Editing Bengali Scene Text With Style Retention

A thesis
Submitted in partial fulfillment of the requirements for the Degree of
Bachelor of Science in Computer Science and Engineering

Submitted by

Dip Chowdhury	**170104003**
Fuad Hasan	**170104010**
MD. Abu Yousuf Sajal	**170104025**
Syed Rifat Manjur	**170104026**

Supervised by

Mohammad Imrul Jubair

Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

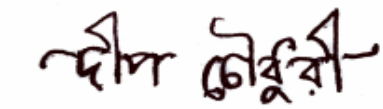
Dhaka, Bangladesh

January 2022

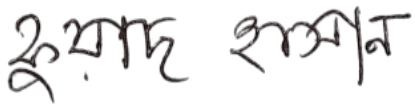
CANDIDATES' DECLARATION

We, hereby, declare that the thesispresented in this report is the outcome of the investigation performed by us under the supervision of Mohammad Imrul Jubair, Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, Dhaka, Bangladesh. The work was spread over two final year courses, CSE4100: Project and Thesis I and CSE4250: Project and Thesis II, in accordance with the course curriculum of the Department for the Bachelor of Science in Computer Science and Engineering program.

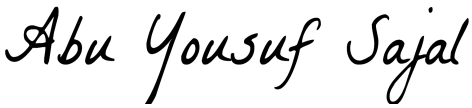
It is also declared that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.



Dip CHowdhury
170104003



Fuad Hasan
170104010



MD. Abu Yousuf Sajal
170104025

Syed Rifat Manjur
170104026

CERTIFICATION

This thesis titled, **"Editing Bengali Scene Text With Style Retention"**, submitted by the group as mentioned below has been accepted as satisfactory in partial fulfillment of the requirements for the degree B.Sc. in Computer Science and Engineering in January 2022.

Group Members:

Dip Chowdhury	**170104003**
Fuad Hasan	**170104010**
MD. Abu Yousuf Sajal	**170104025**
Syed Rifat Manjur	**170104026**

Mohammad Imrul Jubair
Assistant Professor & Supervisor
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

Prof. Dr. Mohammad Shafiul Alam
Professor & Head
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

ACKNOWLEDGEMENT

First and foremost, We have to thank our research supervisor, Mr. Mohammad Imrul Jubair. Without his assistance and dedicated involvement in every step throughout the process, this paper would have never been accomplished.

We take this opportunity to express gratitude to all of the Department faculty members of Ahsanullah University of Science and Technology for their guidance and support throughout this journey.

We would also like to thank our parents for their unceasing encouragement, support, and attention. Without that support, we couldn't have succeeded in this journey.

Last but not least, we would like to thank everyone who helped and motivated us to work on this project.

Dhaka
January 2022

Dip Chowdhury

Fuad Hasan

MD. Abu Yousuf Sajal

Syed Rifat Manjur

ABSTRACT

We intend to edit Bengali scene text images. The text in a scene image is very hard to edit without losing its natural look, background, color, and style. This is because the transformation has to be done in a way that both the background and the text remain the same as before. It becomes even more complicated when the text is in Bengali since the Bengali script features typographic ligature called the consonant conjunct. Besides, the graphemes of the Bengali script are very complex compared to the English alphabet. So, we intend to address this challenge by splitting the task into three independent sections. One of the modules is used for altering the text. This module modifies the input text according to the wording of the style image. It not only alters the text but also copies the style of the text from the style image. The second module extracts the text from the style image. After that, the module paints the extracted region with the help of the other elements of the current background. Another tough issue is integrating the two outputs of the two modules. This combination is done by the third module. This module combines the output of the other two modules in such a way that the image looks natural and lifelike. To the best of our knowledge, no work has been done in editing Bengali text in scene images; so we focus on this area in this study. For this objective, we generated a dataset that comprises of roughly 48,000 Bengali words synthesized with more than 11 $\times 10^3$ random background images and 110 Bengali font styles. We used the style retention network as our model and trained it using the dataset. Finally, we explored by evaluating realistic scene text graphics and received satisfactory results. This work will be advantageous in numerous ways, including as hiding information, erasing sensitive or offending language, correcting errors, and so on.

Contents

CANDIDATES' DECLARATION i

CERTIFICATION ii

ACKNOWLEDGEMENT iii

ABSTRACT iv

List of Figures vii

List of Tables x

1 Introduction 1

1.1 Problem Domain 1

1.2 Motivation 1

1.3 Our Contribution 2

2 Literature Review 4

2.1 Background Study 4

2.1.1 Convolutional Neural Network 4

2.1.2 Generative Models 5

2.1.3 Neural Style Transfer 8

2.2 Related Works 8

2.2.1 SwapText: Image Based Texts Transfer in Scenes 8

2.2.2 TextStyleBrush: Transfer of text aesthetics from a single example . 10

2.2.3 Synthetic Data for Text Localisation in Natural Images 10

2.2.4 Editing Text in the Wild 12

3 Our Contribution 14

3.1 Dataset 14

3.1.1 Artificial Data Generation 14

3.1.2 Rendering complex Unicode scripts 14

3.1.3 Artificial Dataset 15

3.1.4 Real-world Dataset 20

3.1.5 Other use cases of our Dataset 21
3.2 Application Methodology 22

4 Experiments and Results **24**
4.1 Experimental Setup 24
4.2 Results 24
4.2.1 Results on synthetic Dataset 24
4.2.2 Results on real-life scene images 28
4.3 Failure Cases 40
4.4 Evaluation 43
4.4.1 MSE 43
4.4.2 PSNR 43
4.4.3 SSIM 44
4.4.4 Quantitative results on synthetic test dataset 44

5 Conclusion **45**
5.1 Summary 45
5.1.1 Limitations 45
5.2 Future Works 46

References **47**

List of Figures

1.1 Style image 2
1.2 Input image 2
1.3 Output image 2
1.4 An example from **"Editing Text in the wild"** where an input text fig. 1.2 copies the font style and background of a source image fig. 1.1. The output is in fig. 1.3. 2

2.1 Example of the GAN Generator Model 6
2.2 Example of the GAN Discriminator Model 7
2.3 An example of "**Neural Style Transfer**". 8
2.4 Model of "**SwapText: Image Based Texts Transfer in Scenes**" [1]. 10
2.5 Model of "**TextStyleBrush: Transfer of text aesthetics from a single example**" [2] 11
2.6 Model of "**Editing Text in the Wild**" [3]. 13

3.1 Font rendering for complex script 15
3.2 Examples of generated training data. From top to bottom: I_s, I_t, t_{sk}, $mask_t$, t_t, t_b, t_f. See section 3.1.3 for details 16
3.3 Frequency of vowels in the dataset image texts 17
3.4 Frequency of consonants in the dataset image texts 17
3.5 Frequency of vowels in the dataset image texts after normalization 18
3.6 Frequency of consonants in the dataset image texts after normalization 19
3.7 Frequency of neumeric characters in the dataset image texts 19
3.8 Real-life images from fieldwork 20
3.9 Real-life images from Google Maps Street View 20
3.10 Dataset structure for text recognition task 21
3.11 The architecture of SRNet [3] 23

4.1 Example 1: Results on synthetic dataset 25
4.2 Example 2: Results on synthetic dataset 26
4.3 Example 1: Results on real-life scene images 28
4.4 Example 2: Results on real-life scene images 29
4.5 Example 3: Result of editing text book (black and white) 30

4.6 Example 4: Result of editing text book (black and white) 31
4.7 Example 5: Result of editing text book (grayscale) 32
4.8 Example 6: Result of editing text book (grayscale) 33
4.9 Input Style Image 34
4.10 Output Image 34
4.11 Example 7: Result of editing scene image 34
4.12 Input Style Image 34
4.13 Output Image 34
4.14 Example 8: Result of editing scene image 34
4.15 Input Style Image 35
4.16 Output Image 35
4.17 Example 9: Result of editing scene image 35
4.18 Input Style Image 35
4.19 Output Image 35
4.20 Example 10: Result of editing scene image 35
4.21 Input Style Image 36
4.22 Output Image 36
4.23 Example 11: Result of editing scene image 36
4.24 Input Style Image 36
4.25 Output Image 36
4.26 Example 12: Result of editing scene image 36
4.27 Input Style Image 37
4.28 Output Image 37
4.29 Example 13: Result of editing scene image 37
4.30 Input Style Image 37
4.31 Output Image 37
4.32 Example 14: Result of editing scene image 37
4.33 Input Style Image 38
4.34 Output Image 38
4.35 Example 15: Result of editing scene image 38
4.36 Example 16: Results of editing on handwritten image data. (Note: We did not train our model with handwritten data) 39
4.37 Example 17: Results of editing on handwritten image data (Note: We did not train our model with handwritten data) 40
4.38 The failure cases 41
4.39 Input Style Image 42
4.40 Output Image 42
4.41 Example 1: Failure case 42
4.42 Input Style Image 42

4.43 Output Image 42
4.44 Example 2: Failure Case 42

List of Tables

3.1 Supplementary materials 20
3.2 Distribution of Data in Dataset 21

4.1 Quantitative results on synthetic test dataset 44

Chapter 1

Introduction

1.1 Problem Domain

With the ever-expanding field of digital marketing and content development in every field, picture and text editing have become routine tasks. Professional picture editing of text images (images including text) necessitates the use of a variety of image editing tools as well as the expertise of photo editors. Imagine being able to change or replace text in an image while maintaining the original color, font, font size, and other features without any prior image editing knowledge. This eliminates the need for specialized expertise or searching for certain typefaces. Ordinary people can remove sensitive or objectionable content from photographs without affecting the image's original, genuine appearance. Professionals can also profit from editing scene texts at the word level because applying multiple fonts and colors to a specific logo to please a client takes a long time. Bengali is the world's sixth most spoken language, with around 228 million native speakers [4]. Although there have been a few studies on this topic for other languages such as English and Chinese, but none for Bengali. Text style retention has a wide range of applications. Some of the important ones are content generation, text detection, text correction, text replacement, hiding offensive and sensitive terms, and translation.

1.2 Motivation

This is a relatively unexplored topic, and fewer investigations have been done over the years. Some of the notable ones are Wu *et al* [3], Yang *et al* [5] and Roy *et al* [6]. However, these works' implementations are mainly focused on English and Chinese. They all have their fair share of perks and limitations. However "**Editing Text in the wild**" by Wu *et al* [3] catches our attention with its sophisticated and synthetic data generation method and word-

level editing capabilities. Unlike some of the previous works that require character-level editing, word-level editing allows us more freedom in text placement and broader editing options. The usefulness of the topic also draws our attention such as hiding information, erasing sensitive or offensive words, correcting errors, augmented reality (AR), removing dependency from editing tools, reducing the time for searching and pleasing clients, and so on. The desire for contributing to the Bengali dataset also inspires us.

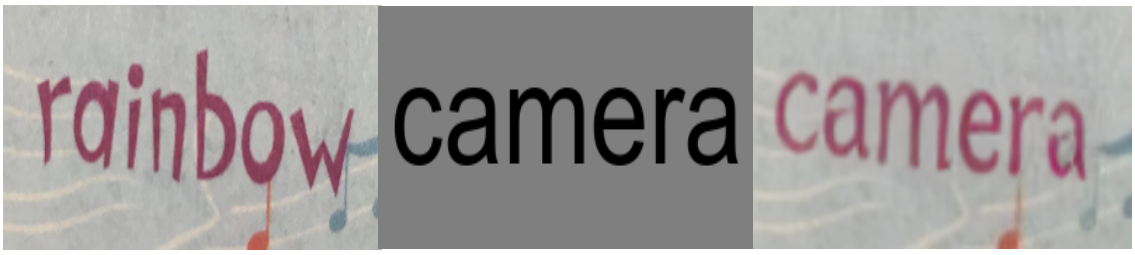


Figure 1.1: Style image Figure 1.2: Input image Figure 1.3: Output image

Figure 1.4: An example from **"Editing Text in the wild"** where an input text fig. 1.2 copies the font style and background of a source image fig. 1.1. The output is in fig. 1.3.

1.3 Our Contribution

The first important difficulty that came to our mind was the lack of any dataset on Bengali. Moreover, this endeavor demands a vast dataset for training. We are contributing a large, rich, and unique Bengali dataset. For training and testing purposes, we have generated a dataset that consists of both synthetic and real-life data. In summary, the contributions of our paper are as follows:

- We propose an artificial Bangla scene text dataset consisting of 2,00,000 images. This will help detection and recognition experiments with ease.
- We formed a real-world dataset of Bangla scene text consisting of 400 different images of text banners, boards, nameplates, street walls, and google images.
- Our modified SRNet model can remove Bengali scene text and repaint with original font and background style.

For the time being, we are experimenting with the original SRNet [3] and have made improvements in the model to make it more suitable for the Bengali language. We are making sure that our dataset can be used dynamically in other activities such as text detection and character recognition. We have built the dataset in a way that can be used for text recognition in a supervised manner. Very few works in text recognition have been done in the

Bengali language. Moreover, most of the works are on the character level. So, we are creating a new beginning in the Bengali language by editing Bengali words in scene images.

Chapter 2

Literature Review

Text appears in many of the design and scene images, providing visitors with essential context information. It's tough to create invisible characters with acceptable visual consistency due to the limited amount of visible characters. Natural features such as brightness, contrast, shadows, perspective distortion, and others make replacing a character straight from a picture more challenging.

We examined related articles, analytical postings, and narrative blogs to complete this work, and we exercised coding to put the model into action. We've tried to figured out how and why all of these systems function at this point. To date, we've learned a variety of topics in machine learning, artificial intelligence, and deep learning to expand our knowledge base. We created an entirely new collection of Bengali Scene Text Images from the ground up. We used SRNet to train this dataset and tweaked the hyperparameters to get a precise result.

2.1 Background Study

2.1.1 Convolutional Neural Network

A Convolutional Neural Network (CNN) [7] is a Deep Learning algorithm. First, it takes an input image and adds variable weights and biases to distinct objects. Each neuron receives inputs and follows a non-linear relationship. By implementing an optimizer, a loss function, and forward and backward propagation, the network is able to discern one from the other. There are numerous architectures of CNNs accessible. CNN has two primary operations. They are convolution and pooling. Each dataset contains a feature map. Convolution can extract out those features from the data set. CNN saves the related information as a result of this extraction. The pooling proce-

dure is also known as subsampling. The feature map has dimensionality. The pooling operation decreases that dimensionality from the convolution technique. The most widely used pooling operations are max pooling and average pooling. The activation function that is employed largely in CNN is called "relu". Relu conveys gradient descent using backpropagation while training.

2.1.2 Generative Models

Variational Autoencoders

The main notion of autoencoders is very straightforward and consists of setting an encoder and a decoder as neural networks and learning the best encoding-decoding scheme via an iterative optimization method. We feed the autoencoder (the encoder followed by the decoder) architecture some input and compare the encoded-decoded output with the starting data to backpropagate the errors and update the weights of the networks. Actually, an autoencoder is nothing but a combination of an encoder and a decoder.
In terms of content production, autoencoders have a big limitation: that is, the training part is not regularised, so the decoder can not decode a new space from latent space.
Variational Autoencoder (VAEs) [8] is nothing but an autoencoder where the latent space is sufficiently regularized (well "organized" by the encoder during the training phase) such that we can choose a point randomly from the latent space and decode it to retrieve fresh content. In recent years, deep generative models have proven to have an exceptional ability to produce very realistic bits of content such as images, texts, and sounds. Among these, VAEs demand special attention. Its encoding process is monitored during the training program to assure the generation of new data.
Dimensionality reduction is one of the key features of VAE. In this procedure, the encoder encodes "new features" from existing features. And the decoder undertakes the reverse operation. The main objective is to maintain the maximum amount of information while encoding and the minimum amount of reconstruction error when decoding. The overall architecture is obtained by compiling the encoder and decoder elements.
Principal Components Analysis is the idea of building additional features in a linear combinational fashion that projects the data onto the subspace such that the initial data can obtain minimal errors. Just like a typical autoencoder, VAE is an architecture consists of both an encoder and a decoder that is trained to minimize the reconstruction error between the encoded and decoded data.
This architecture has contributed a little to the style transfer procedure of our method.

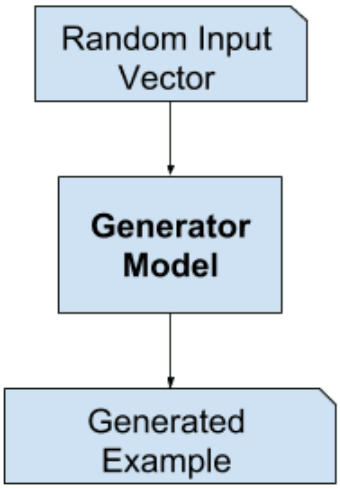


Figure 2.1: Example of the GAN Generator Model

Generative adversarial networks

GANs: Generative adversarial networks [9] present a means of researching deep learning. This is a developing process for both semi-supervised and unsupervised learning. Basically, there are two processes simultaneously operating together. One is a generator, while the other is a discriminator. The generator simulates random test instances according to the dataset or mimics them, while the discriminator tries to differentiate whether they are real or phony. Today, GANs may be applied in a range of applications and research domains. Mostly, image synthesis, style transfer, resolution, and classification solutions.

There are two sub-models in the GAN model architecture. They are:

- **Generator:** It generates new reasonable examples from the problem domain.
- **Discriminator:** It classifies examples as real or fake.

DCGAN: DCGAN or Deep Convolutional GAN suggested by Radford et al. [10] It is a model based on CNN [7]. It is one of the most popular and successful network architectures for GAN. The fundamental strategy is to replace the fully connected layers in the generator with upsampling convolutional layers. In creating this architecture, the authors explored three sources of inspiration: the replacement of pooling processes with spatial downsampling convolutions; deleting fully connected layers; and batch normalization.

DCGAN is utilized to increase the training stability of the GAN architecture, which has made the generator and discriminator more efficient.

PatchGAN: PatchGAN [11] is a specialized discriminator. When an input image is presented to the discriminator, it needs to evaluate how likely it is that an image is real

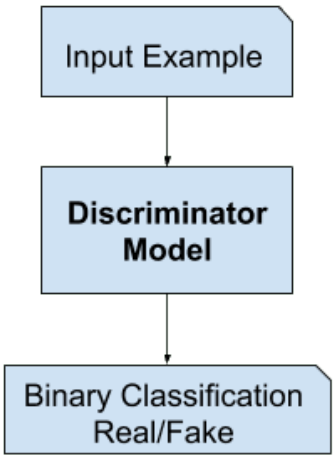


Figure 2.2: Example of the GAN Discriminator Model

or fraudulent. Furthermore, it needs to be evaluated on how close the decision was. The discriminator can be evaluated by estimating how probable an image is real or fake by predicting if a patch of an image is real or false, then evaluating how it did for each patch. A receptive field of one means that it will be examined per pixel, and make one prediction per pixel. What makes the PatchGan Discriminator special is that it lowers the stride of the last two of convolutional layers, resulting in a larger output size.

Pix2Pix: pix2pix [12] learns a mapping from input images to output images. It is one sort of conditional generative adversarial network(cGAN) (cGAN). Pix2pix is not specific to a single application. It has a large range of applications. It synthesizes photographs from label maps. It makes colorized photographs from black and white images. It turns Google Maps photographs into aerial images. It can even transform doodles into photographs. Pix2pix conditions incoming photos and generates appropriate output images.

CycleGAN: An unpaired image is translated into another unpaired image using the condition Gans's. This translation is known as CycleGAN [13]. Each picture domain has specific properties. The properties of one picture domain are translated to another image domain. This translation is done without any training of matched data. When a similar output image is formed via conditioning, the input image is called pix2pix cGAN. CycleGan has an additional loss function and unpaired training data compared to pix2pix cGAN.

CycleGAN doesn't need one-to-one mapping because it exploits cycle consistency losses for enabling training. The source and the target dataset are only essential here.

2.1.3 Neural Style Transfer

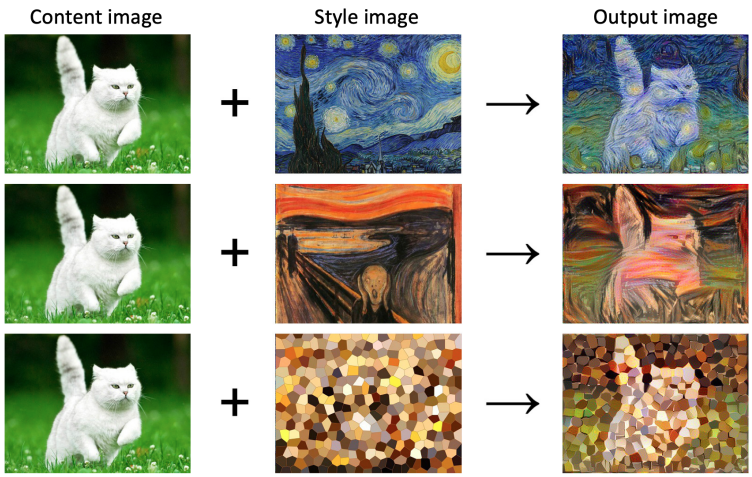


Figure 2.3: An example of "**Neural Style Transfer**".

Neural style transfer is an optimization technique. It receives two images as input: a target image and a style image. The style image may feature an extremely intricate or artistic architecture. Neural Style Transfer blends the two images in such a way that the target image looks like it is painted as the style image. Both the style image and the target image have their own statistics. Neural Style Transfer optimizes the output image to match the statistics of both images. So, the extraction of statistics from the photos is another key aspect of this transformation. This transition is done by the convolutional network. An example showed in Fig.fig. 2.3.

2.2 Related Works

This problem domain is extremely fresh to the research field. After inventing GANs [9] more networks came and practically enhanced computer vision difficulties. Previously, there was a little development on this topic, but nothing for Bengali.

2.2.1 SwapText: Image Based Texts Transfer in Scenes

Recently a substantial effort has been done by Yang et al. "**SwapText: Image Based Texts Transfer in Scenes**" [5]. They introduced SwapText, a three-stage system to

transfer text across scene images. The concept is quite similar like [3] but instead of editing the scene text, they augmented the text with another. This approach not only blends text with on-scene visuals but also conducts different translations between multilinguals. Initially, Chinese to English and vice versa were tested. Their Content Shape Transformation Network (CSTN) was incorporated to function as their framework. They employed the same technique as [3] to make synthetic images.

Text replacement or swapping is beneficial in a variety of contexts, including style transfer, art works, creative applications, and many more. The swapping technique is highly beneficial in data augmentation and computer vision. In recent years, image generation models such as GANs [14], VAE [14], have given strong tools for the realistic image generating process.

Adopting GANs, they provided an end-to-end unified framework entitled Swap-Text that integrates three more sub-networks, Text Swapping Network, Background Completion Network, and The Fusion Network. In the Text Swapping Network, the features of the target image and the style image are extracted and combined using a self-attention network. To improve geometrical and curved qualities, they have created the Content Shape Transformation Network (CSTN). This has considerably increased the quality of numerical prospects and ratios. Lastly, the Background Completion Network developed the background of the style image and improved the textured quality. Finally, the output of the text swapping network and the background completion network were integrated into the Fusion Network to provide a more practical-looking image.

One of the most important aspects of this application is image-based text translation. Initially, they experimented with English and Chinese. But it is claimed that the target language is no matter what option is made, the structure can be implemented in any other language.

The experiment was benchmarked on numerous public datasets such as IIT 5K-Words [15], Street View Text [16], ICDAR 2013 [17], ICDAR 2015 [18], CUTE80 [19]

Although the Text Swapping Network is a new architecture for altering scene text images, it has few limitations. The model fails when it has a wavy image. As they have said, given to the small amount of training data, this kind of failure could be predicted.

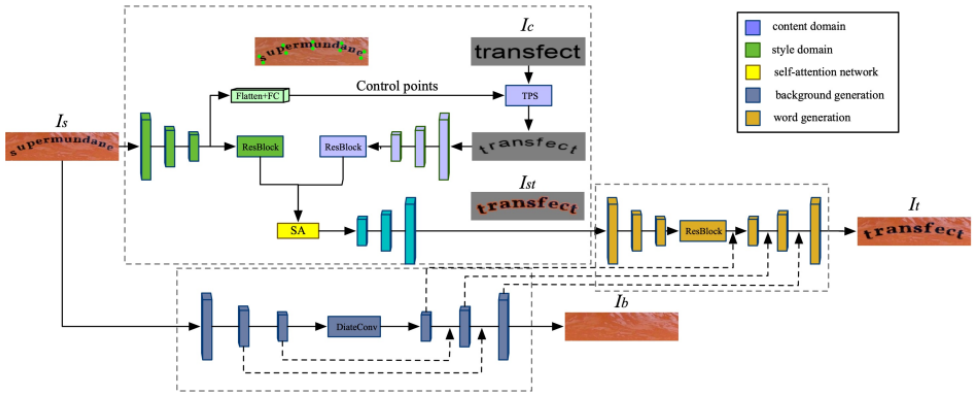


Figure 2.4: Model of "**SwapText: Image Based Texts Transfer in Scenes**" [1].

2.2.2 TextStyleBrush: Transfer of text aesthetics from a single example

This is a very new study by Krishnan $et.al.$ [2]. They proposed text style brush (TSB) for text-style disentanglement and transfer. Their technique is self-supervised and trained on text samples from real images, sans ground truth style labeling. Given a recognized text box containing a source style, it extracts an opaque latent style representation. It's not essential to define this representation utilizing semantic parameters, typeface, color encoding, and spatial transformation). Instead, they optimize its representation to allow photo-realistic display of new information in the original style. This transfer is applied in a one-shot method, using only the single source sample to learn the required text style. However, soft segmentation masks are learned, however, in a self-supervised manner, in order to increase the quality of the foreground (text) produced by the generator. They proposed style and content encoders for one-shot disentanglement of a source style image. Additionally, an unique generator, conditioned on content and multi-scale style, generates the target style picture along with the foreground mask. The method is self-supervised, learning to maintain both source style and target content using innovative loss functions.

2.2.3 Synthetic Data for Text Localisation in Natural Images

Text spotting, specifically the manner of reading text in natural images, is a highly exciting aspect in applications of computer vision. "**Synthetic Data for Text Localisation in Natural Images**" [1] contributed two approaches for text detection in the scene images. They suggested a new way to generate synthetic images of text that seamlessly mixes text into existing natural scene images. With this strategy, an entirely new dataset dubbed "SynthText in the Wild" was developed that is suited for training high-performance scene text detectors. The fundamental distinction with previous synthetic text datasets is that [1] dataset has character level areas whereas

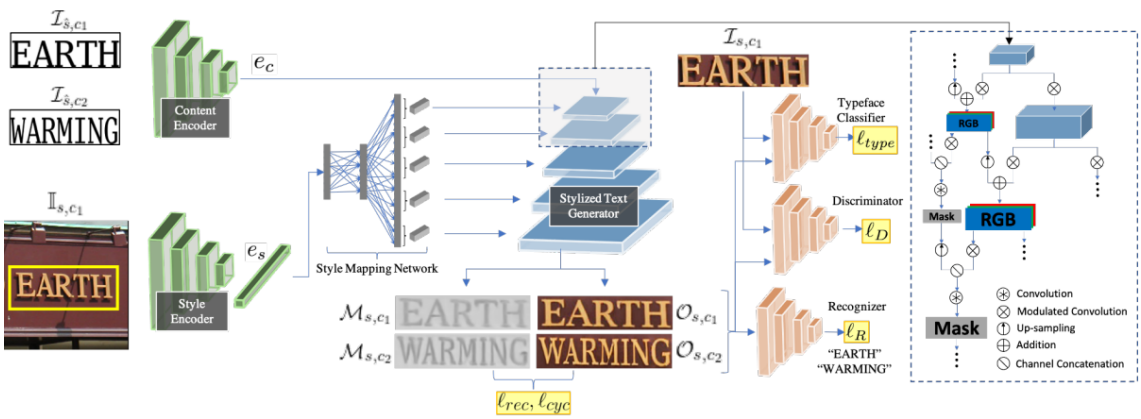


Figure 2.5: Model of "**TextStyleBrush: Transfer of text aesthetics from a single example**" [2]

others are word level and unsuitable for training detectors.
This research is all about text detection in scene images. Since the introduction of state-of-the-art algorithms, text detection has come a long way. Since the implementation of this paper, text detection networks have been trained based on real images such as scene images containing text. But this brought a tremendous difference in this procedure. To make the model a better detector, they have added an image synthesis approach. With this technology, they built their dataset artificially, and that has been shown more effective than existing datasets. As, we have to execute a similar task, we are focused about this model, not the rest.

Their synthetic engine comprises a few processes to generate realistic scene text images. First, they identify relevant text and image samples, and then the image is split into unbroken parts based on local color and textures. After that, CNN [**?**] is applied to measure the depth of the pixel and the region's color to render the text into the local surface orientation. The text is blended into the scene using Poison Image Editing [20]. According to their allegation, the engine takes around half a second to build a new scene-text image. They have made the Dateset [21] public, adding 800K scene images.

The Synthetic Dataset [21] is analyzed and achieved a fantastic score over the existing ICDAR [17], SVT [16] datasets. It is correctly labeled and stored in a systematic fashion.

Although this dataset is beneficial for detectors, it is limited to the English language.

2.2.4 Editing Text in the Wild

One of the leaders of this sort of work is "**Editing Text in the Wild**" [3]. This study is the first work to alter text images at the word level. In this paper, Wu et al. [**?**] introduced a network termed SRNet that comprises of three components. Text Converter, Background Image restoration, and Composite Module. The major objective of their research was to alter a text in a scenario without hurting its realistic vibe. Text style uniformity and backdrop texture retention were tough components of their procedure. Considering these obstacles, they introduced their paradigm and combined modules to perform word-level editing tasks.

Editing Scene Text has gained attention in the business for its various practical uses such as text image synthesis, advertising photo editing, text image correction, augmented reality translation, you name it.

They noted two key obstacles for scene text editing. One is text style transfer, and another is backdrop texture retention. Considering all the problems, their suggested SRNet split the big task into simple tasks. Firstly, the Text Converter Module converts the text style to the destination text, including details of characteristics. Another module termed the Skeleton Guided Learning Mechanism holds the geometrical semantics. Secondly, the Backdrop Restoration Module erases the source text stroke pixels and replaces them with a background texture in a bottom-up fusing fashion, following the U-Net [22] architecture. Finally, the Fusion Module completes the cycle through the encoder-decoder FCN [23] framework.

For training reasons, Synthetic Dataset [1] Real-world Dataset used however ICDAR 2013 [17] used for testing purposes.

According to their assertion, SRNet results are better than pix2pix [12] which is a successful technology used in image to image translation. They confirmed the usefulness and robustness of SRNet on word-level editing.

The research was also undertaken for "Cross-Language Editing" from English to Chinese, and little, but not as much as it should be. Another subtask they highlighted was "Text Information Hiding" as it may wipe specified sections freely without hindering the backdrop texture, color, and information.

This approach also contains failure cases. It has been found that a shadow of the original text remains in the output photos. It performs poorly when the structure is

spatial and intricate. They claimed these failed situations were due to the insufficiency of training data and samples.

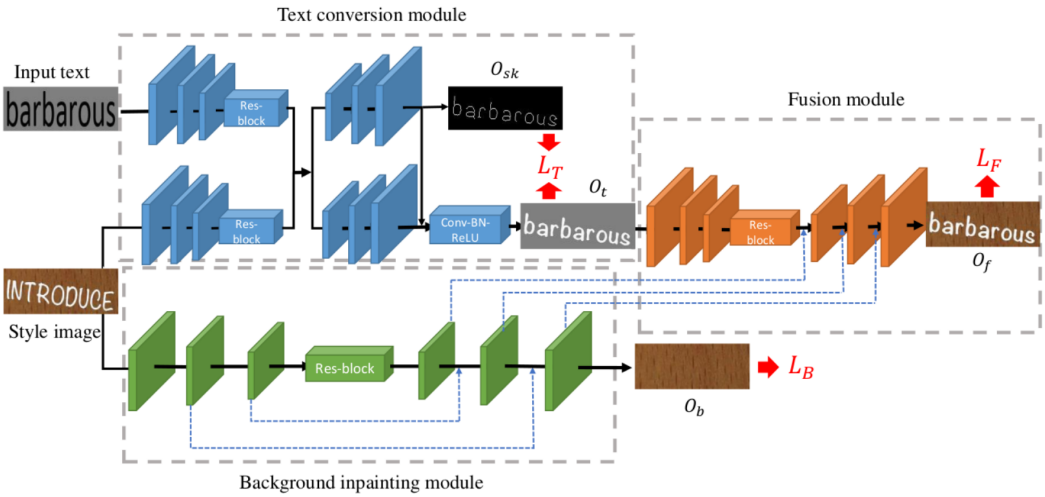


Figure 2.6: Model of "**Editing Text in the Wild**" [3].

Chapter 3

Our Contribution

3.1 Dataset

3.1.1 Artificial Data Generation

To our best knowledge, no earlier study has been done in this topic in the Bengali language. This leads in a lack of any form of dataset availability. Prior work in other languages, such as English, involved datasets comprising millions of examples [3]. As the Bengali language has more alphabets and a more complex alphabet structure, it requires more examples than the English language. Building a dataset with this big number of samples from scratch is tough without a lot of community support. In addition, we are conducting this assignment in a supervised manner for which the real outcome after modifying an image (ground truth) needs to be present. This is not achievable for real-life datasets. That is why we explored working with an artificially generated dataset where the instances would imitate the real-life scene text graphics. We applied the synthetic data production method in [3] to generate artificial data. For this reason, we used the SynthText [1] dataset, which includes roughly 11,000 backdrop pictures. We used Bengali Unicode typefaces and a world list of roughly 48,000 Bengali dictionary words. The procedure entails selecting the backdrop image, font, text and other style elements such as brightness, contrast, color, curve, rotation, shadow, border randomly and rendering the text on the background image to recreate the real-life events as closely as feasible.

3.1.2 Rendering complex Unicode scripts

Complex text layout (CTL) [24] or complex text rendering is the typesetting of writing systems in which relations to other graphemes can influence the shape or positioning of

(a) without libraqm patch (b) with libraqm patch

Figure 3.1: Font rendering for complex script

a grapheme. For instance, Bengali, which evolved from the Brahmic [25] family, cannot be typeset by merely displaying each character one after another in straight rows. This is because of the context-sensitive shape and ligatures of Bengali alphabets. We have used Pyagame [26] library to generate and edit scene text images. As latest versions of Pygame do not support complex text layout and Bengali complex text gets broken while rendering on an image, we had to manually patch using Libraqm [27] library to render properly. Pyagame [26] module patched [28] with Libraqm [27] library was used to repair broken ligatures when rendering Unicode Bengali ligatures. Figure 3.1 depicts such scenarios in action, where instances of displayed text images are presented before and after Pygame was patched with the libraqm library.

3.1.3 Artificial Dataset

Each example of the dataset comprises 7 image pairs. For each example, we prepared 2 input images (I_s, I_t) and 5 label images(t_{sk}, $mask_t$, t_t, t_b, t_f)

1. I_s: Style image that contains the style text 'a' and the background image
2. I_t: Input image that contains the input text 'b' and gray background
3. t_{sk}: Skeletonization of the style text 'b'
4. $mask_t$: Binary mask of styled text 'b'
5. t_t: Style text 'b' rendered on the gray background
6. t_b: Background image without the style text
7. t_f: Style text 'b' rendered on the background image

We have generated almost 200000 image pairs, each having seven images of the type indicated above. Altogether, we have created 7 * 200000 = 1400000 images for the training dataset. fig. 3.2 provides samples of training data. We have utilized 140 unicode fonts and 48,000 Bengali words for constructing the dataset. Table 3.1 indicates the supplemental materials utilized for data production. Figure 3.3 displays frequency of Bengali vowels and Figure 3.4 indicates frequency of Bengali consonants in the dataset images.

Figure 3.2: Examples of generated training data. From top to bottom: I_s, I_t, t_{sk}, $mask_t$, t_t, t_b, t_f. See section 3.1.3 for details

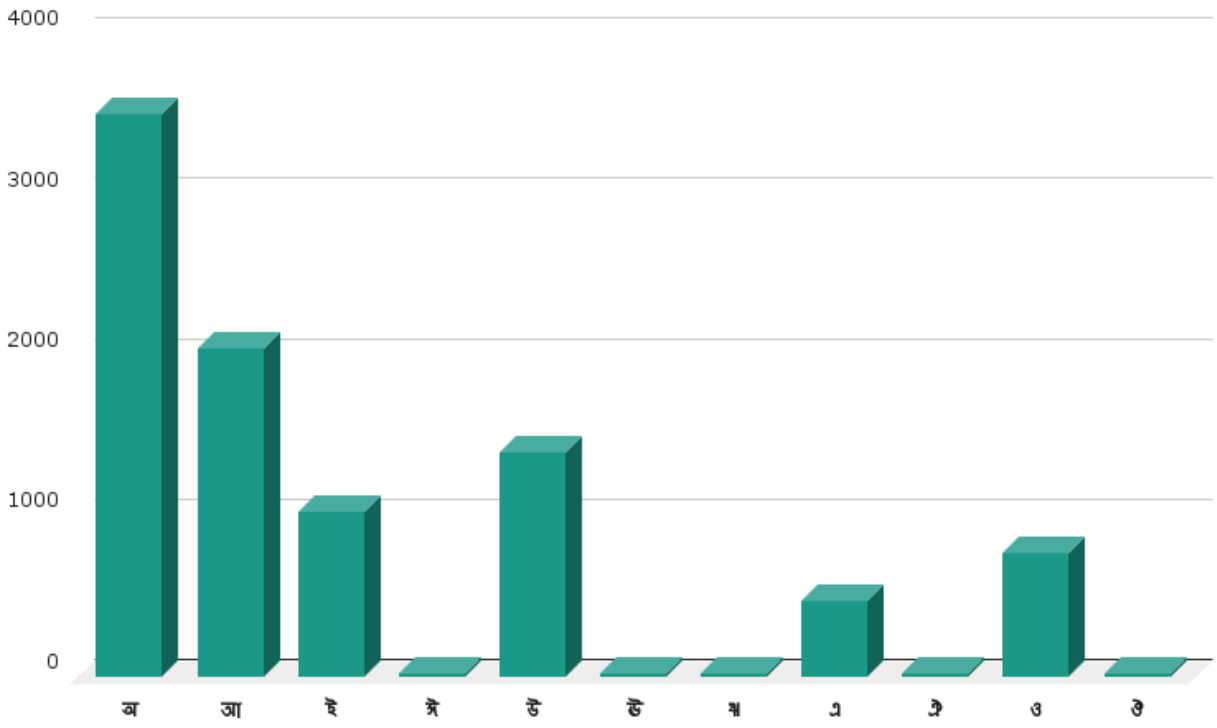


Figure 3.3: Frequency of vowels in the dataset image texts

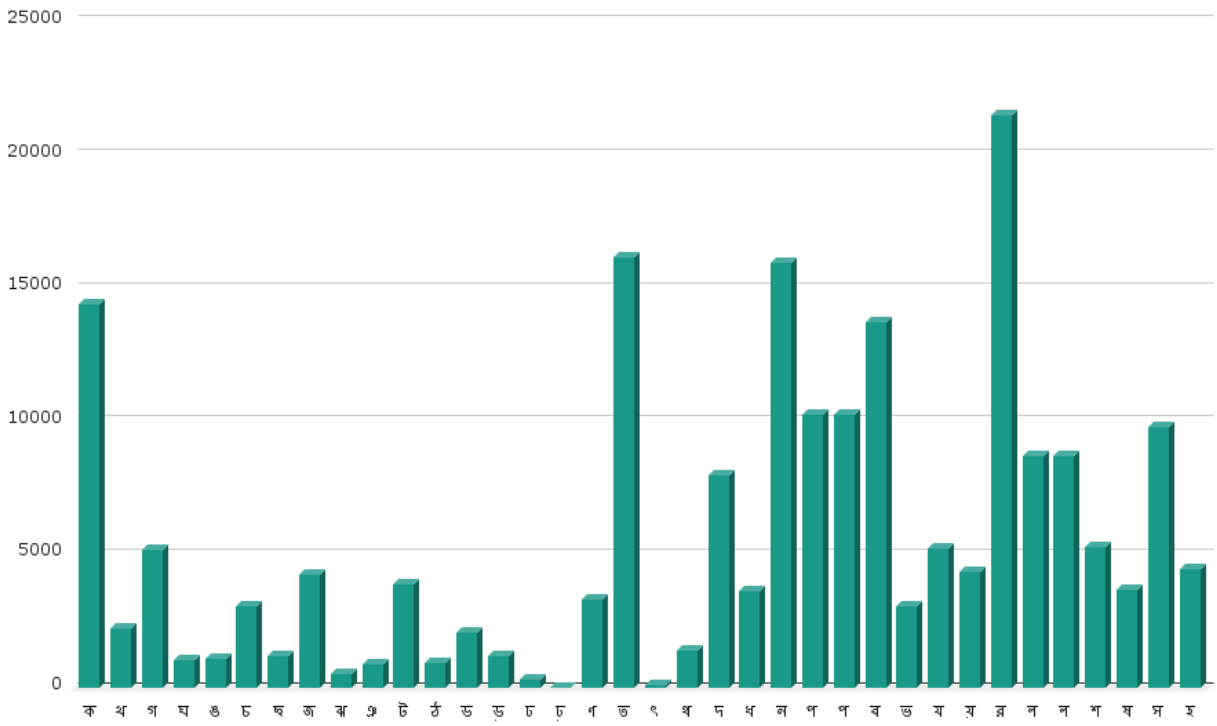


Figure 3.4: Frequency of consonants in the dataset image texts

In the initial phase, we saw unsatisfactory results for several characters due to character frequency in the wordlist not being regularly distributed. Hence, we did normalization and raised the character frequency of those characters. Figure 3.5 displays the frequency of Bengali vowels and Figure 3.6 indicates the frequency of consonants in the dataset pictures after normalization. We have also included neumeric characters to our wordlist, Figure 3.7.

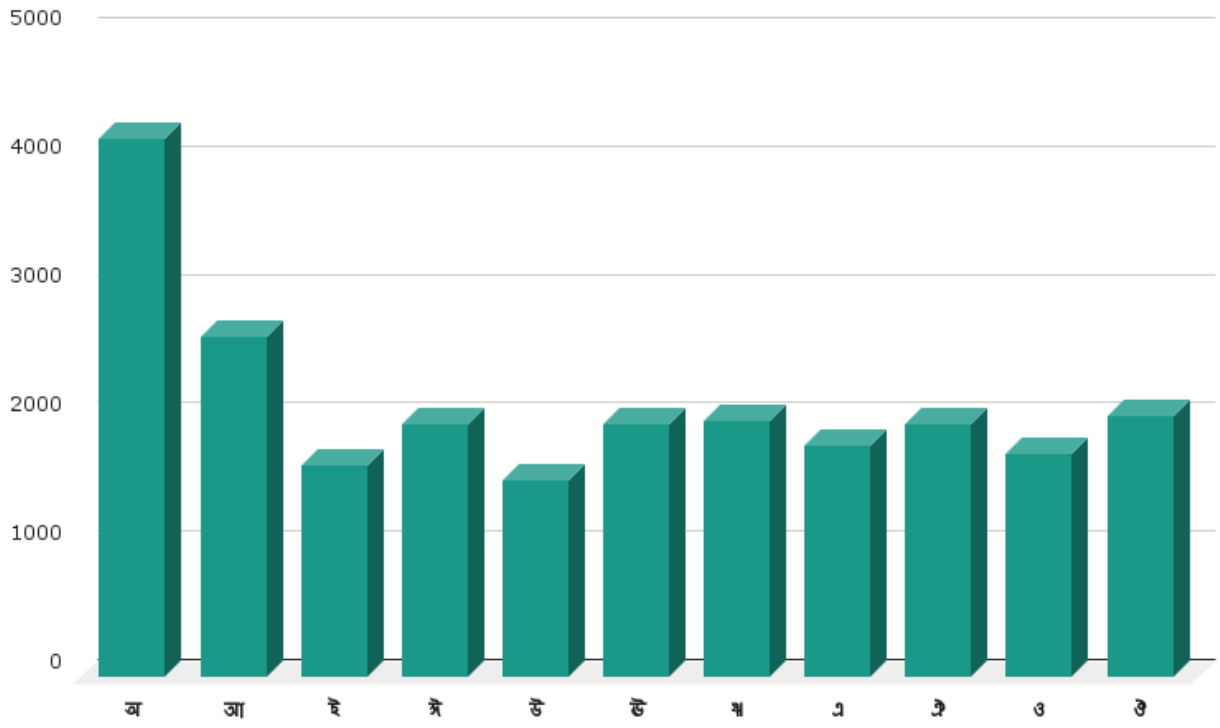


Figure 3.5: Frequency of vowels in the dataset image texts after normalization

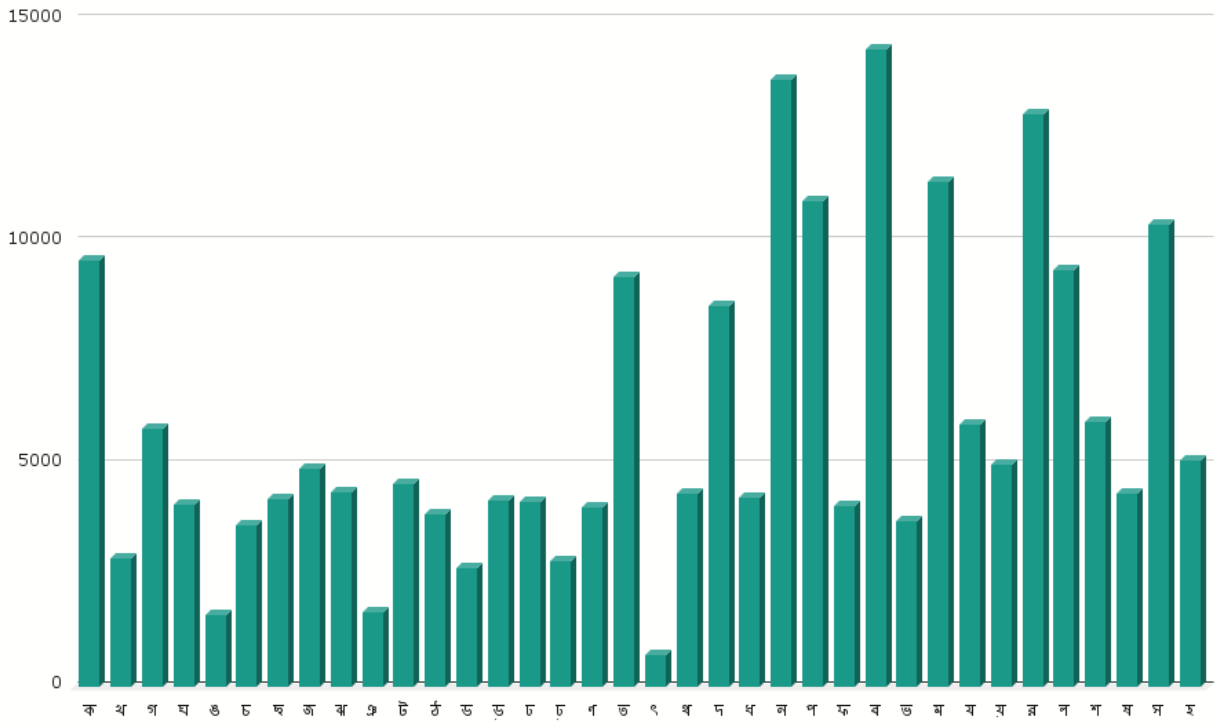


Figure 3.6: Frequency of consonants in the dataset image texts after normalization

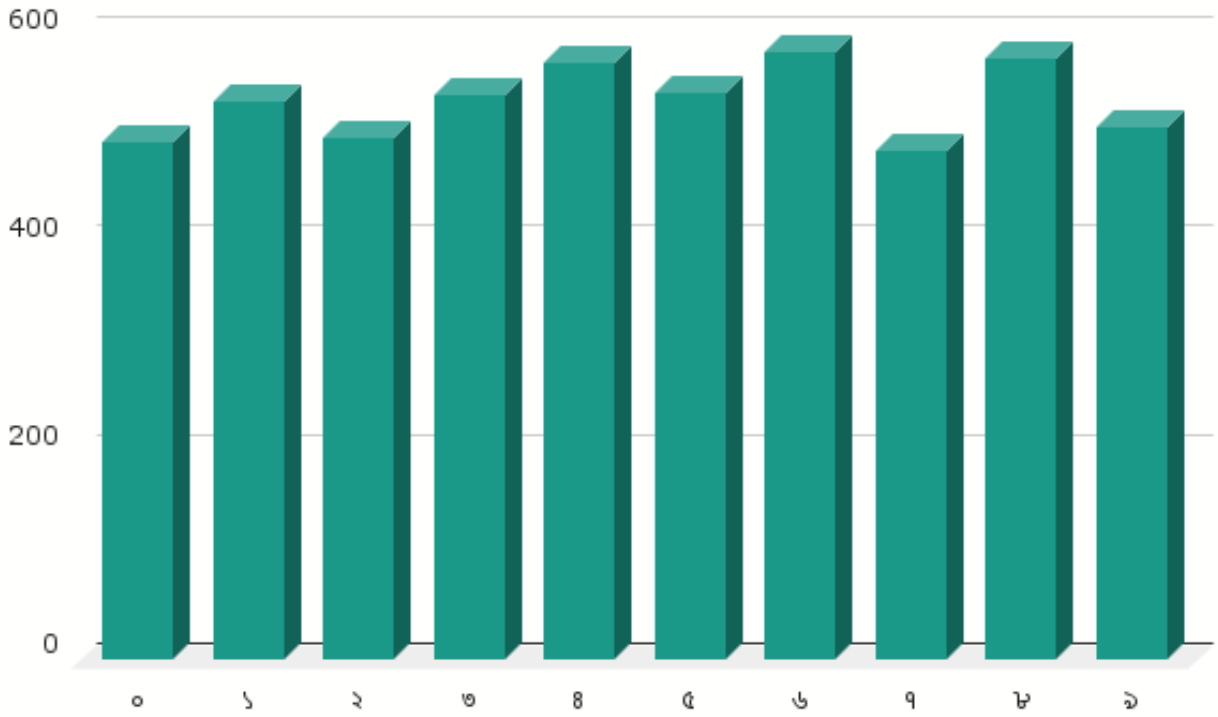


Figure 3.7: Frequency of neumeric characters in the dataset image texts

Table 3.1: Supplementary materials

Material	Total No. (Initial Experiment)	Total No. (Final Experiment)
Background Image	11000	11000
Bengali Unicode Font	80	140
Word Count	40000	48000

3.1.4 Real-world Dataset

Real-world data was used for testing reasons only. Real-life photographs were acquired through fieldwork. We have gathered Bengali Scene text images from (a) different shopping malls, shops, and street-walls, (b) screenshots from Google Maps street view, and (c) Bengali scene images from Google images. Figure 3.8 displays images we obtained via fieldwork, and Figure 3.9 shows images we collected from street view. The distribution of total, training, and testing data is shown in Table 3.2

Figure 3.8: Real-life images from fieldwork

Figure 3.9: Real-life images from Google Maps Street View

Input Source Text	Input Source file	Input Target Text	Input Target file	Mask Image Text	Mask Image file
ইঞ্জিন	0.png	স্পিকিং	0.png	স্পিকিং	0.png
তদ্ভব	1.png	অব্যয়	1.png	অব্যয়	1.png
প্রিসিশন	2.png	অব্যয়	2.png	অব্যয়	2.png
অব্যয়	3.png	রিডিং	3.png	রিডিং	3.png
চৌত্রিশ	4.png	বিদেশি	4.png	বিদেশি	4.png
স্পিকিং	5.png	জাভা	5.png	জাভা	5.png
জাভা	6.png	স্পিকিং	6.png	স্পিকিং	6.png
রিডিং	7.png	চৌত্রিশ	7.png	চৌত্রিশ	7.png
সাহিত্যের	8.png	রিডিং	8.png	রিডিং	8.png
ছত্রিশ	9.png	রিডিং	9.png	রিডিং	9.png
বিদেশি	10.png	প্রিসিশন	10.png	প্রিসিশন	10.png
তদ্ভব	11.png	অব্যয়	11.png	অব্যয়	11.png
সাহিত্যের	12.png	ছত্রিশ	12.png	ছত্রিশ	12.png
বত্রিশ	13.png	প্রিসিশন	13.png	প্রিসিশন	13.png
ইঞ্জিন	14.png	ছত্রিশ	14.png	ছত্রিশ	14.png
বিদেশি	15.png	ইঞ্জিন	15.png	ইঞ্জিন	15.png

Figure 3.10: Dataset structure for text recognition task

Table 3.2: Distribution of Data in Dataset

Dataset	**Total Image**	**Training**	**Testing**	**Validation**
Generated Dataset	200000	196000	2000	2000
Real-life Images	400	0	0	400

3.1.5 Other use cases of our Dataset

We developed our dataset with various possible use cases in mind. We noticed that, unlike English and other languages, Bengali currently has no robust text recognition dataset available. There are some datasets for handwritten character recognition such as Ekush [29] and BanglaLekha-Isolated [30] but these datasets contain only single-character black and white or grayscale images and are not ideal for scene text recognition related tasks. Figure 3.10 shows the dataset structure which we are planning to use for Bengali scene text recognition task in future.

3.2 Application Methodology

Our aim is to take the source text style image and I_s and the target text image I_t as input and transfer the source text style to the target text style, and replace the source text with the now modified target text. To do this, we used our prepared dataset and the SRNet proposed by Wu *et al.* [3]. The structure of SRNet involves three phases. (1) copy the text style from the source image text to the target text, (2) delete the source text from the source text image and fill in the blanks, and (3) merge the target text and the source text image. SRNet conducts these stages with three modular and joint-trainable sub networks.

1. Text conversion module.
2. Background inpainting module.
3. Fusion module.

As, illustrated in Fig. Figure 3.11 ...

1. TCM (text conversion module) copies text styles from source text images and transfers them to the target text. It encompasses font, color, position, rotation, perception, and scale. Here we adopted, skeleton-guided learning mechanism [3] to keep the semantics of the target text. TCM takes two images as input: the target text image and the source style image. Here, the target text image duplicates the font, style, color, geometric form, backdrop, etc. from the source style text picture. TCM provides an output image. The output picture contains the target text and the source style. To accomplish this, we require an encoder-decoder FCN. Both the input photos are encoded. Both the images are encoded using four residual blocks and three down-sampling convolutional layers. After that, those two photos are combined. The output image is produced through decoding. One Convolution-BatchNorm-LeakyReLU block and three up-sampling transposed convolutional layers are employed in the decoding stage. The Skeleton Guided Mechanism has been utilized for making powerful texts.
2. BIM (Background inpainting module) erases the text area from the source style text image and fills up the unoccupied areas by restoring appropriate texture from the nearby region. It follows a bottom-up feature fusion strategy, which is the overall architecture of U-Net [31]. BIM receives the background after eliminating the text from the style source image. BIM takes the style source image as input. The background of the style source image is the output of BIM. The blank space after deleting the text is filled with a suitable surface. The input image is encoded by three down-sampling convolutional strides, two layers, and followed by four residual blocks. After that, the decoder creates the output image using three up-sampling convolutional layers.

A Leaky ReLU activation function has been utilized in each layer, but for the output layer, we have used the tanh function.

3. FM (Fusion module) fuses the output from TCM and BIM meaning it places the output style text with the source background image. FM incorporates both the target text image and background surface information. The encoder-decoder FCN framework is utilized in FM. The output of the TCM is encoded by three down-sampling convolutional layers and residual blocks. The ultimate altered image is formed by decoding it with upsampling transposed convolutional layers and Convolution-Batch-Norm-LeakyReLU blocks.

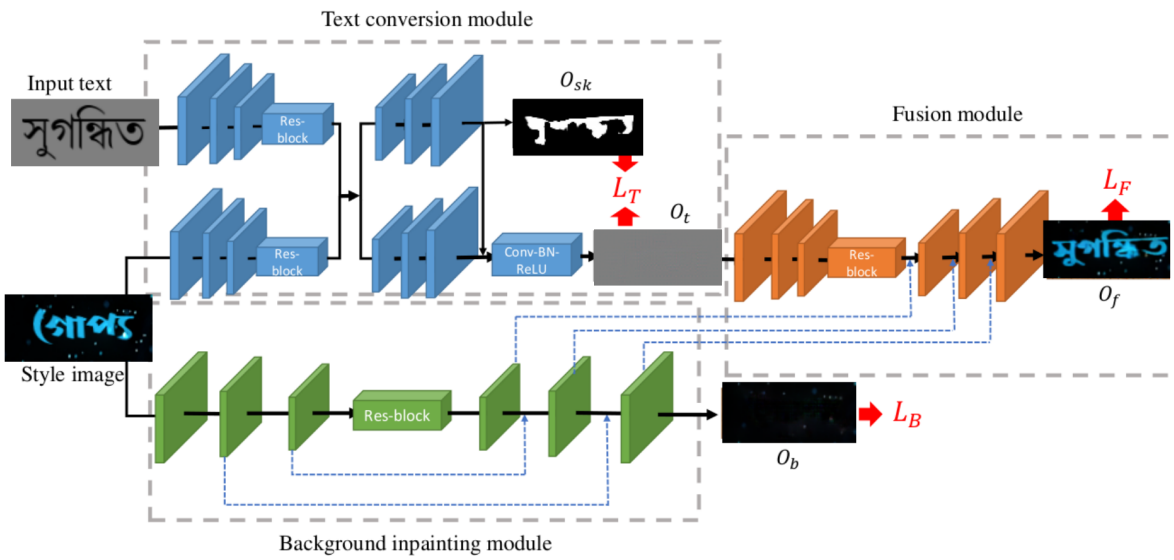


Figure 3.11: The architecture of SRNet [3]

Chapter 4

Experiments and Results

4.1 Experimental Setup

Experiments were run on a local machine. We used the Adam [14] optimizer to train the model. The learning rate was set to 1e-4 with a batch size of 4. The decay rate for learning rate was 0.9, and the values of $beta1$ and $beta2$ were 0.9 and 0.999. The hyperparameters for dice loss $alpha$ was 1 and for l1 loss $theta1$ was 10. For vgg loss, $theta2$ and $theta3$ were 1 and 500 respectively. We modified the SRNet such that the model can be trained with variable image sizes. We trained the model for 405,000 iterations. The model took about 78 hours to train with a single NVIDIA RTX 2060 Super graphics card.

4.2 Results

4.2.1 Results on synthetic Dataset

Figure 4.1: Example 1: Results on synthetic dataset

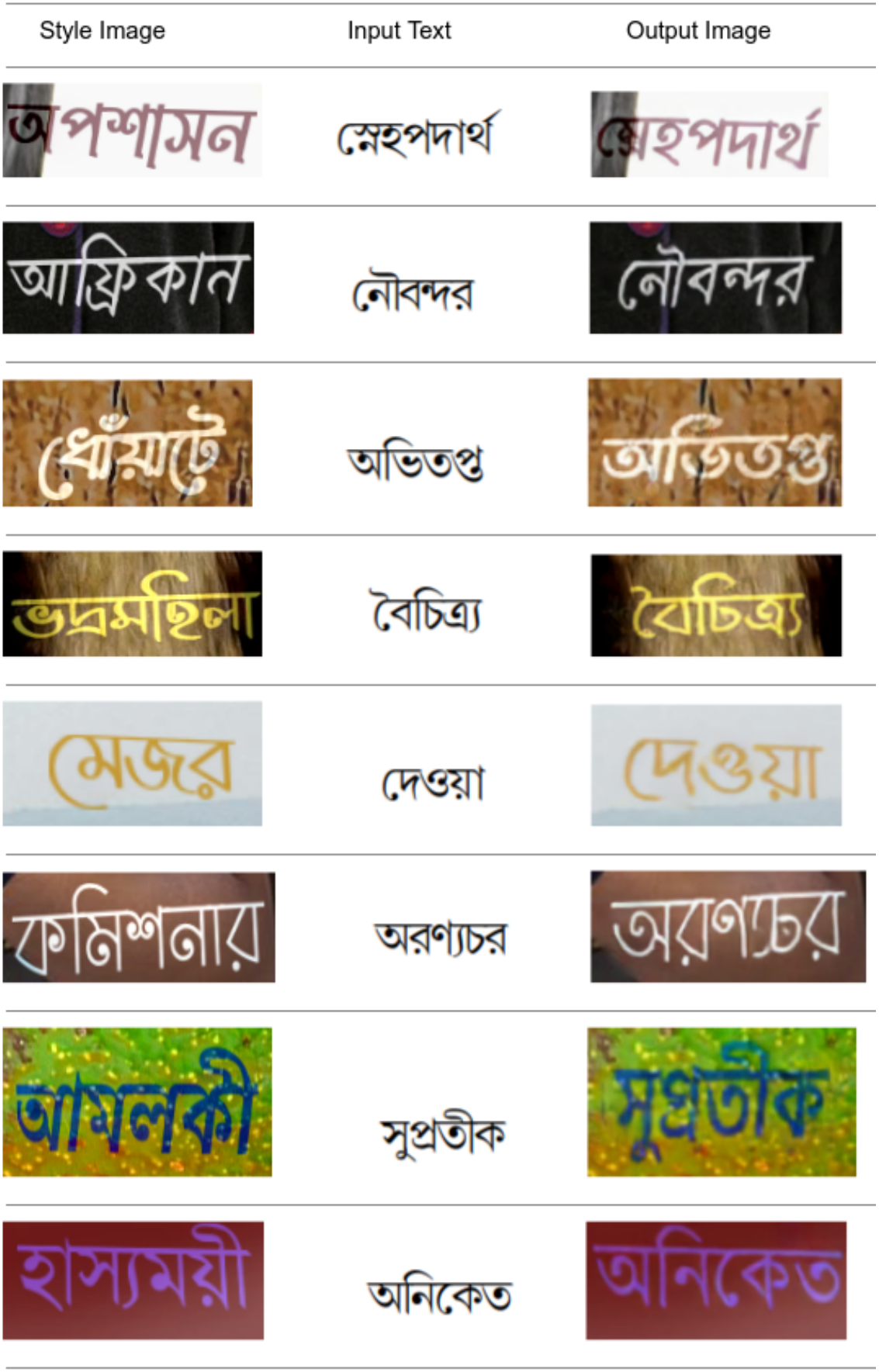
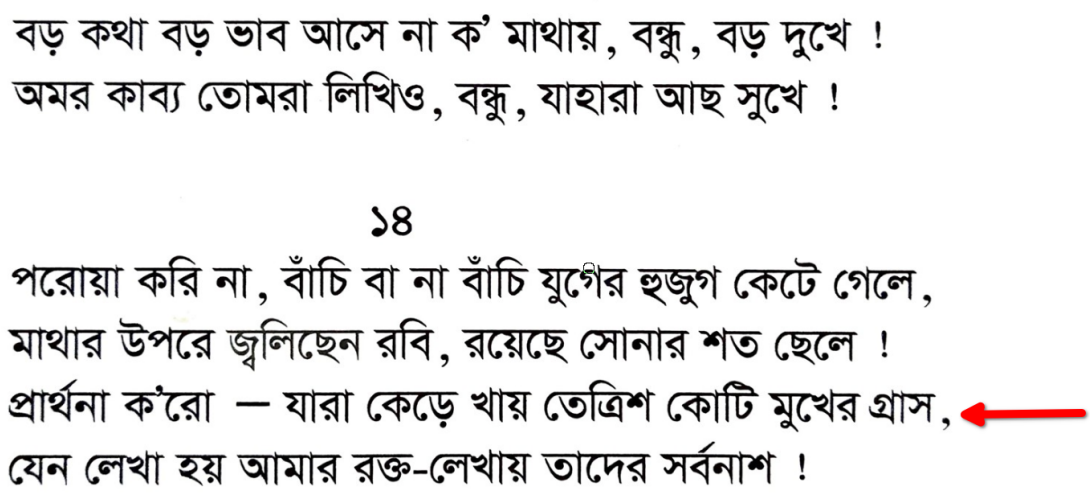


Figure 4.2: Example 2: Results on synthetic dataset

4.2.2 Results on real-life scene images

Figure 4.3: Example 1: Results on real-life scene images

Figure 4.4: Example 2: Results on real-life scene images

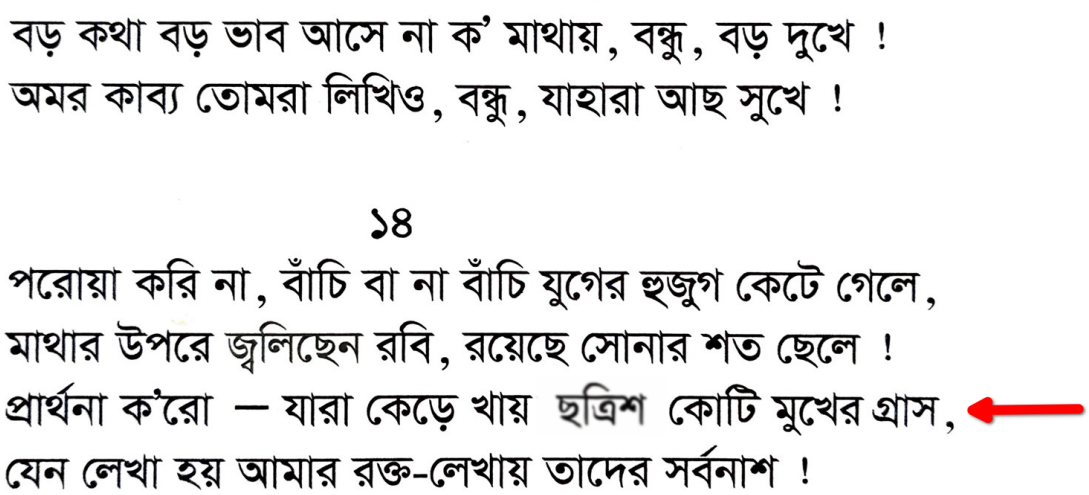

বড় কথা বড় ভাব আসে না ক' মাথায়, বন্ধু, বড় দুখে !
অমর কাব্য তোমরা লিখিও, বন্ধু, যাহারা আছ সুখে !

১৪

পরোয়া করি না, বাঁচি বা না বাঁচি যুগের হুজুগ কেটে গেলে,
মাথার উপরে জ্বলিছেন রবি, রয়েছে সোনার শত ছেলে !
প্রার্থনা ক'রো — যারা কেড়ে খায় তেত্রিশ কোটি মুখের গ্রাস,
যেন লেখা হয় আমার রক্ত-লেখায় তাদের সর্বনাশ !

(a) Input Style Image

(b) Input Text

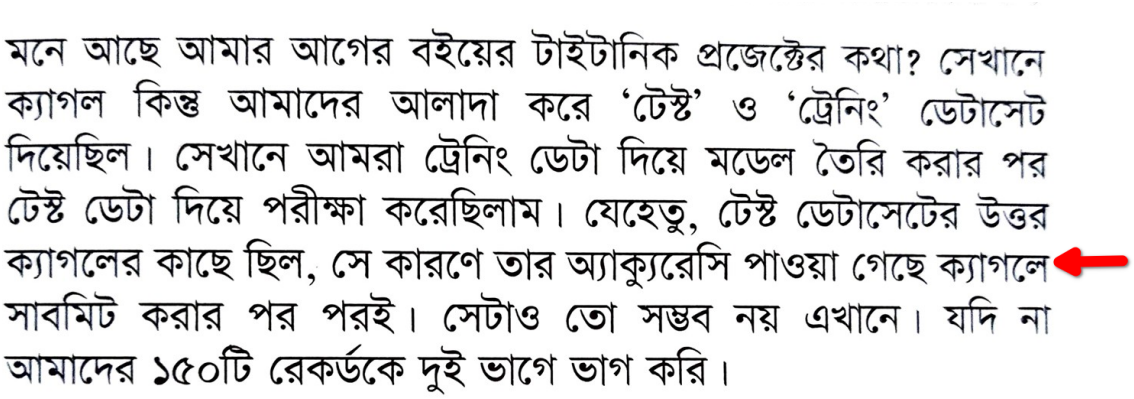

বড় কথা বড় ভাব আসে না ক' মাথায়, বন্ধু, বড় দুখে !
অমর কাব্য তোমরা লিখিও, বন্ধু, যাহারা আছ সুখে !

১৪

পরোয়া করি না, বাঁচি বা না বাঁচি যুগের হুজুগ কেটে গেলে,
মাথার উপরে জ্বলিছেন রবি, রয়েছে সোনার শত ছেলে !
প্রার্থনা ক'রো — যারা কেড়ে খায় ছত্রিশ কোটি মুখের গ্রাস,
যেন লেখা হয় আমার রক্ত-লেখায় তাদের সর্বনাশ !

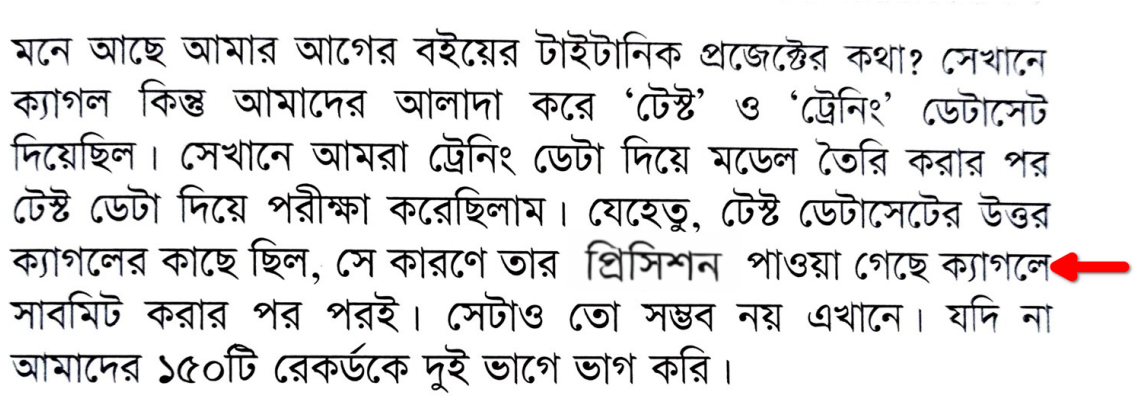
(c) Output Image

Figure 4.5: Example 3: Result of editing text book (black and white)


মনে আছে আমার আগের বইয়ের টাইটানিক প্রজেক্টের কথা? সেখানে ক্যাগল কিন্তু আমাদের আলাদা করে 'টেস্ট' ও 'ট্রেনিং' ডেটাসেট দিয়েছিল। সেখানে আমরা ট্রেনিং ডেটা দিয়ে মডেল তৈরি করার পর টেস্ট ডেটা দিয়ে পরীক্ষা করেছিলাম। যেহেতু, টেস্ট ডেটাসেটের উত্তর ক্যাগলের কাছে ছিল, সে কারণে তার অ্যাক্যুরেসি পাওয়া গেছে ক্যাগলে সাবমিট করার পর পরই। সেটাও তো সম্ভব নয় এখানে। যদি না আমাদের ১৫০টি রেকর্ডকে দুই ভাগে ভাগ করি।

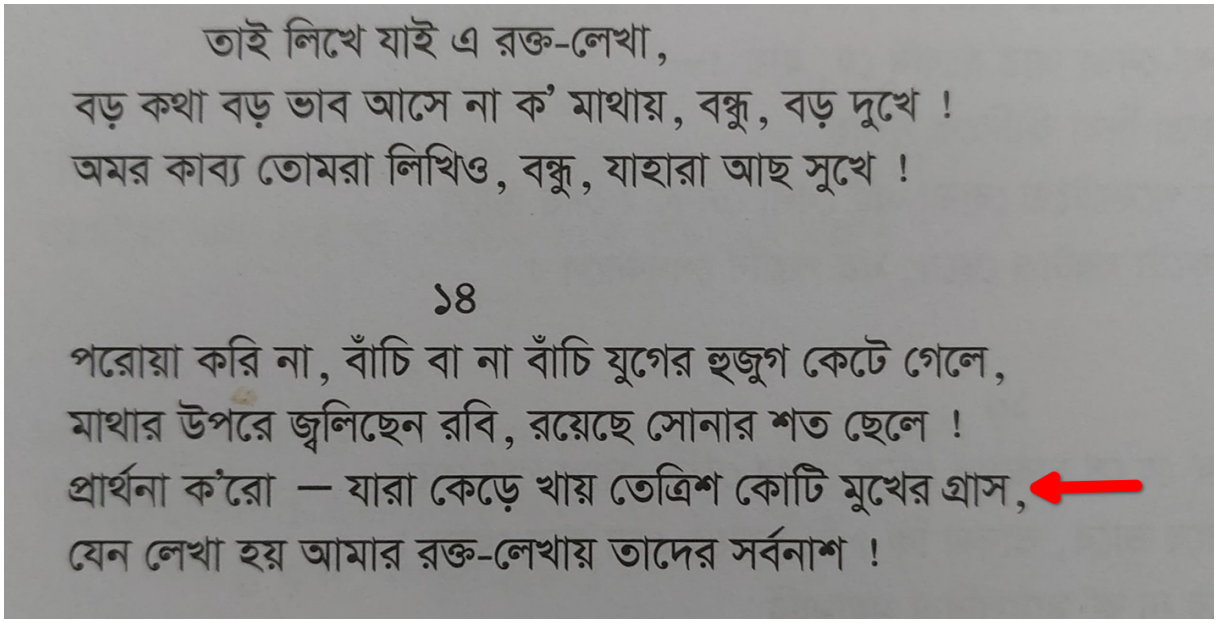
(a) Input Style Image

(b) Input Text


মনে আছে আমার আগের বইয়ের টাইটানিক প্রজেক্টের কথা? সেখানে ক্যাগল কিন্তু আমাদের আলাদা করে 'টেস্ট' ও 'ট্রেনিং' ডেটাসেট দিয়েছিল। সেখানে আমরা ট্রেনিং ডেটা দিয়ে মডেল তৈরি করার পর টেস্ট ডেটা দিয়ে পরীক্ষা করেছিলাম। যেহেতু, টেস্ট ডেটাসেটের উত্তর ক্যাগলের কাছে ছিল, সে কারণে তার প্রিসিশন পাওয়া গেছে ক্যাগলে সাবমিট করার পর পরই। সেটাও তো সম্ভব নয় এখানে। যদি না আমাদের ১৫০টি রেকর্ডকে দুই ভাগে ভাগ করি।

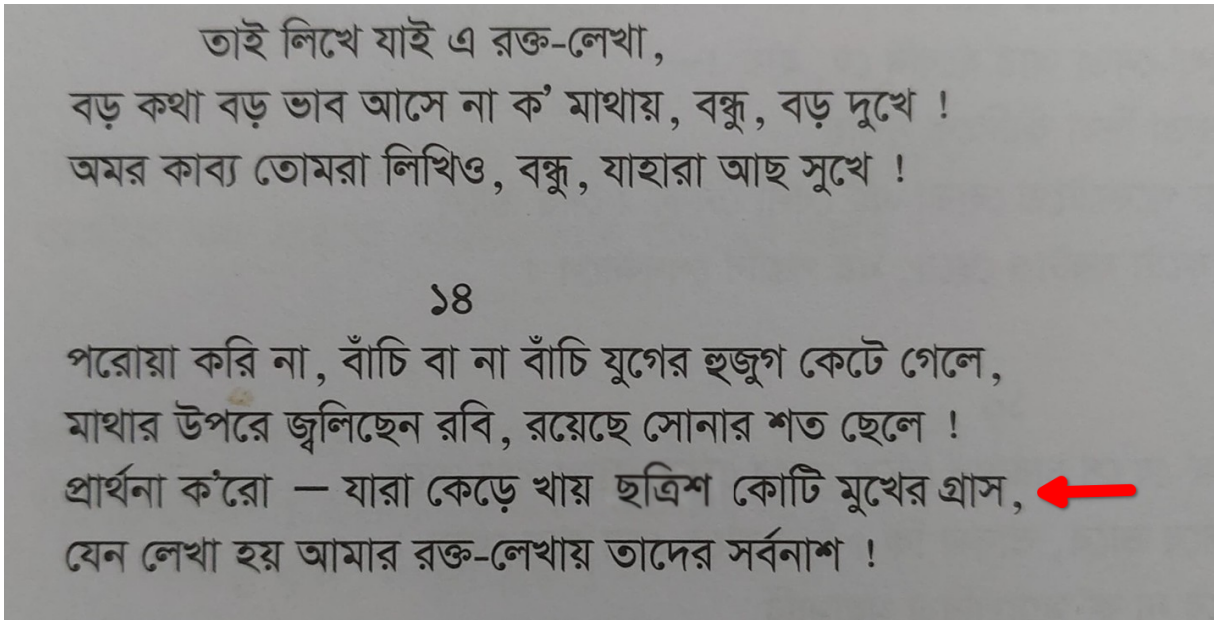
(c) Output Image

Figure 4.6: Example 4: Result of editing text book (black and white)



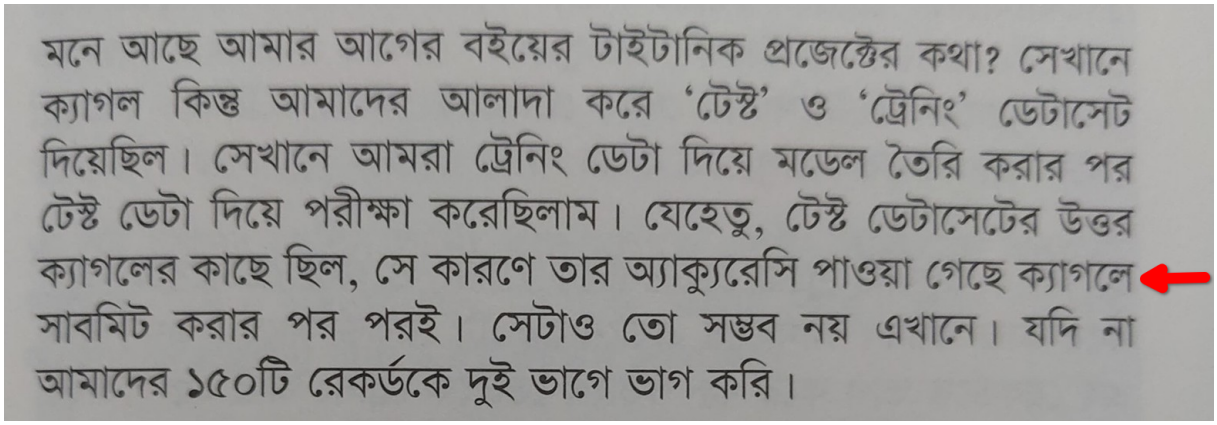
(a) Input Style Image

(b) Input Text



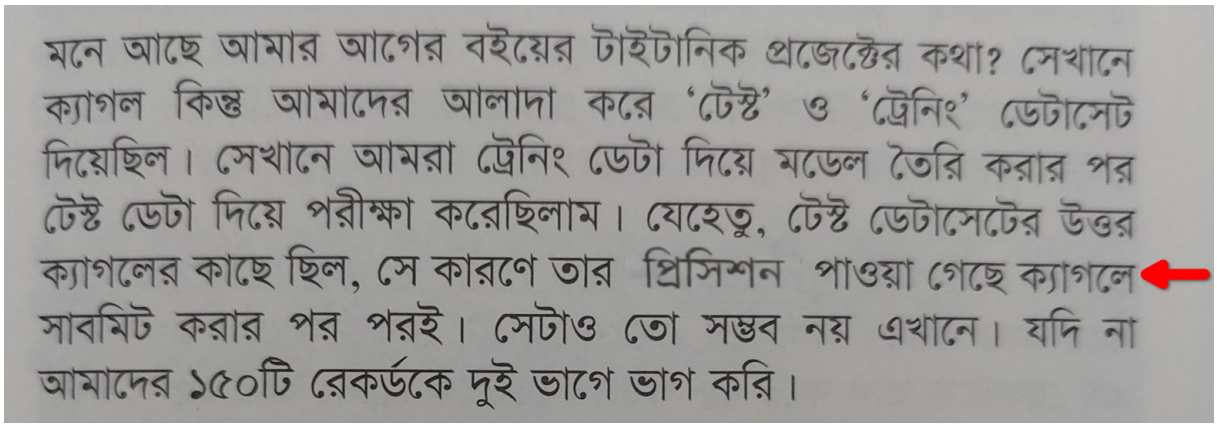
(c) Output Image

Figure 4.7: Example 5: Result of editing text book (grayscale)



(a) Input Style Image

(b) Input Text



(c) Output Image

Figure 4.8: Example 6: Result of editing text book (grayscale)

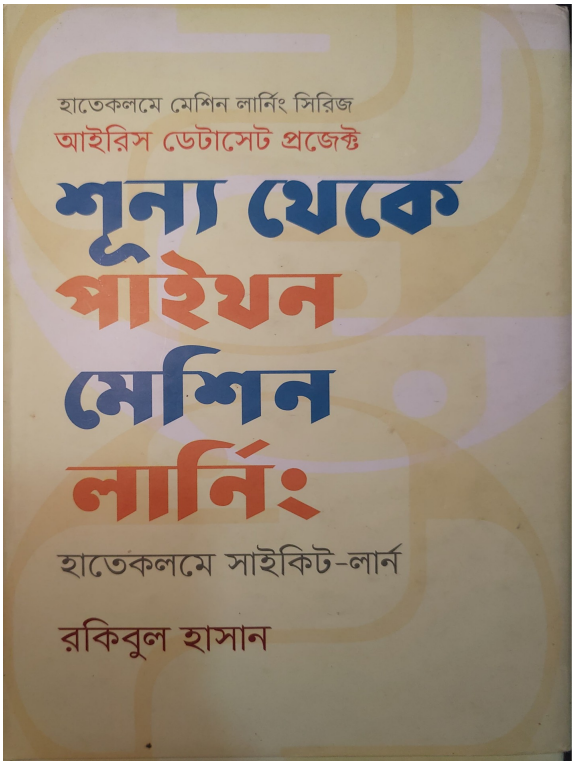


Figure 4.9: Input Style Image

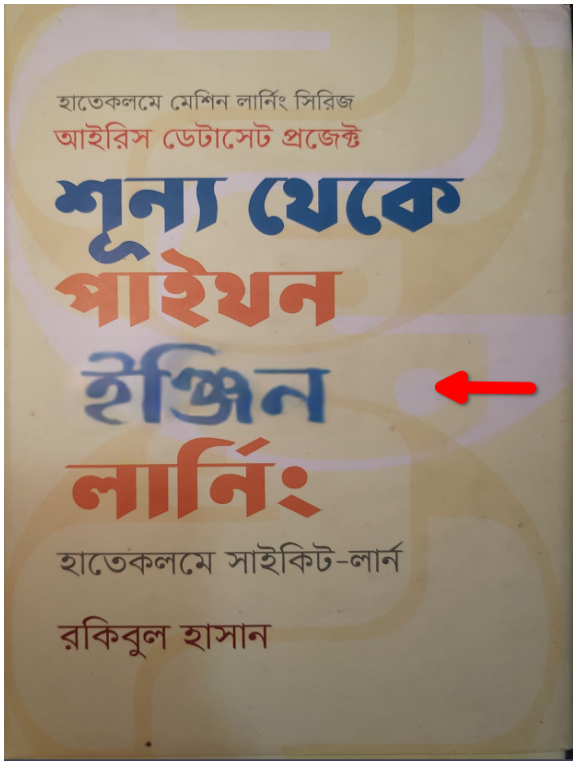


Figure 4.10: Output Image

Figure 4.11: Example 7: Result of editing scene image

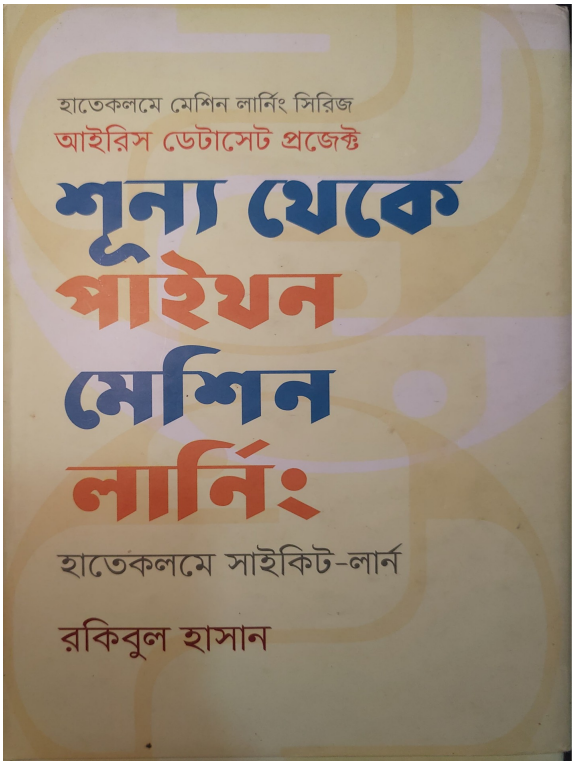


Figure 4.12: Input Style Image

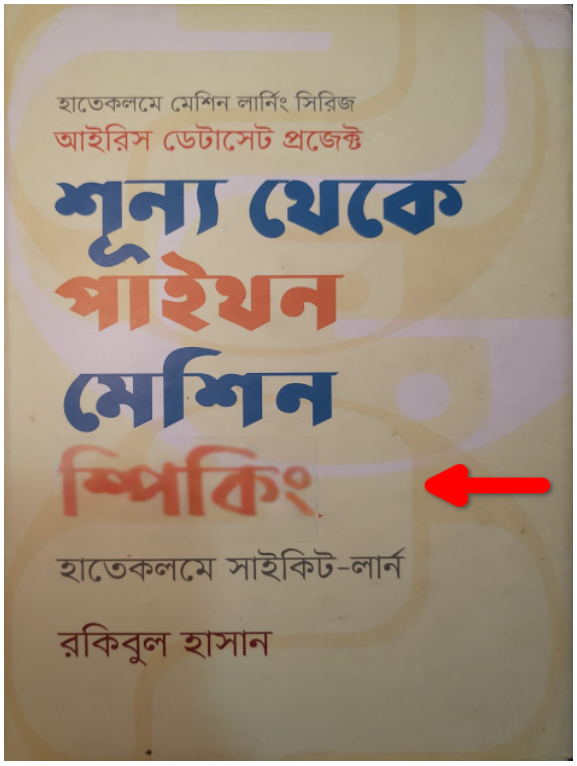


Figure 4.13: Output Image

Figure 4.14: Example 8: Result of editing scene image

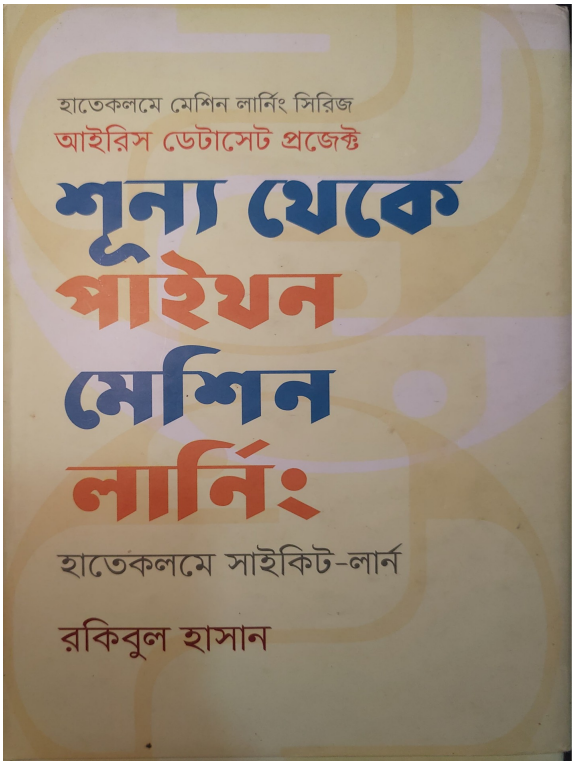


Figure 4.15: Input Style Image

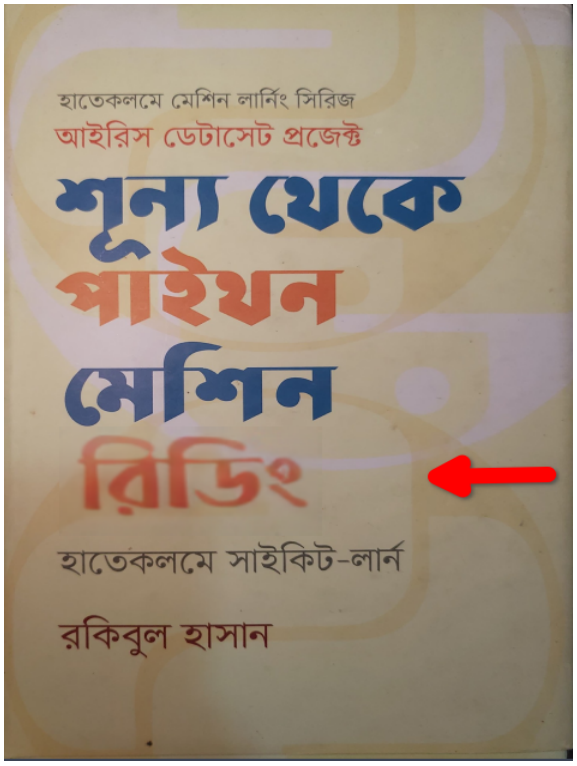


Figure 4.16: Output Image

Figure 4.17: Example 9: Result of editing scene image

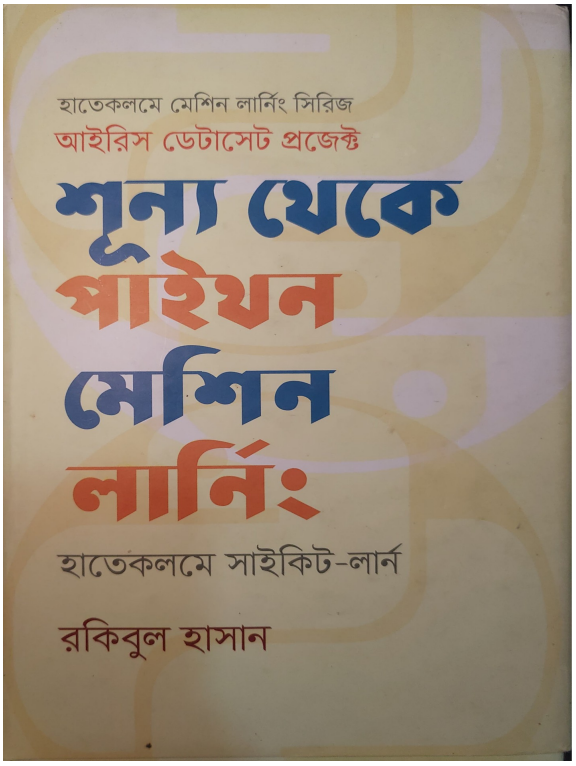


Figure 4.18: Input Style Image

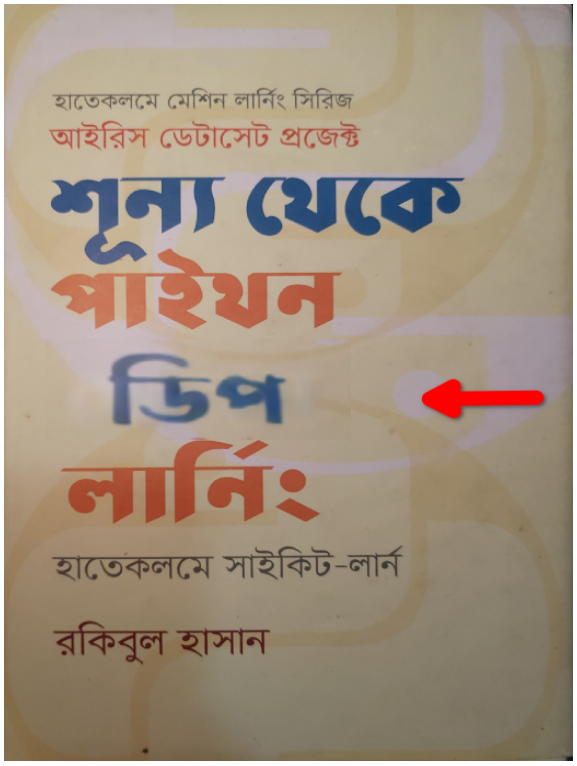


Figure 4.19: Output Image

Figure 4.20: Example 10: Result of editing scene image

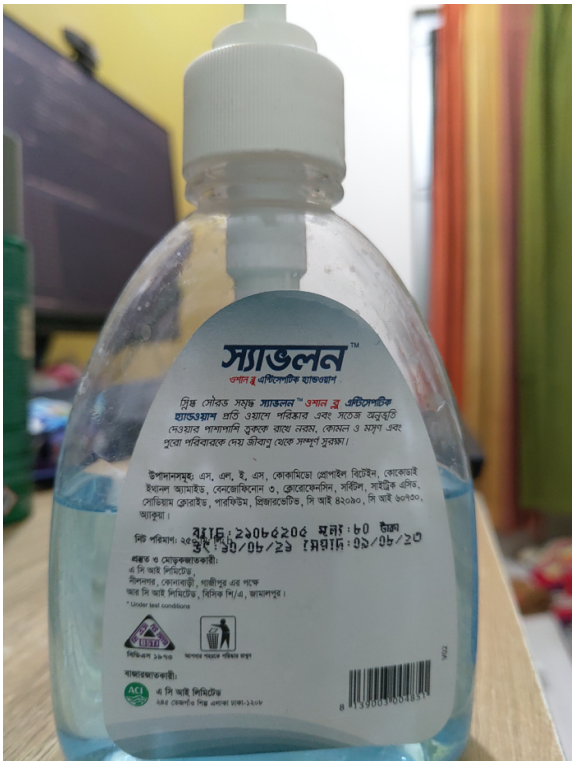


Figure 4.21: Input Style Image

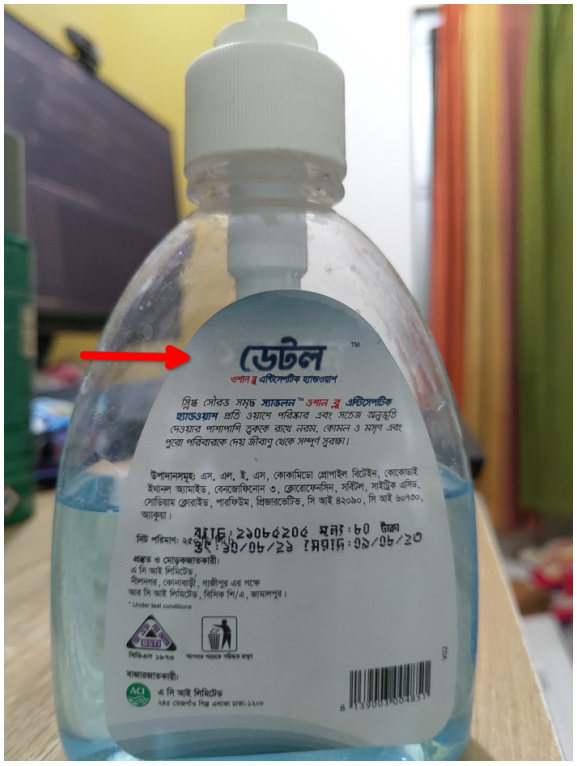


Figure 4.22: Output Image

Figure 4.23: Example 11: Result of editing scene image

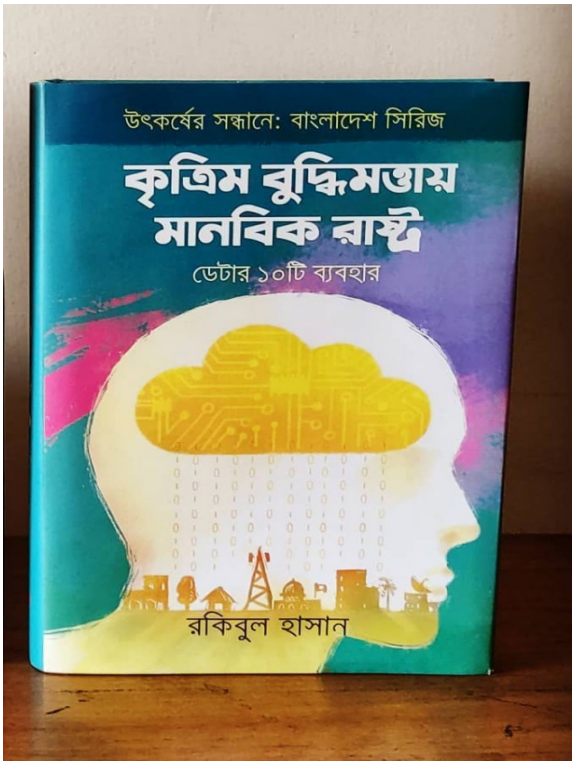


Figure 4.24: Input Style Image

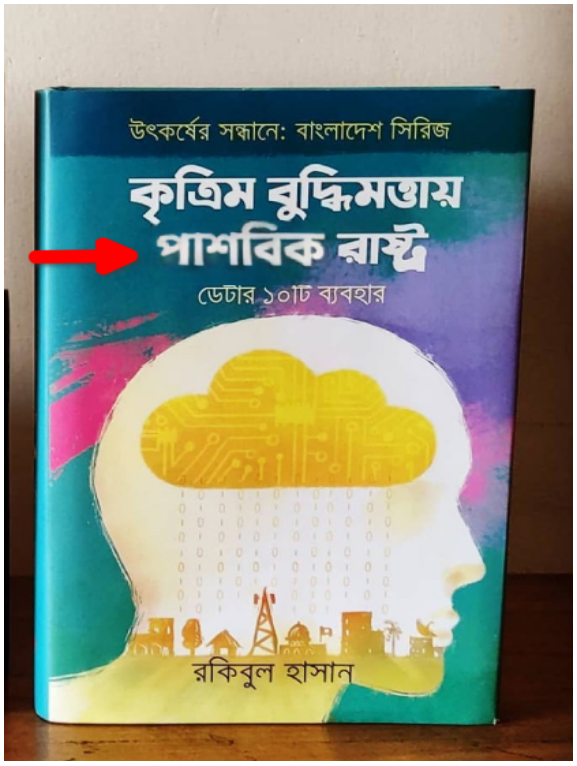


Figure 4.25: Output Image

Figure 4.26: Example 12: Result of editing scene image

Figure 4.27: Input Style Image

Figure 4.28: Output Image

Figure 4.29: Example 13: Result of editing scene image

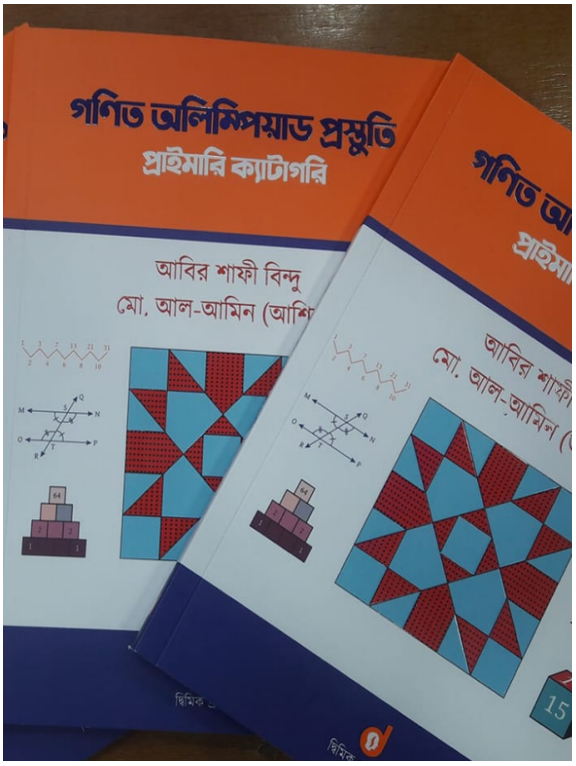


Figure 4.30: Input Style Image

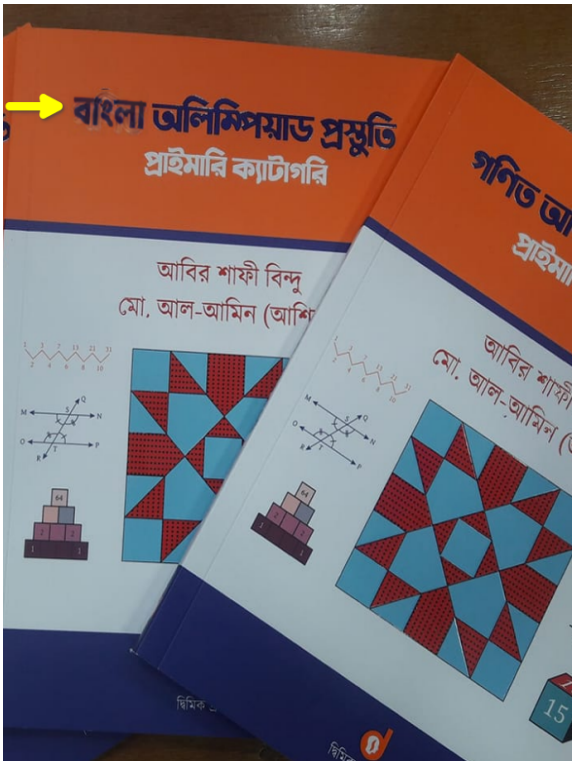


Figure 4.31: Output Image

Figure 4.32: Example 14: Result of editing scene image

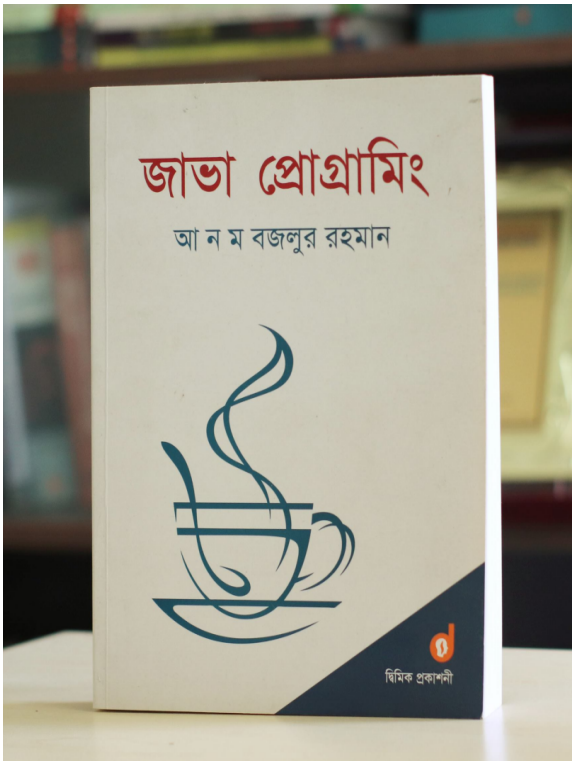


Figure 4.33: Input Style Image

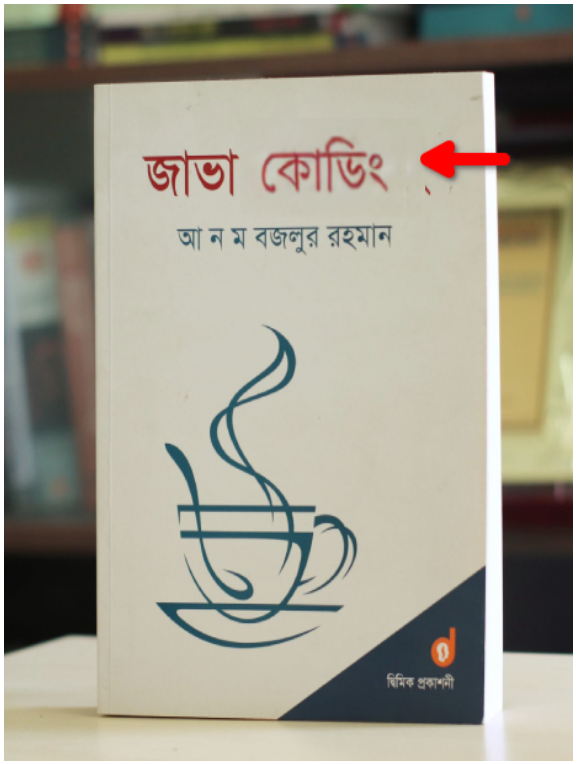


Figure 4.34: Output Image

Figure 4.35: Example 15: Result of editing scene image

Figure 4.36: Example 16: Results of editing on handwritten image data. (Note: We did not train our model with handwritten data)

Figure 4.37: Example 17: Results of editing on handwritten image data (Note: We did not train our model with handwritten data)

4.3 Failure Cases

Despite being capable of performing style retention from most of the scene images, our implemented model does not perform up to expectation while working with complex text structure. It also struggles to capture shadows. We accuse less variation in training data and limited number of Bengali fonts for these cases. Figure 4.38 shows some failure cases from our artificially generated dataset. The probable cause is that in the source text style image(left) the text color and the background are not clearly distinguishable from each other. Thus the model fails to extract the text from source text style image. This happens due to random selection of background image and text color and text stroke. Though this

examples is highly unlikely to appear in real-life cases, we can take measure so that such cases does not generated while generating dataset. Figure 4.41, Figure 4.44 shows some failure cases in Real-life scenarios. In the top most example, the model could not perform well due to the source text style being curve and rounded. This is because our dataset does not contain examples for this kind of images. In the second and third example, the result is not optimal because the source text style has odd border padding and shadows for depth. This failure cases occurs probably because of lack of training examples in the dataset. This problems could be solved by making dataset more robust and complex text style generation

(a) Real-life Text Style Image

(b) Input Text Image

(c) Output Image

Figure 4.38: The failure cases

Figure 4.39: Input Style Image

Figure 4.40: Output Image

Figure 4.41: Example 1: Failure case

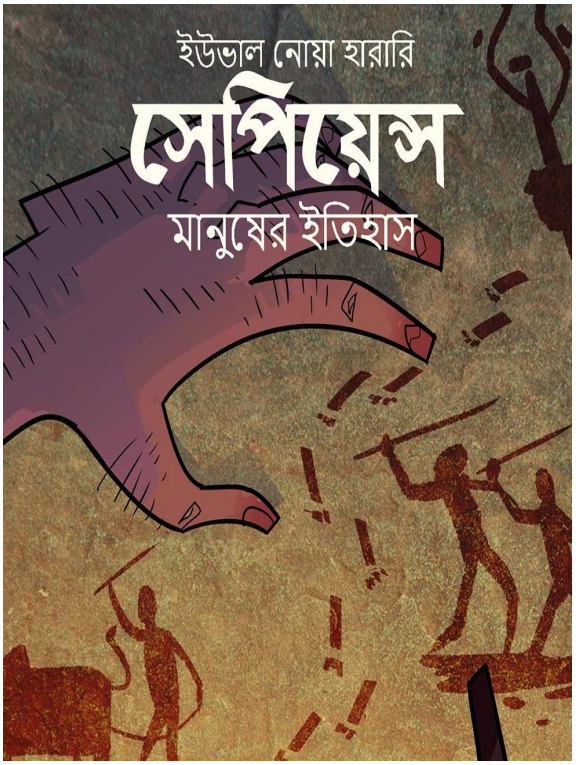


Figure 4.42: Input Style Image

Figure 4.43: Output Image

Figure 4.44: Example 2: Failure Case

4.4 Evaluation

Because the real dataset does not include ground truth, where we do not have a prediction of how new text will look with an example style, it's hard to establish adequate evaluation metrics other than visual assessment to qualitatively compare the performance. Similar works such as [2], [5], [3] used text recognition and text detection to evaluate the quality of the created result which was not possible for our work due to the unavailability of Bengali scene text recognition methods. For synthetic dataset, we follow the same quantitative evaluation metrics as prior scene text editing methods. [2], [5], [3]. We compare target style images with generated results using these measures

4.4.1 MSE

Mean Square Error also known as $l2$ error. The operations (difference and square) are bitwise (pixel by pixel), then it computes the mean, so it divides for the number of values (pixel). A lower l2 error means the results are similar to ground truth.

4.4.2 PSNR

It [32] computes the peak signal-to-noise ratio, between two pictures in decibels. This ratio is used to compare the quality of the original and compressed images. The higher the PSNR, the better the quality of the compressed, or reconstructed image. The mean-square error (MSE) and the peak signal-to-noise ratio (PSNR) are used to compare image compression quality. The MSE is a measure of the peak error between the compressed and original image, whereas the PSNR is a measure of the cumulative squared error. The smaller the MSE number, the smaller the error. The PSNR is calculated by first calculating the mean-squared error using the equation:

$$\boldsymbol{MSE} = \frac{\sum_{M,N} [I_1(m,n) - I_2(m,n)]^2}{M * N}$$

here, M and N are the number of rows and columns in the input images. Then we can computes the PSNR using the following equation:

$$\boldsymbol{PSNR} = 10log_{10}(\frac{R^2}{MSE})$$

here R is the maximum possible pixel value of the image. A higher PSNR score mean the results are similar to ground truth.

4.4.3 SSIM

Structural Similarity Index (SSIM) [32] is a perceptual metric that measures the loss of image quality. The structural similarity between two similar images is measured using SSIM. SSIM is based on observable structures in the image, unlike PSNR (Peak Signal-to-Noise Ratio). The results are more similar to ground truth if the SSIM score is higher. On several windows of a picture, the SSIM index is calculated. The distance between two windows x and y of common size NN is:

$$\boldsymbol{SSIM(x,y)} = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

- μ_x the average of x;
- μ_y is the average of y
- σ_x^2 the variance of x
- σ_y^2 the variance of y
- σ_{xy} the covariance of x and y;
- $c_1 = (k_1L)^2, c_2 = (k_2L)^2$ two variables to stabilize the division with weak denominator;
- L the dynamic range of the pixel-values
- $k_1 = 0.01$ and $k_2 = 0.03$ by default.

These metrics can only be used with a synthetic image set, where we can generate a corresponding target style image

4.4.4 Quantitative results on synthetic test dataset

For testing, we used 2000 images and another 2000 images for validation for quantitative analysis on the systhetic dataset. The quantitative results on the synthetic test dataset are shown in the table 4.1.

Table 4.1: Quantitative results on synthetic test dataset

Method	**MSE ↓**	**PSNR ↑**	**SSIM ↑**
SRNet (English) [3]	0.014	14.91	0.63
SWAPText (English) [5]	0.038	16.04	0.66
TSB (English) [2]	0.016	19.23	0.44
Ours (Bengali)	**0.014**	**19.91**	**0.69**

Chapter 5

Conclusion

5.1 Summary

We proposed a workaround and the only rich dataset for the Bengali text style retention task, replacing the text from scene text image while keeping the original text style untouched. To the best of our knowledge, this is the first work to editing scene text images in Bengali. We also make our dataset open to the researchers. Our implemented method comprises three main steps; (1) Transferring text style from source image text to target text, (2) Removing source text from source text image and restoring empty areas, and (3) Merging target text and source text image. To our best knowledge, this is the first work to edit text images in Bengali. Through this implementation, we have achieved notable results. Its other purposes like usage in the visual translation are yet to be explored. In the future, we hope to make this capable of editing in such scenarios where text structure is more complex and unusual. We expect to increase our fonts collection and do further experiments on cross-language text style retention as well.

5.1.1 Limitations

From our experiments, we have identified multiple limitations of our implemented model. This includes not being able to perform well when background image and text foreground color is hard to distinguish between. Another significant limitation is when source text structure is complex and have shadows, it performs poorly. Less frequent alphabets in the word lists are also hard to transform for the model.

5.2 Future Works

We hope to overcome our limitations by building a more robust dataset and improving model architecture. We are planning to adopt more sophisticated data generation technique to mimic more real-life scenarios. We would like to offer better evaluation metrics to measure the standard of editing Bengali scene text properly. So far, our implemented model works for Bengali to Bengali texts. We have plan to add English text feature by generating cross language image pair dataset and to make a platform to make real time image editing system.

References

[1] A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic data for text localisation in natural images," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[2] P. Krishnan, R. Kovvuri, G. Pang, B. Vassilev, and T. Hassner, "Textstylebrush: Transfer of text aesthetics from a single example," 2021.

[3] L. Wu, C. Zhang, J. Liu, J. Han, J. Liu, E. Ding, and X. Bai, "Editing text in the wild," in *Proceedings of the 27th ACM international conference on multimedia*, pp. 1500–1508, 2019.

[4] "libraqm library." https://en.wikipedia.org/wiki/Bengali_language. Accessed: 2021-06-25.

[5] Q. Yang, J. Huang, and W. Lin, "Swaptext: Image based texts transfer in scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14700–14709, 2020.

[6] P. Roy, S. Bhattacharya, S. Ghosh, and U. Pal, "Stefann: scene text editor using font adaptive neural network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13228–13237, 2020.

[7] "Convolutional neural networks (cnns / convnets)." https://cs231n.github.io/convolutional-networks/, Spring - 2021. Stanford Education Course.

[8] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *CoRR*, vol. abs/1906.02691, 2019.

[9] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[10] S. C. Alec Radford, Luke Metz, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2019.

[11] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CoRR*, vol. abs/1611.07004, 2016.

[12] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CoRR*, vol. abs/1611.07004, 2016.

[13] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *CoRR*, vol. abs/1703.10593, 2017.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[15] A. Mishra, "Iiit 5k-word (iiit 5k-word)," 2015.

[16] K. Wang, B. Babenko, and S. Belongie, "End-to-end scene text recognition," in *2011 International Conference on Computer Vision*, pp. 1457–1464, 2011.

[17] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. G. i Bigorda, S. R. Mestre, J. Mas, D. F. Mota, J. Almazán, and L.-P. de las Heras, "Icdar 2013 robust reading competition.," in *ICDAR*, pp. 1484–1493, IEEE Computer Society, 2013.

[18] X. Zhou, S. Zhou, C. Yao, Z. Cao, and Q. Yin, "ICDAR 2015 text reading in the wild competition," *CoRR*, vol. abs/1506.03184, 2015.

[19] A. Risnumawan, P. Shivakumara, C. S. Chan, and C. L. Tan, "A robust arbitrary text detection system for natural scene images," *Expert Systems with Applications*, vol. 41, no. 18, pp. 8027–8048, 2014.

[20] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM Trans. Graph.*, vol. 22, p. 313–318, July 2003.

[21] T. M. K. Lang, "Newsgroup 20 dataset," 1999.

[22] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015.

[23] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CoRR*, vol. abs/1411.4038, 2014.

[24] "Complex text layout." https://en.wikipedia.org/wiki/Complex_text_layout. Accessed: 2021-06-25.

[25] "Brahmic scripts." https://en.wikipedia.org/wiki/Brahmic_scripts. Accessed: 2021-06-25.

[26] "Pygame module." https://www.pygame.org/. Accessed: 2021-06-25.

[27] "libraqm library." https://github.com/HOST-Oman/libraqm. Accessed: 2021-06-25.

[28] "Pygame patched with libraqm." https://github.com/dipch/pygame_raqm_support. Accessed: 2021-06-25.

[29] A. S. A. Rabby, S. Haque, M. S. Islam, S. Abujar, and S. Hossain, *Ekush: A Multipurpose and Multitype Comprehensive Database for Online Off-Line Bangla Handwritten Characters*, pp. 149–158. 07 2019.

[30] M. Biswas, R. Islam, G. Shom, M. Shopon, N. Mohammed, S. Momen, and A. Abedin, "Banglalekha-isolated: A multi-purpose comprehensive dataset of handwritten bangla isolated characters," *Data in Brief*, vol. 12, pp. 103–107, 06 2017.

[31] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[32] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

Generated using Undegraduate Thesis LaTeX Template, Version 1.4. Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, Dhaka, Bangladesh.

This thesis was generated on Tuesday 8th August, 2023 at 2:11pm.