Face image generation using GANs

Bogdan Vlad
Babeș-Bolyai University
bogdan.vlad@stud.ubbcluj.ro

Abstract

Generative adversarial networks have revolutionised the way we approach generative models in deep learning. In this work we attempt to generate celebrity face images of a low resolution using such an approach.

1. Introduction

In a GAN (Generative Adversarial Network) [2] we train two neural networks: A discriminator *D* which is a classifier, and a generator *G* which generates outputs corresponding to the type of inputs *D* accepts. The idea came from Goodfellow et. al [2] in 2014, after a couple of years in which it was clear we could build good image classifiers (AlexNet [5]). Now the task was to build good generators but instead of starting from scratch, the authors cleverly came up with the idea of harnessing the power of the already existing powerful image classifiers.

2. Related work

We now look at the original GAN idea and the improvements which were subsequently made.

2.1. Original GAN

Goodfellow et al, [2] pioneered the idea of generative adversarial networks in 2014. Many of the ideas presented in this paper are still relevant and used today, even if there were many problems initially.

As stated already, in a GAN there are neural networks, the discriminator *D* and the generator *G*. The job of the discriminator D is to classify if the input x comes from the distribution of the data or is a so called "fake" input. The generator G creates (from random noise) inputs x. Through an adversarial contest between D and G, the generator G becomes better at creating "fake" inputs which look like real input, while the discriminator D becomes better at classification between real and fake inputs.

The discriminator and the generator are put into an adversarial competition, where the discriminator is trying to discern the generated inputs of the generator. The model is trying to optimize the following function:

$$\min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))]$$

It turns out that the nature of the GAN architecture and the above loss function will create a number of problems, among which:

- Oscillation between generator and discriminator loss
- Mode collapse
- Discriminator is too strong, such that the gradient for the generator vanishes and thus the generator can't keep up
- Discriminator is too weak
- Hyperparameter sensitivity

Some of these problems are still ongoing research subjects, while solutions for others have been found. For example, through some mathematical tricks we can improve stability and at the same time simplify the cost function. [1].

In the paper, the authors give a visual representation of what is actually happening, at least theoretically:

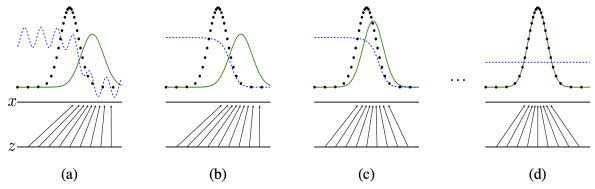

Figure 1. Generator distribution changing over time

In the above figure, we have the real data points with black dots, the distribution of the fake data points with the green curve, and the discriminator with the blue dotted line. We can see that in the beginning, point **(a)**, the distribution of the real and fake data points are pretty different. Also, the discriminator is making a bad job as discerning the origin of the data points. At **(b)**, the discriminator has been trained and can now make the difference between real and fake data

points. At this point, point **(c)**, the generator is modifying its parameters based on the new discriminator and shifts and scales its distribution to be more similar to the real data distribution. These steps are (theoretically) then repeated up until point **(d)**. Here, the distribution of the fake data points coincides with the distribution of the original data points, and the discriminator cannot make the difference between the two anymore. It is important to realize that the fake distribution never interacts in any way with the real distribution, it only interacts with the discriminator to improve the quality of its generated fake data points.

2.2. GANs using convolutions

In the original GAN paper [2], the layers used in the discriminator and generator were fully connected layers, even tough the task was of generating images. With the rise and popularisation of convolutional neural networks in that period, the natural and necessary for the next version of GANs to implement their layers using convolutional neural networks. This is what happend in [7] where Radford et. al used convolutional layers in the discriminator and transposed convolutional layers in the generator to create the GAN architecture.

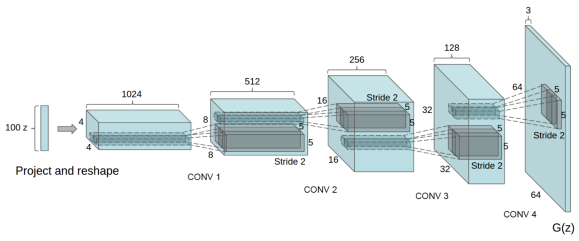

Figure 2. Convolutional GAN generator

We can see in Figure 2. how the random noise input is transformed by the generator into the desired image output shape using transposed convolutional layers only, no fully connected layers were used.

3. Dataset

We used the CelebA dataset [6], which is a large-scale face attributes dataset with more than 200K celebrity images, each with 40 attribute annotations (which we do not need to use). The images in this dataset cover large pose variations and background clutter. It comprises:

- 10,177 number of identities
- 202,599 number of face images

Since in the case of GANS we still do not have a standard and universally accepted method of evaluating the results of the model, we will simply check the results visually. Therefore, we do not need to split the dataset into train/valid/test sets, so we use all the available data for training.

Figure 3. CelebA dataset examples

4. Model

The input images are center cropped with resolution 160 x 160, resized to 64 x 64, and normalized in the range $[-1, 1]$. We used 20 training epochs and a batch size of 32.

The discriminator has 4 blocks of: convolutional layer followed by leaky ReLU. In each block the input doubles its number of feature maps (starting from 64) and halves its height and width (starting from the input height and width, 64). At the end, another convolutional layer is used to transform what the model has so far into just one number, corresponding to the probability of the input being drawn from the real data distribution. We can see how the discriminator has no dense layers and is therefore a fully convolutional network [8]. The learning rate used for the discriminator is $2 * 10^{-4}$.

The generator has 4 blocks of: transposed convolutional layer, followed by batch norm [3], followed by leaky ReLU. Each of these transposed convolutional layers halves the number of channels of the input (starting from 512) and doubles the height and width (starting from 4 x 4). At the end there is another transposed convolutional which has the job of mapping its input into the desired output shape (3 x 64 x 64), followed by a tanh activation function (since we normalized our pixel values in the dataset images in the range $[-1, 1]$). Similarly to the discriminator, we can see how the generator uses no dense layers and is therefore also a fully connected neural network. The learning rate used for the generator is $2 * 10^{-4}$. The loss function used was binary cross entropy. The optimizers for the generator and discriminator were Adam optimizers [4].

5. Results

Below we can see some examples of generated images the model created:

Figure 4. Examples of generated face images

While the generated images are not perfect, they are visually better than the ones generated in the original GAN paper [2], and comparable to the ones from the DCGAN paper [7]

References

[1] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017. 1

[2] J Goodfellow Ian, Pouget-Abadie Jean, Mirza Mehdi, Xu Bing, Warde-Farley David, Ozair Sherjil, and C Courville Aaron. Generative adversarial nets. In *Proceedings of the 27th international conference on neural information processing systems*, volume 2, pages 2672–2680, 2014. 1, 2, 3

[3] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 2

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012 alexnet. *Adv. Neural Inf. Process. Syst*, pages 1–9, 2012. 1

[6] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015. 2

[7] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks (2016). *arXiv preprint arXiv:1511.06434*, 2016. 2, 3

[8] Weiwei Sun and Ruisheng Wang. Fully convolutional networks for semantic segmentation of very high resolution remotely sensed images combined with dsm. *IEEE Geoscience and Remote Sensing Letters*, 15(3):474–478, 2018. 2