

Generative Adversial Networks

Deep Learning



Marco Teran
Universidad Sergio Arboleda

2023

Contenido

1 Introducción

2 Autoencoders

- Estructura y Función de un Autoencoder
- Stacked Autoencoders
- Entrenamiento no supervisado con Autoencoders Apilados
- Autoencoders en Redes Convolucionales
- Denoising Autoencoders

3 Sparse Autoencoders

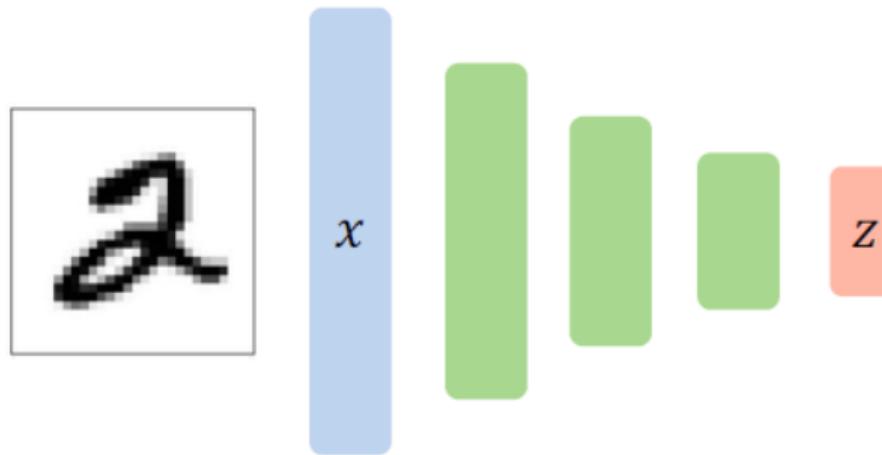
4 Variational Autoencoders (VAEs)

5 Resumen

Autoencoders

Autoencoders: background

Enfoque no supervisado para aprender una representación de características de menor dimensión a partir de datos de entrenamiento no etiquetados

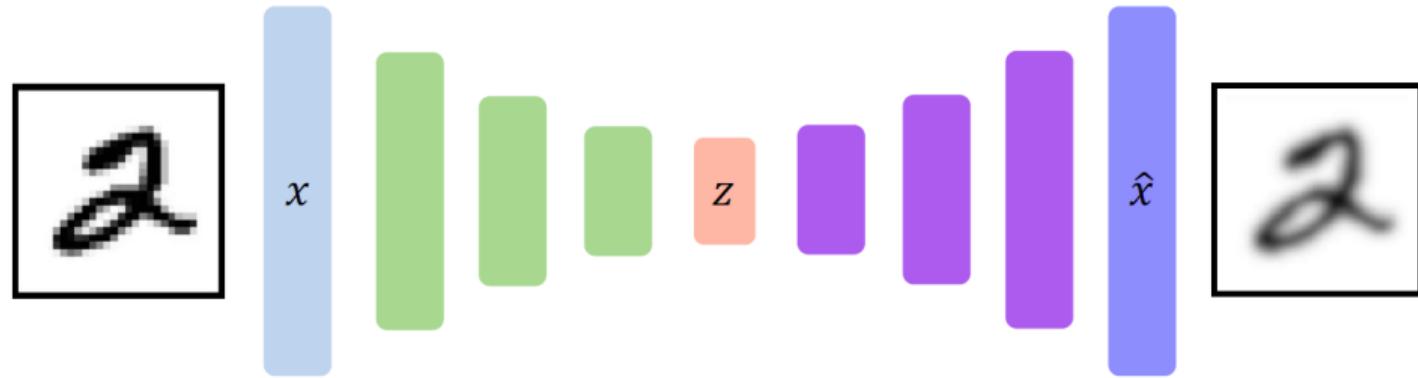


El **Codificador** (*encoder*) aprende el mapeo de los datos, x , a un espacio latente de baja dimensión, z

Autoencoders: background

¿Cómo podemos aprender este espacio latente?

Entrena al modelo para que use estas características para **reconstruir los datos originales**

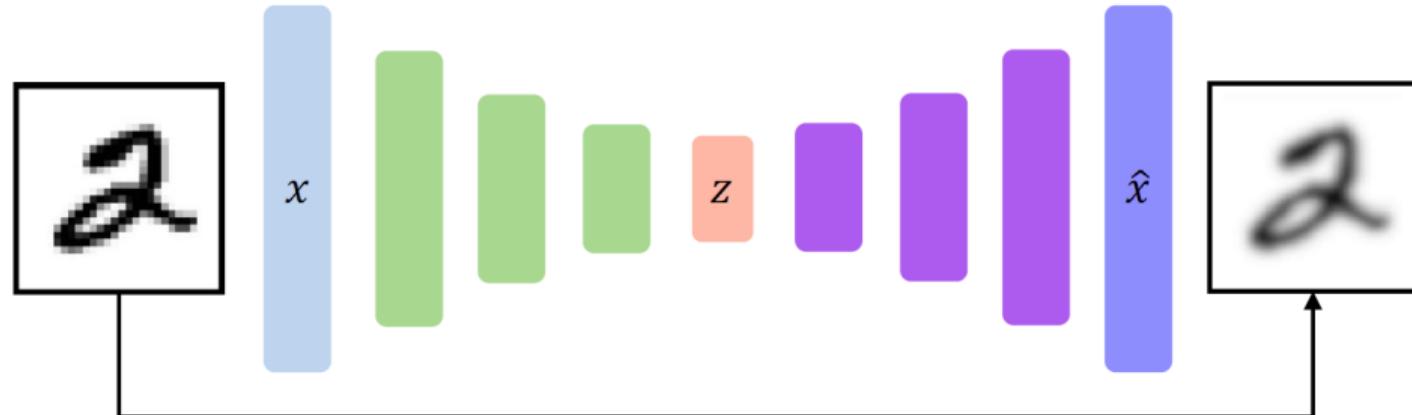


El **Decodificador** (*decoder*) aprende a mapear desde la latente, z , hasta una observación reconstruida, \hat{x}

Autoencoders: background

¿Cómo podemos aprender este espacio latente?

Entrena al modelo para que use estas características para **reconstruir los datos originales**



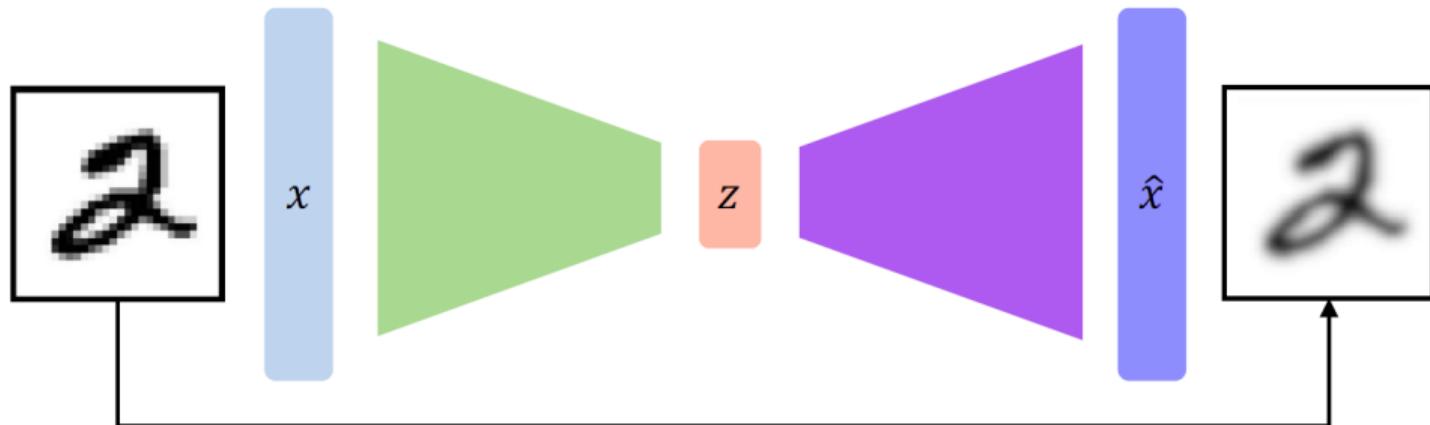
$$\mathcal{L}(x, \hat{x}) = \|x - \hat{x}\|^2$$

La función de pérdida no usa
ninguna etiqueta!!

Autoencoders: background

¿Cómo podemos aprender este espacio latente?

Entrena al modelo para que use estas características para **reconstruir los datos originales**



$$\mathcal{L}(x, \hat{x}) = \|x - \hat{x}\|^2$$

La función de pérdida no usa
ninguna etiqueta!!

Autoencoders

Autoencoders

Los autoencoders son redes neuronales artificiales capaces de aprender **representaciones densas** de datos de entrada, conocidas como *representaciones latentes*, sin supervisión. Estas codificaciones generalmente tienen una **dimensionalidad mucho menor** que los datos de entrada, lo que los hace útiles para la **reducción de dimensionalidad**.

Autoencoders

- Útiles para reducción de dimensionalidad y visualización.
- Actúan como detectores de características.
- Pueden ser utilizados para preentrenamiento no supervisado de redes neuronales profundas.
- Algunos autoencoders son modelos generativos: pueden generar nuevos datos similares a los de entrenamiento.

La dimensionalidad del espacio latente → calidad de la reconstrucción

- ¡La autocodificación es una forma de compresión!
- Un espacio latente más pequeño forzará un mayor cuello de botella de entrenamiento

2D latent space	5D latent space	Ground Truth
		

Autocodificadores para el aprendizaje por representación

- La **capa oculta cuello de botella** obliga a la red a aprender una comprimida representación latente
- La **pérdida por reconstrucción** obliga a la representación latente a capturar (o *codificar*) tanta **información** sobre los datos como sea posible
- Autocodificación = **Codificación** automática de datos

Representaciones Eficientes de Datos

Representaciones Eficientes de Datos

¿Cuál de las siguientes secuencias numéricas encuentra más fácil de memorizar?

- 40, 27, 25, 36, 81, 57, 10, 73, 19, 68
- 50, 48, 46, 44, 42, 40, 38, 36, 34, 32, 30, 28, 26, 24, 22, 20, 18, 16, 14

Representaciones Eficientes de Datos

- Las secuencias más cortas no siempre son más fáciles de memorizar.
- Reconocer un patrón en los datos puede simplificar la memorización.
- Las restricciones durante el entrenamiento de un autoencoder lo empujan a descubrir y explotar patrones en los datos.

Representaciones Eficientes de Datos

¿Cuál de las siguientes secuencias numéricas encuentra más fácil de memorizar?

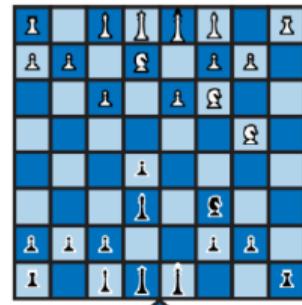
- 40, 27, 25, 36, 81, 57, 10, 73, 19, 68
- 50, 48, 46, 44, 42, 40, 38, 36, 34, 32, 30, 28, 26, 24, 22, 20, 18, 16, 14

Una secuencia que muestra números pares decrecientes desde 50 hasta 14 es más fácil de recordar que una secuencia aleatoria de números, debido al patrón reconocible.

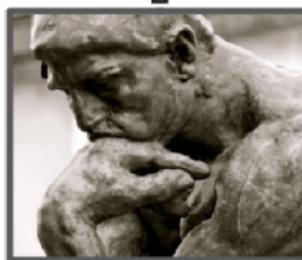
Estudios de Chase y Simon

Los expertos en ajedrez memorizan posiciones de piezas rápidamente si estas provienen de partidas reales. La experiencia ayuda a reconocer patrones y almacenar información de manera eficiente.

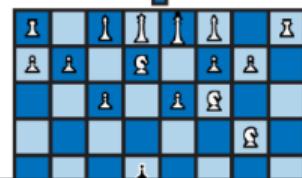
Relación entre Memoria, Percepción y Correspondencia de Patrones



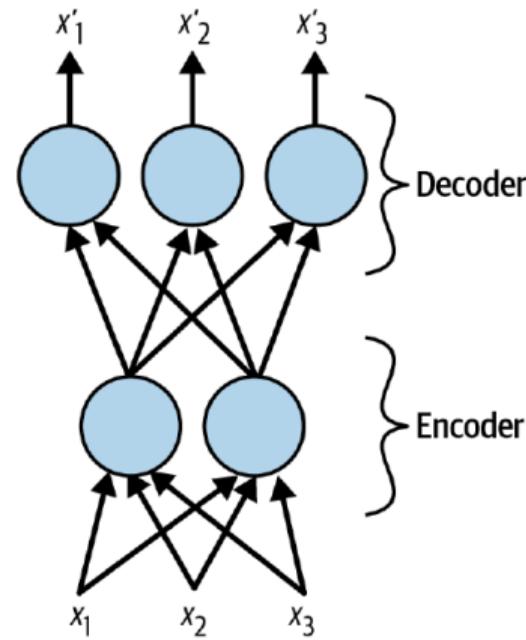
Outputs
(≈inputs)



Latent representation



Inputs



Relación entre Memoria, Percepción y Correspondencia de Patrones

- La percepción de patrones es crucial para una memorización efectiva.
- Los expertos no tienen mejor memoria, pero reconocen patrones con mayor facilidad.
- Un autoencoder convierte entradas a una representación latente eficiente.

Estructura y Función de un Autoencoder

Estructura y Función de un Autoencoder

Compuesto por dos partes:

- Un codificador que convierte las entradas en una representación latente
- Un decodificador que convierte esta representación en salidas.

Estructura y Función de un Autoencoder

- Arquitectura similar a un perceptrón multicapa.
- El número de neuronas en la capa de salida es igual al número de entradas.
- El autoencoder trata de reconstruir las entradas.
- Los autoencoders incompletos.^aprenden características importantes de los datos.

Un autoencoder con una representación interna de menor dimensión que los datos de entrada aprende a destacar y retener solo las características más importantes de los datos.

Stacked Autoencoders

Stacked Autoencoders

Autoencoders Apilados

Al igual que otras redes neuronales, los autoencoders pueden tener múltiples capas ocultas. En este caso, se les denomina autoencoders apilados o profundos. Estos aprenden codificaciones más complejas al agregar más capas.

Stacked Autoencoders

- Pueden tener múltiples capas ocultas.
- Aprender codificaciones más complejas.
- No deben ser demasiado potentes para evitar sobreajuste.
- La arquitectura es típicamente simétrica respecto a la capa central.
- Su diseño se asemeja a un sándwich.

Arquitectura Simétrica

La arquitectura de un autoencoder apilado es generalmente simétrica en relación con la capa oculta central, conocida como la capa de codificación.

Arquitectura Simétrica

- Diseño similar a un sándwich.
- La capa central es la capa de codificación.
- Las capas anteriores y posteriores a la capa central son simétricas.
- La entrada y salida tienen dimensiones similares.

Stacked autoencoder

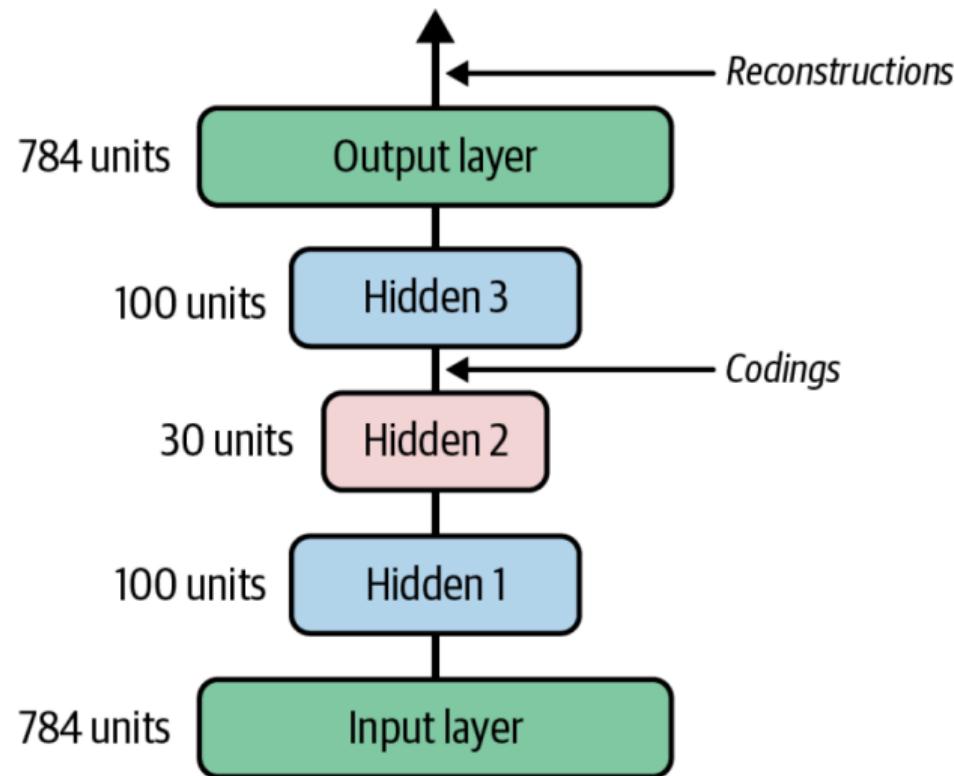
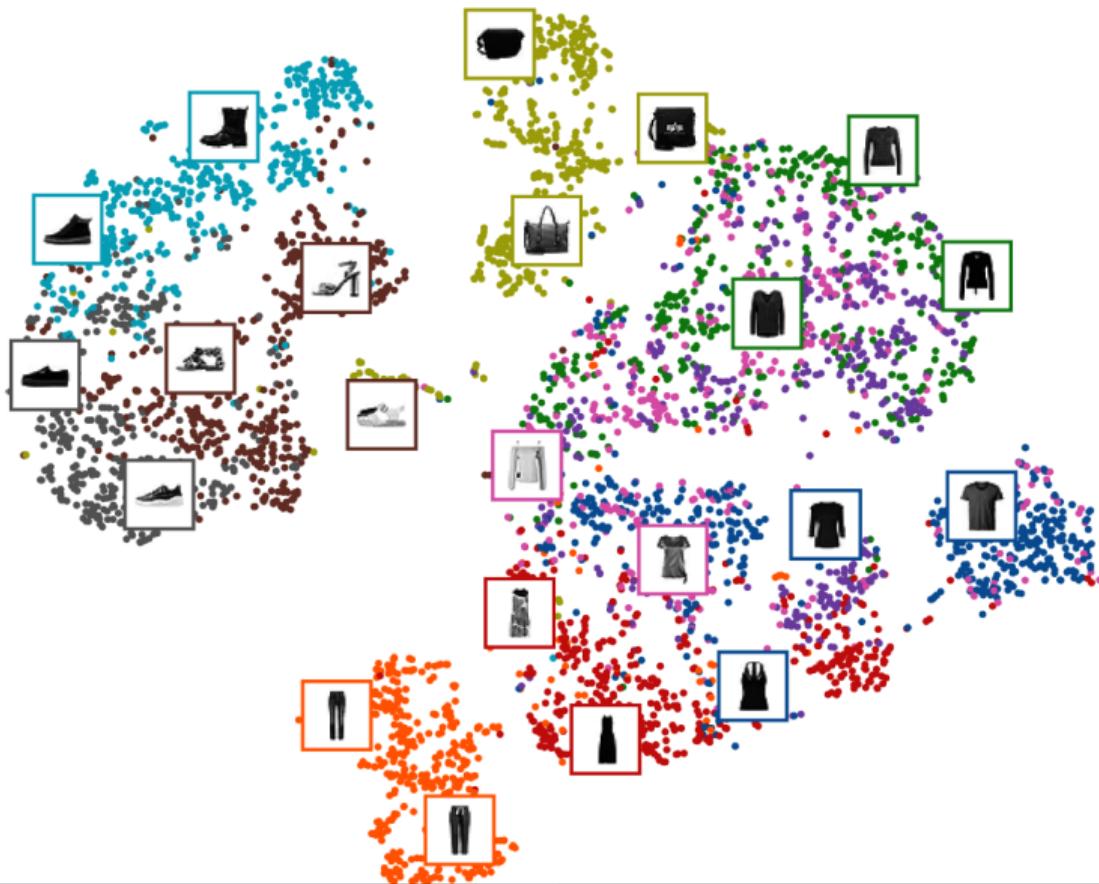


Figura 2: Stacked autoencoder

Stacked autoencoder



Entrenamiento no supervisado con Autoencoders Apilados

Entrenamiento no supervisado con Autoencoders Apilados

Preentrenamiento no supervisado

Si se enfrenta a una tarea supervisada compleja con pocos datos etiquetados, se puede reutilizar las capas inferiores de una red neuronal que realice una tarea similar. Esto permite entrenar un modelo de alto rendimiento con pocos datos, reutilizando detectores de características de la red existente.

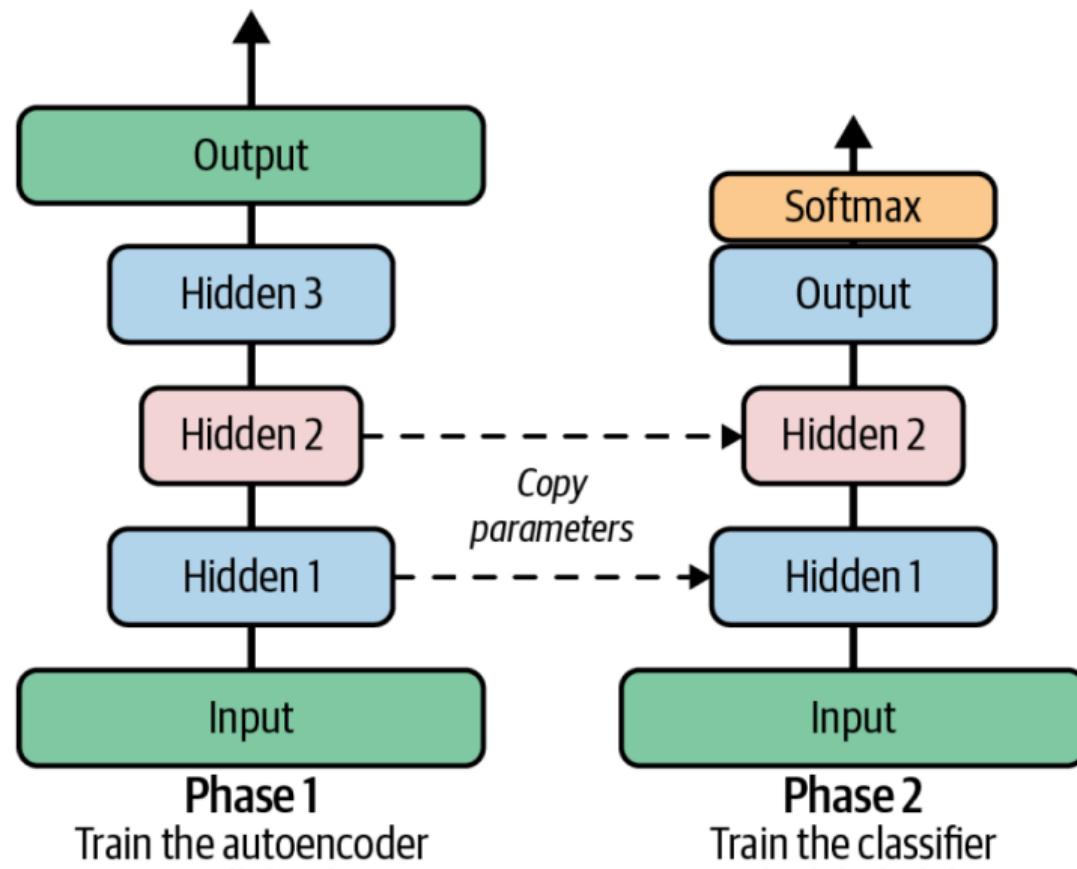
Entrenamiento no supervisado con Autoencoders Apilados

- Reutilización de capas inferiores para tareas similares.
- El modelo no necesita aprender todas las características de bajo nivel.
- Los autoencoders apilados pueden ser entrenados con datos no etiquetados.
- Se pueden congelar las capas preentrenadas para mejorar la formación.
- El preentrenamiento no supervisado es útil para redes neuronales de clasificación.

Entrenamiento no supervisado con Autoencoders Apilados

Utilizar un autoencoder apilado para preentrenamiento no supervisado en una red neuronal de clasificación. Si hay pocos datos etiquetados, congelar las primeras capas del autoencoder.

Stacked autoencoder



Training one autoencoder at a time

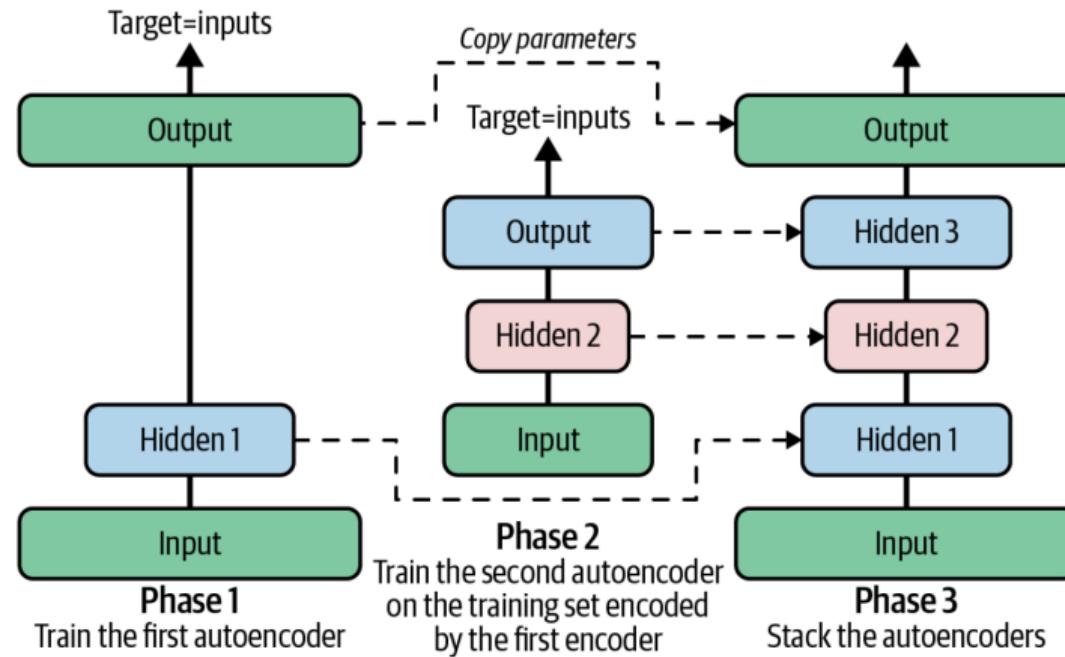


Figura 5: Training one autoencoder at a time

Convolutional Autoencoders

Autoencoders en Redes Convolucionales

Los autoencoders no se limitan a redes densas; también es posible construir autoencoders convolucionales, que utilizan convoluciones para detectar patrones espaciales.

Definición de Autoencoders Convolucionales

Si se trabaja con imágenes, los autoencoders tradicionales no serán eficientes a menos que las imágenes sean pequeñas. Para imágenes, es preferible usar autoencoders convolucionales, diseñados usando redes neuronales convolucionales (CNN).

Autoencoders en Redes Convolucionales

- Los autoencoders convolucionales funcionan con datos estructurados espacialmente.
- Utilizan operaciones de convolución para la compresión y deconvolución para la reconstrucción.
- Permiten la detección de características jerárquicas en imágenes y otros datos espaciales.
- Pueden ser entrenados capa por capa o de una sola vez.
- El codificador (encoder) es una CNN típica con capas convolucionales y de agrupación.
- Reduce la dimensionalidad espacial mientras aumenta la profundidad (número de mapas de características).
- El decodificador realiza el proceso inverso utilizando capas convolucionales transpuestas.

Autoencoders en Redes Convolucionales

En el procesamiento de imágenes, un autoencoder convolucional puede aprender a reconocer patrones como bordes y texturas en sus capas inferiores.

Tipos Avanzados de Autoencoders

Más allá de los autoencoders básicos, existen variantes avanzadas como autoencoders de ruido, autoencoders dispersos y autoencoders variacionales que se adaptan a diferentes necesidades y aplicaciones.

Tipos Avanzados de Autoencoders

- Autoencoders de ruido: Introducen ruido en la entrada durante el entrenamiento.
- Autoencoders dispersos: Usan una penalización de dispersión para activaciones raras.
- Autoencoders variacionales: Añaden una capa de aleatoriedad para modelar distribuciones de datos.

Denoising Autoencoders

Introducción a Autoencoders de Denoising

Definición de Autoencoders de Denoising

Una forma de forzar a los autoencoders a aprender características útiles es agregar ruido a sus entradas y entrenarlos para recuperar las entradas originales sin ruido. Esta idea ha estado presente desde la década de 1980.

Introducción a Autoencoders de Denoising

- Yann LeCun mencionó esta idea en su tesis de maestría de 1987.
- Pascal Vincent et al., en 2008, demostraron que los autoencoders pueden usarse para extracción de características.
- En 2010, Vincent et al. presentaron autoencoders de denoising apilados.

Tipos de Ruido en Autoencoders de Denoising

El ruido puede ser ruido gaussiano puro añadido a las entradas, o pueden ser entradas apagadas aleatoriamente, similar al dropout.

- Ruido Gaussiano: Ruido continuo añadido a la entrada.
- Dropout: Apaga aleatoriamente una proporción de las entradas.
- Ambas opciones están visualizadas en la Figura 17-8.

Tipos de Ruido en Autoencoders de Denoising

Imagina añadir un ruido gaussiano aleatorio a una imagen de un gato, haciendo que algunas partes de la imagen sean poco claras. El autoencoder deberá aprender a "limpiar.^{esta} imagen.

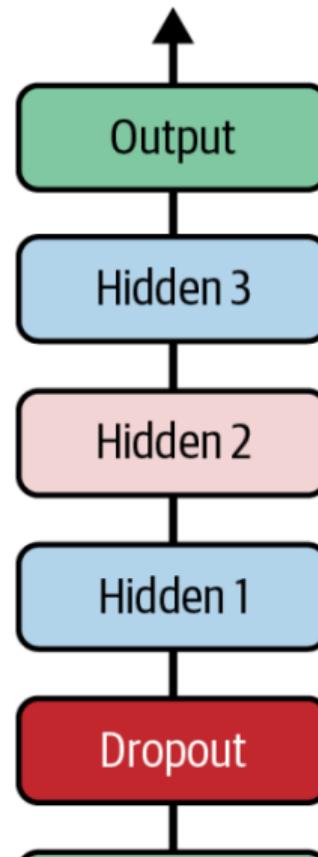
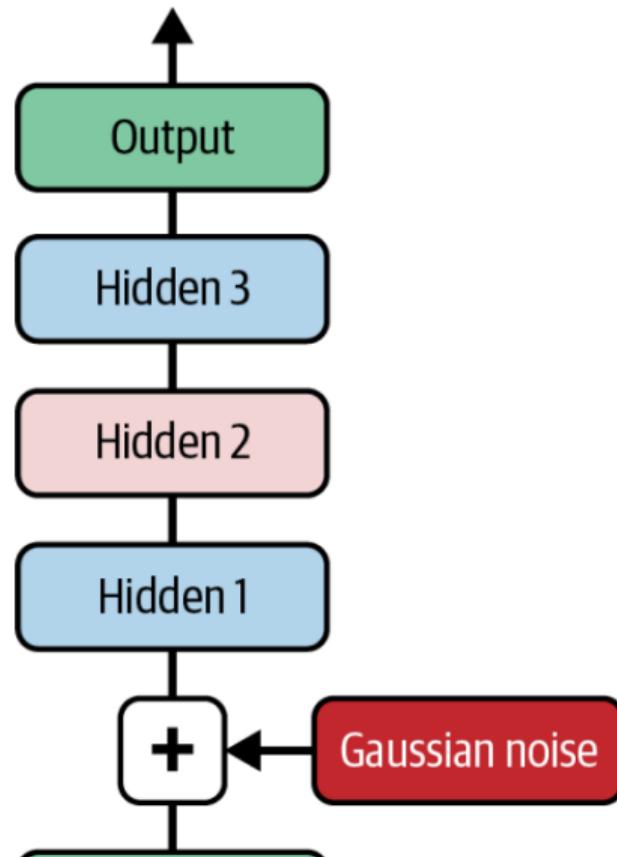
Implementación de Autoencoders de Denoising

La implementación es directa: es un autoencoder apilado regular con una capa adicional de Dropout aplicada a las entradas del codificador. Alternativamente, se puede usar una capa de GaussianNoise.

Implementación de Autoencoders de Denoising

- La capa de Dropout solo está activa durante el entrenamiento.
- Igualmente, la capa GaussianNoise solo se activa durante el entrenamiento.
- Estas capas introducen ruido solamente en la fase de entrenamiento.

Denoising autoencoders



Denoising autoencoders



Figura 7: Noisy images (top) and their reconstructions (bottom)

Sparse Autoencoders

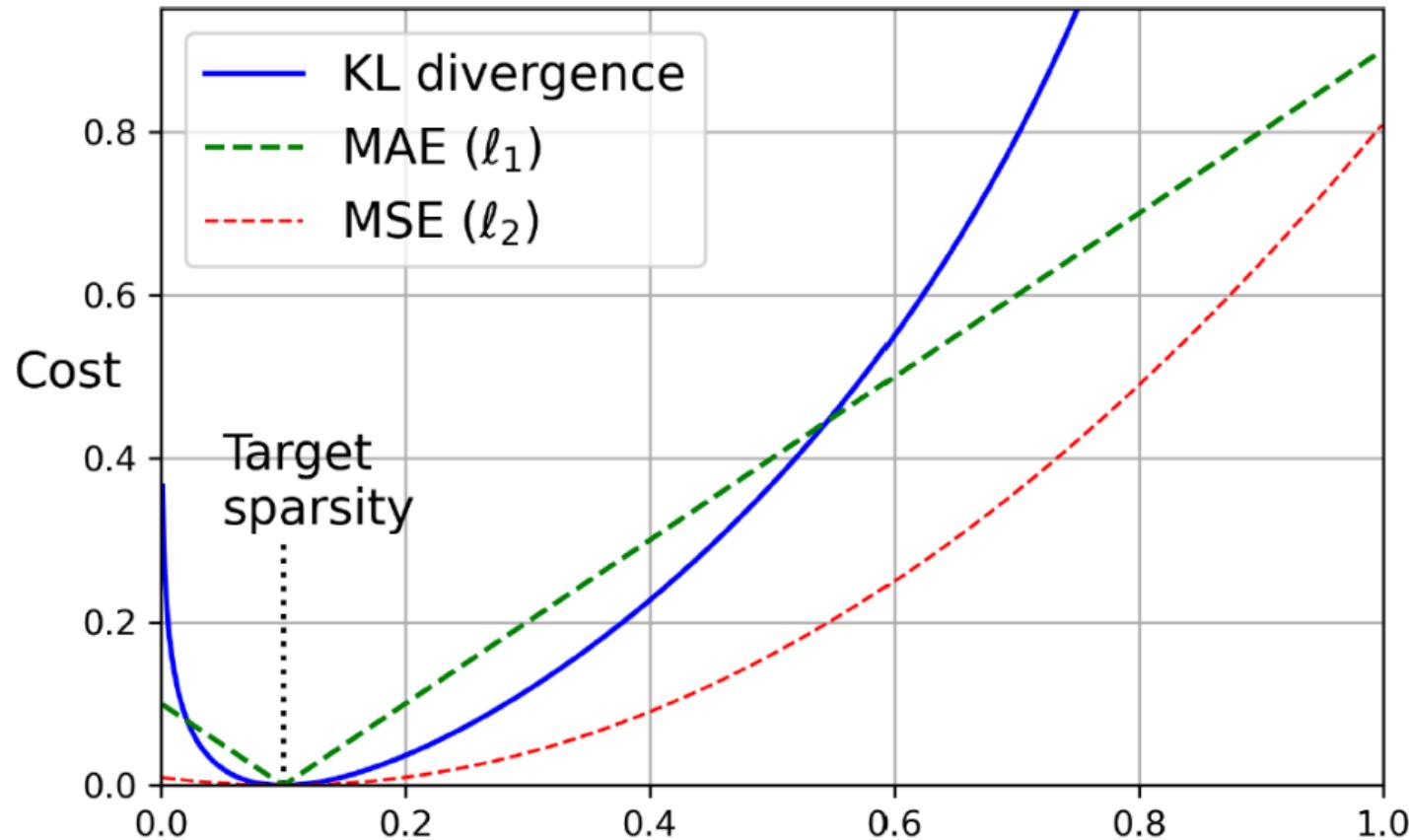
Autoencoders Dispersos (Sparse Autoencoders)

Definición de Autoencoders Dispersos

Un tipo de restricción que a menudo conduce a una buena extracción de características es la dispersión. Mediante la adición de un término adecuado a la función de coste, se impulsa al autoencoder a reducir el número de neuronas activas en la capa de codificación.

- La restricción de dispersión lleva a la representación de cada entrada con un pequeño número de activaciones.
- Se puede usar la función de activación sigmoide en la capa de codificación para limitar los codings entre 0 y 1.
- Se añade una regularización 1 a las activaciones de la capa de codificación.
- Otra técnica es medir la dispersión real de la capa de

Sparse Autoencoders



Variational Autoencoders (VAEs)

Introducción a las Redes Generativas Adversarias (GANs)

Definición

Las GANs fueron propuestas en un artículo de 2014 por Ian Goodfellow y colaboradores. Estas hacen competir redes neuronales entre sí, con la esperanza de que esta competencia las impulse a sobresalir.

- Propuesto en 2014 por Ian Goodfellow.
- Hace competir a dos redes neuronales.
- Compuesto por un Generador y un Discriminador.

Componentes de una GAN

Generador:

- Toma una distribución aleatoria (usualmente Gaussiana) como entrada.
- Produce datos, típicamente imágenes.
- Similar a un decodificador en un autoencoder variacional.

Discriminador:

- Toma imágenes falsas del generador o imágenes reales del conjunto de entrenamiento.
- Adivina si la imagen es falsa o real.

Ejemplo: Un generador podría producir imágenes de rostros, mientras que el discriminador intenta determinar si es una imagen real de una persona o una generada artificialmente.

Entrenamiento de GANs

Objetivos Opuestos

Durante el entrenamiento, el generador y el discriminador tienen objetivos opuestos: el discriminador intenta distinguir las imágenes falsas de las reales, mientras que el generador intenta producir imágenes lo suficientemente reales para engañar al discriminador.

- Primera fase: Entrenamos al discriminador.
- Segunda fase: Entrenamos al generador.
- El generador nunca ve imágenes reales.
- El discriminador mejora la información sobre las imágenes reales.

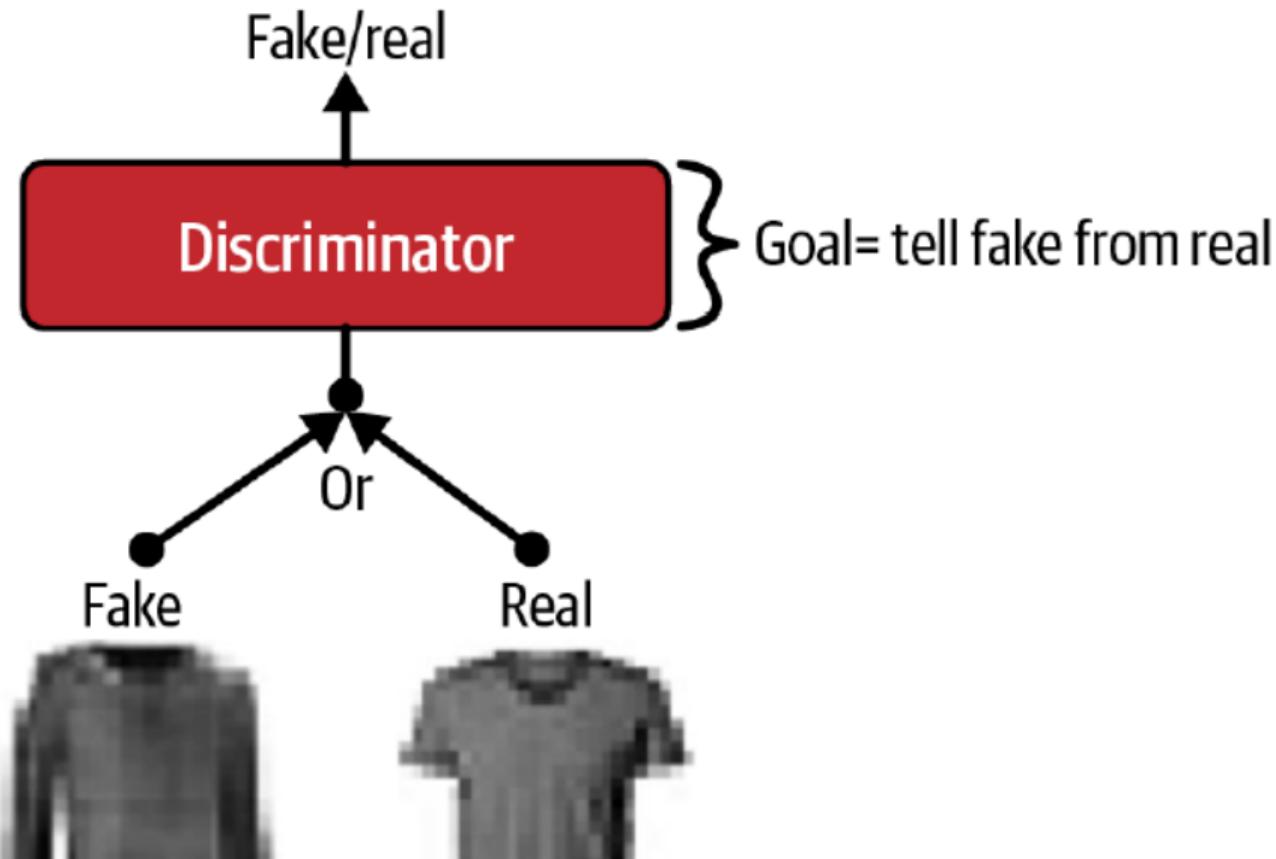
Ejemplo: Es como dos personas jugando un juego, donde una intenta crear moneda falsa y la otra intenta detectarla. Con el tiempo, la primera se vuelve experta en hacer monedas que

Detalles de Entrenamiento

- Fase del discriminador: Usamos imágenes reales y falsas. Etiquetamos como 0 las falsas y 1 las reales.
- Fase del generador: Producimos imágenes falsas. Todas las etiquetas se establecen en 1.
- Solo se optimizan los pesos del discriminador en la primera fase.
- Solo se optimizan los pesos del generador en la segunda fase.

Ejemplo: En la fase del generador, queremos que las monedas falsas parezcan tan reales que incluso un experto no pueda distinguirlas de las reales.

Sparse Autoencoders



Sparse Autoencoders



Figura 10: Images generated by the GAN after one epoch of training

Dificultades al Entrenar GANs

Equilibrio de Nash

Durante el entrenamiento, el generador y el discriminador intentan superarse mutuamente en un juego de suma cero. A medida que avanza el entrenamiento, puede llegar a un estado llamado equilibrio de Nash. Es cuando ningún jugador se beneficiaría cambiando su estrategia, asumiendo que los demás no cambian la suya.

- Un GAN puede alcanzar un solo equilibrio de Nash.
- Esto ocurre cuando el generador produce imágenes perfectamente realistas.
- Sin embargo, no hay garantía de que se alcance el equilibrio.

Ejemplo: Supongamos una situación donde todos conducen en el lado izquierdo de la carretera: nadie se beneficiaría siendo

Colapso del Modo y Problemas Comunes

Colapso del Modo

Ocurre cuando las salidas del generador se vuelven menos diversas. Por ejemplo, si un generador produce más imágenes convincentes de zapatos que de cualquier otra clase, puede olvidar cómo producir otros objetos.

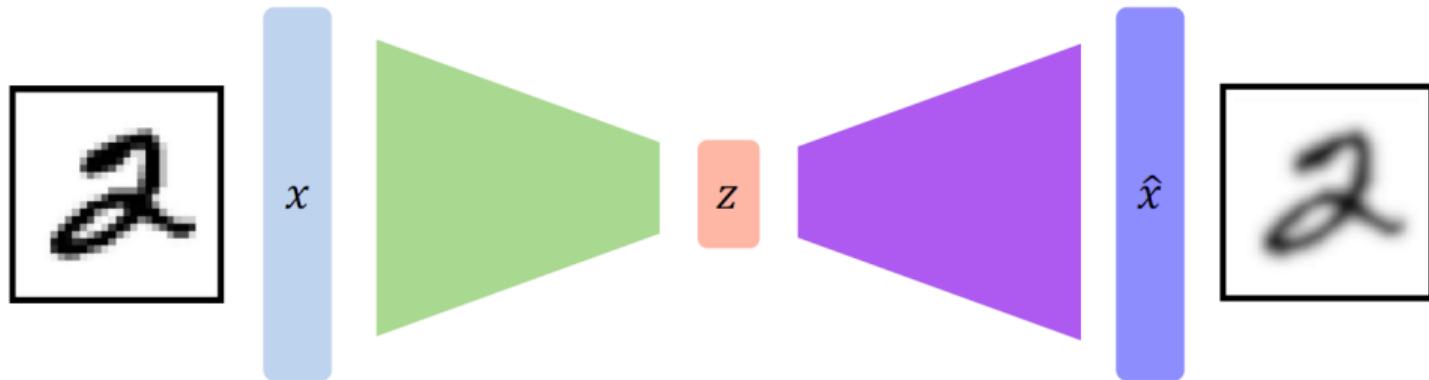
- Los parámetros del generador y del discriminador pueden oscilar y volverse inestables.
- Los GANs son sensibles a los hiperparámetros.
- Se han propuesto diversas técnicas para estabilizar el entrenamiento y evitar el colapso del modo.

Ejemplo: Si un generador produce principalmente zapatos y gradualmente olvida cómo generar otras imágenes, hemos experimentado un colapso del modo.

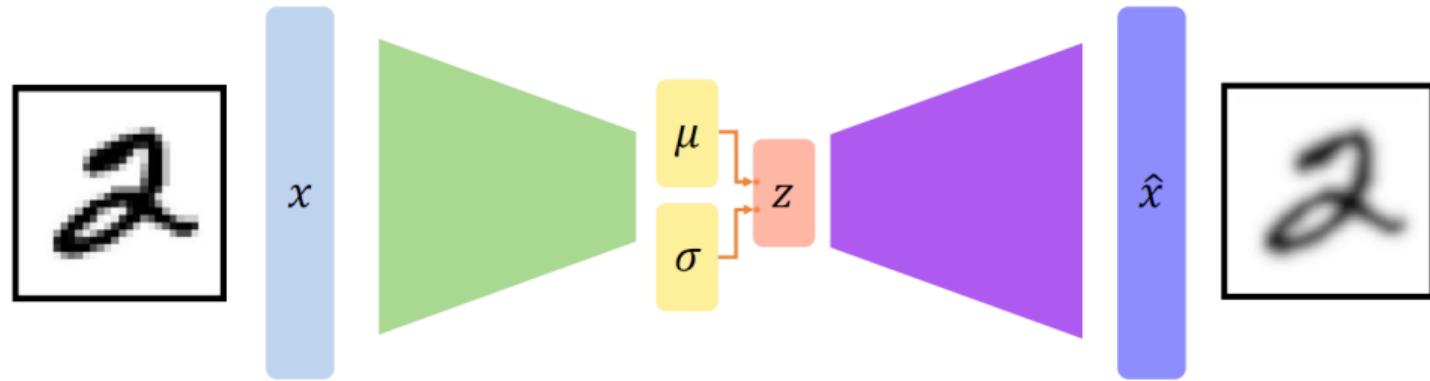
Técnicas y Investigaciones Recientes

- *Experience Replay*: Almacena imágenes producidas por el generador en un búfer y entrena el discriminador con imágenes reales y falsas de este búfer.
 - *Mini-Batch Discrimination*: Mide cuán similares son las imágenes en un lote y proporciona esta estadística al discriminador.
 - Se han propuesto arquitecturas específicas que ofrecen buenos resultados.
- Ejemplo:** Usando la técnica de *Experience Replay*, se reduce la posibilidad de que el discriminador se ajuste demasiado a las últimas salidas del generador, logrando una mayor variedad en las imágenes generadas.

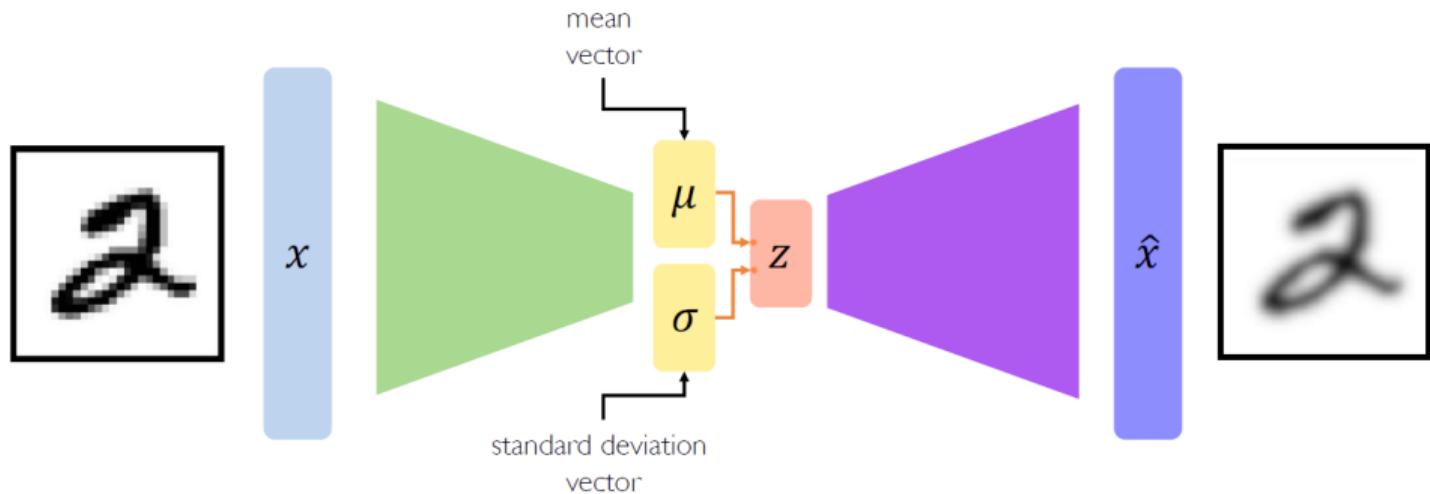
VAEs: diferencia clave con el autoencoder tradicional



VAEs: diferencia clave con el autoencoder tradicional

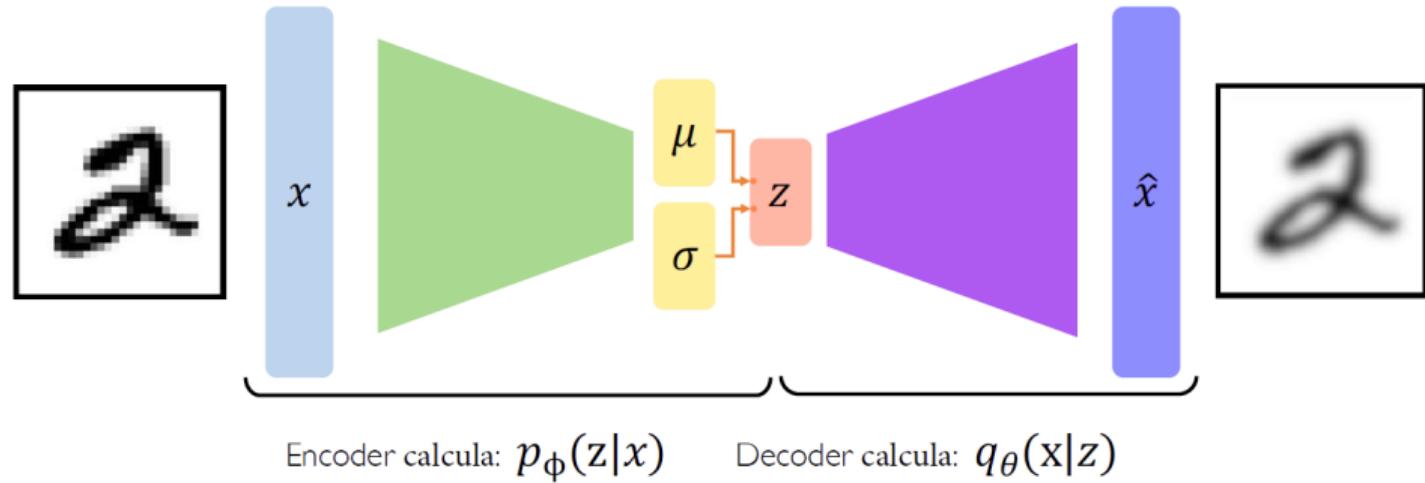


VAEs: diferencia clave con el autoencoder tradicional

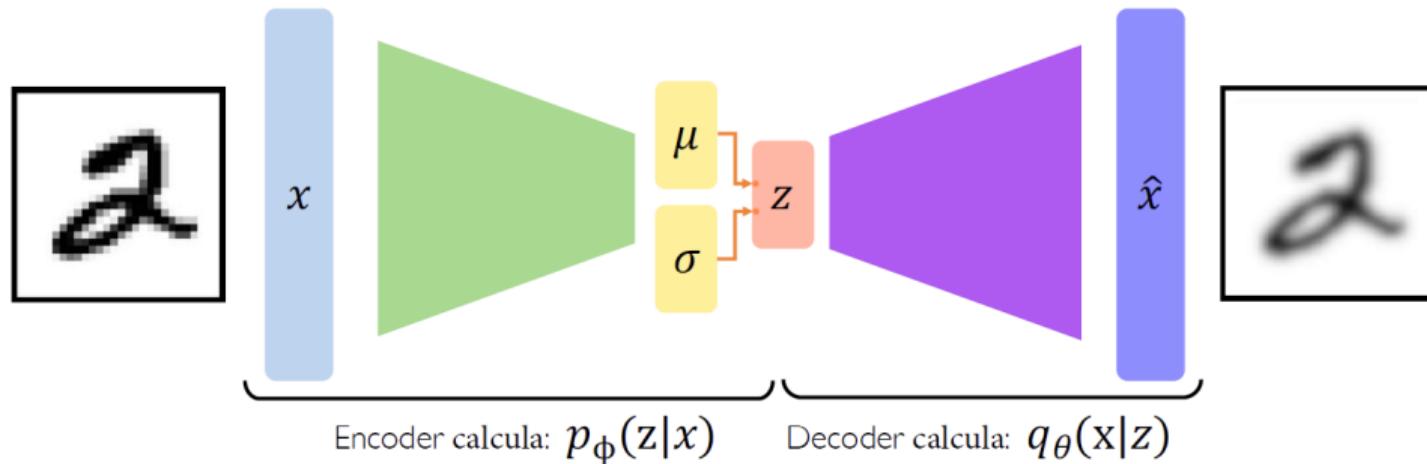


- ¡Los autoencoders variables son un giro probabilístico de los autocodificadores!
- Muestra de la media y la desviación estándar para calcular la muestra latente

Optimización de la VAE

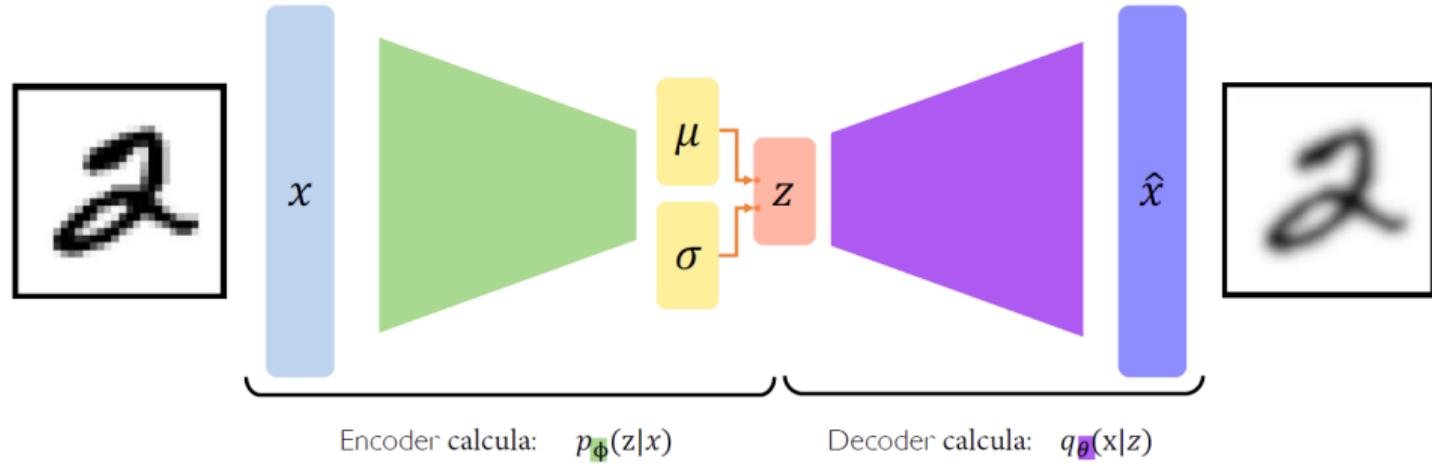


Optimización de la VAE



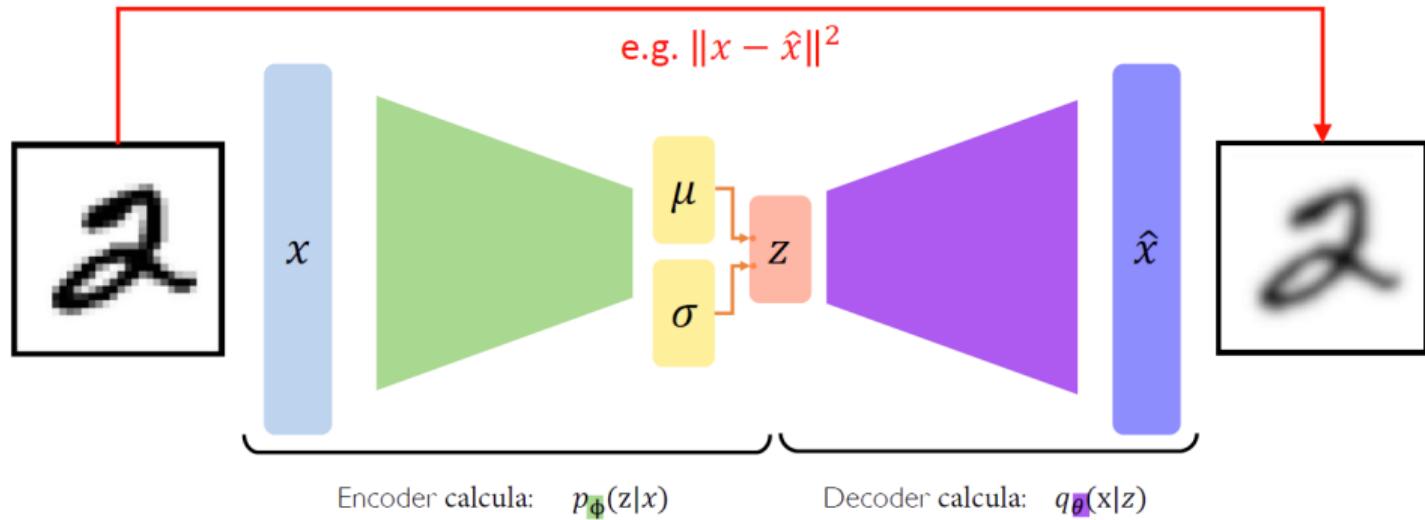
$$\mathcal{L}(\phi, \theta) = (\text{reconstruction loss}) + (\text{termino de regularización})$$

Optimización de la VAE



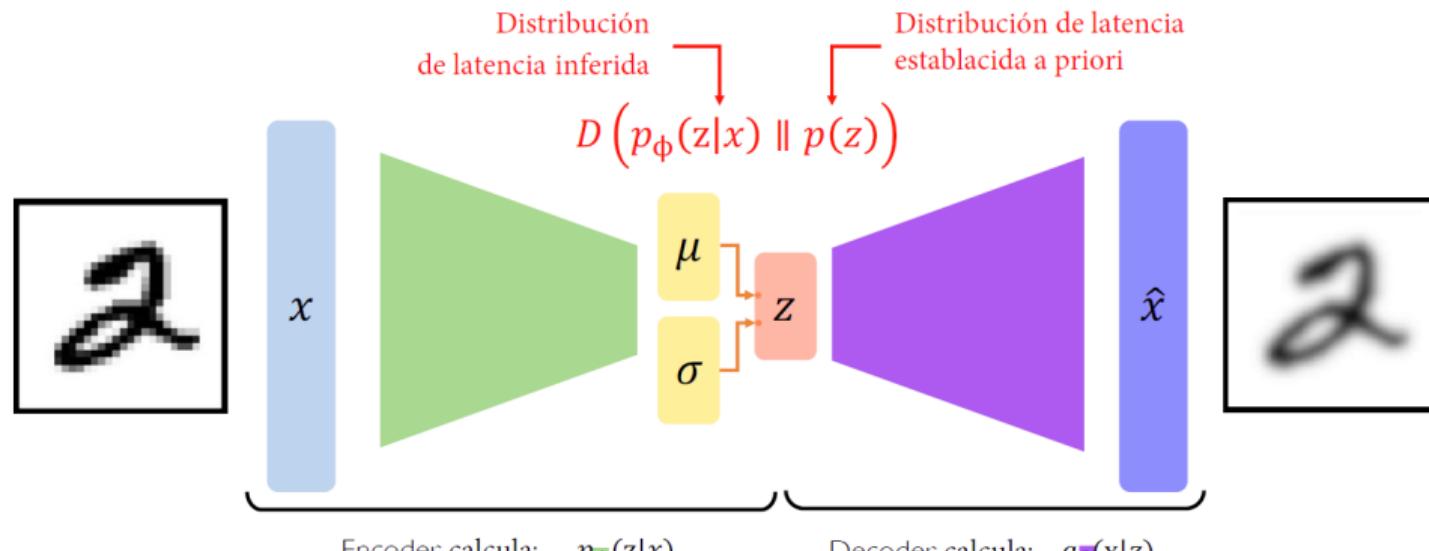
$$\mathcal{L}(\phi, \theta, x) = (\text{reconstruction loss}) + (\text{termino de regularización})$$

Optimización de la VAE



$$\mathcal{L}(\phi, \theta, x) = \text{(reconstruction loss)} + \text{(termino de regularización)}$$

Optimización de la VAE



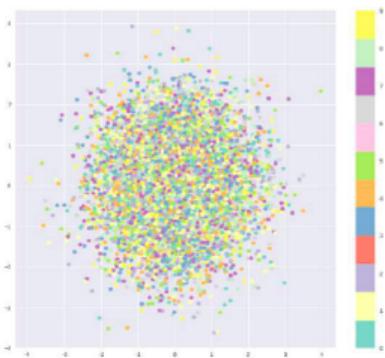
$$\mathcal{L}(\phi, \theta, x) = \text{(reconstruction loss)} + \text{(termino de regularización)}$$

Los *a priors* de la distribución latente

$$D(p_{\phi}(z|x) \parallel p(z))$$

↑ ↑
Distribución Distribución de latencia
de latencia inferida establacida a priori

La elección común del prior:



$$p(z) = \mathcal{N}(\mu = 0, \sigma^2 = 1)$$

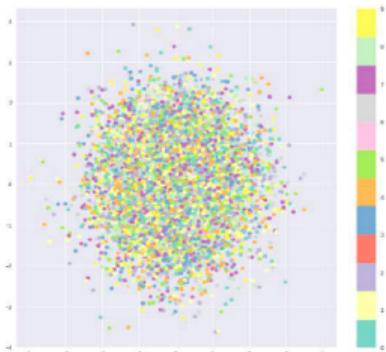
- Alienta a las codificaciones a distribuirlas uniformemente alrededor del centro del espacio latente
- Penaliza a la red cuando intente engañar agrupando puntos en regiones específicas (es decir, memorizando los datos)

Los *a priors* de la distribución latente

$$\begin{aligned} D(p_{\phi}(z|x) \parallel p(z)) \\ = -\frac{1}{2} \sum_{j=0}^{k-1} (\sigma_j + \mu_j^2 - 1 - \log \sigma_j) \end{aligned}$$

KL-divergencia entre las dos distribuciones

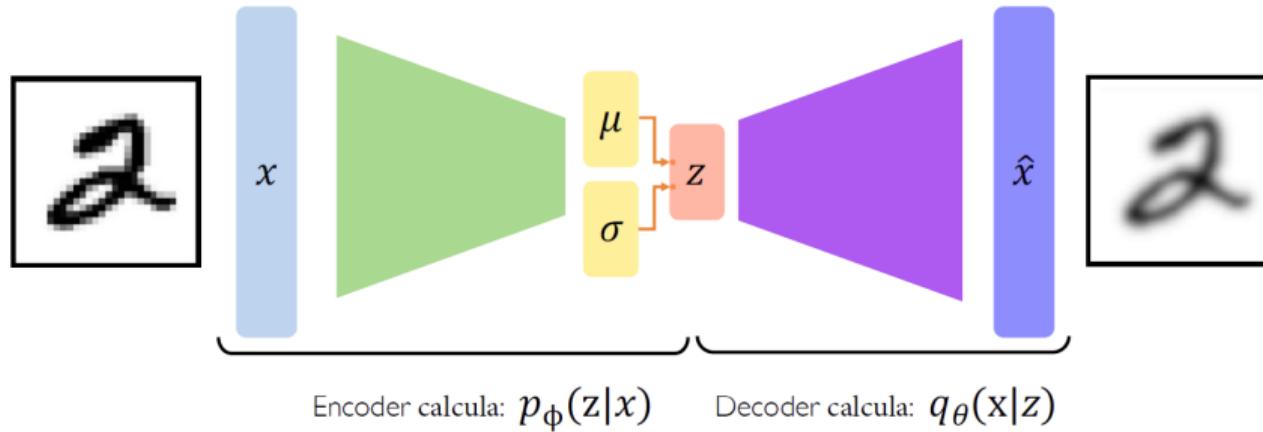
La elección común del prior:



$$p(z) = \mathcal{N}(\mu = 0, \sigma^2 = 1)$$

- Alienta a las codificaciones a distribuir las uniformemente alrededor del centro del espacio latente
- Penaliza a la red cuando intente engañar agrupando puntos en regiones específicas (es decir, memorizando los datos)

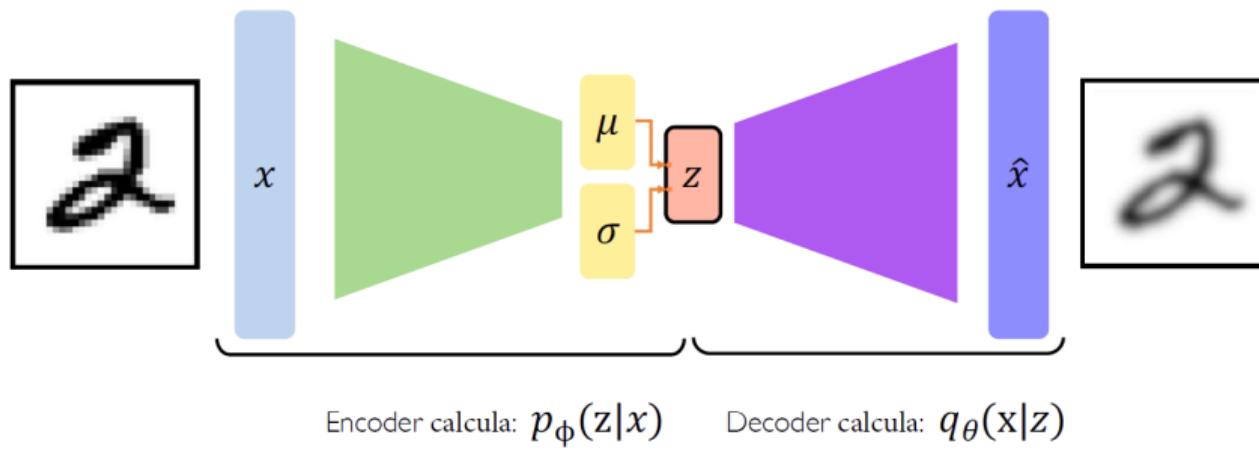
Gráfo de cálculo de la VAE



$$\mathcal{L}(\phi, \theta) = (\text{reconstruction loss}) + (\text{termino de regularización})$$

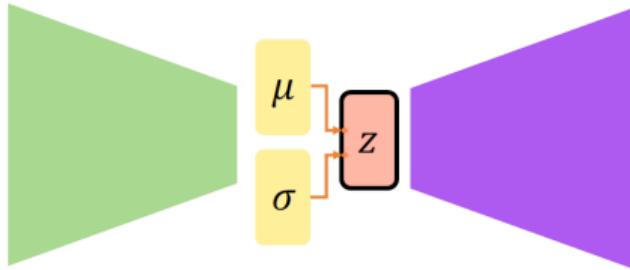
Gráfo de cálculo de la VAE

Problema: No podemos retropropagar los gradientes a través de las capas de muestreo



$$\mathcal{L}(\phi, \theta) = (\text{reconstruction loss}) + (\text{termino de regularización})$$

Reparametrizando la capa de muestreo



Idea clave:

$$z \sim \mathcal{N}(\mu = 0, \sigma^2 = 1)$$

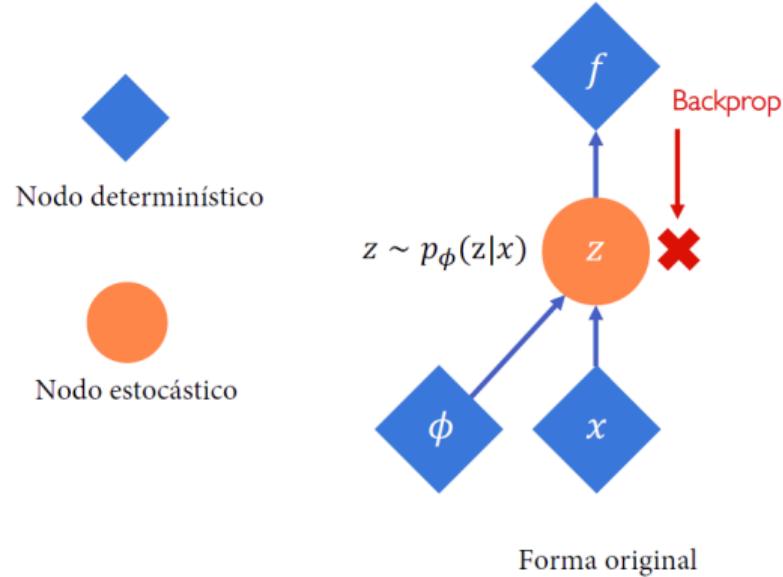
Considerar el vector latente muestreado como una suma de

- un vector μ fijo
- y un vector σ fijo, escalado por constantes aleatorias extraídas de la distribución *a prior*

$$z = \mu + \sigma \odot \epsilon$$

$$\text{donde } \epsilon \sim \mathcal{N}(0, 1)$$

Reparametrizando la capa de muestreo



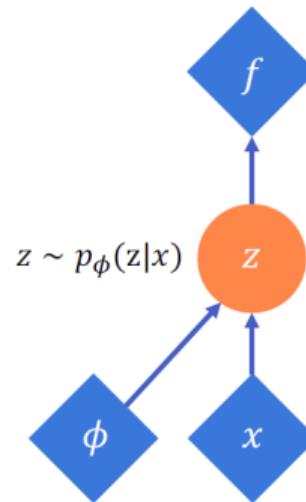
Reparametrizando la capa de muestreo



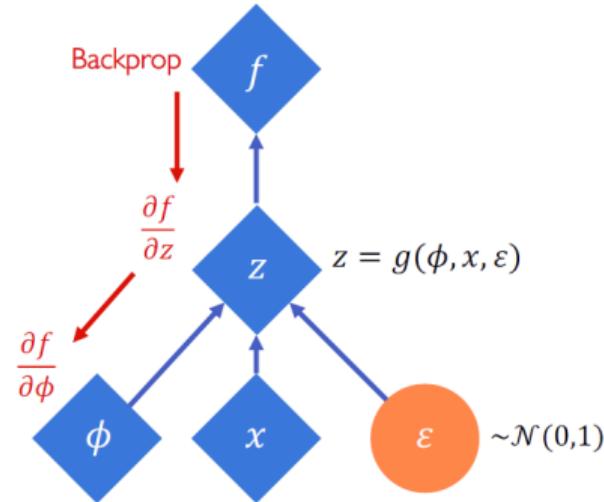
Nodo determinístico



Nodo estocástico



Forma original



Forma reparametrizada

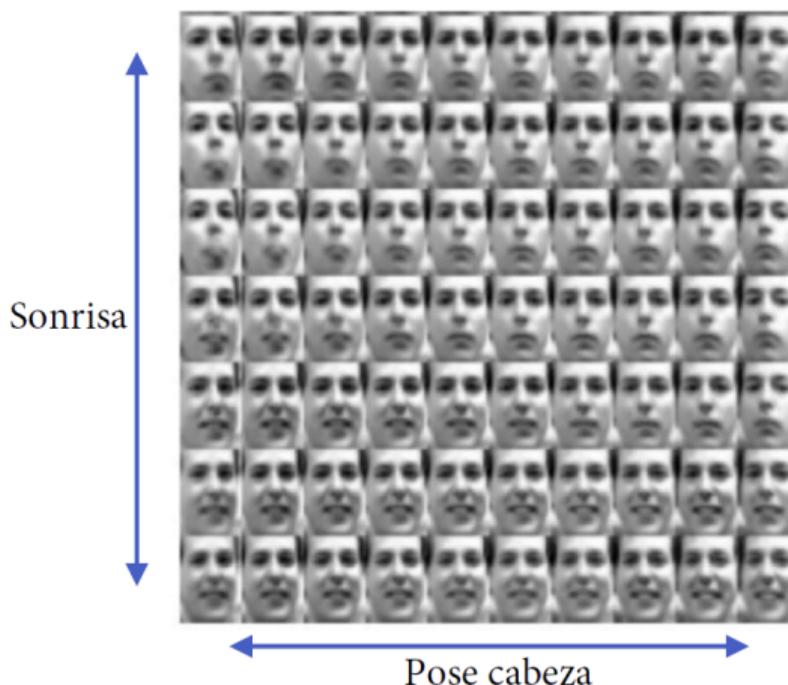
VAEs: Perturbación latente

- Aumentar o disminuir lentamente una **sola variable latente**
- Mantener todas las demás variables fijas



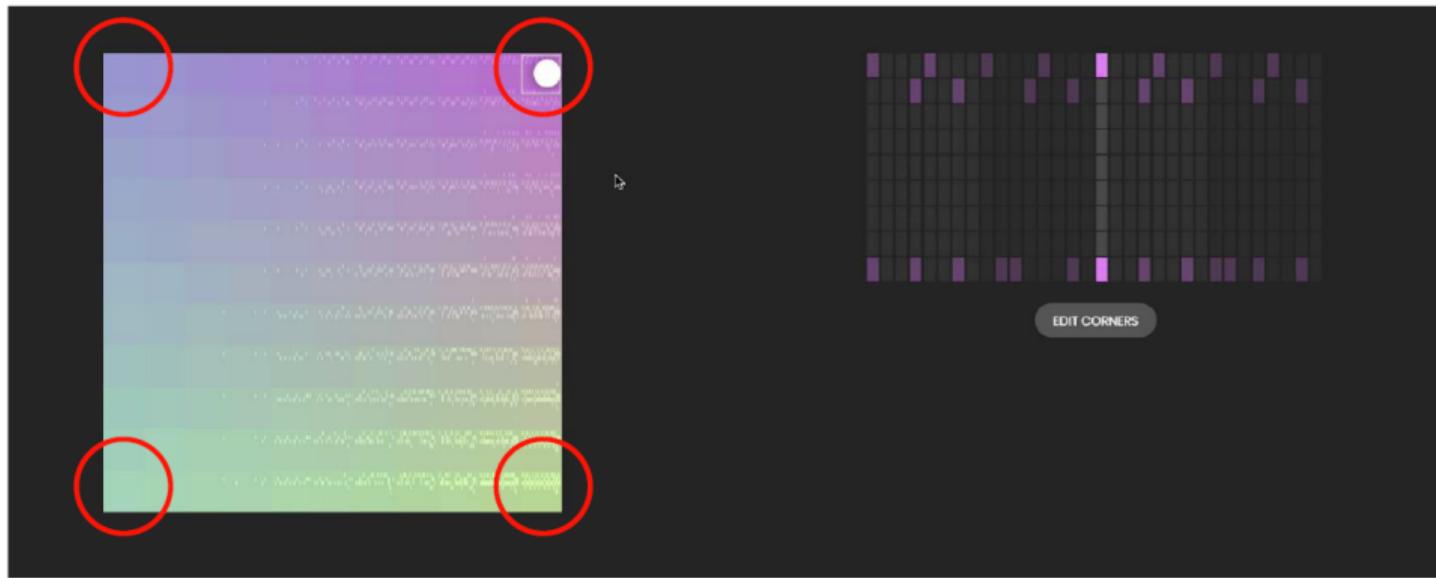
Diferentes dimensiones de z codifica **diferentes características latentes interpretables**

VAEs: Perturbación latente



- Idealmente, queremos variables latentes que no estén correlacionadas entre sí
- Aplicar el priorato diagonal a las variables latentes para fomentar independencia
- **Desenredo** (*Disentanglement*)

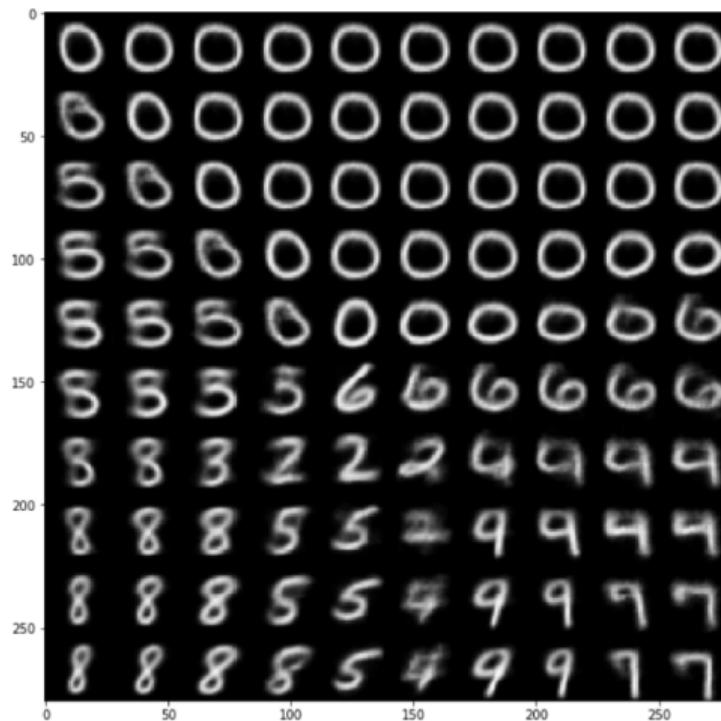
VAEs: Perturbación latente



Google BeatBlender

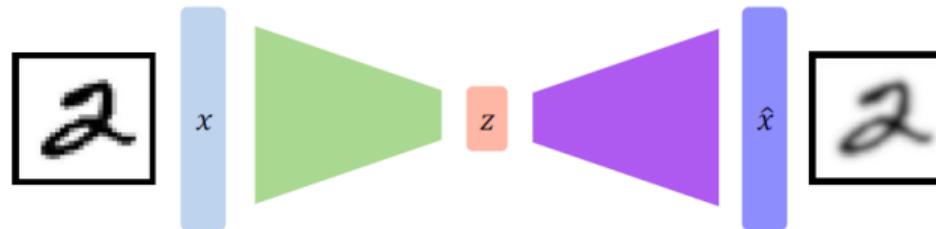
► AI Experiments: Beat Blender

VAEs: Perturbación latente



Resumen de las VAEs

- 1 Comprimir la representación del mundo a algo que podamos usar para aprender
- 2 La reconstrucción permite un aprendizaje no supervisado (¡sin etiquetas!)
- 3 El truco de la reparameterización para entrenar de extremo a extremo
- 4 Interpretar las variables latentes ocultas utilizando la perturbación
- 5 Generar nuevos ejemplos



Comparación con otras redes generativas

Generative Adversarial Networks (GANs)

Las GANs constan de dos redes neuronales: un generador que intenta generar datos que se parezcan a los datos de entrenamiento, y un discriminador que intenta distinguir entre datos reales y falsos. Estas redes compiten entre sí durante el entrenamiento.

- Las GANs pueden generar imágenes extremadamente realistas.
- Utilizadas para super resolución, colorización y edición avanzada de imágenes.
- Entrenamiento adversarial considerado una de las principales innovaciones de la década de 2010.

Ejemplo: Un sitio web muestra rostros generados por una arquitectura GAN llamada StyleGAN, tan convincentes que parecen personas reales.

Más Modelos Generativos y Diferencias

Denoising Diffusion Probabilistic Model (DDPM)

Un DDPM se entrena para eliminar un poco de ruido de una imagen. Al aplicar repetidamente el modelo de difusión en una imagen con ruido gaussiano, emerge gradualmente una imagen de alta calidad.

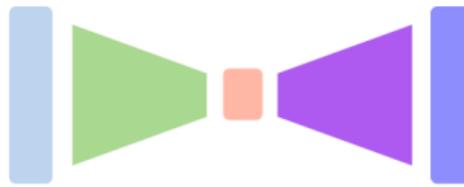
- Autoencoders aprenden a copiar sus entradas a sus salidas con restricciones.
- GANs se componen de generador y discriminador en competencia.
- DDPM elimina ruido de imágenes hasta generar imágenes de calidad.

Ejemplo: Un DDPM utilizado en una imagen con ruido gaussiano puede revelar gradualmente una imagen similar a las imágenes de entrenamiento.

Modelado Generativo Profundo: Resumen

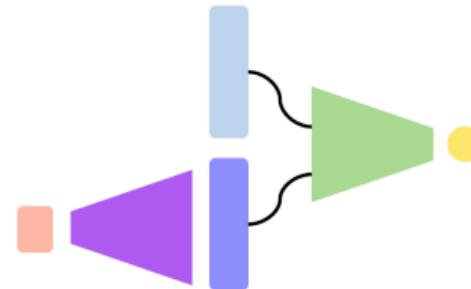
Autoencoders and Variational Autoencoders (VAEs)

- Aprende el espacio latente de **menor dimensión** y toma **muestras** para generar reconstrucciones de entrada



Generative Adversarial Networks (GANs)

- Redes de generadores y discriminadores en competencia



¡Muchas gracias por su atención!

¿Preguntas?



Contacto: Marco Teran
webpage: marcoteran.github.io/
e-mail: marco.teran@usa.edu.co

