Generative Models

Project E: Image-to-Image Translation with Conditional Adversarial Networks (Berkeley AI Research)

Parth Shah
SEAS, University at Buffalo
Buffalo, NY, 14260
Person Number: 50291125
parthnay@buffalo.edu

Abstract

The project aims to provide an explanation of Generative Models. It introduces projects A to G each with a different problem statement. The project picked was Project E: Image-to-Image Translation with Conditional Adversarial Networks (Berkeley AI Research). Presented here is an application of a follow-up paper on GAN's.

1 Introduction

The project reported here is on CycleGAN's. The image-to-image translation is a class of vision and graphics problems where the goal is to learn the mapping between an input image and an output image using a training set of aligned image pairs. However, for many tasks, paired training data will not be available. We present an approach for learning to translate an image from a source domain X to a target domain Y in the absence of paired examples. The goal is to learn a mapping G: $X \rightarrow Y$ such that the distribution of images from G(X) is indistinguishable from the distribution Y using an adversarial loss. Because this mapping is highly under-constrained, we couple it with an inverse mapping F: $Y \rightarrow X$ and introduce a cycle consistency loss to push $F(G(X)) \approx X$ (and vice versa). The report is divided into 4 tasks

- ➔ ConditionalGAN - pix2pix
- ➔ CycleGAN [arxiv 1703.10593, 2017.]
- ➔ Why CycleGAN over CGAN?
- ➔ Emoji CycleGAN
- ➔ Results
- ➔ Conclusion

2 ConditionalGAN - pix2pix

The base of this project group was ConditionalGAN's or its popular implementation pix2pix. The aim was to use Conditional adversarial networks as a general-purpose solution to image-to-image translation problems. These networks not only learn the mapping from the input image to output image but also learn a loss function to train this mapping. The applications for CGAN's can be seen as:

- Map to aerial
- Aerial to map
- Semantic segmentation
- Day to Night
- Edges to handbags

- Edges to shoes
- Sketches to handbags
- Sketches to shoes

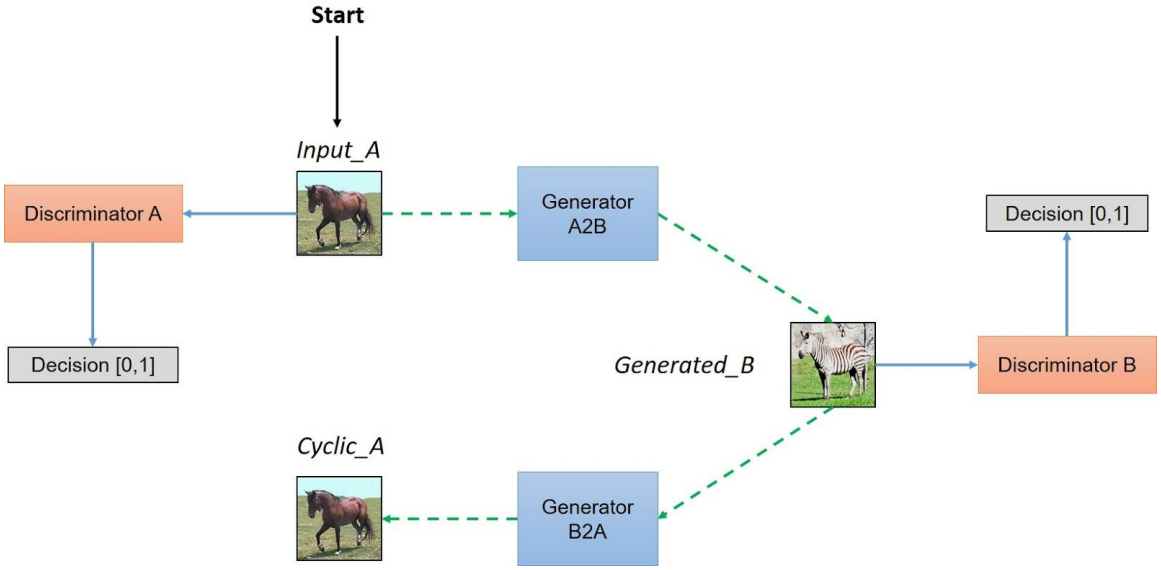
The one constraint for CGAN's is the need for paired images of both domains to train the GAN and to learn from the loss function. The results of pix2pix are pretty good and consistent with the true image. The generated images are very close to the ground truth. Thus pix2pix provides a good solution to the image-to-image translation using CGAN's. However it is a supervised solution to the problem because we need formulated image pairs of both domains which might not be always easily available. Eg - English to a Less spoken language, we may not find enough annotators to correctly translate the sentences to train the CGAN. Thus the need for a better method, Hence CycleGAN.

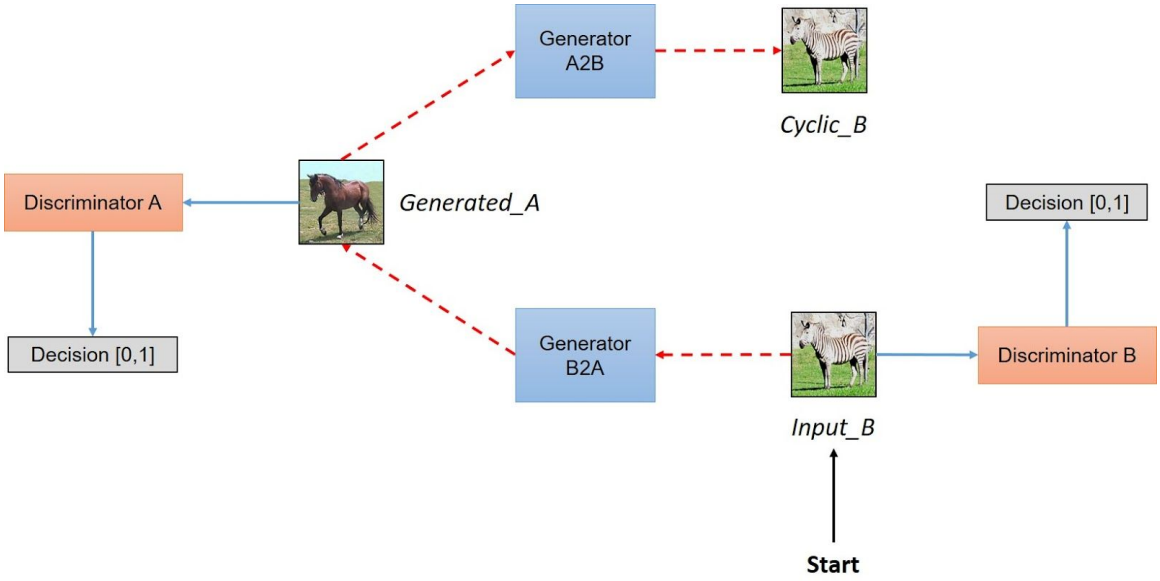
3 CycleGAN [arxiv 1703.10593, 2017.]

To overcome the need for training pairs, and simply use single domain samples the authors of the paper came up with a method and named it CycleGAN. The image-to-image translation is a class of vision and graphics problems where the goal is to learn the mapping between an input image and an output image using a training set of aligned image pairs. However, for many tasks, paired training data will not be available. We present an approach for learning to translate an image from a source domain X to a target domain Y in the absence of paired examples. Our goal is to learn a mapping $G : X \rightarrow Y$ such that the distribution of images from G(X) is indistinguishable from the distribution Y using an adversarial loss. Because this mapping is highly under-constrained, we couple it with an inverse mapping $F : Y \rightarrow X$ and introduce a **cycle consistency loss** to enforce $F(G(X)) \approx X$ (and vice versa). Thus it allows us to use unpaired training data. This means that in order to train it to translate images from domain X to domain Y , we do not have to have exact correspondences between individual images in those domains.
Thus, CycleGANs enable learning a mapping from one domain X (say, images of horses) to another domain Y (images of zebras) without having to find perfectly matched training pairs.

To summarize the differences between paired and unpaired data, we have:
• Paired training data: : $\{(x^{(i)}, y^{(i)})\}_{N_{i=1}}$
• Un-paired training data:
– Source set: $\{x^{(i)}\}_{N_{i=1}}$ with each $x^{(i)} \in X$
– Target set: $\{y^{(j)}\}_{M_{j=1}}$ with each $y^{(j)} \in Y$
– For example, X is the set of horse pictures, and Y is the set of zebra pictures, where there are no direct correspondences between images in X and Y

The basic architecture of the Network can be shown as





In a nutshell, the model works by taking an input image from domain D_A which is fed to our first generator $Generator_{A \to B}$ whose job is to transform a given image from domain D_A to an image in target domain D_B. This new generated image is then fed to another generator $Generator_{B \to A}$ which converts it back into an image, $Cyclic_A$, from our original domain D_A (think of autoencoders, except that our latent space is D_t). This output image must be close to the original input image to define a meaningful mapping that is absent in unpaired dataset.

The application of CycleGAN's are

- Resurrecting Ancient Cities
- Summer to Winter
- Horses to Zebras
- Apples to Oranges
- Monet into Photos
- Monet into Thomas Kinkade
- Animal Transfiguration
- Face to Ramen
- Cats to Dogs

The rendition of this implemented in this project is **Emoji CycleGAN**.

A CycleGAN used to translate emojis between two different styles, in particular, **Windows ↔ Apple** emojis and vice versa

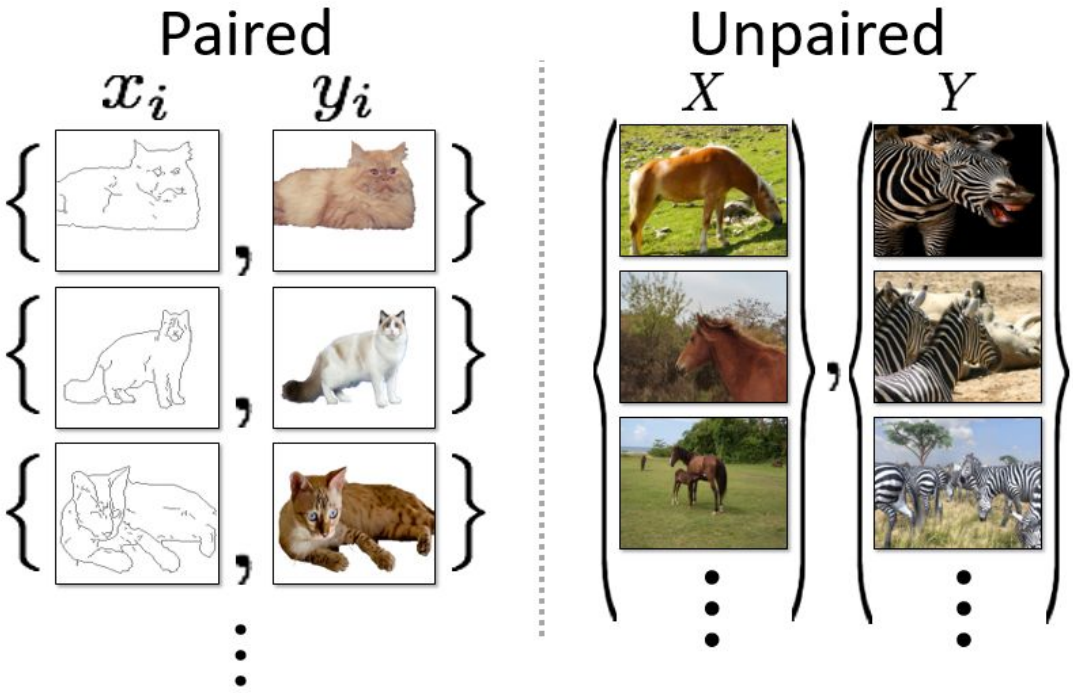
4 Why CycleGAN over ConditionalGAN?

The main reason behind picking CycleGAN over ConditionalGAN was better applications and the elimination of needing to have Image Pairs to train the GAN. Also, the cyclic consistency loss equation implemented to train the same. ConditionalGAN is a supervised method with the need for pairs of images to train, whereas CycleGAN is an unsupervised approach to the same with no need for pairs of images of both domain and the ability to train off images of each domain separately.

To learn from samples of only one domain and calculate loss from the image converted back to the same domain by the generator by testing similarity. The basic idea behind the CGAN-based approaches is to use a conditional GAN to learn a mapping from input to output images. The loss functions of these approaches generally include extra terms (in addition to the standard CGAN loss), to express constraints on the types of

images that are generated.

The method of image-to-image translation called CycleGAN is particularly interesting because it allows us to use unpaired training data. This means that in order to train it to translate images from domain X to domain Y, we do not have to have exact correspondences between individual images in those domains. For example, in the paper that introduced CycleGANs, the authors are able to translate between images of horses and zebras, even though there are no images of a zebra in exactly the same position as a horse, and with exactly the same background, etc.



For methods to tackle the unpaired setting, where the goal is to relate two data domains: X and Y, without pairs of domain X in domain Y or pairs of domain Y in domain X. The introduction of Cyclic Consistency Loss makes the above possible.

Cyclic Consistency

The most interesting idea behind CycleGANs (and the one from which they get their name) is the idea of introducing a cycle consistency loss to constrain the model. The idea is that when we translate an image from domain X to domain Y and then translate the generated image back to domain X, the result should look like the original image that we started with. The core idea of a Cyclic Consistency can be shown as

$$\mathbb{E}_{b\sim q(b|a)}\left[q(a'|b)\right] \approx \mathbb{E}_{b\sim p(b|a)}\left[p(a'|b)\right] \forall a$$

Where In the case of a Deterministic Mapping:

$$G_{BA}(G_{AB}(a)) = a$$

$$G_{AB}(G_{BA}(b)) = b$$

The cycle consistency component of the loss is the mean squared error between the input images and their reconstructions obtained by passing through both generators in sequence (i.e., from domain X to Y via the X → Y generator, and then from domain Y back to X via the Y → X generator). The cycle consistency loss

for the Y → X → Y cycle is expressed as follows:

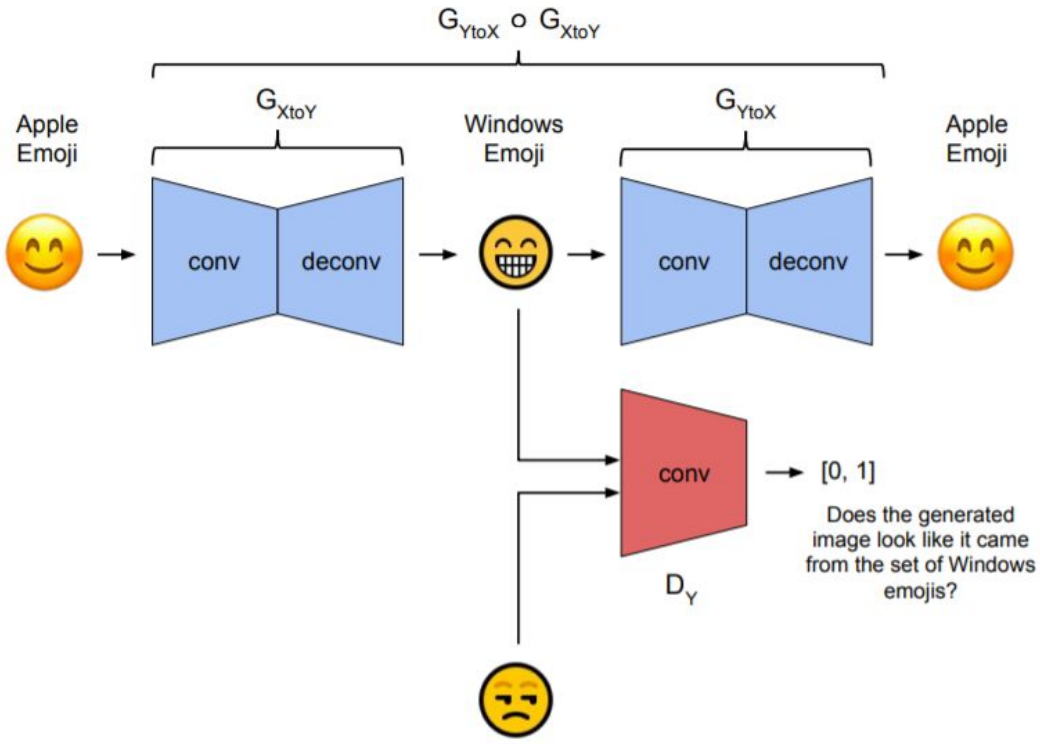
$$\frac{1}{m}\sum_{i=1}^{m}(y^{(i)} - G_{X\to Y}(G_{Y\to X}(y^{(i)})))^2)$$

The loss for the X → Y → X cycle is analogous.

In very simple terms, Cyclic-consistency is if we transform from source distribution to target and then back again to source distribution, we should get samples from our source distribution.

5 Emoji CycleGAN

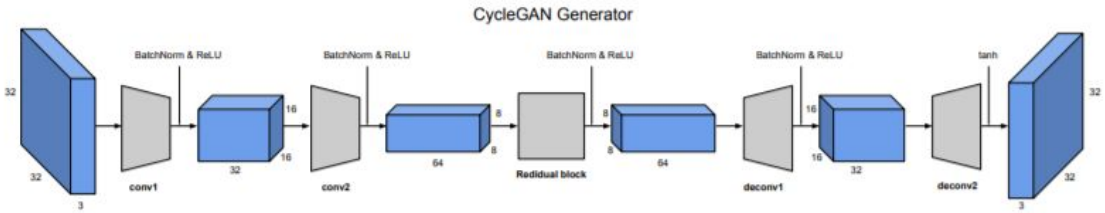
We'll build a CycleGAN and use it to translate emojis between two different styles, in particular, Windows ↔ Apple emojis.
The high-level architecture of the Emoji CycleGAN can be shown as



Our Dataset is a number of Images for Test and Train in each domain, each unique emoji represented by filenames of its ASCII Code. We have our final dataset with
Apple Train - 2270 Images
Windows Train - 2014 Images
Apple Test - 100 Images
Windows Test - 100 Images

CycleGAN Generator

The generator in the CycleGAN has layers that implement three stages of computation: 1) the first stage encodes the input via a series of convolutional layers that extract the image features; 2) the second stage then transforms the features by passing them through one or more residual blocks; and 3) the third stage decodes the transformed features using a series of transpose convolutional layers, to build an output image of the same size as the input.
The residual block used in the transformation stage consists of a convolutional layer, where the input is added to the output of the convolution. This is done so that the characteristics of the output image (e.g., the shapes of objects) do not differ too much from the input.
This generator can be seen as -



CycleGAN Training

The Training Loop for CycleGAN as seen from the paper can be shown as

Algorithm 2 CycleGAN Training Loop Pseudocode

procedure TRAINCYCLEGAN

Draw a minibatch of samples $\{x^{(1)}, \ldots, x^{(m)}\}$ from domain X

Draw a minibatch of samples $\{y^{(1)}, \ldots, y^{(m)}\}$ from domain Y

Compute the discriminator loss on real images:

$$\mathcal{J}^{(D)}_{real} = \frac{1}{m}\sum_{i=1}^{m}(D_X(x^{(i)}) - 1)^2 + \frac{1}{n}\sum_{j=1}^{n}(D_Y(y^{(j)} - 1)^2$$

Compute the discriminator loss on fake images:

$$\mathcal{J}^{(D)}_{fake} = \frac{1}{m}\sum_{i=1}^{m}(D_Y(G_{X\to Y}(x^{(i)})))^2 + \frac{1}{n}\sum_{j=1}^{n}(D_X(G_{Y\to X}(y^{(j)})))^2$$

Update the discriminators

Compute the $Y \to X$ generator loss:

$$\mathcal{J}^{(G_{Y\to X})} = \frac{1}{n}\sum_{j=1}^{n}(D_X(G_{Y\to X}(y^{(j)})) - 1)^2 + \mathcal{J}^{(Y\to X\to Y)}_{cycle}$$

Compute the $X \to Y$ generator loss:

$$\mathcal{J}^{(G_{X\to Y})} = \frac{1}{m}\sum_{i=1}^{m}(D_Y(G_{X\to Y}(x^{(i)})) - 1)^2 + \mathcal{J}^{(X\to Y\to X)}_{cycle}$$

Update the generators

CycleGAN Experiments

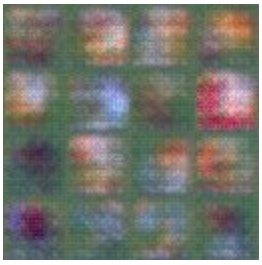
Training the CycleGAN from scratch is time-consuming without a GPU. Thus the training of the CycleGAN was done for 13Hours only and the results derived are of the same.
The Experiments are done on the inclusion and exclusion of Cyclic Consistency Loss to observe the effect it has on the results.

6 Results

The Results shown below represent the First and Last Iteration of the particular experiment and the results that were obtained from them.

1. **Vanilla GAN**

   We Train a Vanilla GAN on one domain and try to sample the other domain.The results for Iteration 200 and Iteration 5600 are given below



Sample-000200

Sample-005600

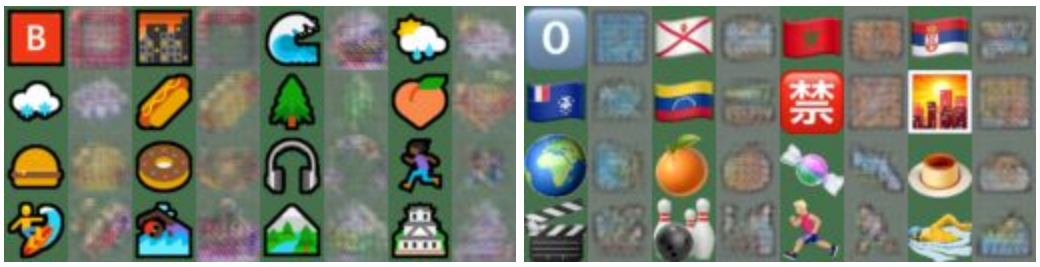
2. **CycleGAN without Cyclic Consistency Loss**

   We train a CycleGAN without Cyclic Consistency Loss and observe the following results for First and Last Iteration

```
Iteration [  600/  600] | d_real_loss: 0.3465 | d_Y_loss: 0.0602 | d_X_loss: 0.1440 | d_fake_loss: 0.2042 | g_loss: 0.7077
```

X-Y-X

First Iteration Y-X-Y

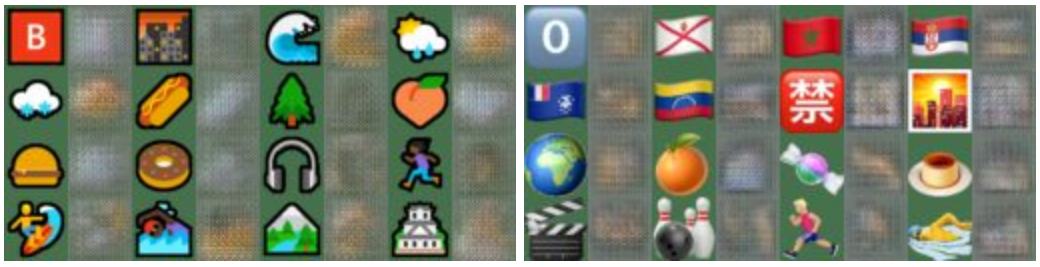


X-Y-X Last Iteration Y-X-Y

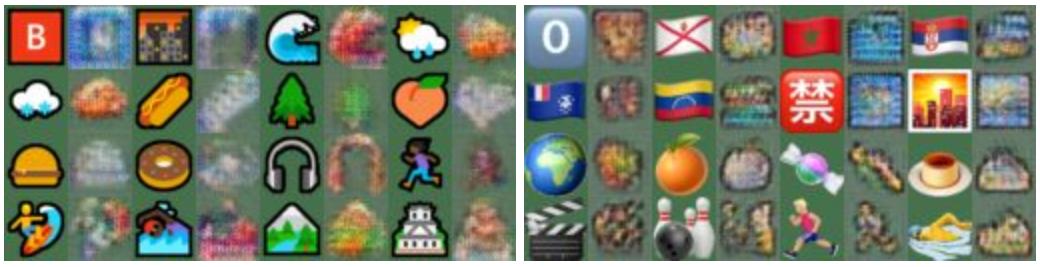
3. CycleGAN with Cyclic Consistency Loss

We train a CycleGAN with Cyclic Consistency Loss and observe the following results for First and Last Iteration

```
Iteration [  600/  600] | d_real_loss: 0.3765 | d_Y_loss: 0.0401 | d_X_loss: 0.1865 | d_fake_loss: 0.2266 | g_loss: 0.8546
```



X-Y-X First Iteration Y-X-Y



X-Y-X Last Iteration Y-X-Y

4. CycleGAN with pre-trained models without Cyclic Consistency Loss

We train a CycleGAN with pre-trained models without Cyclic Consistency Loss and observe the following results for First and Last Iteration

```
Iteration [  100/  100] | d_real_loss: 0.0000 | d_Y_loss: 0.0103 | d_X_loss: 0.0038 | d_fake_loss: 0.0141 | g_loss: 0.9986
```

X-Y-X First Iteration Y-X-Y

X-Y-X Last Iteration Y-X-Y

5. CycleGAN with pre-trained models with Cyclic Consistency Loss

We train a CycleGAN with pre-trained models with Cyclic Consistency Loss and observe the following results for First and Last Iteration

```
Iteration [  100/  100] | d_real_loss: 0.0000 | d_Y_loss: 0.0000 | d_X_loss: 0.0000 | d_fake_loss: 0.0000 | g_loss: 1.0294
```

X-Y-X First Iteration Y-X-Y

X-Y-X Last Iteration Y-X-Y

7 Conclusion

We see the results obtained by our Emoji CycleGAN on Emoji Dataset and the obtained results are comparable to the original image X. During training we noticed that the output results were sensitive to initialization. We do a domain transfer from X to Y and use the image Y to generate X back and calculate the loss from the same. The samples shown above prove the working of the method. We also show the importance of Cyclic Consistency Loss on implementation of CycleGAN and have results to back that up. CycleGAN being an unsupervised solution to Image to Image translation perform fairly well compared to the supervised pix2pix.

Acknowledgements

This report was prepared in accordance with the Project E : Image-to-Image Translation with Conditional Adversarial Networks and the pdf of the book provided, pix2pix implementation and original paper, cycleGAN implementation and original paper, GAN, CyclicConsistency and LossFunction on Wikipedia.

References

[1] pix2pix - https://phillipi.github.io/pix2pix/

[2] pix2pix Paper - https://arxiv.org/pdf/1611.07004.pdf

[3] CycleGAN - https://junyanz.github.io/CycleGAN/

[4] CycleGAN Paper - https://arxiv.org/pdf/1703.10593.pdf

[5] CycleGAN Blog - https://hardikbansal.github.io/CycleGANBlog/