Chapter 20-21 – GAN | Part 1 & 2

Author: Gianmarco Scarano

gianmarcoscarano@gmail.com

1. Generative Adversarial Networks (GANs)

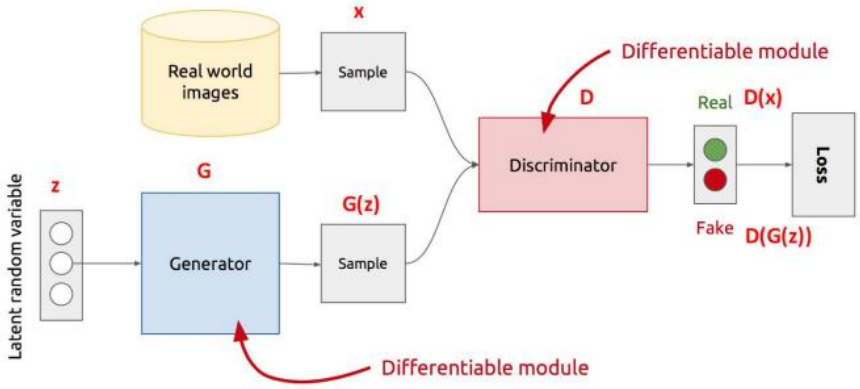
Introducing GANs, we deploy three different kinds of Generative Adversarial Networks:

- **Generative**: They learn a generative model starting from the probabilistic distribution of ***X***.
- **Adversarial**: Trained in an adversarial setting. They compete with each other through the use of generator and discriminator.
- **Networks**: Use Deep Neural Networks

In general, Generative Models resolve multiple issues such as computing the probability of **X** directly and not, instead, estimating the label ***Y*** for ***X***.

Also, we can't sample from ***P(X)*** (meaning that basically we can't generate new images). GANs can basically cope with everything explained above.

1.1 Architecture



Talking about the architecture, as we can see from the image on the left, we have ***D*** and ***G*** which are respectively the *Discriminator* and the *Generator*.

Both must be differentiable, meaning that they have to be optimizable.

After training ***D***, we backpropagate the opposite gradient (*-grad*) to ***G*** and update the ***G***'s weights in order for ***G*** to sample fake images that resemble the true ones.

To real data, we give the true values (Ground Truth), while to the generated data (generated by ***G***) we give the true label. In this way we tell ***G*** to make samples which look like a real image coming from the set of real-world images.

The Discriminator has to detect whether which one between x or $G(z)$ is the real image, where z is some random noise (Gaussian/Uniform) from which the Generator ***G*** samples from.

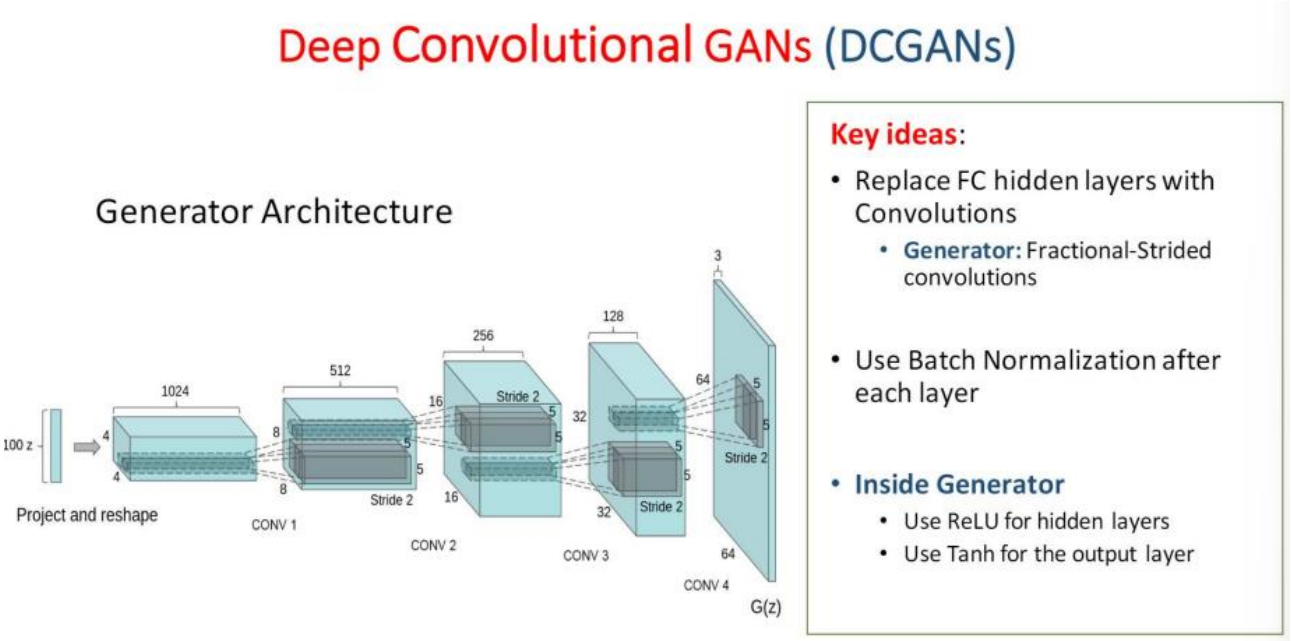
1.2 Training loop

The training loop can be seen as a minimax game, where the Discriminator is trying to maximize its reward $V(D,G)$, while the Generator is trying to minimize the Discriminator's rewards (or equivalently maximize its loss).

$$V(D,G) = \mathbb{E}_{x\sim p(x)}[\log D(x)] + \mathbb{E}_{z\sim q(z)}[\log(1 - D(G(z)))]$$

- The Nash equilibrium of this particular game is achieved at:
  - $P_{data}(x) = P_{gen}(x) \;\; \forall x$
  - $\mathrm{D}(x) = \frac{1}{2} \;\; \forall x$

When training, the Discriminator samples a minibatch of m noise samples and m examples from the real distribution and updates the discriminator by ascending its stochastic gradient. The Generator, instead, samples from the noise and updates the generator by descending its stochastic gradient.



2. Advantages & issues with GANs

As we said, the main advantages for GANs are to estimate ***P(X),*** being robust to overfitting issues (since ***G*** never sees the training data) and also multiple works on GANs already exists, such as Variational AutoEncoders (VAE).

As for the problems, we can't straightforward directly estimate ***P(X)*** since the Probability Distribution is implicit and training is hard (non-convergence, collapsing networks etc → Forced, then, to use *Adam* as optimizer). As for collapsing, the scenario happens when the GAN starts ignoring the code or the noise variables (z), limiting the diversity of images generated.

Some solutions for the non-convergence and collapsing issues are:

- Training with mini-batch GANs:
  - Very simply, we let the ***Discriminator*** look at the entire batch instead of single examples. If there is a lack of diversity, it will mark the examples as fake, forcing ***G*** to produce different samples.
- Apply a supervision with labels:
  - Basically, we apply label information to the real data, which might involve a generation of better samples.

3. Applications & extensions

Conditional GANs are a simple modification of the original GAN frameworks, where we give both G and D a Class (label) in order to have additional information for better multi-modal learning. This results in Supervision learning if we feed also the class label.

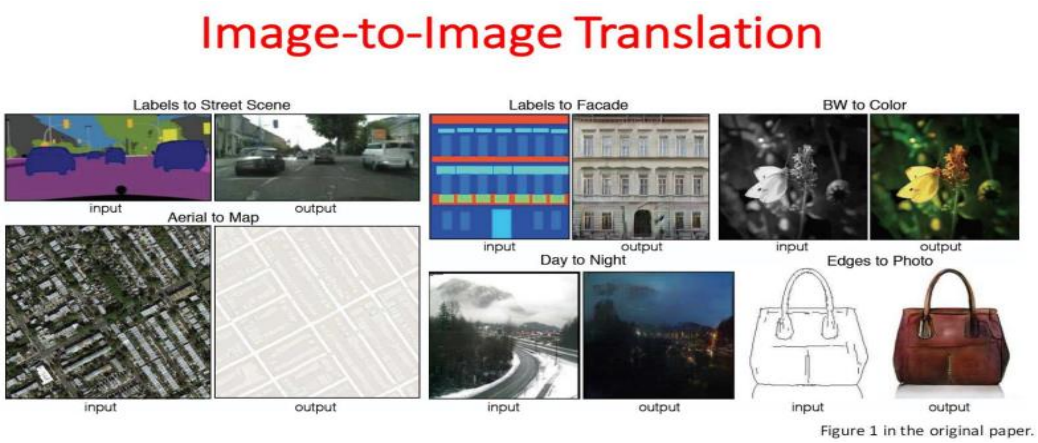


Figure 1 in the original paper.

These, on the left, are some GANs applications. In the architecture of this application, the Discriminator also looks at how different is the generated image w.r.t to the Ground Truth. Training, then, is conditioned on the images from the source domain. This is needed in order for the engineer to not code and design specific loss functions for that specific domain.

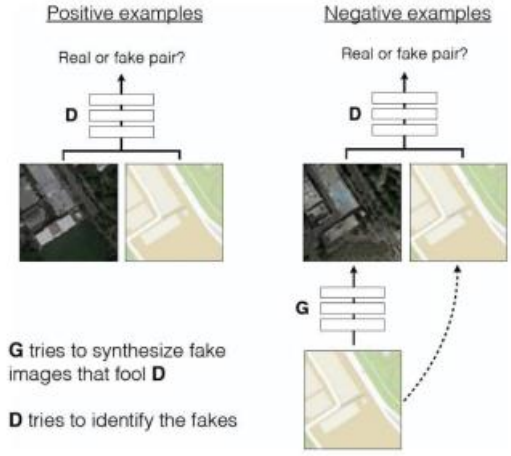


Figure 2 in the original paper.

Down here, we see a text-to-image translation. The architecture follows the one of a Conditional GAN, but the Generator and the Discriminator are conditioned on "Dense text embeddings" instead of noise.