Received May 31, 2020, accepted July 11, 2020, date of publication July 17, 2020, date of current version July 28, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3010166

Offline Hand Written Urdu Word Spotting Using Random Data Generation

FAIQ FAIZAN FAROOQUI[1], MUHAMMAD HASSAN[1], MUHAMMAD SHAHZAD YOUNIS[1], AND MUHAMMAD KASHIF SIDDHU[2]

[1]School of Electrical Engineering and Computer Sciences, National University of Sciences and Technology, Islamabad 44000, Pakistan
[2]School of Computer and Communications Engineering, Universiti Malaysia Perlis, Arau 02600, Malaysia

Corresponding author: Faiq Faizan Farooqui (faiq618@gmail.com)

ABSTRACT Urdu word spotting is among the most challenging tasks in image processing and word spotting of hand written Urdu text is even more so. When it comes to handwritten Urdu documents, variation among the same words of various writers is significant. The orientation and style of the handwriting makes it really challenging for a word spotting system to correctly recognize the instances of the keyword. In this research, we tend to overcome this hurdle. We propose a system that takes a database of hand written Urdu text and generates random, yet, similar images to improve the classifier's ability to recognize variations caused by difference in handwriting. For image generation, we used geometric transformations and variants of Generative Adversarial Network (GAN). For the word spotting process, Histogram of Oriented Gradients (HOG) features are extracted from ligature images and then used to train a Long Short-Term Memory (LSTM) network for the classification task. This is the first study that focuses on improving word spotting by generating arbitrary samples using GANs and its variants. The system achieved a promising recognition rate of 98.96% due to the sample generation using Cycle-GANs.

INDEX TERMS Word spotting, HOG features, hand written text, LSTM, GANs.

I. INTRODUCTION

Word spotting has acquired high significance in the field of Document Analysis and Recognition, as large amount of documents have been and are still being digitized. Researchers have proposed several word spotting techniques for Latin and Hanzi scripts but word spotting in Arabic script serves more of a challenge. The Arabic script is comprised of highly cursive characters. Moreover, shape of the characters also vary with respect to their position in the word. The language consists of many dots and diacritics that are essential for defining the pronunciation and grammar of the word. Also, the intra-word and inter-word spaces vary in the Arabic script. In case of hand written text, the most crucial problem is that the hand writing of each writer differs, which adds to the complications of this task. In Fig. 1, it can be seen that the handwriting can induce a major change in the geometry, shape, orientation and even the size of the words. Due to the varying shape, spaces, diacritics and writing style of the Arabic script, it is very difficult to segment hand written Arabic words into characters. Hence, most of the work on Arabic script is performed on connected components that are extracted from Arabic words called *Ligatures*.

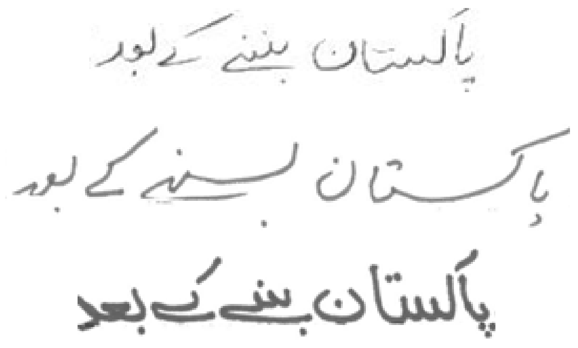

FIGURE 1. **A phrase of Urdu text hand written by three different writers.**

To overcome this hurdle, we have opted for random sample generation to introduce flexibility in the classifier. The system takes the ligature images and generates random ligatures of that image using various data generation methods. Then, Histogram of Oriented Gradients (HOG) features are extracted from the generated ligatures and the original ligatures. These HOG features are used to train the LSTM classifier.

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

For the purpose of random data generation, we have used geometric transforms, as well as, learning based techniques that consist of different variants of Generative Adversarial Network (GAN). GAN [1] is a type of generative model, which means that they can produce new content. GAN consist of two parts; the generator G and the discriminator D. The G generates new samples with an aim to replicate the training samples. The D then classifies the generated sample as real or fake. According to the discriminator's output, the G weights are updated to better replicate the true data and similarly, the D weights are updated to better classify the true data from the false. By competing against each other, the models are trained to create data very similar to the target data using any arbitrary input.

Traditional Neural Networks (NN) do not incorporate the effect of previous output on the next one. Contrarily, Recurrent Neural Networks (RNN) address this short coming by allowing information to persist while creating loop in a cell. Hence, every previous output contributes to the next output. In this study, we have exploited the use of LSTM for word spotting in hand written Urdu text using extracted HOG features as an input. The research shows that training the LSTM on HOG features yields better results as compared to training it on ligature images. The major contributions of this paper are as follows:

- This is the first study that exploits the use of random data generation for word spotting task. The proposed system uses random data generation (geometric and variants of GAN) to train the classifier as a handwriting style independent word spotting system.
- Because the system is trained on randomly varying data, it does not require lengthy transformations and corrections for normalizing the text in the pre-processing phase during run time.
- This is the first study that uses HOG features and LSTM classifier for word spotting of handwritten text, irrespective of any language.
- We empirically demonstrate the effectiveness of our proposed methodology on a trained LSTM network (as shown in Table 2), and compare the results with existing techniques of [2]–[4].

The remaining article is structured as follows. Section II outlines the related work in context of word spotting and data generation methods. Section III presents our methodology for (1) segregation of ligatures from document images (2) advantage of data generation techniques, and (3) extraction of HOG features for classification of ligatures using LSTM network. The section also contains details of each part of the system and their execution. Section IV discusses the results in context of qualitative analysis of GANs, classification results, and presents a performance comparison with existing systems. Finally, Section V concludes the paper with a discussion.

II. BACKGROUND AND RELATED WORK

While performing document analysis, they are categorized into two categories – modern documents and historical documents [5]. The modern documents are further divided into two sub-categories – printed text documents (machine generated) and handwritten text documents. In printed text documents, the characters and hence, the instances of a word are similar, where as, in handwritten text, the characters and words vary with the difference in handwriting styles. Thus, it becomes harder to recognize such texts. For the task of document retrieval, word spotting based methods are favored over traditional OCR, which requires the system to recognize each word [6].

A. WORD SPOTTING METHODS

The word spotting methods can be categorized into two major types: Non Learning-Based and Learning-Based. In Non Learning-Based approach, all the possible instances of query word existing in the training set are compared to the word images in the test set. Among the most popular approaches in this category are Dynamic Time Warping (DTW) [7], [8] and various matching techniques like soft matching done in [9], string matching in [10] and multi-level matching in [11].

In Learning-Based word spotting systems, various learning algorithms and their modified versions are used such as BLSTM [12], CNN [9], HMMs [13] and SVM [2]. These techniques require a learning phase, and generally perform better than the non learning based techniques [5]. However, they do require a large amount of training data for efficient performance.

1) WORD SPOTTING USING LEARNING BASED METHODS

Various attempts have been made to apply word spotting on hand written text of different languages, such as, Urdu, Arabic and Farsi language using various methods, such as, geometric transformations and morphological operations, etc. [5]. In all these methods, the pre-processing phase includes correction of slants, curves, pen pressure and strokes to normalize the text using transformations, as discussed by Abu ain *et al.* in [14]. This approach is effective, however, the pre-processing phase will take longer time if each ligature or character is to be normalized before locating the keyword.

There have been researches that attempt to train a system that can adapt to various handwriting styles, making it easier for a system with a pre-trained classifier to locate the keywords without normalizing each ligature in the document images. Graves and Schmidhuber [15] were the first to use multidimensional long short term memory (MDLSTM) with recurrent neural networks (RNN) for the recognition of handwritten text. Connectionist temporal classification (CTC) layer was used at the end of this network. Text line images are fed to the system. Then the system recognized the letters in the image. In [16], features were extracted from the handwritten word images using Resnet18 and were fed to a BLSTM network which recognized the words. References [17] and [18] used attention based RNN networks. In [17], the framework consisted of an encoder, attention mechanism and a decoder. The encoder had two parts: A VGG-19-BN [19] and a bidirectional Gated Recurrent Unit network (BGRU) [20]. The decoder consisted of unilayered GRUs. While in [18], the authors used Lenet-5 [21] for feature extraction. The encoders and decoders both were LSTM networks. In addition to that, the only attempt to incorporate reinforcement learning in an attention based network for hand writing recognition was made by Gui *et al.* [22].

On the other hand, the researches have extensively used Convolutional Neural Networks (CNN) in combination with embedded attributes. In these techniques the words are transcribed using an attribute representation rather than the labels. So these systems are trained on the attribute representations instead of the labels. The Pyramidal Histogram of Characters (PHOC) embedding [23] is the most popular among the attribute representations. In the testing phase, the attributes predicted by the fully connected layer are used to recognize the word. In [24], the authors proposed to use a spatial pyramid pooling layer [25] before the fully connected layer so that the CNN can handle arbitrary sized images. They termed their model as PHOCNet. Gurjar *et al.* [26], first trained the PHOCNet on synthetic data created using handwriting like fonts and data augmentation techniques e.g. affine transformations and elastic deformations. After that the system was trained on the original hand written data. Retsinas *et al.* [27] used PHOCNet to extract features by using the activations of its different layers. They also proved that the use of manifold learning method called t-sne [28] improved the results.

Biadsy *et al.* [29] and Dreuw *et al.* [30] used HMMs to classify various handwritten texts. Biadsy *et al.* in [29] used local orientation angle, stroke and loops of a character to train the classifier, where as, Dreuw *et al.* in [30] extracted Maximum Mutual Information and used discriminative training to train a writer independent classifier.

B. DATA GENERATION METHODS

When the transcribed data is available in limited amount, it may not be enough to successfully train a learning classifier. A solution to this problem is data augmentation. In data augmentation, new images are generated which are called ''synthesized images''. Data augmentation techniques can be used to enhance the amount of data in the dataset. Data augmentation in handwritten images can be done by many different ways e.g. 1) Using deep learning generative models e.g. GANs [1] 2) applying affine transformations like rotation and scaling on the original data and generating new images [31]. The first technique, i.e. generating images using deep learning generative models, is different from the other techniques. The generative models have the ability to synthesize images, which look very similar to handwritten images. This application of synthesizing images is new but is rapidly improving and getting popularity. These networks have the ability to generate new training data that may result in better performing classification models [32]. Auto encoders and Generative adversarial Networks (GANs) are examples of generative models. Auto encoders suffer from the drawback of generating blurry images while vanilla GANs (based on Multilayer perceptron) generate noisy and incomprehensible images [33]. On the other hand, the deep convolutional GANs (DCGANs) and their variants proposed later on, have continuously improved the quality of generated images. Alonso *et al.* [34] proposed a GAN with the aim of generating string images. Their system architecture consists of a generator (G) and a discriminator (D) network as in any GAN network. They introduced two new networks in the GAN: 1) a bidirectional LSTM network 2) a gated convolutional neural network having convolutional layers followed by LSTM layers. They generated Arabic as well as French handwritten image strings. They included these images in the original datasets and reported improved accuracy. However, recognizer's performance solely on the GAN generated images was not found competitive. Qian *et al.* [35] proposed a generative adversarial classifier to generate high-resolution character images from low-resolution images. They propose an addition of a classifier network in the traditional GAN architecture. The generator network takes a low-resolution image as input and generates a high-resolution image. The discriminator is trained to distinguish between real and high resolution images. The classifier network recognizes the generated image. Chang *et al.* [36] propose a version of GANs with an objective to generate character images of a font using character images from another font. The input character image and the output image do not need to be same. Their proposed network can also generate handwritten character images when given character images from a typed font as input. The network architecture consists of an encoder network, which generates a low dimensional representation of the input image character. This representation is fed to a transfer module that generates the feature representation in the output font style. This output representation is then fed into a decoder module, which generates a character in the target font style. Then the discriminator identifies that whether the generated image is from the target font style or not. The generated images are then fed to HCCRGoogleNet classifier [37] for recognition. GANs were used to generate handwritten text by Wang *et al.* [38]. The generator of this model consisted of LSTM layers with Variational Autoencoder and the discriminator network was made of convolutional layers. Bakker *et al.* in [39] used machine generated and handwritten text simultaneously to train the GAN for hand written text generation.

Rico *et al.* in [40] selected most prominent features by text rendering and applied distortion and curvature to transform these features in machine generated text to generate handwritten text. Kumar *et al.* [41] trained LSTM for writer independent classifier using bernoulli and gaussian distributions to determine strokes and pen lifts in the various handwriting style texts.

III. PROPOSED APPROACH

In this paper, we aim to devise a methodology that can avoid the lengthy transformation and correction procedures, and generates a diverse, yet, similar kind of handwritten language data through geometric and learning based

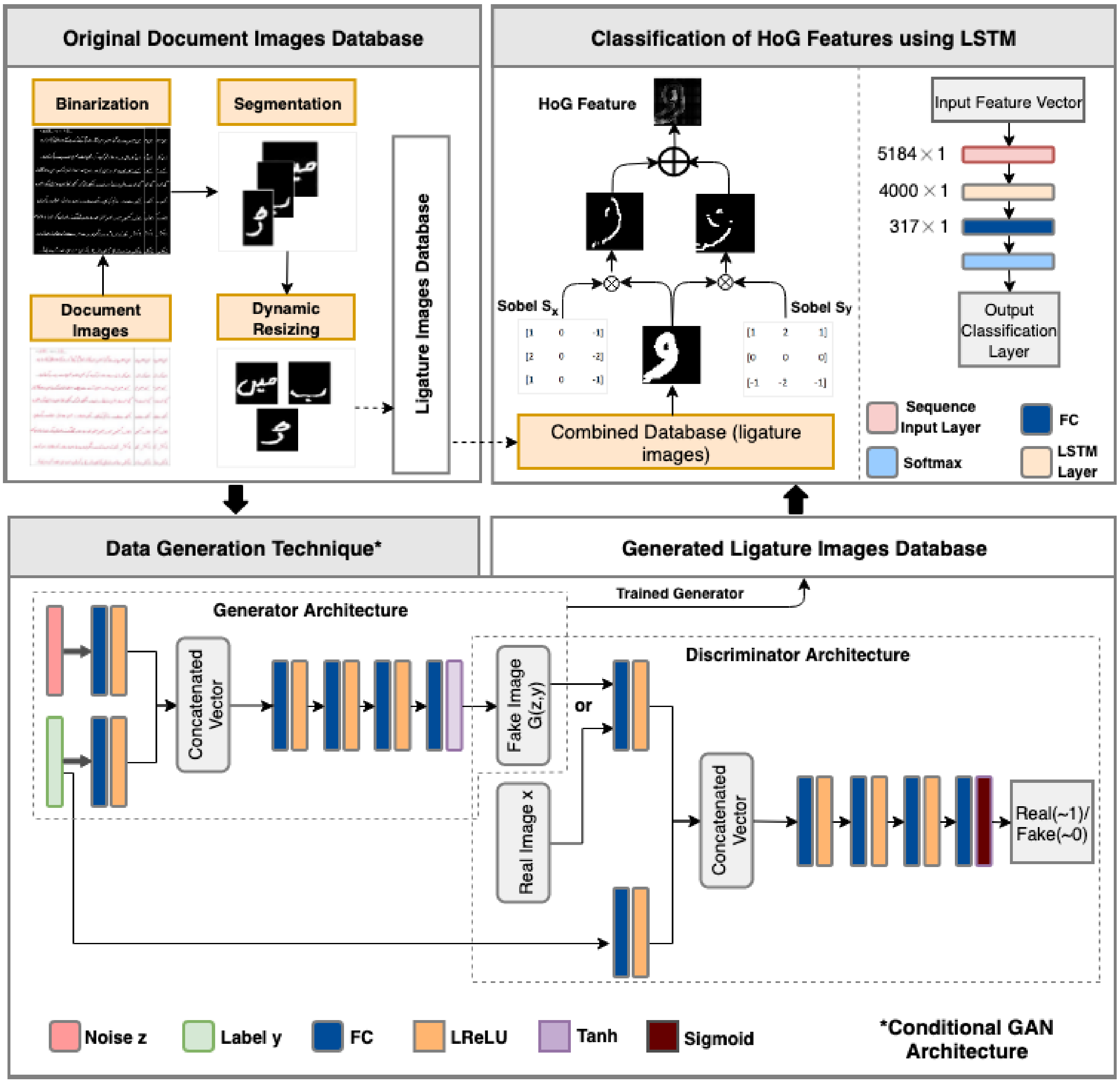

FIGURE 2. Detailed illustration of our proposed methodology.

augmentation techniques. The approach produces an efficient word spotting process with an improved accuracy results which outperforms existing word spotting techniques. In this section, we look forward to address the following three key questions:

1) How to segregate the ligatures and perform word spotting in real-time?
2) Advantage of data generation techniques over lengthy transformation procedures?
3) How HOG features are extracted for classification of ligatures in LSTM network?

The detailed workflow of our proposed methodology comprising of three main steps is illustrated in Fig. 2. We describe the methodology of each individual step in detail below.

A. HOW TO SEGREGATE THE LIGATURES AND PERFORM WORD SPOTTING IN REAL-TIME?

In Arabic word spotting, pre-processing phase is very important due to the cursive nature of the script. It is easier to perform word spotting on a printed text document [8], [13]. However, it is very difficult to process the hand written text documents due to slants, line orientations, writing style variations [2], [42], etc. In case of hand written data, it is essential to apply transforms to normalize the database and make it easier to process. In this research, Urdu Nastaleeq Handwritten Dataset (UNHD) has been used. The dataset was published by Ahmed *et al.* in [43]. The data set contains grayscale images of 6 pages of 8 lines each, hand written by various candidates, as shown in Fig. 1.

In order to augment the dataset and perform ligature classification, we segregate a document image of handwritten lines into individual ligatures by performing binarization, segmentation and resizing. As a result, we generate our ligature images database consisting of 317 different ligatures, i.e., 317 different classes to train a LSTM classifier, as shown in Fig. 2. The details of complete pre-processing phase is explained in the subsection III-E.

Once, the LSTM classifier is trained, the network performs the word spotting in real time using the Query-by-Example (QBE) method. In this method, query image is taken as an input by the system. The image of the query word is pre-processed (binarized, segmented and resized) to determine the composition and sequence of ligatures in that word.

Following the QBE method, all document images are then searched by binarizing, segmenting into ligatures, and dynamically resizing the ligatures. When the system detects the sequence of ligatures in a word in the document similar to the sequence of ligatures in the query word, it will label the word as an instance of the query word. The detailed method for word spotting of a query word is described in Algorithm 1.

Algorithm 1 Pseudo-Code for Query Word Segregation and Spotting Process in Real-Time

```
Require: Extract label sequence of the query word
 1: Load the query word image $I_q$
 2: Binarize the query word image $I_q$ as $I_{qb}$
 3: Segment the binarized word $I_{qb}$ into its respective ligatures $Ligs_q$
 4: Dynamically resize each ligature image and store in $Ligs_q$
 5: Extracting HOG features of each ligature image
 6: Classify the ligatures using the pre-trained LSTM network with cross entropy cost function as $Labels_q = \Sigma_{i=1}^{N} \Sigma_{j=1}^{N} t_{ij} \ln y_{ij}$
Require: Extract label sequence of each word of the test document images
 7: Binarize all test document images
 8: Segment the test document into lines
 9: Segment each line into ligatures $Ligs_{td}$
10: Dynamically resize each ligature image and store in $Ligs_{td}$
11: Extract HOG features of each ligature image $hfeatures_{td}$
12: Classify ligatures using pre-trained LSTM network to get their label and store the ligature labels in the sequence of occurrence $Labels_{td}$
Require: Compare the sequence of ligatures in query word and test document to get the instances of the query word
13: for iterations i = number of ligatures in test document do
14:   if $Labels_q(1) == Labels_{td}(i)$ then
15:     if $Labels_q(2) == Labels_{td}(i+1)$ then
16:       # up till n i.e. the last ligature in the query word
17:       if $Labels_q(n) == Labels_{td}(i+n-1)$ then
18:         mark the ligatures i to i + n − 1 in the test document as instance of the query word
19:       else
20:         # skip the iteration if any of the ligature labels in the query word do not match the ligatures in the test document in their respective sequence
            return
21:       end if
22:     else
23:       return
24:     end if
25:   else
26:     return
27:   end if
28: end for
```

B. ADVANTAGE OF DATA GENERATION TECHNIQUES

In most studies conducted in this area, the system normalizes each ligature or character image before extracting the features and classifying it which takes up more computation time as well as effort [5]. At the same time, it is still necessary to normalize the images for the classifier to correctly recognize them. In our study, we propose a system that will generate images similar to the parent database but with random transformations which we use to train the classifier. Using these images for training will enable the classifier to recognize similar ligatures in various handwriting styles without normalizing the ligature images. This will speed up the classification process, as system will not have a need to apply normalization to each ligature before classifying it.

In our work, we have exploited the use of non-learning and learning based methods for the task of random data generation. In non-learning based method, we have used two geometric transformation techniques – Rotation and Scaling, where as, in learning based method, we have used standard GAN and its convolutional variant, i.e., Deep Convolutional GAN (DCGAN); two GAN models with a control conditioning and structural improvements, i.e, Conditional GAN (CGAN), Auxiliary Classifier GAN (ACGAN), the latest GAN structures with an improved objective function and better performance results, i.e., Wasserstein GAN (WGAN), and Wasserstein GAN with Gradient Penalty (WGAN-GP), and cycle-consistent adversarial network (CycleGAN) which make use of adversarial loss and cycle constraint to achieve the image translation from X domain to Y domain, and vice versa. The data generated using the trained GAN models are stored in generated ligature image database, as shown in Fig. 2.

C. HOW HOG FEATURES ARE EXTRACTED FOR CLASSIFICATION OF LIGATURES IN LSTM NETWORK?

HOG features are very helpful, as they provide information regarding the image geometry even in the form of a vector. It is a feature similar to Scale Invariant Feature Transform (SIFT) in which gradients are calculated in pre-defined patches or cells in an image and are combined later on. This task is performed using complex masks, such as, Prewitt or Sobel operators [44]. The occurrences of similar gradient orientations in each patch of an image are counted. Based on the number of gradient orientations in a cell, histogram is produced. The HOG features are acquired by combining the histograms of all the patches of an image in a vector form. This is the reason that HOG features are chosen for this study. A 1-D LSTM network take images in a vector form, and not directly as input, the geometrical information and shape of an image is lost. Hence, HOG features provide the classifier with image gradients which cater for the geometry and shape of the foreground in an image.

Once the HOG features are acquired, they are given to LSTM network for the classification of ligatures. LSTMs [45] are memory-based learning systems that belong to the category of RNNs. LSTMs are designed to learn sequential information, such as, one dimensional signals and sequences. LSTMs were introduced as a solution to vanishing gradient problem [46] with its main idea being the insertion of gates to an RNN cell. The gates are external control inputs passed through sigmoid activation function to control the functions of a cell. A generic scheme of LSTM is shown in Fig. 3.

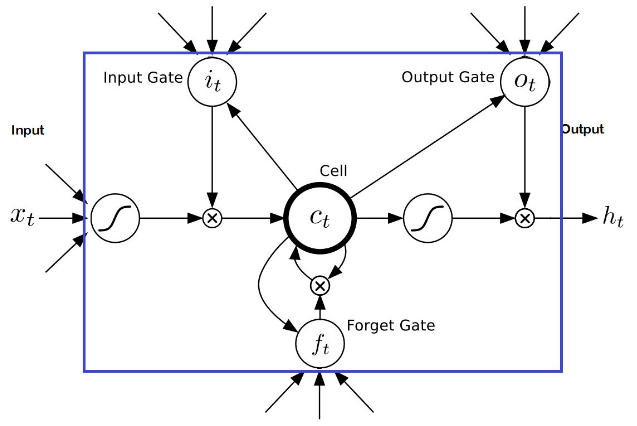

FIGURE 3. A block diagram of the LSTM cell with the gates used to control the weights [47].

In Fig. 3, i, f and o represent the input, forget and output gates respectively, while C represents the hidden state of the cell. The input gate i_t determines the information used to update the hidden state C. The forget gate f determines the information from previous cell state C_{t-1} to be used, while the output gate o_t decides what information to output to get the cell state C_t. Hence, the corresponding gate equations are represented as:

$$f_t = \sigma(W_f[h_{t-1,x_t}] + b_f) \tag{1}$$

$$i_t = \sigma(W_i[h_{t-1,x_t}] + b_i) \tag{2}$$

$$o_t = \sigma(W_o[h_{t-1,x_t}] + b_o) \tag{3}$$

$$h_t = o_t * \tanh(C_t) \tag{4}$$

D. DATA COLLECTION

To demonstrate the effectiveness of proposed framework, we evaluate our methodology on UNHD database, as described in subsection III-A. In this section, we shall explain the experimental procedure to extract the ligature images from handwritten document images (UNHD database). This involves the techniques of binarization, segmentation, and resizing. Following these steps, we shall generate a ligature image dataset comprising of 317 ligature classes (52086 samples). Later on, we shall report the hyperparameters and architectural modifications in a GAN and its variants. Lastly, we shall explain the procedure of HOG features acquisition and report the architecture details of LSTM network.

E. PRE-PROCESSING

Fig. 4 shows the pre-processing phase comprising of three steps (binarize, segment, resize) to extract the ligature images from document images (UNHD database). Each step is explained independently in the next sub-sections.

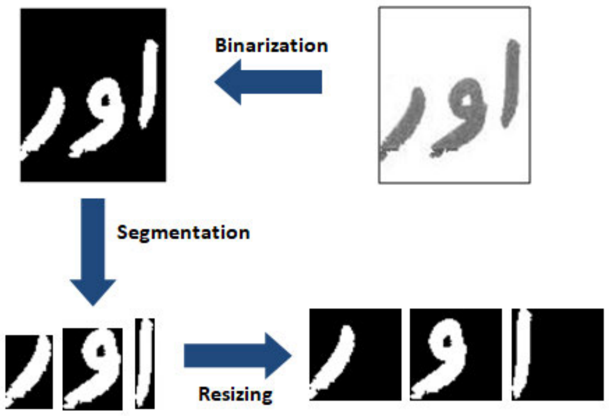

FIGURE 4. Pre-processing phase.The words are binarized, segmented into ligatures and dynamically resized before extracting the HOG features.

1) BINARIZATION

The document images in the database are binarized using Otsu's binarization technique [48]. The algorithm tends to maximize the inter-class variance between foreground (white) and the background (black). First, the probabilities $\omega_0(t)$ (background) and $\omega_1(t)$ (foreground) of each intensity level is computed using a random threshold t, and a histogram is created, as given in following equations:

$$\omega_0(t) = \Sigma_{i=0}^{t-1} p(i) \tag{5}$$

$$\omega_1(t) = \Sigma_{i=t}^{L-1} p(i) \tag{6}$$

where t is the threshold, and L is the number of pixels in a document image. We then calculate the means. Taking μ as the total mean and μ_0 and μ_1 as class means, are given by:

$$\mu_0(t) = \frac{\Sigma_{i=0}^{t-1} ip(i)}{\omega_0} \tag{7}$$

$$\mu_1(t) = \frac{\Sigma_{i=t}^{L-1} ip(i)}{\omega_1} \tag{8}$$

$$\mu_T = \Sigma_{i=0}^{L-1} ip(i) \tag{9}$$

the goal here is to maximize the inter-class variance, as:

$$\sigma_0^2(t) = \sigma^2 - \sigma_1^2(t) = \omega_0\omega_1[\mu_0(t) - \mu_1(t)]^2 \tag{10}$$

The threshold t is increased in each iteration. For each t, we calculate the variance σ_0^2, and we select that value of t for which σ_0^2 is the maximum.

2) SEGMENTATION

The characters in Urdu text change their shapes with their position in the word, which makes it difficult for a machine to recognize them. Hence in our work, we have segmented the images of handwritten documents into ligature images. Following are the steps involved in the segmentation process:

- Detect and segment lines using vertical run length smoothing algorithm. For each line, we performed the following steps
- Connected components are detected using a 2×2 cell in 8 connectivity. The bounding boxes and centroids of all components are extracted.
- Area of each component is calculated from the bounding box.
- The components are split into major and minor components. If a component is less than 1/4 of the average component area, then, it is a minor component.
- Get the baseline of the entire line by performing linear regression of the centroids of all major components.
- Check the position of the minor components. If the minor component is above the baseline, assign it to the major component below it (vertically overlapping) to make the main components. In case, if there are more than one major components vertically overlapping the minor component, assign it to the closest major component. Similarly, if the minor component is below baseline, assign it to the closest major component above it.
- Get the contour of the main components (major and its assigned minors).
- Get the cut points for the segmentation of each main component.
- Segment the main components using the cut points to get the ligature images.

3) DYNAMIC RESIZING

Once, the complete document images are binarized and segmented into ligatures, it is resized to 55 × 55 pixels. In this research, we perform dynamic resizing [49] of images, as shown in Fig 5. The conventional resizing method stretches image to the required size, hence, disturbing the geometry and aspect ratio of an image. In our approach, the longer dimension, irrespective of the image width or height, is stretched to 55 pixels while still maintaining the aspect ratio. The shorter dimension is increased to 55 pixels by padding zeros. The binarization and segmentation of the database is done before the data generation. After the data is generated, the original ligatures, as well as, the generated ligatures are dynamically resized.

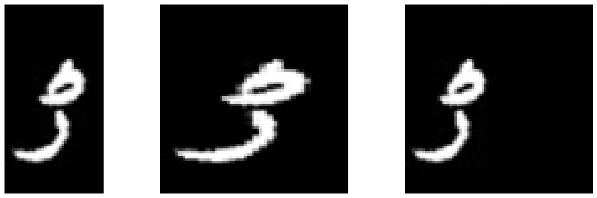

FIGURE 5. Original ligature image (left), Conventionally resized image (middle), Dynamically resized image (right).

F. DATA GENERATION

After image segmentation and binarization, original database is used to generate similar, yet, random images to enhance the classifier's ability to recognize ligatures despite the variation in writing style, size and orientation. The parent database is divided into training and testing data. The training data consists of 41,669 images and test data comprises of 10,417 images non-uniformly distributed in each class due to difference of number of occurrences of ligatures in the database. In each method, 100 samples are generated per class using the training data which made a total of 31,700 generated images that were added to the original database for training the classifier. In this sub-section, we shall explain the basic setup of geometric techniques, as well as, introduce the network architectures of learning-based data generation techniques. Later on, in Section IV, we shall analyze the experimental results with respect to the qualitative analysis, and also compare the classification results acquired from LSTM network.

1) GEOMETRIC TECHNIQUES

In this category, we have used rotation and scaling to generate images.

a: ROTATION

The images from original data set are rotated 5 times through random rotations within the angle range of $(-25, 25)$ degrees. In doing so, each image generates 5 randomly rotated images to create a modified dataset, as shown in Fig. 6.

b: SCALING

To create this variant, each image in the original data set is randomly scaled down 5 times between the range of

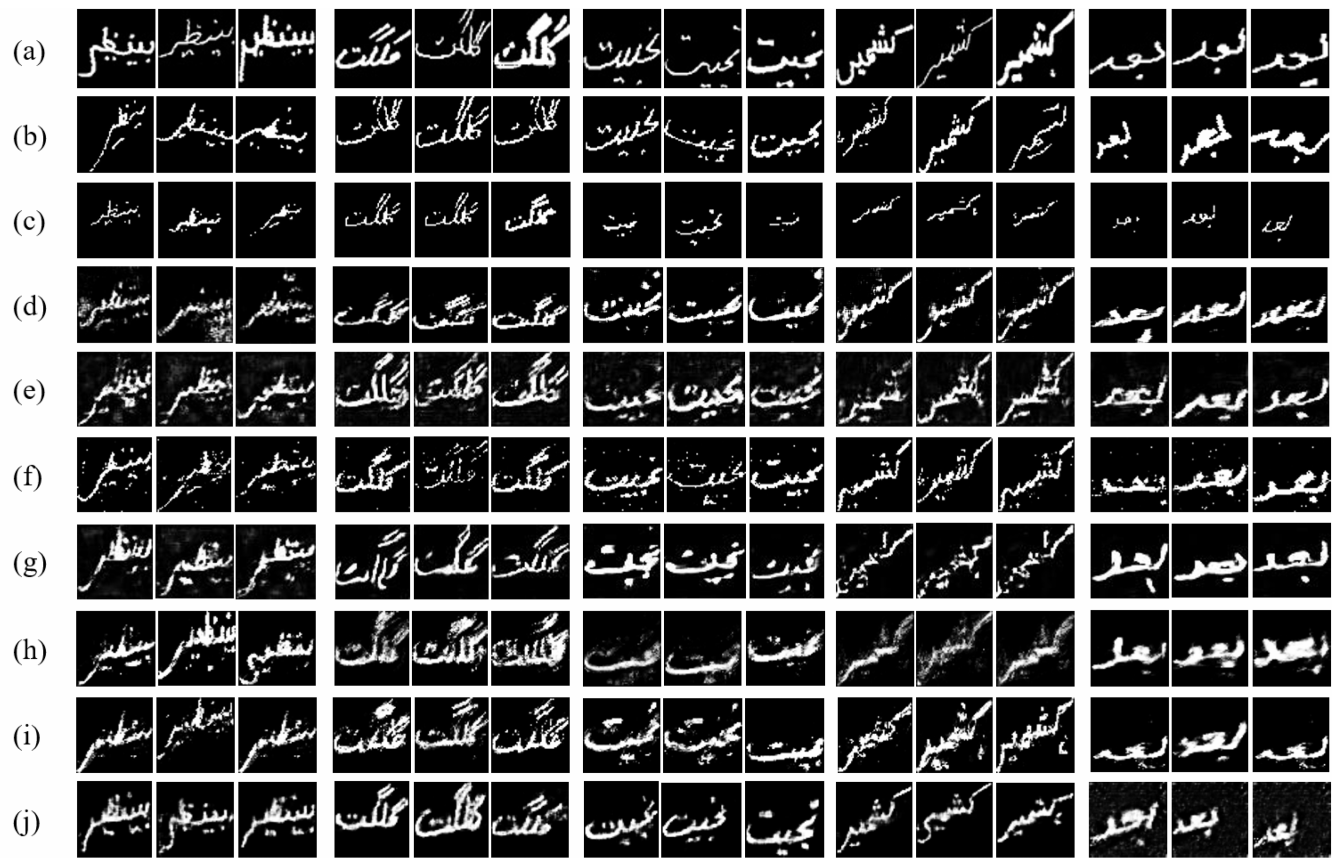

FIGURE 6. Comparisons of images generated through data generation techniques. The samples in the first row of subfigure (a) are original ligature images from UNHD dataset. The remaining rows (subfigure (b)-subfigure (f)) contain samples generated from data generation techniques (b) rotation (c) scaling (d) standard GAN (e) DCGAN (f) CGAN (g) ACGAN (h) WGAN (i) WGAN-GP (j) CycleGAN

0.25 to 0.75, as shown in Fig. 6. After downsizing, each image is padded with zeros around it's border so as to maintain the original image height and width.

2) LEARNING-BASED TECHNIQUES

GAN is known for its delicate and unstable training process, and it may produce blurry image generation for a diversified dataset. In the past, researchers have opted out different customization in the architecture of GAN, thus, providing an improved learning stability and avoiding the mode collapse and balancing problems [50].

To better control the model stability, we have used different variants of GAN (DCGAN, CGAN, ACGAN, WGAN, WGAN-GP, CycleGAN) to diversify our dataset. Each GAN model, except WGAN, is trained using ADAM optimizer [51] with mini-batches of size 64 and initial learning rate of $lr = 0.0002$, $\beta_1 = 0.5$, $\beta_2 = 0.999$ is used. Also, momentum [52] with the value of 0.6 and dropout [53] with probability of 0.25 is applied to the D network of standard GAN, DCGAN, ACGAN, and CGAN. For standard GAN, DCGAN, ACGAN and CycleGAN, ReLU is used as an activation function in G network, whereas, Leaky ReLU (LReLU) [54] with a leaky slope $\alpha = 0.2$ is applied in D network. Contrarily, LReLU with $\alpha = 0.2$ is used in G and D architectures of CGAN, WGAN, and WGAN-GP. Each GAN model, except CycleGAN, is trained for 1000 number of epochs.

For each GAN model, we customized the architectures of G and D networks, which are shown in Appendix A, therefore, allowing the models to reproduce synthesized images of dimension ($56 \times 56 \times 1$) at the G output. In the G network of each GAN (except CycleGAN), a latent noise vector z with dimension 100 is drawn from a uniform distribution. The samples synthesized using standard GAN, DCGAN, WGAN, and WGAN-GP may belong to any class present in the training dataset, as these models do not employ any conditioning content. Hence, we train these models on individual ligature classes. From each class, we generated 100 images. Thus, for each GAN (standard GAN, DCGAN, WGAN, WGAN-GP), a total of 31,700 images were added to the original database.

Contrarily, CGAN and ACGAN provide an approach to produce images with user-specified content, thus, we train each model in the form of subsets. We separated our database comprising of 317 classes into subsets of 10 classes. As a result, we have 31 subsets that consist of 10 classes in each subset, whereas, the last subset has 7 classes. Each GAN

TABLE 1. The loss functions of generator and discriminator network.

Model	Discriminator Loss Function	Generator Loss Function
GAN [1]	$\underset{D}{max} L_D^{GAN} = E_{x\sim p_{data}}[log(D(x))] + E_{z\sim p_z}[log(1-D(G(z)))]$	$\underset{G}{min} L_G^{GAN} = E_{z\sim p_z}[log(1-D(G(z)))]$
DCGAN [33]	$\underset{D}{max} L_D^{DCGAN} = E_{x\sim p_{data}}[log(D(x))] + E_{z\sim p_z}[log(1-D(G(z)))]$	$\underset{G}{min} L_G^{DCGAN} = E_{z\sim p_z}[log(1-D(G(z)))]$
CGAN [56]	$\underset{D}{max} L_D^{CGAN} = E_{(x,y)\sim p_{data}}[log(D(x,y))] + E_{y\sim p_{data}, z\sim p_z}[log(1-D(y, G(z,y)))]$	$\underset{G}{min} L_G^{CGAN} = E_{y\sim p_{data}, z\sim p_z}[log(1- D(y, G(z,y)))]$
ACGAN [57]	$\underset{D}{max}\underset{C}{min} L_D^{ACGAN} = E_{x\sim p_{data}}[log(D(x))] + E_{y\sim p_{data}, z\sim p_z} [log(1-D(G(z,y)))] + E_{x,y\sim p_{data}}[log(1-C(x,y))] + E_{y\sim p_{data}, z\sim p_z}[log(1-C(G(z,y),y))]$	$\underset{C,G}{min} L_G^{ACGAN} = E_{y\sim p_{data}, z\sim p_z}[log(1- D(G(z,y)))] + E_{y\sim p_{data}, z\sim p_z}[log(1-C(G(z,y),y))]$
WGAN [59]	$\underset{D}{max} L_D^{WGAN} = E_{x\sim p_{data}}[D(x)] - E_{z\sim p_z}[D(G(z))]$	$\underset{G}{min} L_G^{WGAN} = -E_{z\sim p_z}[D(G(z))]$
WGAN-GP [61]	$\underset{D}{max} L_D^{WGAN-GP} = L_D^{WGAN} - \lambda E_{x\sim p_{data}}[(\parallel \nabla D(\alpha x + (1-\alpha G(z))) \parallel -1)^2]$	$\underset{G}{min} L_G^{WGAN-GP} = -E_{z\sim p_z}[D(G(z))]$
CycleGAN [62]	$\underset{D}{max} L_D^{CycleGAN} = \underset{D}{max} L_D^{GAN}(G, D_Y, X, Y) + \underset{D}{max} L_D^{GAN}(F, D_X, Y, X) + \lambda E_{x\sim p_{data}}[\parallel F(G(x)) - x \parallel_1] + \lambda E_{y\sim p_{data}}[\parallel G(F(y)) - y \parallel_1]$	$\underset{G}{min} L_G^{CycleGAN} = \underset{G}{min} L_G^{GAN}(G, D_Y, X, Y) + \underset{G}{min} L_G^{GAN}(F, D_X, Y, X) + \lambda E_{x\sim p_{data}}[\parallel F(G(x)) - x \parallel_1] + \lambda E_{y\sim p_{data}}[\parallel G(F(y)) - y \parallel_1]$

variant (CGAN or ACGAN) is trained on individual subset, and from each class, we generated 100 samples. As a result, 31,700 generated images were added to the original database.

a: STANDARD GAN

Goodfellow *et al.* in [1] introduced the concept of GAN, which utilizes two feed-forward neural networks that are pitted against each other in an adversarial manner. It consists of two neural networks, that are, the generator G and the discriminator D. The G learns a mapping from random noise vector z to produce the realistic data samples $G(z)$, approximating the training set. The goal is to generate samples such that the D can not distinguish between generated samples and the real samples from training set. Formally, it can be expressed as $G : G(z) \rightarrow R^{|x|}$, where $z\epsilon R^{|z|}$ is a sample from latent space, $x\epsilon R^{|x|}$ is an image $G(z)$ generated from latent space z, and $|\cdot|$ corresponds to the number of dimensions.

In a basic GAN, the task of D is to estimate, whether an image comes from training dataset (real $\simeq$1) or it is a sample generated by G (fake $\simeq$0), i.e., $D : D(x) \rightarrow (0, 1)$. The goal is to correctly differentiate between samples and not to be get fooled by G. Hence, the approach leads the two networks partake in a min-max zero-sum two player game, which allows the G network to learn the data distribution of training data set. A formal definition of GAN is formulated in Table 1, while the network architecture details are shown in Appendix A (Table 4).

b: DCGAN

A Deep Convolutional or DCGAN [33] works same as the standard GAN, taking noise z as an input to generate similar kind of samples existing in our database. However, in this GAN, the G and the D models consist of convolutional layers instead of fully connected layers which makes it more suitable to learn intrinsic properties of an image. The formal expression of DCGAN is the same as expressed for standard GAN.

The G in a DCGAN consists of a latent noise vector z which is mapped to a dense layer to produce an output of $7 \times 7\times$ 128 (reshaped). The dense layer is passed onto subsequent convolutional layers (Upsampling, Convolution, Zero Padding) to produce an output image $G(z)$ of $56 \times 56\times$ 1. The convolutional layers make use of fractional strided operations which allow upsample operators to be learned during training. Furthermore, to stabilize the training process, all convolutional layers, except the last layer, is passed through batch normalization technique [55] followed by ReLU activation function, whereas, the last convolutional layer is passed through Tanh activation function.

Similarly, D in a DCGAN takes an input image (real image x or fake image $G(z)$) of dimension $56 \times 56 \times 1$ which is passed through a series of strided convolutional layers. The last convolutional layer of dimensions $8 \times 8 \times 256$ is flattened out and passed through sigmoid activation function, therefore, providing the degree of real ($\simeq$1) or fake ($\simeq$0) score of an image. Each convolutional layer (except dense layer) passes through batch normalization technique [55], which is followed by LReLU activation function. Appendix A (Table 5) shows the network architecture of DCGAN, while the loss function of DCGAN is formulated in Table 1.

c: CONDITIONAL GAN

The Conditional GAN or CGAN [56] is an extension of GAN in which the G not only requires random noise z as an input, but also incorporates the label y of a particular sample x from the training data. The G then attempts to generate a fake image $G(z, y)$ with respect to the provided label y. Formally, it can be expressed as: $G : G(z, y) \rightarrow R^{|s|}$, where, $z \epsilon R^{|z|}$ is sample from latent space z, $y \epsilon R^{|x|}$ is the corresponding label of a particular training sample x, $s \epsilon R^{|s|}$ is a generated sample image $G(z, y)$, and $| \cdot |$ corresponds to the number of dimensions.

Similarly, the D network is given fake image $G(z, y)$, label y and real image x, label y as an input. Here, the goal is to fool the G by maximizing the probability of real image, label pair ($\simeq$ 1) at the output. Hence, discriminator classifies between real image, label and fake image, label pair.

In this work, we have followed conventional CGAN architecture which comprises of dense layers in G and D network. However, in the G architecture, we have mapped the latent noise vector z and conditioning label y onto individual dense layers, which comprises of 256 units each respectively. The individual dense operations from z and y are concatenated together (output shape of 512 units) before they are finally fed into the subsequent dense layers. Each dense layer (except last layer) is passed through batch normalization technique, which is followed by LReLU operation.

The similar procedure is adopted in the D architecture, where, the real image x or generated image $G(z, y)$ is mapped to 4096 units, while, the label y is mapped to 256 units respectively. The output from both dense operations is concatenated together to produce a vector of $4352 \times 1 \times 1$. The concatenated vector is passed through series of dense layers to provide the estimate that whether an image, label pair comes from the training dataset or it is a sample generated by G. In each dense layer (except last layer), LReLU and dropout is applied. Appendix A (Table 6) shows the network architecture of CGAN, while, the loss function of CGAN is shown in Table 1.

d: AUXILIARY CLASSIFIER GAN

The Auxiliary Conditional or ACGAN [57] is a further extension of the Conditional GAN. The G of ACGAN functions same as the CGAN. The difference is in the D network. The D in the ACGAN is not provided with a class label c, as in the case of CGAN. The D of the ACGAN takes only the generated image as input and tells if it is a fake or not, while, also predicting a label for the generated image. The G in ACGAN consists of a latent noise vector z and conditional class label y which is concatented together and passed into a dense layer to produce an output of $7 \times 7 \times 128$ (reshaped). The dense layer is projected onto subsequent fractionally-strided convolutional layers (Upsampling, Convolution, Zero Padding) to produce a fake output image $G(z, y)$ of dimensions $56 \times 56 \times 1$. Each convolutional layer passes (except last layer) is passed through batch normalization technique, which is followed by ReLU activation function.

Similarly, D in a ACGAN takes an input image (real image x or fake image $G(z, y)$) of dimension $56 \times 56 \times 1$ which is passed onto series of strided convolutional layers. The last convolutional layer $8 \times 8 \times 256$ is flattened out and passed into sigmoid activation function to provide the degree of real ($\simeq$ 1) or fake ($\simeq$ 0) score of an image. Each convolutional layer (except dense layer) is passed through batch normalization technique, which is followed by LReLU activation function.

Furthermore, to access the side information of predicting class labels y of training data x or generated data $G(z, y)$, the final dense layer of D network is passed through softmax activation function. Appendix A (Table 7) shows the network architecture of ACGAN. The loss function of ACGAN is formulated in Table 1.

e: WASSERSTEIN GAN

In standard GAN, generator is generally an encoding vector sampled from random distribution of low dimensions ($z = 100$), which is then mapped to a high dimensional space (e.g., 4096 dimensions) to generate synthesized images. During the training phase, the G is encouraged to produce the distribution of samples $p_g(x)$ to match the real sample data $p_d(x)$. For an optimal trained GAN, these distributions should nearly match each other [1]. However, probability distribution of generated samples defined by a high-dimensional space (4096 dimensions) is still defined under a definite sample dimensions (100 dimensions). Hence, the support set for generated sample distributions constitutes a low-dimensional manifold with upto 100 dimensions in a 4096-dimensional space. As a result, the probability of overlapping between generated samples $p_g(x)$ and real data samples $p_d(x)$ is close to 0. This leads to gradient disappearance problem, as Jensen-Shannon (JS) divergence, which is a measure of similarity between probability distribution of real samples and generated samples will become a constant, therefore, halting the training process [58].

To solve the unstable training process of GAN, Wasserstein distance, also called as Earth-Mover (EM) distance, is used to measure the distance between real samples and generated samples, defined as:

$$W(p_r, p_d) = \inf_{\gamma \epsilon \prod(p_r, p_d)} \mathbb{E}_{x,y} \sim \gamma [\| \text{x-y} \|] \tag{11}$$

where $\prod(p_r, p_d)$ is set of all joint distributions whose marginals are p_r and p_d respectively. γ is every possible distribution. Unlike the standard GAN cost function, WGAN [59] is more likely to provide gradients update for generator, however, the cost function derived for WGAN relies on discriminator that is usually termed as the *critic*. The *critic* should satisfy strong conditional lipschitz continuity.

Practically, the lipschitz continuity is implemented by clamping the D parameters to be in a certain range. In the G network, a latent noise vector z with dimension 100 is drawn from a uniform distribution, which is mapped to series of high dimensional dense layers to produce fake image $G(z)$ of dimension $56 \times 56 \times 1$. Each dense operation is passed through batch normalization ($\beta = 0.6$) followed by LReLU ($\alpha = 0.2$) activation function. The D network takes real image x or fake image $G(z)$ which is passed through series of multiple dense layers to provide a discrimination of real images from fake ones. We train the WGAN using RMSProp [60] optimizer. After each gradient update, we clamp the D weights to lie in a range of $(-0.01, 0.01)$, whereas, we train the G network after every $n_{critic} = 5$ iterations. Appendix A (Table 8) shows the network architecture of WGAN, and the loss function is tabulated in Table 1.

f: WASSERSTEIN GAN WITH GRADIENT PENALTY

WGAN improves the optimization of standard GAN by improvising constraint without changing the architecture of it. However, WGAN limits the D weights to strongly meet the conditional Lipschitz continuity. Thus, the forced weight cutting can easily cause the gradients to vanish or explode. To solve this problem, Gulrajani *et al.* in [61] proposed a gradient penalty method termed as WGAN with Gradient Penalty (WGAN-GP). Instead of using weight pruning of WGAN, WGAN-GP calculates the weight gradient according to the input of D network, then, it penalizes the gradient norm to satisfy the Lipschitz constraint [61]. The loss function of WGAN-GP is formulated in Table 1.

In order to generate synthetic ligature images using WGAN-GP, we followed the same dense layer architecture that is implemented for WGAN, as shown in Appendix A (Table 8). The value for gradient penalty co-efficient λ is kept to 10, as according to authors in [61], it works well across different architectures and datasets. Furthermore, no batch normalization is applied to the D network, as we penalize the norm of $D's$ gradients with respect to each individual sample, and not on the entire batch. [61]. We train the G after every $n_{critic} = 5$ iterations, whereas, ADAM optimizer with learning rate $lr = 0.0002$, $\beta_1 = 0.5$, and $\beta_2 = 0.999$ is used in the training phase.

g: CYCLE GAN

The above incorporated GAN models produce synthesize images from random noise distribution z or embedding an extra conditioning constraint y to augment a ligature specific images. However, these GAN models can not ensure the validity that the generated ligatures are similar to the input set of ligatures but with different style. Style GAN is another interesting application of GAN, which is also known as image translation, commonly used for transforming images from one style to another. Earlier, image-to-image translation (pix2pix) framework [63] is used to translate one possible representation of a scene into another scene. However, such one-to-one style transfer model requires a paired set of training data forehand, and in many cases, obtaining paired training data can be difficult and equally expensive. To break the constraints of paired datasets, the Cycle Consistent Generative Adversarial Network (CycleGAN) [62] utilizes a cycle loss function that learns image-to-image translation between two un-ordered image collections.

Given one set of domain image in domain X and a different set in domain Y, we train a mapping $G : X \rightarrow Y$, such that, the output $\widehat{y} = G(x)$, $x \in X$, remains indistinguishable from images $y \in Y$ by an adversary D_Y, which is trained to distinguish $\widehat{y}$ from y. Similarly, the other domain Y performs the opposite by training a generator $F : Y \rightarrow X$ to produce $\widehat{x} = F(y)$, which is further passed on to its adversarial discriminator D_X to check whether it is indistinguishable from domain X. We train both the mapping G and F simultaneously, and introduce two cycle consistency losses that encourage the reconstruction of image x using $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$ (forward cycle-consistency loss) and image y using $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$ (backward cycle-consistency loss) respectively. Combining the standard GAN adversarial loss function and cycle-consistency loss function, the loss function of CycleGAN is formulated in Table 1.

In our work, we have incorporated CycleGAN to translate original ligatures images to its rotated counterpart and vice versa. The rotation based augmentation technique achieved better accuracy results as compared to the scaling technique. Thus, we created unpaired training dataset comprising of original ligatures and a target set consisting of random rotated ligatures ($-25°$,$25°$), with no information provided as to which original ligature matches which rotated ligature. Each ligature class is trained on an individual CycleGAN model and from each model, we generated 100 ligature images. In the G network, we inserted 9 residual blocks between convolutional blocks which make use of strided and fractional strided operations. Each convolutional layer is instance normalized which is followed by ReLU activation function. For the D network, we used 3×3 overlapping patches at the output of last convolutional layer, whose aim is to classify whether these patches are real or fake. Except first layer, each convolutional layer is instance normalized which is followed by LReLU activation function. For all the experiments, we used ADAM optimizer with a batch size of 1. All the networks were trained from scratch for 10 number of epochs, with a learning rate $lr = 0.0002$, $\beta_1 = 0.5$, and $\beta_2 = 0.999$. Appendix A (Table 9) shows the network architecture of CycleGAN.

G. HOG FEATURE EXTRACTION

Assuming an image with size vector S ([rows columns]), the HOG extraction cell size C, extraction block size B, block overlap B_O and number of bins in the histogram $Bins$, the HOG feature vector size of N elements is calculated by the following equations:

$$BPI = (S./C - B)./(B - B_O + 1) \tag{12}$$
$$N = \Pi(BPI, B, Bins) \tag{13}$$

Here, in equation (12), BPI is the blocks per image. We can get the vector size N by calculating the element-vise product of BPI, B and $Bins$.

The data generation techniques provided a synthesized database comprising of diverse ligature images for a respective class. For each ligature image, we extract a HOG feature vector of 5184 (4×4 cell) and 900 (8×8 cell) elements using equations (12) and (13). The feature vectors are then used to train the LSTM network. In our study, we took $B = [2\ 2]$, $B_O = [1\ 1]$ and $Bins = 9$. The cell sizes used are 4×4 and 8×8. Higher cell sizes were not used because from equation (12) it can be seen that larger cell sizes will produce smaller feature vectors and hence, less information for the classifier.

H. LONG SHORT TERM MEMORY (LSTM) NETWORK

Two different LSTM networks are used according to the input vector sizes. The LSTM layer consists of 4000 neurons and a ReLU as an activation function. The size of output classification layer is equal to the number of classes of the database, which in our case is 317. The database is distributed into mini-batches of 50 images and is trained for 50 epochs. The Stochastic Gradient Descent with Momentum (SGDM) optimizer with L2 regularization factor of 0.0001, initial learning rate of $lr = 0.01$, learning rate drop factor of 0.1 and learn rate drop period 10 is used to train the LSTM Network.

IV. RESULTS

To evaluate the performance of different GAN models, that are, standard GAN, DCGAN, CGAN, ACGAN, WGAN, WGAN-GP, and CycleGAN are compared on UNHD dataset. The results of these models are verified through qualitative analysis and further classifying the HOG features of generated samples using LSTM network.

A. QUALITATIVE ANALYSIS

The basic application of GAN as a generative model is sample generation, however, the quality of generated samples is a key indicator to analyze the performance of GAN models. Fig. 6 shows the ligature images generated using geometric and learning based techniques. The performance results of different GAN models are obtained after evaluating each variant after 1000 number of epochs, except CycleGAN which is evaluated after 10 number of epochs.

The samples generated by standard GAN are not at par as compared to other GAN models with structural improvements. DCGAN upgraded the quality of image synthesis by incorporating a pair of deep convolutional G and D networks. As a result, the generated images are more diverse as compared to standard GAN, as found in Fig. 6. This is due to the inclusion of batch normalization technique [55] and Leaky ReLUs [33] which enhances the network stability and performance. However, in DCGAN, the quality of generated ligatures from UNHD database are still of low resolution. The reason can be incurred that, training dataset contains a versatile pattern of handwritten ligatures. The generated ligatures are diversified, yet, it could not precisely reproduced clean patterns of ligatures.

Compared to image generation of standard GAN and DCGAN, the quality of ligatures generated through CGAN and ACGAN produced better resolution results, as these models add controllable condition to the G network. This results in a faster convergence rate. The earlier implementation of CGAN produced noisy ligature images which contain high volume of salt and pepper noise originating at the background. As a result, lower accuracy was achieved when HOG features were trained on LSTM network. To compensate this problem, we have introduced individual dense operations for random noise vector z and conditional label y. The notion of using such dense operations in CGAN is to capture the required non-spatial mapping from respective latent vector and image label to the intermediate image features. A random distribution of noise vector produces different set of noise values, thus, incorporating a prior dense operation learns the inherent relationship between different noise vectors and intermediate features that belong to an individual ligature class during the training phase [64]. As a result, a better and clean ligature images are generated as compared to the conventional CGAN counterpart.

On the other hand, the quality of generated ligatures in ACGAN are not at par as compared to CGAN. This is because of fractional-srided and strided convolutional layers in a respective G and D architecture of ACGAN. The inclusion of shared weights in convolutional layers prevent the network to gather any subtle variations that occur in different spatial zones of the same convolutional filter, whereas, the initial dense layers of CGAN explicitly transfer the subtle information to intermediate layers, and produce more realistic ligatures at the output [64]. Hence, ACGAN restricted its capacity to generate different writing styles of an individual ligature class.

A better training stability is obtained using WGAN and WGAN-GP, which theoretically modifies the loss function of standard GAN. As WGAN reduces the horizontal distance (Wasserstein distance) between generated samples $G(z)$ and real samples x, the quality of ligatures produced by WGAN are more diverse as compared to earlier versions of GANs (Standard GAN, DCGAN, CGAN, ACGAN). WGAN captured the ability to learn the probability distribution of versatile ligatures originating from a single class. This includes minute details that correspond to dots or a tiny slash running diagonally above the ligature (from upper right to lower right). Fig. 6 shows the ligatures that are produced through WGAN.

Compared to WGAN, the quality of ligatures generated by WGAN-GP is more better and stable, as WGAN-GP further adds a constraint and optimizes the WGAN loss function, as shown in Table 1. WGAN-GP does not require hyperparameter tuning and trains successfully on variety of image generation tasks [61]. However, the training procedure showed that convergence rate of WGAN-GP is slowest, as it took more time to converge under the same UNHD dataset.

CycleGAN model works better when the two datasets have a similar structural pattern. By utilizing a two-step transformation procedure to realize a self-constraint, the model learns to generate rotated ligature images from the original ligatures, and vice-versa. Compared to WGAN and WGAN-GP, the quality of samples generated using both the generators G (original to rotated) and F (rotated to original) of CycleGAN are equally consistent in terms of capturing the minute details of ligature images. Fig. 7 shows the original ligature images generated from the rotated ligature image database. The generator $F(y)$ learns a mapping $F : Y \rightarrow X$ from rotated database to the original database and produces the equivalent distribution of original ligature $\widehat{x}$ from the rotated one y.

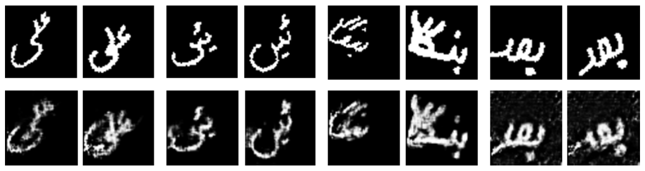

FIGURE 7. Rotated ligatures and its original counterpart. The samples in the first row are rotated ligature images (input y), and the samples in the second row are the generated ligatures $\widehat{x} = F(y)$.

B. CLASSIFICATION OF HOG FEATURES USING LSTM

Table 2 shows the results of this study. Simulations were conducted for each sample generation technique where the cell size from feature extraction process was varied. Note that, LSTM classifier does not yield good results when trained on scaled data. In fact, the accuracy obtained while training the network on original UNHD data set was 10% higher than that obtained from the scaled data. The rotated images produced much better results than the scaled images. This shows that variation within a class varies more in slants and orientation rather than the size. The reason can be incurred that the writing styles of different people varies more in slants and curves of the words instead of the size.

As expected, the learning based techniques proved to be more efficient in recognizing the ligatures than the non-learning based techniques. The rotated images however, out performed the standard GAN generated images, as standard GAN could not capture the prominent ligature patterns in a single class. Compared to the standard GAN, for HOG cell size 4 × 4, almost 15% increase in accuracy is obtained in DCGAN, while, a 10% increase is obtained for HOG cell size 8×8. Therefore, DCGAN performed better than standard GAN, as it consists of convolutional layers which are more prone to locate spatial correlations and thus, a better option for image generation.

TABLE 2. Accuracy results of different data generation techniques when HOG features are classified using LSTM.

Data Generation Technique	HOG Cell size	
	4x4	8x8
Without Data Generation	57.9%	51.9%
Scaled	47.86%	43.1%
Rotated	74.38%	69.38%
GAN	70.01%	58.39%
DCGAN	84.72%	68.09%
ACGAN	85.72%	64.52%
CGAN	96.69%	95.52%
WGAN	97.36%	90.55%
WGAN-GP	98.45%	89.71%
Cycle-GAN	98.96%	94.64%

The CGAN performed even better than the former two. GAN is conventionally an unsupervised learning technique, which means it does not incorporate the effect of labels. Hence, the GAN does not know the identity of each sample with respect to its class. However, in CGAN, the G and D take label of the sample as an input, which makes the job much easier, as the network is now being trained in a supervised manner. This results in a more stable training of the network and allows the G to produce images of the desired class. For HOG cell size 4 × 4, CGAN achieved the notable accuracy of 96.69%, thus, a significant increase in performance (more than 10%) is obtained as compare to the results of DCGAN. For HOG cell size 8 × 8, a 25% increase in accuracy can be seen as compared to the DCGAN results.

The results acquired from ACGAN generated images were not at par with those generated by the CGAN. For HOG cell size 4 × 4, accuracy results are declined by 10% as compared to CGAN, where it achieved an accuracy of 85.72%. This is because, even though the G in both networks are similar, the D in ACGAN is not given the label of target image as an input. Furthermore, the network architecture comparison of ACGAN and CGAN, discussed in the previous subsection IV-A also reveals the reason that why CGAN out performed ACGAN in the word spotting process.

Compared to CGAN, the WGAN and WGAN-GP produced better performance results. For HOG cell size 4 × 4, WGAN and WGAN-GP obtained an accuracy of 97.36% and 98.45% respectively, whereas, for cell size 8 × 8, both GANs (WGAN and WGAN-GP) performed better than the other variants of GAN (except CGAN). This is because of the Wasserstein computation that WGAN and WGAN-GP performs, thus, providing the ability to produce diverse ligatures, and hence, improved performance results.

Under the domain of image translation, we incorporated CycleGAN to translate between (un-ordered) original

TABLE 3. A comparison of proposed methodology with existing word spotting techniques.

System	Language	Segmentation	Classifier	Accuracy
Proposed System (Cycle-GAN using 4x4 cell)	Urdu	Ligatures	LSTM	98.96%
Khayyat et al. [2]	Arabic	Ligatures	SVM	84%
Dehghan et al. [3]	Farsi	Characters	SVM	95%
Patthan et al. [4]	Urdu	Characters	HMM	93.59%

TABLE 4. Standard GAN architecture details for generator and discriminator network.

Generator	Activation	Output Shape
Latent vector	-	$100\times1\times1$
Dense	ReLU	$128\times1\times1$
Dense	ReLU	$256\times1\times1$
Dense	ReLU	$512\times1\times1$
Dense	ReLU	$1024\times1\times1$
Dense	ReLU	$2048\times1\times1$
Dense	ReLU	$4096\times1\times1$
Dense	Tanh	$3136\times1\times1$

Discriminator	Activation	Output Shape
Input Image	-	$3136\times1\times1$
Dense	LReLU	$2048\times1\times1$
Dense	LReLU	$1024\times1\times1$
Dense	LReLU	$512\times1\times1$
Dense	LReLU	$256\times1\times1$
Dense	LReLU	$128\times1\times1$
Dense	Sigmoid	1×1

TABLE 5. DCGAN architecture details for generator and discriminator network.

Generator	Activation	Output Shape
Latent vector	-	$100\times1\times1$
Dense	ReLU	$7\times7\times128$
Upsample Conv 3×3 Zero pad 1	ReLU	$14\times14\times128$
Upsample Conv 3×3 Zero pad 1	ReLU	$28\times28\times64$
Upsample Conv 3×3 Zero pad 1	ReLU	$56\times56\times32$
Conv 3×3 Zero pad 1	Tanh	$56\times56\times1$

Discriminator	Activation	Output Shape
Input Image	-	$56\times56\times1$
Conv 3×3 Zero pad 1 Stride 2	LReLU	$28\times28\times32$
Conv 3×3 Zero pad 1 Stride 2	-	$14\times14\times64$
Zero pad 1	LReLU	$15\times15\times64$
Conv 3×3 Zero pad 1 Stride 2	LReLU	$8\times8\times128$
Conv 3×3 Zero pad 1 Stride 1	LReLU	$8\times8\times256$
Dense	Sigmoid	1×1

ligatures and rotated ligatures, and vice-versa. CycleGAN provides an auxiliary cycle-consistent loss function along with the standard adversarial loss, hence, the generated ligature images are equally diverse as in the case of WGAN and WGAN-GP. We achieved the best accuracy results using the CycleGAN on HOG cell size 4×4. In general, it was observed that training the LSTM network with HOG features extracted using a 4×4 cell produces better results than 8×8 cell. This is because the larger cells extract gradients from a larger area in the image and minor details may get ignored. As 4×4 cell is smaller, it covers a smaller area in the image and hence, captures more minute details of the image. Therefore, it produces better results as training on more detail of the image, enables the LSTM network to classify with better efficiency.

A comparison with previous state of the art works regarding word spotting on Arabic script languages is provided in Table 3. The previous studies use normalization in the pre-processing phase to make the images suitable for the classifier. This will add delay in real-time word spotting process. It can also be seen that random data generated through GANs produced better accuracy results, as compared to normalization and correction techniques required in the pre-processing stage. Due to random data generation, the images generated by GANs are devoid of human intervention. Thus, training the classifier on such images also enables it to recognize words, despite of unforeseen variations in hand writing styles.

Furthermore, the classifiers used in the the previous studies are not deep networks, whereas, in our work, the system learns to capture a better distribution due to the deep network. Even, if it takes a longer time to train, but, during real-time, the classifier operates with a better efficiency.

TABLE 6. Conditional GAN architecture details for generator and discriminator network.

Generator	Activation	Output Shape
Latent Vector+Labels (Concatenated Vector)	-	512×1×1
Dense	LReLU	1024×1×1
Dense	LReLU	2048×1×1
Dense	LReLU	4096×1×1
Dense	Tanh	3136×1×1

Discriminator	Activation	Output Shape
Input Image + Labels (Concatenated Vector)	-	4352×1×1
Dense	LReLU	2048×1×1
Dense	LReLU	1024×1×1
Dense	LReLU	512×1×1
Dense	LReLU	256×1×1
Dense	Sigmoid	1×1

TABLE 7. ACGAN architecture details for generator and discriminator network.

Generator	Activation	Output Shape
Latent vector + Labels	-	110×1×1
Dense	ReLU	7×7×128
Upsample Conv 3×3 Zero pad 1	ReLU	14×14×128
Upsample Conv 3×3 Zero pad 1	ReLU	28×28×64
Upsample Conv 3×3 Zero pad 1	ReLU	56×56×32
Conv 3×3 Zero pad 1	Tanh	56×56×1

Discriminator	Activation	Output Shape
Input Image	-	56×56×1
Conv 3×3 Zero pad 1 Stride 2	LReLU	28×28×32
Conv 3×3 Zero pad 1 Stride 2	-	14×14×64
Zero pad 1	LReLU	15×15×64
Conv 3×3 Zero pad 1 Stride 2	LReLU	8×8×128
Conv 3×3 Zero pad 1 Stride 1	LReLU	8×8×256
Dense	Sigmoid	1×1

V. CONCLUSION

In this study, a system was proposed for the task of word spotting of hand written Urdu text. We proposed the methodology of generating handwritten ligature image database by segregating the ligatures from handwritten document image lines. We also explained the procedure of generating a diversified synthetic database that allows the classifier to spot similar ligatures in various handwriting styles without normalizing the ligatures. We have also shown the effectiveness of extracting HoG features of ligature images that captures the ligature geometrical information in a form of vector. This provides a better recognition rate when trained on LSTM network.

TABLE 8. WGAN and WGAN-GP architecture details for generator and discriminator network.

Generator	Activation	Output Shape
Latent vector	-	100×1×1
Dense	LReLU	128×1×1
Dense	LReLU	256×1×1
Dense	LReLU	512×1×1
Dense	LReLU	1024×1×1
Dense	LReLU	2048×1×1
Dense	LReLU	4096×1×1
Dense	Tanh	3136×1×1

Discriminator	Activation	Output Shape
Input Image	-	3136×1×1
Dense	LReLU	4096×1×1
Dense	LReLU	2048×1×1
Dense	LReLU	1024×1×1
Dense	LReLU	512×1×1
Dense	LReLU	256×1×1
Dense	Sigmoid	1×1

The overall aim of this paper is to present a learning based word spotting approach that has the ability to recognize the instances of a keyword, irrespective of orientation and the style of handwriting. The experimental results of Section IV-B shows a successful impact of learning based data generation results on defined LSTM network (Fig. 2).

The overall result shows that the system trained on scaled data did not yield good results, but rotated data produced much better results. This proves that the variation in writing style is more due to rotation and slants, rather than in the size of text. The standard GAN produced much better results than the scaled data. However, the DCGAN outperformed standard GAN due to convolutional layers in the generator and the discriminator. The ACGAN also showed a higher recognition rate than standard GAN and DCGAN. This is because the generator in ACGAN is provided with the label of the target image to be produced. The other GAN with conditioning label, i.e., CGAN outperformed ACGAN, as this GAN provides label to the generator, as well as, the discriminator making the task easier for both networks.

TABLE 9. CycleGAN architecture details for generator and discriminator network.

Generator	Activation	Output Shape
Input Image	-	56×56×1
ReflectionPad2D Conv 2×2 InstanceNorm2d	ReLU	57×57×64
Conv 3×3 InstanceNorm2d	ReLU	29×29×128
Conv 3×3 InstanceNorm2d	ReLU	15×15×256
9 Residual blocks of ReflectionPad2D Conv 2×2 InstanceNorm2d	ReLU	15×15×256
Upsample Conv 3×3 Zero pad 1 InstanceNorm2d	ReLU	30×30×128
Upsample Conv 3×3 Zero pad 1 InstanceNorm2d	ReLU	60×60×64
ReflectionPad2D Conv 7×7	Tanh	56×56×1

Discriminator	Activation	Output Shape
Input Image	-	56×56×1
Conv 4×4 Zero pad 1 Stride 2	LReLU	28×28×64
Conv 4×4 Zero pad 1 Stride 2 InstanceNorm2d	LReLU	14×14×128
Conv 4×4 Zero pad 1 Stride 2 InstanceNorm2d	LReLU	7×7×256
Conv 4×4 Zero pad 1 Stride 2 InstanceNorm2d	LReLU	3×3×512
Zero pad 2D Conv 4×4 Zero pad 1 (patchGAN)	-	3×3×512

The best performance was shown by WGAN-GP and Cycle-GAN, i.e., 98.45% and 98.96% respectively, as these GANs, because of their strong theoretical loss function foundation learn to replicate the ligature images from training data. This proves that the minute patterns in ligature images which can be a dot or slash running diagonally above the ligature (from upper right to lower right) is equally important to achieve a high recognition rate. Therefore, the higher recognition rate is what makes a network robust to learn the distribution of a language, irrespective of different orientation, pattern or styles.

ACKNOWLEDGMENT

(Faiq Faizan Farooqui and Muhammad Hassan contributed equally to this work.)

APPENDIX A GAN ARCHITECTURES

See Tables 4–9.

REFERENCES

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[2] M. Khayyat, L. Lam, and C. Y. Suen, "Learning-based word spotting system for arabic handwritten documents," *Pattern Recognit.*, vol. 47, no. 3, pp. 1021–1030, Mar. 2014.

[3] M. Dehghan, K. Faez, M. Ahmadi, and M. Shridhar, "Handwritten farsi (Arabic) word recognition: A holistic approach using discrete HMM," *Pattern Recognit.*, vol. 34, no. 5, pp. 1057–1065, May 2001.

[4] I. K. Pathan, A. A. Ali, and R. Ramteke, "Recognition of offline handwritten isolated urdu character," *Adv. Comput. Res.*, vol. 4, no. 1, p. 5, 2012.

[5] A. P. Giotis, G. Sfikas, B. Gatos, and C. Nikou, "A survey of document image word spotting techniques," *Pattern Recognit.*, vol. 68, pp. 310–332, Aug. 2017.

[6] I. Shamsher, Z. Ahmad, J. K. Orakzai, and A. Adnan, "OCR for printed Urdu script using feed forward neural network," *World Acad. Sci., Eng. Technol.*, vol. 23, pp. 172–175, Aug. 2007.

[7] R. Saabni and J. El-Sana, "Keywords image retrieval in historical handwritten arabic documents," *J. Electron. Imag.*, vol. 22, no. 1, p. 13016, 2013.

[8] A. Abidi, I. Siddiqi, and K. Khurshid, "Towards searchable digital urdu Libraries–A word spotting based retrieval approach," in *Proc. Int. Conf. Document Anal. Recognit.*, Sep. 2011, pp. 1344–1348.

[9] J. Tariq, U. Nauman, and M. Umair Naru, "Softconverter: A novel approach to construct OCR for printed urdu isolated characters," in *Proc. 2nd Int. Conf. Comput. Eng. Technol.*, 2010. p. 495.

[10] T. Sari and A. Kefali, "A search engine for arabic documents," in *Proc. Colloque Int. Francophone Ecrit Document*, 2008, pp. 97–102.

[11] Y. Pourasad, H. Hassibi, and A. Ghorbani, "A word spotting method for farsi machine-printed document images," *Turkish J. Electr. Eng. Comput. Sci.*, vol. 21, no. 3, pp. 734–746, 2013.

[12] A. Ul-Hasan, S. B. Ahmed, F. Rashid, F. Shafait, and T. M. Breuel, "Offline printed urdu nastaleeq script recognition with bidirectional LSTM networks," in *Proc. 12th Int. Conf. Document Anal. Recognit.*, Aug. 2013, pp. 1061–1065.

[13] S. Shabbir, "Optical character recognition system for urdu words in Nastaliq font," Tech. Rep., 2016.

[14] T. Abu-Ain, S. Abdullah, S. N. Huda, K. Omar, A. Abu-Ein, B. Bataineh, and W. Abu-Ain, "Text normalization method for arabic handwritten script," *J. ICT Res. Appl.*, vol. 7, no. 2, 2013.

[15] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 545–552.

[16] P. Krishnan, K. Dutta, and C. V. Jawahar, "Word spotting and recognition using deep embedding," in *Proc. 13th IAPR Int. Workshop Document Anal. Syst. (DAS)*, Apr. 2018, pp. 1–6.

[17] L. Kang, J. I. Toledo, P. Riba, M. Villegas, A. Fornés, and M. Rusinol, "Convolve, attend and spell: An attention-based sequence-to-sequence model for handwritten word recognition," in *Proc. German Conf. Pattern Recognit.* Springer, 2018, pp. 459–472.

[18] J. Sueiras, V. Ruiz, A. Sanchez, and J. F. Velez, "Offline continuous handwriting recognition using sequence to sequence neural networks," *Neurocomputing*, vol. 289, pp. 119–128, May 2018.

[19] 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

[20] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*. [Online]. Available: http://arxiv.org/abs/1412.3555

[21] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Feb. 1998.

[22] L. Gui, X. Liang, X. Chang, and A. G. Hauptmann, "Adaptive context-aware reinforced agent for handwritten text recognition," in *Proc. BMVC*, 2018, p. 207.
[23] J. Almazan, A. Gordo, A. Fornes, and E. Valveny, "Word spotting and recognition with embedded attributes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 12, pp. 2552–2566, Dec. 2014.
[24] S. Sudholt and G. A. Fink, "PHOCNet: A deep convolutional neural network for word spotting in handwritten documents," in *Proc. 15th Int. Conf. Frontiers Handwriting Recognit. (ICFHR)*, Oct. 2016, pp. 277–282.
[25] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015.
[26] N. Gurjar, S. Sudholt, and G. A. Fink, "Learning deep representations for word spotting under weak supervision," in *Proc. 13th IAPR Int. Workshop Document Anal. Syst. (DAS)*, Apr. 2018, pp. 7–12.
[27] G. Retsinas, G. Sfikas, and B. Gatos, "Transferable deep features for keyword spotting," *Proceedings*, vol. 2, no. 2, p. 89, Jan. 2018.
[28] L. van der Maaten, "Accelerating t-SNE using tree-based algorithms," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3221–3245, Oct. 2014.
[29] F. Biadsy, J. El-Sana, and N. Y. Habash, "Online arabic handwriting recognition using hidden Markov models," Tech. Rep., 2006.
[30] P. Dreuw, G. Heigold, and H. Ney, "Confidence-based discriminative training for model adaptation in offline arabic handwriting recognition," in *Proc. 10th Int. Conf. Document Anal. Recognit.*, 2009, pp. 596–600.
[31] P. Krishnan and C. Jawahar, "Matching handwritten document images," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2016, pp. 766–782.
[32] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, p. 60, Dec. 2019.
[33] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015, *arXiv:1511.06434*. [Online]. Available: http://arxiv.org/abs/1511.06434
[34] E. Alonso, B. Moysset, and R. Messina, "Adversarial generation of handwritten text images conditioned on sequences," 2019, *arXiv:1903.00277*. [Online]. Available: http://arxiv.org/abs/1903.00277
[35] Z. Qian, K. Huang, Q. Wang, J. Xiao, and R. Zhang, "Generative adversarial classifier for handwriting characters super-resolution," 2019, *arXiv:1901.06199*. [Online]. Available: http://arxiv.org/abs/1901.06199
[36] B. Chang, Q. Zhang, S. Pan, and L. Meng, "Generating handwritten chinese characters using CycleGAN," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2018, pp. 199–207.
[37] Z. Zhong, L. Jin, and Z. Xie, "High performance offline handwritten chinese character recognition using GoogLeNet and directional feature maps," in *Proc. 13th Int. Conf. Document Anal. Recognit. (ICDAR)*, Aug. 2015, pp. 846–850.
[38] H. Wang, Z. Qin, and T. Wan, "Text generation based on generative adversarial nets with latent variables," in *Proc. Pacific-Asia Conf. Knowl. Discovery Data Mining*. Springer, 2018, pp. 92–103.
[39] T. Bakker, "Image-to-image translation for handwriting generation using Gans," Ph.D. dissertation, 2019.
[40] A. R. Blanes, "Synthetic handwritten text generation," Tech. Rep.
[41] K. M. Kumar, H. Kandala, and N. S. Reddy, "Synthesizing and imitating handwriting using deep recurrent neural networks and mixture density networks," in *Proc. 9th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Jul. 2018, pp. 1–6.
[42] M. W. Sagheer, N. Nobile, C. L. He, and C. Y. Suen, "A novel handwritten urdu word spotting based on connected components analysis," in *Proc. 20th Int. Conf. Pattern Recognit.*, Aug. 2010, pp. 2013–2016.
[43] S. B. Ahmed, S. Naz, S. Swati, I. Razzak, A. I. Umar, and A. A. Khan, "Ucom offline dataset-an urdu handwritten dataset generation," *Int. Arab J. Inf. Technol.*, vol. 14, no. 2, 2017.
[44] S. A. Korkmaz, A. Akcicek, H. Binol, and M. F. Korkmaz, "Recognition of the stomach cancer images with probabilistic HOG feature vector histograms by using HOG features," in *Proc. IEEE 15th Int. Symp. Intell. Syst. Informat. (SISY)*, Sep. 2017, p. 000.
[45] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proc. 15th Annu. Conf. Int. speech Commun. Assoc.*, 2014, pp. 1–5.
[46] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.*, vol. 06, no. 02, pp. 107–116, Apr. 1998.
[47] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 6645–6649.
[48] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-9, no. 1, pp. 62–66, Jan. 1979.
[49] N. Javed, S. Shabbir, I. Siddiqi, and K. Khurshid, "Classification of urdu ligatures using convolutional neural Networks–A novel approach," in *Proc. Int. Conf. Frontiers Inf. Technol. (FIT)*, Dec. 2017, pp. 93–97.
[50] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2234–2242.
[51] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980
[52] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1139–1147.
[53] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[54] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," 2015, *arXiv:1505.00853*. [Online]. Available: http://arxiv.org/abs/1505.00853
[55] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*. [Online]. Available: http://arxiv.org/abs/1502.03167
[56] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014, *arXiv:1411.1784*. [Online]. Available: http://arxiv.org/abs/1411.1784
[57] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier Gans," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 2642–2651.
[58] Y.-J. Cao, L.-L. Jia, Y.-X. Chen, N. Lin, C. Yang, B. Zhang, Z. Liu, X.-X. Li, and H.-H. Dai, "Recent advances of generative adversarial networks in computer vision," *IEEE Access*, vol. 7, pp. 14985–15006, 2018.
[59] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," 2017, *arXiv:1701.07875*. [Online]. Available: http://arxiv.org/abs/1701.07875
[60] T. Tieleman and G. Hinton, "Divide the gradient by a running average of its recent magnitude. Coursera: Neural networks for machine learning," Tech. Rep., 2017.
[61] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5767–5777.
[62] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2223–2232.
[63] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1125–1134.
[64] S. Barua, S. Monazam Erfani, and J. Bailey, "FCC-GAN: A fully connected and convolutional net architecture for GANs," 2019, *arXiv:1905.02417*. [Online]. Available: http://arxiv.org/abs/1905.02417

FAIQ FAIZAN FAROOQUI received the B.S. degree in electrical engineering with major in electronics from Air University, Islamabad, Pakistan, in 2016, and the M.S. degree with a specialization in digital systems and signal processing from the National University of Sciences and Technology, Islamabad, Pakistan, in 2019.

He is currently working as a Research Assistant with the School of Electrical Engineering and Computer Sciences, National University of Science and Technology. His research interests include deep learning and image processing.

MUHAMMAD HASSAN received the B.S. degree in electrical engineering from the Center for Advanced Studies in Engineering (CASE), Pakistan, in 2014, and the M.S. degree in electrical systems engineering with a specialization into machine learning from Paderborn University, Germany, in 2017.

He worked as a Research Associate with ROSEN GmbH, Germany, and later on, as a Design Engineer with Technology Spirits, Pakistan. He is currently leads the machine learning group with the ASP Lab, NUST-SEECS. His research interests include generative models, data analytics and prediction, and developing AI based products.

SHAHZAD YOUNIS received the B.S. degree from the National University of Sciences and Technology, Islamabad, Pakistan, in 2002, the M.S. degree from the University of Engineering and Technology, Taxila, Pakistan, in 2005, and the Ph.D. degree from University Technology PETRONAS, Malaysia, in 2009, respectively.

Before joining the National University of Sciences and Technology (NUST), he was Assistant Manager at a research and development organization named AERO, where he worked on different signal processing and embedded system design applications. He is currently working as an Assistant professor with the Department of Electrical Engineering, School of Electrical Engineering and Computer Science (SEECS). He has published more than 25 articles in domestic and international journals and conferences. His research interests include statistical signal processing, adaptive filters, convex optimization biomedical signal processing, wireless communication modeling, and digital signal processing.

MUHAMMAD KASHIF SIDDHU received the B.Sc. degree in mathematics and physics from the University of Punjab, in 1998, the M.Sc. degree in computer science from International Islamic University Islamabad, in 2002, and the M.S. degree in computer engineering from the University of Engineering and Technology, Taxila, Pakistan, in 2010.

He is currently a Ph.D. Scholar with the School of Computer and Communication Engineering, Universiti Malaysia Perlis, Malaysia. His research interests include document image analysis, optical character recognition, machine learning, and deep learning.