Generative Adversarial Networks (GANs)

You might not think that programmers are artists, but programming is an extremely creative profession. It's logic-based creativity. - John Romero

Generative adversarial networks (GANs) are deep neural net architectures comprised of two nets, pitting one against the other (thus the "adversarial").

GANs were introduced in a paper by Ian Goodfellow and other researchers at the University of Montreal, including Yoshua Bengio, in 2014. Referring to GANs, Facebook's AI research director Yann LeCun called adversarial training "the most interesting idea in the last 10 years in ML."

GANs' potential is huge, because they can learn to mimic any distribution of data. That is, GANs can be taught to create worlds eerily similar to our own in any domain: images, music, speech, prose. They are robot artists in a sense, and their output is impressive – poignant even.

In a surreal turn, Christie's sold a portrait for $432,000 that had been generated by a GAN, based on open-source code written by Robbie Barrat of Stanford. Like most true artists, he didn't see any of the money, which instead went to the French company, Obvious.

Generative vs. Discriminative Algorithms

To understand GANs, you should know how generative algorithms work, and for that, contrasting them with discriminative algorithms is instructive. Discriminative algorithms try to classify input data; that is, given the features of a data instance, they predict a label or category to which that data belongs.

For example, given all the words in an email, a discriminative algorithm could predict whether the message is `spam` or `not_spam`. `spam` is one of the labels, and the bag of words gathered from the email are the features that constitute the input data. When this problem is expressed mathematically, the label is called `y`and the features are called `x`. The formulation `p(y|x)` is used to mean "the probability of y given x", which in this case would translate to "the probability that an email is spam given the words it contains."

So discriminative algorithms map features to labels. They are concerned solely with that correlation. One way to think about generative algorithms is that they do the opposite. Instead of predicting a label given certain features, they attempt to predict features given a certain label.

The question a generative algorithm tries to answer is: Assuming this email is spam, how likely are these features? While discriminative models care about the relation between `y` and `x`, generative models care about "how you get x." They allow you to capture `p(x|y)`, the probability of `x` given `y`, or the probability of features given a class. (That said, generative algorithms can also be used as classifiers. It just so happens that they can do more than categorize input data.)

Another way to think about it is to distinguish discriminative from generative like this:

- Discriminative models learn the boundary between classes
- Generative models model the distribution of individual classes

How GANs Work

One neural network, called the *generator*, generates new data instances, while the other, the *discriminator*, evaluates them for authenticity; i.e. the discriminator decides whether each instance of data it reviews belongs to the actual training dataset or not.

Let's say we're trying to do something more banal than mimic the Mona Lisa. We're going to generate hand-written numerals like those found in the MNIST dataset, which is taken from the real world. The goal of the discriminator, when shown an instance from the true MNIST dataset, is to recognize them as authentic.

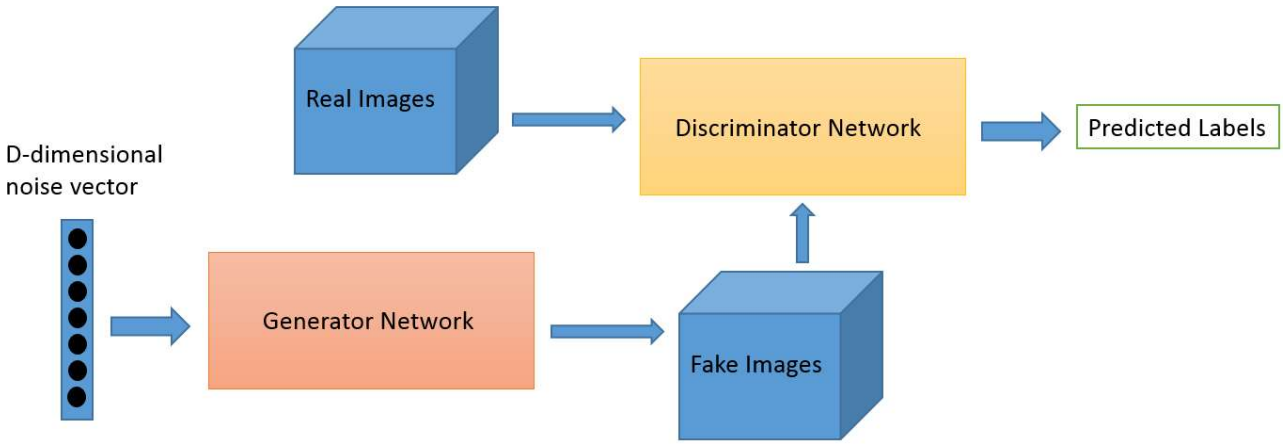
Meanwhile, the generator is creating new images that it passes to the discriminator. It does so in the hopes that they, too, will be deemed authentic, even though they are fake. The goal of the generator is to generate passable hand-written digits, to lie without being caught. The goal of the discriminator is to identify images coming from the generator as fake.

Here are the steps a GAN takes:

- The generator takes in random numbers and returns an image.
- This generated image is fed into the discriminator alongside a stream of images taken from the actual dataset.
- The discriminator takes in both real and fake images and returns probabilities, a number between 0 and 1, with 1 representing a prediction of authenticity and 0 representing fake.

So you have a double feedback loop:

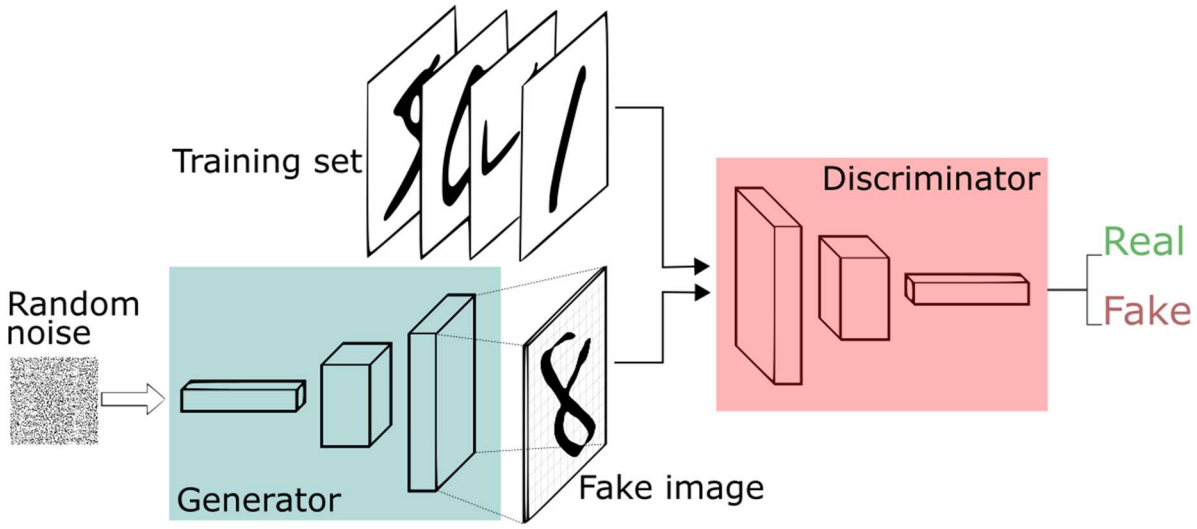
- The discriminator is in a feedback loop with the ground truth of the images, which we know.
- The generator is in a feedback loop with the discriminator.



You can think of a GAN as the combination of a counterfeiter and a cop in a game of cat and mouse, where the counterfeiter is learning to pass false notes, and the cop is learning to detect them. Both are dynamic; i.e. the cop is in training, too (maybe the central bank is flagging bills that slipped through), and each side comes to learn the other's methods in a constant escalation.

The discriminator network is a standard convolutional network that can categorize the images fed to it, a binomial classifier labeling images as real or fake. The generator is an inverse convolutional network, in a sense: While a standard convolutional classifier takes an image and downsamples it to produce a probability, the generator takes a vector of random noise and upsamples it to an image. The first throws away data through downsampling techniques like maxpooling, and the second generates new data.

Both nets are trying to optimize a different and opposing objective function, or loss function, in a zero-zum game. This is essentially an actor-critic model. As the discriminator changes its behavior, so does the generator, and vice versa. Their losses push against each other.

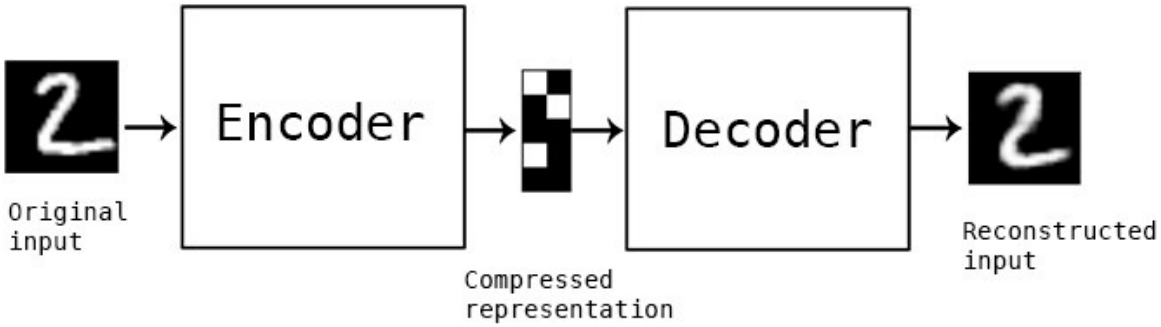


If you want to learn more about generating images, Brandon Amos wrote a great post about interpreting images as samples from a probability distribution.

GANs, Autoencoders and VAEs

It may be useful to compare generative adversarial networks to other neural networks, such as autoencoders and variational autoencoders.

Autoencoders *encode* input data as vectors. They create a hidden, or compressed, representation of the raw data. They are useful in dimensionality reduction; that is, the vector serving as a hidden representation compresses the raw data into a smaller number of salient dimensions. Autoencoders can be paired with a so-called decoder, which allows you to reconstruct input data based on its hidden representation, much as you would with a restricted Boltzmann machine.



Variational autoencoders are generative algorithm that add an additional constraint to encoding the input data, namely that the hidden representations are normalized. Variational autoencoders are capable of both compressing data like an autoencoder and synthesizing data like a GAN. However, while GANs generate data in fine, granular detail, images generated by VAEs tend to be more blurred. Deeplearning4j's examples include both autoencoders and variational autoencoders.