2020

# Sparse GAN

SPARSE AND REDUNDANT REPRESANTATIONS – FINAL PROJECT

ROY GANZ | TSACHI BLAU

## Table of Contents

## Introduction

In the recent years, Generative Adversarial Networks (GANs) achieved State of The Art results in various image generation problems. A Vanilla GAN architecture consists of two main parts: A Generator that synthesizes various images from an input random vector, and a Discriminator that examines the quality of the generated images.

In GANs, the input, a random vector, is mapped into a “fake” image, by the generator. The discriminator receives fake and real images and returns as the GAN output whether the Images are real or fake.

The Generator and the Discriminator are trained in a min-max optimization framework, in which the generator model is being updated in order to create more realistic images, and the discriminator model is being updated in order to detect whether its input images are real or fake.

Due to the complex structure of images, generated images often lack fine details and sometimes don’t look completely realistic. Furthermore, since estimating images distribution is a very difficult task, GAN training tends to be very unstable, and phenomenon like *mode collapse* are very common.

We hence propose in this project an architecture for Generative Adversarial Networks (GANs), based on Sparse-Land. Our architecture consists of two essential units: Generator and discriminator, and optional unit that deals with *mode collapse*. In this approach, the generator task is to generate a sparse representation vector, instead of an image, which we claim to be a relaxed problem. The sparse vector will be multiplied by a pre-trained dictionary to create an image.

## Architecture Overview

Our architecture consists of three Deep Neural Network units: Generator, Discriminator and an Encoder. The generator and the discriminator are essential, and the encoder goal is to prevent the *mode collapse* phenomenon.

### Generator

The generator function maps random vectors into sparse representation vectors.

$$G : \bar{Z} \rightarrow \bar{A}$$

$$\bar{Z} = \{\bar{z} | \bar{z} \sim N(0, I)\}, \bar{A} = \{\bar{a} | \bar{a} \text{ } is \text{ } sparse \text{ } vector\}$$

The above function is modeled by a Fully Connected Deep Neural Network, consists of four layers, with batch normalization and non-linear activation. As the last layer, we added a soft-thresholding layer, that enables the output vector to truly sparse. The process of generating sparse model can be formulated as below:

$$\bar{a} = T_{\lambda}(G(\bar{z}))$$

$\lambda$ is a hyper parameter that defines the soft thresholding function, and therefore controls the sparsity level.

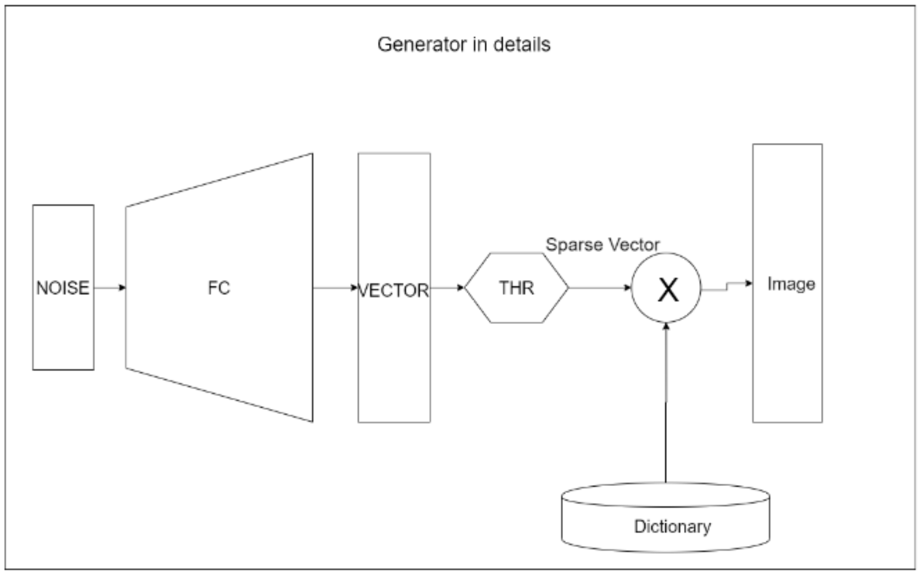


Figure 1: **generator architecture.** The generator's input is a random noise, and its output is a sparse vector that can be transformed into an image by multiplication with a pre-trained dictionary.

## Discriminator

The discriminator function maps images (real or fake) into labels.

$$D : \bar{I} \rightarrow O$$

$$\bar{I} = \{\bar{\iota} \mid \bar{\iota} \in fake\ images \cup\ real\ iamges\}, O = \{o \mid o \in [0,1]\}$$

The discriminator model will be trained to label real images as 1 and fake images as 0. The function above will be modeled by a Deep Convolutional Neural Network, consists of three convolutional layers and one fully connected layer. The last layer activation is *sigmoid* in order to map the output to range [0,1].

## Encoder

The encoder is a unit that should prevent cases of *mode collapse*. In these cases, the generator maps different inputs into the same output. The result of this phenomenon is that the synthesized image for different outputs is the same.

Adding an encoder unit and treating the generator as its decoder, constraints the generator to map different noise vectors to different outputs, that match the encoder input. Otherwise, the generator will be penalized for reconstruction error.

The encoder model is similar to an Encoder in Variational Auto-Encoder model – maps input images into output random vector, from a known distribution.

$$E : \bar{I} \rightarrow \bar{Z}$$

$$\bar{I} = \{real\ images\}, \bar{Z} = \{\bar{z} | \bar{z} \sim N(0, I)\}$$

## Dictionary

The dictionary has a crucial part in the success of our suggested model. A superposition of small number of atoms that the dictionary consists of, should be able to create real-world images. We have trained a dictionary using '*sklearn.decomposition.DictionaryLearning'* with sparsity level of 0.1 (support size is 0.1 of the sparse vector length).

In order to obtain the dictionary, the following problem is being solved:

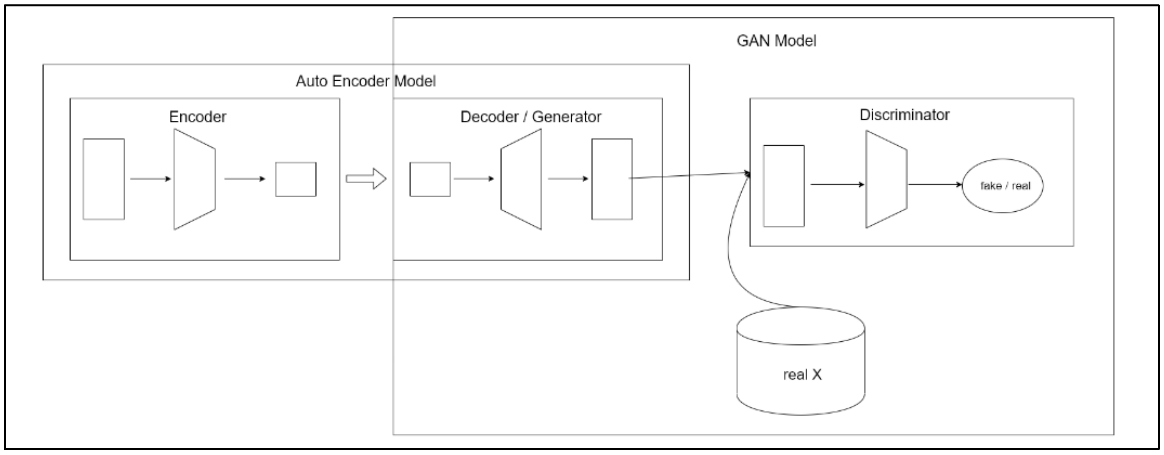
$$(D, A) = argmin_{D,A}\left||Y - DA|\right|_2^2 + \alpha\left||A|\right|_1 \ s.t \left||D_i|\right|_2 = 1, \forall i$$

Dictionary's atoms on Fashion MNIST

Dictionary's atoms on MNIST

### Putting it all together

In the sections above we explained about the different building blocks of our architecture. The Encoder and the Decoder create together a Variational Auto-Encoder model, and the Generator and the Discriminator create together a GAN. The Generator is contained both in the Variational Auto-Encoder model and in the GAN model.



## Optimization and Loss functions

The GAN model is trained in a min-max optimization framework:

$$\min_{G} \max_{D} \ [\log (D(x)) + \log (1 - D\big(G(z)\big))]$$

The Variational Auto-Encoder optimization goal is to reconstruct the input image correctly, while keeping the latent space normal distribution.

### Discriminator loss

For an input image, the discriminator outputs a number in range [0,1]. The discriminator should assign a real-word image a value of 1, and a generated image a value of 0. This problem can be treated as binary classification and hence, we use binary cross entropy loss.

$$BCE(x) = \ -ylog(\hat{y}) - (1 - y)\log (1 - \hat{y})$$

### Generator loss

We would like the generator to create realistic images, and to obey to the sparsity model. Namely, the generator should generate sparse vectors, that after multiplication with a pre-trained dictionary, will create realistic looking images.

In order to generate realistic images, we will use binary cross entropy – we would like our generator to synthesize images that the discriminator will label them as real.

In order to generate sparse vectors, we will use $L_1$ regularization over the output vector of the generator, which causes sparse solutions.

$$generator\ loss = BCE + \beta * L_1(\bar{a})$$

$\beta$ is a hyper parameter that controls the sparsity level of the generator's output vectors.

## Variational Auto-Encoder loss

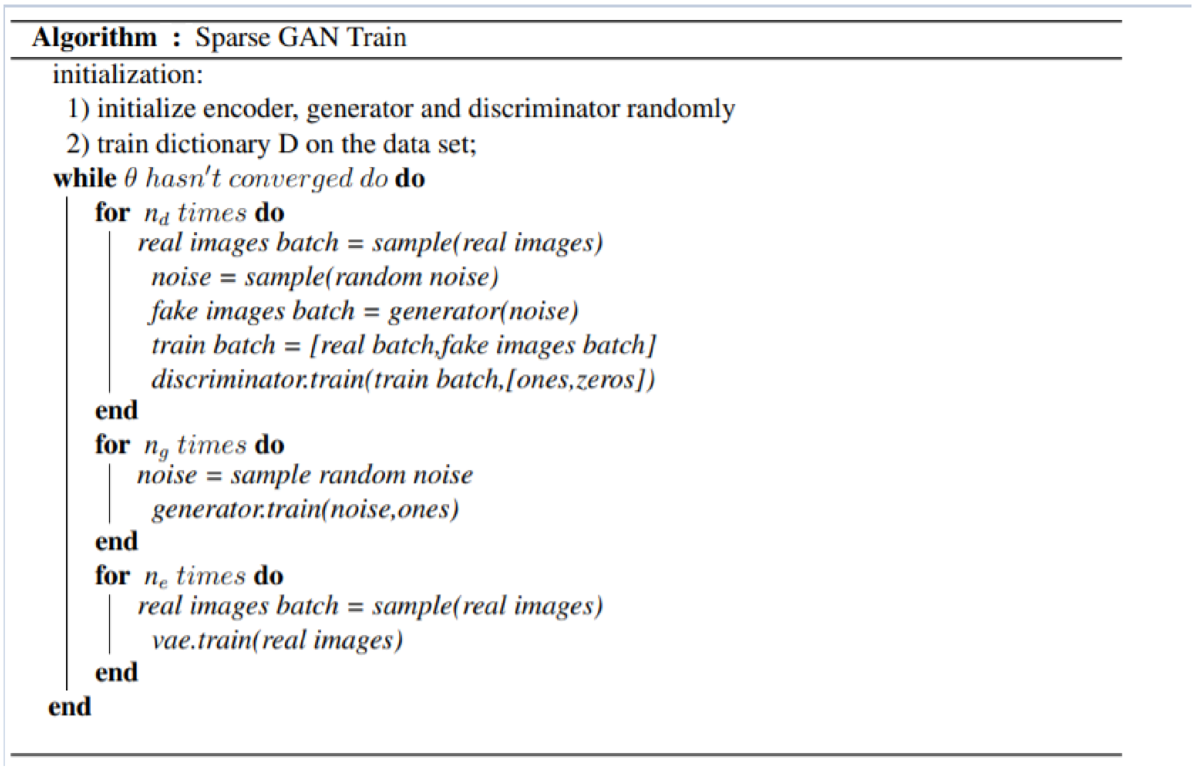
The auto-encoder has two main goals:

- Reconstruct the input image, as accurate as possible. To achieve this goal, we will use a $L_2$ loss between input image and reconstructed image.

- Latent vector's distribution gaussian - $\bar{Z} \sim N(\bar{0}, I)$. To achieve this goal, we will use The Kullback–Leibler divergence, that measures the distance between two probability distributions. In our case, between our latent space distribution Q, and our desired distribution P.

$$VAE\ loss = L_2(X, \hat{X}) + D_{KL}(P||Q)$$

## Training Scheme



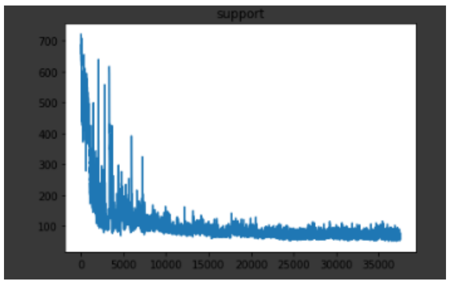
```
Algorithm : Sparse GAN Train
initialization:
  1) initialize encoder, generator and discriminator randomly
  2) train dictionary D on the data set;
while $\theta$ hasn't converged do do
    for $n_d$ times do
        real images batch = sample(real images)
        noise = sample(random noise)
        fake images batch = generator(noise)
        train batch = [real batch,fake images batch]
        discriminator.train(train batch,[ones,zeros])
    end
    for $n_g$ times do
        noise = sample random noise
        generator.train(noise,ones)
    end
    for $n_e$ times do
        real images batch = sample(real images)
        vae.train(real images)
    end
end
```

The above training scheme consists the following parts:

- Initialization: before training the neural network models, we initialize them with random weights, and train our dictionary on the data set
- Training loop consists of the following parts:
    - Training the discriminator to classify real images as real (1), and fake images as fake (0)
    - Training the generator to “fool” the discriminator to classify the generated images as real (1)
    - Training the Variational Auto-Encoder to reconstruct the input images with minimum reconstruction error.

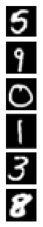
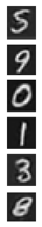
## Results

During the training process, the generated vector support decreases rapidly towards the true sparsity level, as our dictionary dictates. Furthermore, the generator is producing realistic images, in a relatively short amount of training.

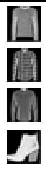
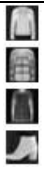


Support size during training procedure

### MNIST

| Real | Generated |
|---|---|
|  |  |

### Fashion MNIST

| Real | Generated |
|---|---|
|  |  |