Better GAN Architectures: WGAN and PGAN

What we will be covering in this module?

- Introduction to Image Generation
- What are Generative Models?
- Understanding Generative Adversarial Networks
- Project on Texture Generation using GANs
  - Simple Implementation
  - Better GAN Architectures
    - Wasserstein GAN
    - Progressive growing of GANs
- What's Next?

Better Architectures : Wasserstein GAN

Wasserstein GAN

Martin Arjovsky[1], Soumith Chintala[2], and Léon Bottou[1,2]

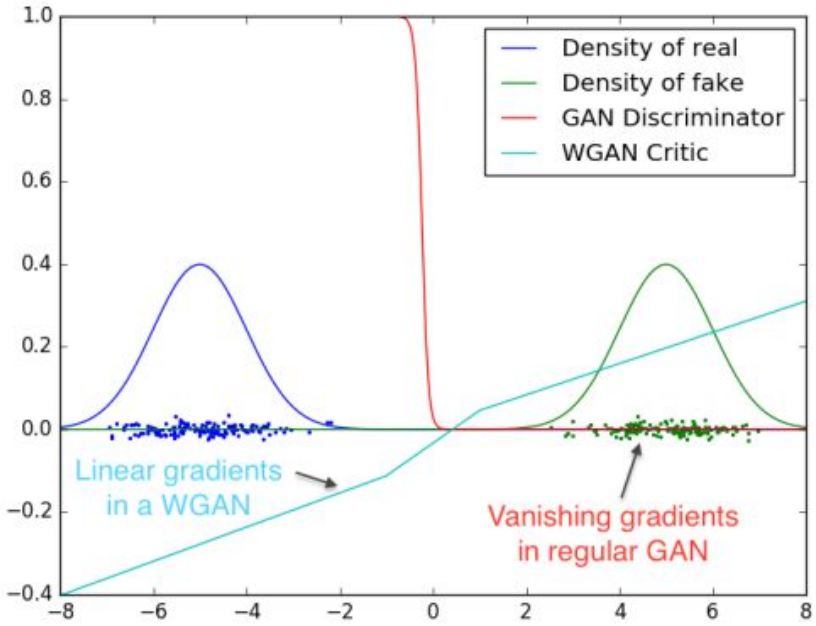
[1]Courant Institute of Mathematical Sciences
[2]Facebook AI Research

1 Introduction

The problem this paper is concerned with is that of unsupervised learning. Mainly, what does it mean to learn a probability distribution? The classical answer to this is to learn a probability density. This is often done by defining a parametric family of densities $(P_\theta)_{\theta \in \mathbb{R}^d}$ and finding the one that maximized the likelihood on our data: if we have real data examples $\{x^{(i)}\}_{i=1}^m$, we would solve the problem

$$\max_{\theta \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^{m} \log P_\theta(x^{(i)})$$

Better Architectures : Wasserstein GAN



Analytics Vidhya

Better Architectures : Wasserstein GAN

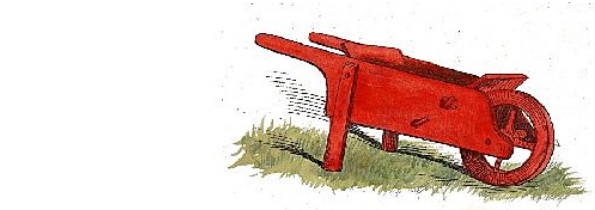
The *Earth-Mover* (EM) distance or Wasserstein-1

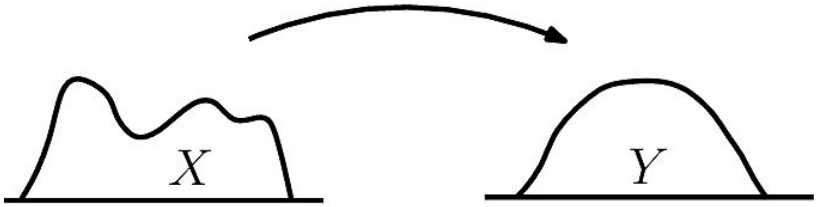
$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\big[\, \|x - y\| \,\big]$$

Better Architectures : Wasserstein GAN

The *Earth-Mover* (EM) distance or Wasserstein-1

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\big[\, \|x - y\| \,\big]$$





Analytics Vidhya

Better Architectures : Progressive growing of GANs

PROGRESSIVE GROWING OF GANS FOR IMPROVED QUALITY, STABILITY, AND VARIATION

Tero Karras NVIDIA
Timo Aila NVIDIA
Samuli Laine NVIDIA
Jaakko Lehtinen NVIDIA and Aalto University
{tkarras,taila,slaine,jlehtinen}@nvidia.com

ABSTRACT

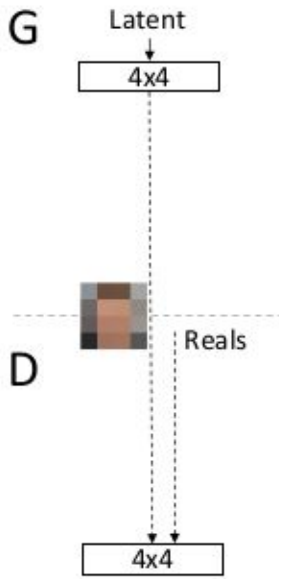
We describe a new training methodology for generative adversarial networks. The key idea is to grow both the generator and discriminator progressively: starting from a low resolution, we add new layers that model increasingly fine details as training progresses. This both speeds the training up and greatly stabilizes it, allowing us to produce images of unprecedented quality, e.g., CELEBA images at 1024^2. We also propose a simple way to increase the variation in generated images, and achieve a record inception score of 8.80 in unsupervised CIFAR10. Additionally, we describe several implementation details that are important for discouraging unhealthy competition between the generator and discriminator. Finally, we suggest a new metric for evaluating GAN results, both in terms of image quality and variation. As an additional contribution, we construct a higher-quality version of the CELEBA dataset.

Analytics Vidhya

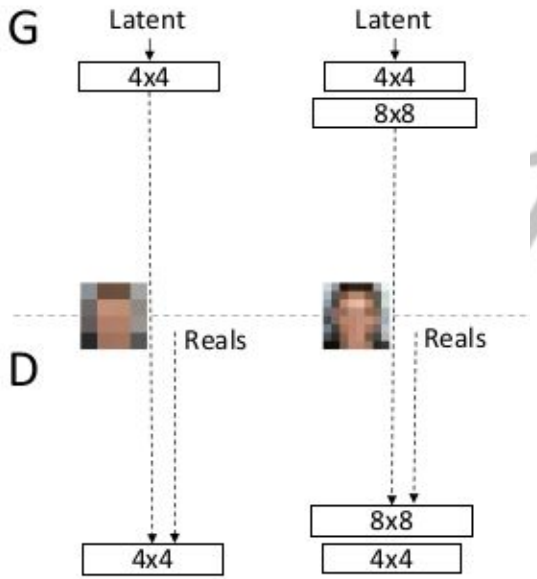
Better Architectures : Progressive growing of GANs

- Starts with low-resolution images, and then progressively increases the resolution by adding layers to the networks

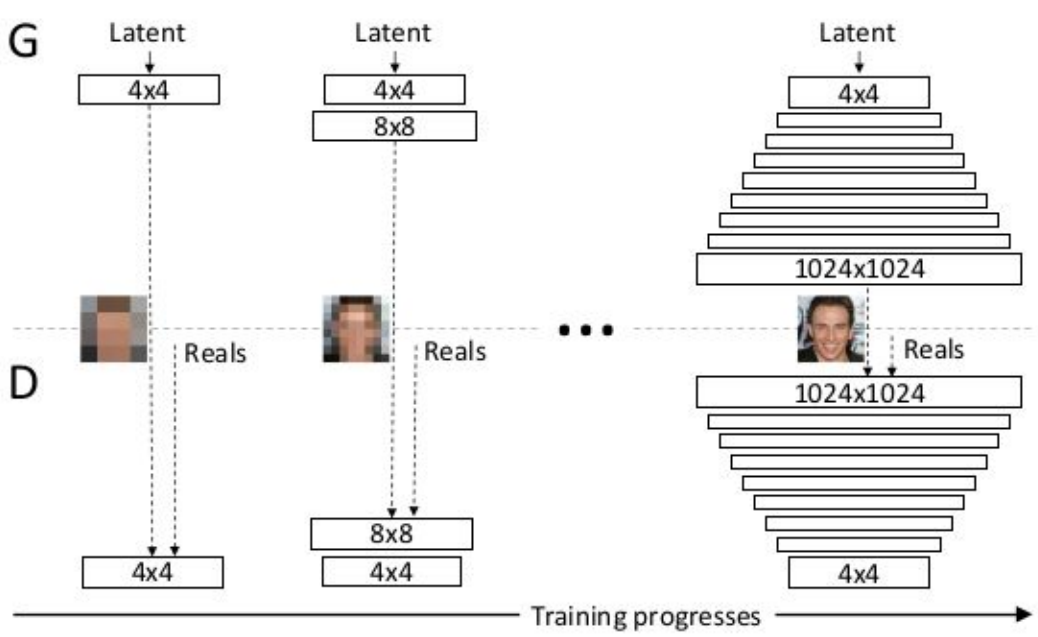
Better Architectures : Progressive growing of GANs



Better Architectures : Progressive growing of GANs



Better Architectures : Progressive growing of GANs



Analytics Vidhya

Better Architectures : Progressive growing of GANs

- Starts with low-resolution images, and then progressively increases the resolution by adding layers to the networks

- Primarily uses Wasserstein loss and DC-GAN architecture

Steps for solving Texture Generation using PGANs

1. Data Loading and Preprocessing

1.1 Load the Data

1.2 Define custom Dataset and Dataloader

1.3 Data Exploration

2. Image Generation using PGANs

2.1 Define model architecture

2.2 Train the model

2.3 Generate random images

Thank you

Analytics
Vidhya