Difference between AutoEncoder (AE) and Variational AutoEncoder (VAE)

medium.com/towards-data-science/difference-between-autoencoder-ae-and-variational-autoencoder-vae-ed7be1c038f2

Aqeel Anwar November 4, 2021

How can you compress data or even generate data from random values? That is what Autoencoder and Variational Autoencoder are.

> The ability to simplify means to eliminate the unnecessary so that the necessary may speak — **Hans Hofmann**

Context— Data compression

Data compression is an essential phase in training a network. The idea is to compress the data so that the same amount of information can be represented by fewer bits. This also helps with the problem of the curse of dimensionality. A dataset with many attributes is different to train with because it tends to overfit the model. Hence dimensionality reduction techniques need to be applied before the dataset can be used for training.

This is where the Autoencoder (AE) and Variational Autoencoder (VAE) come into play. They are end-to-end networks that are used to compress the input data. Both Autoencoder and Variational Autoencoder are used to transform the data from a higher to lower-dimensional space, essentially achieving compression.

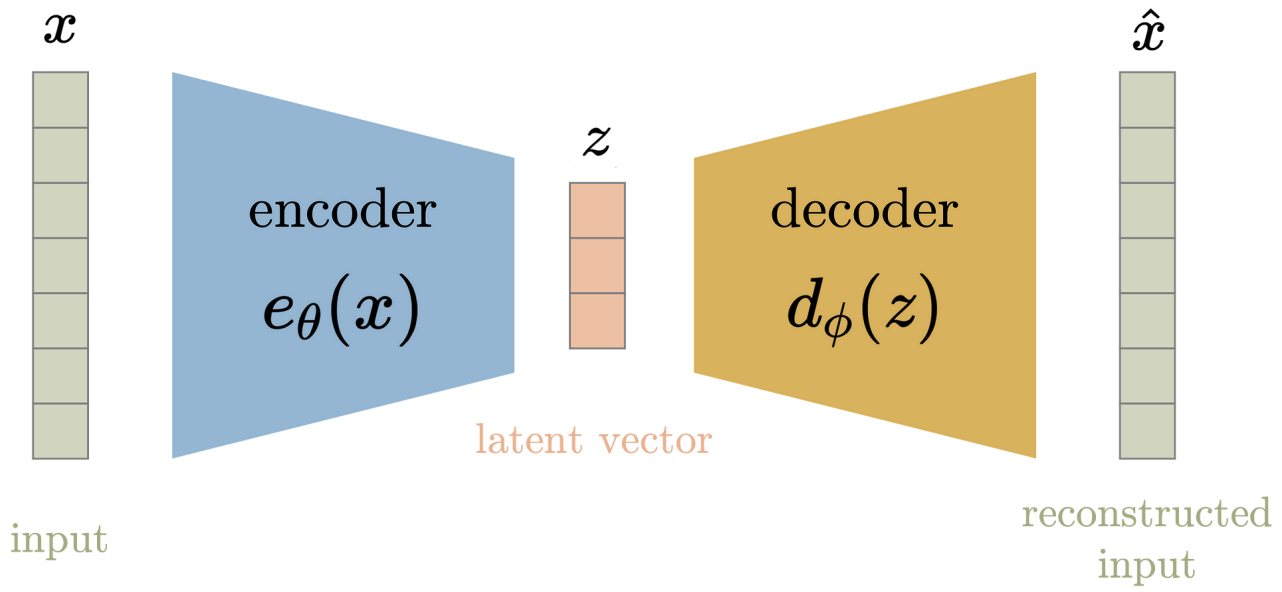
Autoencoder — AE

What is it?

Autoencoder is used to learn efficient embeddings of unlabeled data for a given network configuration. The autoencoder consists of two parts, an encoder, and a decoder. The encoder compresses the data from a higher-dimensional space to a lower-dimensional space (also called the latent space), while the decoder does the opposite i.e., convert the latent space back to higher-dimensional space. The decoder is used to ensure that latent space can capture most of the information from the dataset space, by forcing it to output what was fed as input to the decoder.

Block Diagram

The block diagram can be seen below.



$$loss = \|x - \hat{x}\|_2 = \|x - d_\phi(z)\|_2 = \|x - d_\phi(e_\theta(x))\|_2$$

AutoEncoder — Image by Author

During training, the input data x is fed to the encoder function . The input is passed through a series of layers (parameterized by the variable theta) reducing its dimensions to achieve a compressed latent vector . The number of layers, type and size of the layers, and the latent space dimension are user-controlled parameters. Compression is achieved if the dimension of the latent space is less than that of the input space, essentially getting rid of redundant attributes.

The decoder usually (but not necessarily) consists of near-compliment layers of the layers used in the encoder but in reverse order. A near-complement layer of a layer is the one that can be used to undo the operations (to some extent) of the original layer such as transposed conv layer to conv layer, pooling to unpooling, fully connected to fully connected, etc.

Loss function

The entire encoder-decoder architecture is collectively trained on the loss function which encourages that the input is reconstructed at the output. Hence the loss function is the mean squared error between the encoder input and the decoder output.

$$loss = \|x - \hat{x}\|_2 = \|x - d_\phi(z)\|_2 = \|x - d_\phi(e_\theta(x))\|_2$$

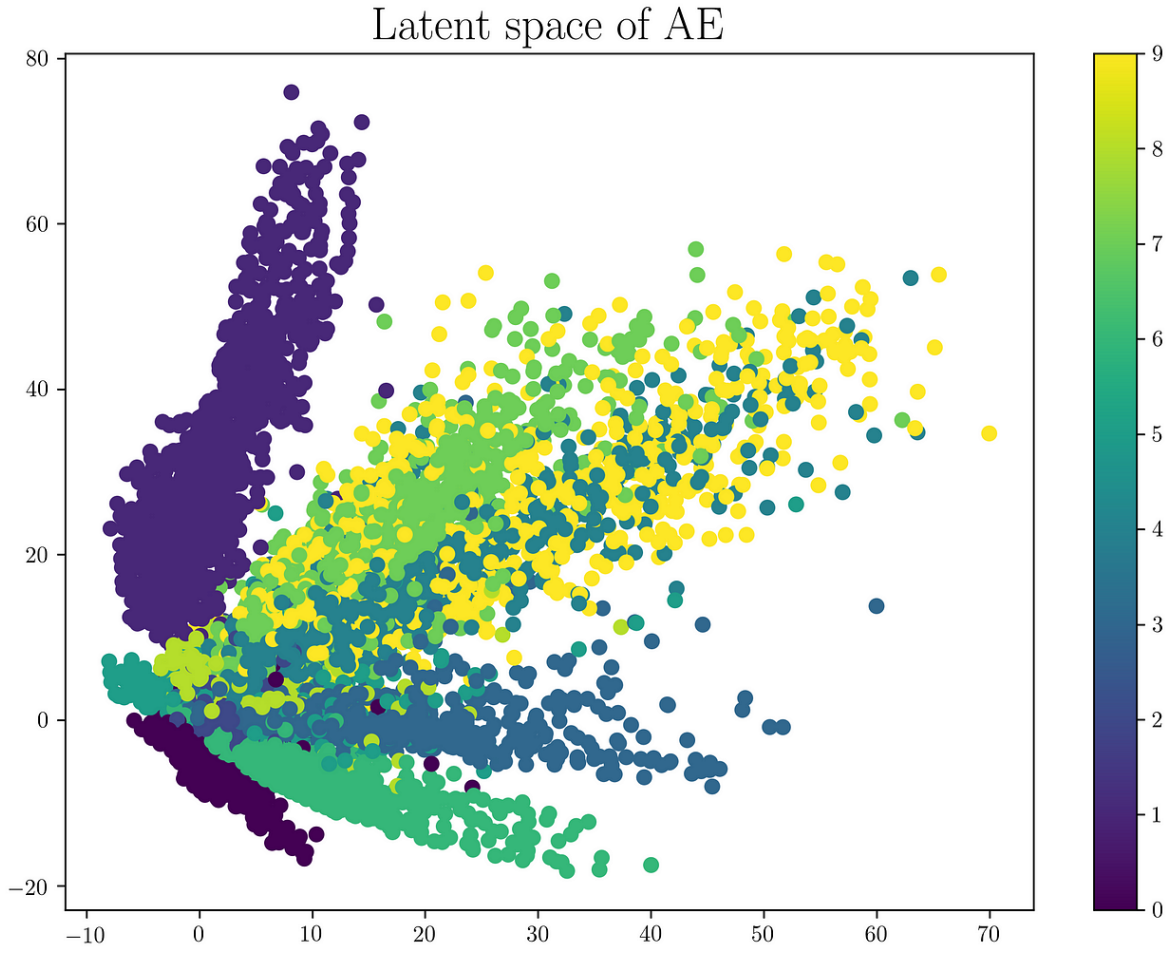
Autoencoder loss function — Image by Author

The idea is to have a very low dimensional latent space so that maximum compression is achieved, but at the same time, the error is small enough. Reducing the dimension of the latent space beyond a certain value will result in a significant loss of information.

There are no constraints on the values/distribution of the latent space. It can be anything, as long as it can reconstruct the input when the decoder function is applied to it.

Latent space visualization

Below is an example of the latent space generated by training the network on an MNIST dataset.



Latent space of Autoencoder — Image by Author

It can be seen that the same digits tend to cluster themselves in the latent space. Another important thing to note is that there are parts of the latent space that doesn't correspond to any data point. Using those as inputs to the encoder will result in an output that doesn’t look like any digit from the MNIST data. This is what we mean by that the latent space is not regularized. Such a latent space only has a few regions/cluster that has the generative capability, which means that sampling any point in the latent space that belongs within a

cluster will generate a variation of the data that the cluster belongs to. But the entire latent space does not have the generative capability. The regions which do not belong to any cluster will generate garbage output. Once the network is trained, and the training data is removed, we have no way of knowing if the output generated by the decoder from a randomly sampled latent vector is valid or not. Hence AE is mainly used for compression.

> For valid inputs, the AE is able to compress them to fewer bits essentially getting rid of the redundancy (Encoder) but due to non-regularized latent space AE, the decoder can not be used to generate valid input data from latent from vectors sampled from the latent space.

Why use AE for compression?

The latent space of a linear autoencoder strongly resembles the eigenspace achieved during the principal component analysis of the data. A linear autoencoder with input space dimension and latent space dimensions set to result will span the same vector space as spanned by the first eigenvectors of PCA. If AE is similar to PCA, why use AE? The power of AE comes with its non-linearity. Adding non-linearity (such as nonlinear activation functions, and more hidden layers) makes AE capable to learn rather powerful representations of the input data in lower dimensions with much less information loss.

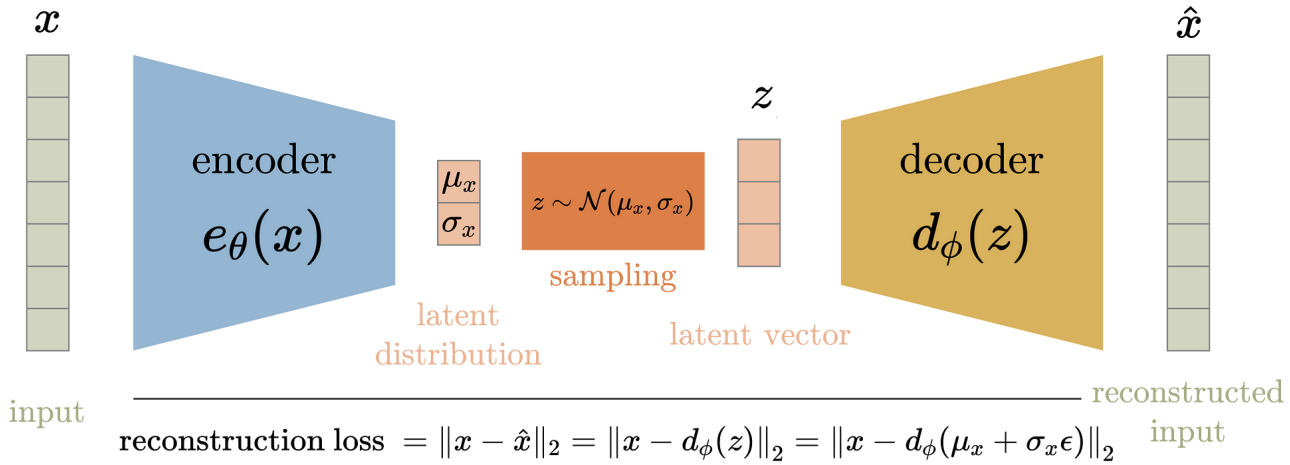
Variational AutoEncoders

What is it?

Variational autoencoder addresses the issue of non-regularized latent space in autoencoder and provides the generative capability to the entire space. The encoder in the AE outputs latent vectors. Instead of outputting the vectors in the latent space, the encoder of VAE outputs parameters of a pre-defined distribution in the latent space for every input. The VAE then imposes a constraint on this latent distribution forcing it to be a normal distribution. This constraint makes sure that the latent space is regularized.

Block Diagram

The block diagram of VAE can be seen below. During training, the input data x is fed to the encoder function . Just like AE, the input is passed through a series of layers (parameterized by the variable theta) reducing its dimensions to achieve a compressed latent vector . However, the latent vector is not the output of the encoder. Instead, the encoder outputs the mean and the standard deviation for each latent variable. The latent vector is then sampled from this mean and standard deviation which is then fed to the decoder to reconstruct the input. The decoder in the VAE works similarly to the one in AE.



$$\text{reconstruction loss} = \|x - \hat{x}\|_2 = \|x - d_\phi(z)\|_2 = \|x - d_\phi(\mu_x + \sigma_x \epsilon)\|_2$$

$$\mu_x, \sigma_x = e_\theta(x), \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$similarity\ loss = KL\ Divergence = D_{KL}(\mathcal{N}(\mu_x, \sigma_x) \parallel \mathcal{N}(\mathbf{0}, \mathbf{I}))$$

$$loss = reconstruction\ loss + similarity\ loss$$

Loss function

The loss function is defined by the VAE objectives. VAE has two objectives

1. Reconstruct the input
2. Latent space should be normally distributed

Hence the training loss of VAE is defined as the sum of these the and the . The reconstruction error, just like in AE, is the mean squared loss of the input and reconstructed output. The similarity loss is the KL divergence between the latent space distribution and standard gaussian (zero mean and unit variance). The loss function is then the sum of these two losses.

$$\text{reconstruction loss} = \|x - \hat{x}\|_2 = \|x - d_\phi(z)\|_2 = \|x - d_\phi(\mu_x + \sigma_x \epsilon)\|_2$$

$$\mu_x, \sigma_x = e_\theta(x), \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$similarity\ loss = KL\ Divergence = D_{KL}(\mathcal{N}(\mu_x, \sigma_x) \parallel \mathcal{N}(\mathbf{0}, \mathbf{I}))$$

$$loss = reconstruction\ loss + similarity\ loss$$

Variational Autoencoder loss function — Image by Author

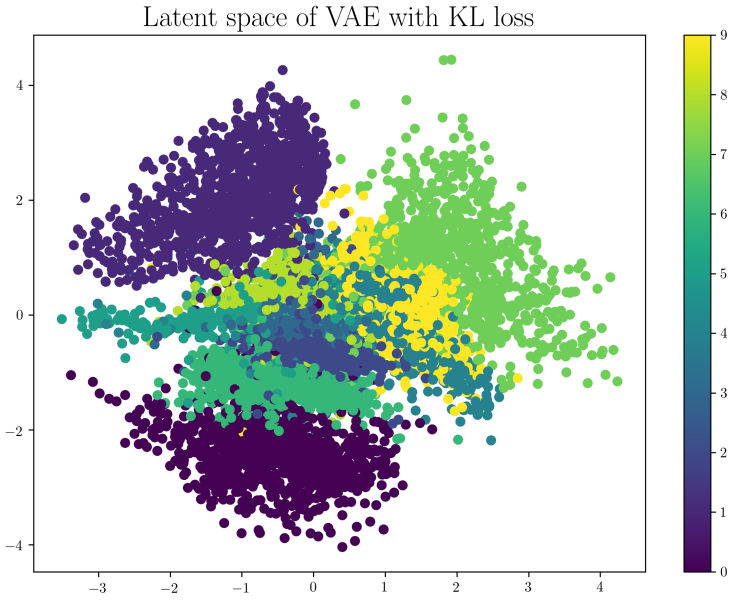
As mentioned before, the latent vector is sampled from the encoder-generated distribution before feeding it to the decoder. This random sampling makes it difficult for backpropagation to happen for the encoder since we can't trace back errors due to this random sampling. Hence we use a reparameterization trick to model the sampling process which makes it possible for the errors to propagate through the network. The latent vector z is represented as a function of the encoder's output.

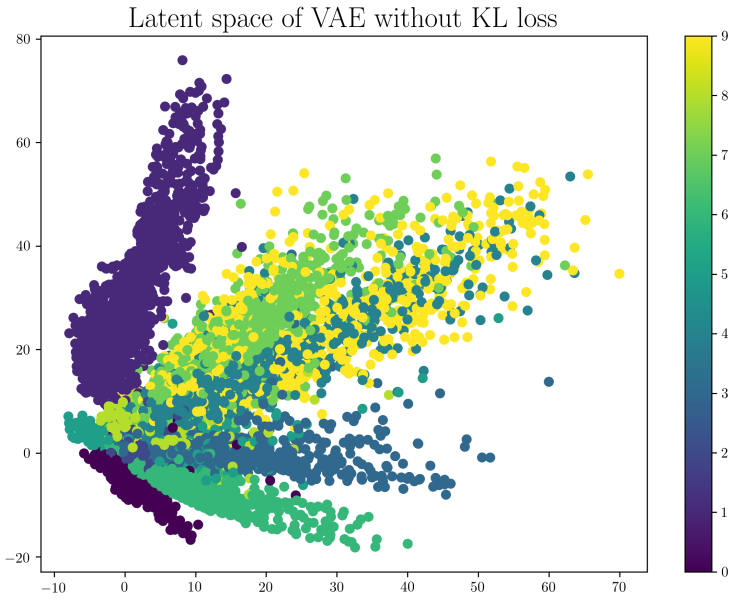
$$z = \mu_x + \sigma_x \epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

The reparameterization trick is used to represent the latent vector z as a function of the encoder's output.

Latent space visualization

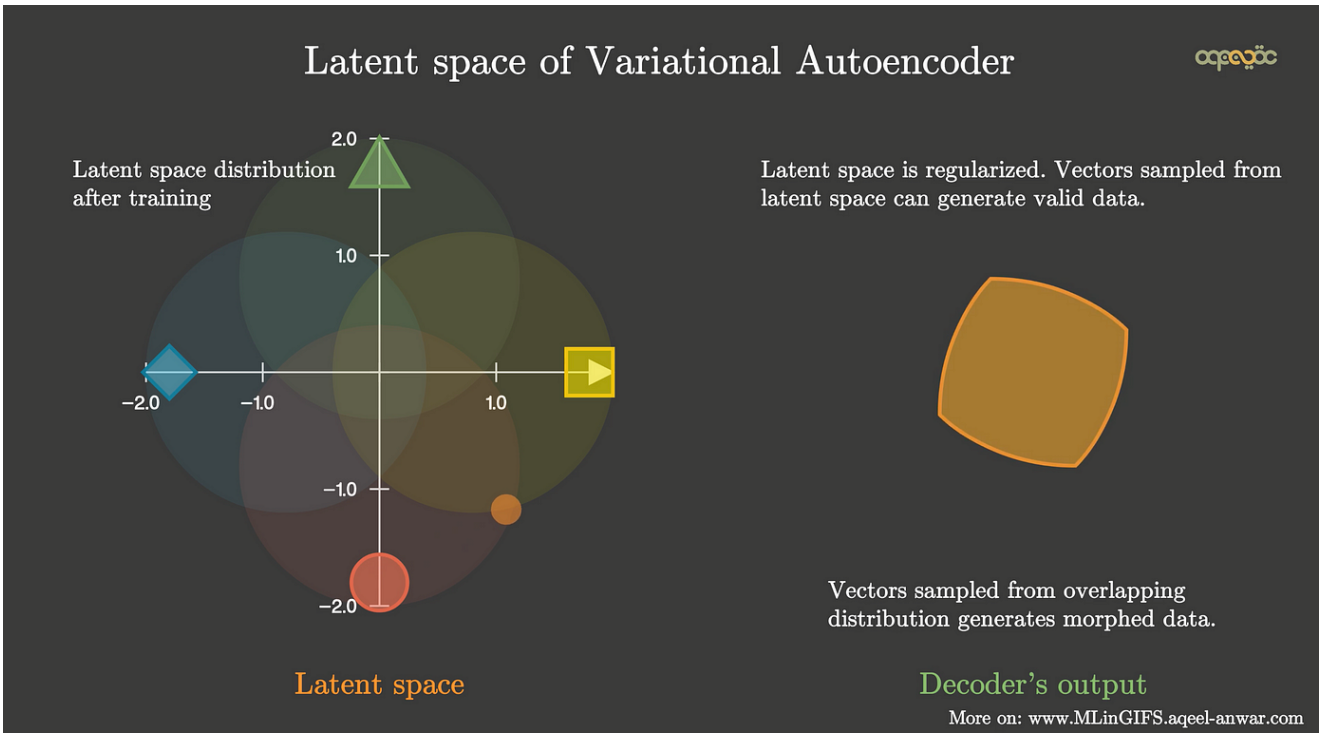
The training tries to find a balance between the two losses and ends up with a latent space distribution that looks like the unit norm with clusters grouping similar input data points. The unit norm condition makes sure that the latent space is evenly spread out and does not have significant gaps between clusters. In fact, the clusters of similar-looking data inputs usually overlap in some regions. Below is an example of the latent space generated by training the network on the same MNIST dataset, as was used to visualize the latent space of AE. Note how there are no gaps between clusters and the space resembles the distribution of unit norm.





Latent space of Variational Autoencoder — Image by Author

An important thing to note is that when the latent vector is sampled from the regions with overlapping clusters, we get morphed data. We get a smooth transition between the decoder's output when we sample the latent space moving from one cluster to the other.



Latent space of VAE is regularized — Image by Author

Summary

This article covered the understanding of Autoencoder (AE) and variational Autoencoder (VAE) which are mainly used for data compression and data generation respectively. VAE addresses the issue of non-regularized latent space of AE which makes it able to generate data from randomly sampled vectors from the latent space. The key summary points of AE and VAE are

Autoencoder (AE)

· Used to generate a compressed transformation of input in a latent space

· The latent variable is not regularized

· Picking a random latent variable will generate garbage output

· The latent variable has a discontinuity

· Latent variable is deterministic values

· The latent space lacks the generative capability

Variational Autoencoder (VAE)

· Enforces conditions on the latent variable to be the unit norm

· The latent variable in the compressed form is mean and variance

· The latent variable is smooth and continuous

· A random value of latent variable generates meaningful output at the decoder

· The input of the decoder is stochastic and is sampled from a gaussian with mean and variance of the output of the encoder.

· Regularized latent space

· The latent space has generative capabilities.

If this article was helpful to you or you want to learn more about Machine Learning and Data Science, follow , or connect with me on

Join Medium with my referral link - Aqeel Anwar

As a Medium member, a portion of your membership fee goes to writers you read, and you get full access to every story…

aqeel-anwar.medium.com