Gender Transformation using Adversarial Networks

Rahul Ethiraj
University of Southern California
ethiraj@usc.edu

Sithara Kamalakkannan
University of Southern California
kamalakk@usc.edu

Abstract

The most common application of Generative Adversarial Networks (GANs) has been in image generation. The objective of this project is to generate gender transformation in a face image data set. That is, given an image of a male, the generator network is expected to produce a realistic version of the same person as a female and vice versa. This objective motivated the study towards the usage of GANs in image translation.

1 Introduction

Generative Adversarial Networks were introduced by Dr. Ian Goodfellow and other researchers from the University of Montreal. A generative model learns the joint probability distribution p(x,y) whereas discriminative model learns the conditional probability distribution p(ylx). Discriminative models have been quite popular especially for classification tasks but with the introduction of GANs generative models are also gaining significance.

2 Methodology

GANs are unsupervised learning algorithms that try to understand the underlying structure of the data. GANs consist of two neural networks (D,G) that compete against each other by playing a minimax game.

$$minmaxV(D,G) = E_{x\ pdata)(x)}[logD(x)] + E_{z\ pz(z)}[log(1 - D(G(z)))] \quad (1)$$

where, the log probability of D is predicting that the real-world data is genuine and log probability of D is predicting that G's generated data is not genuine.

The generator G, takes a random set of inputs and tries to generate the data while the discriminator D, takes in the data and samples from the generator and tries to determine if it is "fake" (created by the generator network) or "real" (from the original data set). The name therefore comes from the fact that the two networks are fighting against each other as adversaries.

3 Data set Description

The CelebFaces Attributes dataset (CelebA) that was used for training, is a large-scale face attributes data set. It consists of more than two hundred thousand celebrity images. The images in the data set cover large pose variations and background clutter. It consists of:

- 202,599 number of face images
- 40 binary attributes annotations per image

The CelebA data set was also used for testing. However, for testing the same trained model was tested with other data sets such as IMDB and WIKI face data set. The IMDB-WIKI data set consists of 500k+ face images with age and gender labels. However, due to the labels being too noisy since they were auto-generated, only 200 images from this were used to test how well the generator performs. A few images of INF-552 Summer 2018 classmates were also used for testing.

4 The Learning Process

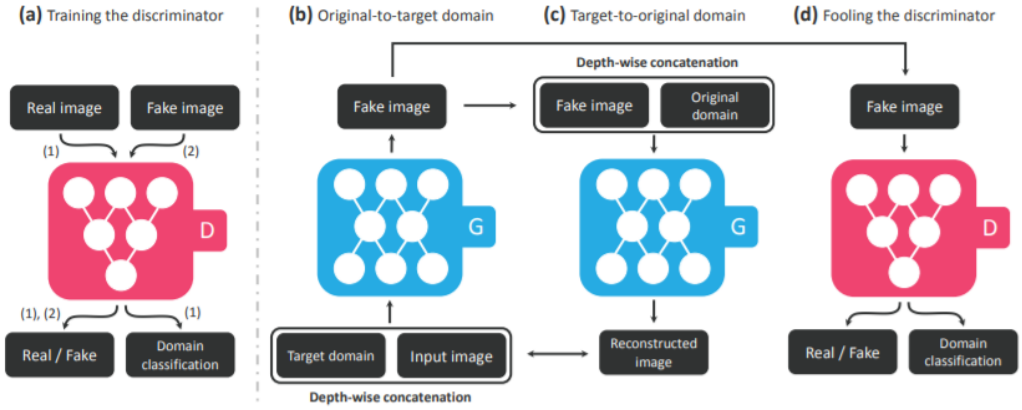
Implementing gender transformation in images, required studying about the different types and applications of GANs. There exists several image generation algorithms using GANs. Many of them are implemented using MxNet API in python. A combination of Convolution Neural Networks and GANs which gives rise to DCGANs can be used for the generation of face images. HyperGans, implemented using Tensorflow can be used for applications such as auto encoding to reconstruct images, classification, alignment of images, etc

Two such GANs that play an important role in Image Translation include : Conditonal Adversarial Nets [3] that are networks that can be extended to a conditional model if both the generator and discriminator are conditioned on some extra information y. y could be any kind of auxiliary information, such as class labels or data from other modalities. This could be used to generate images conditioned on the gender label for our application.

CycleGANs [2] can learn great translations without having explicit X/Y training images. This is done by introducing the idea of a full translation cycle to determine how good the entire translation system is. This improves both generators at the same time. One example to understand this is English-Spanish translation. That is, given an English phrase and it's English-to-Spanish translation, if you take the translated result and re-translate it through a complementary Spanish-to-English translator, a full cycle of translation will be obtained.

Finally, StarGans[1] that were published in 2017 seemed to be the best solution for the problem of Gender Transformation. StarGans which has the above two GANs as it's baseline models, works well for multi domain image-image translations.

5 StarGan Architecure



5.1 Novelty of StarGAN

The improvement that StarGAN has over other implementations of GANs is that it uses a single generator-discriminator model to define a multi domain image translation problem. When originally Gender Translation required the implementation of one model that can be trained to convert male images into their female counterparts, and one model to convert female images to their male counterparts, with StarGAN a single model is sufficient to do both these conversions.

This is especially evident for the use-case taken in the paper. For a given data set of faces that have multiple domains such as *gender, hair color, age*, the StarGan defines one unified model that can learn translations between any combination of these features into any combination for the output.

5.2 Steps taken in the Architecture

- Training the Discriminator:
  The discriminator is trained as in any GAN by providing it with both real images from the data set and fake images from the generator and it learns to classify them as real or fake. Along with it, the discriminator is trained to determine the domain classification of the image, that is *female or male* in this use-case.
- Training the Generator:
  The generator is given an input image and a target domain. The novelty in this approach is that the attribute values for the target domain are generated randomly. This way, by training the generator to be able to convert any given domain into any other target domain randomly, the model becomes very flexible and is not prone to over-fitting.
- Enhancing the Generator:
  In order to ensure that the generated fake images preserve the content of the input image, the generated fake image is compared with the original domains and a reconstruction loss is used to update the weights of the model.

5.3 Network Architecture

An overview of the Network Architecure:

```
Generator(
  (main): Sequential(
    #DownSampling
    Conv2d(4, 64, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3), bias=False)
    InstanceNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    ReLU(inplace)
    Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    InstanceNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    ReLU(inplace)
    Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    InstanceNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    ReLU(inplace)
    #BottleNeck
    (6) ResidualBlock(
    (main): Sequential(
        Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        InstanceNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        ReLU(inplace)
        Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        InstanceNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    #UpSampling
    ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    InstanceNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    ReLU(inplace)
    ConvTranspose2d(128, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    InstanceNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    ReLU(inplace)
    Conv2d(64, 3, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3), bias=False)
    Tanh()
  )
)
The number of parameters: 8417984
```

```
Discriminator(
  (main): Sequential(
    #InputLayer
    Conv2d(3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    LeakyReLU(negative_slope=0.01)
    #HiddenLayer
    Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    LeakyReLU(negative_slope=0.01)
    Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    LeakyReLU(negative_slope=0.01)
    Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    LeakyReLU(negative_slope=0.01)
    Conv2d(512, 1024, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    LeakyReLU(negative_slope=0.01)
    Conv2d(1024, 2048, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    LeakyReLU(negative_slope=0.01)
  )
  #OutputLayer
  (conv1): Conv2d(2048, 1, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (conv2): Conv2d(2048, 1, kernel_size=(2, 2), stride=(1, 1), bias=False)
)
The number of parameters: 44729280
```

5.4 Evaluation Criteria

The evaluation metrics used include:

Adversarial Loss

$$L_{adv} = E_x\left[logD_{src}(x)\right] + E_{x,c}[log(1 - D_{src}(G(x,c)))] \tag{2}$$

where, D_{src} refers to the probability distribution of the sources given to D and G(x,c) is the fake image from the generator conditioned on target domain c. The Discriminator tries to maximize this in the objective while the Generator tries to minimize it.

Domain Classification Loss

The loss that the Discriminator needs to minimize to correctly classify images to their original domain c' is given by

$$L^r_{cls} = E_{x,c'}[-logD_{cls}(c'/x)] \tag{3}$$

The loss that the Generator needs to minimize to generate images that can be classified to their target domain c is given by

$$L^f_{cls} = E_{x,c}[-logD_{cls}(c/G(x,c)] \tag{4}$$

Reconstruction Loss

L1 norm is taken as the reconstruction loss over G that takes in the translated image G(x, c) and the original domain label c as input and tries to reconstruct the original image x

$$L_{rec} = E_{x,c,c'}[||x - G(G(x,c),c')||_1] \tag{5}$$

Objective Function

$$L_D = -L_{adv} + \lambda_{cls} * L^r_{cls} \tag{6}$$

$$L_G = L_{adv} + \lambda_{cls} * L^f_{cls} + \lambda_{rec} * L_{rec} \tag{7}$$

where, λ_{cls}, λ_{rec} are hyper-parameters that control the relative importance of domain classification and reconstruction losses

6 Improvements to the Model

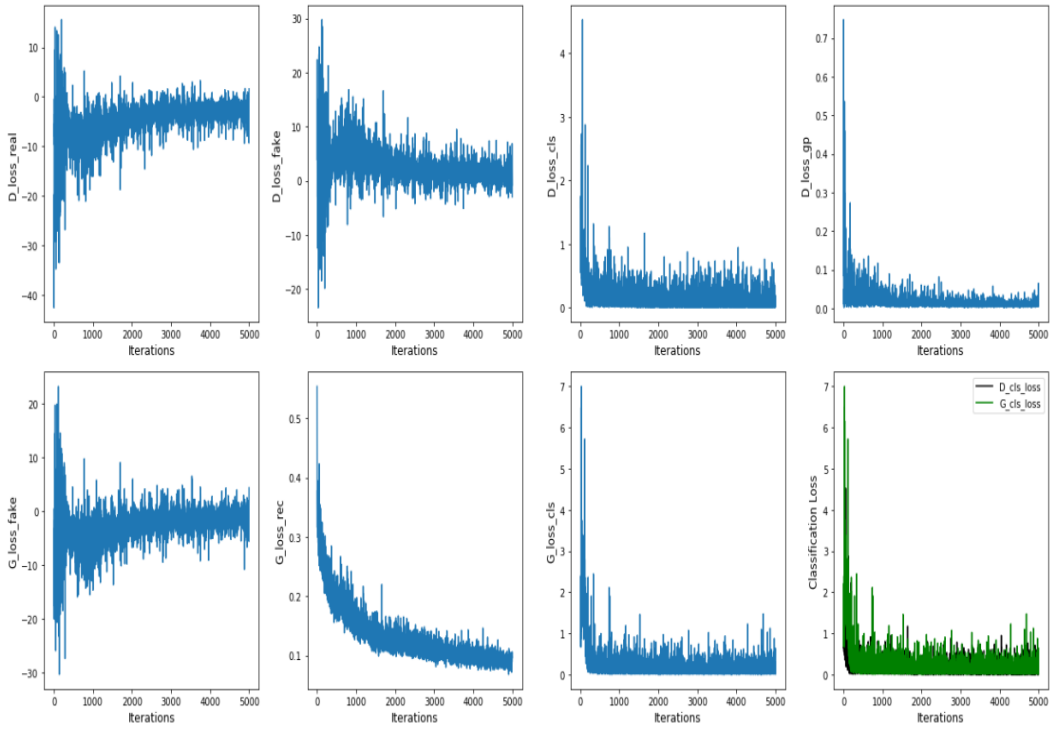
- Changing the Method of Creating Target Labels:
  In [1], the target labels were created randomly to ensure that multi domain to domain image translation is flexible. However, for the use-case in just converting Male to Female and Female to Male, instead of generating the target labels randomly, the target labels for the single model were generated by inverting the original domain labels. That is, it was ensured that the model gets trained to inverse the given image's domain label. If the original label is 1 for a Male Image, then the target label for the generator will be 0 for it to generate the Female version of the same image and vice versa.
- Plotting graphs to Evaluate the Model:
  Graphs were plotted to understand the loss in categorizing real and fake images correctly to their respective domain labels, and the classification loss of the Generator and Distributor. The classification losses converged approximately to 0.5 at 80K iterations.
- Changing the Testing Data set:
  The testing data set was changed to include 200 images from the IMDB, Wiki data set and a few images of the students in INF-552 Summer 2018 batch to evaluate the performance of the StarGan model. Ideally, the training set should have been changed but the reason for not doing so is mentioned under section 7.
- Modifying testing function to Evaluate without Labels:
  In the implementation of the code in the paper [1], the Testing mode had required the labels of original domain. However, due to the lack of accurate labels, the code was enhanced to calculate labels based on the domain classification probability of the discriminator. This label was then inversed to generate the target domain label.

7 Difficulties Faced

- Compatibility Issues between Tensorflow-gpu, CUDA, CDNN, VisualStudio, pyTorch.
- Changing the Data set:
  The unavailability of huge Face data sets that have gender labels was a major issue in trying to change the data set to evaluate the performance of StarGan. IMDB, Wiki, CMU Multi-PIE, Yale Face, UTKFacse Databases were all alternate data sets that training could have been done on. However all of these either had noisy labels, or less people with more augmented images or pictures of babies. None of these could be used for the problem of Gender Transformation. Therefore, the only option left was to change the testing data with IMDB, Wiki images without labels.
- Shortage of Computational Resources
  On a laptop with 6GB Nvidia GeForce GTX1060 GPU, core-i7 7700HQ CPU @ 2.81GHz, it took two days to train 200K images with 80K iterations.
- Challenges in using USC High Performance Computing
  The maximum memory space permitted in the root directory is 1GB. However, anaconda installation itself requires 1.3GB and the steps to rectify this are quite tedious. The number of files that could be placed in the project space for the HPCs is only 10K while the training data set itself had over 200K images.
- Challenges in using Amazon Ec2
  Permissions had to be acquired for using GPU instances on AWS. PyYAML compatibility issues arose when trying to install tensorflow in the virtualenv created by AWS that already had pytorch and cuda installed.

8 Results

The problem faced in evaluating the performance of GANs is that this can only be done by human evaluation and does not have any set metrics or accuracy to measure by. However, below are a few graphs plotted to understand the overall performance of the model. It can be seen that $D_{_{src}}$, $D_{_{cls}}$ and $D_{_{rec}}$ for the Discriminator and Generator converge towards zero as the number of iterations increases, in the following figure.



The best and worst images based on Discriminator Classification Loss for the testing data over CelebA and IMDB data sets are shown in the below figure:

The improvement in the performance of the GAN is evident as the number of iterations increases, as per below figure.

Acknowledgments

We would like to thank our professor Dr Mohammad Reza Rajati for guiding us throughout this course and project. We would also like to thank our professor and University for giving us access to High Performance Computing systems.

References

[1] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, Jaegul Choo (2017) StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation *arxiv.org/pdf/1711.09020*

[2] https://www.microsoft.com/developerblog/2017/06/12/learning-image-image-translation-cyclegans/

[3] M. Mirza and S. Osindero. Conditional generative adversarial nets *arXiv preprint arXiv:1411.1784*,2014

[4] Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros (2018) *https://arxiv.org/pdf/1703.10593.pdf*

[5] Zhifei Zhang, Yang Song, Hairong Qi, Age Progression/Regression by Conditional Adversarial Autoencoder (2017)

[6] https://github.com/zackthoutt/face-generation-gan

[7] https://www.oreilly.com/ideas/building-deep-learning-neural-networks-using-tensorflow-layers

[8] https://github.com/255BITS/HyperGAN