Hindi Handwriting Generation Using Advanced Deep Learning Models

Abhishek Sinha, Aditi Dhavale, Amrut Ghadge, Dev Bhanushali, Harsh Kotadiya

***Abstract*—Handwriting generation for complex scripts like Hindi is difficult because of their intricate character formations, conjunct consonants, and diacritical marks. In this paper, we introduce a new way of generating Hindi handwriting by creating a Pix2Pix model with PatchGAN for image to image translation[1]. We synthesized Hindi words using CALAM to leverage a Hindi dictionary and combinations of character and maatra. The conditioned Pix2Pix model generates realistic handwritten images of Hindi text conditioned on textual input, evaluated by the PatchGAN discriminator on localized patches to achieve increased quality and consistency. We then used a Handwritten Text Recognition (HTR) model to evaluate the legibility of text generated by the GPT2 model; CAR and WAR were computed based on Edit Distance metric. We find that our model sufficiently captures the subtleties of Hindi script, achieving CAR = 23% and WAR = 0.56%. For script specific handwriting, this research involves script specific synthesis and is a first step for future work on high complexity scripts.**

***Index Terms*—Hindi handwriting generation, CALAM dataset, RNN, GAN, Sequence-to-sequence, HTR evaluation, Edit Distance.**

I. Introduction

Deep learning has become the field of handwriting generation and, more specifically, its development has been huge with languages such as English — having simpler scripts, though the languages with more complex scripts still remain quite challenging to generate handwriting for. This however will pose challenges if we were to extend these methods to languages that have complex scripts, such as, for instance Hindi. Because the Hindi script is complicated by a big character set, the difficult ligatures and multifarious writing styles, Hindi handwriting generation is notably difficult. Addressing challenges that call for such approaches that can not only form characters but also compensate for the spatial and contextual variances of the Devanagari script are these.

Though efforts on handwriting synthesis for Latin scripts have progressed, little research has been done on Hindi, and only little studied is devoted to creating realistic handwritten text. In particular, existing models, focusing mostly on English or similar scripts, fail to generalize when using them to the case of Hindi as it possesses elaborated structure and exhibits contextual dependencies. The natural variations and application usage of handwritten text requires specialized solutions to generate high quality handwritten text, which can also be used for further downstream applications such as handwriting recognition, and this script complexity makes it hard to learn.

Here, in this study an AI system architecture based for Hindi handwriting generation, designed on CALAM dataset is proposed. To guide authentic word formation, we formed words with every possible character-maatra combination in a synthetic dataset. a Pix2Pix image-to-image translation model with a PatchGAN discriminator[1] was built, where the generator produces high fidelity handwritten text image from text and the discriminator performs patch by patch evaluation for more detailed accuracy. We taped generated handwriting to assess the quality and used a Handwritten Text Recognition (HTR) model to measure Character Accuracy Rate (CAR) and Word Accuracy Rate (WAR) using Edit Distance. The findings indicate the effectiveness of the model in capturing the fine grain of Hindi script, and make an important contribution to future handwriting generation for complex scripts.

II. Related Work

A growing number of advances in handwriting generation have been made, notably for languages with simple scripts such as English. Despite that, very little effort has been put into applying these techniques to languages with complex scripts, like Hindi. The ledger of Hindi's Devanagari script consists of a large number of characters, sophisticated ligatures, diacritical marks and a variety of writing styles, that makes it difficult to generate natural handwriting. In this section we discuss how our approach has been impacted by relevant research, and the limitations of existing models for generating Hindi handwriting in this section.

In Kang et al. (2020) they introduced *GANwriting*, a generative model that synthesizes realistic handwritten text conditioned on both textual content and calligraphic style features. To draw on, the model has proven to

be a valuable contribution to natural handwriting generation in English because of its ability to adapt to a variety of writing styles. But a problem with Hindi's script complexity, its broad array of conjunct consonants and intricate diacritic placements, makes this approach untenable without considerable adaptation. [2].

In Zdenek and Nakayama (2020), *JokerGAN* was developed, a memory efficient GAN based model for Handwritten Text Recognition (HTR) dataset augmentation. The purpose of this model is that it can serve languages with a large character sets with enabling of character alignment while reducing memory usage. However, JokerGAN's design is efficient for languages with complex characters, but still rudimentary in dealing with the spatial and contextual dependencies specific to the Devanagari script of Hindi, containing multiplex character combinations and variations based on context. [3].

Based on this, Fogel et al. (2020) propose what they refer to as *ScrabbleGAN*, which only operates with a semisupervised approach to manipulate handwriting styles. We model this handwritten generation procedure, allowing us to generate diverse handwriting styles by manipulating stroke and cursive styles, thus generating varied handwriting in English independent of style. But this versatility is not suitable for Hindi, which has the script's complexity and conjuncts abound making specialized handling beyond style manipulation necessary. [4].

Graves (2013) also shown handwriting generation with Recurrent Neural Networks (RNNs) namely LSTMs. The main advantage of his model is its ability to generate conditioned handwriting looking similar to handwriting by humans and projecting long range dependencies in text. While Hindi has been successfully tackled with this method, in languages with simpler scripts, the specificities of Hindi's often complex character forms and the way its characters have no or varied spatial relationships to one another make this solution inadequate. [5].

In Senthil et al. (2020) they used Hindi handwritten word generation with a GAN based approach to solve the scarcity of Hindi handwriting data. With segmentation free recognition using Few Instance Learning, we generate word images by synthesizing character combinations in their model. Our choice of the CALAM dataset to populate the model with diverse word forms, diversifying synthetic Hindi words into this work has influenced our choice in the dataset. [6].

Although these existing models have come a long way in generating handwriting for simpler scripts, none address the particular challenges unique to the Hindi script. To address this limitation, our research presents a novel

Fig. 1: Research Methodology

method that takes advantage of a Pix2Pix[1] architecture along with a PatchGAN discriminator. Unlike traditional encoder decoder frameworks, the Pix2Pix[1] model enables context rendered handwritten text images produced from synthetic word data, with the use of a PatchGAN discriminator to focus on local image details to help render a better script. We make use of the CALAM dataset to generate a lot of Hindi words, by combining all possible single character and maatra setups. We also introduce a Handwritten Text Recognition (HTR) model used to evaluate the quality of the generated handwriting using Character Accuracy Rate (CAR), and Word Accuracy Rate (WAR) to show the model's capability of capturing the inherent Hindi script nuances. We address a critical gap in the literature by designing a tailored, script specific solution for Hindi handwriting generation which can be extended to complex scripts.

III. Proposed Methodology

In the following section, we detail how the handwritten Hindi text images are generated from text input, starting from generation of the dataset, word forming, the selection of architecture and finally model training.

A. Dataset Generation using the CALAM Dataset

As there is a shortage of diverse Hindi samples in handwriting, we took our Hindi handwritten text dataset as a base on which we were to work with using the CALAM dataset. The Hindi characters and the combinations of the characters along with *maatraas* constitute this dataset. Finally we put them together, together to form our Hindi handwritten text dataset. Following methodology proposed by Senthil [6], we generated words by concatenation of *maatraas* and characters based on Hindi grammar rules.

To fetch the characters from CALAM Dataset we applied Operation like Hough Line Transform. A powerful feature extraction technique in Image processing one which helps to detect the lines in an image even if they are faint or partially occluded is the Hough Transform. For a 2D image, a line is representative in slope intercept form, that is y = mx+c. However, this form becomes impractical for vertical lines (where m is undefined), so

$$\rho = x \cos\theta + y \sin\theta$$

Fig. 2: eq

we often use the polar coordinate form: And per this line, ρ is the perpendicular distance from the origin to the line and θ is the angle of this perpendicular to the line.

Our main reason for detecting lines using Hough Transform was for *shirorekha* detection. Detecting *shirorekha* will help us in concatenating the letters to create the words we need. After line detection, we use contraints and filtering methods to detect *shirorekha*. The methods used are as following:

1) Angle Thresholding: Lines having angle θ such that 80 ¡ $< \theta <$ 100.
2) Length Thresholding: Lines sorted by length and the longest continuous line is considered as the *shirorekha*.
3) Position Constraints: Lines lying only in top half of the image and not in first 30 pixels of image in case of upper *matra* are considered.

B. Word Formation with Hindi Dictionary

As a reference we had used a Hindi dictionary to make sure that generated Hindi words are valid. This process involved:

- Selecting characters from CALAM and generating valid Hindi words by logically applying *maatraas* and grammatical rules.
- Filtering nonsensical or invalid word combinations to ensure a high-quality dataset of Hindi words suitable for handwriting generation.

Training data in the form of generated words was used with a diverse set of real Hindi words which made the model learn from the generated words.

C. Handwriting Generation Model: Pix2Pix Architecture

To produce the handwritten text images we used the Pix2Pix[1] architecture, a deep learning modeling to perform image to image translation by utilizing a conditional generative adversarial network (cGAN). The Pix2Pix consists of a generator network, G, and a discriminator network D. The generator learns to transform input text to handwriting images, and the discriminator judges if provided handwriting images are real or tampered. Now these two Generator and Discriminator compete against each other and both learn iteratively by

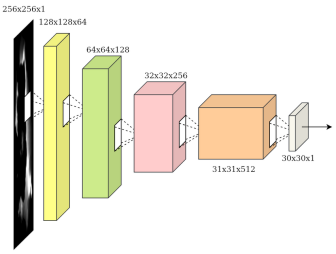

Fig. 4: Architecture of Discriminator, i.e. PatchGAN

learning from each others advice. The objective function for Pix2Pix is defined as:

$$\mathcal{L}_{\text{Pix2Pix}}(G, D) = \mathrm{E}_{x,y}\left[\log D(x, y)\right] \quad (1)$$
$$+ \mathrm{E}_{x}\left[\log(1 - D(x, G(x)))\right] \quad (2)$$

In the following case, where x is input text and y is real handwriting image. For that, the PatchGAN discriminator pays attention to local patches of the image (in the fashion of using ConvLSTMs), and though the discriminator 'sees' the font very similar to the style of writing they should produce, the model captures the fine details of Hindi script and generates high quality handwritten text images.

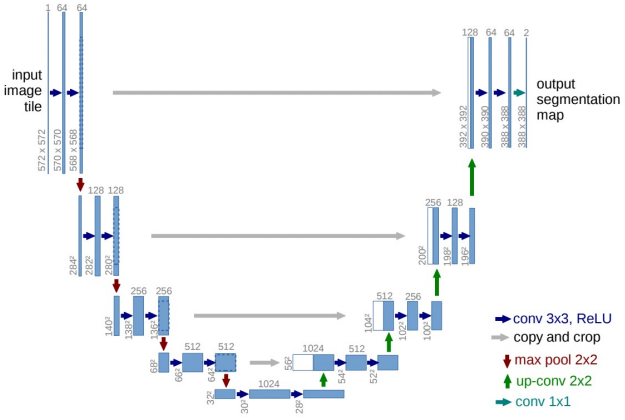

Fig. 3: Architecture of Generator(Unet) for Hindi handwriting generation using Pix2Pix with PatchGAN.

D. Evaluation using Handwritten Text Recognition (HTR) Models

We also used one HTR (Handwritten Text Recognition) model from [6] based on the CRNN architecture proposed by Senthil. The features are extracted by convoluted layers, and sequential dependencies in the handwriting image are captured by bidirectional LSTMs, and the model is trained with CTC (Connectionist Temporal Classification) loss for the text alignment.

$$P(Y|X) = \sum_{A \in \text{Align}(X,Y)} P(A|X)$$

Fig. 5: CTC Formula

$$\mathcal{L}_{\text{CTC}} = -\log P(Y|X)$$

Fig. 6: CTC Loss

CTC seeks to obtain maximum probability of correct transcription by summing over all possible alignments leading to ground true sequence. Below is the form of the problem, where X is considered the input sequence (image feature sequence), Y is the ground truth sequence (text). We give a language of X that the CTC computes the probability of Y — X based on summing over all valid alignments and let A be the set of all possible alignment whose length is X. For each possible alignment, CTC uses a forward-backward algorithm to compute the probability efficiently.

Character Accuracy Rate (CAR) and Word Accuracy Rate (WAR), using Edit Distance, were the evaluation metrics for computed discrepancies between the handwritten text generated by the model and actual handwritten text, and offered explanation of the performance of the model in producing accurate and natural handwriting. Put simply, Edit distance is the distance between the recognized text and the ground truth text. The minimum number of character level operations (insertions, deletions, substitutions) for matching the recognized sequence to the ground truth sequence is measured. Edit Distance is mathematically calculated dynamically by the use of Dynamic Programming. We first use HTR to predict the recognized sequence and compare it to the ground truth sequence to calculate WAR & CAR for HTR. To elaborate:

1) Once trained on the model, we pass the corresponding test image into the model to get the predicted text sequence.
2) We then compute edit distance (character level for CAR and word level for WAR) between each ground truth and predicted pair.
3) Finally, for the above derived values, we find the CAR and WAR through the same above mentioned formulae.

Over all, CTC loss is used by HTR models to align and predict sequences of text displayed in images, then evaluated for CAR and WAR by comparing the predictions to ground truth using edit distance.

$$\text{CAR} = 1 - \frac{\text{ED}(\text{GT}, \text{REC})}{\text{LEN}_{\text{GT}}}$$

Fig. 7: CAR formula

$$\text{WAR} = 1 - \frac{W_{\text{ED}}}{W_{\text{GT}}}$$

Fig. 8: WAR formula

IV. Results and Discussion

A. Character Accuracy Rate (CAR)

The model had a Character Accuracy Rate (CAR) of 23%. This accuracy at the character level is a true metric of the accurate character generation of Pix2Pix architecture is reproduced to reproduce Hindi characters. Below formula simply stands for: ED(Edit Distance), GT – ground truth text, REC – Recognized text, LENgt – length of ground truth text. The range of value of CAR is from 0 to 1 since 1 denotes perfect match without errors. The CAR demonstrates that the model is capable of learning basic character shapes but further character segmentation and placement improvements could be achieved.

B. Word Accuracy Rate (WAR)

At the word level more than character level, WAR cares. As with CAR, WAR is on a 0 to 1 scale, with 1 indicating no word errors. The below formula refers to the number of word level edits (insertions, deletions, substitutions) in W(ed). It was found that the Word Accuracy Rate (WAR) was 0.56%, which is the percentage of correctly generated words. This relatively low WAR then also highlights the difficulty in applying fluent and consistent word level synthesis to a complex script like Hindi. Future work might focus on increasing spatial arrangement and modeling of character dependencies in order to improve overall word coherence.

C. Discussion

It is found that Pix2Pix architecture[1] works satisfactorily to generate simple Hindi words, however, shows limited success in generating more complex Hindi words with conjunct characters and complex diacritical marks. These limitations might be addressed by expanding the dataset and vocabulary. Moreover, the model can also be fine tuned with state of the art technique like transformer based attention mechanisms to enable the model to more refined the unique characteristic of Hindi script.

D. Sample Input and Output Images

Figures 9 and 10 showcase examples of input text images and their corresponding generated handwritten output images. These examples illustrate the model's capability to convert printed Hindi text into realistic handwritten script, handling challenges like character formation and maatra positioning.

1) Sample Input Text Image: The input text image shown in Fig. 9 represents a A synthetically generated Hindi word (9) is represented by this using the CALAM dataset. The input to the Pix2Pix based model is this image with each characters and maatra carefully chosen according to Hindi grammar rules.

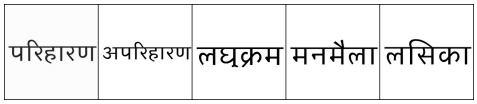

Fig. 9: Sample Input Text Image: A synthetically generated Hindi word used as input to the Pix2Pix model.

2) Generated Handwritten Output Image: Figure 10 The following handwritten output generated by the model is shown in sample output. The output of the model shows the attempt it makes to capture the subtleties of the handwritten Hindi script, such as, complex ligatures and diacritical marks specific to Devanagari. However, the model achieves only moderate character accuracy while maintaining word level coherence is challenging.

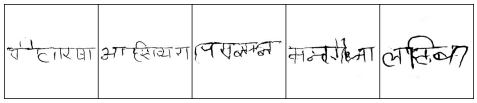

Fig. 10: Generated Handwritten Output Image: The handwritten image generated by the model for the given input text, showcasing character formation and maatra placement.

The input and output examples in Figures 9 and 10, figures of our proposed model solving the problem of transforming printed Hindi text into a plausible handwritten style are given. These images showcase the properties of Hindi script on which the model can operate and handle such challenging problems in Hindi script that include character formation, ligatures, and maatra positioning.

V. Conclusion

In this paper, we present a novel approach of generating Hindi handwritten text using deep learning models and specifically employing a Pix2Pix architecture which was trained on a dataset developed based on the CALAM dataset. Combining Pix2Pix architecture with HTR based evaluation, our method achieves a Character Accuracy Rate of 23% and a Word Accuracy Rate of 0.56%. The results highlight the difficulties in generating Hindi handwriting, in particular the challenges of arranging the character and the contextual accuracy. The result may not be spectacular but this does tell how much research is needed in the complex domain of Hindi Hand-written text generation. Future work might investigate other architectures like transformers, or better GANs, for further improvement in fluency and legibility of generated handwriting. Overall, our result adds to the demand for synthetic handwriting of complex scripts and identifies empirical gaps that are worth to investigate.

REFERENCES

[1] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1125–1134. [Online]. Available: https://doi.org/10.1109/CVPR.2017.632

[2] L. Kang, P. Riba, Y. Wang, M. Rusiñol, A. Fornés, and M. Villegas, "Ganwriting: Content-conditioned generation of styled handwritten word images," *arXiv preprint arXiv:2003.02567*, 2020, accepted to ECCV2020. [Online]. Available: https://doi.org/10.48550/arXiv.2003.02567

[3] J. Zdenek and H. Nakayama, "Jokergan: Memory-efficient model for handwritten text generation with text line awareness," in *Proceedings of the Conference on Pattern Recognition and Computer Vision*. Tokyo, Japan: The University of Tokyo, 2020. [Online]. Available: https://janzdenek.github.io/publication/jokergan/jokergan.pdf

[4] S. Fogel, H. Averbuch-Elor, S. Cohen, S. Mazor, and R. Litman, "Scrabblegan: Semi-supervised varying length handwritten text generation," *arXiv preprint arXiv:2003.10557*, 2020. [Online]. Available: https://arxiv.org/pdf/2003.10557

[5] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013. [Online]. Available: https://arxiv.org/pdf/1308.0850

[6] G. Senthil, K. Nandhakumar, and G. R. K. S. Subrahmanyam, "Handwritten hindi word generation to enable few instance learning of hindi documents," in *2020 International Conference on Signal Processing and Communications (SPCOM)*, 2020, pp. 1–5.