Deep Generative Models

Aishwarya Venkataramanan

What is Generative Modelling?

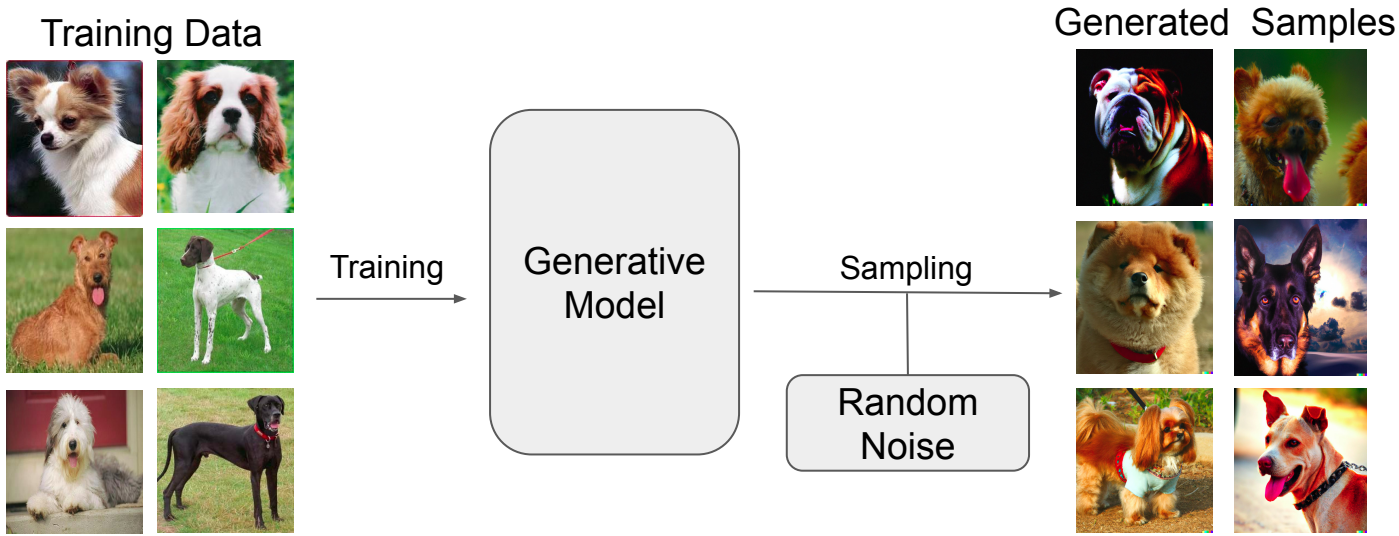
Goal: Create a model that can learn and approximate the underlying probability distribution of a given dataset.

Two main forms of generative models:

- Density Estimation – estimating the underlying probability distribution of a dataset

- Sample Generation – learn the underlying probability distribution of the training data and use that knowledge to produce new samples

Sample Generation Framework



Generative vs Discriminative Modelling

Generative Models

Goal: Model the underlying probability distribution of the input data; generate new data.

Learning Approach: Capture the joint probability distribution P(X, Y) or P(X). X-data, Y-labels

Examples: GDA, VAEs, Naive Bayes

Discriminative Models

Goal: Learn the mapping from input features to the output labels, without explicitly modelling the underlying data distribution.

Learning Approach: Directly model the conditional probability distribution P(Y|X). X-data, Y-labels

Examples: SVM, Logistic Regression, Random Forest

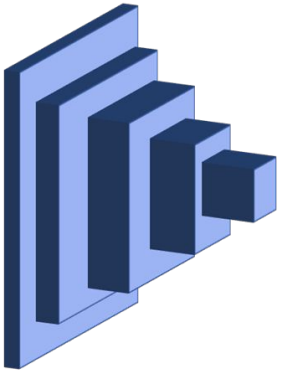
Latent Space

Learn a low dimensional representation of high dimensional data, such as images.

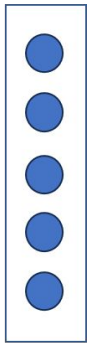
Latent Space

Why? 🧐

High Dimensional Input

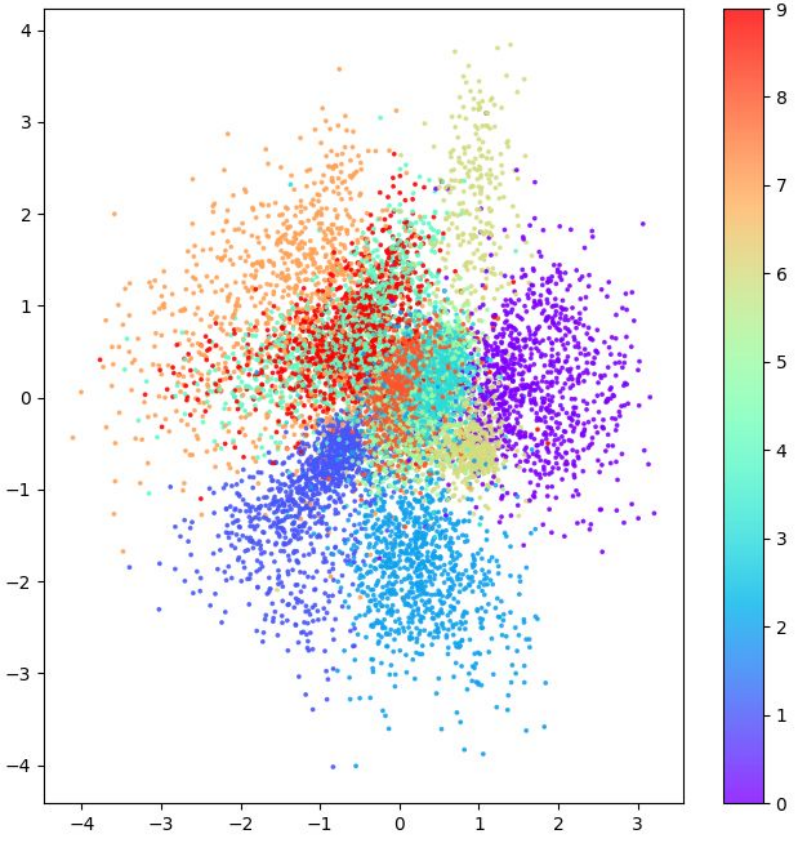


Encoder (Feature Extractor)



Reduced Dimensional Representations (Latent Space)

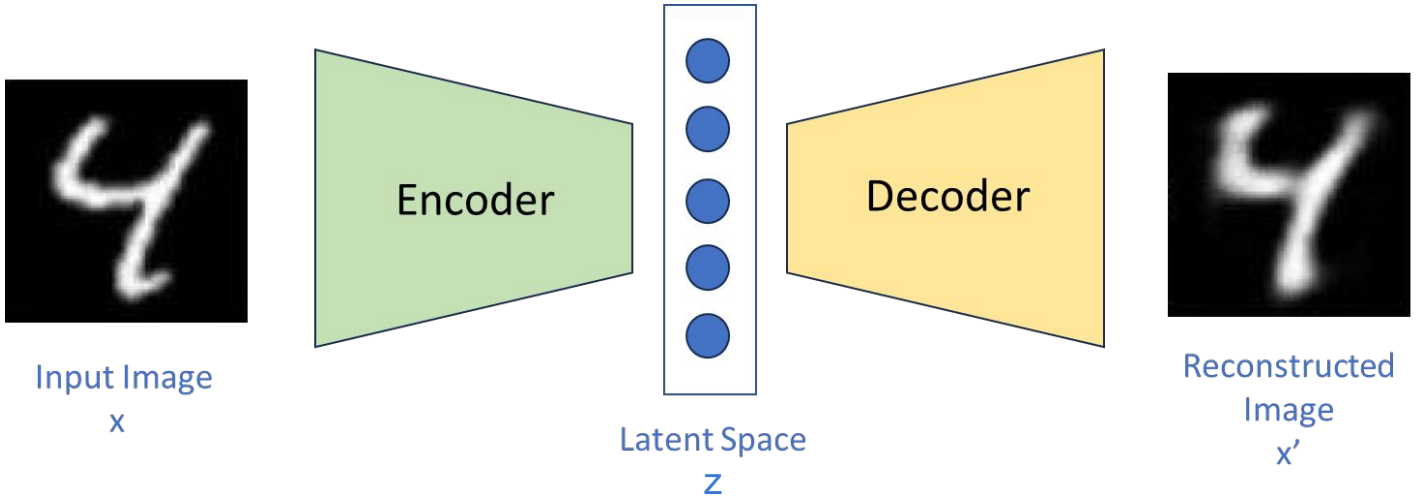
Latent Space of MNIST dataset



A 2D visualisation of latent space. Each image appears as a point in the latent space.

Autoencoders

Learn to encode data into a latent space and decode it back to the original data space.



$$\text{Loss} = ||x - x'||^2$$

Encoder - Compresses the input data into a low dimensional representation (latent space)

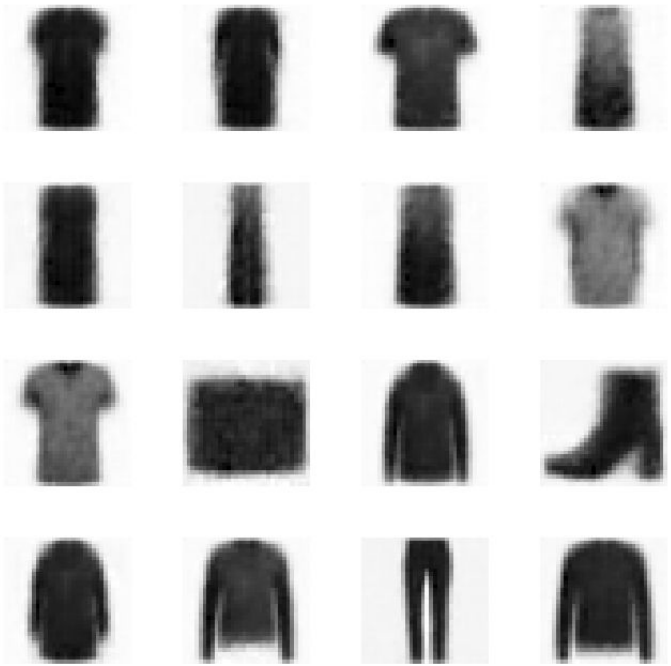
Decoder - Reconstructs the input data from latent space.

Some applications of autoencoders:

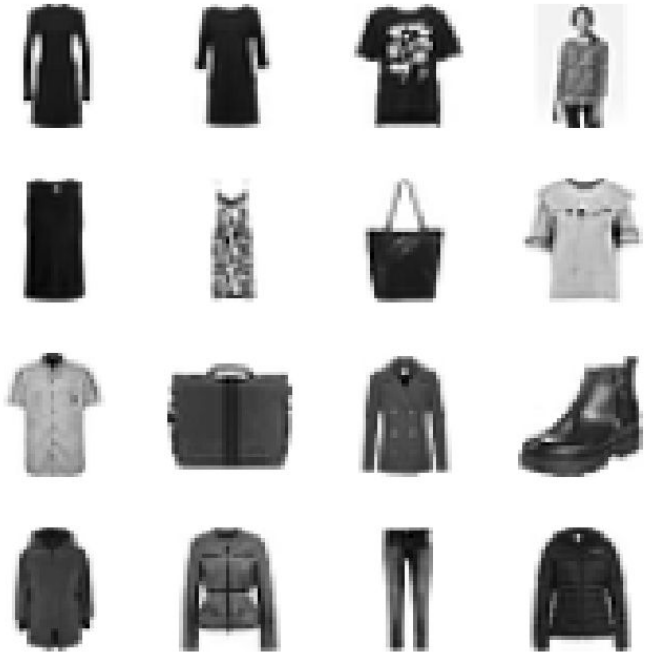
- Generating new images
- Image Denoising
- Image Compression
- Dimensionality Reduction

Image Reconstruction

Reconstructed Images - 2D Latent Space

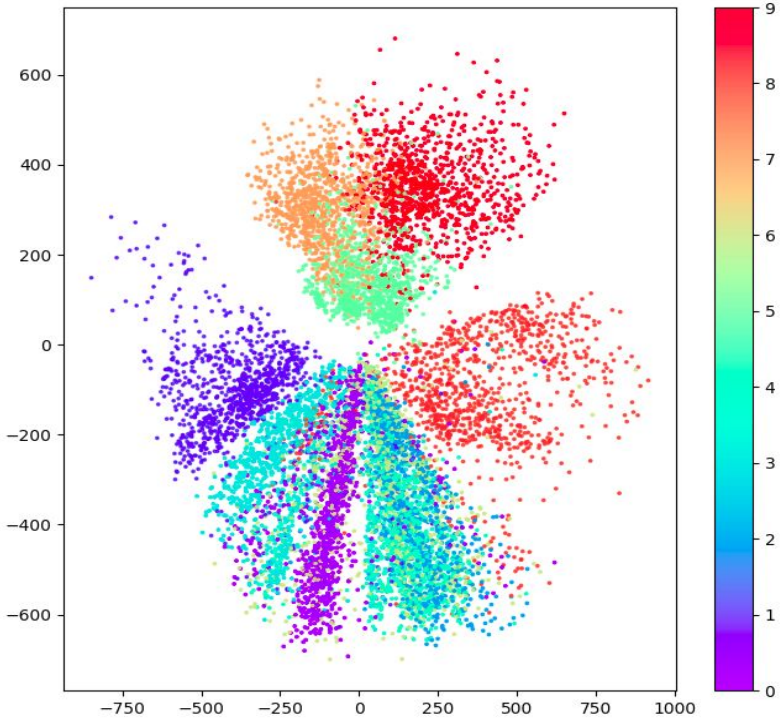


Ground Truth



Why are the reconstructed images blurry?
How can we improve the quality?

2D Latent Space of Fashion MNIST

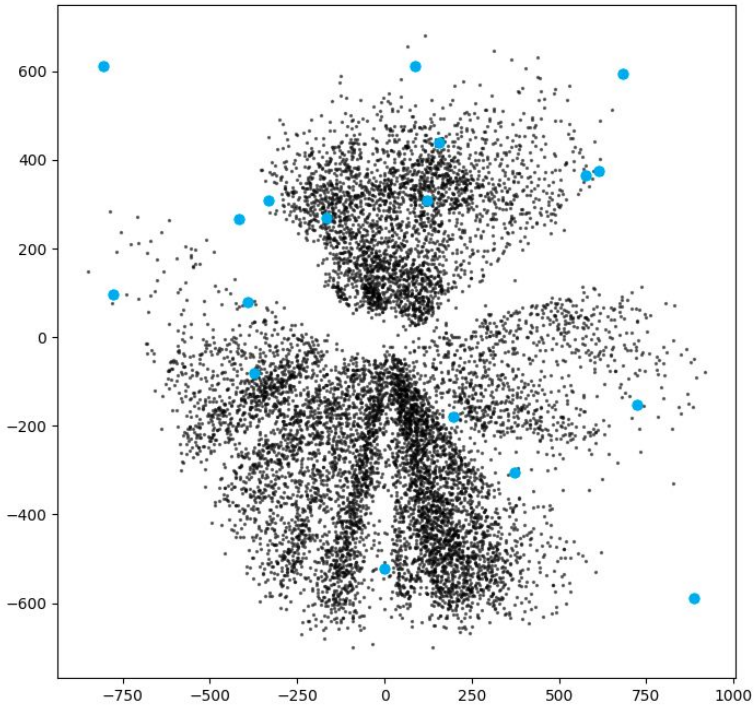


Class Mappings

0: T-Shirt/Top
1: Trouser
2: Pullover
3: Dress
4: Coat
5: Sandal
6: Shirt
7: Sneaker
8: Bag
9: Ankle Boot

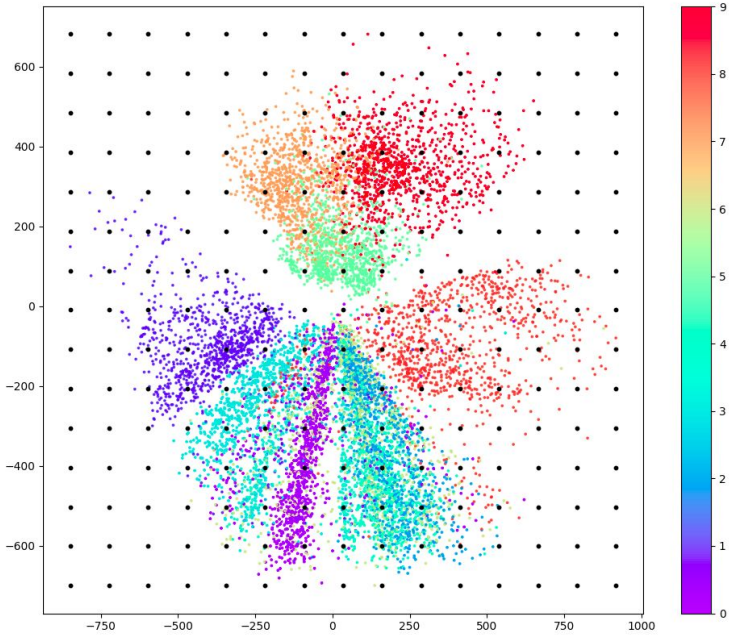
Notice how the autoencoder has naturally grouped items that look alike into the same parts of the latent space, even though the class labels were not used for training.

Sampled Points (In Blue)

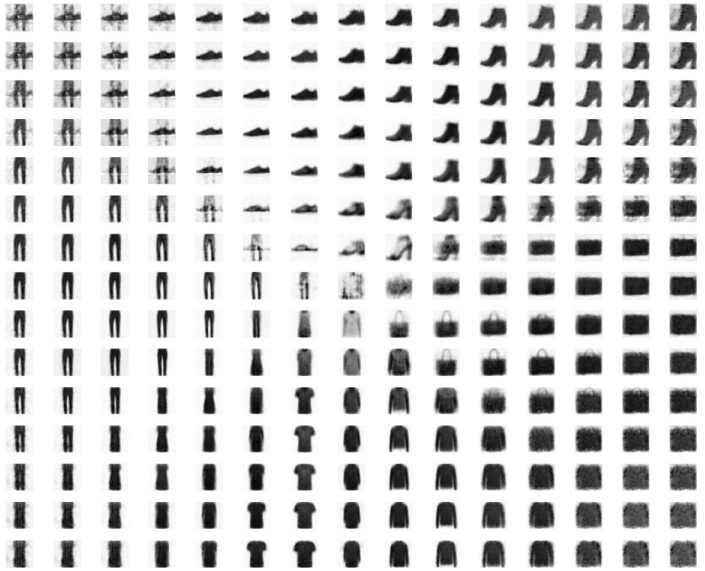


Generated Images

Sampled Points (In Black)



Generated Images



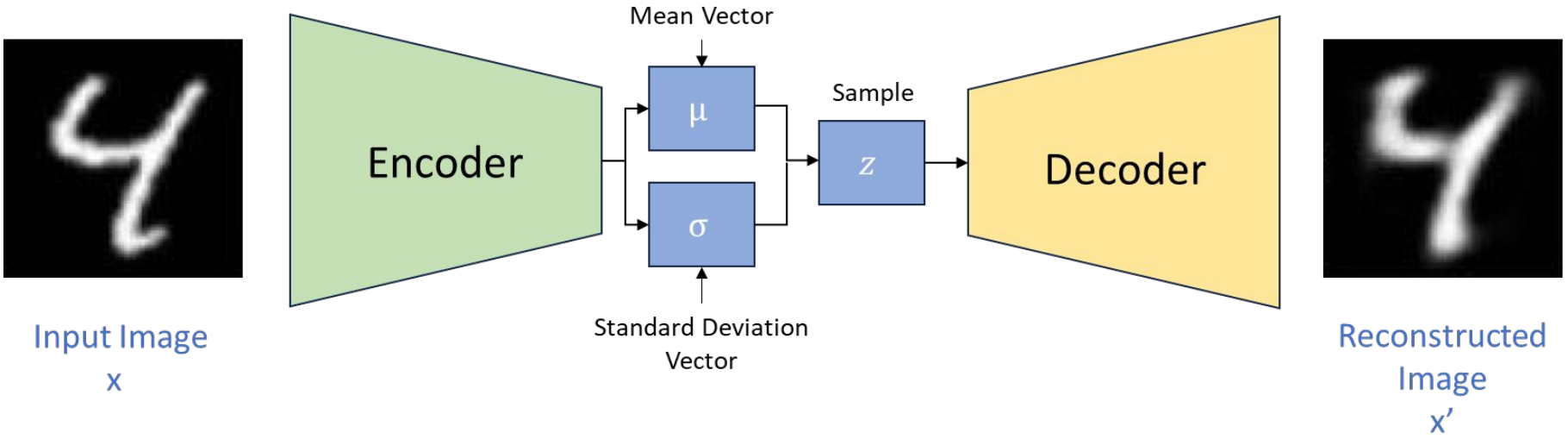
Drawbacks:

1. The underlying distribution of the latent space is unknown. This makes sampling challenging.

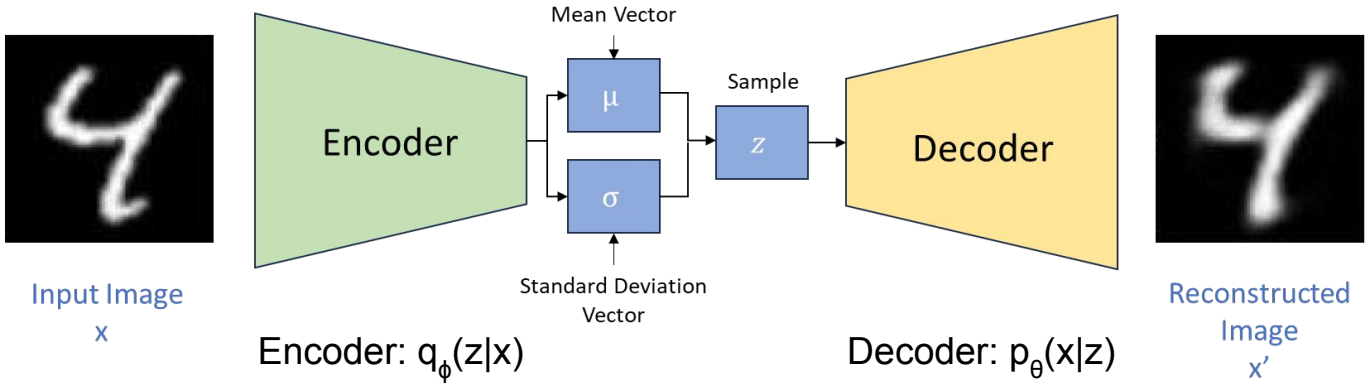
2. There is no constraint on the area/volume occupied by the class features. This means that features of a single class can occupy a very large volume compared to the others. During sampling, this class will be over-represented.

3. Local discontinuities: The latent space may not be continuous. For example, even though the point (–1, –1) might be decoded to give a satisfactory image of a sandal, there is no mechanism in place to ensure that the point (–1.1, –1.1) also produces a satisfactory image of a sandal.

Variational Autoencoders



The encoder takes an input image and encodes it into two vectors that defines a multivariate normal distribution in the latent space.



$$\text{Loss} = \underbrace{||x-x'||^2}_{\text{Reconstruction Loss}} + \underbrace{KL\,(q_\phi(z|x)\,||\,p(z))}_{\text{Regularisation}}$$

Kullback-Leibler Divergence Loss

$$KL\ (q_\phi(z|x)\ ||\ p(z))$$

Distribution in latent space

Desired prior distribution

The prior distribution p(z) is usually chosen to be a multivariate Gaussian with mean 0 and variance 1 for simplicity.

$$\mathrm{KL}(q_\phi(\mathrm{z}|\mathrm{x})||p(\mathrm{z})) = -\frac{1}{2}\sum_{j=0}^{k-1}(\sigma_j + \mu_j^2 - 1 - \log\sigma_j)$$

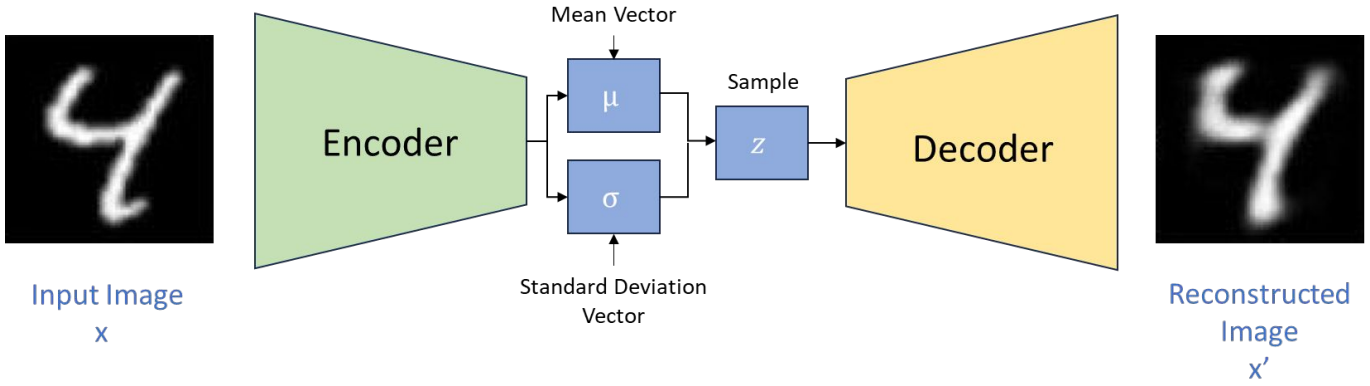
What does regularization do?

- Constrain the latent variables to follow a desired distribution, which makes the latent space more meaningful and interpretable.

- Enables smooth interpolation between data points in the latent space. This means that points that are close in the latent space contains similar content.

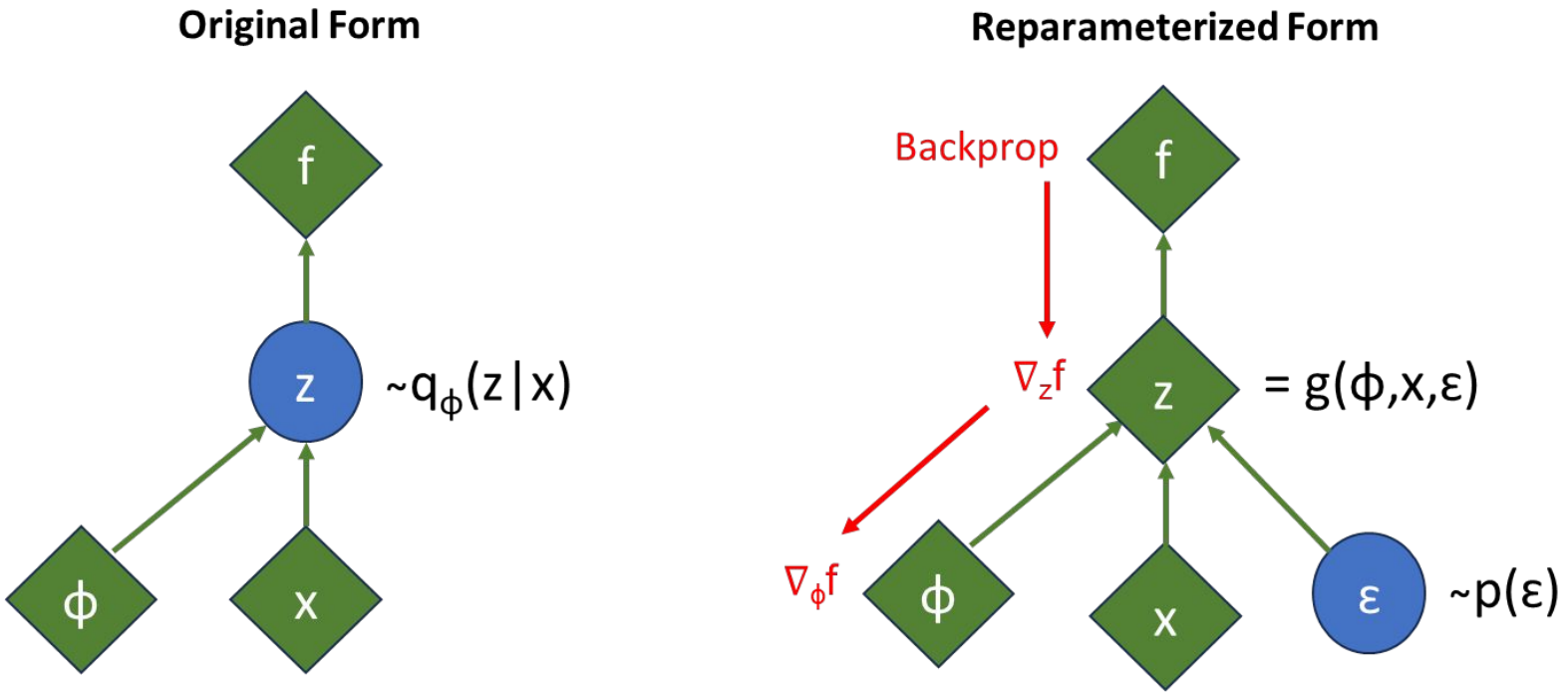
- Encourages the model to learn more general features of the data, rather than memorizing specific data points. This helps the VAE to generate more diverse and coherent samples during the generative phase.

The reparameterization trick



We cannot backpropagate gradients through sampling layer

Solution: Use the Reparameterization Trick



Idea: Make the randomness an input to your model instead of something within it.

Sampling of latent variables is transformed into a combination of a deterministic function and a separate random noise term.

Reparameterized form for Gaussian distribution:

$$z = \mu + \sigma * \varepsilon$$

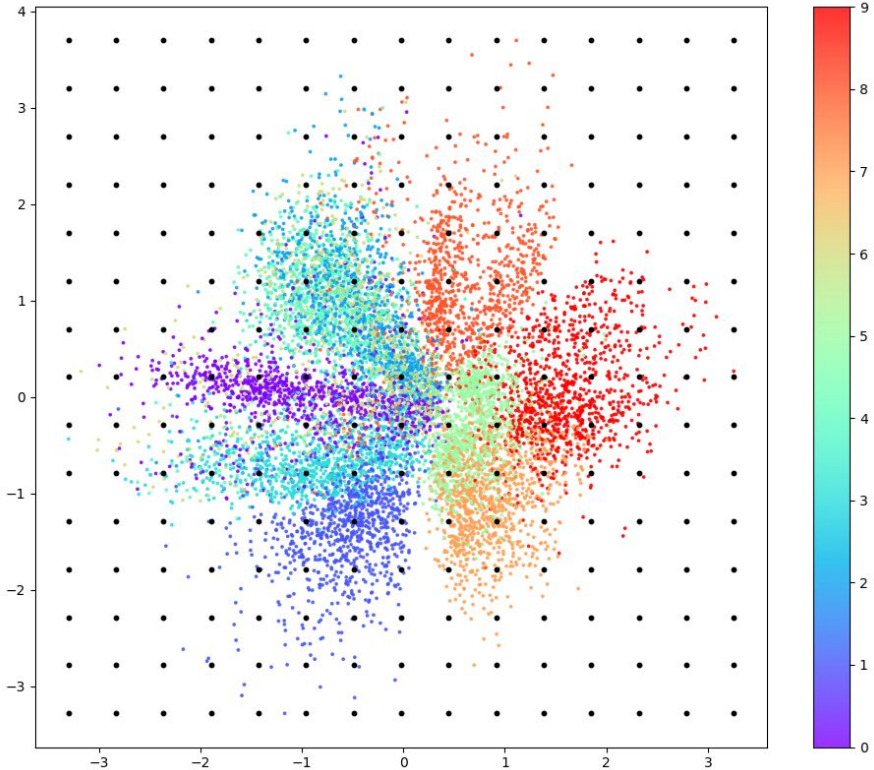
where:

- z is the sampled latent variable.
- μ is the mean of the Gaussian distribution.
- σ is the standard deviation of the Gaussian distribution.
- ε is a sample from a standard Gaussian distribution (i.e., $\varepsilon \sim N(0, 1)$).

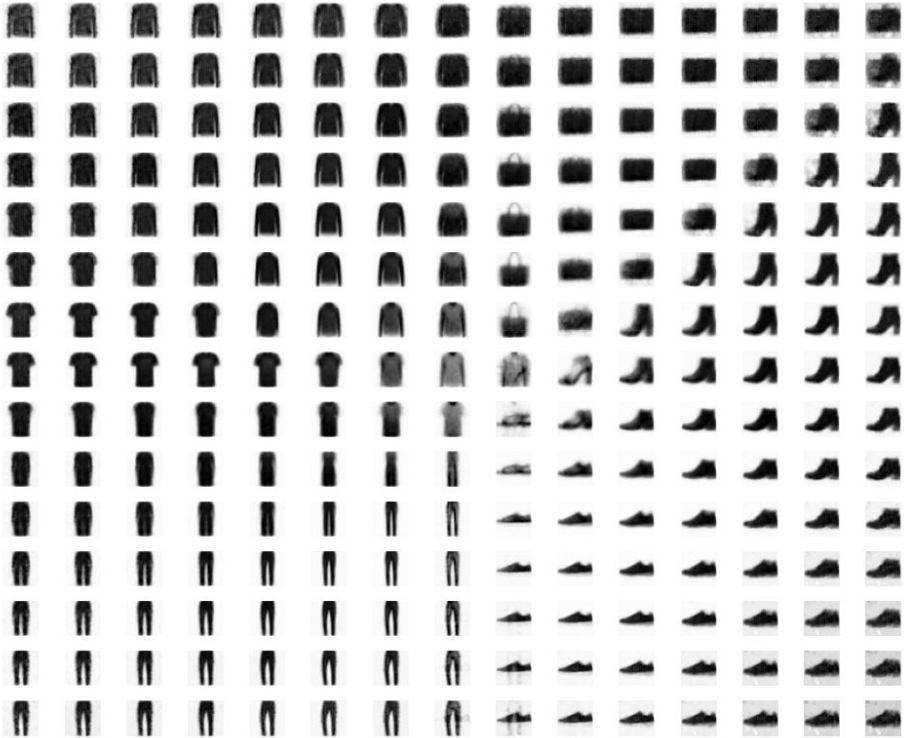
The process of sampling z is transformed into a deterministic transformation of μ and σ, combined with the randomness introduced by ε.

Results of VAE trained on Fashion MNIST

Sampled Points (In Black)



Generated Images

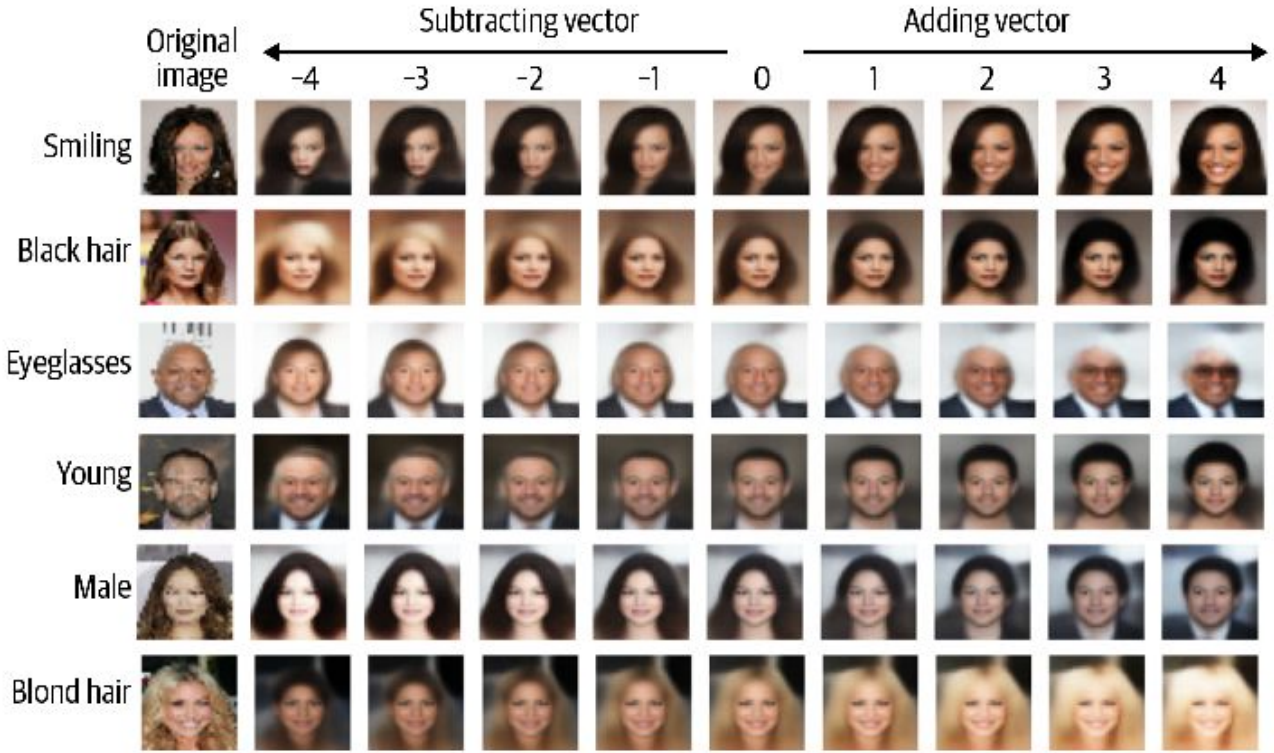


Application - Image Manipulation

- Image consists of various features. The vectors in the latent space can be modified to change the corresponding visual features.
- For example, if we want to change the hair color of a person in an image, we need to find a vector in the latent space that points in the direction of the desired color.
- To find this vector,
  - Collect encoded representations of images with the desired attribute.
  - Gather encoded representations of images lacking that attribute
  - Calculate the difference between these two sets of representations to obtain the attribute vector.
- Adding this vector to the encoding of the original image in the latent space will give us a new point which, when decoded, should give us a new manipulated image.

$$z\_new = z + alpha * feature\_vector$$

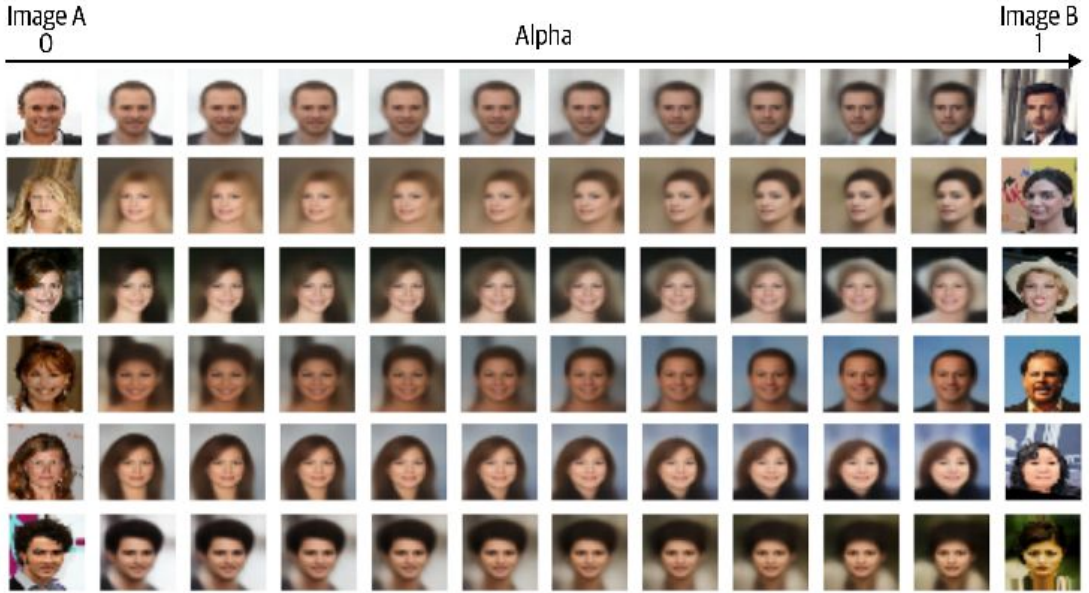
where, alpha is a weighting factor for the desidered feature.


Original image
Subtracting vector
Adding vector
-4
-3
-2
-1
0
1
2
3
4
Smiling
Black hair
Eyeglasses
Young
Male
Blond hair

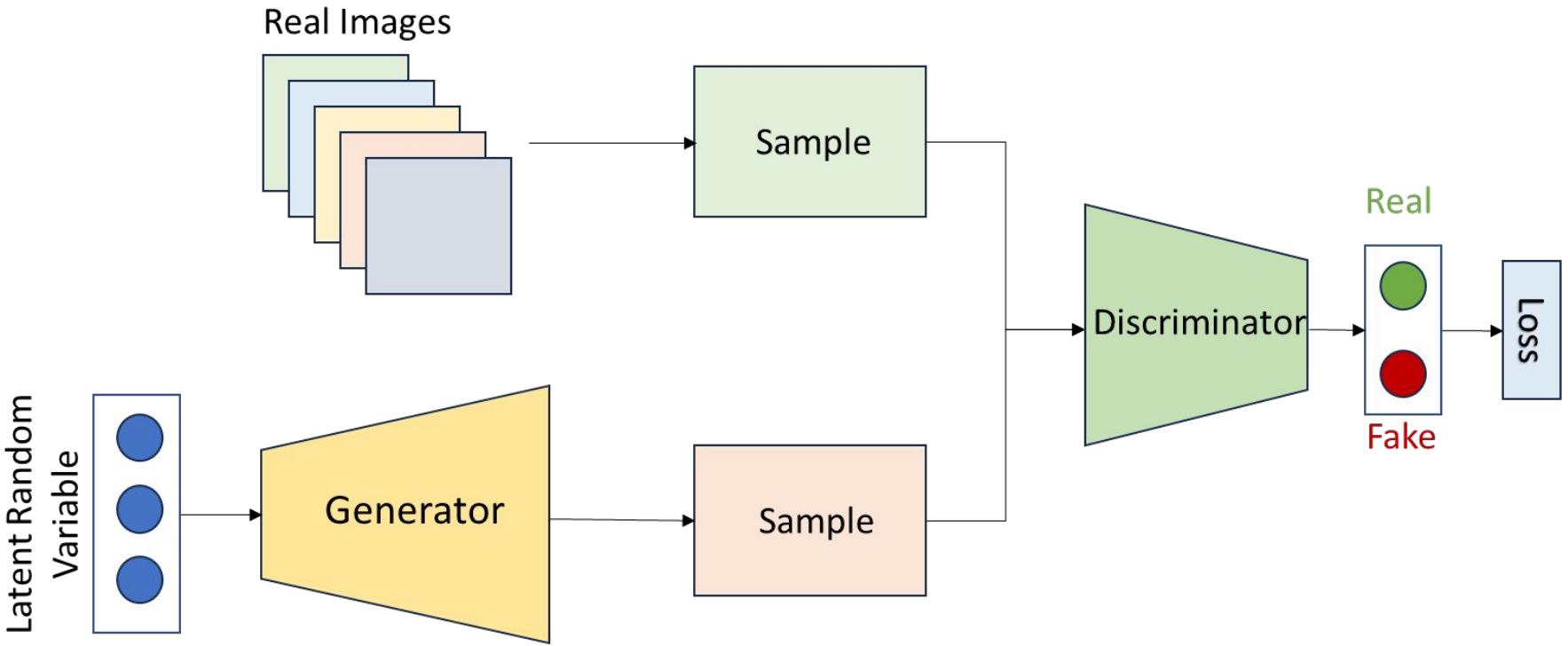
Application - Image Morphing

Imagine two points in the latent space, A and B, that represent two images. If you started at point A and walked toward point B in a straight line, decoding each point on the line as you went, you would see a gradual transition from the starting face to the end face.

$$z_{new} = z_A * (1- alpha) + z_B * alpha$$



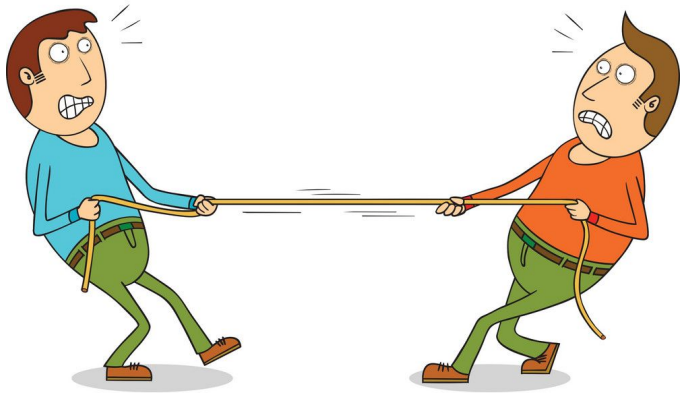
Generative Adversarial Networks (GANs)



Generator

Input: Vector drawn from a multivariate standard normal distribution.

Output: Image of the same size as the training images.



Discriminator

Binary Classifier - Predicts if an image is real or fake

The generator and discriminator compete against each other.

The generator tries to trick the discriminator into identifying the generated image as real.
The discriminator tries to correctly identify the real and the fake images.

Training a GAN

Discriminator Loss: $$\arg\max_D \mathbb{E}_{\mathbf{z},\mathbf{x}}[\underbrace{\log D(G(\mathbf{z}))}_{\text{Fake}} + \underbrace{\log(1 - D(\mathbf{x}))}_{\text{Real}}]$$

Generator Loss: $$\arg\min_G \mathbb{E}_{\mathbf{z},\mathbf{x}}[\log D(G(\mathbf{z})) + \log(1 - D(\mathbf{x}))]$$

Combined Loss: $$\arg\min_G \max_D \mathbb{E}_{\mathbf{z},\mathbf{x}}[\log D(G(\mathbf{z})) + \log(1 - D(\mathbf{x}))]$$

Intuition behind the objective functions:

- The loss is a binary cross-entropy loss, which is commonly used when training a binary classifier.
- The discriminator loss tries to assign a label value of 1 to the fake images and a label value of 0 to the real images.
- The generator loss, on the other hand, tries to assign a label value of 0 to the fake images and a label value of 1 to the real images.
- In other words, the discriminator and generator are in ‘adversary’ to each other as they attempt to achieve opposite objectives.
- The training goal is to find an equilibrium between the two objective functions.

Training a GAN can be tricky … some tips and tricks

1. Discriminator overpowers the generator -

   Discriminator perfectly learns to separate real images from fake images and the gradients vanish completely, leading to no training.

   a. Increase the rate parameter of the Dropout layers in the discriminator to dampen the amount of information that flows through the network.
   b. Reduce the learning rate of the discriminator.
   c. Reduce the number of convolutional filters in the discriminator.
   d. Add noise to the labels when training the discriminator.
   e. Flip the labels of some images at random when training the discriminator.

2. Generator overpowers the discriminator -

The generator would be inclined to find a single observation (also known as a mode) that always fools the discriminator and would start to map every point in the latent input space to this image. This is also known as **mode collapse.**

a. Strengthen the performance of discriminator
b. Use Wasserstein Loss.
c. Decrease the learning rate of both the networks and increase the batch size.

DCGAN to generate human faces

Celeb-A dataset

Real Images

Fake Images

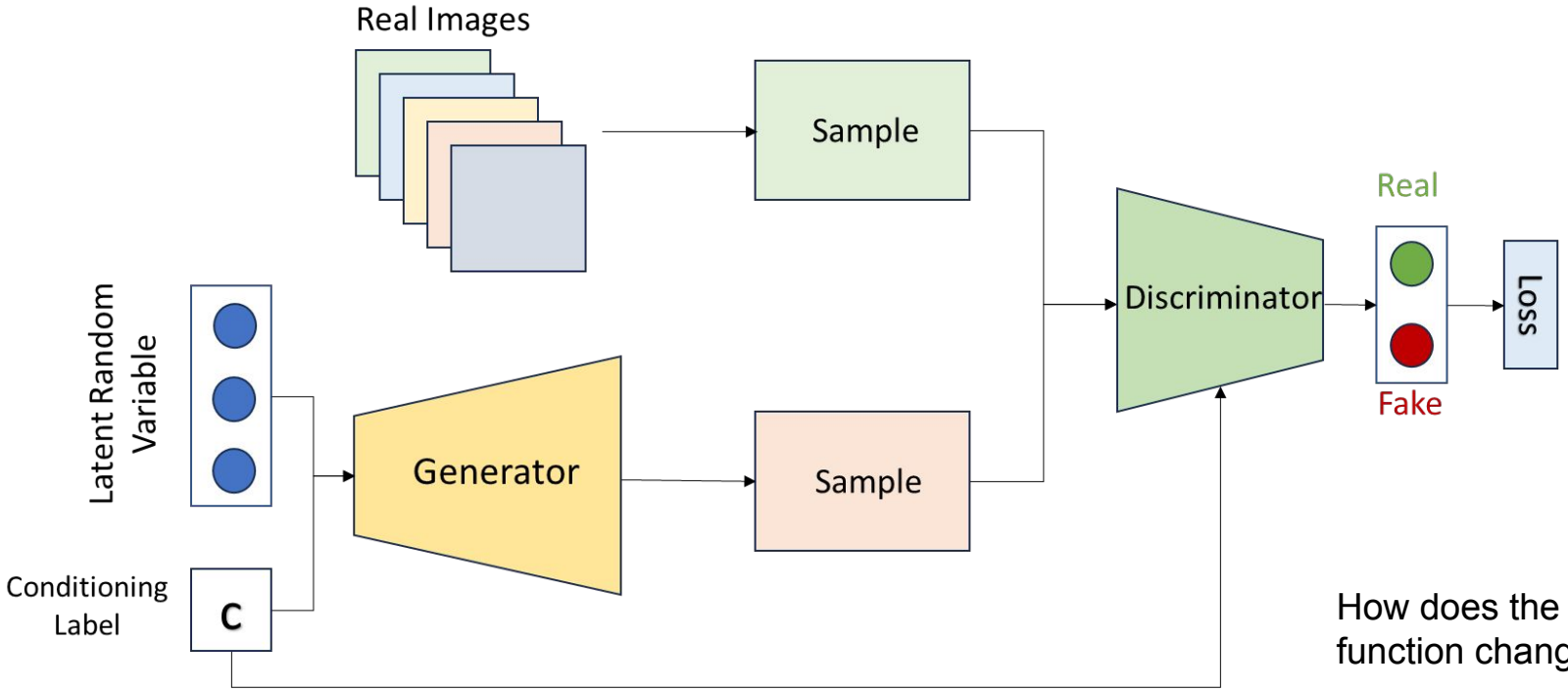
Some state-of-the-art GANs

Progressive GAN

StyleGAN

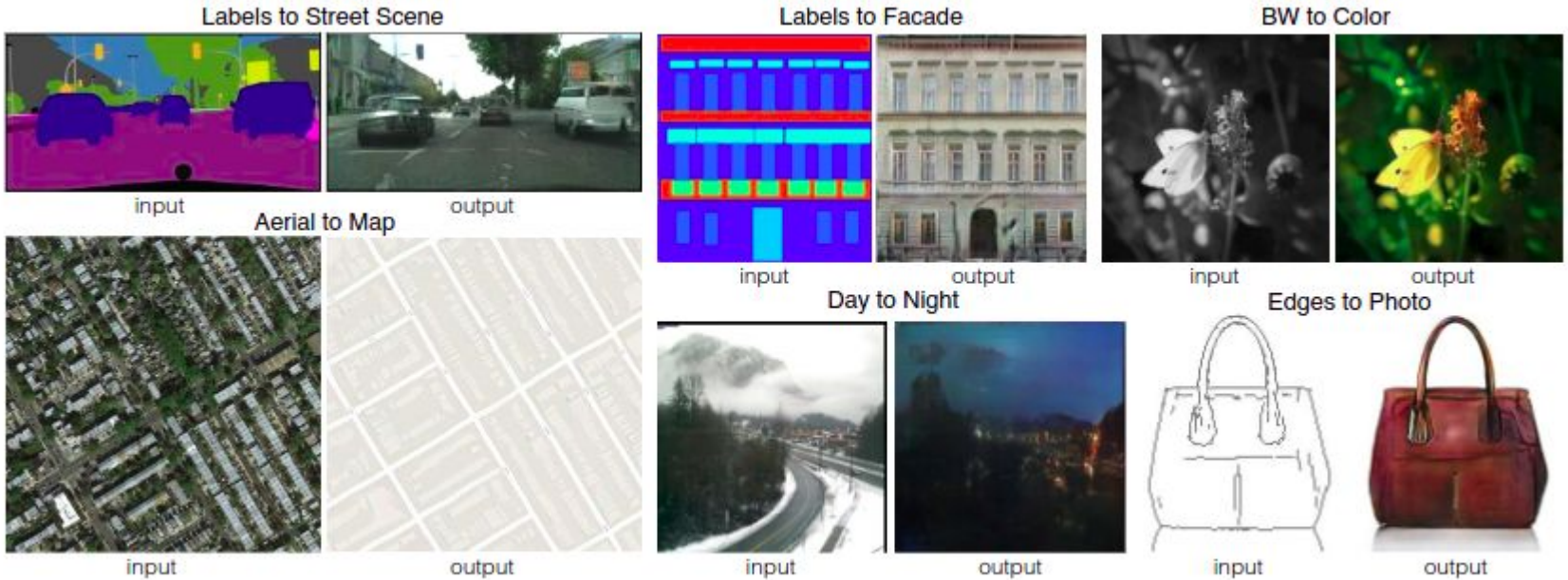
Conditional GANs

To control the type of image that we want to generate. The conditioning labels control the generated images.

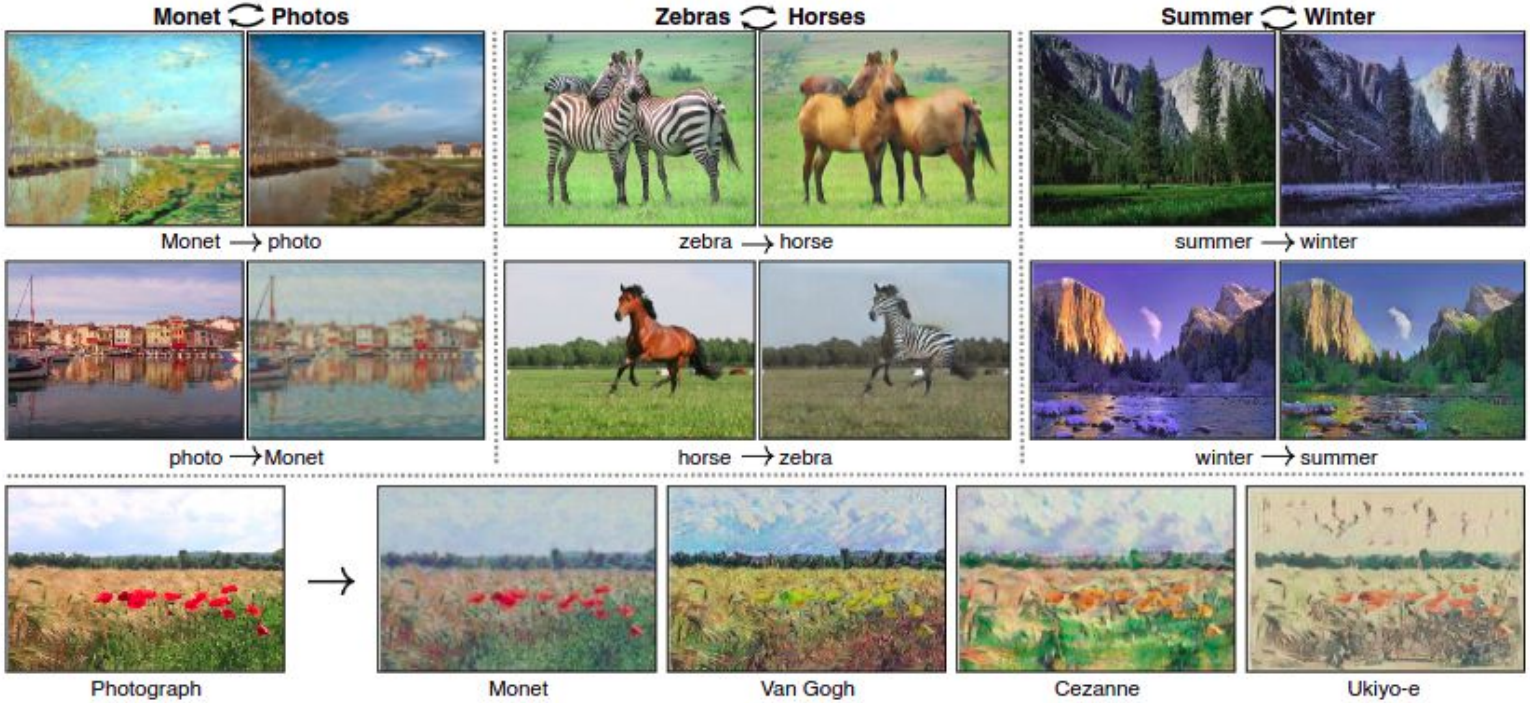


How does the loss function change?

Pix2pix - paired image-to-image translation



CycleGAN - unpaired image-to-image translation (domain transformation)



Diffusion Model

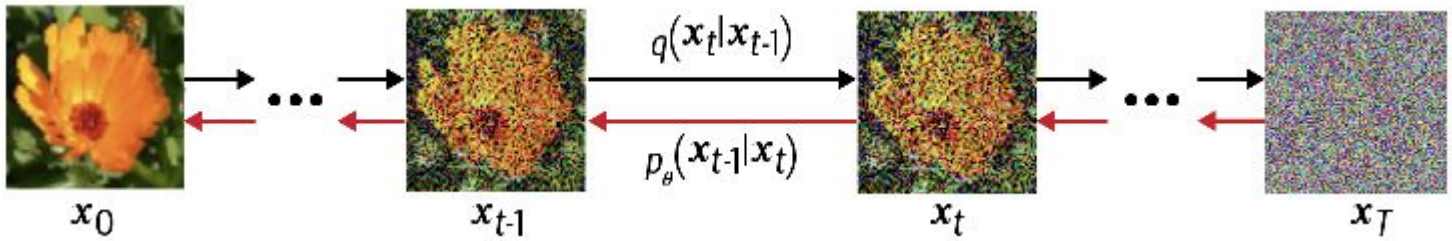
A photo of a Persian cat wearing sunglasses and red shirt playing a guitar on top of a mountain. (Imagen)

A stern looking owl dressed as a librarian, digital art (DALL-E)

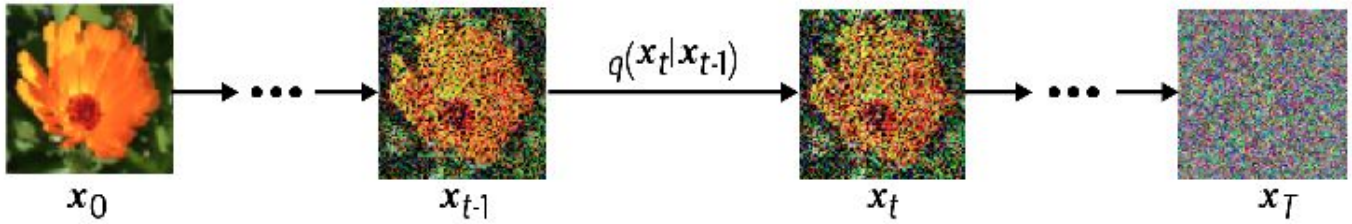
Denoise Diffusion Probabilistic Models

Working Principle: During training, noise is iteratively added to an image dataset (forward process). The model is trained to undo each of the steps to recover the original image (backward / sampling process).

If we are able to effectively learn the backward process, we can develop a model capable of creating images from entirely random noise.



Forward Process



At each step, we add a small amount of Gaussian noise with variance β_t to an image x_{t-1} to generate a new image x_t.

$$\mathbf{x}_t = \sqrt{(1-\beta_t)}\mathbf{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}_{t-1}$$

$\boldsymbol{\epsilon}_{t-1}$ - Standard Gaussian with zero mean and unit variance

If we assume that x_{t-1} has zero mean and unit variance, then $\sqrt{1-\beta_t}x_t$ will have a variance of $1 - \beta_t$ and $\sqrt{\beta_t}\epsilon_{t-1}$ will have a variance of β_t.

This step ensures that the output image x_t has unit variance over each time step. The final image x_T will be a standard Gaussian for a large enough T.

The forward noising process q can also be written as follows:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) \coloneqq \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

We apply q repeatedly from timestep 1 to T. Suppose T=1000, we need to apply q 1000 times. A desirable scenario is where we are able to jump straight from an image x_0 to any noised version of the image x_t without having to go through t applications of q .

We can use a reparameterization trick here.

If we define $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, then,

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\epsilon \\
&= \cdots \\
&= \sqrt{\overline{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t}\epsilon
\end{aligned}$$

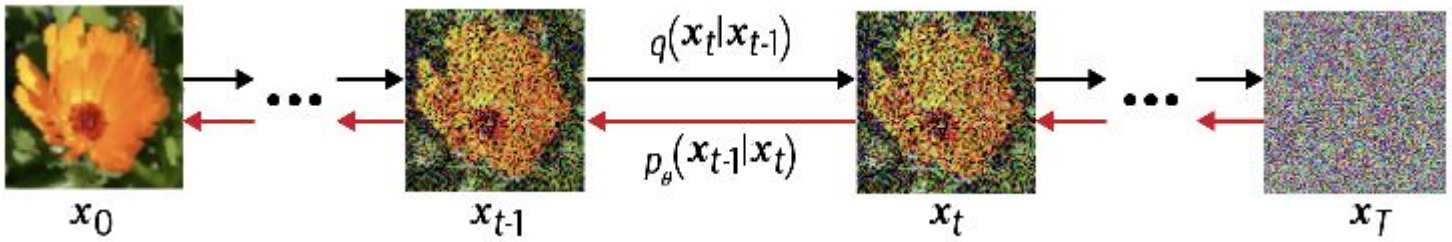
To produce a sample x_t we can use the following distribution:

$$\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

Diffusion Schedules

We are also free to choose a different β at each timestep. How the β values change with t is called the diffusion schedule. There are several scheduling types such as linear, quadratic, cosine etc.

Backward Process



To perform the reverse diffusion process, we are looking to build a neural network $p_\theta(x_{t-1}|x_t)$ that can undo the noising process—that is, approximate the reverse distribution $q(x_{t-1}|x_t)$.

If we can do this, we can sample random noise from N(0,I) and then apply the reverse diffusion process multiple times in order to generate a novel image.

The objective function can be optimized by training a network ϵ_θ to predict the noise ϵ that has been added to x_0 at timestep t.

Recall that at time step t, the image is given by $\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$

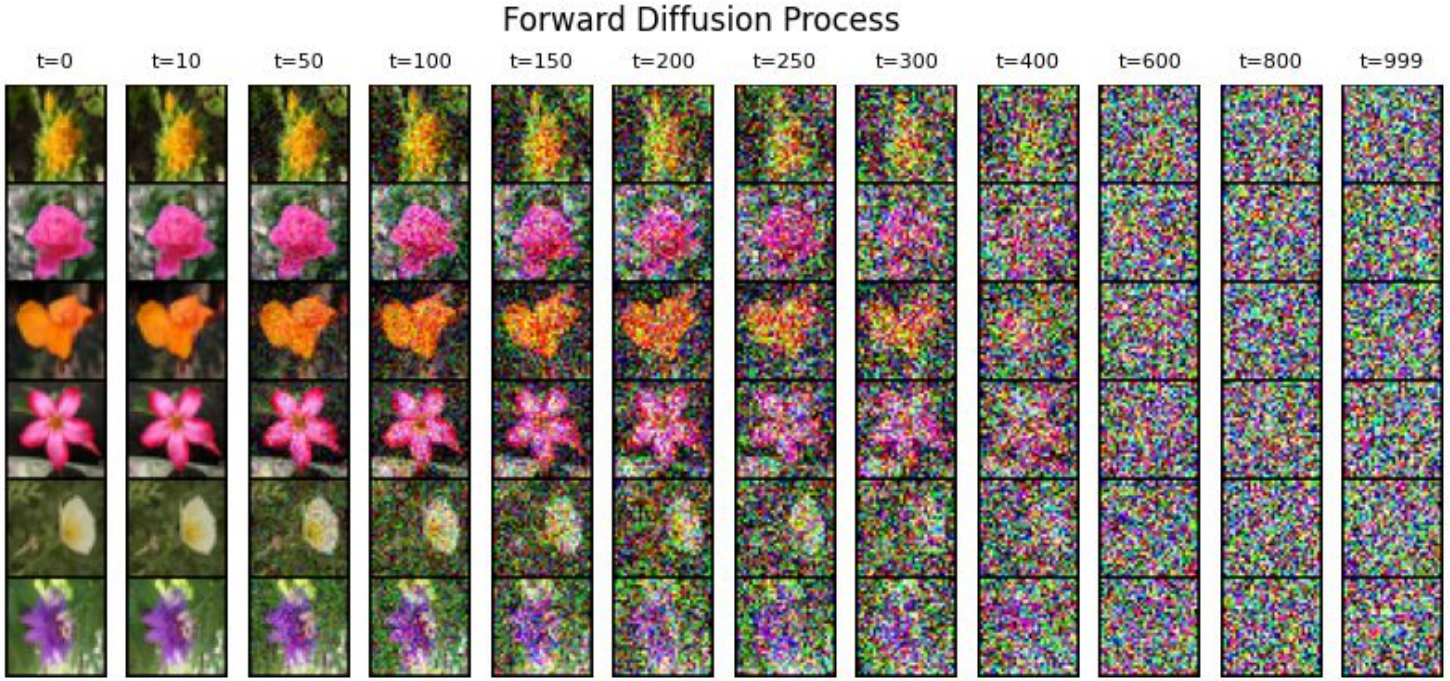
We give this new image and the noising rate $\bar{\alpha}$ to the neural network and ask it to predict ϵ.

The loss function is the square error between the predicted ϵ and the true ϵ.

$$\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2$$

Experiments

Oxford 102 Flowers Dataset

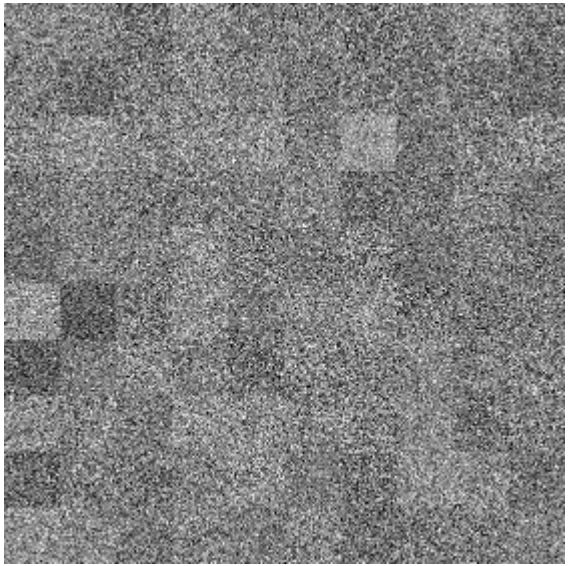

Forward Diffusion Process
t=0
t=10
t=50
t=100
t=150
t=200
t=250
t=300
t=400
t=600
t=800
t=999

Backward Diffusion Process

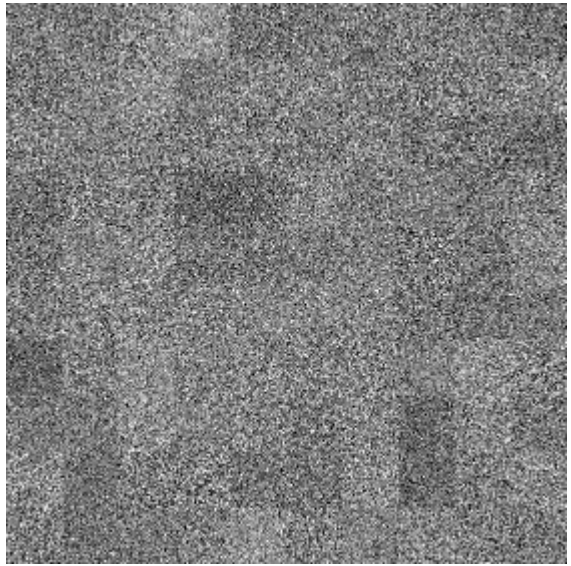
Final Generated Images

Some more results

MNIST



Fashion MNIST



Try it yourself:

https://github.com/davidADSP/Generative_Deep_Learning_2nd_Edition.git (Tensorflow)

https://github.com/vaishwarya96/Deep-Generative-Models.git (PyTorch)(Work in progress)

Reference:

Generative Deep Learning Second edition by David Foster