Urdu Handwritten Ligature Generation using Generative Adversarial Networks (GANs)

Marium Sharif[1 [0000−0003−1302−0636]], Adnan Ul-Hasan[2 [0000−0001−6126−7137]], and Faisal Shafait[1,2 [0000−0002−0922−0566]]

[1] School of Electrical Engineering and Computer Sciences, National University of Sciences and Technology, Islamabad, Pakistan
Email: {msharif.msee19seecs,adnan.ulhassan,faisal.shafait}@seecs.edu.pk
[2] Deep Learning Lab, National Center of Artificial Intelligence, National University of Sciences and Technology, Islamabad, Pakistan

Abstract. Deep learning has significantly improved handwriting text recognition, esp. for Latin scripts. Arabic scripts including Urdu is a family of complex scripts and they pose difficult challenges for deep learning architectures. Data availability is a significant obstacle in developing Urdu handwriting recognition systems. Since gathering data is a costly and challenging task, there is a need to increase training data using novel approaches. One possible solution is to make a model that can generate similar yet different samples from the existing data samples. In this paper, we propose such models based on Generative Adversarial Networks (GANs) that have the ability to synthesize realistic samples similar to the original dataset. Our generator is class conditioned to produce Urdu samples of varying characters that differ in style. Visual and quantitative analysis convey that generated samples are of realistic nature and can be used to increase datasets. Synthesized samples integrated with the existing training set is shown to increase the performance of a handwriting recognition system.

Keywords: Generative Adversarial Networks · Handwriting Generation

1 Introduction

Handwritten documentation of knowledge is known to be a great achievement of mankind: the transition into history from prehistory is marked by the first ever written record. Mostly historic events have been documented as handwritten markings and scripts. Handwritten data has numerous applications, especially in healthcare and financial sectors whereby all data is still being predominantly saved in handwritten form. All of this written data has to be processed one way or the other. Although, modern Optical Character Recognition Systems (OCRs) have exhibited good performance against printed text [1], [2], handwritten text recognition is still lacking and could do with better performance results. This might be because of the lack of available data containing versatile handwritten

text. This is more so in the case of Arabic script as it comprises of cursive characters [8]. The shape of the characters also varies with respect to its placement in a given word. The many diacritics and dots are used to define the correct grammar and pronunciation of the word. There is also a difference in the inter-word and intra-word spacing in the Arabic script. This holds true for all languages that make use of the Arabic Script including Urdu, Farsi, Sindhi, Punjabi and many others. Most of the words in the Arabic script have one or more ligatures that are made up of a combination of two or more characters. This behaviour of the Arabic script further create complications in addition to the fact that every writer has their own unique handwriting style.

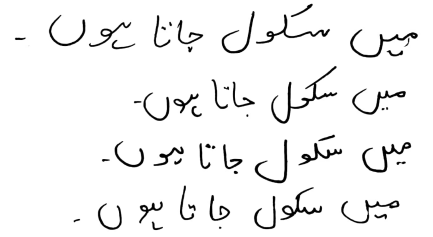

Fig. 1. Urdu phrase written by different writers. Some letters can get confused with others due to very similar shapes.

As in Figure 1, major changes in shape, geometry, orientation and size of words can be seen as a result of difference in handwriting styles. Due to this property of the Arabic script, mostly work is done on Arabic ligatures, which are the connected components of the Arabic alphabet. This work is focused on generation of these ligatures to increase training data samples having versatility.

Our work is focused on networks designed for generative modeling known as Generative Adversarial Networks (GANs) [11]. These adversarial networks consists of two components; the discriminator D and the generator G. G synthesizes new image samples and tries to replicate the original data images. D then tries to distinguish whether the image being scrutinized is real or fake. The primary objective of this study is thus to design and build a system that is capable of generating handwritten text images for the Urdu language thereby extending already available datasets with realistic samples.

This paper is further organized as follows. Section 2 outlines the relevant work done previously. Section 3 provides the detail description of the proposed approach. Section 4 describes the experimental details and Section 5 discusses the results, compares them with other similar works and also evaluates performance enhancement of a text recognizer by extending dataset with generated images. Section 6 concludes the paper with a summary of our contributions and future directions.

2 Previous Work

Available training data is often limited as data gathering is a costly and challenging task. This constricts the ability of a learning classifier to be successfully trained. One possible solution for increasing training samples is data augmentation [18]. Different data augmentation methods, mostly affine transformations such as scaling and rotation are used in handwritten images. Another novel solution is generative models wherein images are synthesized from an arbitrary input. This method is relatively new, gaining popularity in the deep learning world and rapidly improving.

Generative Adversarial Networks (GANs) and auto encoders are two examples employing generative modeling. Both have certain drawbacks; Auto encoders output blurry images [6] while traditional GAN outputs incomprehensible and noisy images. Further variations in the architectures and loss function of the original GAN network were introduced, each resulting in different improvements causing generation of higher resolution images such as that required for medical applications.

Alonso et al. [4] presents a GAN architecture that is able to produce images that contain string of characters. The network comprises of discriminator D and generator G. Two other networks are also introduced; first is a bidirectional LSTM network and second is a convolutional neural network (CNN) that has convolutional layers with LSTM layers at the end. Their work focused on generation of French and Arabic handwritten strings of fixed length and width. The generated images were incorporated with the existing dataset and improved accuracy was observed.

Fogel et al. [10] introduces a GAN network where along with the discriminator D, the generated image is also passed through a text recognizer R for evaluation. G and D used in this work are fully convolutional. G is proposed as a concatenation of generators that are class conditioned where each class is a single individual alphabet. Each generator outputs a patch having its respective input alphabet. All these patches are then upsampled causing them to overlap and ultimately output a string of characters. Evaluation results by R incorporated with the discriminator loss is used for weight update.

Farooqui et al. [9] focuses on the task of improvement of hand written Urdu word spotting using generation of data samples. GAN variants were used to generate sample images of handwritten Urdu Ligatures for increasing the training data. Seven GAN variants are implemented. Each GAN has been trained to produce only a single class of Urdu ligature at a time or at most 10 Urdu ligatures for class conditioned variants of GANs.

Chang et al. [7] proposes an architecture that uses samples of a font to construct samples of another font. Architecture compromises of an encoder network that produces low dimensional representation for input image. Feature representation of output font is generated by a transfer module. The decoder network generates the target font character and discriminator network is used to classify this generated font character. The produced images are also evaluated by HCCRGoogleNet [19] classifier.

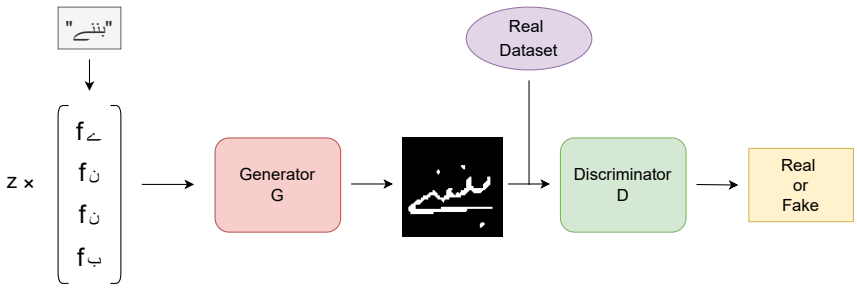

Fig. 2. Flowchart of proposed approach. For a four character ligature, four filters are concatenated. Note that filter for Urdu alphabet "ن" is used twice. Concatenated vector is multiplied with noise vector z and fed to G. Mix of resulting image from G and real data is given to D for evaluation.

3 Proposed Approach

The concept of GAN was first introduced by Goodfellow et al. in [11], whereby the network made use of two separate neural networks i.e. the Generator G and the Discriminator D. G approximates the training dataset by mapping random noise vector z and produces realistic data samples $G(z)$. We can express it as $G : G(z) \rightarrow R^{|x|}$ where $z \in R^{|z|}$ is latent vector noise, $x \in R^{|x|}$ is image generated from latent space and $|\cdot|$ is number of dimensions. The task of D, on the other hand, is to make an estimation such that $D : D(x) \rightarrow (0, 1)$, meaning it scores input image as either coming from the dataset (real$\simeq$1) or from the generator G (fake$\simeq$0). This approach permits G to learn about the underlying data distribution of the training dataset. In this way, the two network join in on a zero-sum min-max two player game. Figure 2 clearly demonstrates this concept being carried out for an Urdu ligature. Our approach focuses on reconstructing architectures for producing handwritten data samples for the Urdu language including all of its perplexing intricacies using an offline database. Analyzing the process in depth and the resultant samples also gives insight as to how the Urdu language poses challenges not present in case of the languages using the Latin script.

3.1 Fully Convolutional Generator

Handwriting is considered to be a local process when for this proposed network. Each handwritten character is only influenced by the letter before and after it respectively. Evidence supporting this theory can be found in previous works like [10] where they have successfully trained a generative architecture to produce strings of characters to form complete words for the Latin script. The proposed approach closely follows the concept of using a fully convolutional generator architecture as used in [10]. G can be thought of as generating each individual

Urdu alphabet, instead of the whole ligature at once. Then due to the overlapping feature of receptive fields in CNN [16], influence of neighbouring letters will be taken into account and effect the overall output of the architecture. Consequently, the generator is seen to be a concatenation of multiple generators that are identical and class conditioned. The class of each generator is a single alphabetical character in the Urdu lexicon. A single patch containing the required character is produced by each individual generator.

Convolution layers with upsampling layers are used for each layer that widens the overlap between the neighbouring characters thereby widening the receptive field. This works in a similar fashion as convolutional-transpose layer, which is the opposite of a convolutional layer. This allows these neighbouring characters to interact and create smooth transition within a ligature. For every character in a given ligature, a filter f^* is chosen from the filter bank that is as wide as the Urdu alphabet lexicon. For a ligature containing four characters, four filters from the filter bank will be concatenated and then multiplied by latent vector z of comparable dimensions. Region generated by filter of each character f^* is of the same dimension and receptive field of adjacent filters end up overlapping. This allows for flexible size and versatile cursive style for the output character. The overlap is responsible for different alphabets combining together to form connected ligatures which is a necessity. Moreover, learning dependencies between neighbouring characters allows the generator network to create different shapes and variations of the same character depending on the adjacent characters. This behavior is specially desired in Arabic script which assumes different character shape depending on its position in a ligature.

3.2 Fully Convolutional Discriminator

In the traditional GAN, role of D is to accurately distinguish between the original data samples and samples synthesized by G. Similarly, in our proposed model, D is used to score images as real or fake. The dicriminator is fully convolutional, just like the generator with its architecture almost the opposite to that of G. Actual handwritten samples mixed in with the synthesized samples are both given as input to the discriminator that evaluates these images and gives output. This output is then used in the loss function to update the weights of both the generator and discriminator respectively.

3.3 Objective Function

Training of GAN is a delicate and unstable process that may result in blurry images due to the diversification of dataset. Previously, researchers have tried different customization and optimizations to achieve different GAN variations thus attaining better learning stability [17].

For generation of diverse and distinct image samples, different loss functions have been implemented. The first is DCGAN [16] that uses the same loss function as that of the Standard GAN but differs in the architectures of the G and D. Along with that, the other two implementations carried out are Wasserstein GAN

(WGAN) [5] and Wasserstein GAN with Gradient Penalty (WGAN-GP) [12]. Using improved variations of standard GAN, provides better learning stability and helps in evading the potential mode collapse and balancing problems that are usually encountered with the training process of the traditional GAN.

DCGAN: Similar to standard GAN, a Deep Convolutional GAN (DCGAN) [16] employs the same training process and the same objective function. For DCGAN, the distinguishing factor from standard GAN is the change in architectures of G and D whereby convolutional layers replace the fully connected layers. Evidently, these layers prove to be more suitable for learning intrinsic properties of images. For G, transpose convolution (upsampling) is used. Objective function remains the same as that of standard GAN as is shown in Table 1.

Table 1. Objective functions for respective D and G networks

Model	Discriminator Loss Function	Generator Loss Function
DCGAN	$\underset{D}{max} L_D = E_{x\sim p_{data}}[log(D(x))] + E_{z\sim p_z}[log(1 - D(G(z)))]$	$\underset{G}{min} L_G = E_{z\sim p_z}[log(1 - D(G(z)))]$
WGAN	$\underset{D}{max} L_D = E_{x\sim p_{data}}[D(x)] - E_{z\sim p_z}[D(G(z))]$	$\underset{G}{min} L_G = -E_{z\sim p_z}[D(G(z))]$
WGAN-GP	$\underset{D}{max} L_D = L_D - \lambda E_{z\sim p_{data}}[\|\nabla D(\alpha x + (1 - \alpha G(z)))\| - 1)^2]$	$\underset{G}{min} L_G = -E_{z\sim p_z}[D(G(z))]$

WGAN: Conventionally in the training phase, G is pushed to produce samples whose distribution $p_g(x)$ matches real sample distribution $p_d(x)$. Ideally, this should work, but that is not always the case and gradient disappearance problem can occur making the training process unstable. To overcome the instability, Wasserstein distance is incorporated that quantifies the minimum cost that is utilized in transporting mass for converting data distribution q to data distribution p.

Wasserstein GAN (WGAN) [5] provides a much better gradient update for generator than the conventional cost function. Cost function is dependant upon D, also termed as the critic, satisfying strong conditional lipschitz continuity. For implementation purposes, D parameters are clipped to a certain range for lipschitz continuity. Respective loss functions of G and D are mentioned in Table 1.

WGAN-GP: Wasserstein GAN does not introduce any change in the architecture but rather improves performance by improvising the imposed constraint in

WGAN. Limiting the D weights to comply to the conditional Lipschitz continuity causes the gradients to either explode or vanish. This problem was easily solved by applying a gradient penalty method as proposed by Gulrajani et al. in [12] and was named as Wasserstein GAN with Gradient Penalty (WGAN-GP). Weight trimming was replaced by calculation of weight gradient in accordance with the D network inputs which then penalizes the gradient norm so that it satisfies the Lipschitz constraint [5]. Modified objective function of discriminator is mentioned in Table 1.

4 Experiment

4.1 Dataset

To test the proposed network paradigm, we use UCOM database [3]. The dataset contains only 48 unique lines of Urdu text, written by 100 different authors. Adopting the scheme explained in [9], the ligatures are segmented out from images of Urdu sentences through binarization, segmentation and then resizing to get ligatures of fixed dimension. The Urdu language comprises of a total of 40 unique alphabets. Standalone Urdu alphabets are also considered to be ligatures of a single character, most of which have between 200 to 300 repetitions. This holds true for some of the most common used 2 and 3 character ligatures as well. All of this pre-processing yields a total of 317 unique handwritten ligatures with varying number of samples in each class. A total of approximately 32k sample ligatures were obtained from the dataset.

4.2 Implementation Details

The network architecture is set to generate images at a fixed size of 64×64 pixels. G comprises of a filter bank as large as the Urdu alphabet. Size of each filter has been set to 32×8192. As described in Figure 2, for generation of a n-character ligature, n filters of the filter bank are selected in accordance to the characters. These filters are then concatenated and multiplied by a latent vector z to yield a vector of size 8×8192. This tensor is reshaped and then passed onto the convolutional layers followed by upsampling layer. LReLU and batch normalization [14] is used between these layers and a sigmoid activation function is used to produce the final output of size 64×64.

D network is almost the opposite of the G network without the spatial embeddings layer, that is, the filter bank. An image of 64×64 is given as input to the discriminator, which is passed through a series of layers i.e., the convolutional layer, LReLU layer, batch normalization, and max pool layer. Last layer is a linear layer that outputs a single output representing the score or probability of image being *real* or *fake*.

Same D and G networks are used for all three variants of GANs with only the varying loss functions being the determinant factor. Table 2 and Table 3 show implemented architectures of D and G respectively. For each GAN variant,

Table 2. Generator Architecture

Generator	Activation	Output Shape
Embedding Layer × Latent Vector z	-	8192×8
Conv	LReLU	$32 \times 4 \times 256$
Batch Normalization	-	$32 \times 4 \times 256$
Conv	LReLU	$32 \times 8 \times 128$
Batch Normalization	-	$32 \times 8 \times 128$
Conv	LReLU	$32 \times 16 \times 128$
Batch Normalization	-	$32 \times 16 \times 128$
Conv	LReLU	$32 \times 32 \times 64$
Conv	LReLU	$64 \times 64 \times 64$
Conv	LReLU	$32 \times 64 \times 64$
Conv	LReLU	$16 \times 64 \times 64$
Conv	Sigmoid	$1 \times 64 \times 64$

Table 3. Discriminator Architecture

Discriminator	Activation	Output Shape
Input Vector	-	64x64x1
Conv	LReLU	64x64x32
Conv	LReLU	32x32x64
Conv	LReLU	16x16x128
Conv	LReLU	16x8x256
Conv	LReLU	16x8x256
Batch Normalization	-	16x8x256
Conv	LReLU	16x4x256
Batch Normalization	-	16x4x256
Conv	LReLU	16x4x256
Batch Normalization	-	16x4x256
Conv	LReLU	16x2x256
Linear	Sigmoid	1x1

different hyper parameter settings were explored and the ones with the lowest FID score were then used for the generation of samples. For each mini-batch, the weights of D are updated 5 times as compared to a single weight update for G. Weight clipping was employed for both networks and D loss was given a gradient penalty of 10 in case of WGAN-GP implementation (Table 1).

5 Results

5.1 Qualitative Analysis

Application of GANs is the production of samples similar to the dataset whereby the performance of the generative models are analyzed using the quality of samples generated. Figure 3 shows the ligatures generated by different GAN variants.

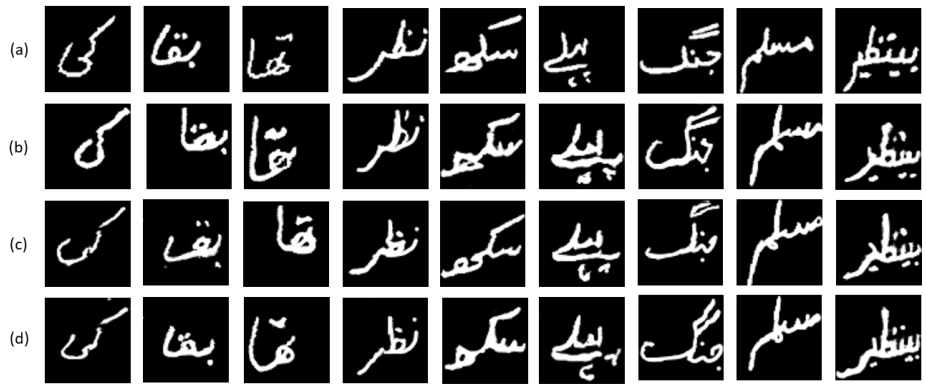

Fig. 3. Comparison of ligatures generated by GAN variants. Samples arranged in subfigures are (a) Original ligature samples (b) DCGAN (c) WGAN (d) WGAN-GP.

Ligatures generated by DCGAN show more diversity and have refined quality because of convolutional nature of the network architectures. This combined with the usage of techniques of batch normalization and Leaky ReLUs, increases the performance and stability of both the networks.

WGAN and WGAN-GP introduce further stability in the training process with imposed constraint on D to comply to the conditional Lipschitz continuity. Quality of sample ligatures produced are more diverse and finer as WGAN exercises reduction in distance between the generated $G(z)$ and real samples x. WGAN accurately enhances the model's capability to learn the probability distribution of the diversified ligatures belonging to the same class. This includes minuscule details such as a one or more dots or the tiny slash that is drawn diagonally over some of the ligatures. In comparison to WGAN, further quality improvement is seen in the samples produced by WGAN-GP. It practices further constraint for optimization of Wasserstein loss function. No hyperparameter tuning is required and successful training can be acheived for a number of image synthesizing tasks but, the convergence rate is the slowest in case of WGAN-GP as observed in the training process.

Comparison to Farooqui et al. [9] Main focus of their work was on the task of Urdu work spotting. They had explored various techniques for increasing dataset including augmentation and generative modeling. Seven GANs variants were implemented for dataset expansion to achieve better results for the aforementioned task. UNHD [3] dataset was used for this work. A single GAN was implemented to learn to generate only a single ligature class or at most 10 ligatures for class conditioned variants. Our model however is a sinlge architecture that is capable of producing all ligatures present in the dataset.

Since their main focus was not on the generation of images, their GAN variants had not been evaluated quantitatively and were subjected to qualitative

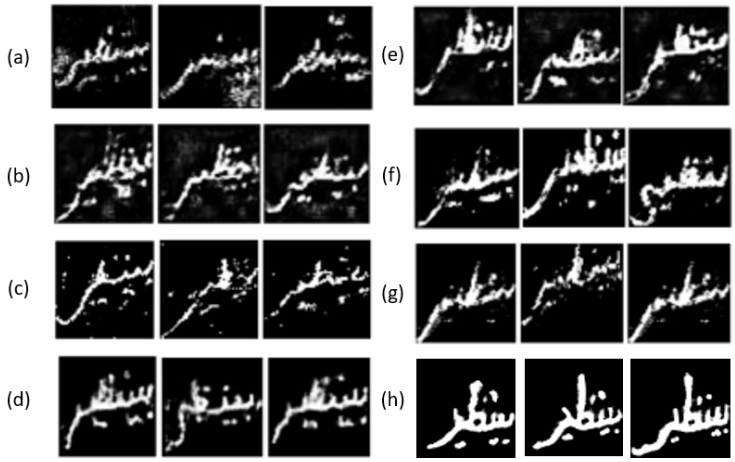

Fig. 4. Comparison of ligatures generated by different GANs for target ligature "بينظير". Samples arranged in subfigures (a) - (g) are GAN variant outputs of Farooqui et al. [9] namely, (a) Standard GAN, (b) DCGAN, (c) CGAN, (d) CycleGAN, (e) ACGAN, (f) WGAN, (g) WGAN-GP, subfigures (h) are outputs from our models

evaluation only. Figure 4, subfigrure (a) - (g) displays the results of their work while ours is displayed in Figure 4 subfigure (h). Observing these samples side by side, it can be seen that for GAN variants (a) - (c), the results are a bit blurred and have additional pepper noise present in each sample. For GAN samples (e) - (g), the results are a bit better but alot of details that are crucial for the identification of ligatures are lost. As they have claimed in their work, visual inspection agrees that the samples from CycleGAN [20] gave the best results. This variant best learns the details enclosed in a ligature and finer samples than the rest are also achieved. Our samples however are still better in quality with lesser noisy pixels and are visually more resolute. They are more accurately detailed with cleaner and sharper edges around each character and dots are drawn with better pixel intensities.

Comparison to Alonso et al. [4] Work in [4] was done on two databases namely the RIMES (French) and OpenHaRT (Arabic) database. Their model produced whole Arabic words rather than ligatures. All their evaluations were carried out on the RIMES dataset, no evaluation scores are available for the OpenHART database. Thus, qualitative comparison is carried out. As seen in Figure 5, the results given by their model do pick up details but the edges of words are rather bleeding out and a little murky whereas our samples have more clearer and crisper edges while encapsulating all the finer details.

Fig. 5. Comparing results presented in Alonso et al. [4] on the left with our results on the right

Table 4. FID and GS comparison for GANs Variants

GANS	FID Score	Geometric Score
DCGAN	21.45	7.82×10^4
WGAN	17.97	7.46×10^4
WGAN-GP	15.74	7.14×10^4

5.2 Quantitative Analysis

For evaluating the performance of proposed method quantitatively, Fréchet Inception Distance (FID) [13] and Geometric Score [15] were used. FID is used in the measurement of the feature distance between the generated and the real samples i.e. it measures the similarity between two sets of images. It is obtained by fitting two Gaussians on the feature representations of Inception Network and then calculating the Fréchet distance between them. GS compares the geometrical properties of the fundamental real and fake data manifold and provides a means to quantify mode collapse.

For every experiment conducted, FID was computed on the whole dataset vs equivalent number of generated samples i.e. approximately 32k samples, and GS was calculated on 5k real vs 5k generated samples with default parameter settings. Experiments run with different hyper parameters had FID calculated after every 10 iterations. Best FID was chosen from all experiments carried out and GS was also computed for this model setting. Visual inspection was relied upon for the verification of textual content. FID had shown to be in correlation with human judgement for visual quality of generated image samples and GS scores were also in favour of these findings.

Table 4. shows FID Scores and Geometric Scores computed for the different GAN implementations to better compare their performance. Lower score is better for both the indicators. As such, no quantitative results are available for comparison in the Urdu or Arabic language and hence Table 4 only shows results of our implementations for analysis. These are validated by observing the performance of a handwriting recognition system in the next section.

Table 5. Extending UCOM ligature dataset and evaluating the impact on handwriting recogniser performance

Data	Character Error Rate
UCOM only	7.12
UCOM + 15k	6.77

The lowest FID score was recorded with WGAN-GP architecture which was marked to be 15.74 at it's lowest. This score is comparable to state of the art scores available for other languages. DCGAN, with all its various settings, scored higher than both WGAN and its gradient penalty variant. This is also in accordance with the qualitative analysis where it was made evident that the other two architectures were better in picking up and thus producing more detailed samples as compared to DCGAN. Lowest possible FID score achieved by DCGAN was 21.45 while it was 17.97 with architecture of WGAN. GS scores were also in correlation with FID scores with WGAN-GP giving the lowest score, implying it to be the best model out of all three. Followed by WGAN and then DCGAN giving the highest score, GS scores proved true to the findings based on FID.

5.3 Data generation for handwritten text recognition

Main purpose for data generation using GANs is to increase performance of any model making use of respective dataset. For this purpose, an experiment is carried out to evaluate performance of a handwriting recognition system with and without generated samples. The recogniser consists of six convolutional layers, two Bidirectional Gated Recurrent Units (BiGRU) layers and a Connectionist Temporal Classification (CTC) output layer. Samples are created using WGAN-GP variant. An additional 50 samples are created for each ligature class resulting in approximately 15k synthesized samples. All samples of size 64x64 are scaled down and padded to have a size of 64x128 to be used as a training set for the recognition system. Table 5 evaluates recogniser performance in terms of Character Error Rate (CER). It is observed that recogniser performance is slightly increased by incorporating synthetic samples.

5.4 Discussion

Urdu alphabets, similar to the English alphabets, have varying widths and so does the resulting ligatures when these are joined together. Some ligatures, formed mostly from 6 or more alphabets, might result in cramped ligatures as a result of the initial pre-processing step. The proposed method fails to fully learn to distinguish between the small details in such ligatures and thus, squeezed variations of these ligatures are produced which do not clearly encapsulates all the details. An output width of 64 can thus be ruled as insufficient to cater to all characters required to be produced for ligatures with larger number of alphabets.

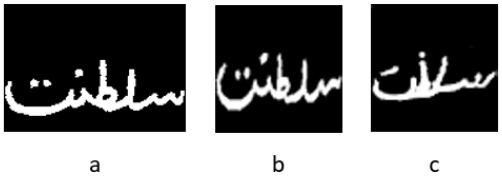

Fig. 6. Ligature "سلطنت". Subfigure (a) is the actual ligature sample without any resizing, subfigure (b) is resized sample after pre-processing steps, subfigure (c) is output of proposed model

Figure 6, subfigure (a) shows that the actual width of the ligature is almost twice that of the resized ligature in subfigure (b). Output from our architecture in subfigure (c) shows three alphabets jumbled together and are not easily distinguishable. There cases are rare for Urdu ligatures whereby ligatures largely are made of less than 6 alphabets and hence the proposed approach is sufficient for production of ligatures. For production of complete Urdu words however, generators with variable length outputs should be implemented.

Furthermore, the proposed model is capable of only producing 317 ligatures accurately for which it was trained. It fails to generate sequence of alphabets that are not present in the training set. Most of the Urdu alphabets change their shape depending on whether it is at the start, middle or end of a ligature and have a different shape when they are being used as standalone ligatures i.e. being used as a single character ligature. The generator is unable to learn these characteristics of the Urdu language fully. Giving character sequences for ligatures not available in the training set may result in production of arbitrary shapes that do not resemble any ligature from the Urdu language. Figure 7 shows the varying shapes the alphabet "گ" takes when used in different positions of a ligature.

In view of the above, we can conclude that the diversity in the Urdu alphabets differs from the English alphabets wherein there is some sort of a definite correlation between widths of the lowercase and uppercase forms of an alphabet. They are also considered to be different characters in the lexicon. Urdu alphabets however do not follow any such pattern, a single character has varying width and shape depending on its neighbouring characters that needs to be learned by the architecture. This also poses a challenge for implementing a generator with variable output length in case of Urdu language.

6 Conclusion

In this study, different architectures were implemented for generation of Urdu handwritten samples using a small dataset. Convolutional nature of architectures allows for cursive handwriting and synthesis of connected components which is a requirement for the generation of Urdu ligatures. A single generator is capable of

Fig. 7. Alphabet "گ". Subfigure (a) is single alphabet that is used as standalone ligature, subfigure (b)-(d) shows the various shape the alphabet acquires when used at the start, middle and end of ligature respectively.

producing multiple ligatures that has not been done before solely for Urdu language and evaluations were carried out that can be used later for future studies. Keeping in mind the complex nature of Urdu handwritten samples, the proposed approach, especially network architecture in conjunction with WGAN-GP objective function, is able to produce high quality images and give FID scores comparable to those present for state of the art in other languages. Lastly, the synthesized ligature samples were shown to improve performance of a handwriting recognition system validating the fact that even with a small dataset, performance for Urdu handwriting recognition can still be further improved with data generation.

The proposed adversarial model is a modified architecture used for the task of generation of Urdu handwriting samples which does not require any auxiliary networks and is not a massive network either as compared to others used for this task previously. Other GAN improvements and optimizations can be incorporated to the fully convolutional networks to further increase the quality of images. For generation of more diverse ligatures, a larger dataset, consisting of even more unique and complex ligatures, can be synthesized and used for training of GANs. The proposed model is sufficient for production of ligatures but variability in output sequence needs to be incorporated for production of Urdu words.

References

1. Amazon textract: Intelligently extract text and data with ocr, 2019.
2. Cloud vision api: Detect text in images, 2019.
3. Saad Bin Ahmed, Saeeda Naz, Salahuddin Swati, Imran Razzak, Arif Iqbal Umar, and Akbar Ali Khan. Ucom offline dataset-an urdu handwritten dataset generation. *International Arab Journal of Information Technology (IAJIT)*, 14(2), 2017.
4. Eloi Alonso, Bastien Moysset, and Ronaldo Messina. Adversarial generation of handwritten text images conditioned on sequences. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 481–486. IEEE, 2019.
5. Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

6. Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37–49. JMLR Workshop and Conference Proceedings, 2012.
7. Bo Chang, Qiong Zhang, Shenyi Pan, and Lili Meng. Generating handwritten chinese characters using cyclegan. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 199–207. IEEE, 2018.
8. Mehdi Dehghan, Karim Faez, Majid Ahmadi, and Malayappan Shridhar. Handwritten farsi (arabic) word recognition: a holistic approach using discrete hmm. *Pattern Recognition*, 34(5):1057–1065, 2001.
9. Faiq Faizan Farooqui, Muhammad Hassan, Muhammad Shahzad Younis, and Muhammad Kashif Siddhu. Offline hand written urdu word spotting using random data generation. *IEEE Access*, 8:131119–131136, 2020.
10. Sharon Fogel, Hadar Averbuch-Elor, Sarel Cohen, Shai Mazor, and Roee Litman. Scrabblegan: Semi-supervised varying length handwritten text generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4324–4333, 2020.
11. Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.
12. Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.
13. Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.
14. Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.
15. Valentin Khrulkov and Ivan Oseledets. Geometry Score: A Method For Comparing Generative Adversarial Networks. *arXiv preprint arXiv:1802.02664*, 2018.
16. Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.
17. Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.
18. Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.
19. Zhuoyao Zhong, Lianwen Jin, and Zecheng Xie. High performance offline handwritten chinese character recognition using googlenet and directional feature maps. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 846–850. IEEE, 2015.
20. Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.