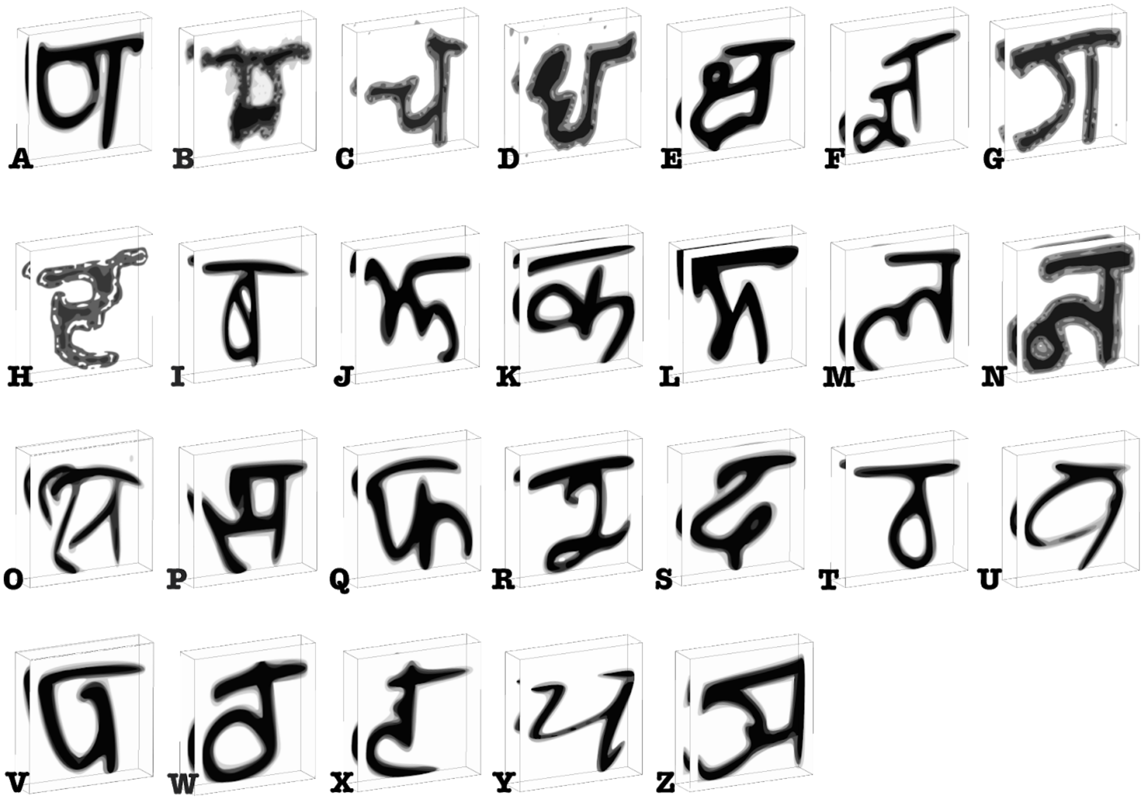
ART & ML PROJECT 2
3/10/2018

The Generation of New Set of Alphabet Style: DCGAN

Candy Dong, Vera Schulz, Zhuona Ma

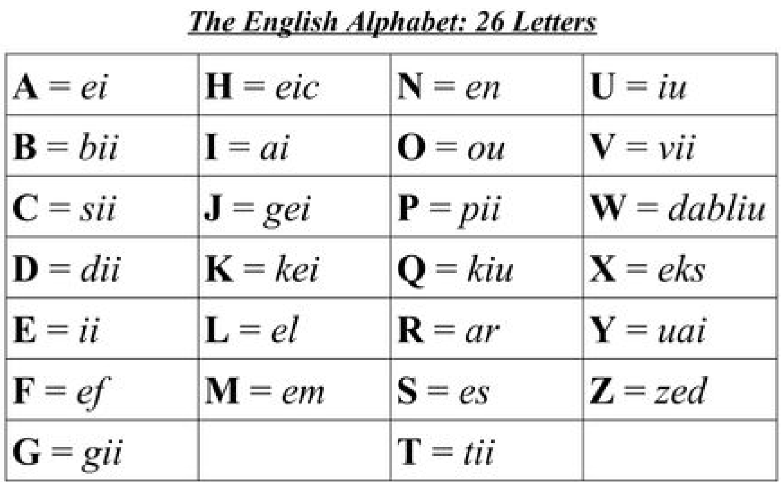
Github link: *https://github.com/foodiebirb/ArtML_Project2*



Concept

In this project, we investigate the collision of cultures by merging different styles of letters in different languages. We generate a new set of alphabet style on a training set which contains 1253 different English fonts, and 2000 handwritten Hindi letters for each letter from A-Z in the alphabet. We chose Hindi because we were fascinated with how the letters of the alphabet were written, and how they were phonetically expressed. The individuality of most languages can be told by how the alphabet is ordered and how the individual letters are arranged to make sense. Hindi has a very unique alphabet and each of the characters seem to have an artistic contruct to the outsider eye and we wanted to understand this.

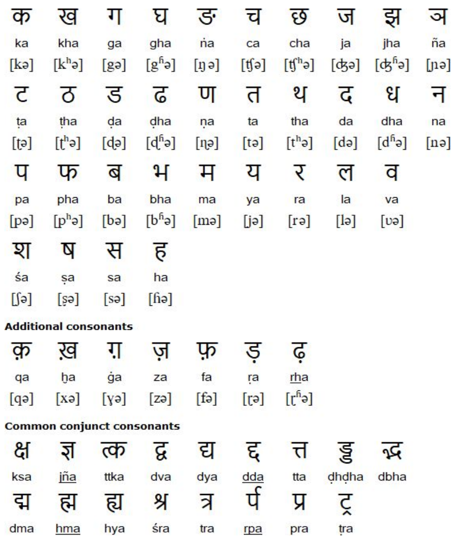
An extension of this project would look at how different individual's writing styles represents the regions of India they may be from, this would include looking at how they handwrite and also the varying slang that is used from region to region. By doing this we would be able to analyze any handwritten note in Hindi and be able to potentially pinpoint from what region in India the person comes from.



The English Alphabet: 26 Letters

A = *ei*	**H** = *eic*	**N** = *en*	**U** = *iu*
B = *bii*	**I** = *ai*	**O** = *ou*	**V** = *vii*
C = *sii*	**J** = *gei*	**P** = *pii*	**W** = *dabliu*
D = *dii*	**K** = *kei*	**Q** = *kiu*	**X** = *eks*
E = *ii*	**L** = *el*	**R** = *ar*	**Y** = *uai*
F = *ef*	**M** = *em*	**S** = *es*	**Z** = *zed*
G = *gii*		**T** = *tii*	

Figure 1. English Letters



क	ख	ग	घ	ङ	च	छ	ज	झ	ञ
ka	kha	ga	gha	ṅa	ca	cha	ja	jha	ña
[kə]	[kʰə]	[gə]	[gʱə]	[ŋə]	[tʃə]	[tʃʰə]	[dʒə]	[dʒʱə]	[ɲə]
ट	ठ	ड	ढ	ण	त	थ	द	ध	न
ṭa	ṭha	ḍa	ḍha	ṇa	ta	tha	da	dha	na
[ʈə]	[ʈʰə]	[ɖə]	[ɖʱə]	[ɳə]	[t̪ə]	[t̪ʰə]	[d̪ə]	[d̪ʱə]	[nə]
प	फ	ब	भ	म	य	र	ल	व	
pa	pha	ba	bha	ma	ya	ra	la	va	
[pə]	[pʰə]	[bə]	[bʱə]	[mə]	[jə]	[rə]	[lə]	[ʋə]	
श	ष	स	ह						
śa	ṣa	sa	ha						
[ʃə]	[ʂə]	[sə]	[ɦə]						

Additional consonants

क़	ख़	ग़	ज़	फ़	ड़	ढ़
qa	ḫa	ġa	za	fa	ṛa	ṛha
[qə]	[xə]	[ɣə]	[zə]	[fə]	[ɽə]	[ɽʱə]

Common conjunct consonants

क्ष	ज्ञ	त्क	द्व	द्य	द्द	त्त	ड्ढ	द्भ
ksa	jña	ttka	dva	dya	dda	tta	ḍhḍha	dbha
द्म	ह्म	ह्य	श्र	त्र	र्प	प्र	ट्र	
dma	hma	hya	śra	tra	rpa	pra	ṭra	

Figure 2. Hindi Letters

Technique

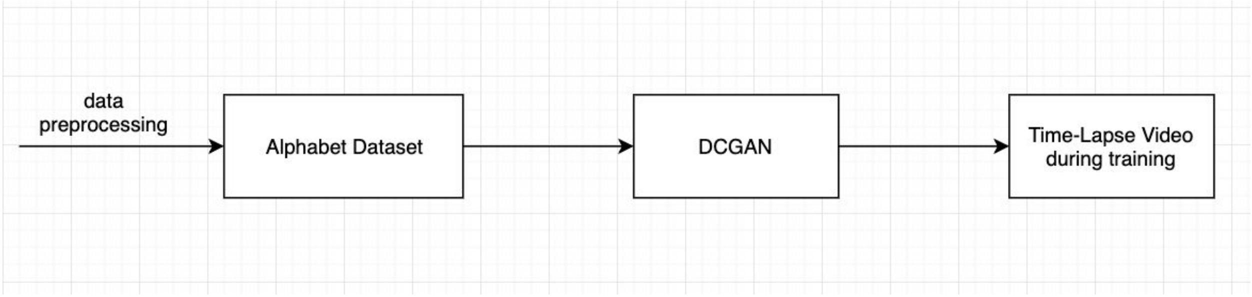


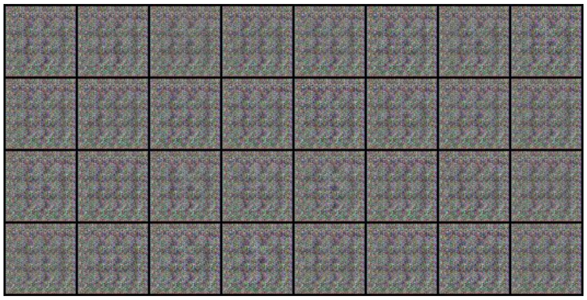
Figure 3. The pipeline

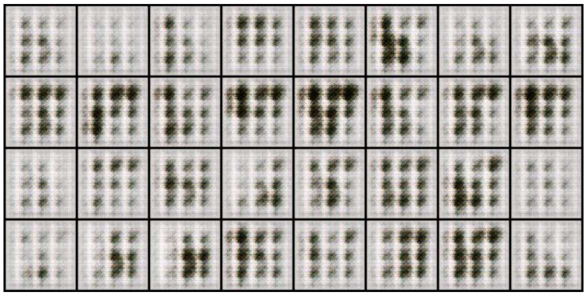
We first transferred characters in the two raw dataset into same size and format, i.e. 64x64, black characters + white background, and .png format. Then we combined the hindi(Nagari) characters and English ones from A to Z respectively into individual dataset. (26 in total). We trained every combination for 34 epoches (1 epoch with approximately 3000 images, 1000 English and 2000 Hindi). After training, we manually picked the final character from 32 outcomes that are in-between English and Hindi and aesthetically pleasing. We then post-processed the alphabet we got into vector forms and produced a typeface and called it Enagarish (Nagari + English).

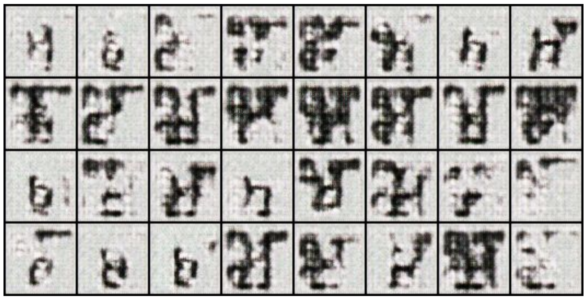
Process

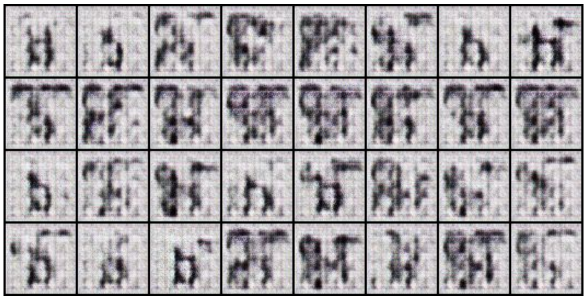
Before we decided to work with an alphabet dataset we were interested in working with Kandinsky's artworks and different songs, to assign emotion to Kandinsky's artworks. We had to do a complete 180 degree turn because the GANs did not converge with the small dataset of songs and artworks we had each had a 100 data points, and we had to consider the lack of resources we had in terms of Amazon AWS and time to train the datasets. After doing the 180 degree shift we went back to the drawing board with an even smaller time frame to work with to decide what we wanted project 2 to be. We needed to consider the time we had to complete this project, our individual skillsets and the resources we had available from Amazone AWS. We decided to go with a project focused on language and decided that our database would be individual alphabet letters, because each letter has a unique form and phonetic sound.

Our dataset was created using the 26 lowercase alphabet letters from 3,125 different fonts, in which 2000 fonts are Hindi(Nagari), and 1125 fonts are English. The characters were rendered into 64px by 64px binary images with size normalization. We performed color inverting and image resizing to make all the images the same size and have the same color channels which consist of only black and white color. Also, we ensured that no outline fonts or uppercase fonts were used during the preprocessing phase.









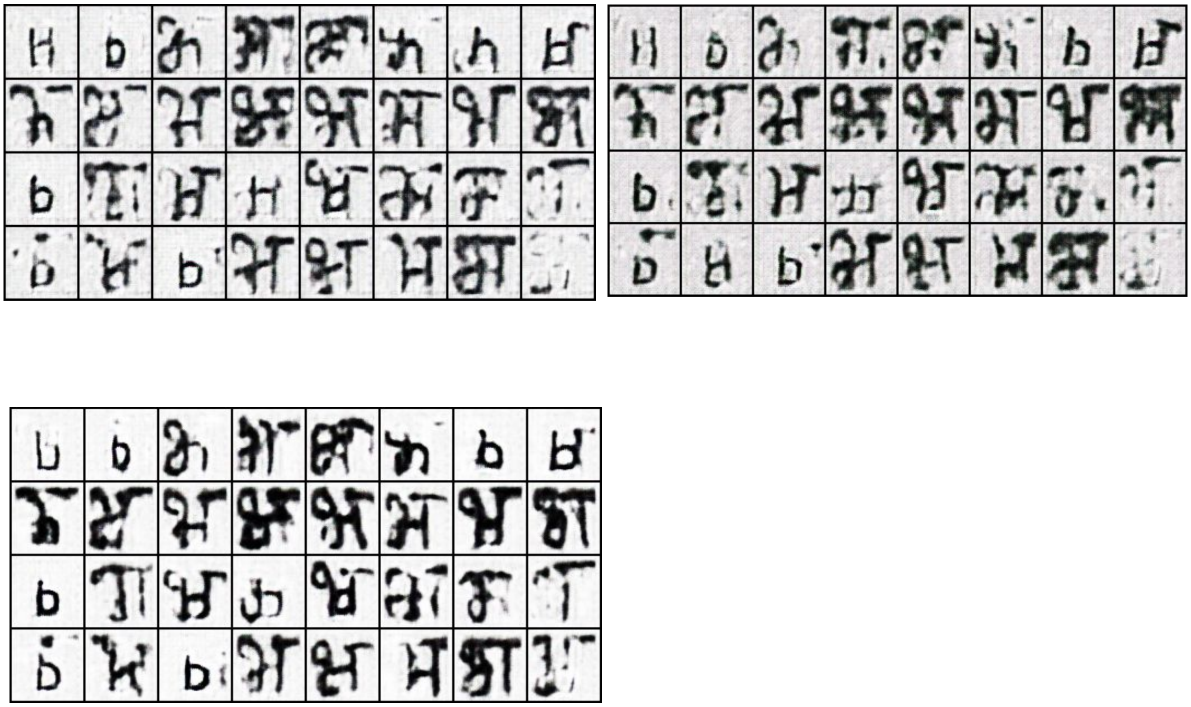


Figure 4. Letter 'B' After Epochs 28 to 34

Our Results

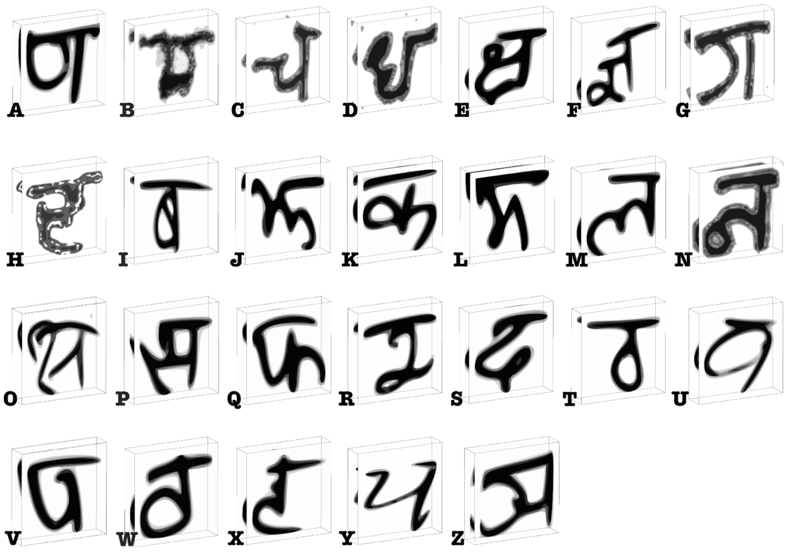
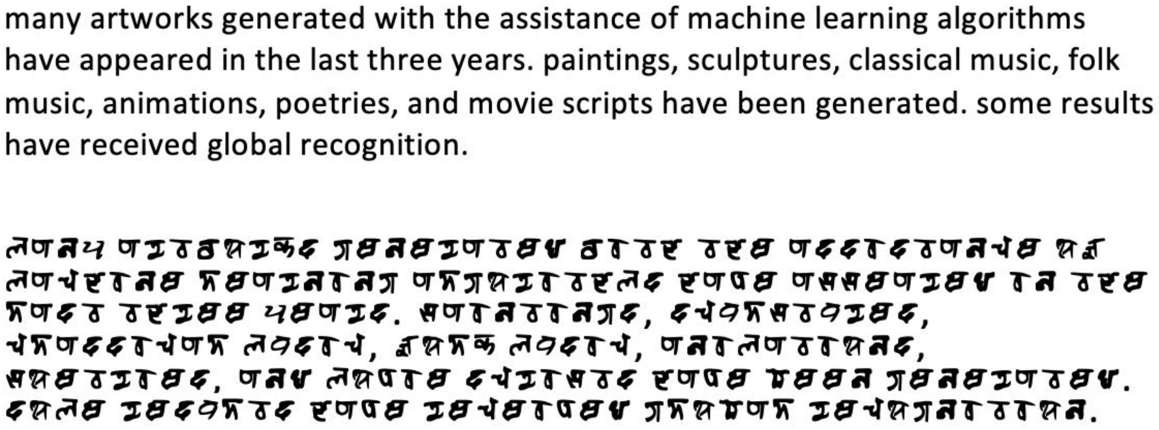


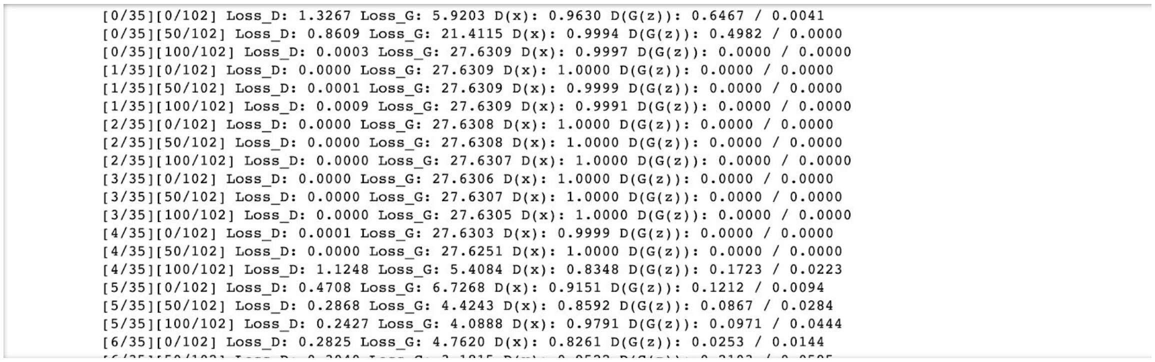
Figure 5. Final Results from A-Z



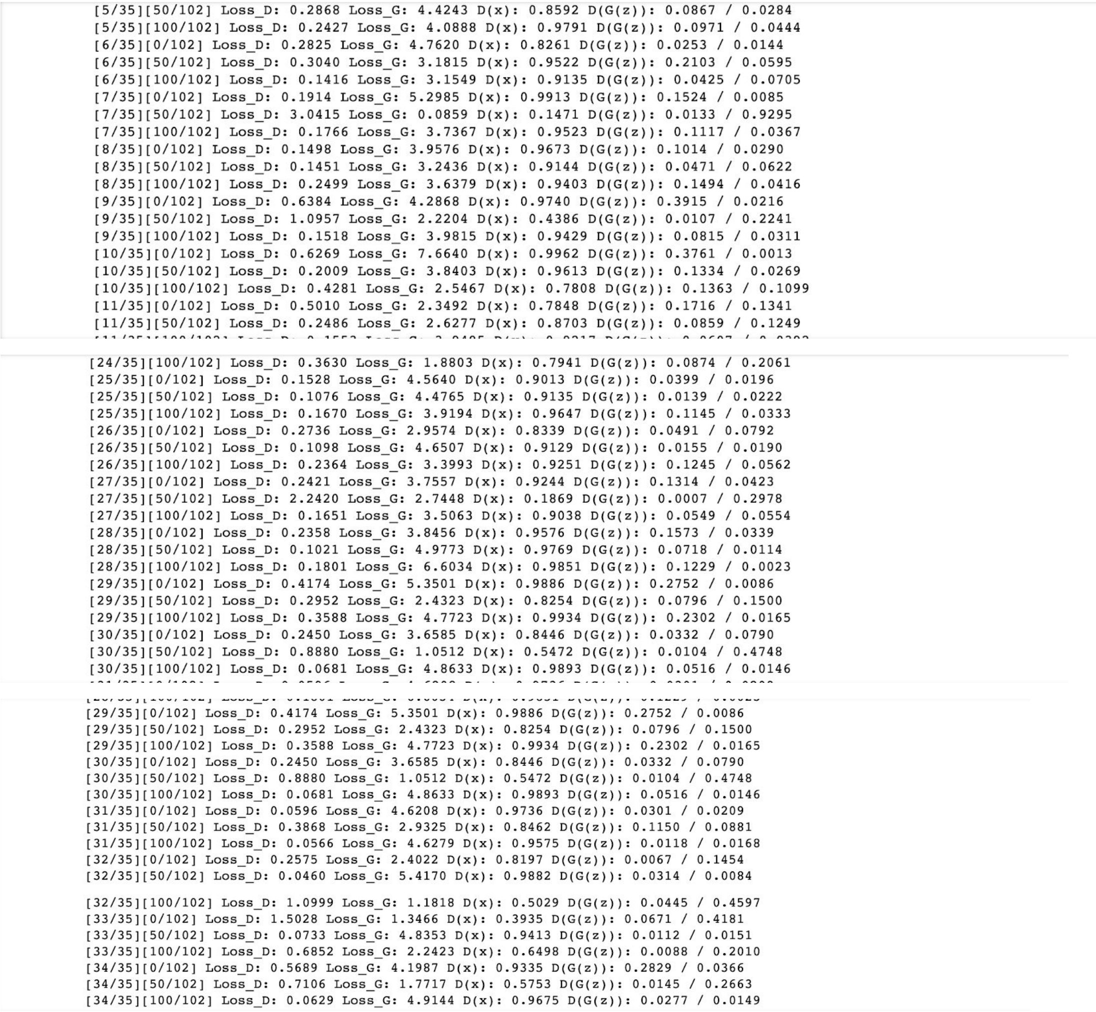
many artworks generated with the assistance of machine learning algorithms have appeared in the last three years. paintings, sculptures, classical music, folk music, animations, poetries, and movie scripts have been generated. some results have received global recognition.

Figure 6. Rewriting an English Paragraph with Our Font

We found that for a certain character that if Hindi and English characters look similar in shapes, then the losses of discriminator and generator tend to converge to 0. Otherwise, the algorithm does not converge and result in blurry images. Below is the loss for character B in 34 epochs.



```
[0/35][0/102] Loss_D: 1.3267 Loss_G: 5.9203 D(x): 0.9630 D(G(z)): 0.6467 / 0.0041
[0/35][50/102] Loss_D: 0.8609 Loss_G: 21.4115 D(x): 0.9994 D(G(z)): 0.4982 / 0.0000
[0/35][100/102] Loss_D: 0.0003 Loss_G: 27.6309 D(x): 0.9997 D(G(z)): 0.0000 / 0.0000
[1/35][0/102] Loss_D: 0.0000 Loss_G: 27.6309 D(x): 1.0000 D(G(z)): 0.0000 / 0.0000
[1/35][50/102] Loss_D: 0.0001 Loss_G: 27.6309 D(x): 0.9999 D(G(z)): 0.0000 / 0.0000
[1/35][100/102] Loss_D: 0.0009 Loss_G: 27.6309 D(x): 0.9991 D(G(z)): 0.0000 / 0.0000
[2/35][0/102] Loss_D: 0.0000 Loss_G: 27.6308 D(x): 1.0000 D(G(z)): 0.0000 / 0.0000
[2/35][50/102] Loss_D: 0.0000 Loss_G: 27.6308 D(x): 1.0000 D(G(z)): 0.0000 / 0.0000
[2/35][100/102] Loss_D: 0.0000 Loss_G: 27.6307 D(x): 1.0000 D(G(z)): 0.0000 / 0.0000
[3/35][0/102] Loss_D: 0.0000 Loss_G: 27.6306 D(x): 1.0000 D(G(z)): 0.0000 / 0.0000
[3/35][50/102] Loss_D: 0.0000 Loss_G: 27.6307 D(x): 1.0000 D(G(z)): 0.0000 / 0.0000
[3/35][100/102] Loss_D: 0.0000 Loss_G: 27.6305 D(x): 1.0000 D(G(z)): 0.0000 / 0.0000
[4/35][0/102] Loss_D: 0.0001 Loss_G: 27.6303 D(x): 0.9999 D(G(z)): 0.0000 / 0.0000
[4/35][50/102] Loss_D: 0.0000 Loss_G: 27.6251 D(x): 1.0000 D(G(z)): 0.0000 / 0.0000
[4/35][100/102] Loss_D: 1.1248 Loss_G: 5.4084 D(x): 0.8348 D(G(z)): 0.1723 / 0.0223
[5/35][0/102] Loss_D: 0.4708 Loss_G: 6.7268 D(x): 0.9151 D(G(z)): 0.1212 / 0.0094
[5/35][50/102] Loss_D: 0.2868 Loss_G: 4.4243 D(x): 0.8592 D(G(z)): 0.0867 / 0.0284
[5/35][100/102] Loss_D: 0.2427 Loss_G: 4.0888 D(x): 0.9791 D(G(z)): 0.0971 / 0.0444
[6/35][0/102] Loss_D: 0.2825 Loss_G: 4.7620 D(x): 0.8261 D(G(z)): 0.0253 / 0.0144
```



```
[5/35][50/102] Loss_D: 0.2868 Loss_G: 4.4243 D(x): 0.8592 D(G(z)): 0.0867 / 0.0284
[5/35][100/102] Loss_D: 0.2427 Loss_G: 4.0888 D(x): 0.9791 D(G(z)): 0.0971 / 0.0444
[6/35][0/102] Loss_D: 0.2825 Loss_G: 4.7620 D(x): 0.8261 D(G(z)): 0.0253 / 0.0144
[6/35][50/102] Loss_D: 0.3040 Loss_G: 3.1815 D(x): 0.9522 D(G(z)): 0.2103 / 0.0595
[6/35][100/102] Loss_D: 0.1416 Loss_G: 3.1549 D(x): 0.9135 D(G(z)): 0.0425 / 0.0705
[7/35][0/102] Loss_D: 0.1914 Loss_G: 5.2985 D(x): 0.9913 D(G(z)): 0.1524 / 0.0085
[7/35][50/102] Loss_D: 3.0415 Loss_G: 0.0859 D(x): 0.1471 D(G(z)): 0.0133 / 0.9295
[7/35][100/102] Loss_D: 0.1766 Loss_G: 3.7367 D(x): 0.9523 D(G(z)): 0.1117 / 0.0367
[8/35][0/102] Loss_D: 0.1498 Loss_G: 3.9576 D(x): 0.9673 D(G(z)): 0.1014 / 0.0290
[8/35][50/102] Loss_D: 0.1451 Loss_G: 3.2436 D(x): 0.9144 D(G(z)): 0.0471 / 0.0622
[8/35][100/102] Loss_D: 0.2499 Loss_G: 3.6379 D(x): 0.9403 D(G(z)): 0.1494 / 0.0416
[9/35][0/102] Loss_D: 0.6384 Loss_G: 4.2868 D(x): 0.9740 D(G(z)): 0.3915 / 0.0216
[9/35][50/102] Loss_D: 1.0957 Loss_G: 2.2204 D(x): 0.4386 D(G(z)): 0.0107 / 0.2241
[9/35][100/102] Loss_D: 0.1518 Loss_G: 3.9815 D(x): 0.9429 D(G(z)): 0.0815 / 0.0311
[10/35][0/102] Loss_D: 0.6269 Loss_G: 7.6640 D(x): 0.9962 D(G(z)): 0.3761 / 0.0013
[10/35][50/102] Loss_D: 0.2009 Loss_G: 3.8403 D(x): 0.9613 D(G(z)): 0.1334 / 0.0269
[10/35][100/102] Loss_D: 0.4281 Loss_G: 2.5467 D(x): 0.7808 D(G(z)): 0.1363 / 0.1099
[11/35][0/102] Loss_D: 0.5010 Loss_G: 2.3492 D(x): 0.7848 D(G(z)): 0.1716 / 0.1341
[11/35][50/102] Loss_D: 0.2486 Loss_G: 2.6277 D(x): 0.8703 D(G(z)): 0.0859 / 0.1249
```

```
[24/35][100/102] Loss_D: 0.3630 Loss_G: 1.8803 D(x): 0.7941 D(G(z)): 0.0874 / 0.2061
[25/35][0/102] Loss_D: 0.1528 Loss_G: 4.5640 D(x): 0.9013 D(G(z)): 0.0399 / 0.0196
[25/35][50/102] Loss_D: 0.1076 Loss_G: 4.4765 D(x): 0.9135 D(G(z)): 0.0139 / 0.0222
[25/35][100/102] Loss_D: 0.1670 Loss_G: 3.9194 D(x): 0.9647 D(G(z)): 0.1145 / 0.0333
[26/35][0/102] Loss_D: 0.2736 Loss_G: 2.9574 D(x): 0.8339 D(G(z)): 0.0491 / 0.0792
[26/35][50/102] Loss_D: 0.1098 Loss_G: 4.6507 D(x): 0.9129 D(G(z)): 0.0155 / 0.0190
[26/35][100/102] Loss_D: 0.2364 Loss_G: 3.3993 D(x): 0.9251 D(G(z)): 0.1245 / 0.0562
[27/35][0/102] Loss_D: 0.2421 Loss_G: 3.7557 D(x): 0.9244 D(G(z)): 0.1314 / 0.0423
[27/35][50/102] Loss_D: 2.2420 Loss_G: 2.7448 D(x): 0.1869 D(G(z)): 0.0007 / 0.2978
[27/35][100/102] Loss_D: 0.1651 Loss_G: 3.5063 D(x): 0.9038 D(G(z)): 0.0549 / 0.0554
[28/35][0/102] Loss_D: 0.2358 Loss_G: 3.8456 D(x): 0.9576 D(G(z)): 0.1573 / 0.0339
[28/35][50/102] Loss_D: 0.1021 Loss_G: 4.9773 D(x): 0.9769 D(G(z)): 0.0718 / 0.0114
[28/35][100/102] Loss_D: 0.1801 Loss_G: 6.6034 D(x): 0.9851 D(G(z)): 0.1229 / 0.0023
[29/35][0/102] Loss_D: 0.4174 Loss_G: 5.3501 D(x): 0.9886 D(G(z)): 0.2752 / 0.0086
[29/35][50/102] Loss_D: 0.2952 Loss_G: 2.4323 D(x): 0.8254 D(G(z)): 0.0796 / 0.1500
[29/35][100/102] Loss_D: 0.3588 Loss_G: 4.7723 D(x): 0.9934 D(G(z)): 0.2302 / 0.0165
[30/35][0/102] Loss_D: 0.2450 Loss_G: 3.6585 D(x): 0.8446 D(G(z)): 0.0332 / 0.0790
[30/35][50/102] Loss_D: 0.8880 Loss_G: 1.0512 D(x): 0.5472 D(G(z)): 0.0104 / 0.4748
[30/35][100/102] Loss_D: 0.0681 Loss_G: 4.8633 D(x): 0.9893 D(G(z)): 0.0516 / 0.0146
```

```
[29/35][0/102] Loss_D: 0.4174 Loss_G: 5.3501 D(x): 0.9886 D(G(z)): 0.2752 / 0.0086
[29/35][50/102] Loss_D: 0.2952 Loss_G: 2.4323 D(x): 0.8254 D(G(z)): 0.0796 / 0.1500
[29/35][100/102] Loss_D: 0.3588 Loss_G: 4.7723 D(x): 0.9934 D(G(z)): 0.2302 / 0.0165
[30/35][0/102] Loss_D: 0.2450 Loss_G: 3.6585 D(x): 0.8446 D(G(z)): 0.0332 / 0.0790
[30/35][50/102] Loss_D: 0.8880 Loss_G: 1.0512 D(x): 0.5472 D(G(z)): 0.0104 / 0.4748
[30/35][100/102] Loss_D: 0.0681 Loss_G: 4.8633 D(x): 0.9893 D(G(z)): 0.0516 / 0.0146
[31/35][0/102] Loss_D: 0.0596 Loss_G: 4.6208 D(x): 0.9736 D(G(z)): 0.0301 / 0.0209
[31/35][50/102] Loss_D: 0.3868 Loss_G: 2.9325 D(x): 0.8462 D(G(z)): 0.1150 / 0.0881
[31/35][100/102] Loss_D: 0.0566 Loss_G: 4.6279 D(x): 0.9575 D(G(z)): 0.0118 / 0.0168
[32/35][0/102] Loss_D: 0.2575 Loss_G: 2.4022 D(x): 0.8197 D(G(z)): 0.0067 / 0.1454
[32/35][50/102] Loss_D: 0.0460 Loss_G: 5.4170 D(x): 0.9882 D(G(z)): 0.0314 / 0.0084

[32/35][100/102] Loss_D: 1.0999 Loss_G: 1.1818 D(x): 0.5029 D(G(z)): 0.0445 / 0.4597
[33/35][0/102] Loss_D: 1.5028 Loss_G: 1.3466 D(x): 0.3935 D(G(z)): 0.0671 / 0.4181
[33/35][50/102] Loss_D: 0.0733 Loss_G: 4.8353 D(x): 0.9413 D(G(z)): 0.0112 / 0.0151
[33/35][100/102] Loss_D: 0.6852 Loss_G: 2.2423 D(x): 0.6498 D(G(z)): 0.0088 / 0.2010
[34/35][0/102] Loss_D: 0.5689 Loss_G: 4.1987 D(x): 0.9335 D(G(z)): 0.2829 / 0.0366
[34/35][50/102] Loss_D: 0.7106 Loss_G: 1.7717 D(x): 0.5753 D(G(z)): 0.0145 / 0.2663
[34/35][100/102] Loss_D: 0.0629 Loss_G: 4.9144 D(x): 0.9675 D(G(z)): 0.0277 / 0.0149
```

Figure 7. Loss for Character B in all 34 epochs

Reflection

We first tried testing on a dataset of a 100, but realized that the noise cannot converge with the DCGAN algorithm. After this realization, we then collected even more data (3215 data for each letter) for the algorithm to converge to distinguishable results.

We chose the final results because firstly, we were able to identify and differentiate each individual/letter and symbol from another. It was very important to be able to see that the results we created would have the characteristics from both Hindi symbols and English letters. We also believed that it represented a new style of writing in terms of stroke and weight of writing.

DCGAN's interpolation enables us to find a new style which is in-between English and Hindi. We concluded that future steps would include: using larger datasets, collecting dataset only on handwritten data or printed data so that data has more consistency among themselves and manually combine characters with similar shapes regardless of their semantic meanings to produce clearer outcomes. Ideally, we would also be able to try other languages like Russian, where the characters have more overlap and equivalence with English.

We hope that our project can aid in the understanding of language learning and also understanding that rich history of language creation and evolution. To help understand how relevant the difference between languages are and how they can represent a culture's changes throughout history.

Individual Contributions

- **Candy**: Dataset collection. Data pre-processing. DCGAN implementation. Post-process.
- **Vera**: Linguistic research on Hindi and English. Literature reviews on the relationship between characters that evolved in different culture. Dataset collection.
- **Zhuona**: Dataset collection. Data pre-processing. DCGAN implementation. Post-process. Produced Enagarish-Regular font.

References

[1] Schmandt-Besserat, Denise, The Evolution of Writing
https://sites.utexas.edu/dsb/tokens/the-evolution-of-writing/

[2] Vadalkar, Suniti, Multiple Perspecrives of the Origin of Devanagri Script
http://www.academia.edu/21545244/MULTIPLE_PERSPECTIVES_OF_THE_ORIGIN_OF_DEVANAGARI_SCRIPT

[3] Radford, Alec, Metz, Luke, Chintala, Soumith, Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
https://arxiv.org/abs/1511.06434