DAT300 Generative models

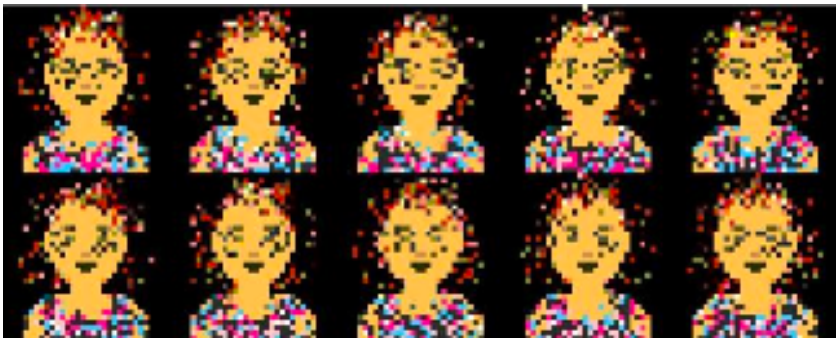
Jorid Holmen
Applied Robotics, Norwegian University of Life Sciences

This chapter is about the generative modeling framework. We have a dataset of observations X and assume that the observations have been generated according to some unknown distribution, P_{data}. A generative model, P_{model}, tries to mimic P_{data}. If we achieve this goal, we can sample from P_{model} to generate observations that appear to have been drawn from P_{data}.

P_{model} is impressive if it can generate examples that appear to have been drawn from P_{data}, and if it can generate examples that are suitably different from the observations in X. In other words, the model should not simply reproduce things it has already seen.

1 Naive Bayes

Naive Bayes is part of a family of generative learning algorithms, based on Bayes' theorem. It is named "naive" because it assumes that each feature x_j is independent of every other feature x_k. In an image the neighbouring pixels are highly related, but Naive Bayes assumes the pixels are independent, and therefore often fail at image generation. It also struggles with high-dimensional data (like 1,024 pixels) and can't capture complex or non-linear relationships between pixels. More advanced models are better suited for handling the problem.



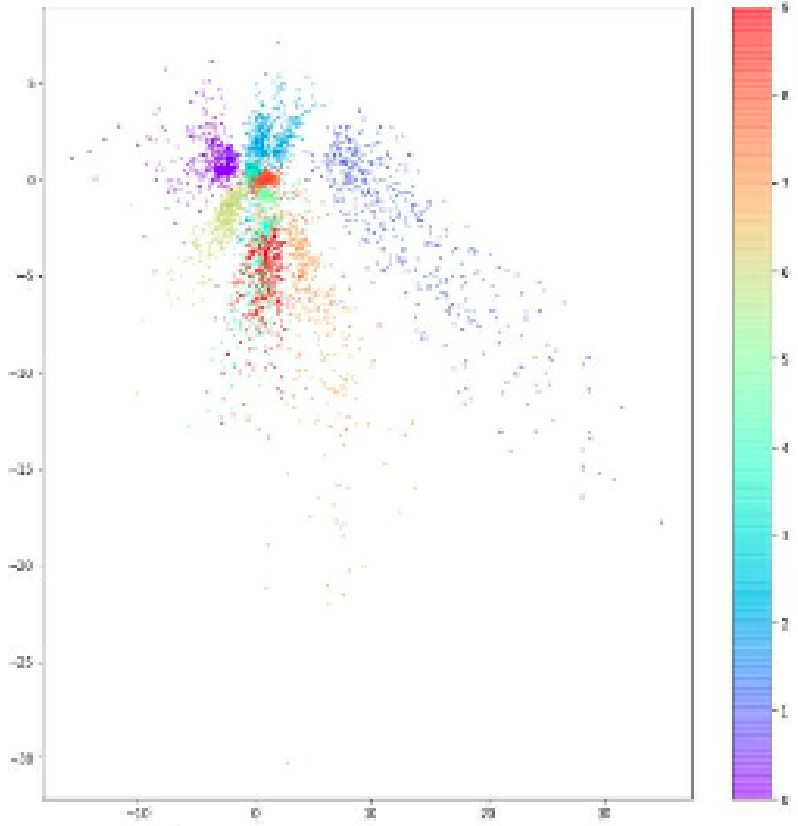
The equations for Naive Bayes are

$$p(x_j|x_k) = p(x_j) \tag{1}$$

$$p(x) = \prod_{k=1}^{K} p(x_k) \tag{2}$$

2 Autoencoders

An autoencoder is a type of artificial neural network used for unsupervised learning, and has a lot of similarities to PCA from machine learning. Its primary goal is to learn a compressed representation of the input data, and it does this by encoding the data into a **lower-dimensional latent space** and then decoding it back to its original form. The latent space could look like this, where each colour represents a different class:



The entire process is meant to capture the most relevant features of the data. Standard autoencoders are trained to minimize the reconstruction error between the input data and their output, and not necessarily to generate new images.

The process of encoding and decoding is **deterministic**, meaning if you input the same data point multiple times, you will get the same encoded representation and reconstructed output every time.

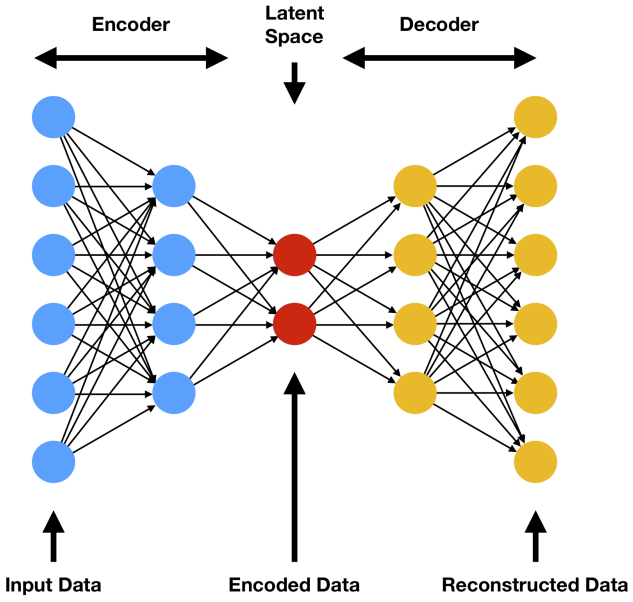
2.1 The autoencoder structure

The autoencoder structure is divided into an encoder, a latent space and a decoder.

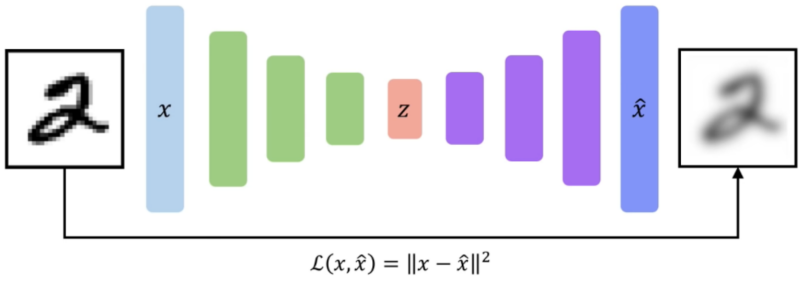
- The **encoder** part of the network compresses the input into a latent-space representation. It encodes the input data as an internal fixed-size representation in reduced dimensionality. In other words, an encoder network compresses high-dimensional input into a lower-dimensional representation vector.

- The **latent space** is the compressed representation of the input data. It holds key features necessary to reconstruct the input data.
- The **decoder** is the part of the network that reconstructs the input data from the internal representation. It maps the encoded data back to the original space. In other words, a decoder network decompresses a given representation vector back to the original domain.

The parameters of an autoencoder consists of weights and biases of the encoder and the decoder networks.



The loss function minimizes the reconstruction error between the input and the output.



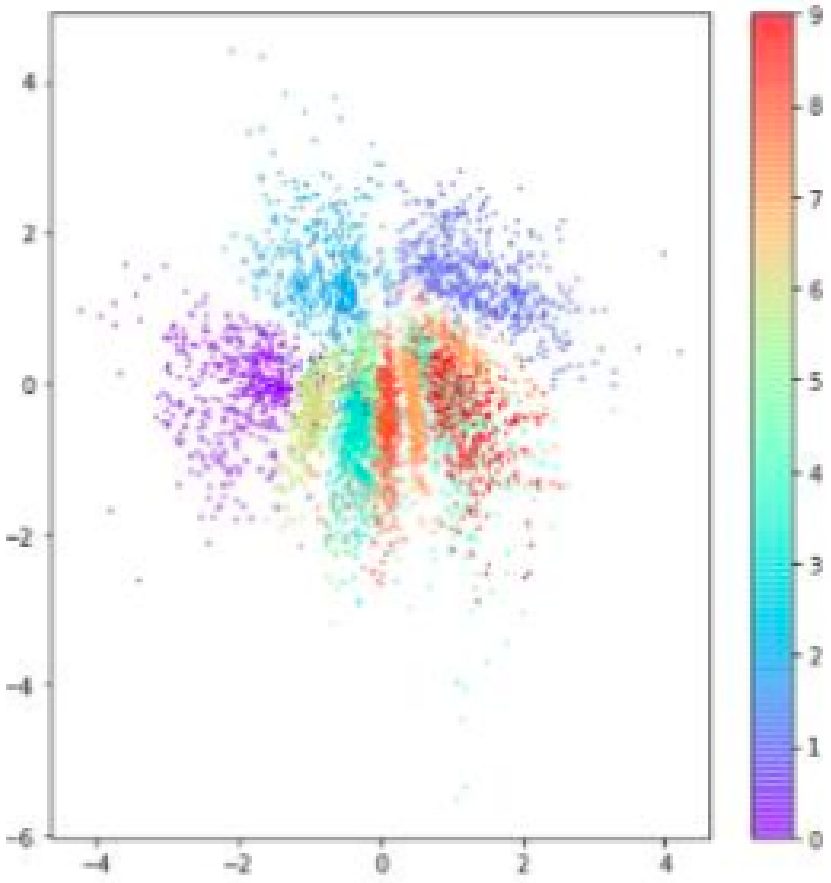
Examples of applications:

- Autoencoders are used to reduce the size of data for faster storage and transmission.

- Autoencoders can learn to remove noise from corrupted data (e.g., denoising images).
- Autoencoders can identify unusual or rare patterns (anomalies) in data because they will struggle to reconstruct patterns that are not similar to the training data
- They help extract meaningful features from data, often used as a pre-training step in more complex models

3 Variational autoencoders

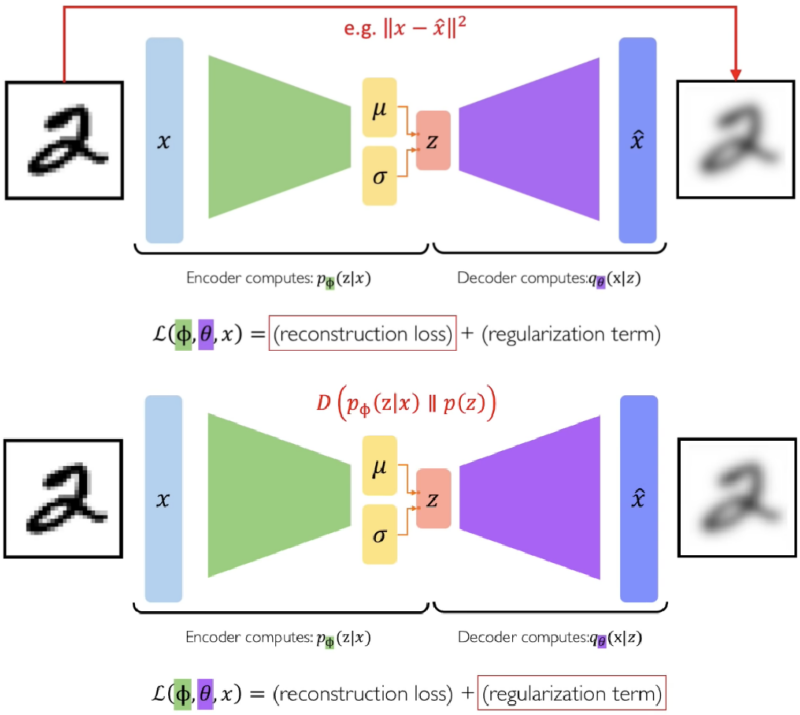
Variational autoencoders (VAEs) are a type of autoencoders that introduces probabilistic reasoning and optimization techniques from variational inference to create a generative model. While standard autoencoders are trained to minimize the reconstruction error between the input data and their output, variational autoencoders aim to generate new samples that looks like they could have been produced by the input data. A variational autoencoder can be defined as being an autoencoder whose training is regularised to avoid overfitting and ensure that the latent space has good properties that enable a generative process. The biggest difference from normal autoencoders is that variational autoencoders seeks to learn the distribution of the input data in the latent space.



Variational autoencoders are stochastic, meaning the encoding process incorporates radnomness. The encoder outputs parameters that define a probability distribution, and the latent distribution is sampled from this distribution.

- The **encoder** maps the input data to a latent-space, like in traditional autoencoders. However, instead of encoding the input as a single fixed point in the latent space, the VAE encoder outputs parameters of a probability distribution (usually Gaussian) over the latent space.
- During **sampling** a sample is drawn from this distribution to provide a randomized latent space representation.
- The **decoder** generates a reconstruction of the input by passing through the sampled latent points.

The reconstruction loss is mean squared error, and encourages the decoded samples to match the original input.



σ: the standard deviation vector
μ: the mean vector
z: the latent space

We seek to approximate the true posterior distribution $p(z|x)$ (probability of z when you have the observation x) with a variational distribution $q_\phi(z|x)$ (probability of x if you have z). The goal is to minimize the divergence between these distributions. The Kullback-Leibler (KL) divergence measures the difference between the two probability distributions, in addition to regularizing and encourages the latent variables to be close to a standard Gaussian distribution.

$$D_{\mathrm{KL}}(q(z|x) \parallel p(z)) = -\frac{1}{2} \sum \left(1 + \log(\sigma_i^2) - \mu_i^2 - \sigma_i^2\right) \tag{3}$$

ϕ and θ are optimized to maximize the training obejctive techniques, like stochastic gradient descent or its variants.

Similar to autoencoders, variational autoencoders have weights and biases in their encoder and decoder network as their parameters. The encoder in variational autoencoders outputs parameters for the mean and variance of a probability distribution rather than a single point in the latent space. This

means that for each dimension in the latent space, a variational autoencoder has two sets of parameters: one for the mean and one for the variance.

3.1 Variational autoencoders - Reparameterization

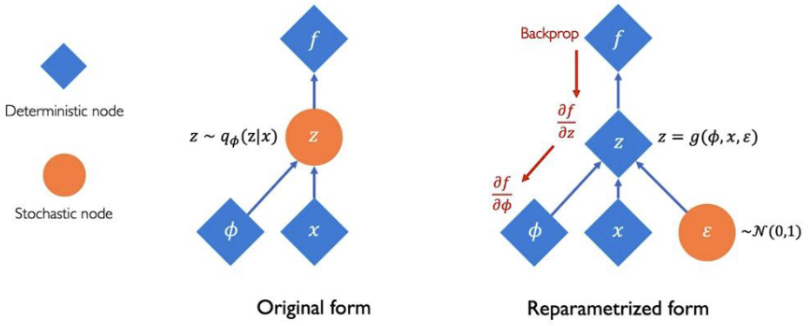
To enable backpropagation through the stochastic nodes in the network, the reparameterization trick is used. This involves sampling from a simple, fixed distribution (like standard gaussian) and then transforming the sample to obtain a sample from the desired distribution. This is why variational autoencoders are considered a probabilistic model. We use the log variance instead of the variance for numerical stability and to ensure the variance is always positive when we convert it back using an exponential function.

If the latent space is of dimensions d, the encoder will produce d means and d log variance parameters. Instead of sampling directly from $q_\phi(z|x)$, a sample ϵ is drawn from a fixed distribution, and the sample is transformed using the mean, μ, and the standard deviation, σ predicted by the decoder:

$$z = \mu + \sigma \odot \epsilon \tag{4}$$

$$\epsilon \sim \mathcal{N}(0, 1) \tag{5}$$

This makes the sampling operation differentiable, allowing gradient to flow through backpropagation.

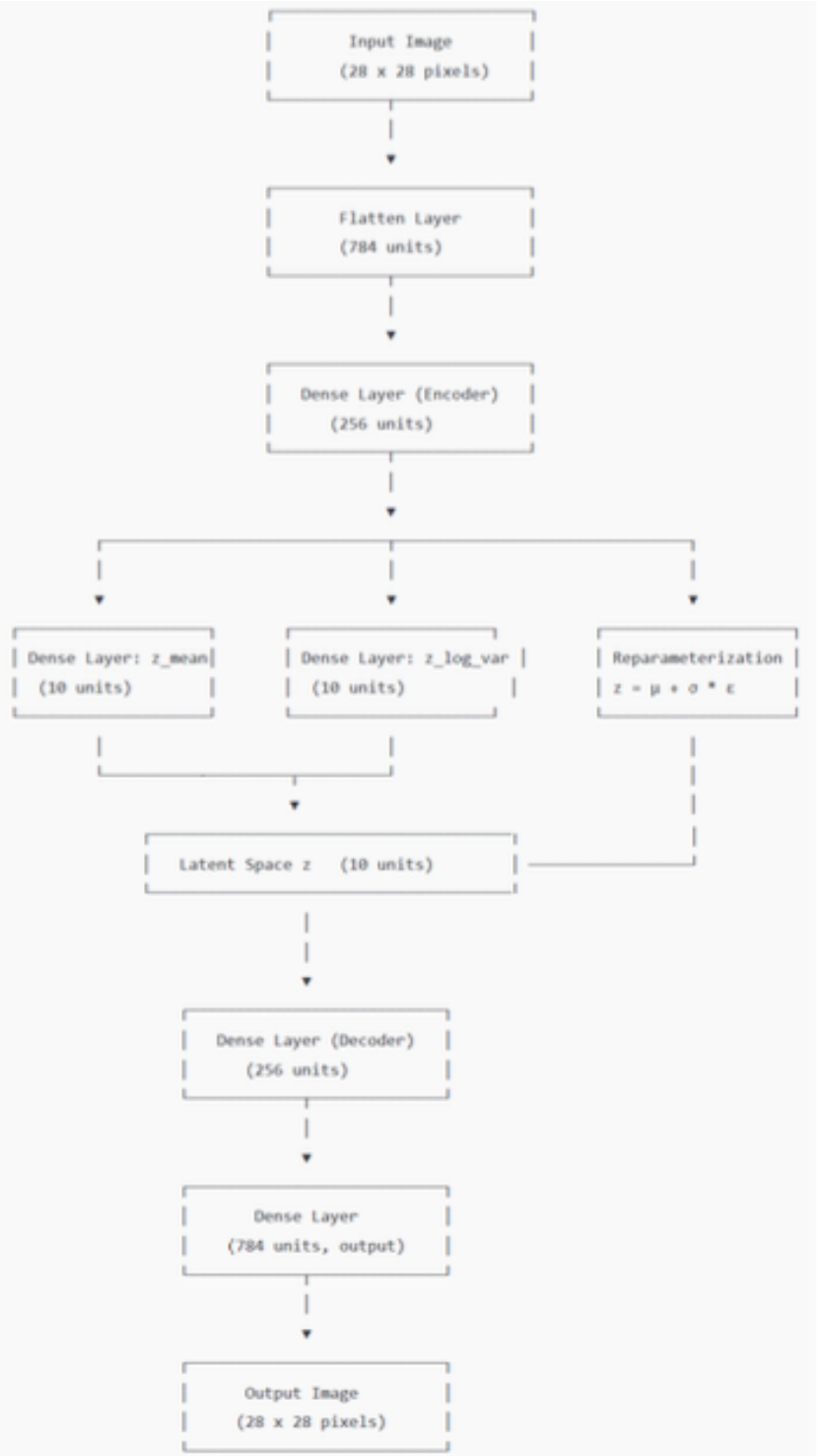


4 Mathematics in Autoencoders

In the following image there is a variational autencoder. This is how we calculate the number of trainable paramters:

- $784 * 256 + 256 = 200960$
- $256 * 10 + 10 = 2570$
- $256 * 10 + 10 = 2570$

- $10 * 256 + 256 = 2816$
- $256 * 784 + 784 = 201488$
- Total trainable parameters: 200960+2570+2570+2816+201488 = 410404


Input Image
(28 x 28 pixels)
Flatten Layer
(784 units)
Dense Layer (Encoder)
(256 units)
Dense Layer: z_mean
(10 units)
Dense Layer: z_log_var
(10 units)
Reparameterization
z = μ + σ * ε
Latent Space z (10 units)
Dense Layer (Decoder)
(256 units)
Dense Layer
(784 units, output)
Output Image
(28 x 28 pixels)

5 Usefull pages and videos

What are Autoencoders - IBM technology

Variational autoencoders - Arxiv Insights

Simple Explenation of AutoEncoders - WelcomeAIOverlords

What is an Autoencoder? - Two Minute Papers

Variational Autoencoders - EXPLAINED! - CodeEmporium

Reparameterization Trick - Tutorials Collection