Unpaired Image-to-Image Translation using CycleGaN

Sara Shoouri, Elsa Mathew, Amogh Rane, Nidhi Sridhar
University of Michigan, Ann Arbor - 48105 - USA
sshoouri, elmathew, aarane, nidhisri@umich.edu

Abstract

This project aims to implement unpaired image-to-image translation by reproducing cycle-consistent adversarial networks (CycleGAN) model implemented by Jun-Yan Zhu et al. Using the Horse2zebra dataset from UC Berkeley's repository, we have successfully generated images for horse-to-zebra and zebra-to-horse tasks. We have tried U-net and ResNet blocks as the generators and Nlayer and Pixel blocks as the discriminators. We have used visual inspection and the FID metric to evaluate the performance of our implementation. The experimental results show that the ResNet-9 generator and the N-layer discriminator produces superior quality images.

1. Introduction

The goal of the image-to-image translation is to learn the mapping functions between the input and output images by training the model on a dataset containing aligned image pairs. However, there is limited availability datasets of paired images so instead, we try to learn a mapping function from one image domain to another with unpaired image data. The most important concept being used here is the Generative Adversarial Networks (GANs), which consists of two models. The generative model generates candidates for testing while the discriminator model evaluates them. The generative network learns to map from a latent space to a data distribution of interest. In contrast, the discriminator network distinguishes candidates produced by the generator from the actual data distribution. The generative network's training objective is to make the generated image indistinguishable from the real images so as to fool the discriminator network, which gives us an idea of how different the generated image is from the actual given data.

2. Related Work

The idea of image-to-image translation goes back at least to Hertzmann et al.'s Image Analogies [4] who employ a non-parametric texture model on a single input-output training image pair. Recently, however, Generative Adversarial Networks (GANs) have been widely studied since [2] in an unsupervised setting and have achieved remarkable results for style transfer [9] and image generation [1]. The adversarial loss from GANs is utilized in the CycleGAN model described by Jun-Yan Zhu, et al. in [15] to obtain mapping such that the translated images are indistinguishable from the images in the target domain. It is to be noted that Jun-Yan Zhu's approach build's on the "pix2pix" framework of Isola et al.[7], which uses a conditional generative adversarial network to learn a mapping from input to output images.

3. Method

3.1. Dataset and Data Preprocessing

3.1.1 Dataset

We used the horse2zebra dataset from UC Berkeley's repository of unpaired images [11]. It consists of 1187 horse images divided into a train - test split of 1067 and 120 respectively, and it consists of 1474 zebra images, divided into a train - test split of 1334 and 140, respectively. Each image is square and composed of RGB channels.

3.1.2 Data Pre-processing

Data pre-processing involves cropping the images to 256 × 256 due to computational constraints.

3.2. Formulation

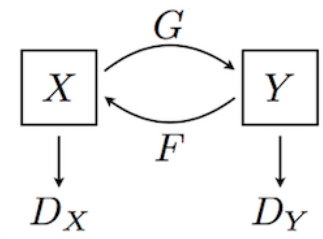

Figure 1. Basic Structure of the CycleGAN from [15]

Our reimplementation of CycleGAN which is shown in Fig. 1 consists of:

1. Two mappings $G : X \rightarrow Y$ and $F : Y \rightarrow X$
2. Corresponding Adversarial Discriminators D_x and D_y

Role of G: G is trying to translate X into outputs, which are fed through D_y to check whether they are real or fake according to Domain Y.
Role of F**:** F is trying to translate Y into outputs, which are fed through D_x to check if they are indistinguishable from Domain X.

Our objective function consists of three types of losses: the Adversarial Loss, which ensures that the learned mapping G and F produce outputs that match the distribution of Y and X respectively, the Cycle Consistency Loss to constrain the mappings further to ensure that G and F do not contradict one another, and the Identity Loss.

3.2.1 Losses

1. **Adversarial Loss:** The adversarial losses are applied to both G and F. G produces translated images G(x) that match the distribution of Y, while D_y attempts to distinguish between the translated images and the real images y, similarly, for the mapping F.

   We use the least squares loss, found by Mao et al. [10] to be more effective than the typical log likelihood loss.

$$Loss_{adv}(G, D_y, X) = \frac{1}{m}\sum_{i=1}^{m}(1 - D_y(G(x_i)))^2 \quad (1)$$

$$Loss_{adv}(F, D_x, Y) = \frac{1}{m}\sum_{i=1}^{m}(1 - D_x(F(y_i)))^2 \quad (2)$$

2. **Cycle Consistency Loss:** Adversarial losses do not ensure that the produced outputs are visually identical to images in the respective domain. Therefore, when we map an image x from the X domain using G, an image $G(x)$ is produced which only matches the distribution of Y and hence can be any random permutation of images in the Y domain, which might not be identical to the input image, x. As a result, the cycle consistency loss is introduced to augment the adversarial loss, which shrinks the space of possible mapping results.

   This means that for each image x from the domain X, an image translation to the domain Y and back to the domain X should bring x back to the original image. that is $x \rightarrow G(x) \rightarrow y \rightarrow F(y) \approx$ x. This is equivalent to x$\rightarrow$G(x)$\rightarrow$F(G(x)) $\approx$ x. They term this as forward cycle consistency which is shown in Fig. 2. Similarly, for each image y from domain Y, G and F should satisfy the mapping $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$ (backward cycle consistency Fig. 3).

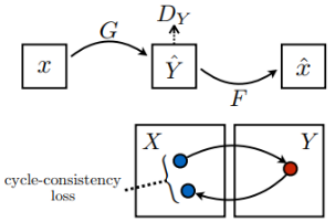

Figure 2. Forward Cycle-Consistency Loss from [15]

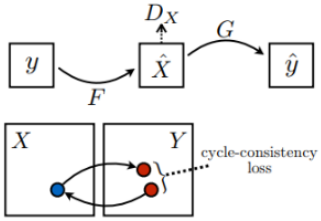

Figure 3. Backward Cycle-Consistency Loss from [15]

$$Loss_{cycle}(G, F, X, Y) = \frac{1}{m}\sum_{i=1}^{m}[F(G(x_i)) - x_i] + [G(F(y_i)) - y_i] \quad (3)$$

3. **Identity Loss:** The technique of identity loss by Taigman et al. [14] further ensures that outputs from a mapping visually match the images from the domain to which they map. It should be considered that the identity mapping loss helps preserve the color of the input images. Without $Loss_{identity}$, the generator G and F are free to change the tint of input images when it is not necessary.

$$Loss_{identity} = |G(Y) - Y| + |F(X) - X| \quad (4)$$

3.2.2 Objective Function:

We can create the full objective function by putting the loss terms together, and weighting the cycle consistency loss by a hyperparameter λ. As per [15], λ = 10.

$$Loss_{full} = Loss_{adv} + Loss_{identity} + \lambda Loss_{cycle} \quad (5)$$

3.3. Network Architecture

In Fig. 4, we see that the model works by taking an input image from domain D_X, which is fed to our first generator Generator $X \rightarrow Y$, whose job is to transform a given image from domain D_X to an image in target domain D_Y. This new generated image is then fed to another generator Generator $Y \rightarrow X$, which converts it back into an image, $Cyclic_X$ from our original domain D_X.

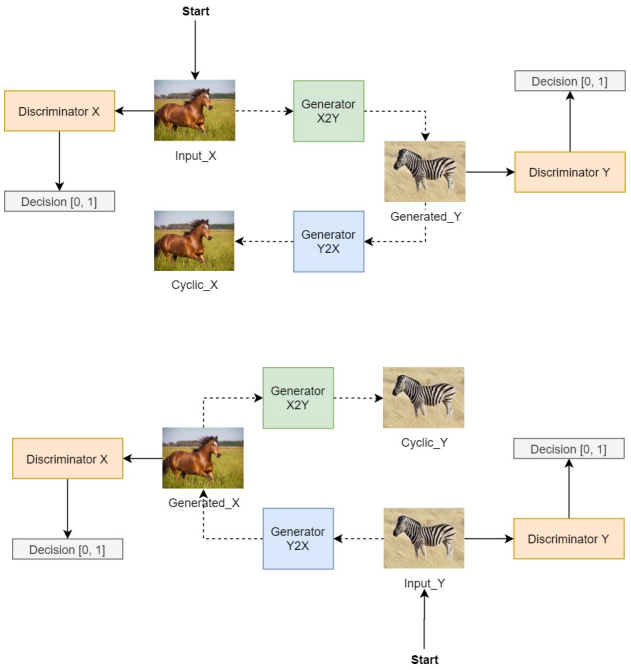

Figure 4. Simplified View of CycleGAN Architecture

We have developed our training framework in PyTorch and both variations of the Generator process an image of size 3x256x256 and generate an image of the same size. We have explored the following different network architectures for generators and discriminators.

3.3.1 Generator:

U-Net: U-net is a convolutional neural network which is proposed in [13]. It consists of two paths. One is an encoder path, and the other is a decoder path. The encoder path captures the context of the image, producing feature maps. Encoder path is just a stack of convolution and max-pooling layers. Decoder path used to enable precise localization using transposed convolutions. U-net only contains Convolutional layers and does not contain any Dense layer because of which it can accept the image of any size. Moreover, it concatenates the ith layer's feature map to the $(n-i)$th layer, where $1 < i < 8$, in order to avoid the loss of border pixels.

The encoding path is composed of 4 blocks, where each block consists of two $4*4$ convolution layers, ReLU activation, and max pooling. The expansion path consists of a deconvolution layer with stride 2, concatenation with the corresponding cropped feature map from the contracting path, and ReLU activation. The U-Net combines the location information from the downsampling path with the contextual information in the upsampling path to finally obtain a general information combining localization and context, which is necessary to predict a good result. The structure of the U-net generator is shown in Fig 5.

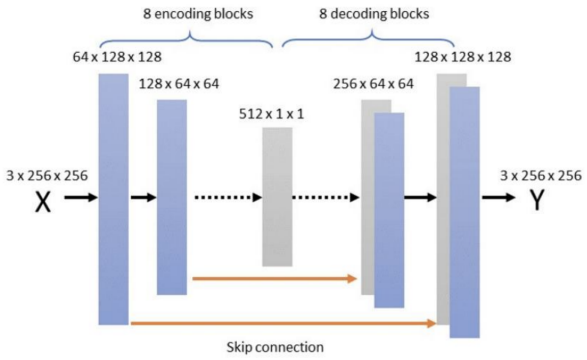

Figure 5. U-net Generator Architecture.

ResNet: The Resnet-based generator consists of several Resnet blocks (6 or 9 Residual Blocks) between a few downsampling and upsampling operations. All non-residual convolutional layers are followed by instance normalization and ReLU non-linearities except for the output layer, which instead uses a scaled Tanh to ensure that the output has pixels in the range [0, 255]. The first and last layers use 7×7 kernels; all other convolutional layers use 3×3 kernels.

The instance normalization layer ensures that the input to each layer is normalized to have zero mean and unit variance, which helps to stabilize the learning as evidenced by [6] and deal with training problems that arise due to poor initialization and helps gradient flow in deeper models [12]. Instance normalization helps to prevent the generation from collapsing of all samples to a single point. However, instance normalization is not applied to the discriminator input layer and generator output layer as it causes sample oscillation and model instability [12].

The networks use two stride-2 convolutions to downsample the input followed by several residual blocks (6 or 9) and then two convolution layers with stride 2 to upsample.

In [3], He et. al have implemented residual connections in deep networks for image recognition as residual connections make the identity function very easy to learn. This is an attractive quality for image transformation networks

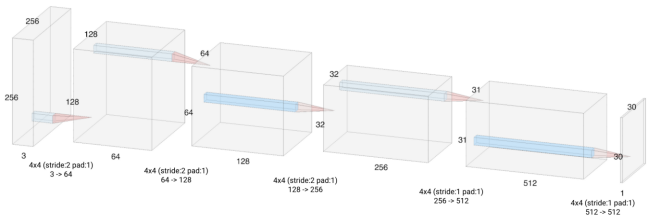

Figure 6. PatchGAN 70 x 70 Architecture

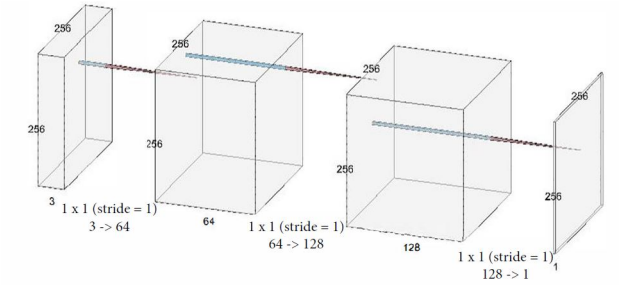

Figure 7. Architecture of Pixel Discriminator

[8], as the translated image must share features and structure with the original input image. The body of the network we have implemented in PyTorch consists of 6 or 9 residual connection blocks, thereby implementing both the Resnet-6 and the Resnet-9 Generators.

3.3.2 Discriminator:

Nlayer Discriminator: The Nlayer discriminator is nothing but a convolution network or a 70 x 70 PatchGAN used in [7]. It splits the raw input image into some small local patches, runs a general discriminator convolutionally on every patch, and average all the responses to obtain the final output indicating whether the input image is fake or not. PatchGAN classifies individual (N x N) patches in the image as "real vs. fake", opposed to classifying the entire image as "real vs. fake", which enforces more constraints that encourage sharp high-frequency detail. Additionally, the PatchGAN has fewer parameters and runs faster than classifying the entire image. Fig.

According to [7], using this discriminator is sufficient because the problem of blurry images caused by failures at high frequencies like edges and details can be alleviated by restricting the GAN discriminator to only model high frequencies, and the Nlayer discriminator is designed to stress this.

The architecture of the 70x70 PatchGAN discriminator is shown in Fig. 6. The 70 x 70 PatchGAN has a fixed number of 3 layers (excluding the output and second last layers). It uses a fixed stride of 2 x 2 (except in the output and second last layers) and a fixed kernel size of 4 x 4.

Pixel Discriminator: The Pixel Discriminator is similar to the Nlayer discriminator mentioned above, with the only difference being that a 1 x 1 receptive field is used.[7]

Instead of classifying whether entire patches of a given size are fake or not, the Pixel Discriminator checks if the pixel is fake. As mentioned in [7], it enables greater diversity in color but does not have any effect on spatial statistics. The architecture of Pixel Discriminator is shown in Fig. 7

3.3.3 Training:

For each iteration, during training we update one step of gradient descent on D and G using Adversarial, Cycle Consistency, and Identity losses as explained in the prior section. Table 1 shows the different losses obtained for the test image under consideration for N-layer Discriminator, while Table 2 shows the losses obtained for the test image Pixel Discriminator. The discriminator trains faster than the generator because classification is a relatively simpler task than a generation. We use two separate Adam Solvers for generator and discriminator with β_1=0.5, β_2=0.999, learning rate α=0.0002,λ=10. We train the model for 200 epochs until loss plateaus.

4. Experiment Results:

The generated images from the different generators using N-layer Discriminator are shown in Figures 8,9,10,11.

Figure 8. Images generated with Generator U-net128 and Discriminator-N layer

Figure 9. Images generated with Generator U-net256 and Discriminator-N layer

Figure 10. Images generated with Generator Resnet6 and Discriminator-N layer

Figure 11. Images generated with Generator Resnet9 and Discriminator-N layer

The discriminator and generator losses for 1 test image using each generator an the N-layer discriminator is show in table 1.

The generated images from the different generators using Pixel Discriminator are shown in Figures 12,13,14,15.

Table 1. Obtained losses for 1 test image: N-layer Discriminator

Architecture	Adversarial Loss (MSE)	Cycle Consistency Loss (L1)	Total Generator Loss
Generator:U-net128 Discriminator: N layer	D1:1.6013 D2:3.1557 Total: 4.7571	FCL:0.5480 BCL:0.4784	7.2245
Generator:U-net256 Discriminator: N layer	D1:1.6966 D2:1.6846 Total: 3.3812	FCL:0.4250 BCL:0.4682	4.6146
Generator:Resnet6 Discriminator: N layer	D1:1.7254 D2:1.9546 Total: 3.680	FCL:0.8247 BCL:0.8250	6.2284
Generator:Resnet9 Discriminator: N layer	D1:1.5627 D2:1.5931 Total: 3.1557	FCL:1.2255 BCL:0.8823	6.2406

Figure 12. Images generated with Generator U-net128 and Discriminator-Pixel

Figure 13. Images generated with Generator U-net256 and Discriminator-Pixel

Figure 14. Images generated with Generator Resnet6 and Discriminator-Pixel

Figure 15. Images generated with Generator Resnet9 and Discriminator-Pixel

The discriminator and generator losses for 1 test image using each generator an the Pixel discriminator is show in table 2.

Table 2. Obtained losses for 1 test image: Pixel Discriminator

Architecture	Adversarial Loss (MSE)	Cycle Consistency Loss (L1)	Total Generator Loss
Generator:U-net128 Discriminator: Pixel	D1:0.5287 D2:0.5156 Total: 1.0443	FCL:0.1522 BCL:0.1270	2.3455
Generator:U-net256 Discriminator: Pixel	D1:0.4935 D2:0.4611 Total: 0.9547	FCL:0.1778 BCL:0.1544	2.3455
Generator:Resnet6 Discriminator: Pixel	D1:0.4765 D2:0.4707 Total: 0.9472	FCL:0.8839 BCL:1.0972	4.7710
Generator:Resnet9 Discriminator: Pixel	D1:0.4584 D2:0.5553 Total: 1.0137	FCL:0.6514 BCL:0.6503	3.9002

4.1. Model Evaluation:

We use two evaluation metrics to comment on the quality of images generated by the CycleGAN. We visually inspect the generated images to evaluate the synthetic image quality; however visual inspection can fail in cases where the differences are small. To counter this, we have also computed the FID score.

Visual Inspection: Visually, we see that the UNet-256 Generator with the N-layer Discriminator outperforms the UNet-128 Generator, as it is a deeper network. The definition of the stripes on the generated images of the zebra, and the color diversity in the generated horse images are higher for UNet-256. ResNet-9 outperforms ResNet-6 as well as both the UNet Generators, which is abundantly clear via visual inspection with ResNet-9 giving clearly defined stripes in the horse to zebra image translation. This is further confirmed by our obtained losses in Table 1; the total discriminator loss for UNet-256 is less than that of UNet-128, and the total discriminator loss for ResNet9 is less than that of ResNet6.

We also used PixelGAN as our discriminator with each of the 4 generators, however we did not obtain translated images of very good quality as shown in Figures 12, 13, 14 and 15. The 1x1 PixelGAN is the shallowest model of PatchGAN and therefore compromises heavily on image quality although it is faster to train. Visual inspection is a poor metric of evaluation in this case, however we can extrapolate from our discriminator loss values in Table 2 that UNet-256 performs better than UNet-128.

Frechet Inception Score (FID): The FID score summarizes the distance between the Inception feature vectors for real and generated images in the same domain,and lower scores have been shown to correlate well with higher quality images [5].

We computed our FID scores over 50 generated images of the horse to zebra mapping, and summarized its similarity with the corresponding 50 images from the Zebra Test Data within the Dataset to evaluate the quality of the generated zebra images. Similarly, we computed the FID scores between 50 generated horse images from zebra to horse mapping, and the Horse Test Data.

Our visual inspection findings are corroborated by the FID scores, with the ResNet9 Generator with the N-layer Discriminator giving us the lowest average FID score i.e. the best performance.

Table 3. FID Scores for N-layer Discriminator

Model (N-layer Discriminator)	Horse to Zebra	Zebra to Horse	Average
UNet-128	227.31	238.03	232.67
UNet-256	274.31	245.02	259.665
ResNet-6	240.2	193.76	216.98
ResNet-9	190.93	211.1	201.065

Table 4. FID Scores for Pixel Discriminator

Model (Pixel Discriminator)	Horse to Zebra	Zebra to Horse	Average
UNet-128	312.5	250.93	281.715
UNet-256	317.22	255.29	286.255
ResNet-6	340.34	263.87	302.105
ResNet-9	321.67	256.14	288.905

5. Conclusion

We have implemented unpaired image to image translation using CycleGANs for a combination of 4 Generators: UNet-128, UNet-256, ResNet-6 ResNet-9, and 2 Discriminators: N-layer Pixel. In general, we see that the N-layer Discriminator gives us better results than the Pixel Discriminator, and the ResNet Generator performs better than the UNet Generator. Therefore, on visually inspecting the translated images and computing their FID scores, we determine that the ResNet-9 Generator with the N-layer Discriminator gives the best performance with the lowest average FID score of 201.065. This network performs very well on the horse2zebra dataset which primarily involves color and texture translations.

References

[1] Emily Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Deep generative image models using a laplacian pyramid of adversarial networks, 2015. 1

[2] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. 1

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 3

[4] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David Salesin. Image analogies, 2001. 1

[5] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2017. 6

[6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. 3

[7] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks, 2016. 1, 4

[8] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution, 2016. 4

[9] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks, 2016. 1

[10] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks, 2016. 2

[11] Taesung Park. horse2zebra, 2017. 1

[12] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015. 3

[13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. pages 234–241, 2015. 3

[14] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*, 2016. 2

[15] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2017. 1, 2