

Text Style Transfer - The Corpus EXChange

Project Team

Rayed Aslam	p14-6046
Waseef Ullah	p14-6030
Mohsin Mahmood	p14-6089

Session 2014-2018

Supervised by

Muhammad Amin

Co-Supervised by

Dr. Mohammad Nauman



Department of Computer Science

**National University of Computer and Emerging Sciences
Peshawar, Pakistan**

January, 2018

Student's Declaration

We declare that this project titled "*Text Style Transfer - The Corpus EXChange*", submitted as requirement for the award of degree of Bachelors in Computer Science, does not contain any material previously submitted for a degree in any university; and that to the best of our knowledge, it does not contain any materials previously published or written by another person except where due reference is made in the text.

We understand that the management of Department of Computer Science, National University of Computer and Emerging Sciences, has a zero tolerance policy towards plagiarism. Therefore, We, as authors of the above-mentioned thesis, solemnly declare that no portion of our thesis has been plagiarized and any material used in the thesis from other sources is properly referenced.

We further understand that if we are found guilty of any form of plagiarism in the thesis work even after graduation, the University reserves the right to revoke our BS degree.

Rayed Aslam

Signature: _____

Waseef Ullah

Signature: _____

Mohsin Mahmood

Signature: _____

Verified by Plagiarism Cell Officer

Dated:

Certificate of Approval



The Department of Computer Science, National University of Computer and Emerging Sciences, accepts this thesis titled *Text Style Transfer - The Corpus EXChange*, submitted by Rayed Aslam (p14-6046), Waseef Ullah (p14-6030), and Mohsin Mahmood (p14-6089), in its current form, and it is satisfying the dissertation requirements for the award of Bachelors Degree in Computer Science.

Supervisor

Muhammad Amin

Signature: _____

Co-Supervisor

Dr. Mohammad Nauman

Signature: _____

Shakir Ullah

FYP Coordinator

National University of Computer and Emerging Sciences, Peshawar

Dr. Omar Usman Khan

HoD of Department of Computer Science

National University of Computer and Emerging Sciences

Acknowledgements

We would like to express our deepest appreciation to all those who provided us the possibility to complete the Final Year Project I. A special gratitude we give to our final year project supervisor, Mr. Muhammad Amin, and co-supervisor Dr. Mohammad Nauman whose contribution in stimulating suggestions and encouragement, helped us to coordinate and whose have invested his full efforts in guiding our team in achieving the goal which is our project.

Furthermore we would also like to acknowledge with much appreciation the crucial role of the Head of Department Computer Sciences Dr. Omar Usman Khan, who gave the all required and the necessary suggestions, motivation and encouragement to complete the task “*Text Style Transfer - The Corpus EXChange*”. We have to appreciate the guidance given by Evaluators, other supervisors and as well as the whole panel especially in our project presentation that has improved our presentation skills thanks to their comment and advices. We would also like to expand our deepest gratitude to all those who have directly and indirectly especially our classmates and team members itself, have made valuable comment suggestions on this proposal which gave us an inspiration to improve our project.

Rayed Aslam

Waseef Ullah

Mohsin Mahmood

Abstract

It is a Neural Text Style Transfer using Natural Language Processing (NLP). We propose to explore the space of Natural Language Processing (NLP) by transferring writing style. Font design is an important area in digital art. However, designers have to design character one by one manually. With the help of this method;

- Designers finish the process faster,
- They only need to design a small set of characters and other characters will be generated automatically.

The main idea behind style transfer is to take two corpuses, say, a corpus of any author or person, and a corpus of another author or person, and use these to create a third corpus that combines the content of the former with the style of the later. We use neural network as the method so the inputs of the network are corpuses of an Urdu language. First corpus will be taken as Style, and the second corpus will be taken as Content and the output of the network is a file which contains the style of the first corpus and content of the second corpus.

Contents

1	Preliminaries and Introduction	1
1.1	Introduction	1
1.1.1	Overview	3
1.1.2	Problem at Hand	5
1.1.3	Software Framework	5
1.1.4	Related Work	6
2	Review of Literature	9
2.1	Artificial Neural Network	9
2.1.1	How does Neural Network learn?	9
2.1.2	A Working Example of Artificial Neural Network	10
2.1.3	Forward Pass	11
2.1.4	Calculating the Total Error	12
2.1.5	Backward Pass	13
2.2	Literature Review	16
3	Proposed Model	21
3.1	Model Analysis	21
3.1.1	Model	21
3.2	Data Set	22
3.2.1	Data Set Creation	22
3.2.2	Data Set Creation (Bitmaps)	22
3.3	Problems	23
3.3.1	Problems during data set creation	23
3.3.2	Solution of the Problems	24

3.4	Network Structure	25
3.5	Working	26
3.5.1	Convolutional Layer	26
3.5.2	Normalization	26
3.6	Maxpooling	28
3.7	Dropout	28
3.8	Sigmoid	29
3.9	Output	29
3.10	Results	30
4	Background	33
4.1	Corpus	33
4.1.1	Use Case	33
4.1.2	Concordance	33
4.1.3	Frequency Counts	34
4.1.4	Applications of Corpus	34
4.2	Natural Language Processing	34
4.2.1	Natural Language Processing (NLP)	34
4.2.2	Natural Language Analysis	34
4.2.3	Tokenization	35
4.2.4	Challenges in Tokenization	35
4.3	Stylistic Features	36
4.3.1	Sentence Length	36
4.3.2	Tone	37
4.3.3	Level of Formality	37
4.3.4	Vocabulary	39
4.3.5	Use of Passive Voice and Active Voice	39
4.3.6	Sentiments	40
5	Conclusions and Future Work	41
5.1	Conclusions	41
5.2	Future Work	42

List of Figures

1.1	Image Style Transfer ¹⁰	2
1.2	Text Style Transfer	3
1.3	Text Style Transfer Example	4
1.4	Text Style Transfer Flow Diagram	4
1.5	Overview	5
2.1	Artificial Neural Network	10
2.2	Artificial Neural Network with some Weights(w)	11
2.3	Gradient Descent	13
2.4	Back propagation	14
3.1	Model structure ²⁷	21
3.2	Model overview	22
3.3	Degree of 1 Urdu Characters	23
3.4	Degree of 2 Urdu Characters	23
3.5	Data Set Creation using Bitmaps ²⁷	24
3.6	Issues faced during character set generation	24
3.7	Solution of the problem (shown in figure 3.6)	25
3.8	Solution of the problem (shown in figure 3.6)	26
3.9	Network Structure used to get accurate results.	27
3.10	Convolutional Layer (size = 64x64, filters = 8)	27
3.11	Normalization	28
3.12	Maxpooling	28
3.13	Dropout example	29
3.14	Sigmoid function graph	29
3.15	Frame generated after 4000 iterations	30

3.16 Combined Loss Graph	31
4.1 Different Stages involved in Text Style Transfer	36

List of Tables

2.1	Literature Review (1/4)	17
2.2	Literature Review (2/4)	18
2.3	Literature Review (3/4)	19
2.4	Literature Review (4/4)	20

Chapter 1

Preliminaries and Introduction

1.1 Introduction

It is a Neural Text Style Transfer using Natural Language Processing (NLP). We propose to explore the space of Natural Language Processing (NLP) by transferring writing style. One of the most interesting discussions today around within machine learning is how it might impact and shape our cultural and artistic production in the next decades. Most of the previous work in this field focuses on controlling content instead of attempting to learn stylistic features. We aim to bridge this gap with an approach based on neural text style transfer networks, where the strategy often includes a set of layers designated to learn content (i.e. objects in the Corpus and larger structure trends) and another set of layers designated to learn style (i.e. the vector values for individual letter). Central to this discussion is the recent advances in text style transfer using deep learning.

This work is also inspired from related problems such as author disambiguation and neural machine translation. We can use these fields to explore a variety of problem formulations and approaches in NLP to lead to a more general way of style transfer that diverges from recent work that focuses on manually extracted stylistic features. Our main challenge will be separating stylistic from content-based features.

The main idea behind style transfer is to take two corpuses, say, a corpus of any author or person, and a corpus of another author or person, and use these to create a third corpus that combines the content of the former with the style of the later. The figure below (see

figure 1.2) shows an example using one corpus of the author and the corpus of famous author Shakespeare. The working or idea of this project figure 1.1 is that you can simply



Figure 1.1: Image Style Transfer¹⁰

give the machine some input, and desired style, and you will get the desired input merged in the given style. The image example given above is in the context of the image,¹⁰ but in our context i.e. Text, the working is same but the source of content and the source of the style will be different. In the figure 1.2, we have Source Text **Content** of Shakespeare and Source Text **Style**, we merge the Content of the source and merge it with the Style of the source so that the Output Text will be **Content + Style**. In the figure 1.3, we have a simple example of Text Style Transfer. According to the academic literature, text style transfer is defined as follow: given two corpuses on the input, synthesize a third text file that has the semantic content of the first corpus and the texture/style of the second. To work properly we need a way to:

1. Determine the content and the style of any corpus (content/style extractor) and then
2. Merge some arbitrary content with another arbitrary style (merger).

The figure 1.4 shows the work flow of the project.

This definition of style transfer might seem a bit imprecise. Put another way, the central problem of style transfer revolves around our ability to come up with a clear way of computing the "content" of a corpus as distinct from computing the "style" of the 2nd corpus. Before deep learning arrived at the scene, researchers had been handcrafting



Figure 1.2: Text Style Transfer

methods to extract the content and texture of images, merge them and see if the results were interesting or garbage. Even in today's research of style transfer using deep learning there are high impact papers proposing new ways of using a neural network to extract the content, extract style or combine them. Despite not having an exact idea of what content and style/texture are, we can develop a general idea of what we should expect in a good result to help narrow down our understanding of the problem.

1.1.1 Overview

Font design is an important area in digital art. However, designers have to design character one by one manually. With the help of this method;

- Designers finish the process faster,
- They only need to design a small set of characters and other characters will be generated automatically.

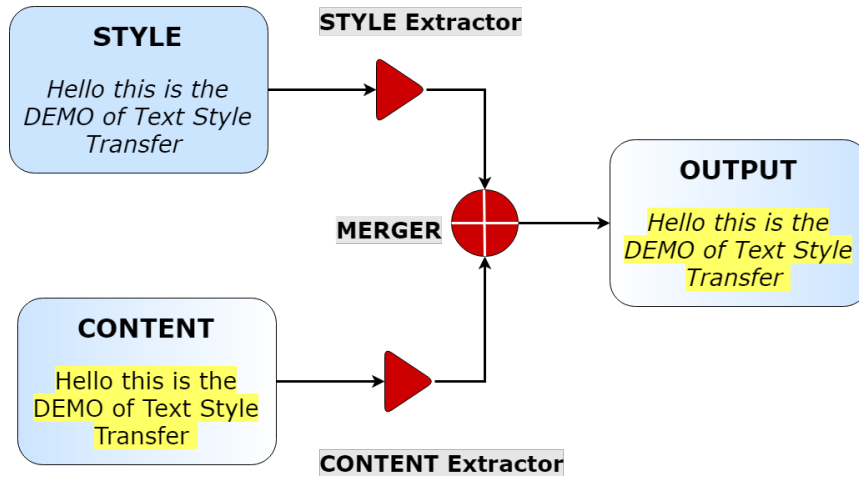


Figure 1.3: Text Style Transfer Example

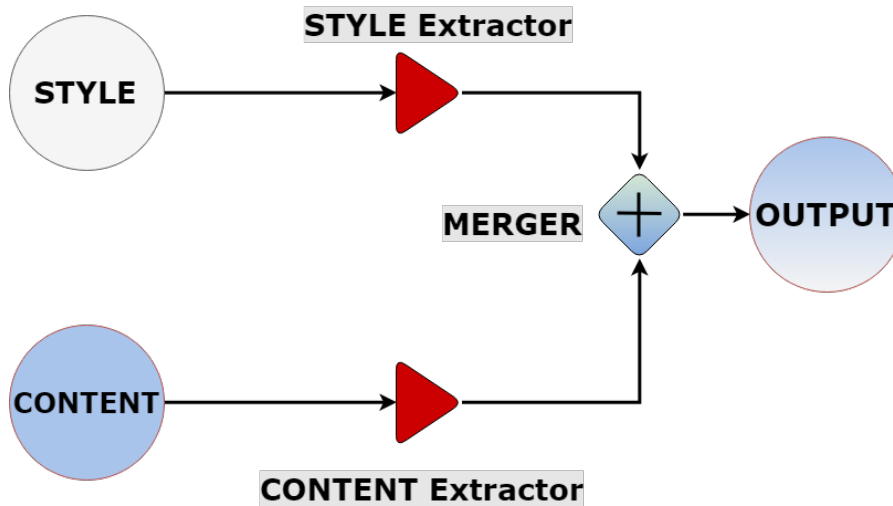


Figure 1.4: Text Style Transfer Flow Diagram

Creating font is hard, and then creating an Urdu font is an even harder.

- Designers will need to design unique looks for every Urdu character,
- Every character making sequence of more than 1,
- A effort that could take years to complete.

Designer creates a subset of characters, then let the machine generate the rest of the fonts for him/her.

- A neural network is trained to approximate the transformation in between two fonts given a subset of pairs of examples.

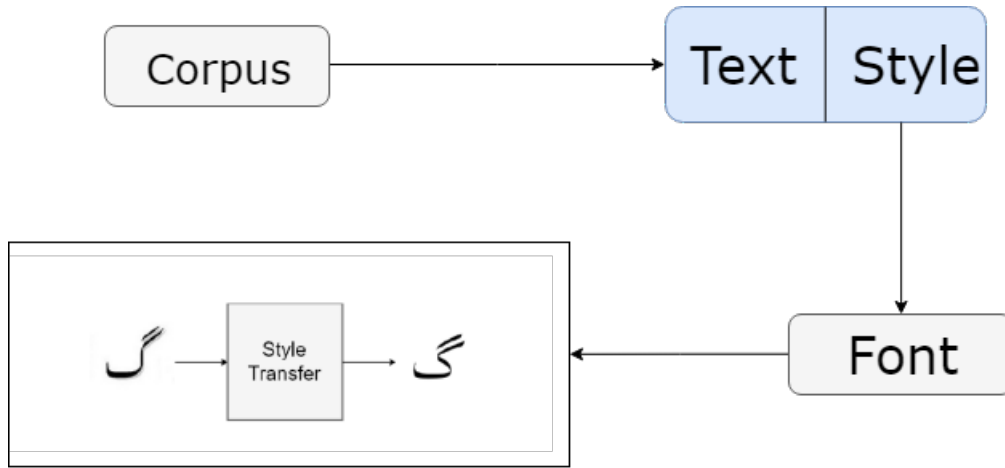


Figure 1.5: Overview

- Once the learning is finished, it can be used to infer the shape for the rest of characters.

1.1.2 Problem at Hand

There is no Data set available for the Urdu language, to put into Neural Network for training and testing purposes.

1.1.3 Software Framework

The code is developed in python. Python has a lot of libraries to support development of neural network. Tensorflow: Tensorflow is an open-source library for machine intelligence. The library is originally developed by researchers and engineers of Google Brain Team, and now is open to all developers. Tensorflow uses data ow graphs. Mathematic operations are presented as node in the graph, while data arrays are edges (Tensor) between the nodes. Tensorflow allows us to build neural network fast and easy to run on CPU and GPU. Numpy: Numpy is a fundamental package for scientific computing with Python. The library allows us to represent image data in multi-dimension vector and compute easily. We present all our data as Numpy multi-dimension vector. The output of the network is also a vector. Python functions help us save them as images. Pillow: Pillow

is an open-source python imaging library developed by Alex Clark and other contributors. The library allows us to present image in python code easily. Currently most font libraries are saved as TTF file. Because the input of our network is RGB image, we need to use Pillow library to save all training characters as PNG file. Other libraries we use are Commentjson, Scipy and Matplotlib. Commentjson helps us to write JSON file. Scipy is a basic library for Numpy. And Matplotlib is designed for generating plot.

1.1.4 Related Work

Researchers and scientists have explored some methods to generate fonts automatically. Some work focus on decomposing Urdu character into different components to allow designer only design components. Then software assemble components together to get characters.¹⁷ and²⁰ both belong to this kind. There are two dis-advantages of these methods. The first one is that it is still a difficult mission to design components because Urdu characters are too complex, for example in Urdu we have issue of joining different characters when we increase the degree or length. The other one is that result is not good enough because same component may different in different Urdu characters. Another kind of method is as,²⁶¹⁸ and.²³ The methods are much more efficient than last one and result is much better. Users are required to design models for each characters in training set and provide a large set of characters in different fonts. Then software generate all the rest characters automatically. The method is close to practical use. However, because the provided models have to be accurate and complex, it still need a lot of manual operation and professional training.¹⁹ provides an interesting method. The generated characters are similar with ground truth. However, users have to decompose training characters manually, which makes the method less efficient.

²⁷ project provides a good direction for Urdu font transfer. The method uses neural network to generate characters. The method is easy to use because designers just need to provide a small character image set. Then the network generates all the rest characters. However, the training set contains 3,000 characters, which is still too large. A professional designer will take about couple of months to design 3,000 characters of more than degree 2 or more. And according to the result provided, the generated characters are still

not good enough.¹ uses similar idea on English and²⁷ also uses the similar idea. The method works well because there are only 62 different characters in English, Chinese has tens of thousands of characters while Urdu has tens of thousands of words in different degrees or length and combination. In this paper, we also use neural network to generate characters. The difference is that²⁷ project uses fully connected network with 3 layers, while our method uses deep fully convolutional network. There are two advantage of our method, the first one is that convolutional network is more suited to image and Convolutional network uses context information to predict result, which is important for image. And because convolutional network always has less parameters than fully connected network, the training process need less time. Another advantage is our network is deeper than the network used by.²⁷ Deeper network always catch features of training data better and thus has better result. However, deeper network has over-fitting problem. We avoid this problem by adding l2 regularization part in loss function. And according to the result, our method indeed generates better result. Deep learning develops fast these years. There are many kinds of deep neural network. With some experiences, we finally use FCN as our network structure.

Chapter 2

Review of Literature

2.1 Artificial Neural Network

Artificial neural network contains a lot of densely interconnected nodes called neurons or perceptrons, so you can get it to learn things, recognize patterns, and make decisions in a human like way.

Artificial neural network consists of **input units** (i), which are designed to receive information from the outside world, there are **output units** (o), which responds to the information it learned and in between them is one or more layers of **hidden units** (h), which performs almost all the work of artificial neural network.

The connection between one unit and another are represented by a number called a **weight**, which can be either positive or negative, the higher the weight, the more influence one unit has on another.

2.1.1 How does Neural Network learn?

Information flows through a neural network in two ways. When it's learning (training) or operating normally (after being trained), patterns of information are fed into the network via the input units, which trigger the layers of hidden units, and these in turn arrive at the output units. This is called **feed forward** or **forward pass**.

Each unit receives inputs from the units to its left, and the inputs are multiplied by the

weights of the connections they travel along. Every unit adds up all the inputs it receives, The sum goes through an activation function (which can be of any type), and triggers the units it's connected to.

Neural networks learn things in exactly the same way, typically by a feedback process called **back propagation**. This involves comparing the output a network produces with the output it was meant to produce, and using the difference between them to modify the weights of the connections between the units in the network, working from the output units (o) through the hidden units (h) to the input units (i) going backward.

2.1.2 A Working Example of Artificial Neural Network

For this example, we are going to use a neural network with two inputs (i_1, i_2), two hidden neurons (h_1, h_2), two output neurons (o_1, o_2). Additionally, the hidden and output neurons will include a bias (b_1, b_2).

figure 2.1 shows the Artificial Neural Network (ANN) where i_1 and i_2 represents the

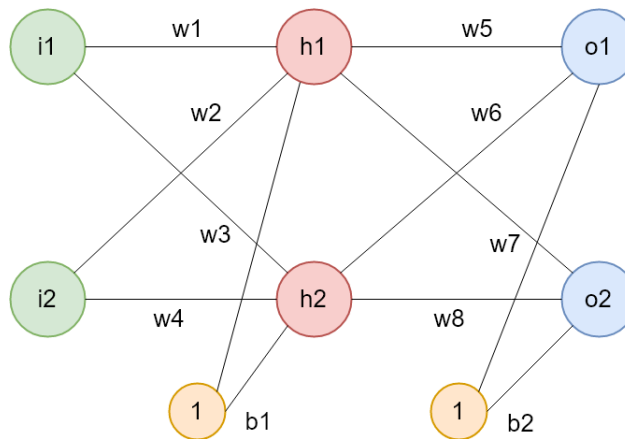


Figure 2.1: Artificial Neural Network

neurons in input layer, h_1 and h_2 represents the neurons in hidden layer and finally, o_1 and o_2 represents the neurons in output layer of the neural network.

In order to have some numbers to work with, here are the initial weights (w)(shown in

figure 2.2), the biases (b), and training inputs (i)/outputs (o):

Where $w_1 = 0.15, w_2 = 0.20, w_3 = 0.25, w_4 = 0.30, w_5 = 0.40, w_6 = 0.45, w_7 = 0.50$ and

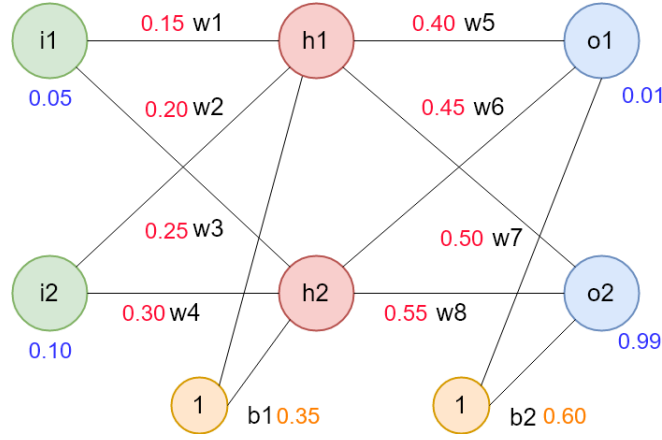


Figure 2.2: Artificial Neural Network with some Weights(w)

$w_8 = 0.55$. Our target outputs are 0.01 at o_1 and 0.99 at o_2 , given the inputs $i_1 = 0.05$ and $i_2 = 0.10$.

2.1.3 Forward Pass

To begin, let's see what the neural network currently predicts given the weights and biases above and inputs of 0.05 and 0.10. To do this we will feed those inputs forward through the network.

Here is how we calculate the total net input for h_1 :

$$net_{h1} = i_1 * w_1 + i_2 * w_2 + b_1 * 1$$

$$net_{h1} = 0.05 * 0.15 + 0.1 * 0.2 + 0.35 * 1 = 0.3775$$

We then squish it using the logistic function to get the output of h_1 :

$$out_{h1} = \frac{1}{1 + e^{-net_{h1}}} = \frac{1}{1 + e^{-0.3775}} = 0.593269992$$

Carrying out the same process for h_2 we get:

$$out_{h2} = 0.596884378$$

We repeat this process for the output layer neurons, using the output from the hidden layer neurons as inputs.

Here is the output for o_1 :

$$net_{o1} = out_{h1} * w_5 + out_{h2} * w_6 + b_2 * 1$$

$$net_{o1} = 0.593269992 * 0.4 + 0.596884378 * 0.45 + 0.6 * 1 = 1.105905967$$

We then squish it using the logistic function to get the output of o_1 :

$$out_{o1} = \frac{1}{1 + e^{-net_{h1}}} = \frac{1}{1 + e^{-1.105905967}} = 0.75136507$$

And carrying out the same process for o_2 we get:

$$out_{o2} = 0.772928465$$

2.1.4 Calculating the Total Error

We can now calculate the error for each output neuron using the squared error function and sum them to get the total error.

Here is the error for o_1 :

$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$

Repeating this process for o_2 (remembering that the target is 0.99) we get:

$$E_{o2} = 0.023560026$$

The total error for the neural network is the sum of these errors:

$$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$

2.1.5 Backward Pass

The goal of back propagation is to update each of the weight in the network so that they cause the actual output to be closer to the target output.

Here we will use the **Gradient Descent**, which is an optimization algorithm used to find the values of parameters of a function that minimizes a cost function. figure 2.3 is the

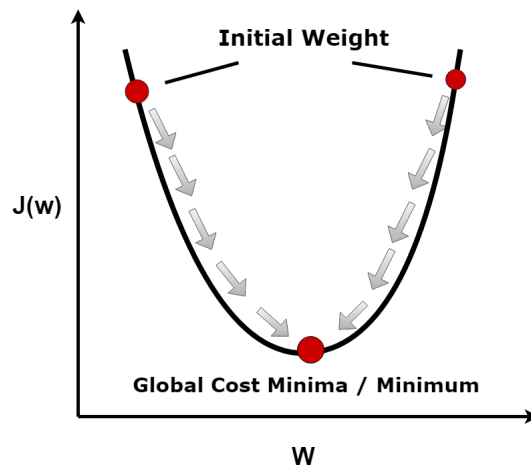


Figure 2.3: Gradient Descent

plot of the **cost function** or **Gradient Descent**. A position on the graph is the cost of the initial values of the weights. The bottom of the graph is the cost of the best set of weights, the minimum of the function.

The goal is to continue to try modifying the values for the weights, evaluate their cost and select new values for the weights that have a lower cost.

Repeating this process enough times will lead to the bottom of the graph and you will know the values of the weights that result in the minimum cost.

Lets consider w_5 . We want to know how much a change in w_5 affects the total error $\frac{\partial E_{total}}{\partial w_5}$.

By applying the **chain rule**:

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

We need to figure out each piece of this equation. Above figure 2.4 shows how much

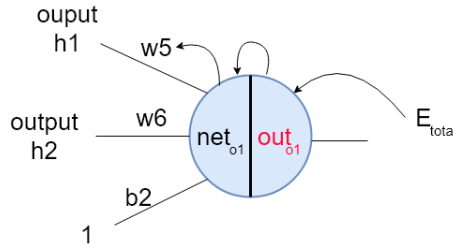


Figure 2.4: Back propagation

does the total error change with respect to the output?

$$E_{total} = \frac{1}{2}(target_{o1} - out_{o1})^2 + \frac{1}{2}(target_{o2} - out_{o2})^2$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1})$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(0.01 - 0.75136507)$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = 0.74136507$$

How much does the output of o_1 change with respect to its total net input?

$$out_{o1} = \frac{1}{1 + e^{-net_{o1}}}$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1})$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = 0.75136507(1 - 0.75136507)$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = 0.186815602$$

Finally, how much does the total net input of o_1 change with respect to w_5 ?

$$net_{o1} = out_{h1} * w_5 + out_{h2} * w_6 + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial w_5} = out_{h1}$$

$$\frac{\partial net_{o1}}{\partial w_5} = 0.593269992$$

Putting it all together:

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.74136507 * 0.186815602 * 0.593269992$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.082167041$$

To decrease the error, we then subtract this value from the current weight (optionally multiplied by some learning rate, alpha, which we will set to 0.5):

$$w_5^+ = w_5 - \alpha * \frac{\partial E_{total}}{\partial w_5}$$

$$w_5^+ = 0.4 - 0.5 * 0.082167041$$

$$w_5^+ = 0.35891648$$

We repeat this process for all the weights in the neural network.

The new weights after the first round are:

$$w_6^+ = 0.408666186$$

$$w_7^+ = 0.511301270$$

$$w_8^+ = 0.561370121$$

$$w_1^+ = 0.149780716$$

$$w_2^+ = 0.19956143$$

$$w_3^+ = 0.24975114$$

$$w_4^+ = 0.29950229$$

When we fed forward the 0.05 and 0.1 inputs originally, the error on the network was 0.298371109. After this first round of back propagation, the total error is now down to 0.291027924.

It might not seem like much, but after repeating this process 10,000 times, the error converges to zero.

After 10,000 iterations,

when we feed forward 0.05 and 0.1, the two outputs neurons generate 0.015912196 (vs 0.01 target) and 0.984065734 (vs 0.99 target).

2.2 Literature Review

Title	Author	Year	Problem	Methods	Results
¹⁰ A Neural Algorithm of Artistic Style	LA Gatys	2015	Creating artistic images by separating and combining content and style of images	VGG-Network, CNN	Creates artistic images of high perceptual quality to separate and recombine content and style of arbitrary images
⁹ Style Transfer in Text: Exploration and Evaluation	Zhenxin Fu	2015	Style transfer with non-parallel corpora	Seq2Seq Model	Learn style transfer from non-parallel data, and the proposed content preservation evaluation metric is highly correlated to human judgment
²² Deep Photo Style Transfer	F Luan	2017	Structure Preservation and Semantic Accuracy	Photo-Realism Regularization and Semantic Segmentation	They Prevented painting-like effects from happening and the image looked visually more satisfying (no blurred parts)
² Zero-Shot Style Transfer in Text Using Recurrent Neural Networks	K Carlson	2017	Stylistic Paraphrasing	Zero-shot Stylistic Paraphrasing	BLEU (BiLingual Evaluation Understudy) score was 20.09% and according to further evaluation the PINC score was 74.04%
⁷ Applying Artistic Style Transfer to Natural Language	T Edirisooriya	2017	Apply the methodology of visual artistic style transfer to the realm of Natural language	Text-Classification, Seq2Seq Style Transfer	Combines features from two images: one that has lower-level features (color and texture), and the other captures content (say buildings in the photograph)

Table 2.1: Literature Review (1/4)

Title	Author	Year	Problem	Methods	Results
¹⁴ Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization	X Huang	2017	Style Transfer in Real-time	Adaptive Instance Normalization	They compare it with other methods on the basis of quantitative, qualitative and speed analysis and the results were good in almost all aspects except for the speed.
²⁵ Style Transfer from Non-Parallel Text by Cross-Alignment	T Shen	2017	Style Transfer in Non-Parallel Texts	Cross-aligned auto-encoder	Sentiment accuracy of transferred sentences, as measured by a pre-trained classifier, Accuracy for Cross-aligned auto-encoder is 78.4%
¹⁶ Stylistic Transfer in Natural Language Generation Systems Using Recurrent Neural Networks	J Kabbara	2016	Style Transfer in NLG systems	LSTM-based RNN model based on the encoder-decoder	The results are still pending in aspects of (soundness, coherence, effectiveness). They want to evaluate it by humans.
¹¹ Image Style Transfer Using Convolutional Neural Networks	LA Gatys	2016	Find image representations that independently model variations in the semantic image content and the style in which it is presented	Deep Convolutional Neural Networks	Can produce new, perceptually meaningful images, and to demonstrate this finding, they generate images that mix the content and style representation from two different source images.
²⁴ Generative Adversarial Text to Image Synthesis	Scott Reed	2016	Text feature representation and using these features to synthesize an image	Natural Language Representation and image synthesis	They Generated the content images with multiple objects and were able to give it variable backgrounds.

Table 2.2: Literature Review (2/4)

Title	Author	Year	Problem	Methods	Results
²⁸ Automatic Semantic Style Transfer using Deep Convolutional Neural Networks and Soft Masks	Huihuang Zhao, Paul L. Rosin, Yu-Kun Lai	2017	Automatic image synthesis method to transfer the style of an example image to a content image	19-layer VGG-Network	VGG Network gives more than 70 accuracy
⁸ Style Transfer via Texture Synthesis	Michael Elad, Peyman Milanfar	2017	Process of migrating a style from one image (the Style-Image) to another (the Content-Image).	Convolutional Neural Networks (CNN)	This paper puts forward a novel texture synthesis-based solution to the style transfer problem.
³ StyleBank: An Explicit Representation for Neural Image Style Transfer	Dongdong Chen, Nenghai Yu, Lu Yuan, Jing Liao, Gang Hua	2017	Multiple convolution filter banks and each filter bank explicitly represents one style, for neural image style transfer.	16-layer VGG-Network.	VGG network gives around 70 accuracy.
¹⁵ Neural Style Representations of Fine Art	Jeremiah Johnson	2017	Style representation of a digitized image of an artwork	Fully - Connected Linear Classifier	Fully-Connected Linear Classifier yielded an accuracy of 13.23
⁴ Fast Patch-based Style Transfer of Arbitrary Style	Tian Qi Chen, Mark Schmidt	2017	An approach that is feed forward, fast, and adaptable to any style image.	Concatenate content and style information and for every content patch, swap it with best matching style patch (style swap).	Computation time on 500x300 images (100 iterations), Style Swap - Time=4.66s VS Gatys et al - Time=10.04s.

Table 2.3: Literature Review (3/4)

Title	Author	Year	Problem	Methods	Results
²⁷ Chinese Font Style Transfer with Neural Network	Hanyu Xue	2017	Generating any Chinese character after training neural network on a small subset of Chinese characters.	16-layer VGG-Network.	Accuracy starts from about 0.82 and ends at about 0.89 after 12,000 iterations for set size = 3000.
¹ Learning Typographic Style	Shumeet Baluja	2017	Analyzing small subset of letters and performing Discrimination and Generation task.	Fully Connected for Discrimination and CNN for Generation	Discrimination = 92.1
¹² Controlling Perceptual Factors in Neural Style Transfer	Leon A. Gatys, Alexander S. Ecker	2017	To enable decomposition of style from many sources to generate a new style	Patch based texture synthesis, color histogram matching and luminance only transfer technique	Able to preserve color of image and access to different factors (space and scale) of image.
¹³ Characterizing and Improving Stability in Neural Style Transfer	Agrim Gupta, Justin Johnson, Alexandre Alahi	2017	Removal of Real time flickering when style is applied to a video.	Recurrent Convolutional Network	Able to stable the method and remove flickering for Real Time Video Style Transfer
²¹ Decoder Network over Lightweight Reconstructed Feature for Fast Semantic Style Transfer	Ming Lu, Hao Zhao, Anbang Yao, Feng Xu	2017	Decode the feature map by an inverse network to image space.	Patch swap used to express style transfer.	Combine several styles with a single network which demonstrates competitive results much faster.

Table 2.4: Literature Review (4/4)

Chapter 3

Proposed Model

3.1 Model Analysis

3.1.1 Model

We use neural network as the method. The input of the network is a character image in an existing font. And the output of the network is a character image in target font. To use the network, designers are expected to provide a training data set and an existing font as reference. Each data in the data set is a pair as [character image in existing font, character image in target font]. The data will be used to train network. After training, given a character image in existing font, the network will generate the character image in target font.

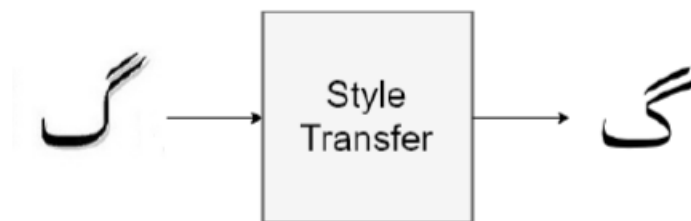


Figure 3.1: Model structure²⁷

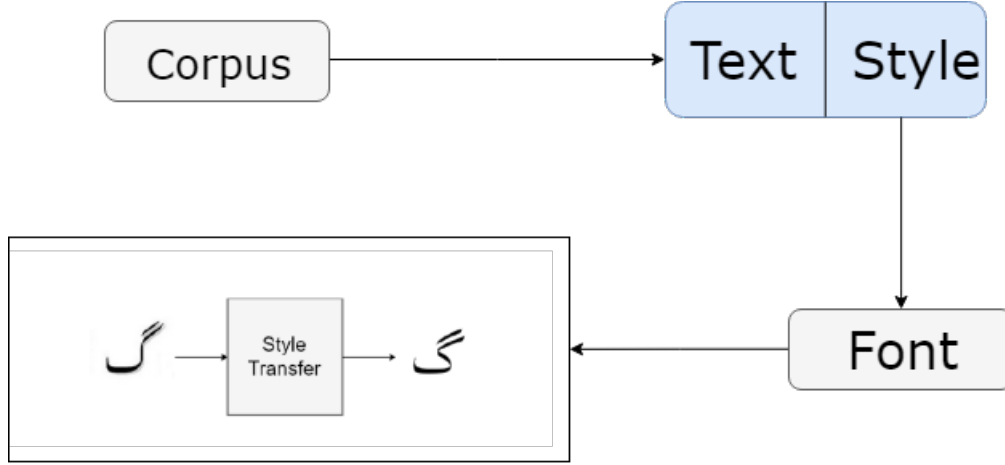


Figure 3.2: Model overview

3.2 Data Set

- For this task, we need a character set that contains every character in Urdu language and combination of different character in Unicode encoding.
- As we will give the representation of those characters in two different fonts to the neural network as an example sets (source font and target font).

3.2.1 Data Set Creation

The first challenge is to create a character set, as Urdu language contains 38 distinct characters, degree **2** combination of those characters is **1,444** characters and degree **3** is over **2,085,136** and so on ...

3.2.2 Data Set Creation (Bitmaps)

- We have to input information bit by bit into the neural network, so first we converted every representation of characters into bitmaps for source and target fonts.
- The source font is stored in one bitmap file and target font in another. A subset of which will be given as an input into the neural network to start training.

آ ا ب پ ت ٹ ث ج چ ح خ د ڈ ذ ر ژ ز س ش ص
ض ط ظ ع غ ف ق ک گ ل م ن و ؤ ء ه ئ ی ے

Figure 3.3: Degree of 1 Urdu Characters

آ آ ا ب آپ ات آٹ اٹ آج آچ آخ آد آڈ آذ آر آر آڑ آس آش اص اص اط آظ
اع آغ اف اق اک آگ آل آم آن اں او آؤ آء آہ آھ آئ آی آے

Figure 3.4: Degree of 2 Urdu Characters

3.3 Problems

3.3.1 Problems during data set creation

We were facing the following problems:

1. Combined character were represented as single characters.
2. Characters were written from left to right(shown in figure 3.6)

The characters are in the isolated form, which means that every character is rendered

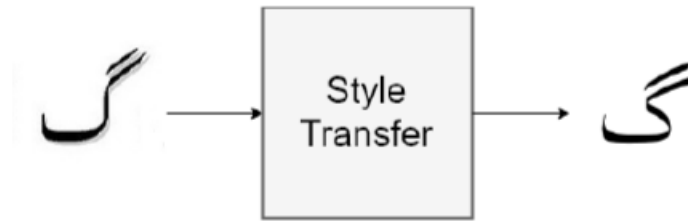
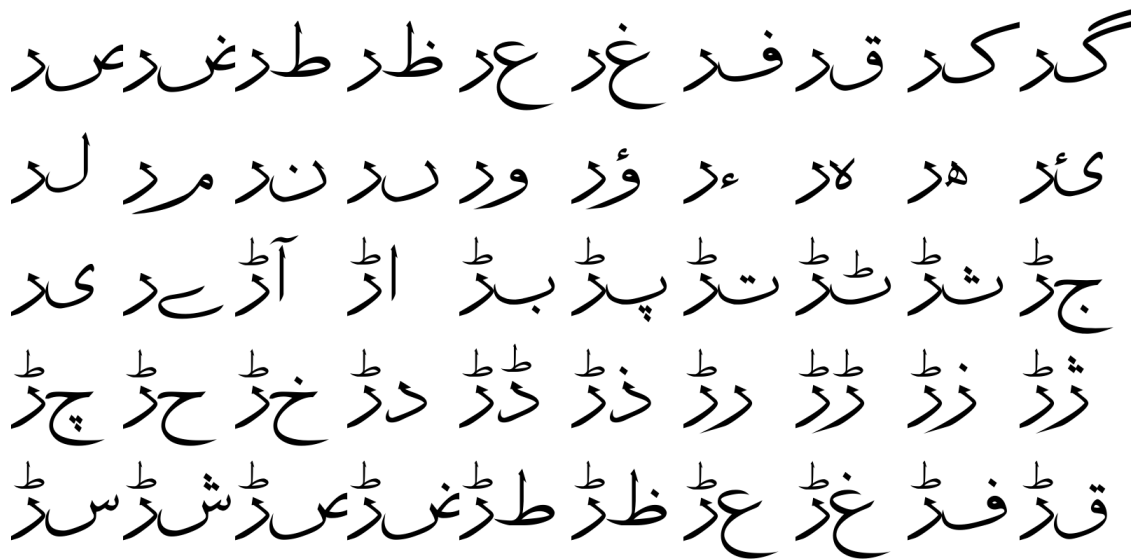
Figure 3.5: Data Set Creation using Bitmaps²⁷

Figure 3.6: Issues faced during character set generation

regardless of its surroundings shown in figure 3.6.

3.3.2 Solution of the Problems

To solve the issues we used the following libraries:

- **Unicode Bi-directional algorithm**,⁵ which is implemented purely in Python in `python-bidi`.

The only issue left to solve is to reshape those characters and replace them with their correct shapes according to their surroundings.

- **Arabicreshaper**⁶ library helps with the reshaping so we can get the proper result.

After using both of these libraries, the results we got were pretty impressive and our both problems were resolved (shown in figure 3.7, figure 3.8).

رہ	آو	نب	گو	چی	چل	زت	طس	صض	قف
دء	دک	پہ	ظل	رت	بط	سم	اغ	چت	کط
رئس	ای	ذخ	تج	ت	ڈص	فق	عت	عخ	ہش
دز	ئہ	ڈت	نآ	ضث	زخ	ظو	حہ	ثا	پب
زو	غن	چق	پڈ	ظھ	زو	یق	تخ	تج	لح
دی	صن	شء	رےپ	نی	با	ئژ	طق	رےا	حص
صپ	آس	وئ	وڈ	گی	تم	چا	چت	لژ	ید
ژٹ	صئ	رےھ	فل	چم	ہج	عی	فج	رک	نم
ثغ	ظء	مح	فض	ثق	ژم	فج	تض	غر	دں
کج	زڑ	صل	رر	بب	خخ	ڈے	رژ	جت	ژک

Figure 3.7: Solution of the problem (shown in figure 3.6)

3.4 Network Structure

We used following structure to get accurate results in reasonable time after considering many architectures.



Figure 3.8: Solution of the problem (shown in figure 3.6)

3.5 Working

3.5.1 Convolutional Layer

Convolutional layer consists of filters, that are applied to the output of previous layer to extract features from it and it can be of different size and can contain different number of filters shown in figure 3.10.

3.5.2 Normalization

After every convolutional layer, we have done normalization to sharpen the image and remove unnecessary pixels as shown in figure 3.11.

Input (160x160)
Conv (64x64, filters:8) x 2
Conv (32x32, filters:32) x n
Conv (16x16, filters:64) x n
Conv (7x7, filters:128) x n
Conv (3x3, filters:128) x 2
MaxPool (2x2)
Dropout
Sigmoid
Output (80x80)

Figure 3.9: Network Structure used to get accurate results.



Figure 3.10: Convolutional Layer (size = 64x64, filters = 8)

ReLu

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

Leaky ReLu

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{otherwise} \end{cases}$$



Figure 3.11: Normalization

3.6 Maxpooling

Maxpooling down-sample the representation by decreasing its dimensionality.

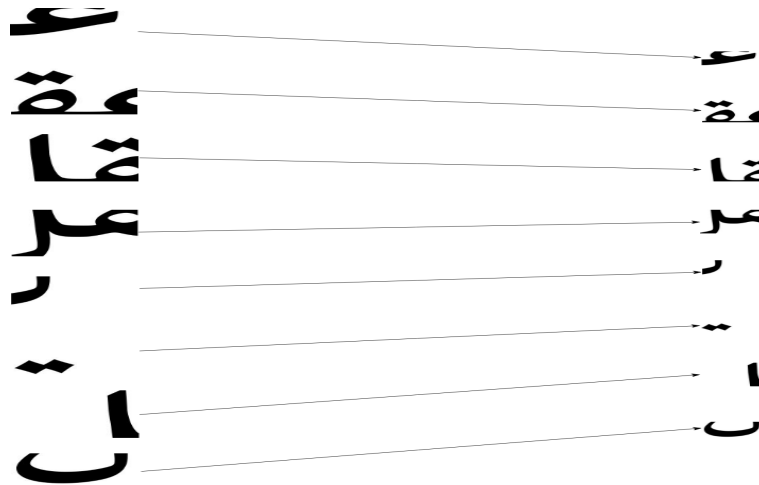


Figure 3.12: Maxpooling

3.7 Dropout

Dropout refers to dropping out units in a neural network to prevent over-fitting shown in [figure 3.13](#)

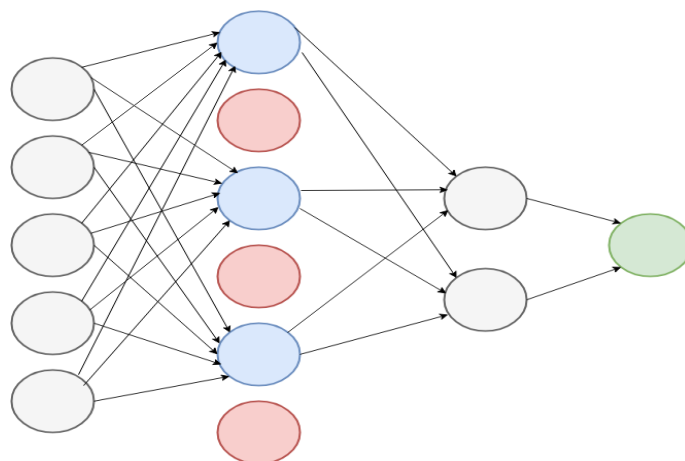


Figure 3.13: Dropout example

3.8 Sigmoid

Using sigmoid function (shown in figure 3.14), normalize the image representations even more by bringing its value between 0 and 1.

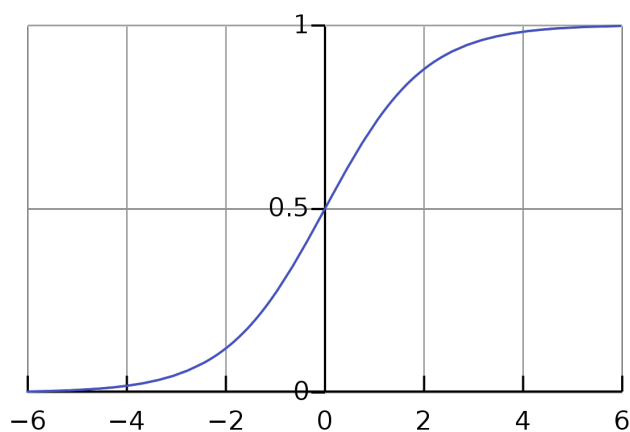


Figure 3.14: Sigmoid function graph

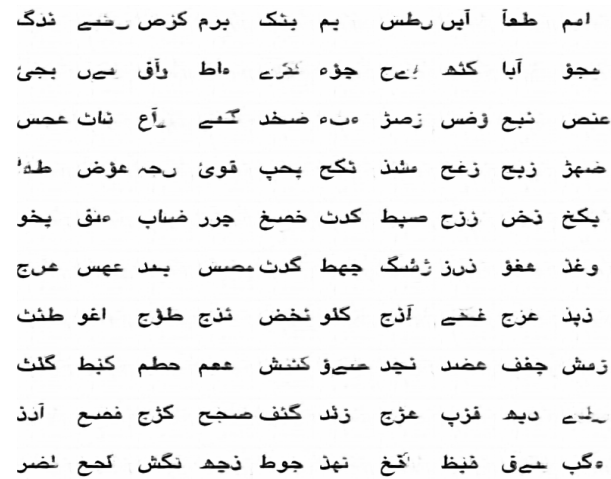
3.9 Output

- A predicted output of dimension 80x80 is created.
- It is compared with the ground-truth and loss is calculated.

- Backpropagation will be used to minimize the loss.

3.10 Results

Frames were created after every 10 iterations on validation set during training and after 4000 iterations, frames can be seen in figure 3.15, and the combined loss graph is shown in figure 3.16



امم طعاً آیں رطس بم بٹک برم کزص رشی ندگ
 عجؤ آبا کئم ریح جؤہ نحرے ااط واق یں بجئ
 عنص نبع وُص زصز ءء صخذ گئے رَآع ثات عص
 ضہڑ زبج زغح شہذ نکج یحپ قوی رحہ عوض طہہ
 یکج نض ززح صبط کدث خمغ چرر ضساب ءبق یخو
 وغد عفؤ نرز رُشگ جھط گدث مصس بحد عہس عرج
 ذہذ عزج غحے آذج کلونخض نذج طلوج اغو طٹٹ
 رَمش جفف عضد نجد عیر وکنش ععم حطم کبط گلث
 رَے دیھ قزپ عزج زُذد گئف صحج کزج فصع آذذ
 ءگب یقی قبط آکخ نہذ جوط نچم نکش کجح لضر

Figure 3.15: Frame generated after 4000 iterations

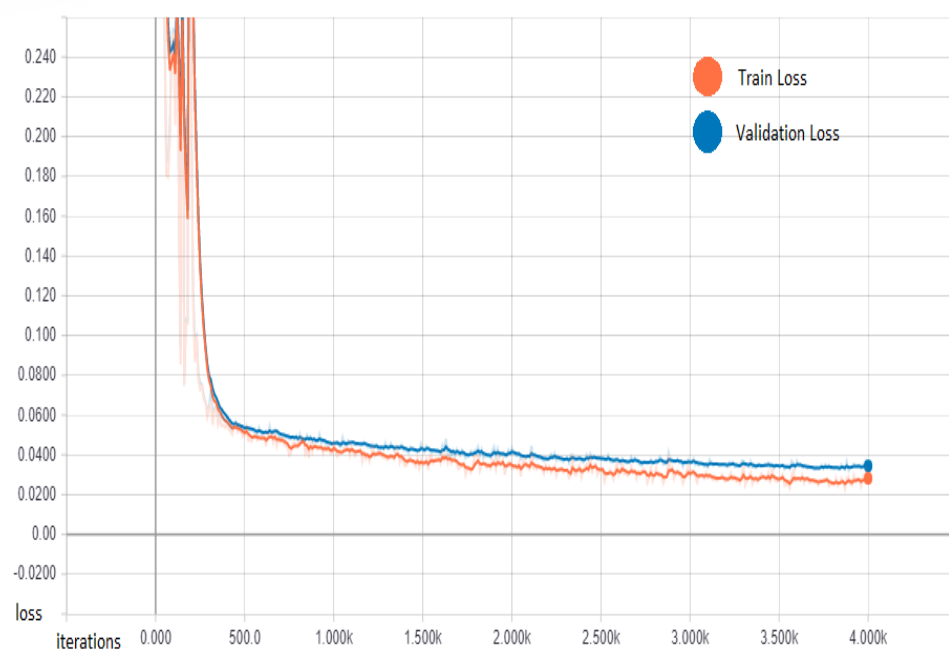


Figure 3.16: Combined Loss Graph

Chapter 4

Background

4.1 Corpus

A corpus or text corpus is a large and structured set of texts. They are used to do statistical analysis and hypothesis testing, checking occurrences or validating linguistic rules within a specific language territory.

4.1.1 Use Case

A corpus provides grammarians, lexicographers, and other interested parties with better descriptions of a language. Computer-process able corpora allow linguists to adopt the principle of total accountability, retrieving all the occurrences of a particular word or structure for inspection or randomly selected samples. Corpus analysis provide **lexical information, morph syntactic information, semantic information and pragmatic information**. -Linguistic information is provided by **concordance** and **frequency counts**.

4.1.2 Concordance

Concordances are listings of the occurrences of a particular feature or combination of features in a corpus. Each occurrence found (or hit) is displayed with a certain amount of context, the text preceding and following it. The most commonly used concordance type

is **KWIC** which stands for **Key Word In Context**. It shows one hit per line of screen or print-out with principal search feature (or focus) highlighted in the center. Concordance is used to determine the syntax in which a form is embedded.

4.1.3 Frequency Counts

It counts the number of hits. Frequency counts require finding all the occurrences of a particular feature in the corpus. So it is implicit in concordance. Software is used for this purpose. Frequency counts can be explained statistically.

4.1.4 Applications of Corpus

Corpora or Corpus are used in the development of NLP tools. Applications include spell-checking, grammar-checking, speech recognition, text-to-speech and speech-to-text synthesis, automatic abstraction and indexing, information retrieval and machine translation. Corpora also used for creation of new dictionaries and grammars for learners.

4.2 Natural Language Processing

4.2.1 Natural Language Processing (NLP)

It is a field of computer science, artificial intelligence concerned with the interactions between computers and human (natural) languages, and, in particular, concerned with programming computers to fruitfully process large natural language data.

4.2.2 Natural Language Analysis

It runs into many stages, namely tokenization, lexical analysis, syntactic analysis, semantic analysis, and pragmatic analysis.

Lexical analysis and **Syntactic analysis** provides an order and structure of each sentence in the text.

Semantic analysis is to find the literal meaning

Pragmatic analysis is to determine the meaning of the text in context.

These major tasks are further broken down into, parsing and so on.

4.2.3 Tokenization

Tokenization is the process of breaking up the given text into units called tokens. The tokens may be words or number or punctuation mark. Tokenization does this task by locating word boundaries. Ending point of a word and beginning of the next word is called word boundaries. Tokenization is also known as **word segmentation**.

4.2.4 Challenges in Tokenization

Challenges in tokenization depends on the type of language. Languages such as English and French are referred to as space-delimited as most of the words are separated from each other by white spaces. Languages such as Chinese and Thai are referred to as unsegmented as words do not have clear boundaries. Tokenizing unsegmented language sentences requires additional lexical and morphological information. Tokenization is also affected by writing system and the typographical structure of the words. Structures of languages can be grouped into three categories:

Isolating: Words do not divide into smaller units. Example: Mandarin Chinese

Agglutinative: Words divide into smaller units. Example: Japanese, Tamil

Inflectional: Boundaries between morphemes are not clear and ambiguous in terms of grammatical meaning. Example: Latin.

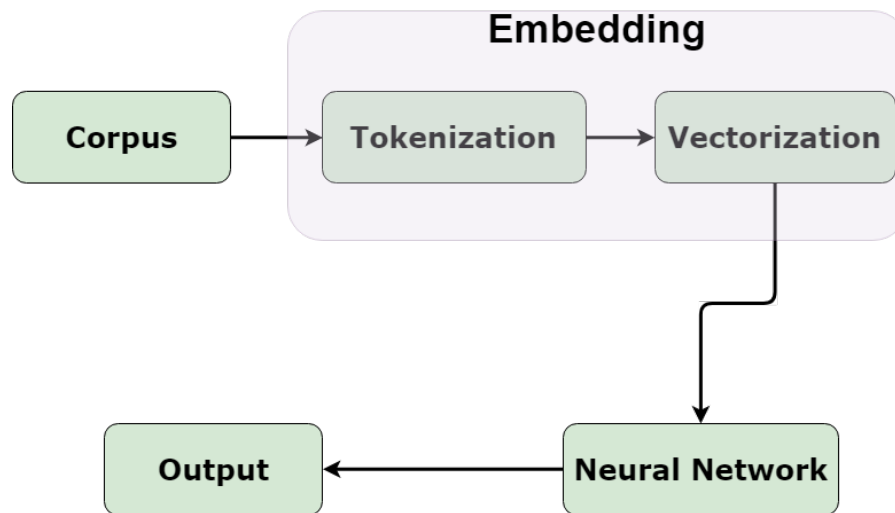


Figure 4.1: Different Stages involved in Text Style Transfer

4.3 Stylistic Features

4.3.1 Sentence Length

In English grammar, sentence length refers to the number of words in a sentence. Most readability formulas use the number of words in a sentence to measure its difficulty. A very effective way, to make your writing deadly and monotonous is by never varying sentence length. One short sentence after another makes your sentence childish. Similarly all long sentences can make your writing hard to read.

Example:

Long Sentence: Last weekend I saw a science fiction film. Three friends went with me. The film focused on the experiments of a mad doctor. He altered his patients' lives by manipulating their dreams.

Short Sentence: Last weekend three friends and I saw a science fiction film in which a mad doctor altered his patients' lives by manipulating their dreams.

4.3.2 Tone

Tone in writing adapts and changes to suit the audience and the situation. The choice of words and the way sentences are put together convey tone. By using tone an author expresses his attitude thorough his writing. The tone can change very quickly, or may remain the same throughout the story. The types of tone are given below

- Logical or emotional
- Intimate or distant
- Serious or humorous

Example:

Without Contractions: It is strange that the professor has not assigned any papers for three weeks.

With Contractions: It's strange that the professor hasn't assigned any papers for three weeks.

4.3.3 Level of Formality

Formality is the degree of courteousness and personalization for a selected audience. There are three levels of formality in English.

- Formal English
- Semi-formal English
- Informal English

Formal English We use formal English in textbooks, official reports, essays, contracts and official speeches. Punctuation, proper grammar and correct sentence structures are very important in formal English.

Example:

More Formal Way: Research has shown that learning a second language, in addition to leading to expanded career and social opportunities, can also expand the reasoning capability of the brain, although this finding is disputed by some scientists.

Less Formal Way: Research has shown that learning a second language, in addition to leading to expanded career and social opportunities, can also expand.

Semi-formal English We use semi-formal English in day-to-day interaction with colleagues and teachers, popular magazines and interviews. In semi-formal English phrasal verbs, contractions and idioms but avoid slang words.

Example:

Semi-formal: Would you like to join me for lunch?

Informal: Hey, wanna grab a bite to eat?

Informal English We use informal English in chatting online or interacting with friends. In informal English slang words, idioms and expressions are frequently used. Grammar is not as important in informal English.

Example:

Informal: It was awesome!, It was the bomb!

Formal: The conference was great.

4.3.4 Vocabulary

Vocabulary is the all the language and words either used or understood by a person or group of people. Vocabulary can be all the words that a toddler understands or language used by doctors.

Example:

Consider, minute, accord, evidence, practice, intend, concern, commit, issue, approach, establish and utter are the words used in English language.

4.3.5 Use of Passive Voice and Active Voice

In a sentence written in the active voice, the subject of sentence performs the action. But if the sentence written in passive voice, the subject receives the action. Using active voice for the majority of your sentences makes your meaning clear for readers, and keeps the sentences from becoming too complicated or wordy. Use passive voice to emphasize the action, to keep the subject and focus consistent throughout a passage, to be tactful by not naming the actor and to create an authoritative tone. Generally, try to use the active voice whenever possible. Passive voice sentences often use more words, can be vague, and can lead to a tangle of prepositional phrases.

Example:

Active Voice: Researchers earlier showed that high stress can cause heart attacks.

Passive Voice: It was earlier demonstrated that heart attacks can be caused by high stress.

4.3.6 Sentiments

A sentiment that people have is an attitude which is based on their thoughts and feelings. It can be affection, passion, opinion, belief, persuasion or conviction.

Example:

- I agree with your sentiments regarding the road bridge.
- The academy became host to people who shared similar sentiments and many were keen to express them.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

- We have created a font for all characters in Urdu from a subset of characters.
- We have created our character set of Urdu and stored it in a file (Unicode encoded).
- There were some problems in creating a data set of bitmaps for each character (discussed earlier).
- The problems were solved by using some techniques,^{6,5}
- Finally, we have a data set of bitmaps for each character in Urdu up to certain number of length.
- Pre-processing of data set.
- We have removed repetition of characters.
- Finding out perfect sample set to input into neural network for training.
- Figuring out the Neural Network structure to use for our problem.

5.2 Future Work

- Pre-processing of data set.
- Removing repetition of characters.
- Getting rid of combination of characters that didn't join together.
- Finding out perfect sample set to input into neural network for training.
- Figuring out the Neural Network structure to use for our problem.

Bibliography

- [1] Shumeet Baluja. Learning typographic style. *arXiv preprint arXiv:1603.04000*, 2016.
- [2] Keith Carlson, Allen Riddell, and Daniel Rockmore. Zero-shot style transfer in text using recurrent neural networks. *arXiv preprint arXiv:1711.04731*, 2017.
- [3] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. Stylebank: An explicit representation for neural image style transfer. 2017.
- [4] Tian Qi Chen and Mark Schmidt. Fast patch-based style transfer of arbitrary style. *arXiv preprint arXiv:1612.04337*, 2016.
- [5] Mark Davis. Unicode bidirectional algorithm. <http://unicode.org/reports/tr9/>, year = 2017.
- [6] Abdullah Diab. arabic-reshaper 2.0.14. 2017. <https://pypi.python.org/pypi/arabic-reshaper/2.0.14>.
- [7] Thaminda Edirisooriya and Morgan Tenney. Applying artistic style transfer to natural language. 2017.
- [8] Michael Elad and Peyman Milanfar. Style transfer via texture synthesis. *IEEE Transactions on Image Processing*, 26(5):2338–2351, 2017.
- [9] Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. Style transfer in text: Exploration and evaluation. *arXiv preprint arXiv:1711.06861*, 2017.
- [10] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

- [11] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. pages 2414–2423, 2016.
- [12] Leon A Gatys, Alexander S Ecker, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. Controlling perceptual factors in neural style transfer. 2017.
- [13] Agrim Gupta, Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Characterizing and improving stability in neural style transfer. *arXiv preprint arXiv:1705.02092*, 2017.
- [14] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. *CoRR, abs/1703.06868*, 2017.
- [15] Jeremiah Johnson. Neural style representations of fine art. 2017.
- [16] Jad Kabbara and Jackie Chi Kit Cheung. Stylistic transfer in natural language generation systems using recurrent neural networks. pages 43–47, 2016.
- [17] Pak-Keung Lai, Dit-Yan Yeung, and Man-Chi Pong. A heuristic search approach to chinese glyph generation using hierarchical character composition. *Computer Processing of Oriental Languages*, 10(3):307–323, 1996.
- [18] Zhouhui Lian and Jianguo Xiao. Automatic shape morphing for chinese characters. page 2, 2012.
- [19] Zhouhui Lian, Bo Zhao, and Jianguo Xiao. Automatic generation of large-scale handwriting fonts via style learning. page 12, 2016.
- [20] Jeng-Wei Lin, Chih-Yin Wang, Chao-Lung Ting, and Ray-I Chang. Font generation of personal handwritten chinese characters. 9069:90691T, 2014.
- [21] Ming Lu, Hao Zhao, Anbang Yao, Feng Xu, Yurong Chen, and Li Zhang. Decoder network over lightweight reconstructed feature for fast semantic style transfer. pages 2469–2477, 2017.
- [22] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep photo style transfer. *CoRR, abs/1703.07511*, 2017.

- [23] Huy Quoc Phan, Hongbo Fu, and Antoni B Chan. Flexyfont: Learning transferring rules for flexible typeface synthesis. 34(7):245–256, 2015.
- [24] Scott Reed, Zeynep Akata, Xinchun Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*, 2016.
- [25] Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. Style transfer from non-parallel text by cross-alignment. pages 6833–6844, 2017.
- [26] Rapee Suveeranont and Takeo Igarashi. Example-based automatic font generation. pages 127–138, 2010.
- [27] Hanyu Xue. Chinese font style transfer with neural network dartmouth computer science technical report tr2017-830 a thesis submitted to the faculty. 2017.
- [28] Huihuang Zhao, Paul L Rosin, and Yu-Kun Lai. Automatic semantic style transfer using deep convolutional neural networks and soft masks. *arXiv preprint arXiv:1708.09641*, 2017.