CS230

Using GANs for new designs of chairs

Matias Castillo
Department of Electrical Engineering
Stanford University
matiasct@stanford.edu

Abstract

Generative Adversarial Networks (GANs) are one of the most promising tools for generating realistic images. In this project, a Deep Convolutional GAN (DC-GAN) was implemented and trained to generate realistic images of chairs. Training the model with a small dataset of internet images of chairs (2,300 images) allowed the generation of images that replicated some of the chairs shapes. Different transfer learning techniques using the large Imagenet dataset were implemented without major improvements in the results. Finally, an increase in the chairs dataset size (1,000 additional images) and longer training improved the performance of the algorithm significantly, generating some realistic images of chairs. The new designs include chairs with double back, three legs or one arm.

1 Introduction and related work

The state-of-the-art generative models are getting able to generate photorealistic images. If we could use those models to generate realistic images of specific objects like chairs, we could automatically generate new designs of those objects, disrupting the design industry. Moreover, using conditional generative techniques we could based this new designs on our own preferences. With this motivation in mind, in this project I implement and train a generative adversarial network (GAN) to generate realistic images of chairs.

Generative Adversarial Networks were first implemented in 2014 by Ian Goodfellow et.al [1] and are now outperforming most of the state of the art generative models. Their success is due to their innovative approach, which includes both a discriminator and a generator that compete between each other making their learning very efficient. Some of the applications of GANs include super-resolution [2], text translation into images [3], image inpainting [4] and video generation [5]. Current research on generating photorealistic images using GANs have showed great results to the level that people do not recognize fake images [6]. However, most of the successful results have been achieved using huge datasets (over 100,000 images) for training, in which images have all very similar high level features as we see in faces [6][7], shoes [8] or landscape environments [7].

One of the challenges of using GANs is that usually they are very unstable to train, and for that reason several variations of GANs have been developed in the last years [9][10][11]. One of the most successful variations is the one proposed by Radford et. al [7] called Deep Convolutional GAN (DC-GAN), which they proved to show stable training on several datasets and allowed training on higher resolution and deeper models. The main improvements were to replace any pooling layers with strided convolutions to allow the network to learn its own spatial downsampling (discriminator) and upsampling (generator), to use batch normalization with zero mean and unit variance in most of the layers of both the discriminator and generator, and to avoid the use of hidden fully connected layers (to improve convergence speed).

Other techniques that have showed significant improvement in training GANs are feature matching and virtual batch normalization (VBN) which were first introduced by Salimans et. al [12]. Feature matching addresses the instability of GANs by changing the objective of the generator to match the expected values of an intermediate layer of the discriminator, which prevents an overtraining of the generator over the discriminator. On the other hand, VBN avoids the output for a specific input example to be highly dependent on the other inputs of the current minibatch, by normalizing each of the examples based on a reference batch fixed from the beginning of the training.

In this project, I implement a DC-GAN architecture [13] [14] and train it to generate photorealistic images of chairs. I first use a small dataset of only 2,300 chairs and then try to improve the model performance by applying Transfer Learning and other techniques.

2 Methods

2.1 Deep Convolutional GAN

A Deep Convolutional GAN (DC-GAN) was implemented using the following architecture (see Figure 1):

Generator: a 100 dimensional random uniform distribution as input. 5 deconvolution layers (fractional strided convolutions), with 4x4 filters. Batch normalization in all the hidden layers except the first one. Used tanh activation for output layer and ReLU for hidden layers.

Discriminator: a 64x64x3 image as input. 5 strided convolutions, with 4x4 filters. Batch normalization in all the hidden layers except the first one. Used sigmoid function in the output layer, and LeakyReLU in hidden layers.

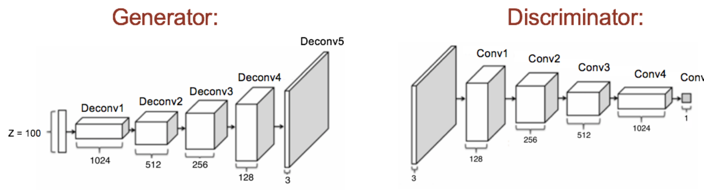

Figure 1: DC-GAN architecture.

2.2 Cost functions

The binary cross entropy loss was used for the cost function of the discriminator:

$$J_D = -\frac{1}{m_D}\sum_{i=1}^{m_D} y_D^{(i)} \log(D(x^{(i)})) - \frac{1}{m_G}\sum_{i=1}^{m_G}(1 - y_G^{(i)}) \log(1 - D(G(z^{(i)})))$$

where m_D are the number of real images, m_G are the number of fake images from the generator that are used as input for the discriminator, $D(x^{(i)})$ is the output of the discriminator for the real input $x^{(i)}$, $D(G(z^{(i)}))$ is the output of the discriminator for the fake input $G(z^{(i)})$, $y_D^{(i)}$ is the correct label of real images (equal to 1) and $y_G^{(i)}$ is the correct label of fake images (equal to 0).

For the generator was used the following non-saturating cost function:

$$J_G = -\frac{1}{m_G}\sum_{i=1}^{m_G} \log(D(G(z^{(i)})))$$

where m_G and $D(G(z^{(i)}))$ are the same as the ones explained above.

2.3 Other implementation details

- Weights initialization: zero mean and 0.02 standard deviation.
- Adam optimization algorithm with $\beta_1 = 0.5$ and $\beta_2 = 0.999$.

- Learning rate: 0.0002.
- Minibaches of size 128.

2.4 Inception score

The inception score is a quantitative measure that is usually highly correlated with human judgment in how realistic are the generated images. Therefore, it is getting very popular as a metric for evaluating GANs performance.

The inception score evaluates a distribution of generated images by classifying the images using an Inception Model [12] and then calculating the KL divergence between the conditional label distribution of each image $p(y|x)$ and the marginal distribution $\int p(y|x = G(z))dz$. Images with meaningful objects should have conditional distribution with low entropy, and distributions with high variety of images should have marginal distribution with high entropy. Therefore, distributions of more realistic images should have higher inception scores.

$$IS = exp(\mathbb{E}_x KL(p(y|x)||p(y)))$$

To calculate the inception score was used the Pytorch inceptionv3 model [15].

3 Datasets and data augmentation

The open source MNIST dataset of 60,000 handwritten digits images was used for testing the DC-GAN implementation.

The open source Imagenet dataset of over 1 million images was used for testing the DC-GAN implementation and for transfer learning.

A new dataset of chairs was created by downloading chairs images from Google Images. The dataset contain 2,300 images of single chairs with white background. The chairs have different styles, sizes, colors and orientations (see Figure 2).

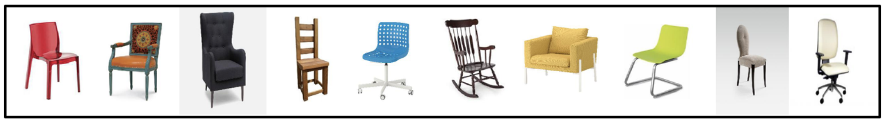

Figure 2: Examples from the Chairs Dataset

The images of the Chairs Dataset were downsized to 64x64 pixels, and data augmentation of horizontal flips was done randomly while training.

4 Experiments and results

4.1 Large datasets training and results

First, the DC-GAN implementation was tested in the MNIST and Imagenet dataset to check if the discriminator and generator were learning correctly, and to finetune the model if necessary. Most of the hyperparameters values (defined in Section 2.3) were chosen from the typical values found in DC-GAN literature and were kept fixed. The only hyperparameter finetuned was the learning rate, which was implemented with both 0.0001 and 0.0002 values. After analyzing the generated images using human judgment, the learning rate was fixed to 0.0002 as its seems to show better results.

The model was able to generate realistic images of handwritten digits after 20 epoch of training (see Figure 3). Also, the model was able to extract several high level features from Imagenet dataset after 5 epoch of training as seen in Figure 4.

4.2 Chairs Dataset training and results

Without any pretraining, the DC-GAN implementation was trained using the Chairs Dataset. As seen in the left graph of Figure 5, the cost functions of both the generator and discriminator were

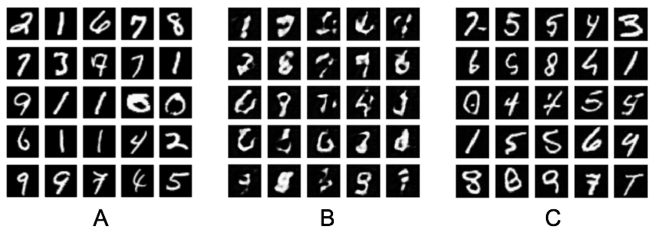
Figure 3: (A) MNIST dataset examples. (B) Generated images after 5 epoch. (C) Generated images after 20 epoch

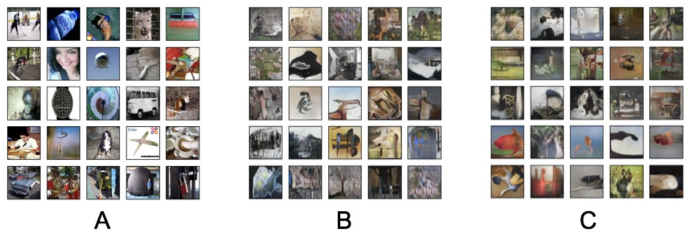
Figure 4: (A) Imagenet dataset examples. (B) Generated images after 1 epoch. (C) Generated images after 5 epoch

oscillating over the training, implying that both were learning at similar rates allowing the generator to keep improving over time. Every time the generator improved, its cost was reduced and the discriminator cost increased because it was harder to detect if the image was fake. On the other side, when the discriminator improved, its cost was reduced and the generator cost increased.

Every 10 epoch of training, 30,000 images were generated using the partially trained generator. For each distribution of 30,000 images, the mean inception score was calculated. As seen in the right graph of Figure 5, the mean inception score was increasing over the training iterations, which means that probably more realistic images are been generated after longer training periods.

Figure 6 shows randomly generated images after 10, 50 and 150 epoch of training. After 10 epoch of training the generator was able to extract some of the main features of chairs and it kept improving over time getting much more photorealistic images after 150 epoch.

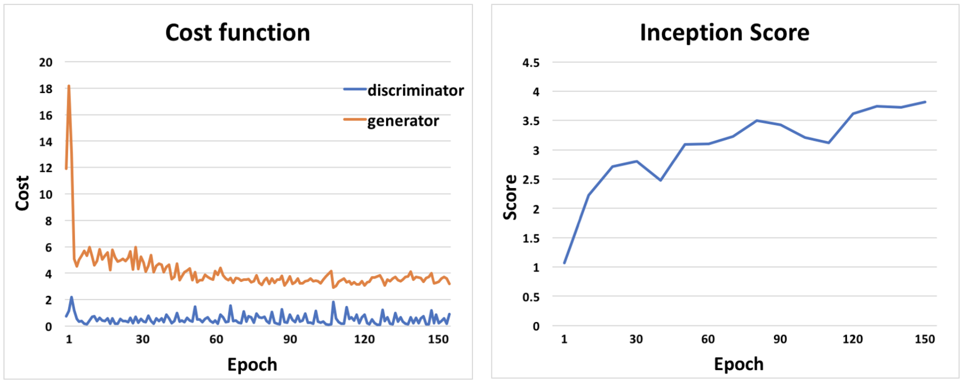

Figure 5: Left: cost function of generator (orange) and discriminator (blue). Right: mean inception score of 30,000 generated samples at different stages of training.

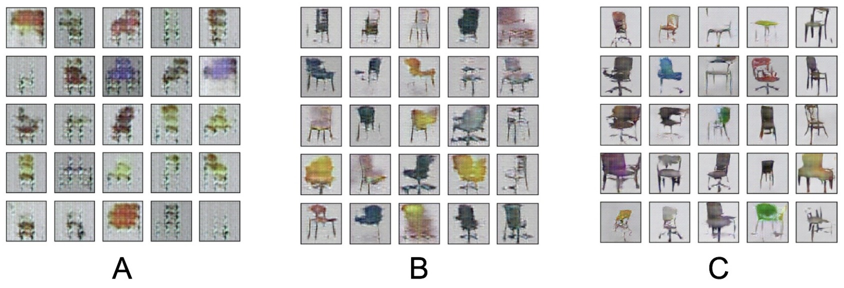
Figure 6: (A) Generated images after 10 epoch. (B) Generated images after 50 epoch. (C) Generated images after 150 epoch

4.3 Transfer Learning

With the goal of improving the current results, transfer learning experiments using the Imagenet dataset were performed. The idea was to analyze if some features extracted from Imagenet images could be useful for generating chairs images. Unfortunately, most of the experiments were unsuccessful and the results obtained were not better than the ones obtained by training our model from scratch.

The model was first trained with Imagenet and later it was retrained using the Chairs Dataset. When retraining all the parameters, the discriminator was too good too fast not allowing the generator to learn. Even restricting the discriminator learning (with a threshold in the cost function value) the generator was not able to improve. Another experiment, in which only the last layer of both generator and discriminator were retrained, ended up with the same results. Finally, when pretraining only the generator parameters and not the discriminator's, the model was able to learn successfully. However, the generated images were not better than the ones generated without any pretraining.

Probably, the features extracted from the Imagenet pretraining were not useful for creating images of chairs because the chairs images were very different (with white backgrounds) and Imagenet generator was not able to generate any objects nor shapes similar to the ones of chairs.

4.4 Additional improvements: more data and longer training

1,000 additional chairs images were added to the dataset to see if this improved the performance of the generator. Also, since Figure 5 showed that the model was still learning after 150 epoch, longer training was applied to see if this also improved the results.

As show in Figure 7, both the additional images and the longer training improved the results significantly. As you can see in Figure 7B, the model was able to generate very realistic chairs images, including some with abnormal shapes, with double back, with three legs and with one arm.

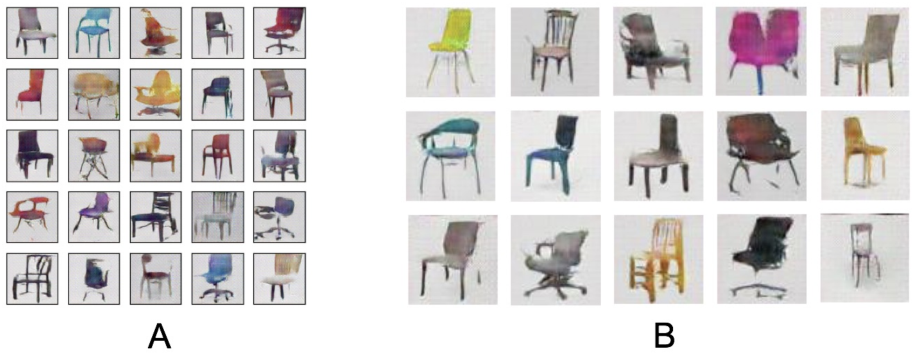
Figure 7: (A) Generated images after 200 epoch. (B) generated images after 400 epoch.

The high variety of chairs types generated shows that Mode Collapse did not occurred in this model. Also, given that some of the images are very different from the chairs of the dataset, we can conclude that the model is not generating these images from pure memorization of the dataset.

5 Conclusion and future work

In this project we proved that GANs can also be trained with small datasets getting photorealistic images of objects.

Transfer learning techniques did not improve the performance of the model. Probably, using a pretrained model from a dataset with similar features as those of chairs could improve the benefits of transfer learning for chairs images generation.

As future work, I want to explore how feature extraction and virtual batch normalization could impact the performance of the model. Also, a quantitative measure is required to confirm that the results were not based on dataset memorization.

Finally, it would be interesting to try conditional GANs to modify the generator according to the user preferences.

Github repository with code

code in `https://github.com/matiasct/GANs_Project/`

References

[1] Goodfellow, I., et. al (2014). *Generative adversarial nets*. In Advances in neural information processing systems (pp. 2672-2680).

[2] Ledig, C., et. al (2016). *Photo-realistic single image super-resolution using a generative adversarial network.* Twitter. arXiv preprint.

[3] Reed, S., et. al (2016). *Generative adversarial text to image synthesis*. arXiv preprint arXiv:1605.05396

[4] Burlin, C., Le Calonnec, Y., and Duperier, L. (2017) *Deep Image Inpainting.* Stanford CS231N course project.

[5] Michael, C., Kang, S., and Kim, S. (2017) *(Re)live Photos: Generating Videos with GANs.* Stanford CS231N course project.

[6] Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2017). *Progressive growing of GANs for improved quality, stability, and variation.* arXiv preprint arXiv:1710.10196.

[7] Radford, A., Metz, L., and Chintala, S. (2015). *Unsupervised representation learning with deep convolutional generative adversarial networks*. arXiv preprint arXiv:1511.06434.

[8] Deverall, J., Lee, J., and Ayala, M. (2017) *Using Generative Adversarial Networks to design shoes: the preliminary steps* Stanford CS231N course project.

[9] Arjovsky, M., Chintala, S., and Bottou, L. (2017). *Wasserstein gan*. arXiv preprint arXiv:1701.07875.

[10] Mirza, M., and Osindero, S. (2014). *Conditional generative adversarial nets*. arXiv preprint arXiv:1411.1784.

[11] Zhu, J. Y., Park, T., Isola, P., and Efros, A. A. (2017). *Unpaired image-to-image translation using cycle-consistent adversarial networks*. arXiv preprint arXiv:1703.10593.

[12] Salimans, T., et. al (2016). *Improved techniques for training gans*. In Advances in Neural Information Processing Systems (pp. 2234-2242).

[13] https://github.com/znxlwm/pytorch-MNIST-CelebA-GAN-DCGAN

[14] https://github.com/pytorch/examples/tree/master/dcgan

[15] https://github.com/sbarratt/inception-score-pytorch