Summer Research Fellowship Programme
(SRFP-2023)

Generating Synthetic Word Images for training Indian Scripts Scene Text OCR

Raunak Nag, KCIS/23/137
Jaypee University of Engineering and Technology (JUET)

Guide: Prof CV Jawahar
International Institute of Information Technology-Hyderabad (IIIT-H)

Acknowledgements

I would like to express my sincere gratitude to my guide, Prof. C.V. Jawahar and supervisor Dr. Ajoy Mondal at IIIT Hyderabad for their wonderful guidance, constant help and constant support at each and every step of the project during my entire internship journey. Also, I am very thankful to all of my friends and colleagues whom I have interacted in the college in this duration of 2 months.

I am extremely grateful to my family for their unlimited support and encouragement throughout the entire duration of the internship. I also thank IAS-NASI-INSA for giving me this “once in a lifetime opportunity” which helped me to know and learn about scientific research and study.

I would also like to thank IIIT Hyderabad for providing me with hostel and other living requirements and making my stay a comfortable one.

The fact that this small journey is nearing its end is sad for me as I would have liked to gain more experience and knowledge. I am really grateful to have been able to gain the experience of spending my summer under such skilled and knowledgeable people at one of the best institutes of India.

TABLE OF CONTENTS

Abstract

Text recognition has always been an active field in computer vision. Optical Character Recognition (OCR) is the technology that enables the conversion of printed or handwritten text into machine-readable characters whereas natural scene images with complex backgrounds fall into the category of Scene Text Recognition. Scene text recognition has become an interesting field of research due to the complexities and difficulties such as complex backgrounds, improper illumination, distorted images with various noises, inconsistent fonts usage, font style and thickness, background as well as foreground color and texture, illumination and image resolution, therefore making the task of scene text recognition more complicated and challenging.

Latin languages were found to be the center of attention till now but the field of scene text recognition for non-Latin languages was still yet to be properly investigated. Scene text recognition in low-resource non- Latin languages is difficult and challenging due to the inherent complex scripts, multiple writing systems, various fonts and orientations. Scene text recognition in low- resource non-Latin languages is difficult and challenging due to the inherent complex scripts, multiple writing systems, various fonts and orientations. The main difference between real data and synthetic data is that synthetic data can be generated as much as possible using different computer algorithms and we can apply different types of variations in the data to make it closely resemble with real data. This data can then be used to train the convolutional neural networks involved in OCR systems. [2]

Introduction

Reading Text from Natural Scene Images

Scene text recognition (STR) refers to the task of reading text from real-world scene images like street signs and billboards. Scene text recognition systems are often used in building and developing technologies involved in autonomous driving systems and traffic monitoring systems for video surveillance like to detect license plates of vehicles or real-time traffic sign detection. [7]

Natural scene images are not always centered around capturing text in a horizontal fashion instead, they are captured randomly, which ultimately gives rise to irregular scene images. We can categorize text in images based on the form of text, the language of the text, etc. Even though there are many similarities between STR and OCR systems, the task of STR is demanding and challenging as compared to the latter. This is because scanned and printed documents are the significant sources for OCR systems which have clear backgrounds with less noise, and is also often organized and consistent in terms of fonts or handwriting. But the data used in STR systems are usually filled with noise and a very complicated background along with inconsistent text with varying font styles and sizes. This resulted in the OCR systems failing miserably on STR data. These were some of the challenges and difficulties that attracted and interested many researchers from around the world towards the field of scene text recognition. [6]

Current Scene Text Recognition systems rely on datasets that mostly contain text in Latin/English. But there has also been a prominence of non-Latin languages like Chinese, Korean, Japanese, Arabic, Hindi, Bengali and other Indian languages, etc. over the years. Hence, it is important to construct recognition models for these non- Latin languages as well. However, to build such models, huge amount of language specific real scene text data is needed for training and testing purposes which unfortunately is scarce, time-consuming and the annotation process is often prone to human error and only available abundantly for the English/Latin languages. One solution to the data scarcity issue is to generate high-quality realistic synthetic data for the recent scene-text recognition models which is cheap and scalable.

Figure 1: Sample images of English Scene Text found at different places

Text detection in natural scene images is much more difficult than text detection in scanned document images because of the various forms of the text. The main features of the scene text are summarized below:

1. Multiple languages may be mixed.

2. Characters may occur in different sizes, fonts, colors, brightness, contrast, etc.

3. Text lines may be horizontal, vertical, curved, rotated, twisted, or in other patterns.

4. The text area in the image may also be distorted (perspective, affine transformation), suffer defects, blurring, or other phenomena.

5. The background of scene images is extremely diverse, and text may appear on a plane, surface, or folded surface.

Scene Text Recognition for non-Latin languages

With due course of time, there has been a prominent rise of language (especially non-Latin languages) other than English over the years. Hence, it has become essential to construct such recognition models for non-Latin languages.

Non-Latin languages include Chinese, Korean, Japanese, Arabic, Hindi, Bangla and other Indian languages, etc. These languages have different writing systems and the complexity of scripts among the languages varies a lot. It has been seen that the performance of recognition algorithms over English datasets is far better when compared to non-Latin languages. It is mostly attributed to the lack of large-scale data for training and testing purposes and the lack of recognition models structured around capturing such texts in these languages. [5]

STR challenges for Indian languages

Out of these 22 main Indian languages, six of them are part of the top twenty most spoken languages in the world. Despite being widely used languages, there have not been many works based on Indian languages. [1]

The lack of large-scale data for training is one of the major hassles behind the low performances of Indian languages. Indian languages, low-resource languages, have unstructured data and the data consists of images in diverse conditions such as scripts, fonts, sizes, and orientations.

Figure 2: Challenges in scene text recognition. A few sample images from the SVT [164] and IIIT 5K-word [107] datasets are shown to highlight the variation in view point, orientation, non- uniform background, non-standard font styles and issues such as occlusion, noise, and inconsistent lighting.

Indian Scene Text Recognition

Despite the recent advancements in English STR, developments in non-Latin languages have been comparatively slow. It is the reason why we take up this task of designing and creating STR systems for Indian languages. On average, most of the present models for non- Latin languages do not perform nearly as efficiently as English STR models. The low performance can be attributed to the complex nature of the glyphs of the non-Latin languages. [4]

Non-Latin languages differ in various ways, including different writing systems (for example, Arabic (Right-to- Left), horizontal and vertical writing systems for many East Asian scripts). The lack of abundant data for low-resource languages is one of the most crucial impediments to the development of STR systems. [10]

Figure 3: Examples of synthetic word images in Gujarati, Hindi, Bangla, Tamil, Telugu, Malayalam, Gurmukhi, Kannada and Odia

Parameters involved in STR

There are many parameters involved in designing scene text recognition systems. Firstly, we identify the parameters and focus on each and every one to determine the crucial parameters. Synthetic data is a crucial part of STR, and it involves the number of training samples, synthetic data generation engine, number of fonts and most frequently used vocabulary of the language. [2]

Most recognition algorithms proposed for the English language are evaluated on seven major public real scene text datasets. These datasets contain a variety of test images with different imaging conditions. For example, they contain images with no distortions that form the easy cases for STR, while images with low resolution or curved text form the harder and corner cases for STR. Training using synthetic data proves to be an important tool in testing and

measuring the eminence of parameters involved in the process. Factors such as the number of fonts and language vocabulary improve the data diversity while training. We can also increase the amount of data by applying augmentation while training. Augmentation alters the data and hence induces diversity into the training data. We use various augmentation methods to imbibe variety and improve performance. In addition to this, we also identify the number of fonts to be crucial in raising the recognition rates of recognition models.

Synthetic Datasets for Scene Text Recognition

Collecting real-world data is time-consuming and is also challenging to garner millions of images for a particular task, and the annotation process is often prone to human error despite the efforts. These limitations resulted in the emergence of an alternative source of data, synthetic data, to mimic the real-world data. [3]

Synthetic data is computer-generated with minimal effort and is cheap and scalable. It has been widely used for numerous applications like text detection models while training instead of manually annotated data. Scene text synthetic datasets have shown potential in improving the performance of both text detection and recognition tasks. We focus on generating synthetic scene text data for one of the ten Indian languages i.e., Bengali and elaborate on the details about the characteristics of the generated data.

Despite the advancements in synthetic datasets for English language, there has not been much focus on non-Latin languages because of different scripts, the complexity of such language scripts, scarcity of data, the availability of fonts, various writing systems, etc.

We generated synthetic data for one of the most popular Indian languages, Bengali. We chose this language based on its popularity and worldwide speakers.

Methodology

Pipeline for Generating Synthetic Data

Synthetic data is crucial for scene text recognition systems. We require large amounts of synthetic data for non-Latin languages similar to other Latin languages. The effect of synthetic data generation engines is in the order of 3 − 5% on the real scene text datasets, and this method is the least expensive method of all in terms of computation and is also cheap and scalable especially to create vast amount of training images.[4]

In the generation process, we can manipulate nuisance factors such as font, lighting, shadow, border, background, image noise, geometric deformation, and compression artifacts. As a result, image features trained on synthetic data with these factors will be robust to their variations, leading to a significant improvement of recognition accuracy.

Our whole project work can be mainly divided into seven phases-

1) Data Preparation- The project uses a collection of fonts compiled for Bengali language. Additionally, a set of images, preferably natural scene images, is needed to serve as background for the rendered word images. Our project uses the Places365 dataset for this purpose. It involves a python script that requires imagemagick, pango, cairo, pangocairo libraries to be installed on the linux machine and invokes bash commands from the command line.
2) Reading a specific language vocabulary file- In our case, we have chosen Bengali as our desired language. The project starts by reading a vocabulary file containing a list of words that need to be rendered as synthetic images. The script then reads each word from the file and then proceeds to render the word image.
3) Choosing suitable rendering parameters- Here, for each word, the script randomly selects various rendering parameters like font name, font stretch, font color, font style, font size, foreground (text) color and background color along with perspective distortion.
4) Rendering the text word as an image- Using the selected rendering parameters, the script invokes the Pango library to render the word as an image by choosing a random font from a list of installed fonts each time the program runs. The rendered image is then saved as a JPG file.

5) Background Blending- The script randomly selects the background should be a natural scene image blended with the text layer or a uniform color. We have chosen the former case where the script takes a random crop from the Places dataset and blends it with the text layer and kept the latter one optional.
6) Image Post-Processing- The script applies optional image post-processing, including erosion, dilation, gaussian blur, median blur, salt and pepper noise and gaussian noise. These post- processing operations add variations to the synthetic images, making them more diverse and suitable for training OCR models.
7) Saving the Final Image- The final synthetic word image, along with its rendering parameters and other metadata, is saved in a specified output directory and all the values of the rendering parameters are written to a detailed annotation csv file for Bengali language.

Figure 4: Block diagram stating the overview of the project

Let's look at the above phases in more detail. Firstly, we collect some Bengali vocabulary files containing the most frequent Bengali words. Then, we install the specific language (Bengali) fonts on the systems in which we want to render the vocabulary file. For the background images, we use have used the validation set from the Places365 dataset. The code uses pango-cairo engine for rendering text with the fonts available on scene text images as background images. As mentioned previously, we vary the parameters related to fonts (font size, font style, and other properties like font thickness/weight, font-stretch), strokes (width and color) and shadow effect (width, opacity). We also incorporate distortions with randomized parameters using skew, affine and perspective projective transformations. Text is then rendered on the foreground text layer, and sometimes, we either apply a gradient or draw the tile of a foreground image over the label text instead of a solid color to create an effect over the text. It is later blended with a cropped part of an image from the Places365 dataset chosen at random as the background image. With millions of synthetic data, many of the conventional algorithms have achieved much success in the field of text detection and recognition. In this section, we try to simulate

same success for Indian languages as well by generating and training millions of synthetic data. The final data can then be processed down. Our work involves using a rendering script along with a collection of Unicode fonts to generate synthetic word images for training Indian Script i.e., Bengali language. In this project, the focus is on Indian scripts, where rendering text correctly with commonly used libraries has been problematic due to the incorrect ordering of certain Unicode glyphs. To address this issue, our work involves using the Pango library with ImageMagick to render Unicode text correctly. The rendering script provided in the repository facilitates the generation of synthetic word images with support for various Unicode fonts for Indian scripts.

Result

Inputs:-

1. Indian language vocabulary word file
2. List of unique Bengali fonts for Bengali language - 86
3. Directory path for the rendered images
4. CSV annotation file with details about rendering parameters
5. Iteration Number

Outputs:-

1. Bengali Annotation CSV file with rendering font parameters:- ImageFilePath, TextWord, Index, FontName, FontSize, FontStretch, ArcValue, PerspectiveBoolean, FGBlendBoolean, BGNaturalImage, Density, Erosion, Dilation, Gaussian Blur, Median Blur, Salt Pepper Noise, Gaussian Noise
2. Rendered Bengali Word Images

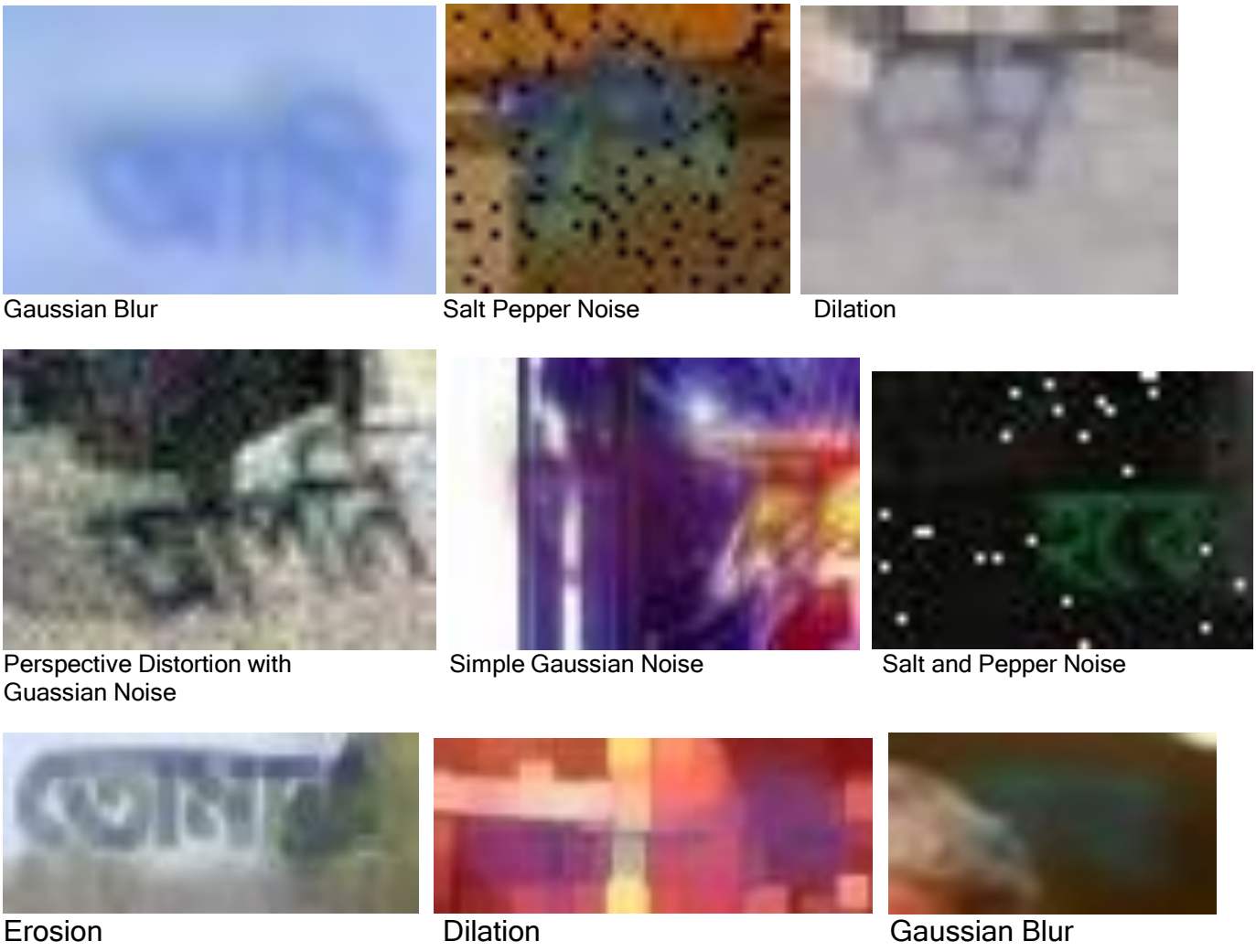

Figure 4: Sample examples of various augmentation techniques applied on rendered Bengali Word images from the synthetically generated dataset

Conclusion

Overall, we have explored and discussed in detail the potential of different parameters and their degree of impact on scene text recognition systems. We have focused more on the elements of synthetic data than on the different model architectures. Through these experiments, we realize the significance of different font parameters for boosting the performance of STR models. They improve the diversity of the training data and enhance the robustness of the systems.

Synthetic data boosts the performance of STR systems, and it is essential to extend the work in STR for low-resource languages. Application of synthetic data in training is the first step towards building robust STR systems for Indian languages. In general, the characteristics (like word length, complexity of the scripts) of all the languages vary greatly and it is essential to analyze such properties to properly understand and experiment to achieve the best results. In our work, we focused our attention on bridging the prevailing gap between the performances of English STR systems and Indic STR systems. We considered the complexity of Indian languages due to the presence of complex glyphs in the scripts, variation of writing systems, lack of fonts and most importantly, real scene text data.[8]

We chose Bengali to be our main language for our project work. Additionally, we investigated the importance of the number of fonts and designed experiments based on different font parameters to find a suitable range of values for each of them. We learned that the number of fonts play a crucial role in enhancing the diversity of data and hence improving the overall performance. We collected fonts and incorporated them to augment the quality of synthetic data for training experiments. Augmentation, on the other hand, further enhanced the overall performance. With the combination of both, we were able to achieve synthetically generated data that is comparable to real data.

Future Scope

Here, we only focus on one of the most popular Indian languages, i.e., Bengali throughout our work. But our work can be extended to other languages as well like Gujarati, Kannada, Oriya, Malayalam, Arabic etc. We showcase the potential of data diversity in STR systems along with the significance of different font parameters for better performance. We strongly presume that if we could improve the data diversity of other languages in a similar way, we would be able to achieve even better results for other languages as well. We also believe that there is a lot of scope left for further enhancing the performance of the current OCR systems through experimentation and testing. [9]

We list down some of the possible takes on future work as follows-

i) Transfer learning is a possible way for enhancing improved system performance and this can be extended from simple and easier languages like the ones mentioned in our work to infamous languages like Maithili, Assamese, Konkani, Santali etc.

ii) curating fonts for languages using font generators. To elaborate, most of these mentioned infamous languages have a scarcity of data that is even lesser than the present languages that we are dealing with right now. Finally, we need to work towards building font generators to exploit the present fonts in English to create fonts for non-Latin languages as well. We can do that by training Generative Adversarial Networks (GANs) to capture the correlations among the glyphs from fonts of source and target to design new fonts.

References

[1] Improving Scene Text Recognition for Indian Languages with Transfer Learning and Font Diversity, Gunna, Sanjana and Saluja, Rohit and Jawahar, C V, Journal of Imaging, vol 8(4), pp. 86, 2022

[2] Generating synthetic data for text recognition, Krishnan, Praveen and Jawahar, CV, Journal of arXiv, 2016

[3] Understanding Text in Scene Images, Mishra, Anand, Journal of International Institute of Information Technology Hyderabad, 2016

[4] Indian script character recognition: a survey, Pal, Umapada and Chaudhuri, BB, Journal of pattern recognition, vol (37), no (9) pp (1887—1899), 2004

[5] https://dropbox.tech/machine-learning/creating-a-modern-ocr- pipeline-using-computer-vision-and-deep-learning

[6] A synthetic recipe for OCR, Etter, David and Rawls, Stephen and Carpenter, Cameron and Sell, Gregory, 2019 International Conference on Document Analysis and Recognition (ICDAR)}, pp (864—869), 2019

[7] https://en.wikipedia.org/wiki/Scene_text

[8] Ocr on-the-go: Robust end-to-end systems for reading license plates & street signs, Saluja, Rohit and Maheshwari, Ayush and Ramakrishnan, Ganesh and Chaudhuri, Parag and Carman, Mark, 2019, International Conference on Document Analysis and Recognition (ICDAR), pp (154—159), 2019

[9] Towards scene text recognition with fewer labels, Baek, Jeonghun and Matsui, Yusuke and Aizawa, Kiyoharu, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pgs (3113—3122), 2021

[10] Scene text detection and recognition: The deep learning era, Long, Shangbang and He, Xin and Yao, Cong, International Journal of Computer Vision, vol (129), pp (161—184), 2021