Conditional Image Generation using DCGAN

Navpreet Singh
CIMS
New York University
Email: ns4647@nyu.edu

Aashiq Mohamed Baig
CIMS
New York University
Email: amb1558@nyu.edu

Sarthak Joshi
CIMS
New York University
Email: sj2810@nyu.edu

Generative Adversarial Networks (GANs) have become the state-of-the-art tool for generative modeling especially with their success in facial image generation. Two of the most remarkable evolutions of GANs are Conditional GANs (CGAN) and Deep Convolutional GANs (DCGANs) which allow us to respectively control the generated output by specifying the class we want to condition on and leverage CNNs to perform unsupervised learning on images. In this work, we combine these 2 nets and apply the resultant Conditional DCGAN (CDCGAN) on 2 distinct domains, namely, celebrity faces using the CelebA dataset and landscapes using a self-compiled landscape image dataset. We demonstrate controlled image generation across these 2 domains using our CDCGAN model and illustrate performance analyses for the same.

1 Introduction

GANs offer a novel way to train generative models and allowed for an interesting application in unsupervised learning in the form of generating hyper-realistic artificial images. DCGANs improved upon this model by leveraging modern CNN capabilities to scale up the training speeds and improve stability of the model. Lastly, conditional GANs extend the utility of GANs by offering control over the output images and allowing manipulation of the generative model's bias towards certain specific features.

We combine these broad ideas and develop a Conditional DCGAN model. Our experiments involve majorly generating recognizable images in the 2 domains of celebrity faces and landscapes. In the former use case, we control output features such as hair-color and gender of the face generated whereas in case of landscape, we manipulate common landscape fetures such as mountains, trees, water bodies, etc.

2 Related Work

2.1 Conditional Generative Adversarial Networks

Generative adversarial nets were recently introduced as an alternative framework for training generative models in order to sidestep the difficulty of approximating many intractable probabilistic computations. CGANs evolved out of this framework as a model that allows us to direct the generation process by conditioning the discriminator and the generator.

GANs can be extended to a conditional model if both the generator and discriminator are conditioned on some extra information $\boldsymbol{y}$. $\boldsymbol{y}$ could be any kind of auxiliary information, such as class labels or data from other modalities. We can perform the conditioning by feeding $\boldsymbol{y}$ into the both the discriminator and generator as additional input layer.

In the generator the prior input noise $pz(z)$, and $\boldsymbol{y}$ are combined in joint hidden representation, and the adversarial training framework allows for considerable flexibility in how this hidden representation is composed. 1 In the discriminator $\boldsymbol{x}$ and $\boldsymbol{y}$ are presented as inputs and to a discriminative function (embodied again by a MLP in this case).

2.2 Deep Convolutional Generative Adversarial Networks

Learning reusable feature representations from large unlabeled datasets has been an area of active research. In the context of computer vision, one can leverage the practically unlimited amount of unlabeled images and videos to learn good intermediate representations, which can then be used on a variety of supervised learning tasks such as image classification.

DCGANs have proven to be a more stable set of architectures for training GANs and allowed for training higher resolution and deeper models.

Fig. 1. Image Tagging

3 Approach

3.1 Preparing the Data

Based on our initial research into GANs and their use cases, we knew that conditionining on GANs, and more specifically, the number of classes significantly inflate the required training data. We, therefore, sought out large, easily available and accurately tagged image datasets (preferably with multiple tags on a single image corresponding to the features we want to control). We ended up with the following two datasets:

3.1.1 CelebA

CelebFaces Attributes Dataset (CelebA) is a large-scale face attributes dataset with more than 200K celebrity images, each with 40 attribute annotations. The images in this dataset cover large pose variations and background clutter. CelebA has large diversities, large quantities, and rich annotations, including

- 10,177 number of identities,
- 202,599 number of face images, and
- 5 landmark locations, 40 binary attributes annotations per image.

3.1.2 Self-compiled Image Landscapes Dataset

As part of our data discovery, we sourced landscape images, both in terms of labelled datasets and bulk image downloads. With the vast majority of our data being unlabelled, we used the pretrained **Resnet50** model in PyTorch to generate confidence scores for our target classes and used these scores to tag the images. The image tags we focused on were Sea, Forest, Mountain, Glacier, River, Snow and Sky.

Using a specific threshold for confidence score, we generated an attribute file containing our **one-hot-encoded** feature vectors for all our images. The following figure illustrates an example of this.

3.2 The Model

GANs can be extended to a conditional model if both the generator and discriminator are conditioned on some extra information $\boldsymbol{y}$. $\boldsymbol{y}$ can typically be a class label or tag.

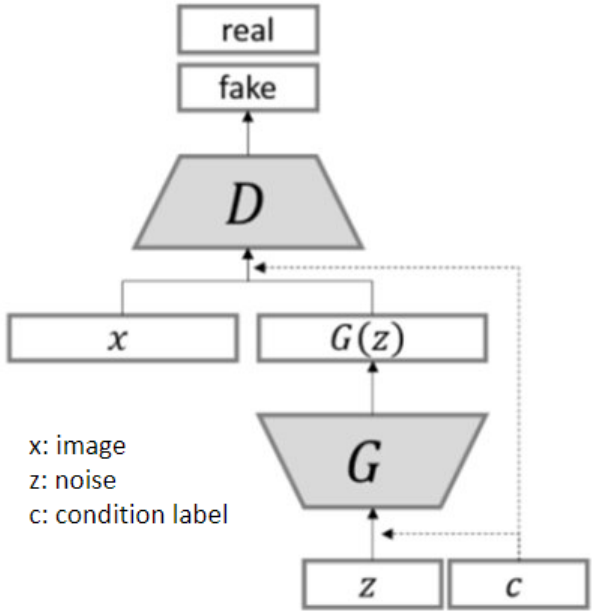

Fig. 2. Conditional GAN

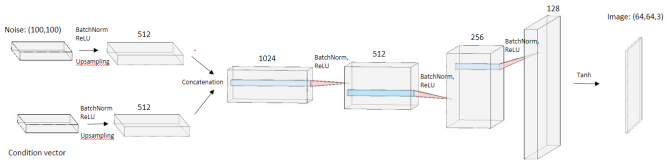

Fig. 3. Generator

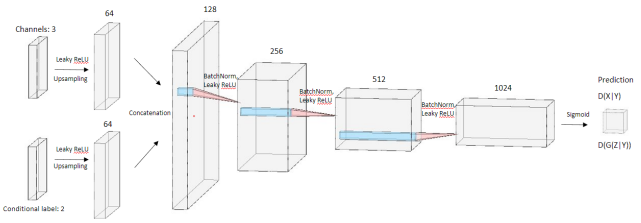

Fig. 4. Dsicriminator

The core idea is to train a GAN with a conditioner, we can perform the conditioning by feeding $\boldsymbol{y}$ into the both the discriminator and generator as additional input layer. This is encapsulated in the following equation that signifies **min-max** game between the discriminator (4) and the generator (3).

$$\min_G \max_D V(D,G) = E_{x\ p_{data}(x)}[logD(x|y)] + E_{z\ p_z(z)}[log(1-D(G(z|y)))]$$

3.2.1 Network architecture

- **Generator**:
  1. Hidden Layers: Four 4x4 strided convolutional layers (1024, 512, 256, and 128 kernels, respectively) with ReLU
  2. Output Layer: 4x4 strided convolutional layer (4096 nodes = 64x64 size image) with Tanh
  3. Batch Normalization is used except for output layer
- **Discriminator**:
  1. Hidden Layers: Four 4x4 convolutional layers (128, 256, 512 and 1024 kernels, respectively) with Leaky ReLU
  2. Output Layer: 4x4 convolutional layer (1 node) with Sigmoid
  3. Batch Normalization is used except for 1st hidden layer and output layer
  4. Loss Function: Binary Cross Entropy used due to two output labels.
- We are using the Adam optimizer.

3.3 Constraints

A critical factor of GAN-based generation is the accuracy and precision of image tags across the training dataset. Given the GAN discriminator and generator both heavily rely on tags, reducing the training loss was a significant challenge as our Landscape Image Dataset,

- Had fewer training samples
- Had fewer cases of images with appropriate multiple tags

4 Experimental Results

4.1 Training

4.1.1 Image Transformations

- The original images from the dataset were sized at 178x218. In order to fit our convolution layer, reduce loss and improve training times, we resized these images to 64x46.
- We centered the images to reduce background noise.

4.1.2 Hyperparameters

We used the following hyperparameters for our model:

- label_dim = 2 : Number of classes per attribute (male/female or black/brown hair)
- G_input_dim = 100x100 : Size of generator input noise which is converted to the generated image.
- G_output_dim = 64x64x3 : Size of generator output.
- D_input_dim = 3
- D_output_dim = 1 : Discriminator predicts a single label (real or fake).
- num_filters = [1024, 512, 256, 128] : Convolution filter sizes.

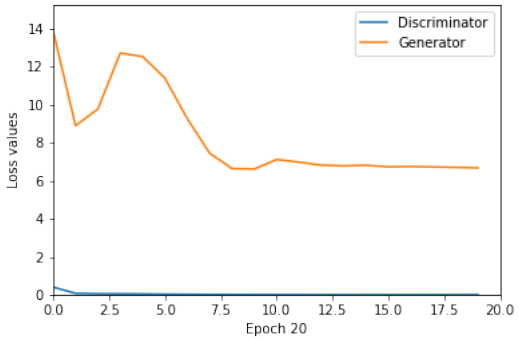

Fig. 5. Controlled Landscapes Loss Plot

- learning_rate = 0.0002
- betas = (0.5, 0.999)
- batch_size = 128
- num_epochs = 60

4.2 Results on Landscapes

Generating controlled landscapes was particularly challenging on the landscape dataset due to the fact that:

- **Insufficient Data**: At best, we had 24,000 images for a given feature attribute, while on an average, each attribute had fewer than 3000 images associated with it.
- **Inadequate Tagging**: Since most of our landscape dataset was manually compiled, there weren't enough attribute labels on a significant chunk of the imagees, with most images largely being associated with a single tag. This prevents the model from understanding the key differences across the attributes on a single image containing multiple such attributes.

We ran a total of 20 epochs on a subset of the dataset containing images with either forests or mountains in them.

We observed that the generator loss does not decrease significantly and as evidenced by the output, the generated output quality is low. The lack of input with multiple overalapping tags, specifically, images with both mountains and forests prevented the model from significantly identifying the key aspects of these attributes.

4.3 Results on CelebA

Going on the **CelebA** dataset, we had a few major advantages:

- 200k images
- 40 feature attributes per image

We performed our training with a focus on 2 specific attributes out of 40, namely hair color and gender. Our attribute choices were motivated mainly by the fact that changes in hair color and gender-specific features are far more conspicuous than changes on the others. In this experiment, we filtered the training data on attribute vectors with either the **black hair** or **red hair** attribute set. No filters were required for genders.

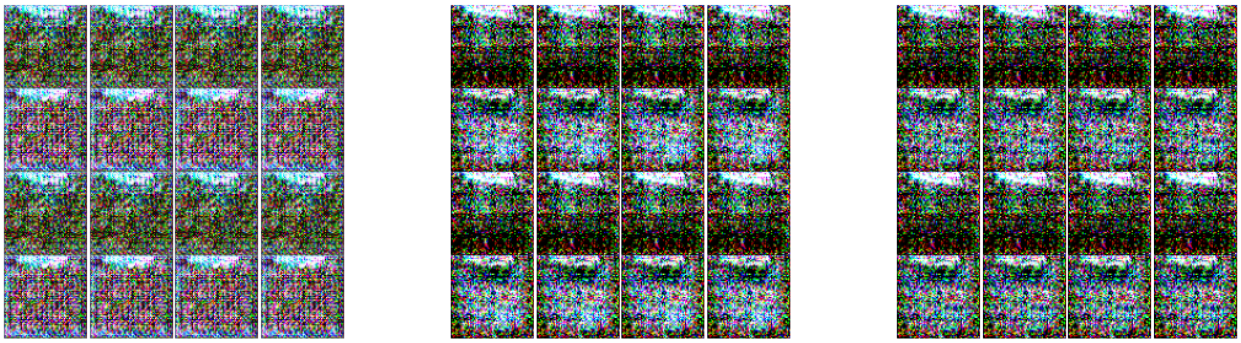

Fig. 6. Generated Landscape pairs

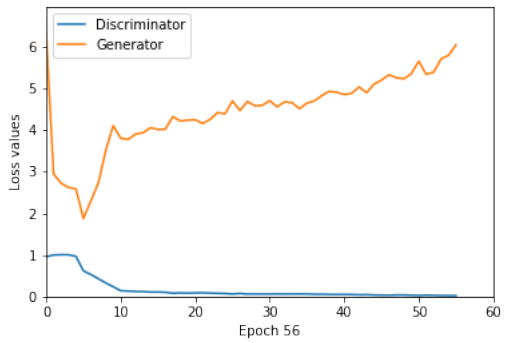

Fig. 7. Controlled Gender Loss Plot

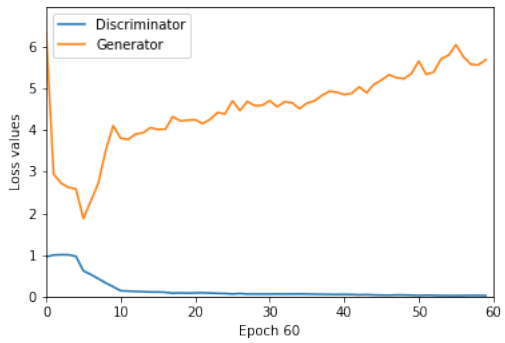

Fig. 8. Controlled Hair Color Loss Plot

We ran a total of 60 epochs on two different sessions (one for the gender attribute and other for hair color). Here are the output progressions:

1. **Controlled Hair Color**: We observe steady rapid increase in output quality from Epoch 1 to 20 (9) and from thereon, the progression plateaues. This is also reflected in the generator/discriminator loss (8).
2. **Controlled Gender**: We observe similar early progression (10) as we did for hair color, but the model output quality is more ambiguous in this case. This could be due to the less obvious differences in terms of gender on the faces. The loss reduction plateaues fairly early as well (7)

5 Conclusion

This work demonstrates the capabilities of Conditional DCGANs as a powerful tool for unsupervised learning and generational applications while also highlighting the limitations of conditional GANs in general. We can conclude that :

1. GANs in general can generate random images with a highly likeliness to the input with a fairly modest input dataset size (3000 images usually enough).
2. Conditional GANs, however, require siginificantly more data with sophisticated tagging. This is due to the fact that conditioning on specific features requires the model to understand nuanced differences. This is highlighted in the performance difference across the results of landscape generation and face generation.
3. Depending on the nature and distinction of features being targeted, the training data requirement can significantly vary. For instance, in the case of face generation, hair color distinction was generated than gender differences.

Code Repository

https://github.com/navpreetnp7/conditional-DCGAN

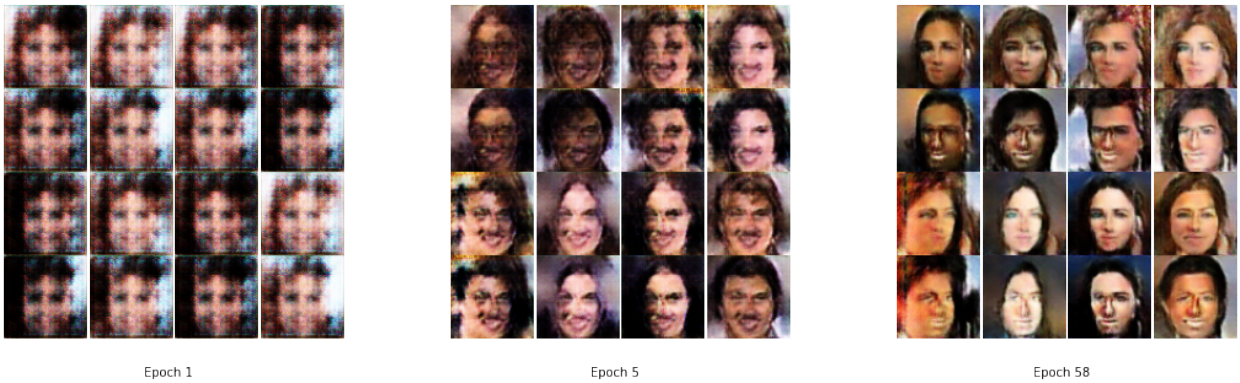

Fig. 9. Generated 8 Face pairs with black and brown hair specified. Pairs are vertically aligned with top image having brown hair color

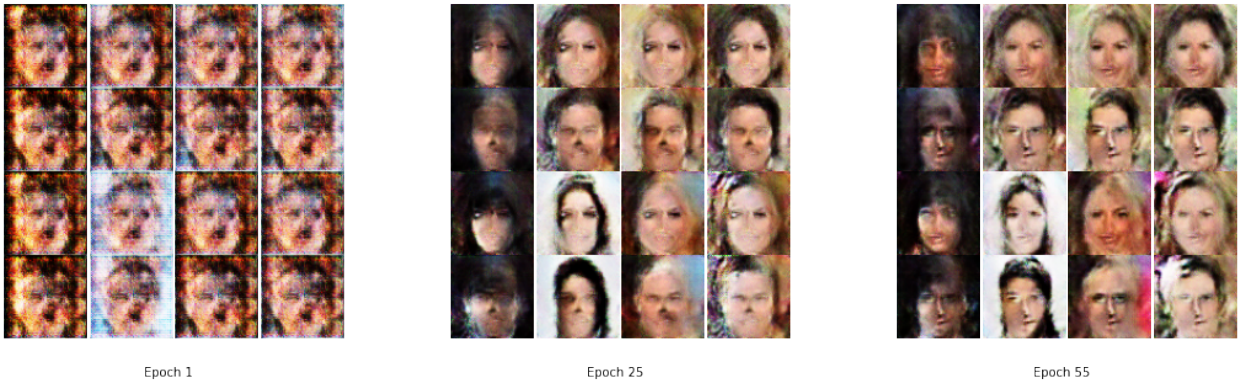

Fig. 10. Generated 8 Face pairs with gender (male/female) specified. Pairs are vertically aligned with top image having female gender

References

[1] Mehdi Mirza, S. O. "Conditional generative adversarial nets". https://arxiv.org/abs/1411.1784.
[2] Alec Radford, Luke Metz, S. C. "Unsupervised representation learning with deep convolutional generative adversarial networks". https://arxiv.org/pdf/1511.06434.pdf.
[3] Large-scale celebfaces attributes (celeba) dataset. http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html.
[4] Kaggle imagecitylandscape. www.kaggle.com/nokaii/imagecitylandscape.
[5] Kaggle landscape classification. www.kaggle.com/huseynguliyev/landscape-classification.
[6] pytorch-mnist-celeba-cgan-cdcgan. github.com/znxlwm/pytorch-MNIST-CelebA-cGAN-cDCGAN.