Milestone Report: cGANs for Cartoon to Real-life Images

Kanya Satis 4777336 Pranjal Singh Rajput 4788087 Sonnya Dellarosa 4783344 Wenxuan Huang 4925777 Obinna Agba 4793765

Delft University of Technology
CS4180 Deep Learning
Group 25

1. Introduction

Image-to-image translation is a topic that has been tackled using many different techniques - both traditional computer vision methods as well as more recent deep learning approaches.

In this project, focus is shifted to Pix2Pix - a conditional generative adversarial network (cGAN) - which was designed to perform Image to Image translation. GANs typically consist of two networks: a generator and a discriminator. The generator attempts to create *fakes* of the input data distribution from a noise distribution while the discriminator classifies whether its input is fake i.e from the generated distribution or real i.e from the original distribution.

cGANs also combine the input data with the noise distribution to condition the resulting output to the original distribution. Pix2Pix applies this technique for Image to Image translation.

The advantage of this approach is that the Pix2Pix model can be applied to a varying degree of Image to Image translation problems without the need to design loss functions specific to the application domain. Already Pix2Pix has been applied to different Image translation problems with notable success. This project aims to apply the Pix2Pix model to a dataset consisting of cartoonized. Using the Pix2Pix model it should be possible to train the network to generate real-life images from the cartoonized images. The questions this project seek to answer are:

1. Do the default set of hyper parameters of the original implementation work well on this cartoonized dataset?
2. Does the model properly recreate facial expressions and postures

In summary, this project hopes to estimate the robustness of the Pix2Pix model.

2. Problem statement

This experiment attempts to explore the robustness of the multi-purpose Pix2Pix model. The problem is to determine whether, under the standard hyper-parameters, can accurately recreate the facial features of 'real_B' (original distribution) images based on cartoonized 'real_A' (input data distribution) images.

2.1. Dataset

The dataset selected to train the Pix2Pix model is based on an adaption of the colorFERET [1] facial image data. The colorFERET database contains 11337 images of faces in different angles and postures. These images are taken from 1199 individuals and are typically used for facial recognition research.
The image data is provided in two versions: a higher resolution (512x768) and a lower resolution (256x384) version. For this project the lower resolution version was used.

2.1.1 Data Pre-Processing

As mentioned earlier, this project utilizes a derivation of the colorFERET images. To obtain the actual data for training/testing the network, a modified version of an openCV cartoonizer script [2] was run on the selected images to generate the cartoon version. The pair of cartoon and real images are then used as an input to the Pix2Pix model.

2.2. Expected Results

Motivated by many successful applications [6] of Pix2Pix to different kinds of data it is expected that the Pix2Pix model would perform similarly well - at least on test data similar to the training data.

2.3. Evaluation

One of the challenges cited by the original Pix2Pix paper was a method of evaluating the model. In their examples they used Amazon Mechanical Turks to evaluate if an image was real or fake. For a more quantitative evaluation they utilized a so-called *FCN* score. For this score the authors trained a fully convolution network to correctly label the original data with high confidence. This trained network

was then used to classify the generated samples with the expectation that it would accurately classify the generated data with a high confidence as well.
For this project however, only qualitative evaluation methods have been employed so far. Other quantitative evaluation methods - FCN score included are being considered.

3. Technical approach

3.1. cGANs

Generative Adversarial Network (GANs) [4] have been used to train generative models - models which can generate data from some input. GANs use an adversarial model where two networks: the discriminator (D) and the generator (G) play a sort of *minimax* game. The generator network attempts to generate *synthetic* data from a noise distribution and the discriminator estimates the probability that a data sample was generated from the original data distribution as opposed to the synthetic i.e generated distribution. Equation 1 below captures the function which this adversarial model tries to optimize.

$$\begin{aligned}\min_G \max_D V(G, D) = \mathbb{E}_{x\sim p_{data}(x)[\log D(x)]} \\ +\mathbb{E}_{z\sim p_z(z)[\log 1 - D(G(z))]}\end{aligned} \tag{1}$$

In equation 1, the model aims to maximize the probability that the discriminator D assigns the correct label - real or fake - to input data while also minimizing the probability that data generated by the generator G is classified as fake as denoted by $\log 1 - D(G(z))$.
Conditional GANs (cGANs) [8] differ from the GAN model described above in that the adversarial model is conditioned using some extra data. This conditioning has the effect on exerting more control (providing direction) on the data being generated. Equation details the optimization function for cGANs.

$$\begin{aligned}\min_G \max_D V(G, D) = \mathbb{E}_{x\sim p_{data}(x)[\log D(x\|y)]} \\ +\mathbb{E}_{z\sim p_z(z)[\log 1 - D(G(z\|y))]}\end{aligned} \tag{2}$$

3.2. Pix2Pix

Pix2Pix [7] uses a cGAN model for image to image translation, i.e. the task of translating one possible representation of a scene to another. In the case of Pix2Pix, the generator and discriminator models are conditioned using the input image itself.

The original Pix2Pix implementation modifies the optimization function of cGANs from equation 2 by adding an L1 loss term when training the Generator model. Equation 3 describes the optimization function for the Generator in Pix2Pix.

$$G^* = \arg\min_G \max_D L_{cGAN}(G, D) + \lambda L_{L1}(G) \tag{3}$$

where $L_{cGAN}(G, D) = \mathbb{E}_{z\sim p_z(z)[\log 1 - D(G(z\|y))]}$

and $\lambda L_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|]$.

The L1 loss in this case models the L1 per-pixel difference between the generated images and the ground truth. The authors picked L1 over L2 as the L2 distance produced more blurry results.

The authors also provided an implementation of the Pix2pix model on github [5]. This implementation was written in *lua*. However, this project uses a *pytorch* [11] implementation of the original paper.

In this implementation, a U-NET 256 network [9] was used as the generator network. For the discriminator a PatchGAN network [7] proposed by the authors was used. This network restricts the evaluation scope to patches instead of the full image i.e it attempts to classify if each NxN patch in the image is real or fake. The authors also showed that N can be made much smaller than the image size without affecting the results.

3.3. Training the Model

When training the model, input images were first scaled, cropped and/or re-sized to 256x256 pixels. This is to condition the input for the U-NET 256 network which requires 256x256 input images. Table 1 show all the hyper-parameters used in training the model.

Training was carried out on an Nvidia GTX 2070 GPU. The training dataset included 6803 images. Training is done for 200 epochs, which took approximately 15 hours to train on the above mentioned GPU configuration.

4. Preliminary results

4.1. Performance on Test Data

In this experiment, the trained model was applied to test data generated in the same way as the training data. As expected, the results were quite satisfactory with the model able to generate close enough samples of the original image (Figure 3 (a) and (b)) . However, from visual inspection, the performance on the test data was not as good as that observed during training. Also it was seen that with grayscale images, the network could not capture the facial expressions properly and the resulting image appears to have some blurry filter applied (Figure 3 (a)). In addition, images for which the face did not occupy a large portion of the image also produced poor results (Figure 3 (d)).
These bad result cases might be explained by the images not having enough data for an accurate translation - in the case

of the grayscale images, only one channel is available and in the case of images with a low face size/image size ratio, at the resolution being evaluated (256x256) there simply is not enough data for a proper reconstruction.

4.2. Testing for Robustness

In this experiment, the aim was to evaluate the model on test data generated differently from the training data to determine how robust the trained model was to out of distribution samples. The following test sets used were :

1. **Cartoon images obtained by other algorithms**: The openCV script used to generate the dataset was slightly modified and produced a different cartoonizing effect. Also, an online tool [3] was used to generate a different style of cartoon images. When these data were tested, the resulting images were of a lower visual quality by inspection when compared to results from the initial distribution. These results can be seen in Figure 2

2. **Hand-drawn sketches of original test data**: Hand-drawn sketches of the colorFERET database [10] are used to generate the real-life image version of the sketch. As the sketches are completely different from the training set, it was expected that the generated results would be different from the real pictures. From Figure 4, it would appear that the network removes the sharp edges from the sketch and blurs the resulting image. Curiously, it also failed to capture details like the eyes and ears.

4.3. Attempts at Evaluating Quality of Generated Images

The following methods were used to evaluate the result:

4.3.1 Visual inspection

The data is visually inspected and compared to the original image to see how close they are. This is similar to the Amazon Turk approach applied by the Pix2Pix authors albeit on a smaller scale and without assigning a score to the visual inspection.

4.3.2 L1 Distance

The L1 per-pixel distance was also used to quantitatively evaluate the network. The L1 loss represents per-pixel based similarity between generated and real images. As can be seen in Figure 5(c), the L1 loss fluctuates significantly and does not seem to steadily increase or decrease. The graph has been normalized by taking average values in a sliding window. It reaches its minimum after 50 epochs and rises after that despite the visual quality of the images improving with increasing number of epochs.

4.3.3 Discriminator Probabilities

D_{real} and D_{fake} represent the probabilities which the discriminator assigns to real and generated images. As is typical of GANs, the discriminator should not be able to distinguish between the real and generated data. As can be seen in Figures 5 (a) and (b), the discriminator correctly labels fake and real images with high confidence. However, as training continues there is a decrease in the confidence being assigned - indicative that it was becoming more difficult to distinguish both.

4.3.4 Feature Matching

In this methodology, a MATLAB script was developed to find and match SIFT features in the original to the fake images that were generated. It was seen that more matches were found in the similar test data as compared to the other data-sets. This is in-line with the visual inspection and our hypothesis as well. However, no quantitative scores have been derived using this approach and is being considered as an area requiring further exploration.

5. Reflection of topic and approach

From the results of this run, the results of the network meet the initial expectations. Also the results suggest that indeed the default configuration of the network can be used for this projects use case.
However, a better evaluation methodology is required. The current methods are either tedious or subjective. The next phase of the project would focus on exploring different methods of quantitative analysis.
Also, as was seen in Figure 3 where grayscale images and images with low face/image ratio produced bad results, a logical next step for the final phase would be to retrain the network with these kind of problematic images filtered out.
Finally, for the final phase, different parameters would be tweaked in search of the best configuration for this application.
It is our belief that given our understanding of the project at this time it would be possible to complete the above mentioned proposals.

References

[1] Color feret database. `https://www.nist.gov/itl/iad/image-group/color-feret-database`. Accessed: 2019-05-27.

[2] Michael Beyeler. Cartoon effect using opencv. `https://github.com/mbeyeler/opencv-python-blueprints/blob/master/chapter1/filters.py`. Accessed: 2019-05-27.

[3] Cloudinary and Jackie Rosenzveig. Cartoonize an image. `https://cloudinary.com/blog/help_users_`

toon_into_your_site_with_a_cartoon_effect. Accessed: 2019-05-27.

[4] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets, 2014.

[5] Phillip Isola. pix2pix on github. https://github.com/phillipi/pix2pix. Accessed: 2019-05-27.

[6] Phillip Isola. Pix2pix project page. https://phillipi.github.io/pix2pix/. Accessed: 2019-05-27.

[7] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks, 2018.

[8] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.

[9] Olaf Ronneberger, Philipp Fischer, and Thomas Brox Alexei. U-net: Convolutional networks for biomedical image segmentation, 2015.

[10] X. Wang and X. Tang. Face photo-sketch synthesis and recognition. http://mmlab.ie.cuhk.edu.hk/archive/cufsf, 2009. Accessed: 2019-05-27.

[11] Jun-Yan Zhu. Pytorch pix2pix on github. https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix. Accessed: 2019-05-27.

A. Appendix: Training Configuration and Hyper-Parameters

Table 1 below details the hyper-parameters of the network as well as other important training configuration.

Name	Value
Batch Size	1
Beta1	0.5
Beta2	0.9999
Discriminator Network	PatchGAN
Epsilon	1e-08
Generator Network	U-NET 256
Learning Rate	0.0002
L1 λ	100.0
Number of Test Images	6803
Optimization Method	Adam
Weight Decay	0

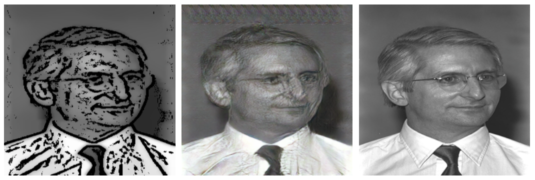
Table 1: Training Configration and Hyper-Parameters

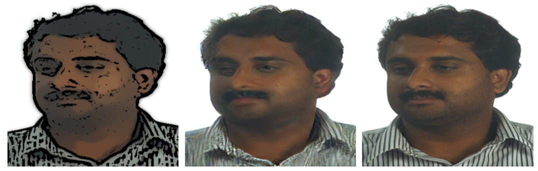
B. Appendix: Result samples

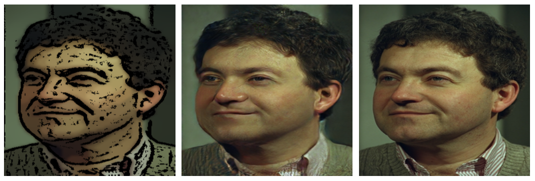
We tested the network on different types of datasets, and the results are shown in the figures below. In figure 1, first row shows the output in case of a black and white input image, second row shows the output in case of a face image without a background, third row shows the output in case of an image with the face covering the right proportion of the image, fourth row shows the output in case of an image with a distant object from the camera.

In figure 2, first row compares 2 different outputs, one from an input cartoon image generated using openCV2 library in python, and the other one is generated using an online software [3].

In figure 3, we compare the feature matching of outputs with the input in two different cases. Figure 4 shows the output in case of a hand sketched input image.


(a) Grayscale Cartoon: Input, Generated, Ground Truth


(b) Sample Result: Input, Generated, Ground Truth


(c) Sample Result: Input, Generated, Ground Truth

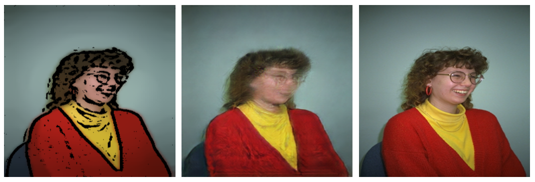

(d) Low Face/Image Ratio: Input, Generated, Ground Truth

Figure 1: Sample Results from Test data generated identical to training data

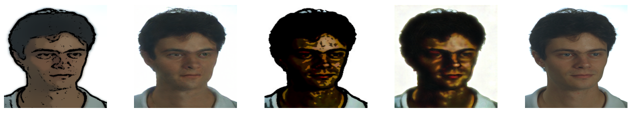
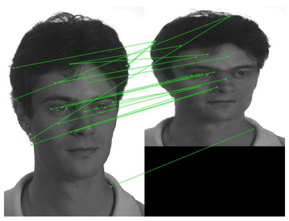

(a) Cartoon from modified algorithm (b) Generated from (a) (c) Cartoon Image from Online Service (d) Generated from (c) (e) Ground Truth

Figure 2: Results with different cartoonizations


(a) Feature matching for cartoon generated by algorithm 1

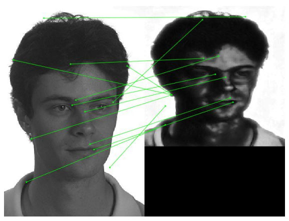
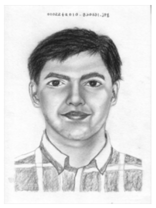
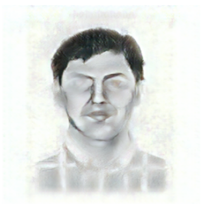

(b) Feature matching for cartoon generated by algorithm 2

Figure 3: Feature matching in 2 different cases


(a) Sketch Input


(b) Generated Image

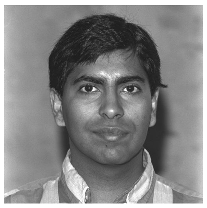
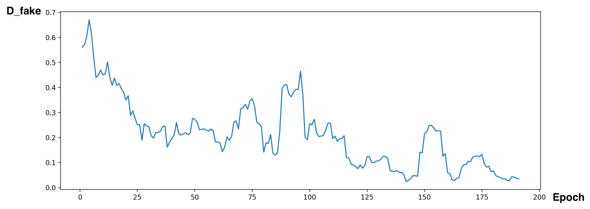
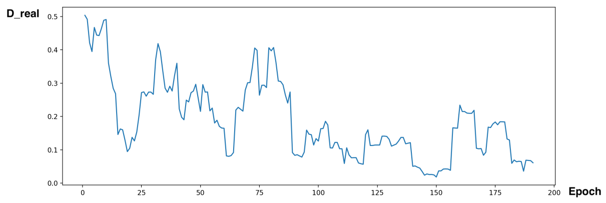

(c) Ground Truth

Figure 4: Image samples in case of hand drawn sketches

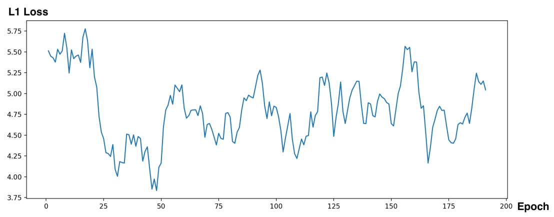
C. Result data



(a) Plot of Discriminator Probabilities for Fake (Generated)images



(b) Plot of Discriminator Probabilities for Real images



(c) Plot of L1 Loss for Generator

Figure 5: Visualizing the Training Results