Implementing Variational Autoencoder (VAE) on MNIST dataset using PyTorch

by

Ravi Saroj

RWI2023003

Assignment: Probabilistic Machine Learning and Graphical Model

25-April-2023

Department of Information and Technology

Indian Institute of Information Technology, Allahabad

Table of Contents

Abstract

Variational Autoencoder (VAE) is a powerful generative model that can learn a compact latent representation of the data and generate new data samples by sampling from the learned latent space. In this report, we implemented a VAE on the MNIST dataset using PyTorch and evaluated the performance of the model. The goal of this study is to develop a generative model for the MNIST dataset that can generate new images that resemble digits.

Introduction

Variational Autoencoder (VAE) is a type of generative model that can learn a low-dimensional representation of the data and generate new data samples from the learned latent space. VAEs are a type of autoencoder, which is a neural network architecture that can learn to encode the input data into a low-dimensional representation and then decode it back to the original data.

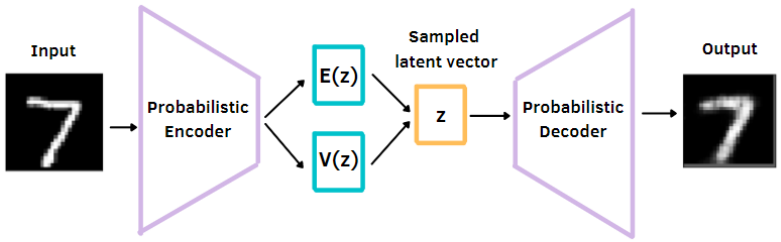
VAEs are trained to maximize the evidence lower bound (ELBO) objective, which consists of two terms: the reconstruction loss and the KL divergence loss. The reconstruction loss measures the difference between the input and the reconstructed output, while the KL divergence loss regularizes the learned latent space by penalizing the deviation from a prior distribution.

The VAE model consists of two parts: an encoder and a decoder. The encoder maps the input data to a mean and variance of a normal distribution in the latent space, and the decoder takes a sample from this distribution and generates the output data. The latent space is typically of lower dimensionality than the input data, allowing for a compact representation of the data.

VAEs have many applications in generative modeling, such as image and text generation. They have also been used in unsupervised learning and semi-supervised learning tasks, where the labeled data is scarce.

One of the limitations of VAEs is that they tend to generate blurry images and lack sharpness in details. This is due to the use of the KL divergence loss, which can constrain the learned latent space too much and result in a loss of information. To overcome this limitation, many variants of the VAE model have been proposed, such as the conditional VAE (CVAE) and the adversarial autoencoder (AAE).

VAEs are a powerful generative model that can learn a useful representation of the data and generate new data samples. They have many applications in various fields and continue to be an active area of research.



Methodologies

Encoder: The first step in implementing a VAE is to design an encoder neural network that maps the input data to a distribution in the latent space. The encoder typically consists of several layers of neurons that apply linear and non-linear transformations to the input data.

The encoder takes an input x and maps it to the mean mu and standard deviation log_var of a normal distribution in the latent space. The encoder consists of two fully connected layers with ReLU activation function followed by two output layers for mu and log_var. The formulas for the encoder are:

Latent Space: The latent space is a low-dimensional representation of the input data that captures the most important features of the data. The size of the latent space is a hyperparameter that needs to be chosen based on the complexity of the data and the desired level of compression.

Sampling: After the encoder maps the input data to the latent space, a sample is drawn from the distribution in the latent space. This sample is used as input to the decoder, which generates the output data.

Decoder: The decoder is a neural network that takes the sample from the latent space and maps it back to the original data space. The decoder typically consists of several layers of neurons that apply linear and non-linear transformations to the sample.

The decoder takes a point z from the latent space and maps it back to the original input space. The decoder consists of two fully connected layers with ReLU activation function followed by an output layer with sigmoid activation function

Loss Function: The loss function in VAE consists of two parts: the reconstruction loss and the KL divergence loss. The reconstruction loss measures the difference between the input data and the reconstructed output, while the KL divergence loss regularizes the learned distribution in the latent space.

The loss function for VAE consists of two parts: the reconstruction loss and the KL divergence loss. The reconstruction loss measures the difference between the input x and the output x_hat obtained from the decoder. The KL divergence loss measures the difference between the latent distribution and the standard normal distribution.

Optimization: The optimization method is used to train the VAE by minimizing the loss function. The most commonly used optimization method is stochastic gradient descent (SGD) with the Adam optimizer.

Hyperparameters: There are several hyperparameters involved in the implementation of VAE, such as the size of the latent space, the number of layers in the encoder and decoder, the learning rate, and the batch size. These hyperparameters need to be tuned to achieve optimal performance.

Training: The VAE is trained by minimizing the loss function with respect to the model parameters using stochastic gradient descent. The training process consists of the following steps:

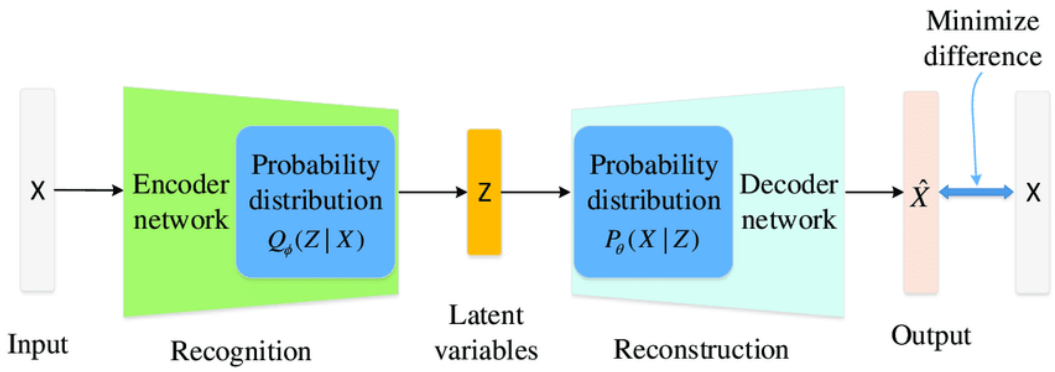
Forward pass: The input x is passed through the encoder to obtain mu and log_var. A point z is then sampled from the normal distribution in the latent space using mu and log_var. The point z is passed through the decoder to obtain x_hat.

Loss calculation: The reconstruction loss and the KL divergence loss are calculated using x, x_hat, mu, and log_var.

Backward pass: The gradients of the loss with respect to the model parameters are calculated using backpropagation.

Parameter update: The model parameters are updated using the gradients and the chosen optimizer.

Overall, the implementation of VAE involves designing and training an encoder and decoder neural network, choosing the size of the latent space, sampling from the learned distribution, defining the loss function, and tuning the hyperparameters to achieve optimal performance.



Implementation:

We implemented the VAE model using PyTorch and trained it on the MNIST dataset for 50 epochs using the Adam optimizer with a learning rate of 1e-3. We stored the training loss for each epoch and visualized the learned latent space by encoding the test dataset and plotting the latent vectors with their corresponding labels. We also generated new images by sampling from the learned latent space and compared them with the real images.

Results:

After training the VAE model for 50 epochs, we evaluated its performance by generating new images from the learned latent space. We randomly sampled 10 latent vectors from a standard normal distribution and used the decoder to generate new images.

Training Data:

The generated 64 images are shown in the below Figure.

Trend of Loss with respect to Epoch: