Image to image translation with pix2pix

Shruti Bendale
Person Number: 50289048,
School of Engineering and Applied Sciences,
University at Buffalo, Buffalo, NY, 14214
shrutitu@buffalo.edu

1. Introduction

1.1. Generative Adversarial Networks:

In general, generative networks are unsupervised learning techniques that seek to learn the distribution of some data (e.g. words in a corpus or pixels in images of cats).

Briefly, GANs consist of two networks with opposing objectives, seeking an equilibrium in a sort of game being played between them. The "**Generator**" transforms some input that is sampled from what is called the "latent space into the output space that contains what we desire to generate. The "**Discriminator**" is simply a classifier that receives both outputs from the Generator, and real objects, then is trained to determine whether the input it is observing is synthetically generated, or real.

The idea is that when both networks are performing optimally, the Generator creates images that are distributed within their respective output space in the same way that real inputs to the Discriminator are.

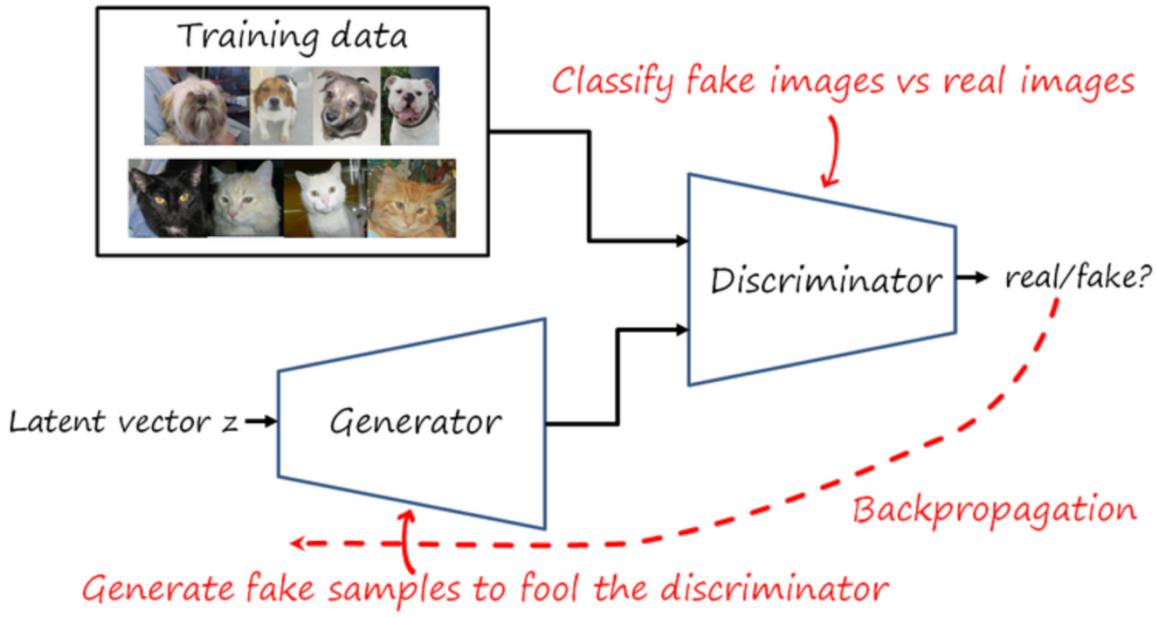


Figure 1.1: A Generative Adversarial Network

Let's now define an **optimization function for the GAN**. The Generator tries to maximize its probability of classifying an image correctly as real or fake. The Discriminator on the other hand tries to minimize the chances of the generator of classifying an image correctly as real or fake.

Mathematically, we define our objective function as

$$\min_G \max_D V(G,D) = \mathbb{E}_{y \sim p_{data}(y)}[logD(y)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

P_{data} -> The original data distribution as in the dataset

$x \sim P_{data}$ -> Data sampled from P_{data}

$z \sim P_{data}$ -> Data sampled from P_z

θ_g -> The parameters of the generator network

θ_d -> The parameters of the discriminator network

Term 1: $\mathbf{E}_{x \sim P_{data}}[\log(D(x; \theta_d))]$

$D(x; \theta_d)$ -> The likelihood of the image 'x' being real

$\log(D(x; \theta_d))$ -> The log likelihood of the image 'x' being real

$\mathbf{E}_{x \sim P_{data}}[\log(D(x; \theta_d))]$ -> The expected log likelihood with input samples from real data

`D(x)` represents the probability that the input image is real. Hence, the discriminator will have to maximize `D(x)` and `log(D(x))`. And hence, Term 1 has to be maximized.

Term 2: $\mathbf{E}_{z \sim P_z}[\log(1 - D(G(z; \theta_g); \theta_d))]$

$G(z; \theta_g)$ -> The generated image from noise sample 'z'

$D(G(z; \theta_g); \theta_d)$-> The likelihood of the image from Generator being real

$\log(D(G(z; \theta_g); \theta_d))$ -> The log likelihood of the image from Generator being real

$\log(1 - D(G(z; \theta_g); \theta_d))$ -> The log likelihood of the image from Generator being fake

The Generator has to maximize the chances of the discriminator getting fooled by the generated images. Which means, the generator should want to maximize `D(G(z))`. Which means, it should minimize `(1-D(G(z))` and hence minimize `log(1−D(G(z))`.

Some popular adversarial network architectures are:

a. **Conditional GANs**, that learn the distribution of output images given paired inputs for applications such as image-to-image translation.
b. **Cycle-Consistent GANs,** which can learn image-to-image mappings without requiring paired inputs.
c. **Deep Convolutional GANs**, that are also used to generate realistic images.

1.2 Conditional Generative Adversarial Networks:

In a GAN, creation starts from white noise. However, in the real world, what is required may be a form of transformation, not creation. Take, for example, colorization of black-and-white images, or conversion of aerials to maps. For applications like those, **we condition on additional input**: Hence the name, conditional adversarial networks.

This means that the generator is passed not (or not only) white noise, but data of a certain input structure, such as edges or shapes. It then has to generate realistic-looking pictures of real objects having those shapes. The discriminator, too, may receive the shapes or edges as input, in addition to the fake and real objects it is tasked to tell apart.

As a CGAN conditions the output data distribution based on a Condition layer, in the objective function, `log(1−D(G(z))` and `D(x)` will be replaced by `log(1−D(G(z|y))` and `D(x|y)`. Rest of it would be the would be taken care of by the respective networks, i.e., creating latent representations and managing weights. The main objective remains the same here with little modifications:

$$\min_G \max_D \left(\mathbb{E}_{y,x\sim p_{data}(y,x)}[logD(y,x)] + \mathbb{E}_{x\sim p_x, z\sim p_{z(z)}}\left[\log\left(1 - D(G(z,x),x)\right)\right]\right)$$

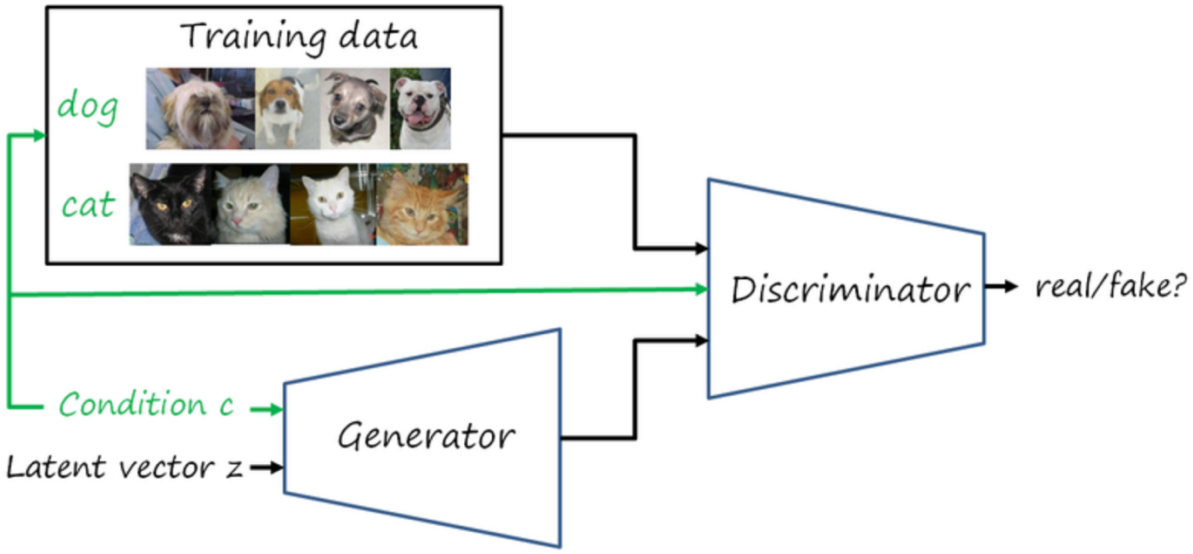


Figure 1.2: A general CGAN Architecture

1.3 Image to image translation with pix2pix:

1.3.1. Image to image translation:

Most tasks in image processing and computer vision can be seen as transforming one input image into an output one (e.g. **filtering, edge detection, image enhancement, colorization, restoration, denoising, semantic segmentation, depth extraction**). The term image-to-image translation has been used recently to refer to general purpose methods that learn transformations directly from datasets with pairs of input and output images. These are some examples:

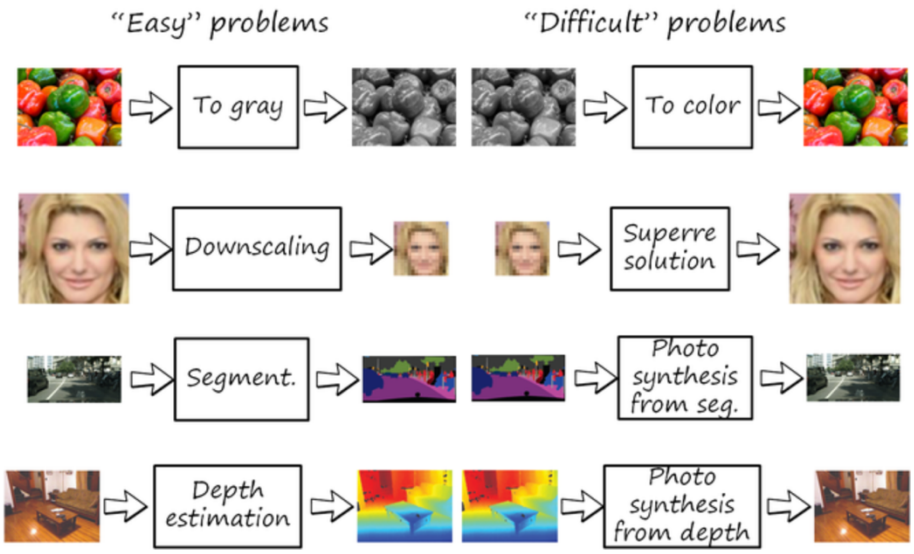


Figure 1.3: Examples of image to image translation

1.3.2. Paired image to image translation:

In many cases we can collect pairs of input-output images. For example, we can easily get edge images from color images (e.g. applying an edge detector), and use it to solve the more challenging problem of reconstructing photo images from edge images, as shown in the following figure:

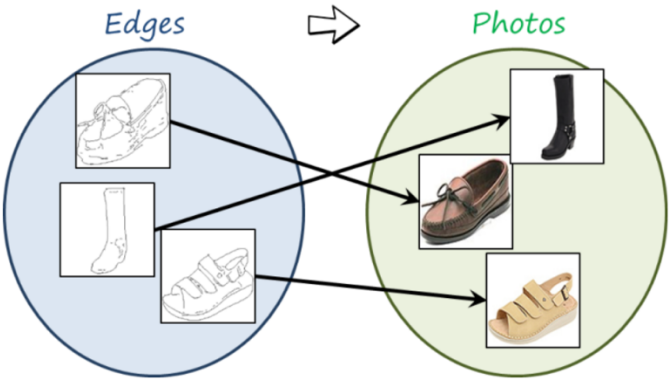


Figure 1.4: Paired image-to-image translation

1.3.2. pix2pix:

In Figure 1.4, we can **consider the edge image as an input condition and then use a conditional GAN to generate the output image**. This is the idea of **pix2pix**. The generator now is a bit more complicated, since it consists of an encoder followed by a decoder (implemented as a convolutional network followed by a deconvolutional network).

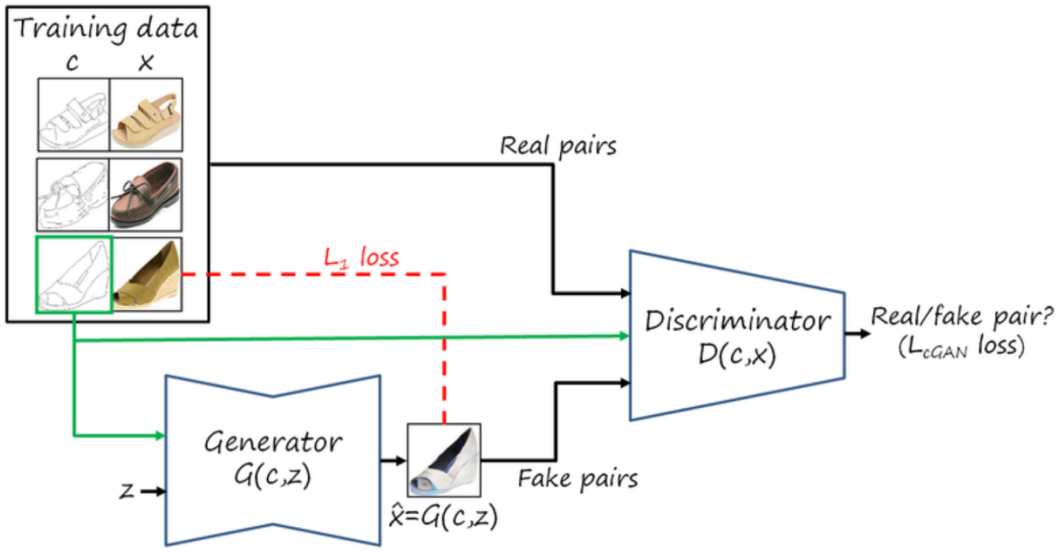


Figure 1.5: pix2pix: conditional GAN for paired image-to-image translation

The authors of the pix2pix paper design a general solution to this image to image translation problem, where the underlying method could be used on almost arbitrary data to perform generic translation across a wide range of image domains with no change to the underlying algorithm. Some of the applications of pix2pix are given below:

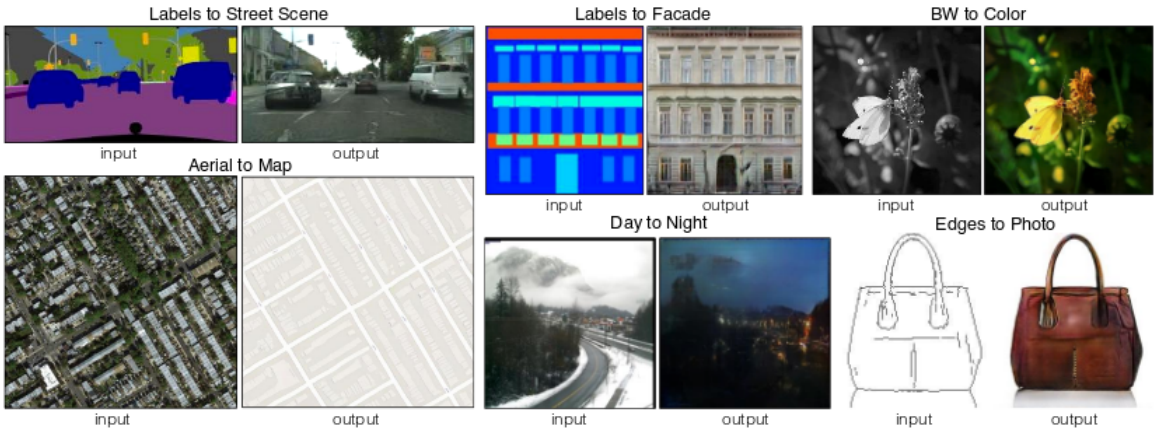


Figure 1.6: Few applications of pix2pix

Objective function:
The pix2pix combines the objective of a conditional GAN with a more traditional loss, such as L1 distance. The idea is that the discriminator's job remains unchanged, but the generator is tasked to not only fool the discriminator but also to be near the ground truth output in an L1 sense.

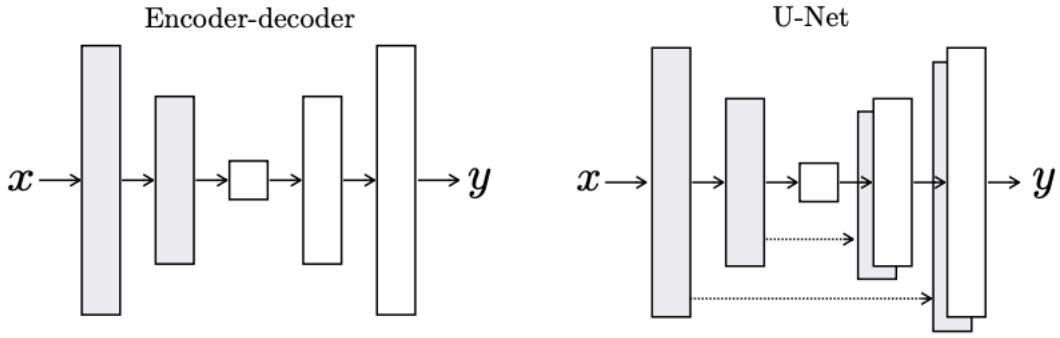
$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z))]$$

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1]$$

So, our final objective is,

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G)$$

Generator:
To give the generator a means to avoid a bottleneck condition, we add skip connections, following the general shape of a "U-Net. The "U-Net" is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks. Specifically, we add skip connections between each layer `i` and layer `n - i`, where n is the total number of layers. Each skip connection simply concatenates all channels at layer `i` with those at layer `n - i`.



Discriminator:
PatchGAN is used as a discriminator as it provides to generate sharper images. This discriminator tries to classify if each N ×N patch in an image is real or fake. We run this discriminator convolutionally across the image, averaging all responses to provide the ultimate output of D.

PatchGAN

The overall architecture of pix2pix along with the generator and the discriminator is given as follows:

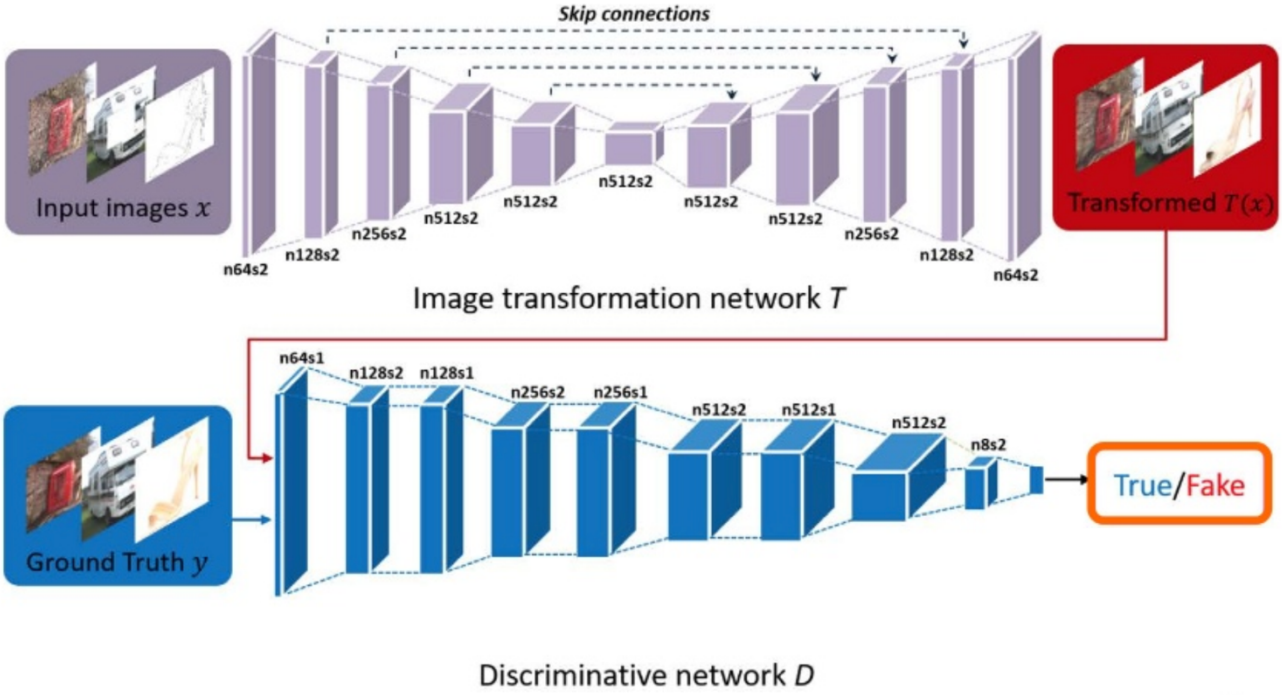


Figure 1.7: Architecture of pix2pix

2. Application

We use the pix2pix architecture and apply it for the purpose of **synthesizing photo-realistic images from the black-and-white sketch images of a *specific* person**.

The reasons for choosing human faces in this study are:

1. Such images or videos are abundant and easy to acquire.
2. Human facial expressions are fairly complex and a good subject for study.
3. We are instinctively sensitive to images of human faces, thus the bar for the experiments is naturally higher than using other types of images. This will allow us to spot problems in the experimental results more quickly.
4. Human faces involve precise geometric relationship among facial features (eyes, nose, etc.). As such, it is a good candidate for studying what it takes in order for a generative system like cGAN to discover feature structure at the instance-level, and not just probability distribution at the population level.
5. There are arguably more practical applications for human faces.

As a first step towards the long-term goal stated above, we choose to use cGAN for building the neural model over the faces of a *specific* person.

2.1. Goals of this experiment

Using human faces as the subject matter for a series of experiments, we seek to answer the following questions:

1. How far can we push cGAN to fill in satisfactory details when only scanty information is provided in the input image, using relatively small training dataset.
2. Overall is cGAN suitable for use as the basis to build the neural model of a specific person's face, representing the multitude of visual details regarding this person's face. For example, can cGAN be trained to accommodate artifacts in the test input image, to recover from aberrant input, recover from missing parts, etc.
3. Is the cGAN neural model of a person transferable to another person.
4. The usefulness of building a universal cGAN neural model for all human faces.

2.2. Dataset

1. The ground truth images are color photos of Barack Obama, manually scrapped from over the Internet.
2. The input images are gray scale images manually created from the ground truth color images using a photoshop filter.
3. Images are cropped to 256x256 pixel size.
4. Each pair of sketch and real image was combined to generate the final data [Figure 1.8]. The pix2pix repository provides a python script to generate training, validation and testing data in the form of pairs of images {A, B}, where A and B are two different depictions of the same underlying scene.

Figure 1.8: Example image given as an input for training.
Ground truth image(left) and input image of sketch(right)

2.3. Training

ngf, ndf defines the factor that manages the number of parameters in the generator and discriminator respectively. In the original paper the default is *ngf* = *ndf* = 64. In our implementation we use 32 to **reduce the number of parameters and enforce faster training time.**

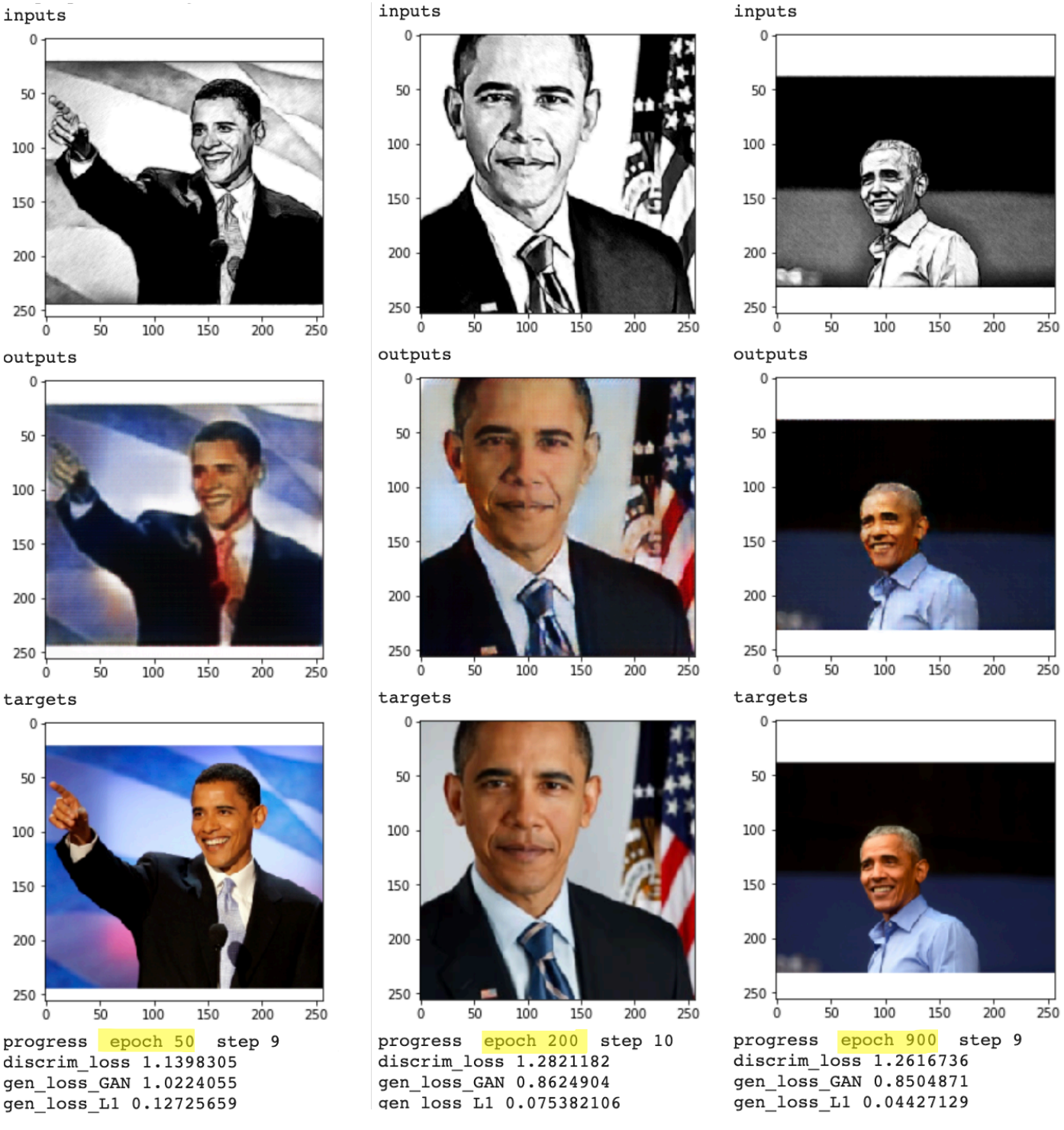
Since we are training with less parameters the visual outputs will be much worse than the original paper. To obtain much better results we introduce a simple trick that augments the training dataset with more images and hence create much detailed images. The idea is similar to pyramid scaling where we train on different random scales of each image. We choose a random scaling between (286, 300) then we apply random cropping back to (256, 256).

Training on different scales helps capturing details when using lesser number of parameters. This is done by transforming the input images using the additional transform function in the code.

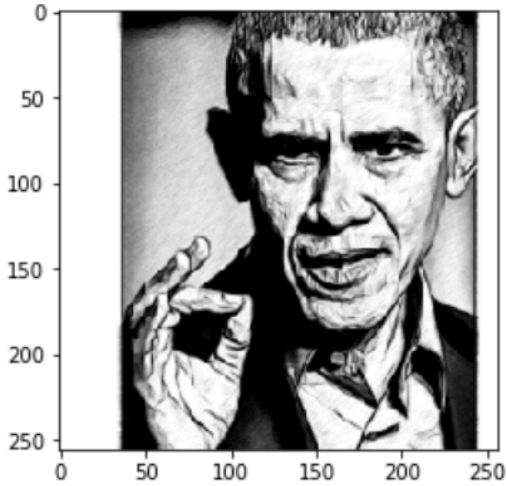
No. of images: 50 original images + images generated by scaling the original images
Number of epochs: 2000 epochs
Time taken for training: 35 minutes

2.4. Results:

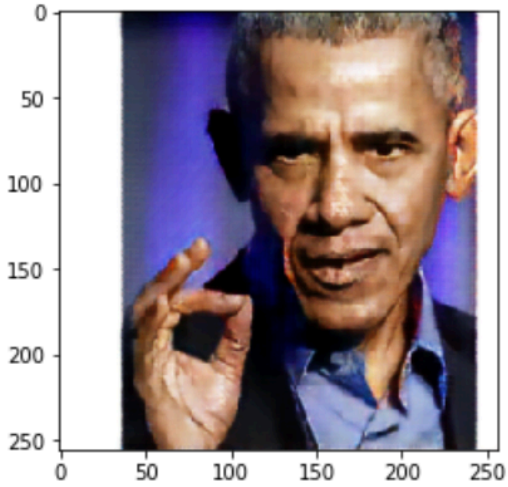
The input images, the output images and the target images as the epochs progress can be seen below:



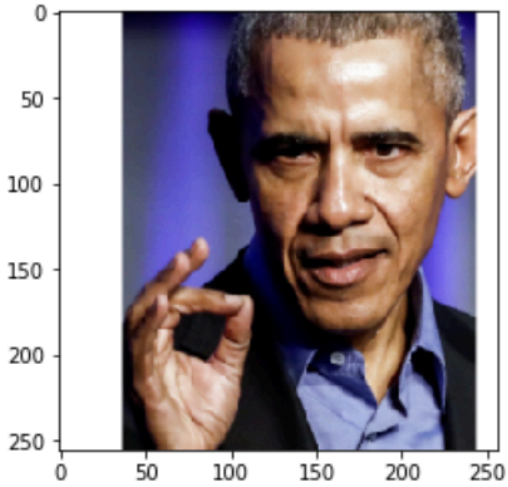
inputs



outputs

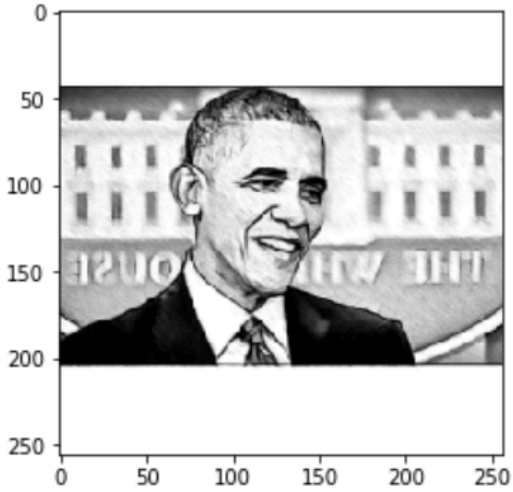


targets

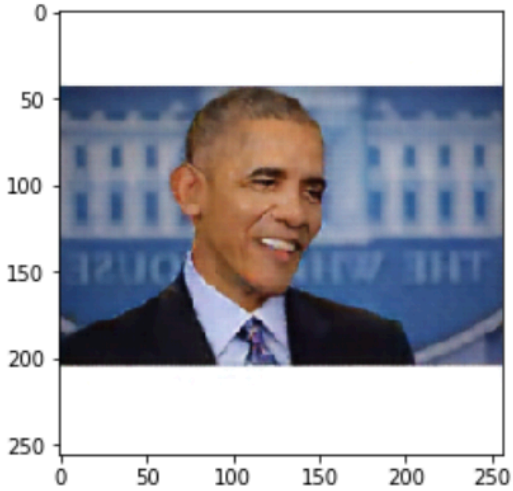


progress epoch 1650 step 9
discrim_loss 1.2123814
gen_loss_GAN 0.94057775
gen_loss_L1 0.039224416

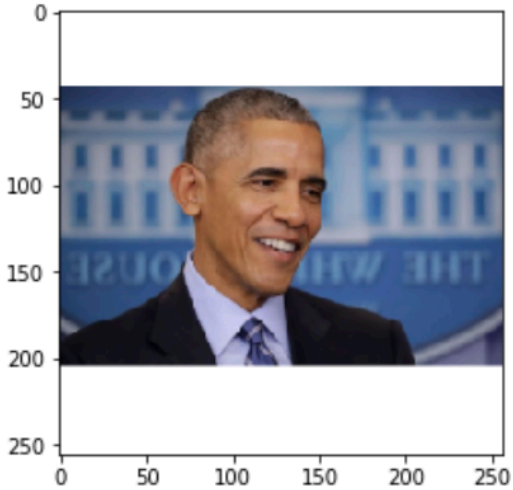
inputs



outputs



targets



progress epoch 2000 step 9
discrim_loss 1.2171968
gen_loss_GAN 1.0071417
gen_loss_L1 0.03817638

3. Conclusion

In this report we have presented a possibility of using cGAN as the basis for building a neural representation of human faces, with an eye towards applying the same technique to other types of physical objects in the future. The generalizability of pix2pix allows us to map from any given input image to another image, which gives us an ability to use cGAN in many ways.

The results in this paper and the results of our experiment suggest that conditional adversarial networks are a promising approach for many image-to-image translation tasks, especially those involving high structured graphical outputs.

Depending on what we want to achieve, one weakness could be the lack of stochasticity in the model, as stated by the authors of the paper themselves. This will be hard to avoid when working with paired datasets as the ones used in pix2pix. An interesting alternative is CycleGAN that lets you transfer style between complete datasets without using paired instances.

4. References

1. P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-toimage translation with conditional adversarial networks.In CVPR, 2017.
2. Image-to-Image Translation with Conditional Adversarial Nets. https://phillipi.github.io/pix2pix/
3. Fast pix2pix in the Browser. https://blog.usejournal.com/fast-pix2pix-in-the-browser-287d9858a5e4
4. Image-to-image translation with pix2pix. https://blogs.rstudio.com/tensorflow/posts/20 18-09-20-eager-pix2pix/