Final Project- Progress Check 2

1st Maitry Trivedi

2nd Kevin Vora

***Abstract*—Our primary aim is to train a GAN model which can find the relationship between two different domains (Learning to Discover Cross-Domain Relations with Generative Adversarial Networks). We reveal our studies for this topic, implementation details and future plans as we progress towards a successful implementation.**

***Index Terms*—GAN, DiscoGAN, Discriminator, Generator**

I. Problem Formulation

Ian Goodfellow et al developed a new class of machine learning algorithms in 2014 called the "Generative Adversarial Network" (GAN). A GAN comprises of 2 networks, generator, and discriminator. These 2 networks compete with each other like in a contest in order to improve the performance of GAN. The key goal of GAN is, it learns to generate new data with the same statistics as the training set. For example: if a dataset of photographs is given as an input, GAN learns to generate new photos that are similar to the dataset but not in the dataset. There are various types of GAN which include Deep Convolutions GANs (DCGANs), Conditional GANs (cGANs), Discover Cross-Domain Relations with Generative Adversarial Networks(Disco GANS), etc. We narrow down our focus to DiscoGANs for this project as it has unique inspiration from how humans perceive.

Humans can easily understand the relationship between two different domains. For example, we always choose a suit jacket with pants or shoes in the same style to wear. DiscoGAN is the way of incorporating this sense into the machines which can be helpful in the recommendation systems. The main idea of DiscoGAN is to generate images of products in domain B given an image of domain A.

II. Key literature

There have been many developments in Deep neural Network, specifically in the domain of Generative Adversarial Networks. DiscoGAN is a unique advancement with real world changing applications. There are various resources openly available for our use in the form of blog, paper, openly available code, etc. We have identified following resources contributing to our project:

- https://github.com/carpedm20/DiscoGAN-pytorch
- https://medium.com/towards-artificial-intelligence/generating-matching-bags-from-shoe-images-and-vice-versa-using-discogans-8149e2cbc02
- https://arxiv.org/pdf/1406.2661.pdf
- https://arxiv.org/pdf/1703.05192.pdf
- https://vinodsblog.com/2018/11/23/generative-adversarial-networks-gans-the-basics-you-need-to-know/

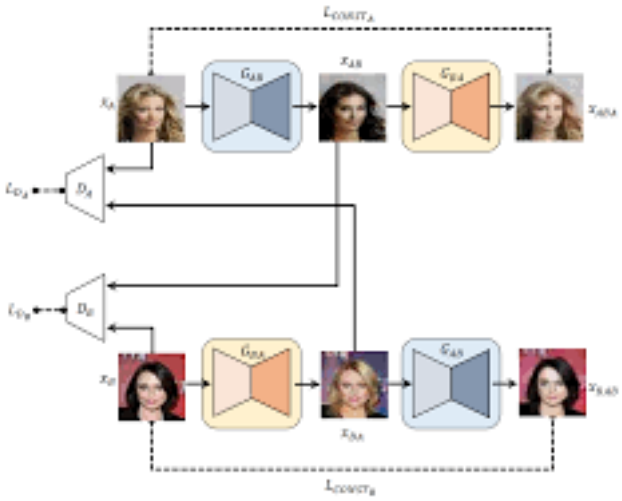


Fig. 1. DiscoGAN network structure

III. Division of Task

A. Mechanics of GAN

Steps involved in GAN can be explained via flowchart in figure 3:

Step 1: Problem definition
Select a specific input for generation of very specific output. For example input can be an image and output can be fake image etc.

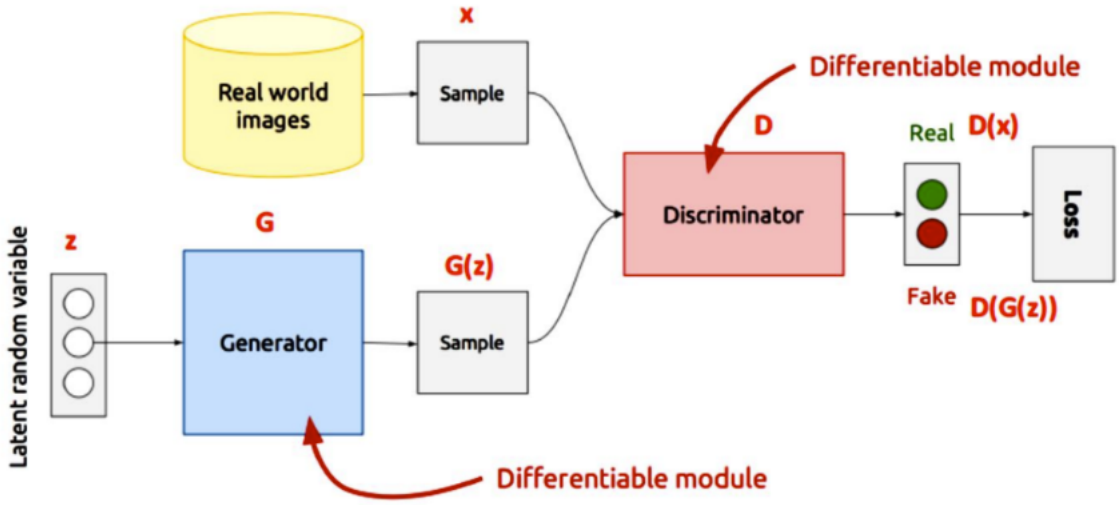
Step 2: Setting up and define architecture
As we can observe in the figure 2a[step 2], there is one generator which tries to generate images and discriminator tries to identify fake vs real images. Here, Z is some random noise(Uniform/Gaussian). Z can be thought as latent representation of the image.

Step 3: Discriminator training with real and fake data
As we observe in figure 2b[step 3], two steps can be described as follows: a) Fix generator weights, draw samples from both real-world and generated images. b) Train discriminator to distinguish between real-world and generated images.

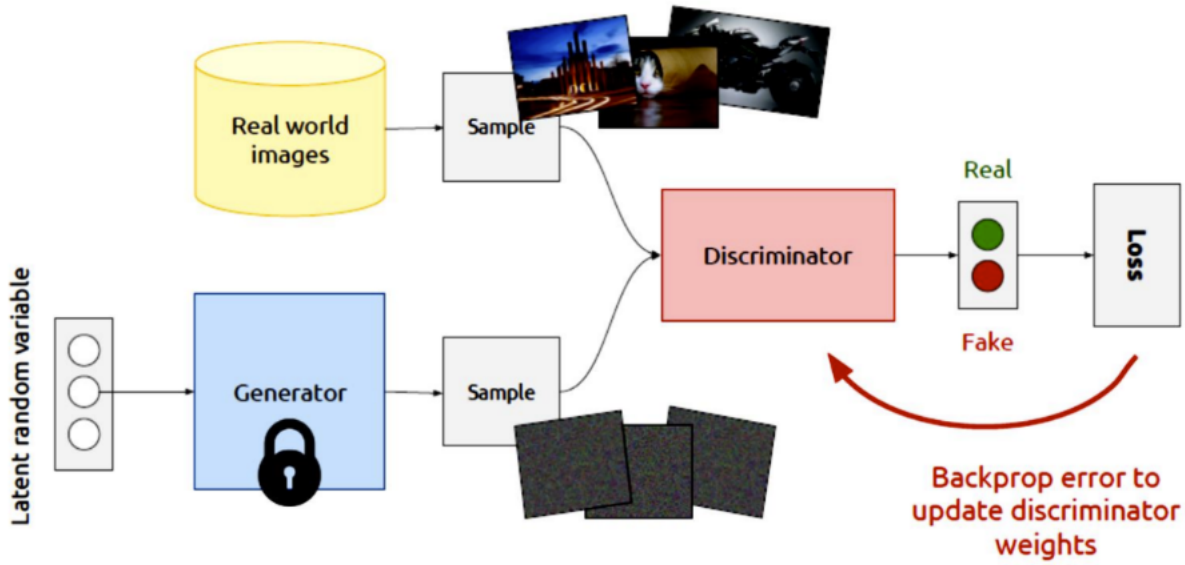
Step 4: Generator training with the output of discriminator
As we see in figure 2c[step 4], Fix discriminator weights, Sample from generator, Back-prop error through discriminator to update generator weights.

Step 5: Loop Iteration 3 to step 5

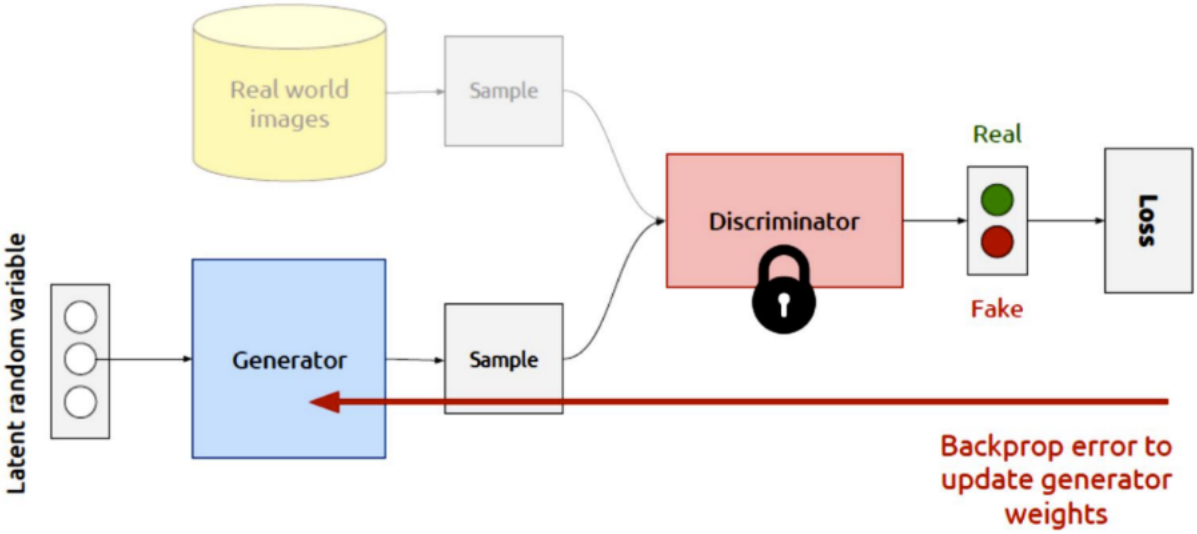
Step 6 & 7: Checking and validating



[Step 2]



[step 3]



[step 4]

Fig. 2. Understanding GAN

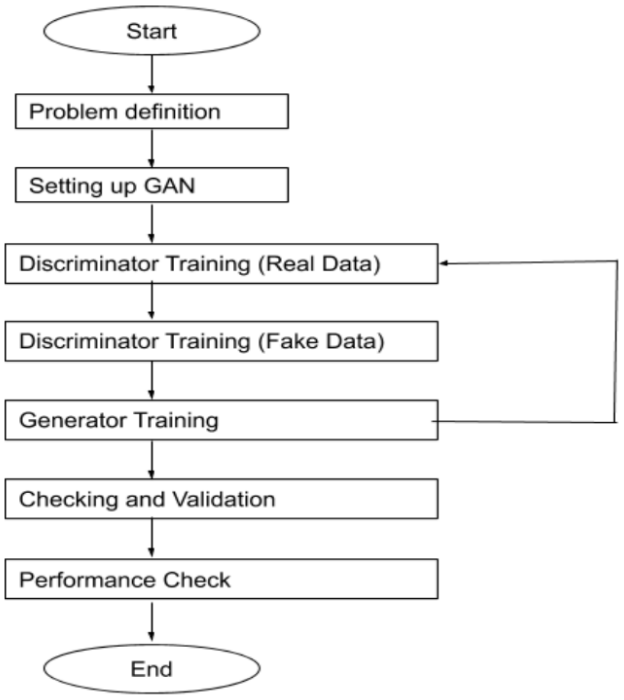


Fig. 3. Flowchart explaiing working of GAN

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$$

Fig. 4. Equation for Discriminator

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

Fig. 5. Equation for Generator

B. Deep dive into DiscoGAN

DiscoGAN is the one variation of GAN which is used to transfer textures and decorations of fashion pattern from one image to another image. As shown in Fig. 6 DiscoGAN consist of two generator G_{AB} and G_{BA}. One generator is to transfer handbags(domain A) to shoes(domain B). The second generator is to transfer shoes(domain B) to the handbags(domain A).
Steps:
1. Images of domain A is mapped into domain B via first generator.
2.The second Generator reconstructs the image from domain B to domain A.
3. The patterns and fashion is transferred in the images of domain B from domain A. During training, there is no need to pair the images from both domains. The model learns by itself.
4. When the image x_A is given as an input to the generator G_{AB} it generates the image x_{AB} which is similar to the image of domain B. Here the discriminator ensures that the generated image looks realistic for domain B.
5. When the image x_{AB} is given as an input to the generator G_{BA}, it generates the image x_{ABA}. Where this generated images is the image in the domain A.
6. Similarly the image of domain B is given as an input and the steps are repeated. After that, the difference between the ground truth image and generated image is calculated. L2 loss takes this error into account and both the generators G(ab) and G(ba) are optimized. But this L2 loss is just not enough to generate the realistic images. If we are mapping bags in domain A to shoes in domain B, we would have to ensure that x_B looks like a shoe. Therefore, both l2 loss and discriminator loss are combined to train the model. The cost function of the generator G_{AB} is shown in the Eq.3. The first term is l2 loss and the second term is discriminator loss. Similarly, the cost function of the generator G_{BA} is defined as Eq. 4.

Discriminator's cost function
The discriminator tries to discriminates the real and fake

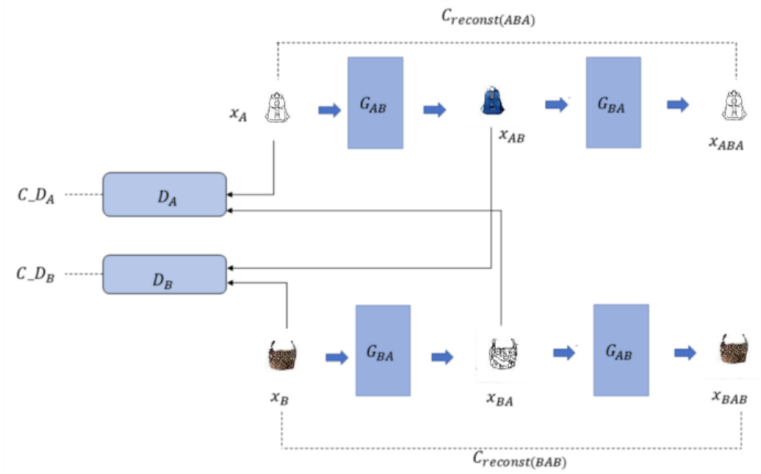


Fig. 6. DiscoGAN Architecture

$$C\_G_{AB} = C_{reconst(ABA)} + C_{D(AB)}$$
$$= \|x_A - G_{BA}G_{AB}(x_A)\|_2^2 - \log(D_B(G_{AB}(x_A)))$$

Fig. 7. Eq. 3

$$C\_G_{BA} = C_{reconst(BAB)} + C_{D(BA)} = \|x_B - G_{AB}G_{BA}(x_B)\|_2^2 - \log(D_A(G_{BA}(x_B)))$$

Fig. 8. Eq. 3

$$C\_D_B = - \mathop{\mathbb{E}}_{x_B \sim P(x_B)} [\log(D_B(x_B))] - \mathop{\mathbb{E}}_{x_A \sim P(x_A)} [\log(1 - G_{AB}(x_A))]$$

Fig. 9. Eq. 4

$$C\_D_A = - \mathop{\mathbb{E}}_{x_A \sim P(x_A)} [\log(D_B(x_A))] - \mathop{\mathbb{E}}_{x_B \sim P(x_B)} [\log(1 - G_{BA}(x_B))]$$

Fig. 10. Eq. 5

images.Eq. 4 and 5 represents the cost function of the both discriminators A and B. The total discriminator cost function is given by the Eq. 6.

C. Applications of DiscoGAN

This novel application as numerous use cases in real world. It has got business insight for recommendation as well as has the ability to make inference and correspondence between images of domain A and domain B. Few of this applications are:

- As we can see in the image (fig 12), bag belongs to domain A and shoes belong to domain B. Now, this process makes it easier for designers of bags and shoes to generate matching pairs of fashion apparels for ladies. Thereby helping the industry grow sales by realistic generated images.
- As shown in figure 13, translation of gender can also be achieved by DiscoGAN. Males belong to domain A and females belong to domain B. This can be fun when an application like snapchat uses realistic translation of gender for enjoyment and socializing.
- Most novel use case could be to know which colour on your hair would suit the most. In fig 14, blonde hair colour belongs to domain A and black hair colour belongs to domain B and we can observe how well the DiscoGAN model can generalize.
- In order to infer meaning out to images and establish correspondence, have a look at fig 15. Here car images belong to domain A and face images belong to domain B. This is a weird domain A and B pair but DiscoGAN is able to learn which direction the car points and generates face images looking in the same direction. Such performance of DiscoGAN reveals how powerful the network is.

$$C_D = - \mathop{\mathbb{E}}_{y_B \sim P(y_B)} [\log(D_B(y_B))] - \mathop{\mathbb{E}}_{x_A \sim P(x_A)} [\log(1 - G_{AB}(x_A))] - \mathop{\mathbb{E}}_{x_A \sim P(x_A)} [\log(D_B(x_A))] - \mathop{\mathbb{E}}_{x_B \sim P(y_B)} [\log(1 - G_{BA}(x_B))]$$

Fig. 11. Eq. 6

Fig. 12. Handbag images (input) Generated shoe images (output)

Fig. 13. Translation of gender

D. Implementation and simulation of DiscoGAN

Based on our understanding of DiscoGAN, we can follow an openly available code at [1] to implement DiscoGAN. Since it is very difficult to train a deep neural network like DiscoGAN even with good computational resources like CoLab.Therefore, we managed to tweak size of image, size of filters and most importantly we settled for a small dataset in order to demonstrate successful implementation of a fully functional DiscoGAN model.

We start by downloading facades dataset available at [2]. The dataset consists of 399 training images, 100 validation images and 106 images are for testing. A sample image can be seen in fig 16. As we can see image consists of 2 sub images of which the one on the left corresponds to domain A and one on the right corresponds to domain B. After downloading the dataset, we begin preprocessing the image by resizing images to 128x128 pixels. Also separation of images with respect to domain is done in preprocessing.

Preprocessed data is given input to our DiscoGAN model whose structure is as we discussed earlier. Various parameters of our model include:

- Number of filters in the first layer of generator and discriminator=32
- Optimizer= Adam
- loss= Mean square error

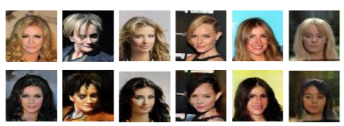


Fig. 14. Blond to black hair color conversion

Fig. 15. car to face translation

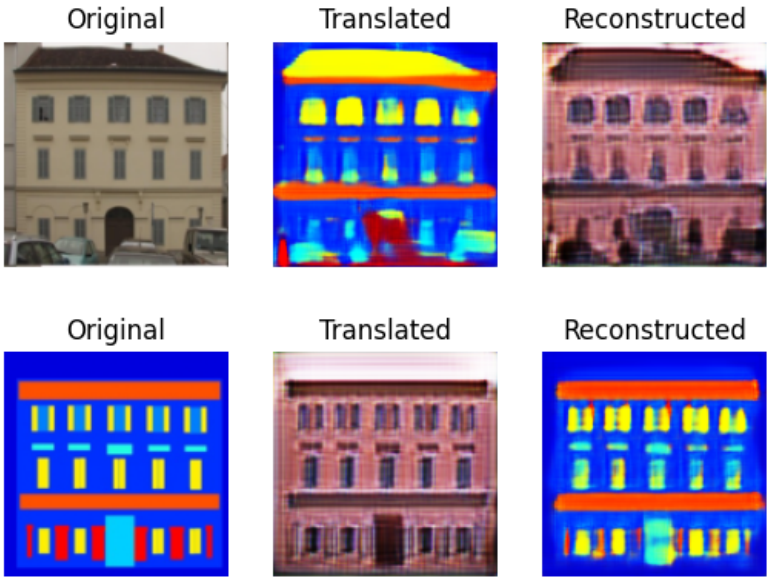


Fig. 16. Result 2

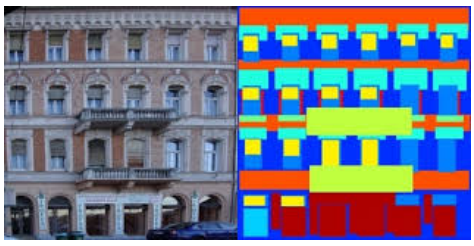


Fig. 17. Instance of dataset

- metrics= accuracy
- batch size= 1
- epoch= 15

IV. Results

In order to evaluate the result, we can consider following methods:

- **Evaluate quality of image reconstructed from translated Image:** Theoretically, if Generator 1 (translation of domain A to domain B) is able to map the details in the image with greater accuracy then using the same image as an input to generator 2 (translation from domain B to domain A) can result into an image very similar to input or exactly as the input(image of domain A). Thus, Looking at the quality of reconstructed image we can say how well the model is learning.
- **Evaluate Loss metrics:** If we compare the input image and the image reconstructed by generator 2 (translation from domain B to domain A) and compute loss, we can have a numeric idea of how well the model performs.

DiscoGAN usually takes a long time in the training process. Primary reason involves training 4 generators and 2 discriminators. As we know, training deep neural networks with compute resources available from google coLab can also take hours. In order to demonstrate working of the model, we performed experiments using 2 scenarios:

- **Case 1:** Using facades dataset with 16x16 filter and iterating over 10 epochs.
- **Case 2:** Using entire facades dataset and iterating over 20 epochs.

The Resulting images from both case 1 and case 2 are shown in Fig 18 and 16 Respectively. In case 2, the model is able to segment minor parts of the building accurately. For example in Figure 16, all the windows and one main gate is segmented properly.Therefore, the reconstructed image of the building from the segmented image is very similar to the original image. Moreover the image of building is generated properly from the given segmented image as an input. But in case 1, the translated and reconstructed images are not clear. As we can observe in Figure 18, all the Windows and one main gate is not segmented clearly. Therefore the reconstructed image is also not similar to the original input. We can observe that the reconstructed image for case 2 is better than case 1 because we have used larger dataset and more epochs for training the model in case 2. We can also conclude this by observing the results from Table 1. The generator and discriminator loss for case 2 is similar or less than case 1. Therefore the generator

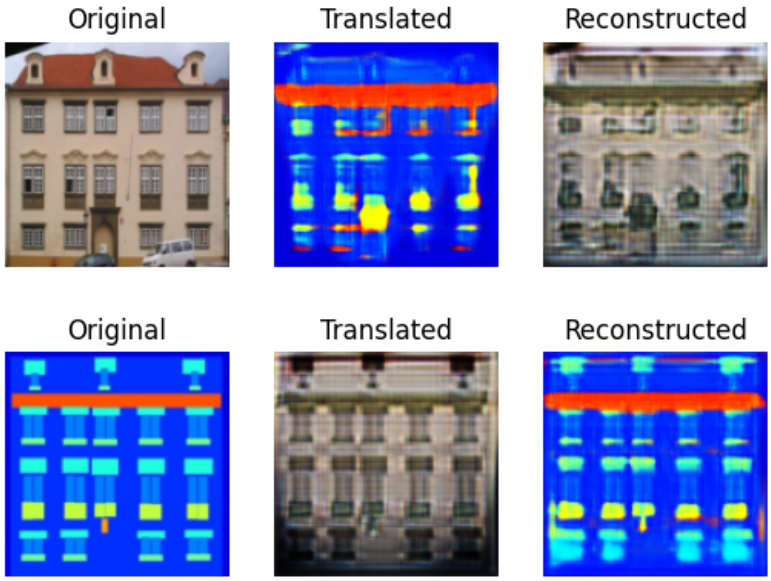


Fig. 18. Result 1

TABLE I
RESULT

	Generator loss	Discriminator loss
Case 1	2.667938	0.158142
Case 2	2.844698	0.070919

and discriminator for case 2 are well trained as the generator is able to create realistic image which discriminator fails to classify as fake image. Therefore, we can conclude that by using more training data, large filter size, and training for large number of epochs, the DiscoGAN model can perform extremely well and generate realistic images.

V. SUMMARY

As a result we can demonstrate the power and use of GAN by completely understanding it's working. In second phase we learn detailed working of DiscoGAN. It is clear from the applications that power held by such networks is beyond what we could have imagined. This motivated us to develop a successful implementation of DiscoGAN on a small facades dataset. Finally, by the end of final report we showcase results of implementation. This helps us conclude that a lot of compute resource is needed to train generative adversarial networks, and precisely DiscoGAN. We can also see how useful these techniques can be and they play a great role in motivation us to work in this direction in the future.

In future this project can be extended to different applications as well. Given more compute resources even better result can be obtained in future.

REFERENCES

[1] https://github.com/eriklindernoren/Keras-GAN
[2] https://people.eecs.berkeley.edu/ tinghuiz/projects/pix2pix/datasets/
[3] https://medium.com/towards-artificial-intelligence/generating-matching-bags-from-shoe-images-and-vice-versa-using-discogans-8149e2cbc02