# Autoencoders (AE) — A Smart Way to Process Your Data Using Unsupervised Neural Networks

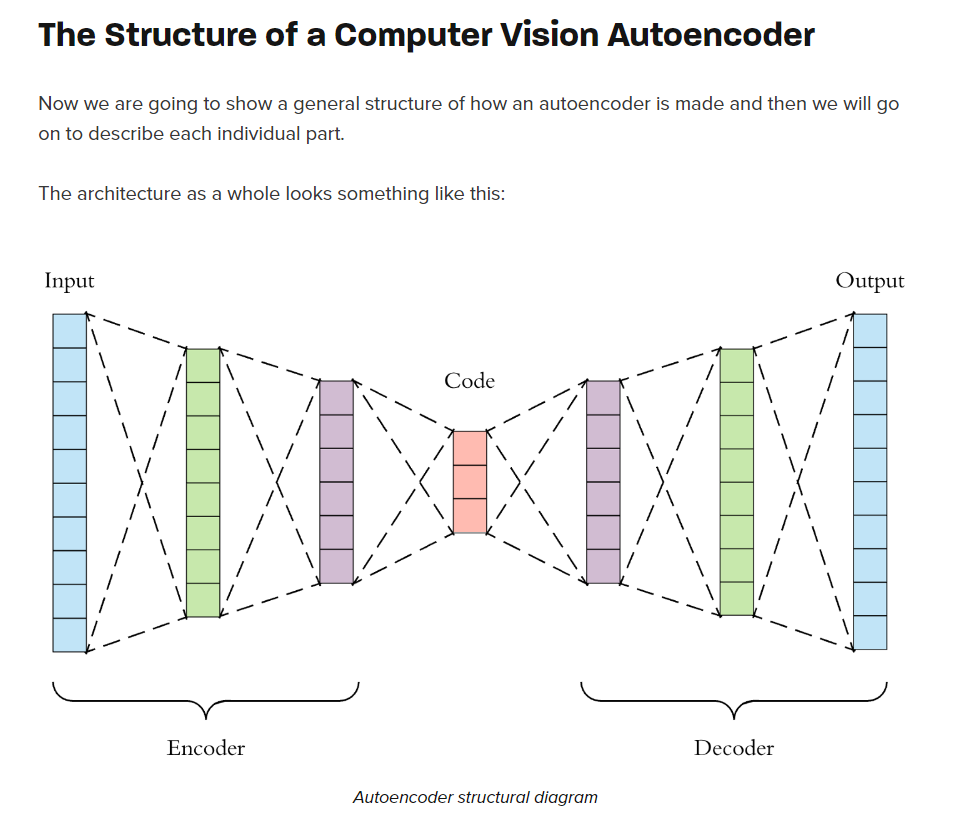
**Intro**

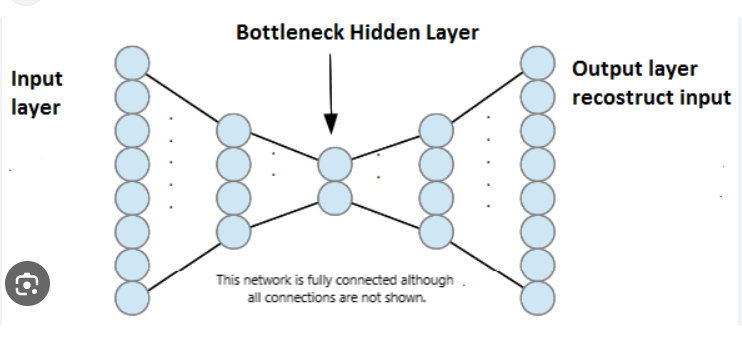
Autoencoders present an efficient way to learn a representation of your data that focuses on the signal, not the noise. You can use them for a variety of tasks such as:

* Dimensionality reduction
* Feature extraction
* Denoising of data/images
* Imputing missing data
* [**Undercomplete Autoencoder (AE)**](https://towardsdatascience.com/autoencoders-ae-a-smart-way-to-process-your-data-using-unsupervised-neural-networks-9661f93a8509)— the most basic and widely used type, frequently referred to as an Autoencoder
* **Sparse Autoencoder (SAE)** — uses sparsity to create an information bottleneck
* [**Denoising Autoencoder (DAE)**](https://towardsdatascience.com/denoising-autoencoders-dae-how-to-use-neural-networks-to-clean-up-your-data-cd9c19bc6915)— designed to remove noise from data or images
* [**Variational Autoencoder (VAE)**](https://towardsdatascience.com/vae-variational-autoencoders-how-to-employ-neural-networks-to-generate-new-images-bdeb216ed2c0)— encodes information onto a distribution, enabling us to use it for new data generation

While we often use Neural Networks in a **supervised** manner with labelled training data, we can also use them in an **unsupervised or self-supervised way**, e.g., by employing **Autoencoders**.



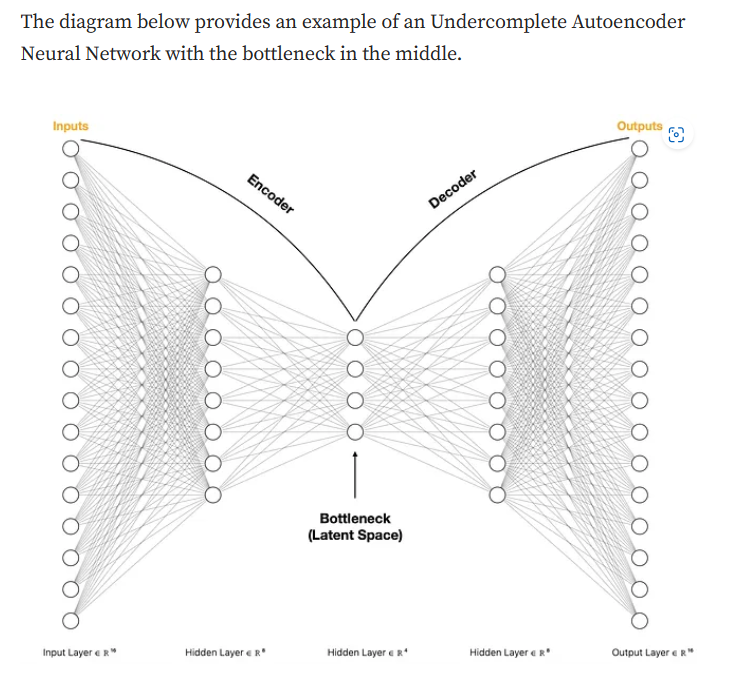




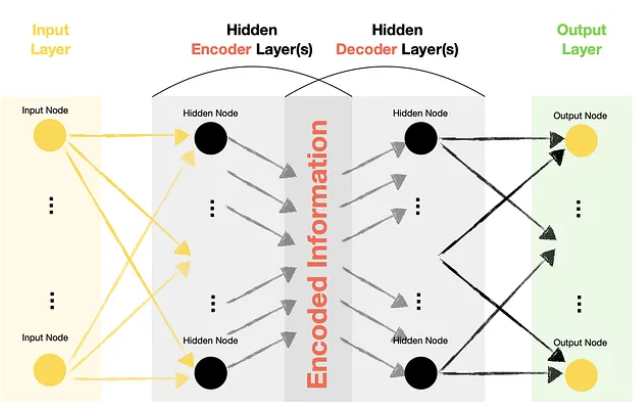
# How are the Autoencoders constructed, and how do they work?

The purpose of an autoencoder is to encode **important** information efficiently. A common approach to achieve that is by creating a bottleneck, which forces the model to preserve what’s essential and discard unimportant bits.

**Autoencoder can distinguish between what’s essential by simultaneously training an encoder and decoder**, with the goal of the decoder being the recreation of original data from encoded representation.



Let’s start by taking a high-level view displayed in the below diagram and review each of the parts:



* Autoencoders have Input, Hidden and Output layers similar to that of other types of Neural Networks.
* Hidden layers of Autoencoders contain two significant parts: **Encoder**and **Decoder**.
* Output nodes within an Autoencoder match the input nodes. Hence, the Autoencoder Neural Network tries to recreate the same feature values that it receives in the Input layer.
* Since we are trying to recreate (predict) features themselves, we do not require labelled target data. Hence, we can refer to Autoencoders as **Unsupervised** models, although some literature refers to them as **Self-Supervised** models.

## **Types of Autoencoders**

The relationship between the number of nodes in each layer determines the type of an Autoencoder.E.g.:

* **Undercomplete Autoencoder** (the focus of this article) — has fewer nodes (dimensions) in the middle compared to Input and Output layers. In such setups, we tend to call the middle layer a “bottleneck.”
* **Overcomplete Autoencoder** — has more nodes (dimensions) in the middle compared to Input and Output layers.

## What is the point?

The critical question is, why would we want to pass data through the Neural Network to get to the same output values that we fed into the network as inputs?

In the case of Undercomplete Autoencoders, we are squeezing the information into fewer dimensions (hence the bottleneck) while trying to ensure that we can still get back to the original values. **Therefore, we are creating a custom function that compresses the data, which is a way to reduce the dimensionality and extract meaningful information.**

**Important point: After training the Undercomplete Autoencoder, we typically discard the Decoder and only use the Encoder part.**

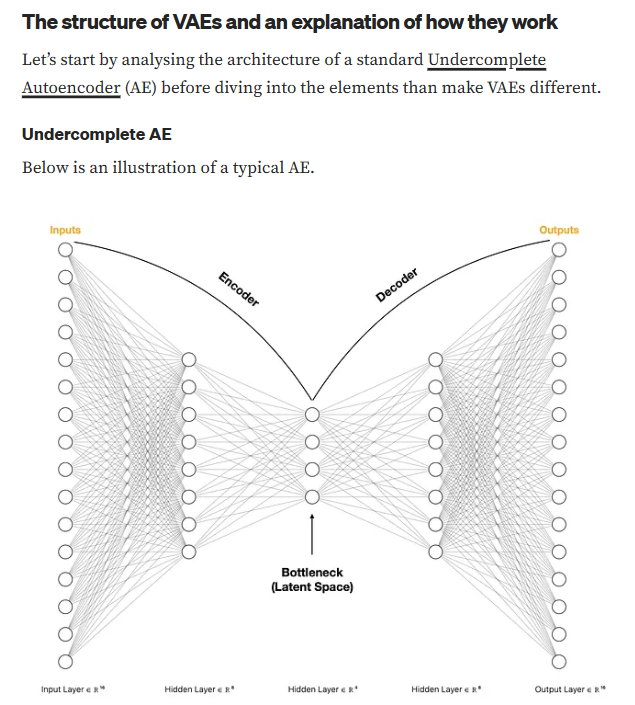
# VAE: Variational Autoencoders — How to Employ Neural Networks to Generate New Images

**Variational Autoencoders (VAE)**, which fall into a broader group of **Deep Generative Models** alongside the famous **GANs (Generative Adversarial Networks)**.

# Encodes information onto a distribution, enabling us to use it for new data generation.

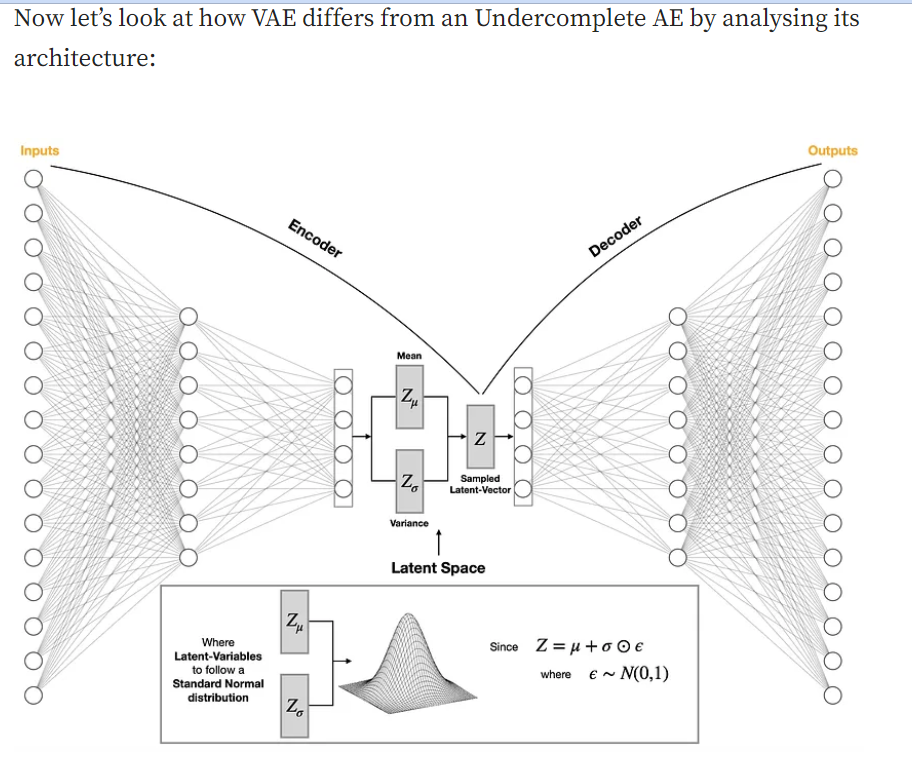
# Unlike GAN, VAE uses an Autoencoder architecture instead of a pair of Generator-Discriminator networks. So, the ideas used in VAEs should be relatively straightforward to understand, especially if you have used Autoencoders in the past.

The inputs are mapped onto a Normal distribution called Z latent space and then we use this space to make predictions. In other words, when we sample a point from such latent space, we generate new data **closely resembling** the training data.

****

The goal of an Undercomplete AE is to efficiently **encode information** from input data into a **lower-dimensional latent space (bottleneck)**. We achieve this objective by ensuring that the inputs can be recreated with minimal loss using a**decoder**.

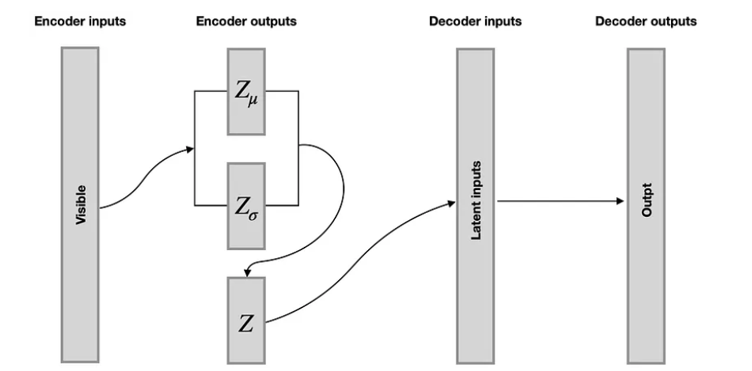
Note that during training, we pass the same set of data into input and output layers as we attempt to discover the parameter values for an “optimal” latent space.



We notice that VAE’s latent space is not made up of point vectors (individual nodes). Instead, the inputs are mapped onto a Normal distribution, where Zμ and Zσ are the mean and variance, the parameters learned during model training.

**Normal distributions have key characteristics that are easy to spot in graphs:**

* The mean, median and mode are exactly the same.
* The distribution is symmetric about the mean—half the values fall below the mean and half above the mean.
* The distribution can be described by two values: the mean and the standard deviation.
* 68 % of the data is between Zμ + Zσ and Zμ – Zσ
* 95% of the data is between Zμ + 2Zσ and Zμ – 2Zσ

****

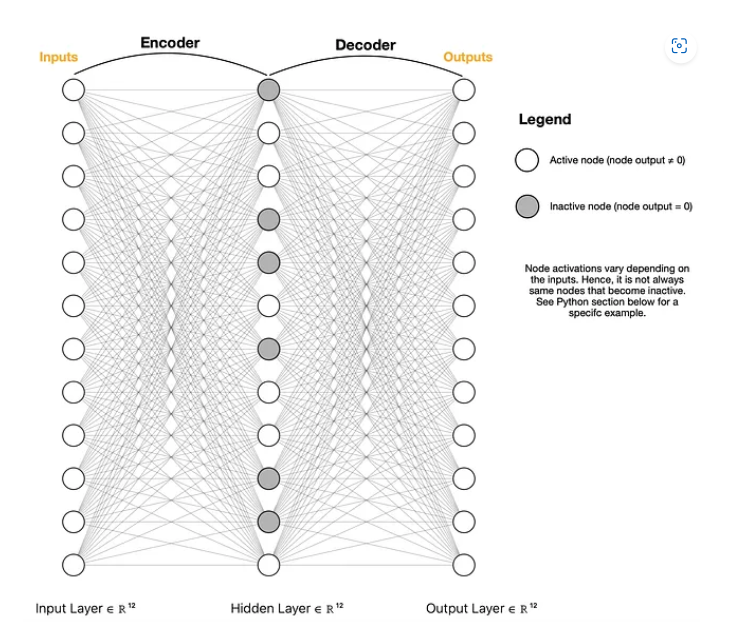
In the case of Variational Autoencoders, we have mapped data as distributions and regularized the latent space, which gives us the **“gradient”** or **“smooth transition”** between distributions. Hence, when we sample a point from such latent space, we generate new data **closely resembling** the training data.

# Sparse Autoencoder Neural Networks — How to Utilise Sparsity for Robust Information Encoding

**Uses sparsity to create an information bottleneck.**

The goal of an SAE is the same as an Undercomplete AE, but it achieves it differently. Instead of (or in addition to) relying on fewer neurons, **SAE uses regularization to enforce sparsity**.

By sparsity, we mean that fewer neurons can be activated at the same time, creating an information bottleneck similar to that of Undercomplete AE. See the below illustration:



Note that the above drawing represents only one of many potential setups. For example, you can have more than one hidden layer or limit the number of neurons in the middle (bottleneck) layer.

**The key difference between SAE and AE is regularization.**

**Denoising Autoencoders (DAE) — How To Use Neural Networks to Clean Up Your Data**

[**Denoising Autoencoder (DAE)**](https://towardsdatascience.com/denoising-autoencoders-dae-how-to-use-neural-networks-to-clean-up-your-data-cd9c19bc6915)— designed to remove noise from data or images.

The purpose of a DAE is to remove noise. You can also think of it as a **customized** denoising algorithm tuned to your data.

Note the emphasis on the word **customized**. Given that we train a DAE on a specific set of data, it will be optimized to remove noise from similar data. For example, if we train it to remove noise from a collection of images, it will work well on similar images but will not be suitable for cleaning text data.

Unlike Undercomplete AE, we may use the same or higher number of neurons within the hidden layer, making the DAE **overcomplete**.

The second difference comes from not using identical inputs and outputs. Instead, the outputs are the original data (e.g., images), while the inputs contain data with some added noise.