Image Domain Transformation using Generative Adversarial Networks

Abhyudai Nouni * and Ananya Suraj †

University of Massachusetts Amherst

Abstract

We present our work on Image domain transformation which is the task of transforming an image from a domain X to an image from domain Y while retaining certain characteristics of domain X. In this paper, we investigate Cyclic Generative Adversarial Networks to perform image transformation across domains without any paired examples. Cycle GANs employ a cycle consistency loss in addition to the adversarial loss in order to enforce the highly constrained mapping to be cyclic in nature i.e. image from X transformed to image from domain Y which can then be transformed back to the image from domain X. We explore the applications of image domain transformation in two new avenues - transforming images of flowers to patterns and transforming images of people who are fat to fitter versions of themselves. We also test the boundaries of the Cycle GAN model to verify the required characteristics of both domain X and Y to successfully be able to transform from one to the other. Qualitative results are presented for all of our tasks where paired images were unavailable. Quantitative evaluation of the model was performed using fine-tuning a pre-trained model and the results of the analysis are discussed.

1. Introduction

The task of transforming images from one domain to a different one while still maintaining certain characteristics of the original domain is an interesting problem that can have a wide range of applications. Something that is almost natural for a human being is a very hard problem for a computer. For instance, it is easy for a fashion designer to draw inspiration from real life objects and replicate similar characteristics in an article of clothing he or she may design. The same process for a computer is not only hard but could result in unsightly designs. However, recent developments in the fields of Computer Vision and Artificial Intelligence have now made this seemingly hard task much more tangible.

In this paper, we present a method that allows computers to try and emulate this human trait by applying it for different domain pairs. More specifically, transforming an image that belongs to one domain to a different one in the absence of any paired training example, which in essence means that given an input image, we try to generate a completely new image with certain characteristics of the input, but in a different domain. We explore Cyclic Generative Adversarial Networks [8] (Cycle-GANs) to perform cross-domain image transformation. As mentioned earlier, image domain transformation is a powerful concept that can be extended to solve problems in different industries. We have employed the Cycle GAN model to learn mappings from images of flowers to patterns and vice versa as a possible solution for automated design generation. Another novel application we explored was the transformation of the human body from fat to fit. We have analyzed the results of applying the model on these domains both qualitatively and quantitatively.

An additional goal of this work is to answer questions such as: Can such a model work across any combination of domains? What are its limitations? In order to answer these questions, we chose two domains that have certain similar characteristics but enough dissimilarity in the structure of the object in the image to warrant doubt as to whether the Cycle GAN model would produce acceptable results. As suspected, we found that the Cycle GAN model does not perform as well as it does for structurally similar images. We have performed further analysis on the results of this model as well as discussed the implications of its limitations.

2. Related Work

Over the last few years, GANs have been studied and experimented with vigorously. Models to generate images using GANs i.e Conditional Adversarial Networks (cGANs) [1] use the MNIST dataset to generate digit images of the specified class. Another approach by Reed et al [5] uses encoded text description of images as the condition to generate

*anouni@umass.edu

†asuraj@umass.edu

images that match the given caption. There has also been prior work on using multiple GANs to achieve results. In 2016, Liu et al [4] proposed a model called Coupled GAN where weight sharing was employed to learn the joint distribution of images in different domains.

The work done by Isola et al. [1] , uses a conditional generative adversarial network to learn a mapping from input to output images. However, this model works only when paired data is available. Taeksoo Kim et al[3] developed a model called Discover GAN that transforms images across domains by forcing all images from one domain to be representable in another domain. Prior work in the domain of image translation in the absence of paired data has been done to transform images of horses to zebras, style transfer from one artist to another, among other things by Jun-Yan Zhu et al[8]. Our work in this paper is inspired by the Cycle GAN model.

We propose to examine and experiment with the CycleGAN model and apply it on completely different domains that we perceive as yet unexplored. In the following sections, we have described the problem we are trying to solve in detail, while providing sufficient motivation about the domains to which this solution can be applied. The expected results of our approach and the technical details of the model have been specified as well. Qualitative and Quantitative Evaluation of the models are discussed and analyzed.

3. Approach

As described in the previous sections, we seek to build a model that can learn to transform images from one domain to another without the need for a paired input-output dataset. Rather than using images that are paired, we instead provide the model with two collections of images from both the domains X and Y that we are interested in.

3.1. Generative Adversarial Networks

A mapping G: X $\rightarrow$ Y is trained such that G transforms an image from domain X to an image from domain Y such that the new images generated by G in domain Y are indistinguishable from the original images provided in the collection of images for domain Y. This is done by training a discriminator that is trained to classify original domain Y images from those generated by the model. In this case, we do this but with an additional condition that the mapping G should be 'cycle consistent'. This means that a mapping G: X $\rightarrow$ Y and a mapping F: Y $\rightarrow$ X are inverses of each other. This is done by simultaneously training G and F and applying a cycle consistency loss that encourages $F(G(x)) = x$ and $G(F(y)) = y$.

3.2. Loss function

The objective function for our GAN comprises of multiple factors. There is an adversarial loss which is being maximized by the discriminator and minimized by the generator.

Consider the domains X and Y. y and x are examples sampled from the data. $G : X \rightarrow Y$ $F : Y \rightarrow X$

Adversarial loss for X $\rightarrow$ Y:
$L^{X \rightarrow Y}_{adversarial} = log(D_Y(y)) + log(1 - D_Y(G(x)))$

Adversarial loss for Y $\rightarrow$ X:
$L^{Y \rightarrow X}_{adversarial} = log(D_X(x)) + log(1 - D_X(G(y)))$

Total adversarial loss = $L^{X \rightarrow Y}_{adversarial} + L^{Y \rightarrow X}_{adversarial}$

Another factor to consider is a cycle consistency loss which tries to complete the cycle of Image translation, which says that an image of domain A when transformed to another domain B by a function F and transformed again by another function G which converts it back to domain A, should be as similar as possible. This tries to make sure that F and G are inverse of each other. This can also be referred to as a reconstruction loss. By combining the cycle consistency loss with the adversarial loss on domain X and Y, we are able to perform unpaired image domain transformation successfully.

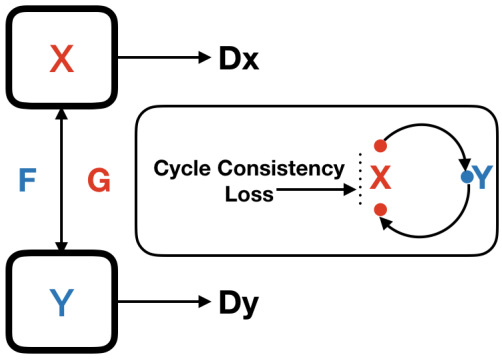


Figure 1. Cycle Consistency loss

Cyclic/Reconstruction loss for X $\rightarrow$ Y:
$L^{X \rightarrow Y}_{cyclic} = \|x - F(G(x))\|_1$

Cyclic/Reconstruction loss for Y $\rightarrow$ X:
$L^{Y \rightarrow X}_{cyclic} = \|y - G(F(y))\|_1$

Note that the 1 subscript refers to L-1 norm.

Combining everything together, we get the total generator loss as sum of adversarial and cyclic loss with a weight factor λ for both directions X $\rightarrow$ Y and Y $\rightarrow$ X.

Total Loss for G:Y $\rightarrow$ X:
$loss_G = L^{X \rightarrow Y}_{adversarial} + \lambda_G * L^{X \rightarrow Y}_{cyclic}$

Total Loss for F:Y $\rightarrow$ X:
$loss_F = L^{Y \rightarrow X}_{adversarial} + \lambda_F * L^{Y \rightarrow X}_{cyclic}$

Final objective function:
$= loss_G + loss_F$
$= L^{Y \rightarrow X}_{adversarial} + L^{X \rightarrow Y}_{adversarial} + \lambda_G * L^{X \rightarrow Y}_{cyclic} + \lambda_F * L^{Y \rightarrow X}_{cyclic}$

Notice that the λ term along with the cyclic loss is akin to a weight for regularization parameter in regular loss functions combining data loss and generalization loss. The idea of cycle-consistency loss is inspired from the CycleGAN[8] approach, however in our experiments we experimented with multiple variants of this objective function like giving different weights to given to adversarial and cycle-consistency loss. The actual λ_G and λ_F values were also manipulated.

3.3. Architecture

3.3.1 Generator

Currently, the generator architecture adapts from Johnson et al[2], which uses a combination of per-pixel loss and perceptual loss for style transfer and super-resolution. It contains convolutions with stride = 2, as well as some fractional convolutions with stride = 1/2. We use images resized and cropped to 256x256 with 9 blocks. We also use InstanceNorm layer similar to Johnson et al[2], which was first introduced by Ulyanov et al[7].

3.3.2 Discriminator

Our discriminator network uses Patch-GANs with patches of the size 70 x70, however we experimented with different patch sizes for different applications. More experimental details are given in Section 4.2. Using a patch-level discriminator greatly reduces the number of parameters and a smaller patch size will benefit even more from this, much similar to the advantages of convolutional neural networks over fully connected ones. The initial layers use Instance Normalization and LeakyReLU with slope 0.2 and generate a high dimensional feature vector. The last layer uses a simple convolutional layer to flatten the vector to produce a boolean 1x1 output.

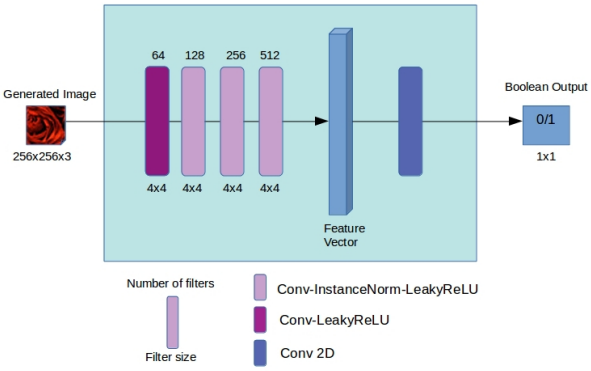


Figure 3. Discriminator architecture

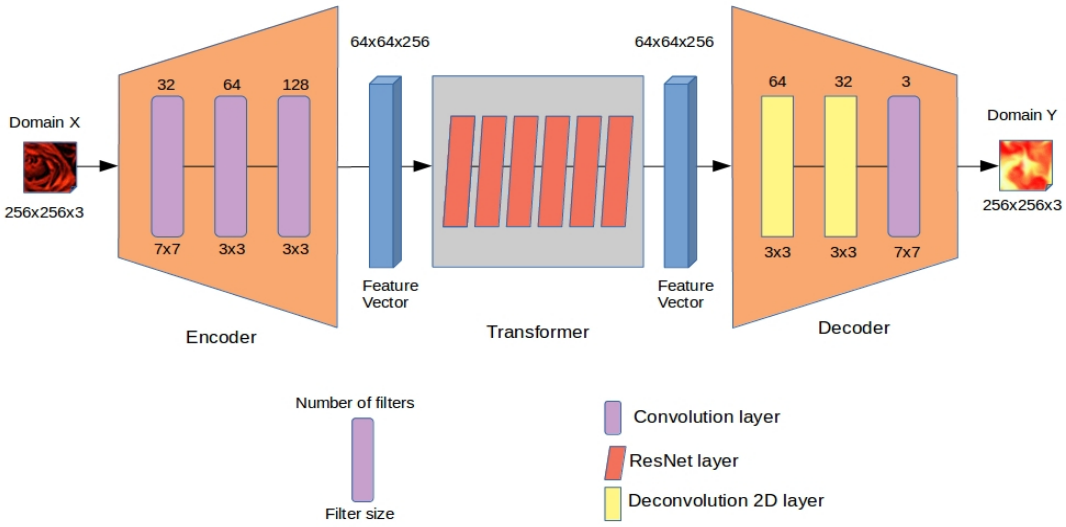


Figure 2. Generator architecture

4. Experimental Setup

4.1. Datasets

Since the idea of image domain transformation is very broad, many different possible applications can be thought of which fit into this category. Hence, we have chosen multiple domain-pairs which each can be thought of as a pair of datasets for a pair of domains corresponding to one single application. Each of the datasets have their own challenges that we need to overcome and require modifications to the overall architecture of the model as well. These different domain pairs are explained in detail in the following subsections. Note that A$\Longleftrightarrow$B refers to one domain pair where A and B are different domains and the task is to be able to learn a model that converts images from A to B and B to A.

4.1.1 Flowers $\Longleftrightarrow$ Floral Design Patterns

The application of Cycle-GANs[8] and Disco-GANs[3] in the fashion industry has been explored before this work. Many researchers have found that designing shoes and handbags from sketches or shoes from handbags and vice-versa, can be a lucrative and innovative method to make products that match and go well together in an outfit. However, not much has been done to explore generating the designs and patterns on clothing themselves. While it is a hard problem to design new clothes entirely, we can attempt to design new fabrics by drawing inspiration from real images of flowers found in nature. This is very useful as floral designs can be used for different types of clothing and fabric. As the data we needed is not a publicly available dataset, we scraped images to make our own dataset. The Flower-ToPattern dataset is comprised of around 1200 images of flowers collected from ImageNet and around 640 images of patterns scraped from Google Images using the search term "floral pattern". We want to be able to train a model which can convert images of real flowers to floral design patterns or vice-versa. The first direction of transformation is useful for designing floral patterns specific to certain flower types, whereas the second direction of transformation can be used to visualize the source of inspiration for a particular floral pattern.

4.1.2 Fat $\Longleftrightarrow$ Fit Body transformation

Visualization is a time-tested, proven technique that involves focusing on positive mental images in order to achieve a particular goal. Visualization is so powerful because it impacts our subconscious minds and without us even realizing it, we are propelled in the direction of our thoughts. Today, fitness has become an integral aspect for every health conscious individual. It can often be hard to find the motivation to work out in order to transform one's body into the shape he or she may desire. What if we had the ability to see how our bodies would look if we lost a few pounds? It could be a very motivating factor to get healthy and fit. Transforming images of human bodies to fitter versions of themselves is an interesting problem that our model can be applied for.

In order to do this, we collected data from Google images by using the search terms - "before and after male fat loss transformation", "male body transformation fat to fit", "female body transformation fat to fit" and "crossfit 1 year body transformation". We found about 600 images of body transformation of different people, which were carefully pruned to about 300 images, considering images of people only front facing poses, without clothes, and without any other people or objects obstructing the way. However, the number of images were not enough so data augmentation was required. Details are discussed in Section 4.2.4.

4.1.3 Testing Boundaries: Sharks $\Longleftrightarrow$ Planes

In addition to the above specified datasets, out of curiosity and in order to stimulate our creativity we have also experimented with Shark and Plane images that are available on ImageNet. However, we believe that this pair of domains will challenge the limits of our network and is to be treated as a highly experimental dataset.

It could be interesting to see if shark images can generate plane images and vice versa. We chose these domains specifically because images of both planes and shark are generally found with a blue background which correspond to the sky and the sea respectively. Yet, structurally, both sharks and planes are pretty different. Not only are their exact shapes and sizes different, but the parts of their bodies i.e. fins and wings for example are different too. The CycleGAN model has been applied to various domains like style transfer, photo colorization, horses to zebras etc. We wished to test the boundaries of this model and verify it's effectiveness when the two domains in question do not have common geometric and structural features.

4.2. Experiments

For our image domain transformation model, we had different experiments with the general CycleGAN model related to changing different generator and discriminator models as well as the loss functions. Once we had found a set of parameter values which provided good results on our test datasets, we fixed those parameters for the model. At training time, we still carried out hyperparameter tuning. In addition to these experiments, we planned our sets of experiments based on task difficulty as well as dataset size. The individual experiments for each dataset are explained in detail below.

4.2.1 Model Parameters

Among the changes that we tried were, some were . Some of the parameter changes were subtle like changing the patch size for discriminator from 70x70 to 50x50, while others were more obvious like changing the number of convolutional layers within the generator. We also experimented with different loss functions, however our final loss function resembles closely to the CycleGAN objective function with a L1 loss instead of a least-squares loss. The final architecture has already been described and shown in previous sections.

4.2.2 Hyperparameter Tuning

During training time, the hyperparameters which needed to be tuned were learning rate, learning rate decay, epoch when to start decaying learning rate and different weights to cyclic loss (Λ). We used a grid search for hyperparameter tuning which went from coarse to fine.

4.2.3 Dataset Subsampling

With the entire datasets even though the number of images is not that high, GANs require high training time. So we subsampled our datasets and ran them for smaller number of epochs. The subsampling ratio was around 1:4. Typically we used around 20-40 epochs for initial experiments and 100-200 epochs for final experiments. This additionally reduced our training time by a factor of around 5. These experiments helped us tune the parameters and hyperparameters of the network. Finally, we trained the networks on entire datasets with the best model and hyperparameters, which took longer but resulted in better quality of generated images.

4.2.4 Data Augmentation

For the Fat to Fit dataset, we ended up having the fewest of images per domain. For this reason, our initial results were also not looking impressive. We decided to use different methods of augmenting our data. These involved adding noise, horizontal and vertical translations, horizontal flipping, and random crops. These techniques not only increased our training data, but they actually resulted in actual improvement in model performance as can be seen from our final results.

4.2.5 Cloud Compute details

The generative adversarial model is complex and involves a large number of parameters. Training it on a regular CPU even on a small dataset is highly cumbersome and time-consuming. We used free credits provided with creating an account on Google Cloud Compute to create 5 VM instances. The general configuration for our VM instance was 8 high-cpu cores, 52 GB RAM, 1 Nvidia K80 Tesla GPU, 30 GB Solid State Drive. This configuration costed around $1 per hour. Training the network took between a few minutes to 30 minutes per epoch. We also had one instance with an additional GPU and used a greater batch size to make use of multiple GPUs for that instance. We ended up using free credits equivalent to $300 USD for total compute time of around 300 hours. Having multiple such instances for 2 of our accounts helped us train our network with different hyperparameters simultaneously.

4.3. Evaluation

This section describes the various evaluation techniques we used to evaluate and analyze our model for all of our datasets. We have performed both Qualitative and Quantitative Evaluation on the models.

4.3.1 Qualitative Evaluation

The first metric of performance testing was visual evaluation to see if a user can tell the difference between computer generated images and real images. This is clearly subjective and not a standard way of evaluating our model but one that will be able to ensure that the results generated are realistic enough. It is not uncommon to use manual evaluation for generative tasks. Often researchers resort to third party reviewers like Amazon Mechanical Turkers who then subjectively evaluate the generated images. For our model, we did the qualitative evaluation ourselves and invite the readers to evaluate our results subjectively as well as objectively.

4.3.2 Quantitative Evaluation

For quantitatively evaluating the images generated by our model, we wanted a quantitative evaluation of how much the images are transformed from one domain to another. One perfect method to evaluate this empirically would be to train a good classifier which can distinguish between the two domains and use that classifier to classify the generated images and verify that the domain of generated image is different from the input image. For example for an input image of a flower, we want the generated image to be classified as a pattern. Note that this classifier is a binary type of classifier in the sense that it has only two output class labels, one for each of the two domains in a particular dataset. We will now explain in detail, how we trained a classifier to evaluate our results in the next subsection.

4.3.3 Fine-tuning a Pretrained Model for Generated Image Classification

The idea of using a pretrained model and fine-tuning its final layers for classifying a custom dataset is not new. However this concept, also referred to as transfer learning is a novel technique for learning a classifier quickly using a pretrained model which has already been trained on larger and broader datasets like the ImageNet and fine-tuning and retraining its final layers on our own dataset for image classification. We took Google's Inception V3 model[6] and retrained its final layer on our training dataset to learn a classifier which can discriminate between our two domains, e.g. Flowers and Patterns. Since we only trained the final layer, it was possible to do so on our personal laptops without dedicated GPU support in less than 30 minutes. The classifier was trained with high training accuracy and was used to classify the generated images. Accuracy is measured as how many images in the test dataset were classified as domain B for the mapping from domain A to B where input images are from domain A. For speeding up the process of training, we used a variant of the Inception V3 model called the MobileNet. The main difference between the two is that Inception V3 is optimized for accuracy, while the MobileNets are optimized to be small and efficient, at the cost of some accuracy.

4.4. Results

The results of our experiments are presented below. Each dataset has a separate section and we present the results of our qualitative and quantitative evaluation. Qualitative results are presented only for some datasets. Generated images can be compared to the original images and we can see the visual change in the images and how the network learned to transform the features across different domains. The analysis of the results are also split across the different datasets since each dataset is a different task with different complexity.

4.4.1 Flower $\Longleftrightarrow$ Pattern

The training loss for our model for this dataset is shown in Figure 4. The ideal hyperparameters were found to be with learning rate 0.0003 and lambda was set to 10 for cycle-consistency loss.

We ran our quantitative evaluation on this dataset by retraining the model pretrained on ImageNet to classify between flowers and floral patterns on our training data. We did not include our test input images in this training. The generated images on our test images were then evaluated using the classifier. The output of the classifier is a probability for each label such that the values for both class labels sum to 1. Binary accuracy refers to the number of times the expected output domain class label had a value greater than 0.5. Mean confidence is the average of probability values for the expected label class across all test images. These quantitative results are shown in Table 1.

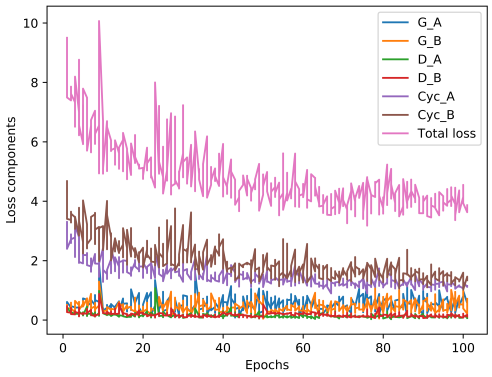


Figure 4. Flower To Pattern training loss curve

Note that in Figure 4, we can see the different components of loss and how they progress over time(epochs). The major contributors to total loss are Cyclic losses A and B but this is also dependent on the setting of λ_A and λ_B. Note that A was flower domain and B was pattern domain. Losses are higher for cycle B which also refers to the flowers generated from patterns. This ties in well with the quantitative results.

Table 1. Flower/Pattern dataset quantitative results

Metric	Generated Patterns	Generated Flowers
Binary Accuracy	0.58	0.3
Mean Confidence	0.6045	0.295

From quantitative results, it seems that the generated patterns are often classified as flowers which means that they must be good enough to fool our classifier. This does not mean that the images that are not classified correctly are not good enough. That is why we need subjective evaluation for these images, since the extent of morphosis of an image domain cannot be acutely measured using trivial methods. We can analyze the results subjectively and try to see visually how the images that score high and low on the classifier look like subjectively. These results are shown in Tables 2, 3 and 4.

It is to be noted that our test dataset size was 50 images each for both domains as we found only 600 odd flower pattern images suitable for training and wanted to keep the number of flower images at a comparable level. Since the images are unpaired, number of flower images was more than number of pattern images, whereas in paired case there has to be an equal number of images of both domains with a one-to-one mapping. Model performance is not the same in both directions and a reason for that could be that our training data had more flower images and less pattern images.

Table 2. Pattern to Flowers results for good quantitative score

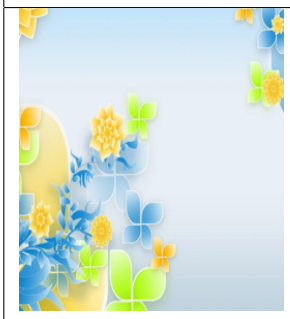
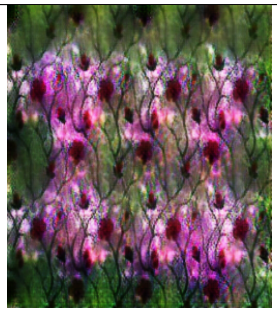
Original Pattern	Generated Flower	Reconstructed Image	Classifier confidence
			99.04%
			99.81%
			99.93%
			99.99%
			99.85%

Table 3. Pattern to Flowers results for ”bad” quantitative score

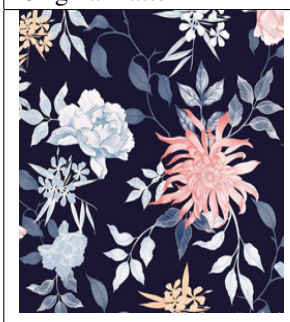
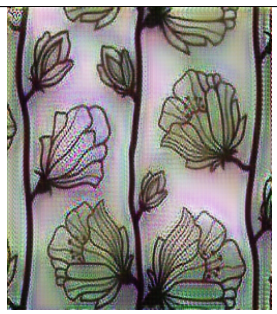
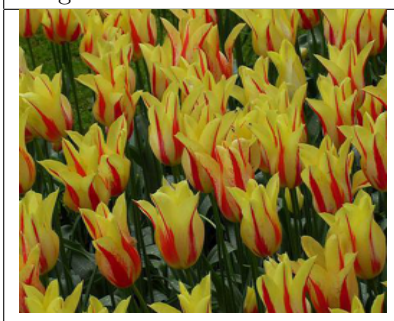
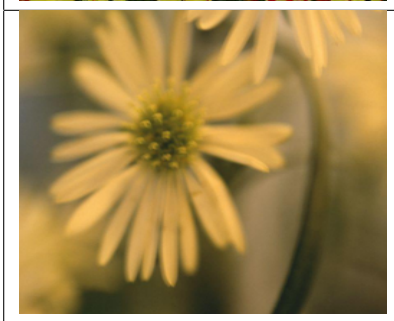
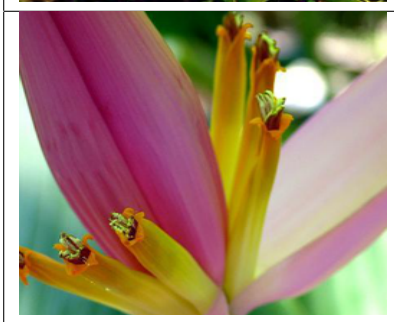
Original Pattern	Generated Flower	Reconstructed Image	Classifier confidence
			46.03%
			29.05%
			12.75%
			6.5%
			0.00%

Table 3: Flowers to Patterns results along with quantitative score

Original Flower	Generated Pattern	Classifier confidence
		99.99%
		99.86%
		20.01%
		00.00%

4.4.2 Fat $\Longleftrightarrow$ Fit

Our second dataset for which we ran the model was Fat to Fit conversion for human males. For this model, we only show the results in one direction although the model was trained for both directions as usual. The training loss curve is shown in Figure 5.

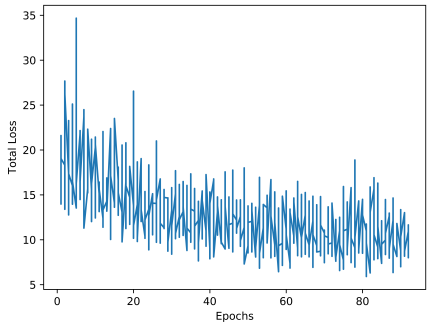


Figure 5. Fat to fit training loss curve

The results for this model are best evaluated subjectively. Since the images are of a human body, there needs to be consistency and realism in the image. Additionally the image needs to be modified enough to look like that of a fit person as compared to the original image. This model had to be finetuned the most along with a lot of data augmentation, however we were able to achieve really good performance in the end. The results are shown on the next page. We think that the model does a very good job of domain conversion for this dataset. The hyperparameters for this model were learning rate at 0.0003 and lambda was increased to 40 for Cycle A to B since that is the direction of preference.

4.4.3 Testing Boundaries: Sharks $\Longleftrightarrow$ Planes

The training loss for our model for this dataset is shown in Figure 6. It can be observed that the training loss curve is not smooth for this dataset and the reason for this is that this model was trained on a smaller dataset and for less number of epochs(25) as compared to 100 and 200 epochs for the other dataset. Some sample results are shown below in Tables 4 and Figure 5.

Looking at our output for test images, we can see that the Cycle GAN model does not work as well for these domains as compared to the previous two datasets. The images are often noisy and the only aspect of the test image that seems to have been successfully transformed is the background i.e. the sea is made to look more like the sky and vice versa. There are some differences in the pattern on the shark or plane's body in generated images but the difference is not sufficient to make the new image appear authentic.

As mentioned earlier, this dataset was chosen for purely experimental purposes. The results go on to show the limits of the Cycle GAN model as it makes an assumption that the two domains have enough structural or texture similarity for the image transformation to occur seamlessly.

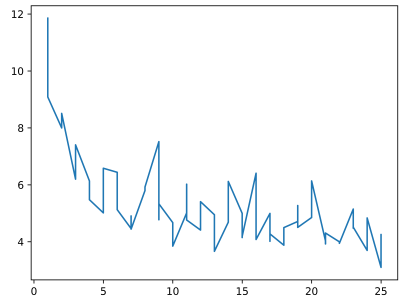


Figure 6. Shark to Plane training loss curve

Table 4. Shark to Plane Transformation

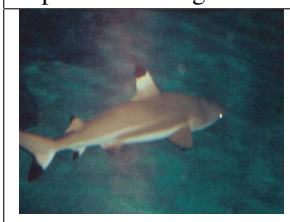
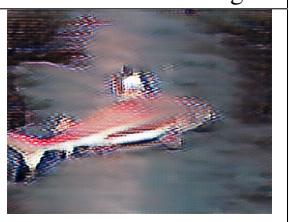
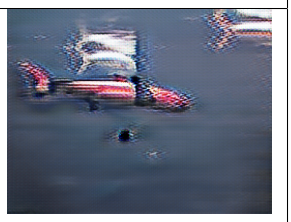
Input Shark Image	Generated Plane Image
	
	

Table 5. Plane to Shark Transformation

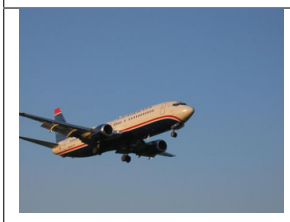
Input Plane Image	Generated Shark Image
	
	

Table 4: Fat to Fit results

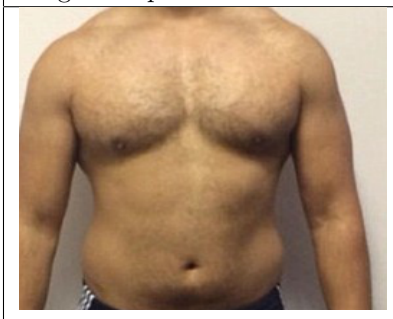
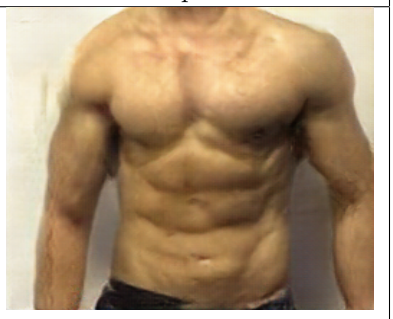
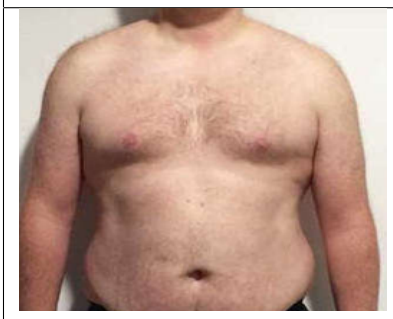
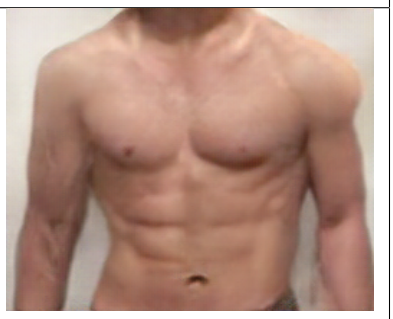
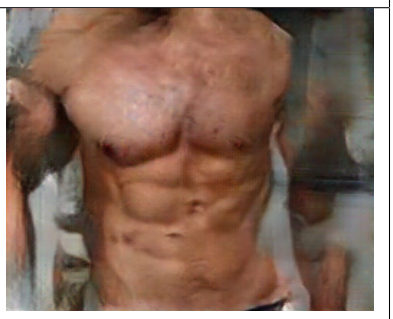
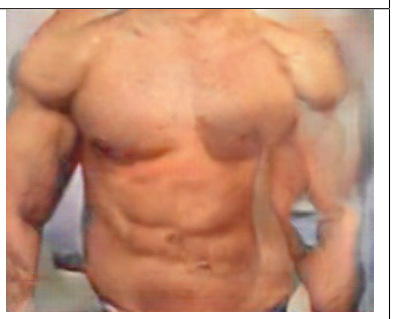
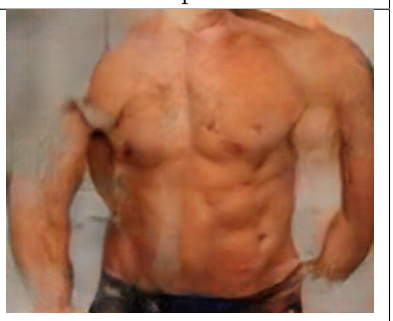
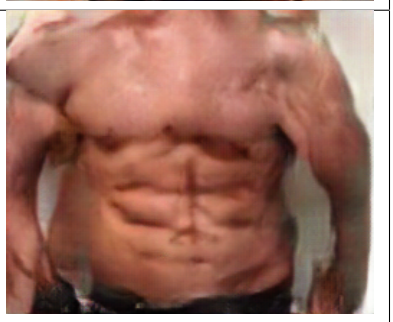
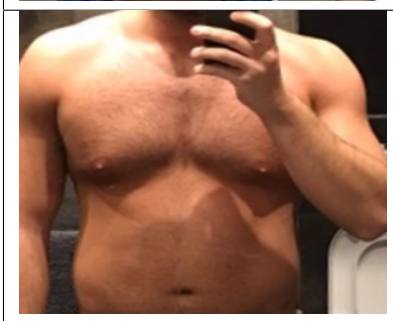
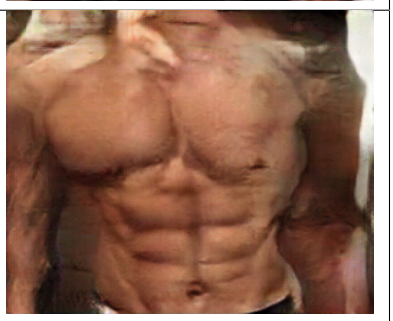
Original Input	Generated Output
	
	
	
	

Table 5: Fat to Fit results contd.

Original Input	Generated Output
	
	
	

5. Conclusion

Training GANs is not just time-consuming, it is also memory and compute intensive. For CycleGAN model, since there are 2 discriminators and 2 generators, training the model is equivalent to training 4 different neural networks at once. This can have practical implications, as resources are often limited and expensive. Training networks in such expensive conditions, required us to employ techniques like dataset sub-sampling and parallel hyperparameter tuning.

The effect of cycle-consistency loss does not necessarily increase the domain transformation property of the model. However, it does make sure that the reconstructed image after applying the domain transformation in both directions results is very close to the input image. This helps maintain a balance between the extent of transformation in both directions. Looking at the loss function, we can see that the cyclic component of loss makes the most impact.

The Cycle GAN model seems to work pretty well for domains like Flower to Pattern and Fat to Fit body transformation. This can be a powerful and lucrative step forward in the fashion and fitness industries if the model is further improved and put to use. The goal of this work was to experiment with and analyze the model to see if it does well for the selected domain pairs, namely the Flowers to Patterns and Fat2Fit datasets. For our Fat2Fit dataset, data augmentation improved our results to a great extent which helped us since we had limited data. Our experiment on the Shark to Plane dataset enabled us to successfully conclude that the Cycle GAN model cannot be applied to just any two random domains especially when geometric and structural changes are necessary and the domains are not very similar.

For future direction, the CycleGAN model can be applied to a variety of different domains. Application focussed tasks can use this model on different domain pairs. The architecture of the model could also be changed to include a better discriminator like the one we used for evaluation. The quality of discriminator is bound to improve the quality of images generated by the generator.

Lastly, if we had paired datasets available, we could have achieved better results. The goal of our work was to perform image domain transformation in the unsupervised setting. We conclude that the Cycle GAN model performs very well for cross domain transformations as long as the two domains have enough structural similarity which invalidates the need for geometric transformation from input to output images.

References

[1] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016.

[2] J. Johnson, A. Alahi, and F. Li. Perceptual losses for real-time style transfer and super-resolution. *CoRR*, abs/1603.08155, 2016.

[3] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim. Learning to discover cross-domain relations with generative adversarial networks. *CoRR*, abs/1703.05192, 2017.

[4] M. Liu and O. Tuzel. Coupled generative adversarial networks. *CoRR*, abs/1606.07536, 2016.

[5] S. E. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. *CoRR*, abs/1605.05396, 2016.

[6] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[7] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016.

[8] J. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017.