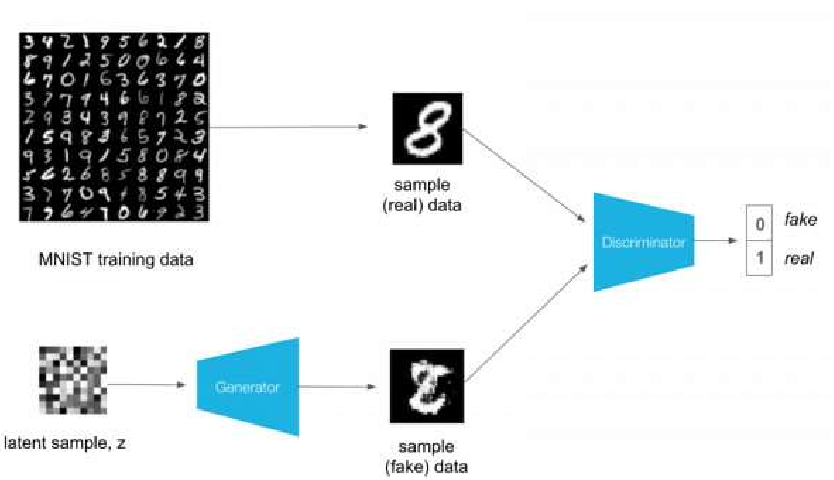
Neural networks

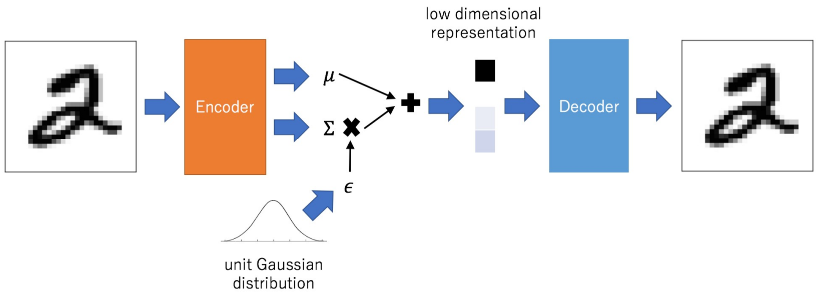
Generative models

Introduction

Create computational models to generate artificial data in a self-supervised manner.



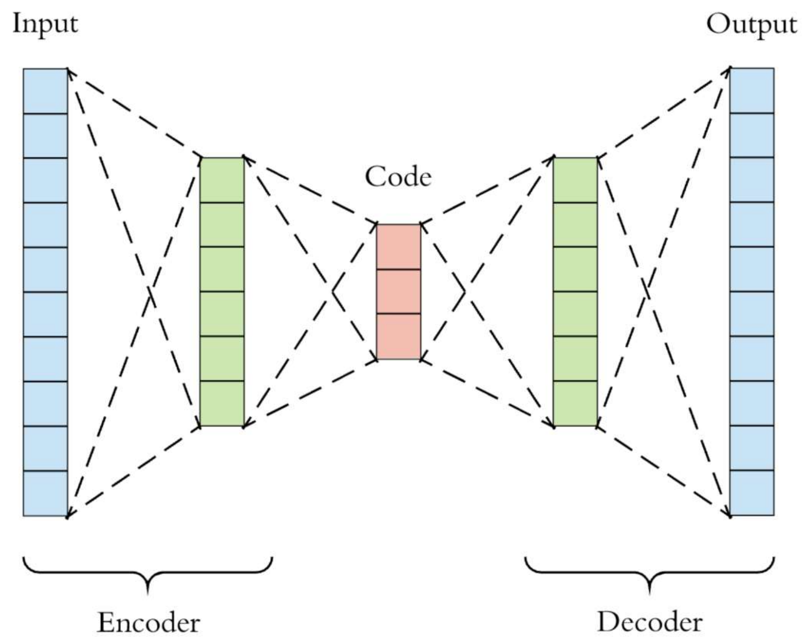
Generative adversarial networks



Variational autoencoders

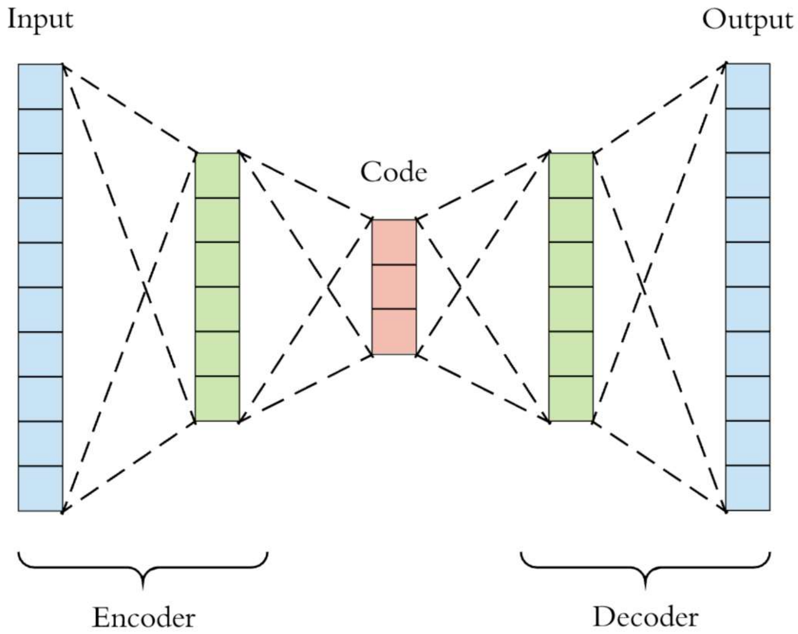
Auto-encoder

- Encoder: transforms the input into a feature vector
- Decoder: transforms the feature vector into the output
- The input and the output are the same



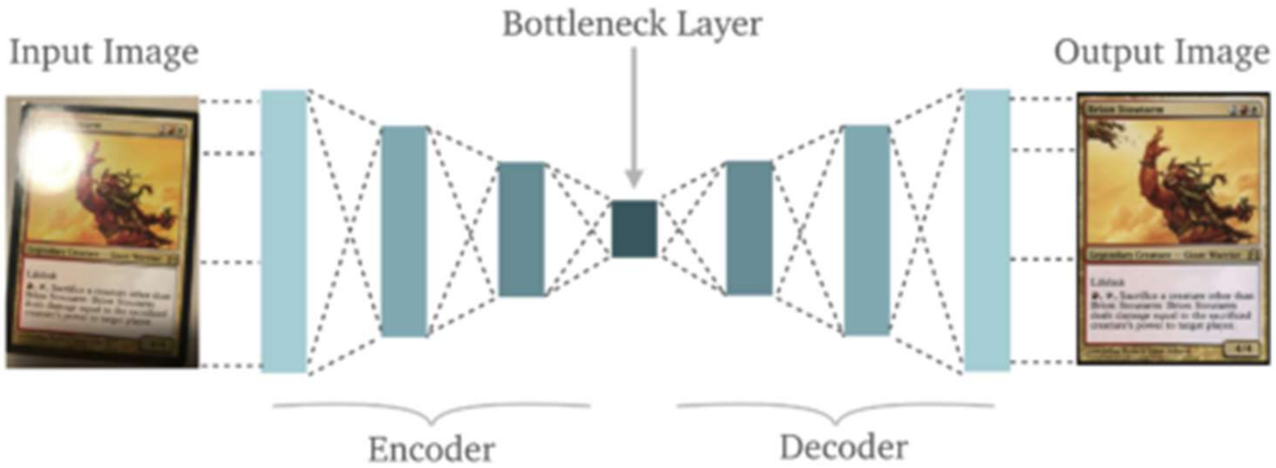
Auto-encoder

What is this architecture used for?



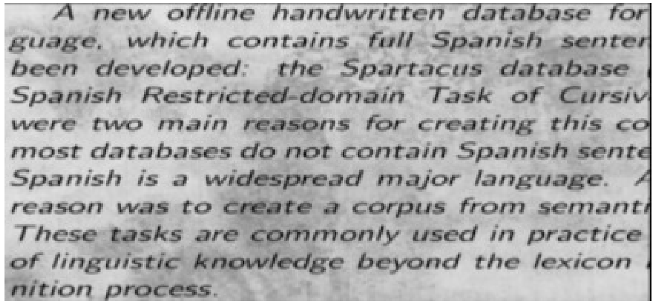
Auto-encoder applications

Applications
- Data compression
- Information correction

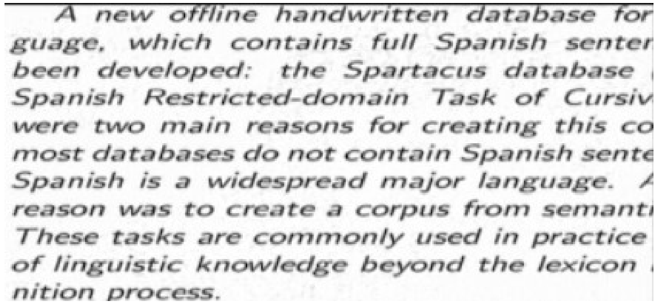


Auto-encoder applications

Image denoising



A new offline handwritten database for
guage, which contains full Spanish senter
been developed: the Spartacus database
Spanish Restricted-domain Task of Cursiv
were two main reasons for creating this co
most databases do not contain Spanish sente
Spanish is a widespread major language. A
reason was to create a corpus from semanti
These tasks are commonly used in practice
of linguistic knowledge beyond the lexicon
nition process.



A new offline handwritten database for
guage, which contains full Spanish senter
been developed: the Spartacus database
Spanish Restricted-domain Task of Cursiv
were two main reasons for creating this co
most databases do not contain Spanish sente
Spanish is a widespread major language. A
reason was to create a corpus from semanti
These tasks are commonly used in practice
of linguistic knowledge beyond the lexicon
nition process.

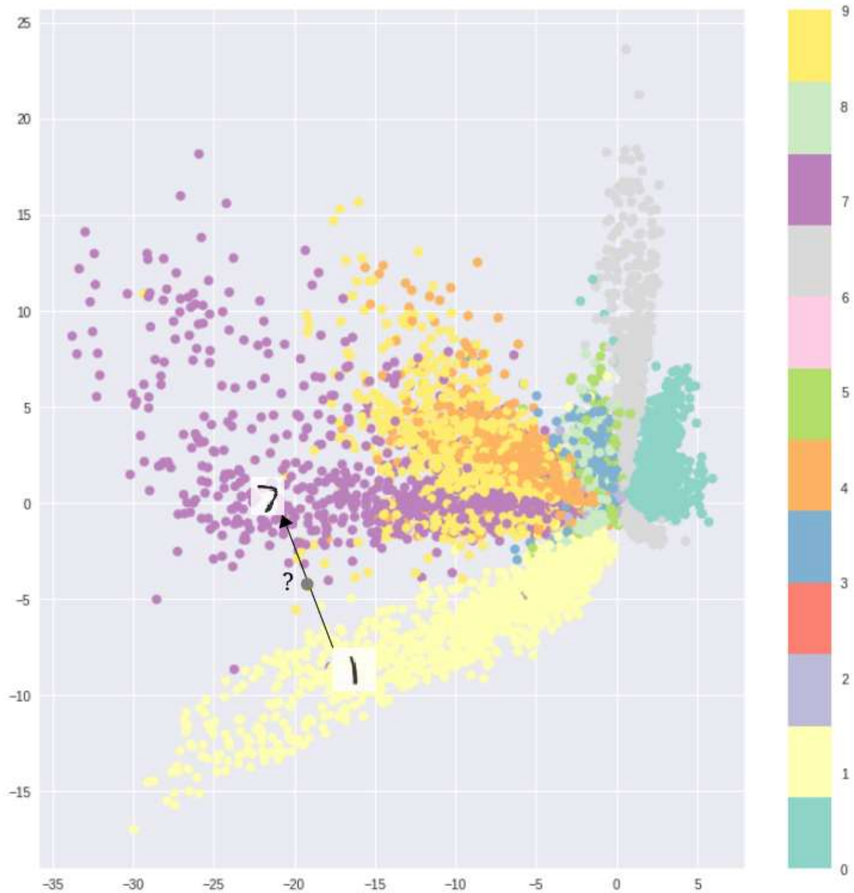
Autoencoders - Problems

Distribution of feature vectors is not continuous.

Feature vectors are learnt by minimizing the reconstruction error.

Difficult to find a correspondence between the feature space and the data.

Difficult to understand the data distribution because the distribution is not regularized.

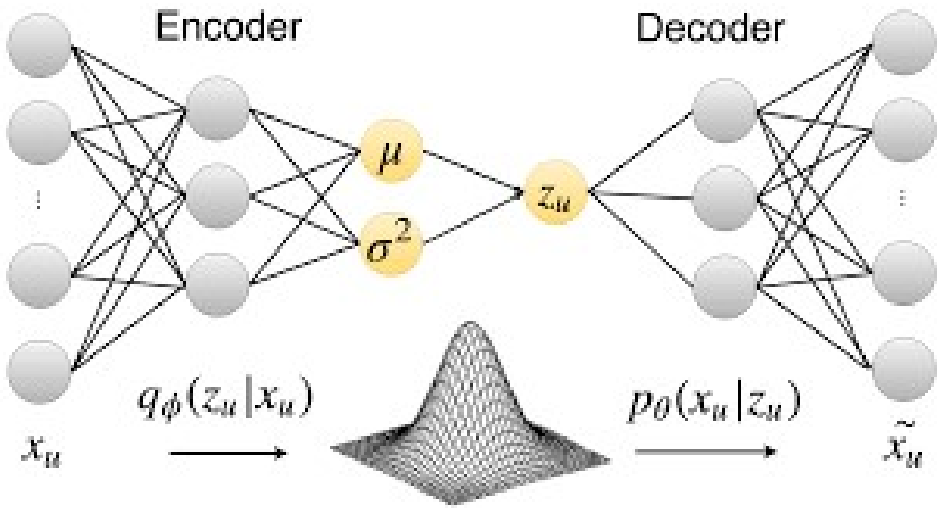


Variational Autoencoders

Regularize the distribution of feature space: we need a continuous distribution

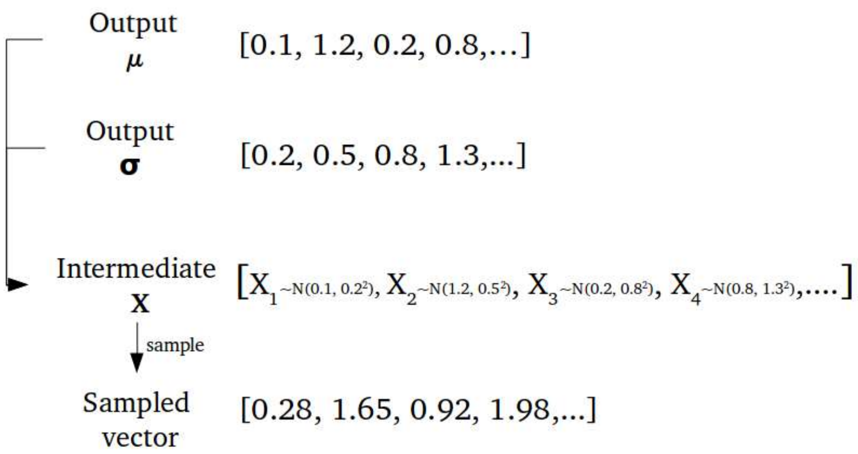
We can use a known distribution.

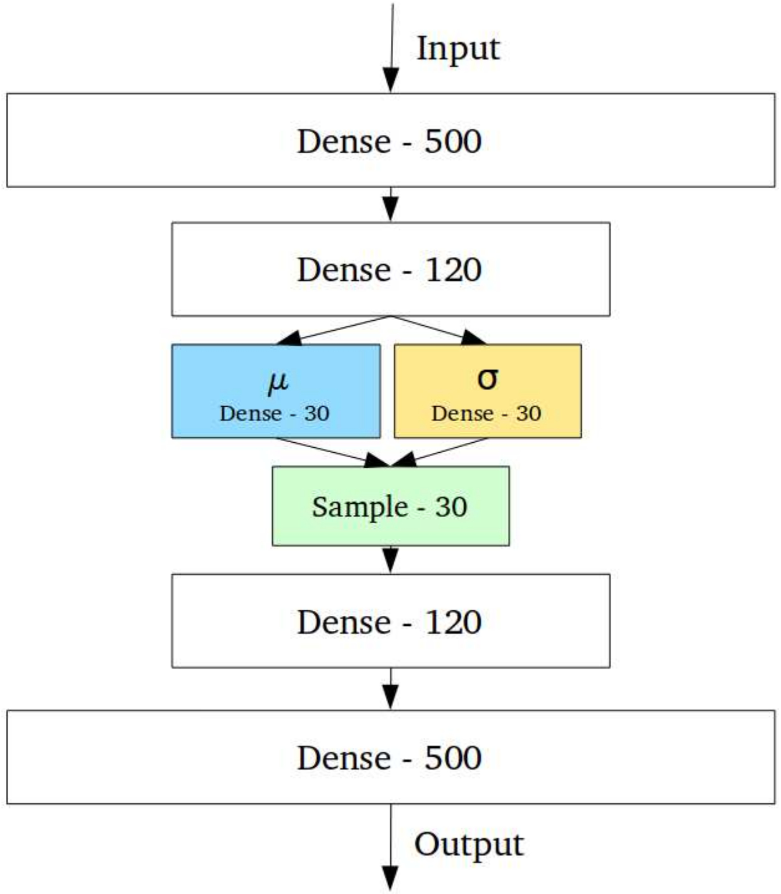
Encoder produces two vectors: mean and standard deviation of the feature space



Variational Autoencoders

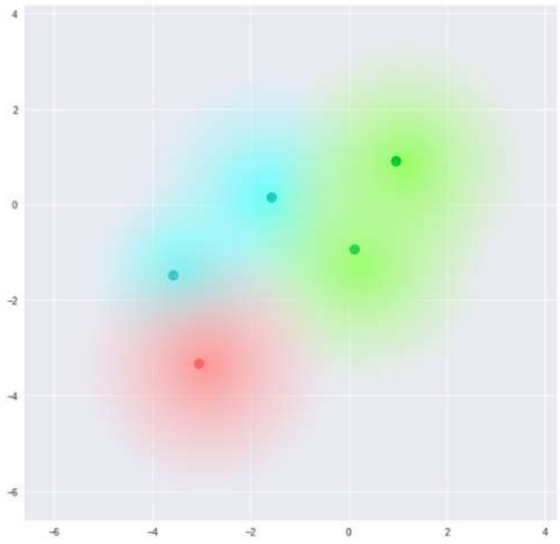
Each element of μ y σ are the mean and standard deviation of a random variable, from which we sample a feature vector



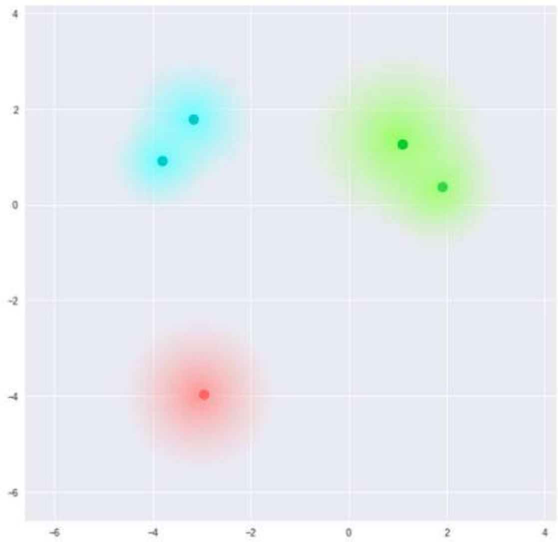


Variational Autoencoders

Create a continuous embedding space



What we require



What we may inadvertently end up with

Variational Autoencoders

Add a regularizer over the distribution. Kullback-Leibler divergence

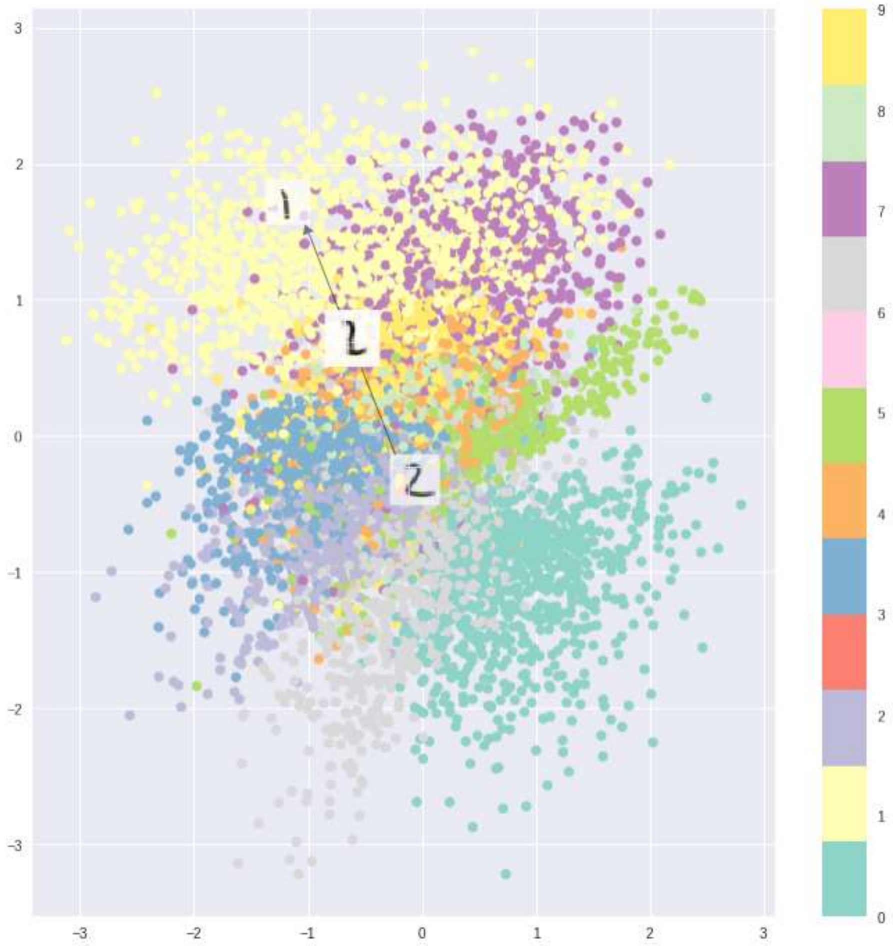
$$D_{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

$$D_{KL}(P||Q) = \int P(x) \log \frac{P(x)}{Q(x)} dx$$

If we use two Gaussian distributions, we get

$$\sum_{i=1}^{n} \sigma_i^2 + \mu_i^2 - \log(\sigma_i) - 1$$

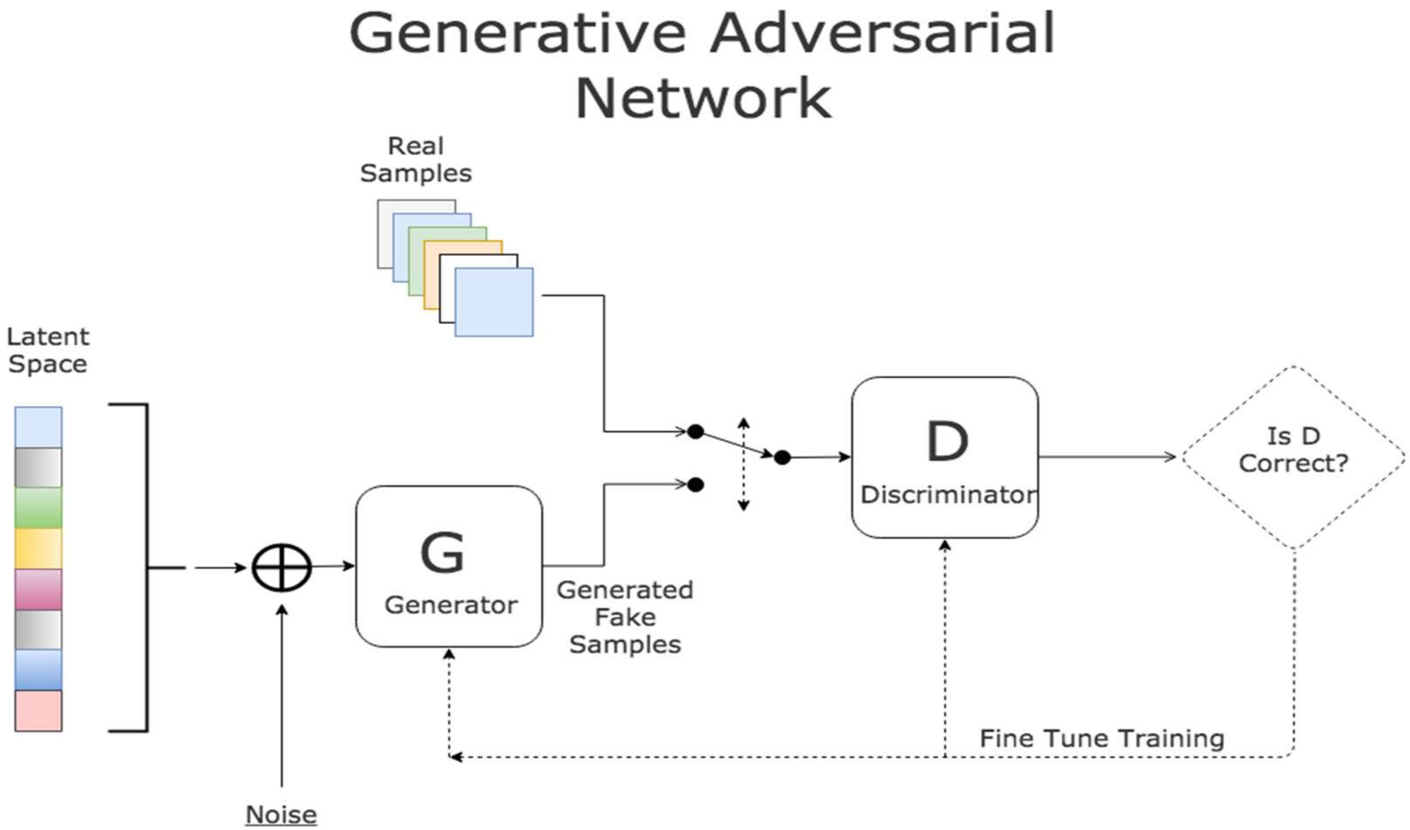
Variational Autoencoders



Neural networks

Generative adversarial networks

GAN



Generative Adversarial Network
Real Samples
Latent Space
D
Discriminator
Is D Correct?
G
Generator
Generated Fake Samples
Fine Tune Training
Noise

GAN ingredients

- To learn the data distribution, we define a latent variable z, and a function to compute the data space $\mathrm{G}(z, \theta_g)$.
- To learn the difference between real data and fake data, another function makes the classification $D(x, \theta_d)$.
- Tasks
    - Maximize of assigning a correct class
    - Minimize the capacity of the classifier to label generated data.

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.

  • Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.

  • Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.

  • Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.