] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980
[68] F. Chollet. (2015). *Keras*. [Online]. Available: https://keras.io
[69] T. Oliphant, *NumPy: A Guide to NumPy*. Spanish Fork, UT, USA: Trelgol Publishing, 2006. [Online]. Available: http://www.numpy.org/
[70] Itseez. (2015). *Open Source Computer Vision Library*. [Online]. Available: https://github.com/itseez/opencv

[71] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 618–626.

M. F. MRIDHA (Senior Member, IEEE) received the Ph.D. degree in AI/ML from Jahangirnagar University, in 2017. He joined as a Lecturer with the Department of Computer Science and Engineering, Stamford University Bangladesh, in 2007. He worked as a CSE Department Faculty Member at the University of Asia Pacific and a Graduate Coordinator, from 2012 to 2019. He was promoted as a Senior Lecturer at the Department of Computer Science and Engineering, in October 2010, and promoted as an Assistant Professor at the Department of Computer Science and Engineering, in October 2011. In May 2012, he joined UAP as an Assistant Professor. He is currently working as an Associate Professor with the Department of Computer Science and Engineering, Bangladesh University of Business and Technology. His research experience, within both academia and industry, results in over 80 journal and conference publications. For more than ten years, he has been a Supervisor with the master's and undergraduate students of their thesis work. His research interests include artificial intelligence (AI), machine learning, deep learning, natural language processing (NLP), and big data analysis. He has served as a program committee member for several international conferences/workshops. He served as an associate editor for several journals.



ABU QUWSAR OHI graduated in computer science. He is currently working as a Lecturer and a Research Assistant with the Department of Computer Science and Engineering, Bangladesh University of Business and Technology. His research interests include deep learning algorithms, pattern recognition, computer vision, and reinforcement learning. Moreover, he is a Proficient in prominent frameworks, including TensorFlow, Keras, NumPy, and Matplotlib. Apart from his research works, he is a Competitive Programmer and has attained more than ten Bangladeshi national programming contests, including the National Collegiate Programming Contest (NCPC) and the International Collegiate Programming Contest (ICPC). He is passionate about learning fundamental algorithms, including data structures, dynamic programming, and game theory. He has experience working in famous languages, including Python and C++.

JUNGPIL SHIN (Senior Member, IEEE) received the B.Sc. degree in computer science and statistics and the M.Sc. degree in computer science from Pusan National University, South Korea, in 1990 and 1994, respectively, and the Ph.D. degree in computer science and communication engineering from Kyushu University, Japan, in 1999, under the scholarship from the Japanese Government (MEXT). He was an Associate Professor, a Senior Associate Professor, and a Professor with the School of Computer Science and Engineering, The University of Aizu, Japan, in 1999, 2004, and 2019, respectively. He has coauthored more than 250 publications published in widely cited journals and conferences. His research interests include pattern recognition, image processing, computer vision, machine learning, human–computer interaction, non-touch interface, human gesture recognition, automatic control, Parkinson's disease diagnosis, ADHD diagnosis, user authentication, machine intelligence, handwriting analysis, recognition, and synthesis. He is a member of ACM, IEICE, IPSJ, KISS, and KIPS. He served several conferences as the program chair and a program committee member for numerous international conferences. He serves as an Editor for journals of IEEE and *Sensors* (MDPI). He serves as a reviewer for several IEEE and SCI major journals.

MUHAMMAD MOHSIN KABIR was born in Dhaka, Bangladesh. He received the B.Sc. degree in computer science and engineering from Bangladesh University of Business and Technology (BUBT). He is currently working as a Lecturer and a Research Assistant with the Department of CSE, BUBT. Also, he works as a Researcher with the Advanced Machine Learning Lab. He has experienced working in Python, Keras, TensorFlow, Sklearn, and Scipy. His particular research interests include deep learning, pattern recognition, computer vision, and natural language processing.

MUHAMMAD MOSTAFA MONOWAR (Member, IEEE) received the B.Sc. degree in computer science and information technology from the Islamic University of Technology (IUT), Bangladesh, in 2003, and the Ph.D. degree in computer engineering from Kyung Hee University, South Korea, in 2011. He worked as a Faculty Member with the Department of Computer Science and Engineering, University of Chittagong, Bangladesh. He is currently working as an Associate Professor with the Department of Information Technology, King Abdulaziz University, Saudi Arabia. His research interests include wireless networks, mostly *ad-hoc*, sensor, and mesh networks, including routing protocols, MAC mechanisms, IP and transport layer issues, cross-layer design, and QoS provisioning, security and privacy issues, and natural language processing. He has served as a program committee member for several international conferences/workshops. He served as an Editor for a couple of books published by CRC press and Taylor & Francis Group. He also served as a guest editor for several journals.

MD. ABDUL HAMID was born in Sonatola, Pabna, Bangladesh. He graduated from Rajshahi Cadet College, Bangladesh. He received the Bachelor of Engineering degree in computer and information engineering from the International Islamic University Malaysia (IIUM), in 2001, and the combined master's-Ph.D. degree from the Computer Engineering Department, Kyung Hee University, South Korea, in August 2009, majoring in information communication. His education life spans over different countries in the world. His high school and college (1989–1995). He has been in the teaching profession throughout his life, which also spans over different parts of the globe. From 2002 to 2004, he was a Lecturer with the Computer Science and Engineering Department, Asian University of Bangladesh, Dhaka, Bangladesh. From 2009 to 2012, he was an Assistant Professor with the Department of Information and Communications Engineering, Hankuk University of Foreign Studies (HUFS), South Korea. From 2012 to 2013, he was an Assistant Professor with the Department of Computer Science and Engineering, Green University of Bangladesh. From 2013 to 2016, he was an Assistant Professor with the Department of Computer Engineering, Taibah University, Madinah, Saudi Arabia. From 2016 to 2017, he was an Associate Professor with the Department of Computer Science, Faculty of Science and Information Technology, American International University-Bangladesh, Dhaka. From 2017 to 2019, he was an Associate Professor and a Professor with the Department of Computer Science and Engineering, University of Asia Pacific, Dhaka. Since 2019, he has been a Professor with the Department of Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia. His research interests include network/cyber-security, natural language processing, machine learning, wireless communications, and networking protocols.