Generative Adversarial Networks (GANs)

Generative Adversarial Networks, known as GANs, are in their most basic form, two neural networks that teach each other how to solve a specific task. It was invented by Goodfellow and colleagues in 2014[[1]](#footnote-1). Intuitively, the two networks help each other with the final goal of being able to generate new data with the same statistics as the data used for training. For example, you may want to train a network to generate human faces that are as realistic as possible. In this case one network will generate human faces as good as it can, and the second network will criticize the results and tell the first network how to correct the faces. The two networks will learn from each other, so to speak. In this chapter we will look in detail how this works, and how to implement an easy example in Keras.

# Introduction to GANs

The best way for us to understand how GANs work, is to base our discussion on the diagram in Figure 11.1. After having understood what is going on under the hood, we will look at a more formal and theoretical discussion about GANs.

## Training Algorithm for GANs

To build a GANs system, we need two neural networks: a *Generator* and a *Discriminator.* The Generator has the goal of producing fake observation[[2]](#footnote-2) , while the Discriminator has the goal of classifying an input as fake or real. Imaging the following classical example: the Generator (let’s call him George) can be an art forger that is trying to produce paintings of some known painter, let’s say Van Gogh. And the Discriminator (let’s call her Anna) is an art critic that gives a judgement if the painting that George has produced looks real or not. They are new to this, so they decide to learn this together[[3]](#footnote-3). George produces a painting. Anna examines it and gives some suggestions to George. Every now and then, Anna also trains with some real Van Gogh paintings to get better in spotting errors from George. This process is repeated many times, until George is so good to fool Anna. At this point George can pain like Van Gogh, can produce many fakes, and get rich by selling his fake paintings[[4]](#footnote-4). This process that we just described, is depicted in Figure 1.11. Let’s see how our story translates in the language of neural networks.

Diagram

Description automatically generated

Figure 1-11: A diagram depicting all components and steps of a GAN setup.

The generator gets as input a noise vector taken from a normal distribution. The size of this vector is not fixed and can be chosen depending on the problem at hand. In the example we will discuss in this chapter we have chosen . The Generator (George) take the random vector and generate a fake observation (as you can see in Figure 1-11). The output will have the same dimension of the observations contained in the training dataset (in this example Van Gogh paintings). If for example if are 1000x1000 pixels images in color, then will also be a 1000x1000 color image. Now is the Discriminator (Anna) turn. It gets as input a (or ) and has a one-dimensional output (the probability of the input of being real or fake). Basically, the discriminator is performing binary classification.

The steps of the training loop are described below.

1. A vector of numbers is generated from a normal distribution.
2. Using this the Generator gives as output an .
3. The discriminator is used two times: one with a real input () and one with the generated in the previous step.
4. Two loss functions are calculated: and
5. Via an optimizer (Adam, Momentum, etc.), the two loss functions are minimized sequentially (sometime for one step for the Generator, multiple steps in updating the weights for the Discriminator are run). Note that minimizing will be done only with respect to the trainable parameters of the Generator, while minimizing will be done only with respect to the trainable parameters of the Discriminator. Sometime for one step for the Generator, multiple steps for the Discriminator are carried out.

## A practical example with Keras and MNIST

Let’s see now a practical example of what we discussed in the previous section implemented with Keras and applied to the MNIST dataset[[5]](#footnote-5). As usual you can find the complete code on <https://adl.toelt.ai> so we will concentrate here only on the relevant parts of the code. In particular we will look at the 5 steps described at the end of the previous section and how to implement them. To start we will first need to create two neural networks: the Generator and the Discriminator. This can be done in the usual way. For example

def make\_generator\_model():

model = tf.keras.Sequential()

model.add(layers.Dense(7\*7\*256, use\_bias=False, input\_shape=(100,)))

model.add(layers.BatchNormalization())

model.add(layers.LeakyReLU())

model.add(layers.Reshape((7, 7, 256)))

assert model.output\_shape == (None, 7, 7, 256) # Note: None is the batch size

model.add(layers.Conv2DTranspose(128, (5, 5), strides=(1, 1), padding='same', use\_bias=False))

assert model.output\_shape == (None, 7, 7, 128)

model.add(layers.BatchNormalization())

model.add(layers.LeakyReLU())

model.add(layers.Conv2DTranspose(64, (5, 5), strides=(2, 2), padding='same', use\_bias=False))

assert model.output\_shape == (None, 14, 14, 64)

model.add(layers.BatchNormalization())

model.add(layers.LeakyReLU())

model.add(layers.Conv2DTranspose(1, (5, 5), strides=(2, 2), padding='same', use\_bias=False, activation='tanh'))

assert model.output\_shape == (None, 28, 28, 1)

return model

The really important part of this network is the input shape: input\_shape=(100,). Remember that the Generator gets as input the random vector that is, in our example, a 100 dimensional vector of random numbers generated from a normal distribution. In Figure 11-2 you can see a better visualization of the network.

1. Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron; Bengio, Yoshua (2014). [Generative Adversarial Nets](https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf) (PDF). Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014). pp. 2672–2680. [↑](#footnote-ref-1)
2. We use here the generic term observation. They could be fake faces, in case you are trying to build a system that generates realistic faces, or an aged version of a face for example. We call an observation one of the inputs in the training dataset. [↑](#footnote-ref-2)
3. How realistic would be that a forger and a critic work together is not the main point of the story. [↑](#footnote-ref-3)
4. Of course, we are not encouraging anyone to become dishonest. Is just a story to let you understand GANs… [↑](#footnote-ref-4)
5. At this point in the book, you should know the MNIST dataset very well. In case you don’t remember, it is a dataset with 70000 handwritten digits, save das gray-level images 28x28 pixel in resolution. [↑](#footnote-ref-5)