Autoencoder y Aprendizaje Profundo

Prof. Wílmer Pereira

Arquitecturas de Deep Learning

Hasta el año 2014, la mayoría de los modelos de *deep learning* supervisados, se usaban para tareas de clasificación y los pocos modelos generativos estaban basados en distribuciones de probabilidad conjunta

- Con el advenimiento de los ***autoencoders*** y las **redes adversarias generativas** fue posible desarrollar eficientes modelos profundos para generar imágenes.

Dall-e DeepDream DeepArt Prisma Midjourney Stable Diffusion

- Los modelos generativos no solo aplican a imágenes, también se usan para generar:

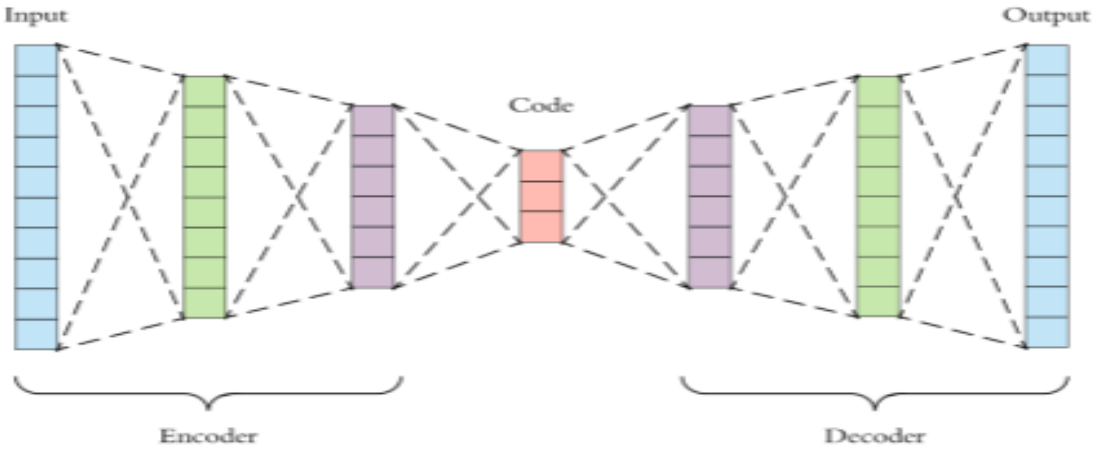
Texto:	ChatGPT (OpenAI)
	Bard (Google)
	Bing (Microsoft).
Video:	Gen1 y Gen2 (RunwayML)
	Make-A-Video (Meta Platform)
Código:	Codex (OpenAI) …

Teatro de Opera Espacial diseñado con Midjourney

Arquitectura básica de autoencoders

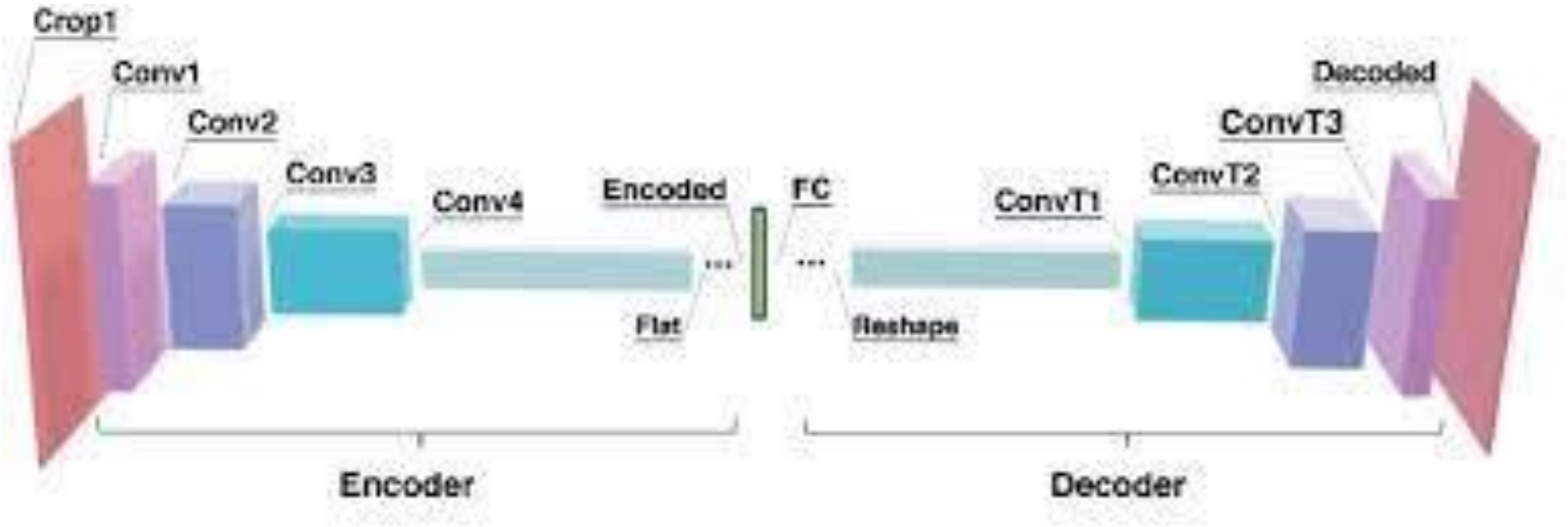
Es una red neuronal que utiliza un aprendizaje de características discriminates, es decir, aprende una representación condensada de un conjunto de datos. En ciertos contextos, es un **aprendizaje auto-supervisado**.

- Su aplicación tradicional es para reducir dimensionalidad o para aprender el **espacio latente** o ***code*** que representa al conjunto de datos de entrada. Eso es deseable porque hay muchos aspectos de la entrada no determinantes …

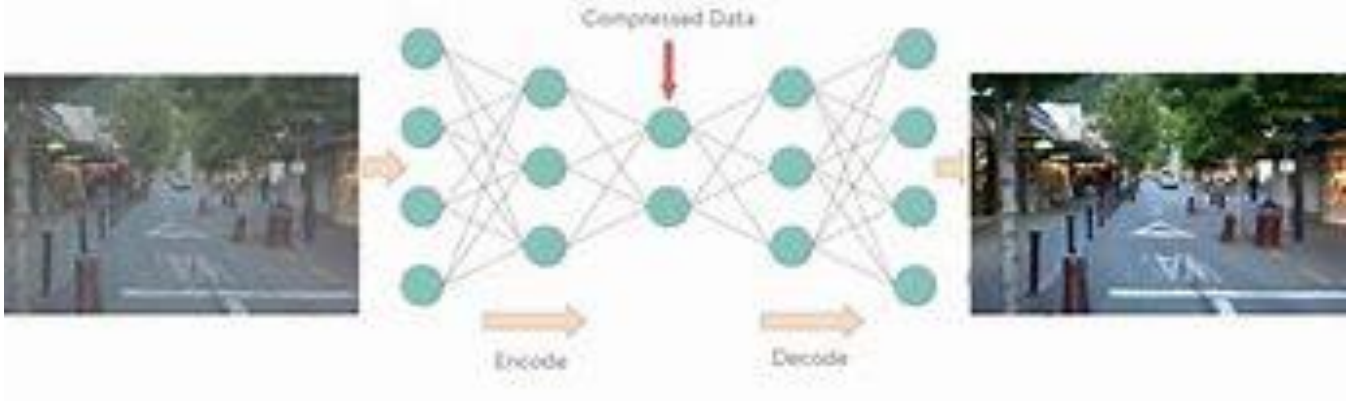


- También se utiliza en redes generativas y empilado en otras arquitecturas de redes neuronales profundas más potentes como los *transformers* o inmerso en redes neuronales convolucionales.

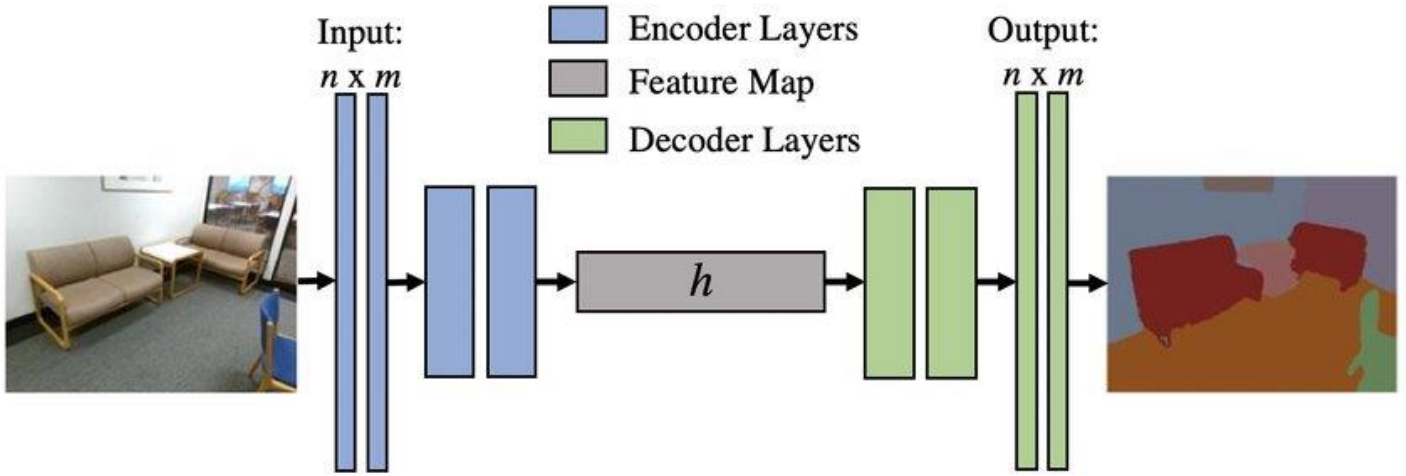
CNN y autoencoders

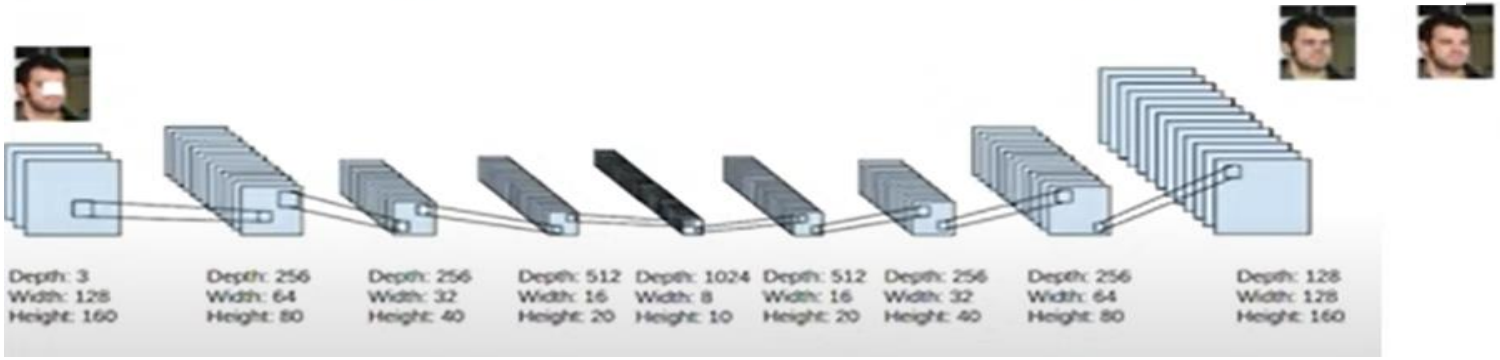


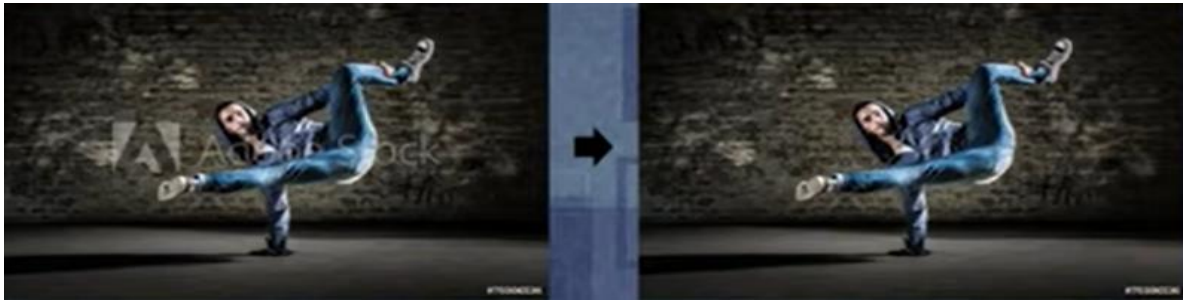
- Existen diversas variantes que ofrecen distintas funcionalidades como: eliminación de ruido en imágenes, segmentación de objetos, completación y eliminación de partes de una imagen, detección de anomalías, reconocimiento facial, …



Ejemplos con autoencoders







Características de un autoencoder

Un *autoencoder* normalmente no es recurrente y muy parecido a un perceptrón multicapas donde la capa de entrada y la de salida tienen la misma cantidad de neuronas

- El objetivo es reconstruir en la capa de salida, a partir de los datos de entrada, comunmente bajo una estrategia no supervisada

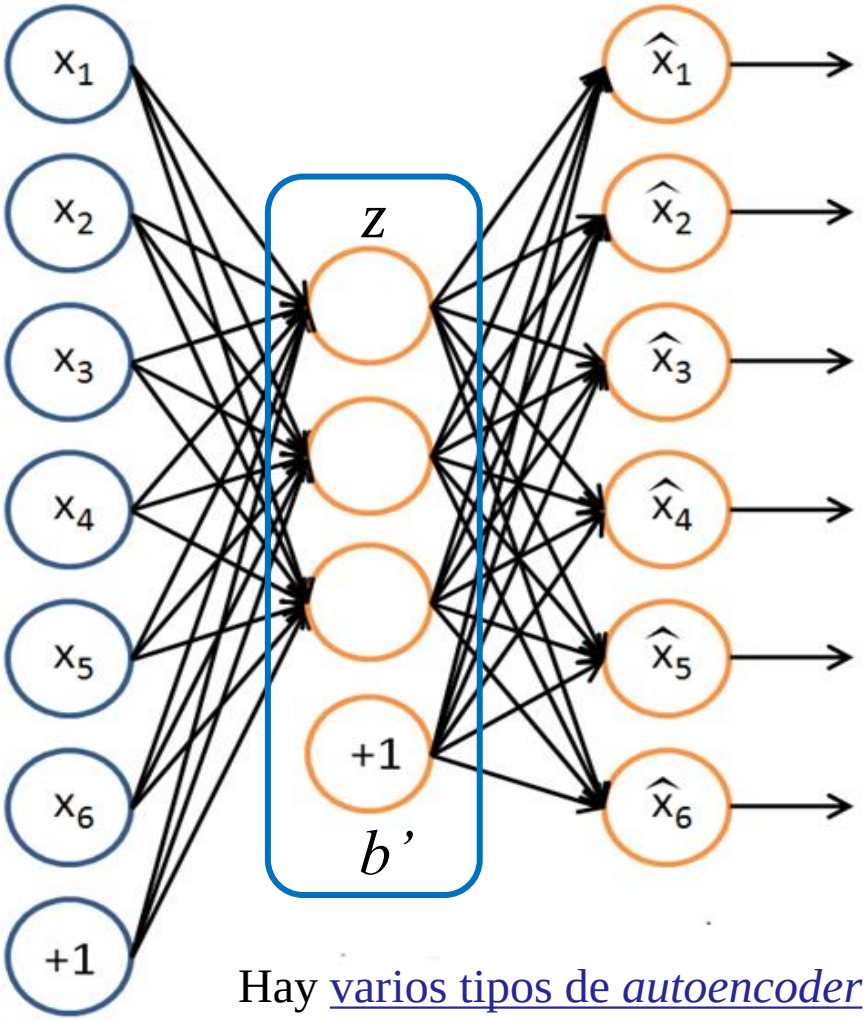
- Si la cantidad de neuronas del espacio latente es mayor que la capa de entrada o salida, potencialmente se puede aprender la función identidad lo cual no reviste mayor interés.

… sin embargo …

- Si se introducen restricciones con las funciones que se usan para evitar el *overfitting*: regularización L1 o L2, *dropout*. También es posible corregir con una distribución de probabilidad que mejor se adapte a la predicción de la buena respuesta con la divergencia de Kullback-Leiber

… es posible aprender ciertas características especiales …

Formulación teórica de un autoencoder simple



Codificar la entrada $x \in X$ para obtener $z \in F$ que es el espacio latente se obtiene como:

$$z = \sigma(Wx + b)$$

Decodificar el espacio latente para reconstruir $\hat{x}$ es:

$$\hat{x} = \sigma'(W'z + b')$$

Minimizar, por ejemplo, el error medio cuadrático es:

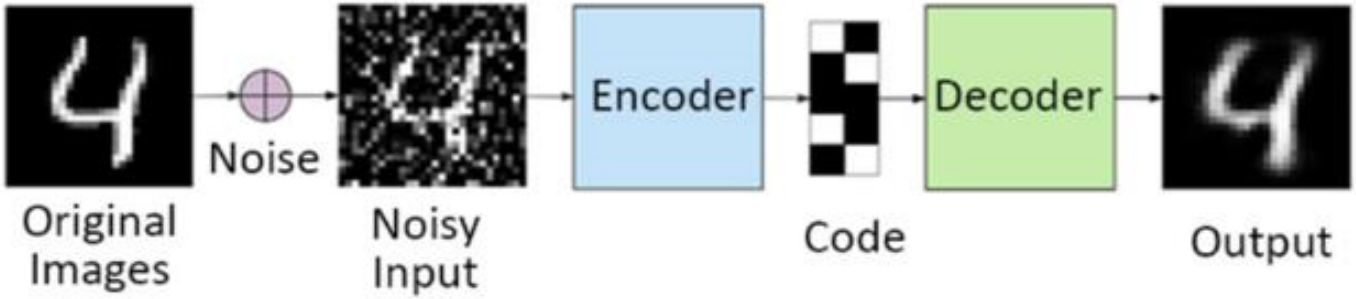
$$E(x, \hat{x}) = \frac{1}{2}(x - \hat{x})^2$$

Hay varios tipos de *autoencoder* con regularizaciones

- *Denoising*
- *Sparse*
- *Variational*

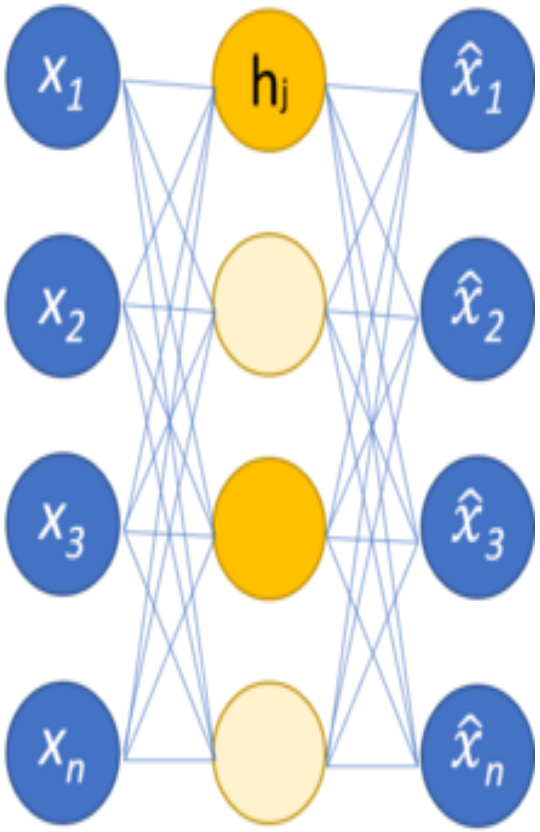
Denoising autoencoder

A partir de la entrada corrupta recupera los datos originales sin ruido. A la entrada se le introduce ruido con una distribución de probabilidad gausiana aleatoria μ_T donde $T: X \longrightarrow X$ es aplicada al cálculo de la función de error.



- La distribución de probabilidad para generar ruido sólo se aplica durante las fases de entrenamiento y prueba. En este caso, es aprendizaje supervisado
- Las hipótesis implícitas son:
  - El espacio latente contiene representaciones estables y robustas de la entrada que resiste a corrupciones.
  - Es imprescindible tener las características justas para los datos de entrada

Sparse autoencoder



Sólo las neuronas en amarillo intenso del espacio latente están activas

- Algunas neuronas se inactivan al estilo de *dropout*, dependiendo de la entrada. Así el espacio latente siempre tendrá menos neuronas que la entrada

- La inactivación de las neuronas se hace en los *k-sparse autoencoder* con una función:

$$f_k = (x_1, x_2, \dots\ x_n) = (x_1 b_1, \dots, x_n b_n)$$

donde $b_i = 1$ si $|x_i|$ está entre los primeros k sino $b_i = 0$

- Otra manera es agregar al cálculo del error un factor de regularización $s(\rho, \hat{\rho})$ de la divergencia de Kullback-Leiber relacionada con la entropía:

$$s(\rho, \hat{\rho}) = \rho\ log\frac{\rho}{\hat{\rho}} + (1 - \rho)\ log\frac{1-\rho}{1-\hat{\rho}}$$

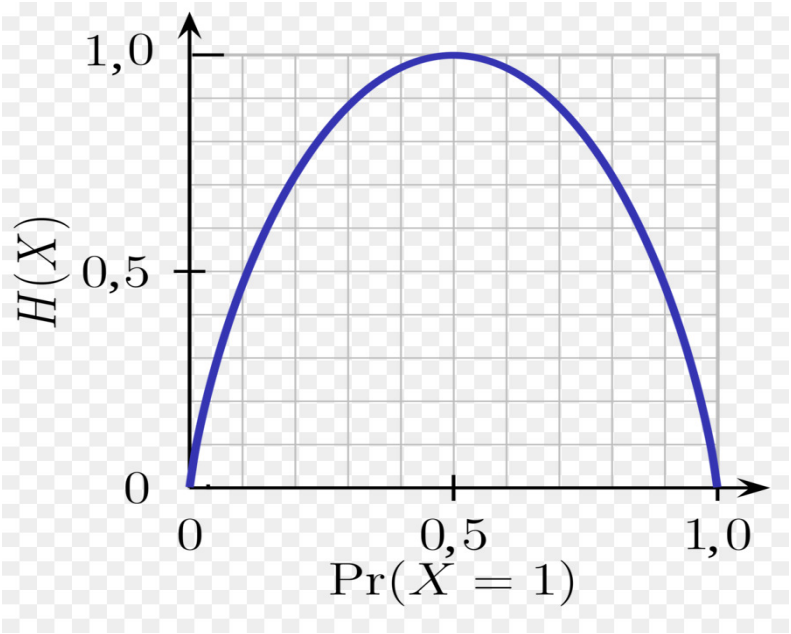
donde $\rho\ y\ \hat{\rho}$ son distribuciones de probabilidad de la entrada (a priori) y la salida (a posteriori) respectivamente … Ejemplo con extraterrestres

Entropía

- La entropía que se basa en la teoría de la información se inspira en la noción de la entropía en termodinámica. Para teoría de la información, sean v_1 distintos tipos de mensajes:

$$I\big(P(v_1), \ldots, P(v_n)\big) = \sum_{i=1}^{n} P(v_i) \log_2 \left(\frac{1}{P(v_i)}\right) = \sum_{i=1}^{n} -P(v_i) \log_2 (P(v_i))$$

Cuanto menos información se tiene, más alta es la entropía → 1 … por lo que se tiene muy poca información para predecir el comportamiento de los datos



Contractive autoencoder

- Este *autoencoder* agrega una regularización dependiente de la matriz jacobiana. El objetivo es hacer la fase de codificación menos sensible a pequeñas variaciones del conjunto de datos de entrenamiento si la entrada no está etiquetada

- Al igual que los otros *autoencoders* esto se logra sumando una regularización a la función que calcula el error que se está intentando minimizar

- El término que se suma a la función de error o pérdida es la norma de Frobenius:

$$\|J_h(X)\|_F^2 = \sum_{ij} \left(\frac{\partial h_j(X)}{\partial X_i}\right)^2$$

- La diferencia entre *denoising autoencoder* y *contractive autoencoder* es:

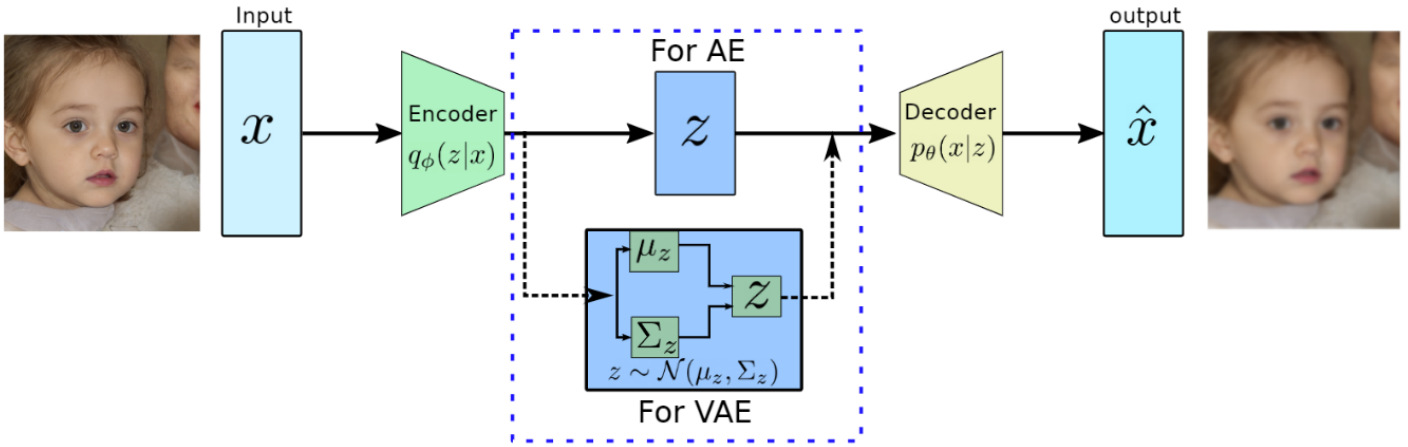
  Denoising autoencoder: hace una función de reconstrucción que resiste a pequeñas perturbaciones de la entrada

  Contractive autoencoder: hace una función extracción de características que resiste a pequeñas perturbaciones de la entrada

Variational autoencoder

Este realiza el ajuste de la minimización del error mediante métodos bayesianos, para deducir el valor de una variable aleatoria en función a otra variable aleatoria. En este caso, la distribución de probabilidad para el *decoder* minimizan la función de error

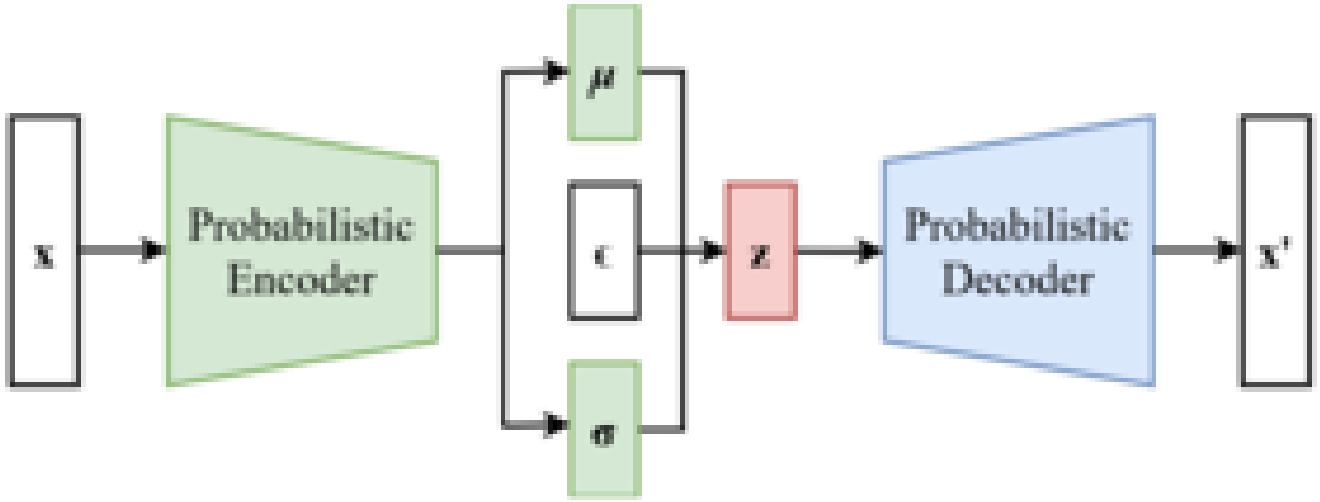
- Con los datos de entrada se tiene una distribución de probabilidad a priori $p(z|x)$ del *encoder.* Para el espacio latente se escoje para $p(z)$ la función de distribución gausiana.



- Se calcula la media y la varianza de la distribución de probabilidad del *encoder*. Así junto con el vector del espacio latente se tienen dos vectores más: media y varianza

Ajuste de Variational autoencoder

- Se busca la mejor distribución de probabilidad del *decoder* $p(x|z)$, comenzando con la gausiana. Gracias a la divergencia de Kullback-Lieber y usando la reparametrización: $z = \mu + \sigma \odot \epsilon$ se calcula $p(x|z)$



- Finalmente, con la mejor función de distribución de probabilidad del *decoder* $p(x|z)$, se ajusta la función de error a minimizar.