

Diffusion models

Métodos Generativos, curso 2025-2026

Guillermo Iglesias, guillermo.iglesias@upm.es

Jorge Dueñas Lerín, jorge.duenas.lerin@upm.es

Edgar Talavera Muñoz, e.talavera@upm.es

5 de noviembre de 2025

Escuela Técnica Superior de Ingeniería de Sistemas Informáticos | UPM



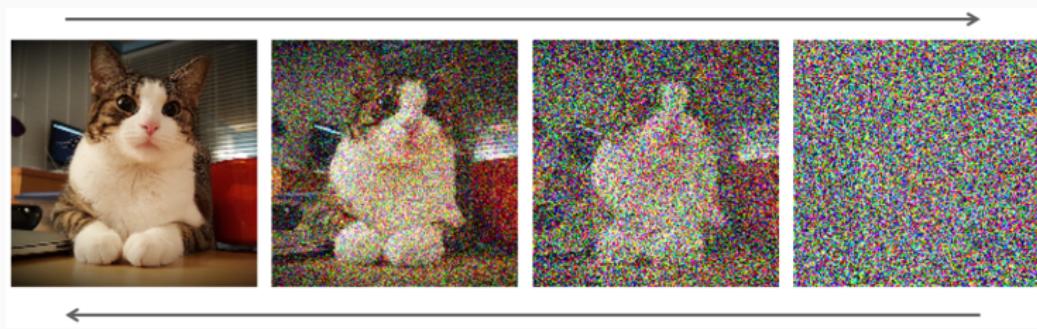
Introducción e intuición previa

¿Qué es un modelo de difusión?

Los modelos de difusión o **diffusion models**, propuestos en el año 2015 [1], tienen su base en la termodinámica.

Se encuentran dentro de los **modelos de variables latentes**, esto es, generan nuevos datos usando un espacio latente, como los Autoencoder (AE), Variational Autoencoder (VAE) o Generative Adversarial Network (GAN).

Su funcionamiento se basa en aplicar **denoising** a un dato de manera iterativa.



Diffusion models en la actualidad

Los diffusion models suponen la tecnología para generación de imágenes **más avanzada** actualmente.

Aplicaciones como **Stable Diffusion** utilizan variaciones de la arquitectura de los modelos de difusión.

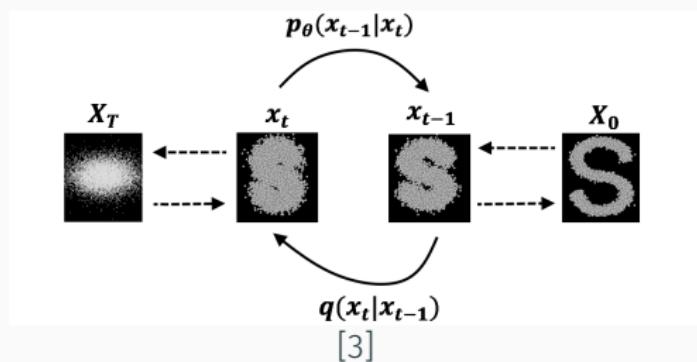
Respecto a arquitecturas anteriores como las GAN, los modelos de difusión solucionan problemas como el **colapso modal o gradient vanishing**, al mismo tiempo que generan imágenes **más definidas y diversas**.

Cadenas de Markov

La arquitectura básica de un diffusion model se basa en las **cadenas de Markov**, que es un modelo probabilístico de cambio de estados.

La principal idea es a partir de una imagen¹ generar **versiones más ruidosas** de la misma de manera iterativa y encadenada.

Finalmente, una **red de neuronas** aprenderá a realizar el proceso inverso, eliminar el ruido de las imágenes.



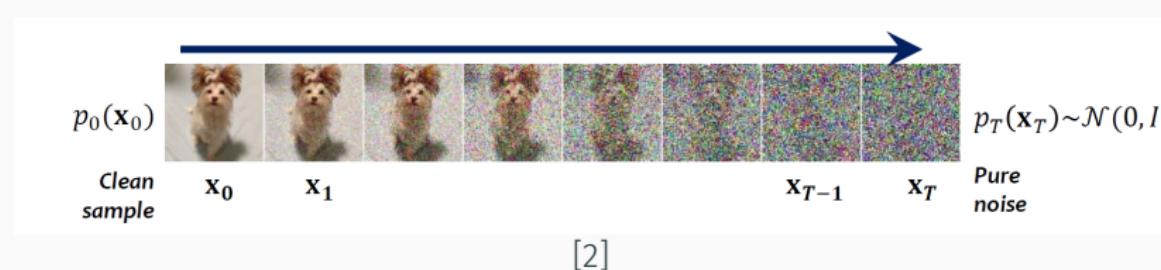
¹Aunque los diffusion models se pueden aplicar a distintas fuentes de datos, por simplicidad de hablará sólo de imágenes.

Sampleo de datos (difusión directa e inversa)

Difusión directa

Como se ha explicado anteriormente, el funcionamiento de los diffusion models se basa en añadir iterativamente **ruido** a la imagen de entrada.

Cada instante $t + 1$ es generado a partir del instante t añadiendo **ruido gaussiano** a la imagen.



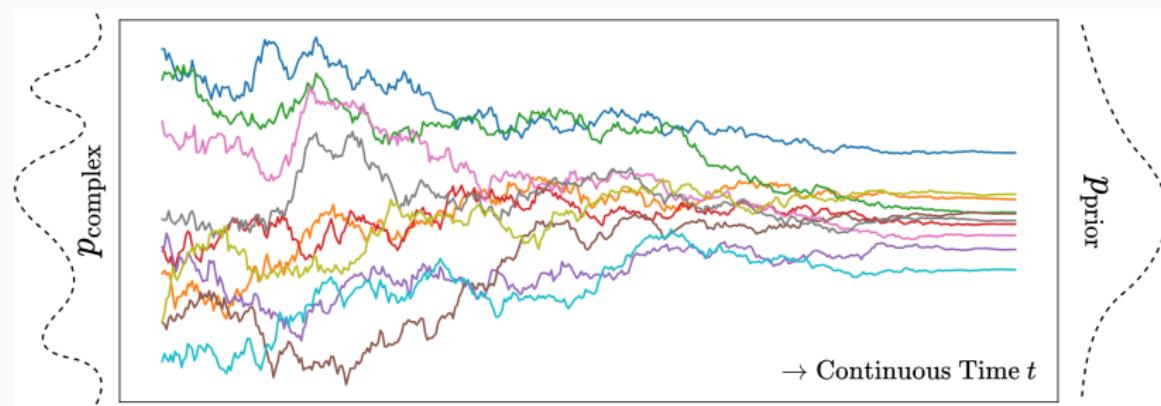
[2]

Cabe destacar que la **magnitud** del ruido generado depende el instante t en el que se encuentre ese ejemplo.

Difusión directa

A través de la introducción de ruido en las imágenes, se consigue cambiar la distribución de datos de los ejemplos del dataset. Cada eslabón de la cadena de Markov de nuestro modelo corresponde a una distribución distinta.

Para el instante T la distribución seguida por los datos será una gaussiana.



[4]

Planificador de ruido

A la hora de introducir ruido de manera **iterativa** en las imágenes que se están procesando hay distintas alternativas.

Para cada imagen generada en un instante t esta se puede definir como:

$$x_t = \sqrt{1 - \beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon_{t-1} \quad (1)$$

donde β es el **diffusion rate**, que se calcula para cada instante t dependiendo de un planificador llamado "**variance scheduler**". Y ϵ_{t-1} es el ruido de cada instante anterior tal que $\epsilon \sim \mathcal{N}(0, 1)$.

¿Por qué llegar hasta imágenes completamente ruidosas?

La idea para generar **nuevas imágenes** es generar **ruido aleatorio** y realizar la difusión inversa.

De esta manera es posible generar **nuevos ejemplos** aleatoriamente, ya que la distribución de datos del instante T es conocida y la del instante 0 se calcula a través del proceso de difusión inversa.

La idea del proceso de inverse diffusion es volver al **ejemplo original** partiendo del **ruidoso**.

Para ello se realiza **iterativamente** la operación de pasar del instante t al instante $t - 1$.

Para generar la imagen del instante $t - 1$ son necesarios:

- La **imagen de entrada** del instante t .
- El **ruido** de dicha imagen calculado con nuestra red de neuronas.
- Un algoritmo para **sustraer** el ruido de la imagen.

Algoritmo DDPM

El algoritmo Denoising Diffusion Probabilistic Model (DDPM), propuesto en el año 2020 [5], es el **causante de la popularidad** que han ganado los diffusion models los últimos años.

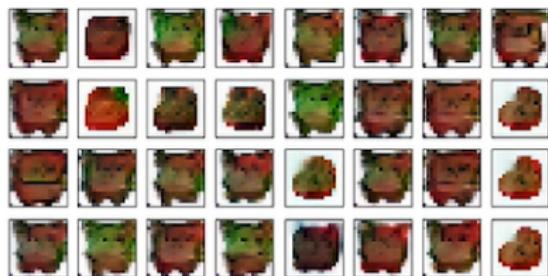
Este algoritmo permite **eliminar** el ruido de una imagen para recuperarla a su estado original.

Una de las características de DDPM es que añade **cierta cantidad de ruido** tras haberlo eliminado. Esto se hace para que la imagen de salida siga la misma **distribución de datos** del instante t .

Algoritmo DDPM

Añadiendo cierto ruido tras cada paso de la difusión inversa, se consigue aumentar la **diversidad** en las imágenes generadas, evitando el colapso modal.

Sampled without noise added



Samples with noise added

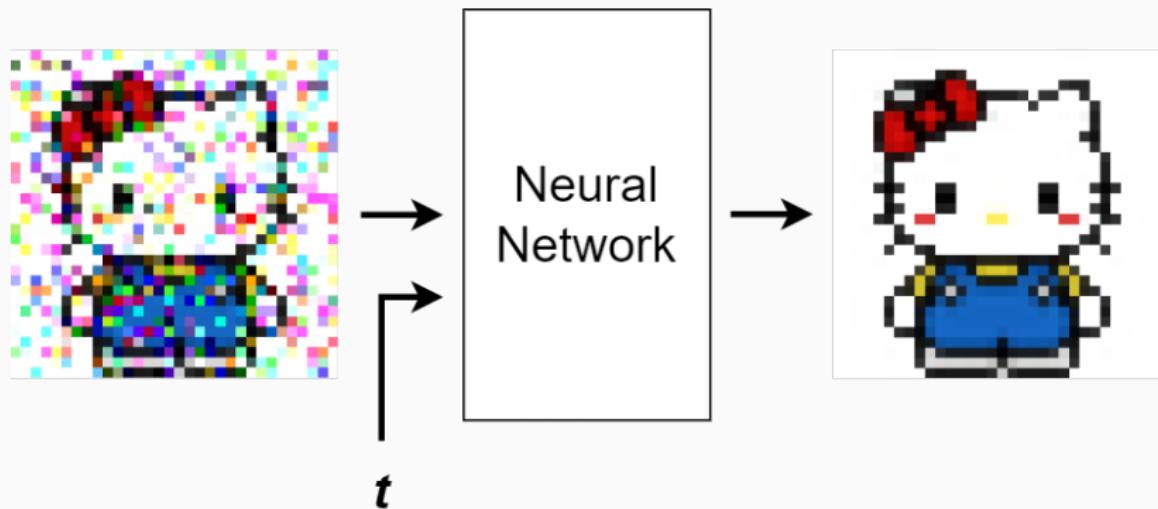


[6]

Redes neuronales y entrenamiento

Para poder realizar el paso de **inverse diffusion** utilizaremos redes neuronales, que serán capaces de calcular el **ruido** de una imagen de entrada.

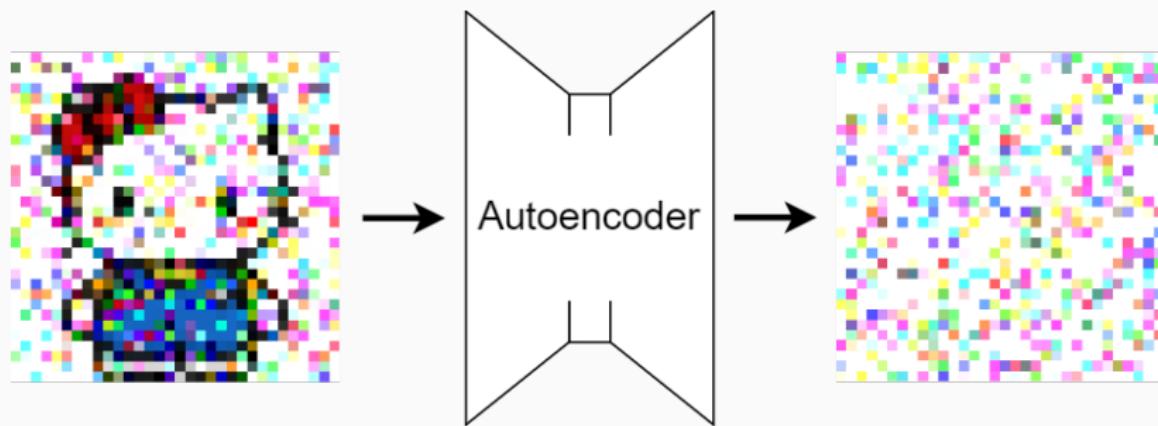
Dichas redes deberán calcular el **ruido** presente en las imágenes que se le presenten como entrada. Para ello es necesario que la red conozca el instante t en el que se encuentra.



Autoencoder para obtener el ruido

La solución más común para este tipo de problemas es el uso de **Autoencoders**. Como se ha visto anteriormente estos modelos generan como salida datos de la misma **dimensionalidad** que los de entrada.

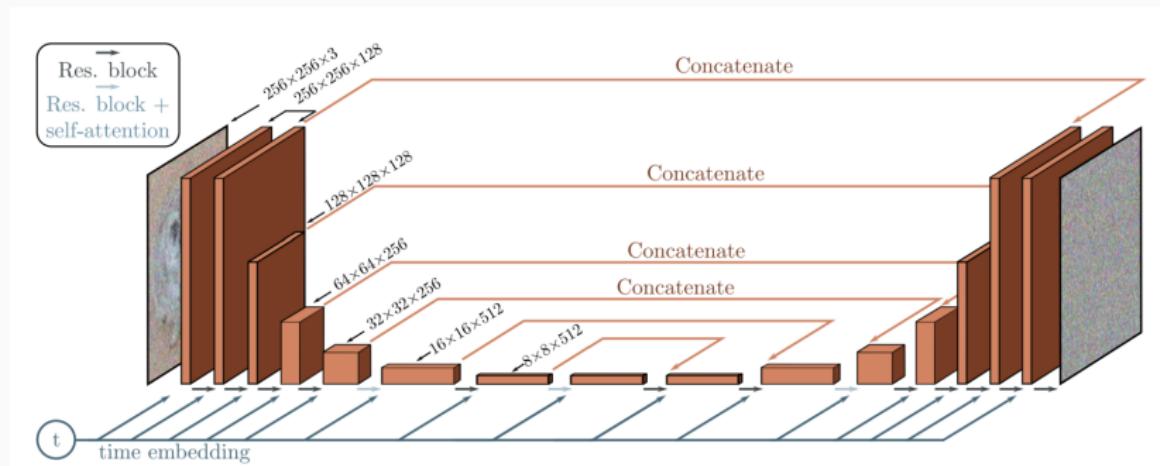
La función del Autoencoder será obtener cuál es el **ruido** de la imagen.



U-Net

La arquitectura **más habitual** para los diffusion models es la U-Net, por sus propiedades a la hora de recuperar información manteniendo la **definición** de las imágenes.

Aunque cabe destacar que dependiendo del **conjunto de datos** que se esté utilizando esto puede cambiar. E.g. diseñar un modelo de difusión para generar música nueva.



Condicionamiento de la salida

Cómo elegir lo que se genera

A veces se desea tener cierto **control** sobre aquello que está generando nuestro modelo. Para ello tenemos que ser capaces de añadir cierta **información extra** a la red de neuronas.

Esta información extra se usará durante el **entrenamiento** para condicionar a la red a aprender la **distribución de datos** de las etiquetas que la condicionen.

De esta manera, cuando se quiera generar nueva información, se usará esas **etiquetas** para generar lo que se quiera.

Context embedding

Esto se realiza a través del conocido como **context embedding**. Este embedding se añade después de cada capa de la misma manera que el **tiempo t** .

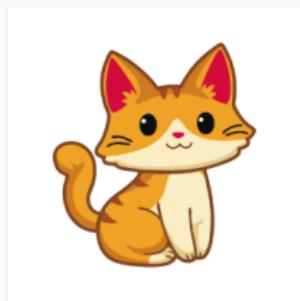
Una de las ventajas de los embeddings es que puedes **combinar información**. Por ejemplo de la siguiente manera:



+



=



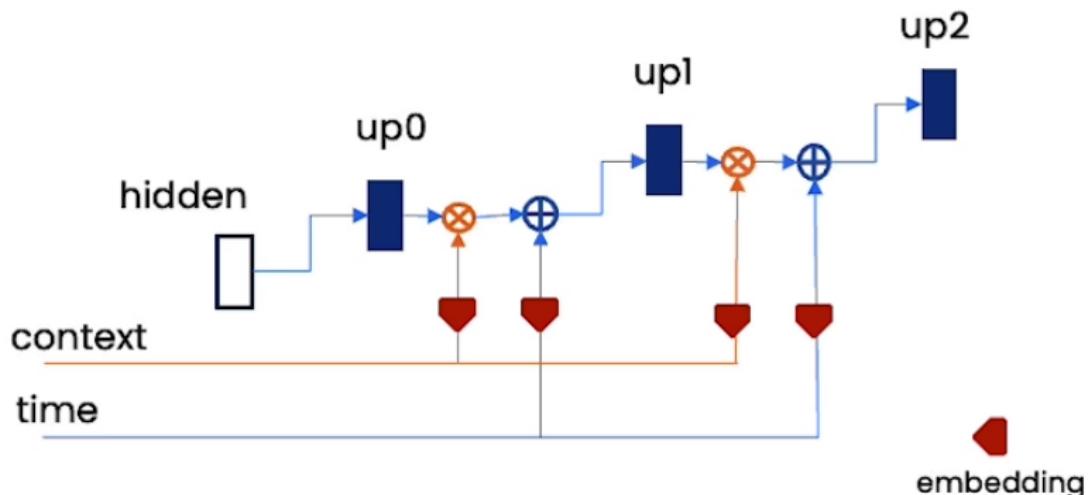
Gato
[1, 0, 0]

Dibujo
[0, 0, 1]

[1, 0, 1]

Context embedding

Este embedding es **alimentado** a la red después de cada bloque convolucional. De esta manera se **condiciona** a la red a que aprenda el contexto de la imagen.



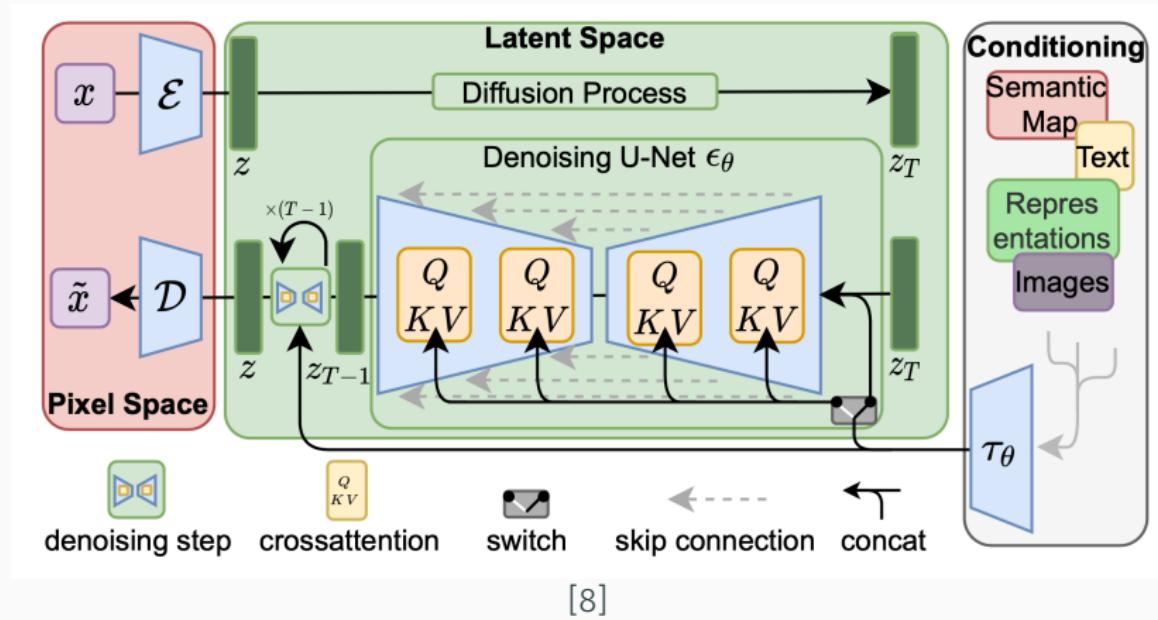
[6]

Combinando con transformers

Si este embedding añadido se realiza con **transformers**, entonces somos capaces, por ejemplo, de controlar la salida a través de texto.

Este tipo de soluciones son las que hoy en dia se utilizan en **dall-e** o **Stable Diffusion**.

Resumen diffusion models

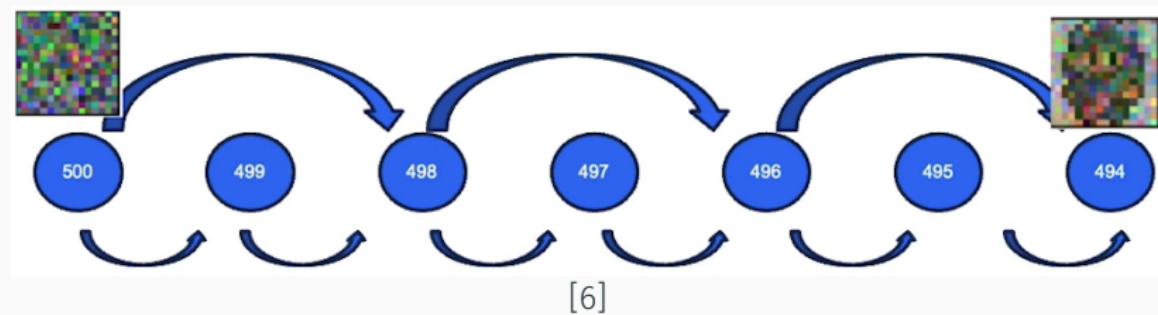


Mejoras en el tiempo de inferencia

Algoritmo DDIM

El algoritmo Denoising Diffusion Implicit Model (DDIM) [9] es una alternativa a DDPM más **rápida y eficiente**.

Se basa en la idea de saltar varios pasos de la **inverse diffusion** de una única vez.



Algoritmo DDIM

La principal característica que permite a DDIM **saltar pasos** del proceso es que no es un modelo probabilístico, si no **determinista**.

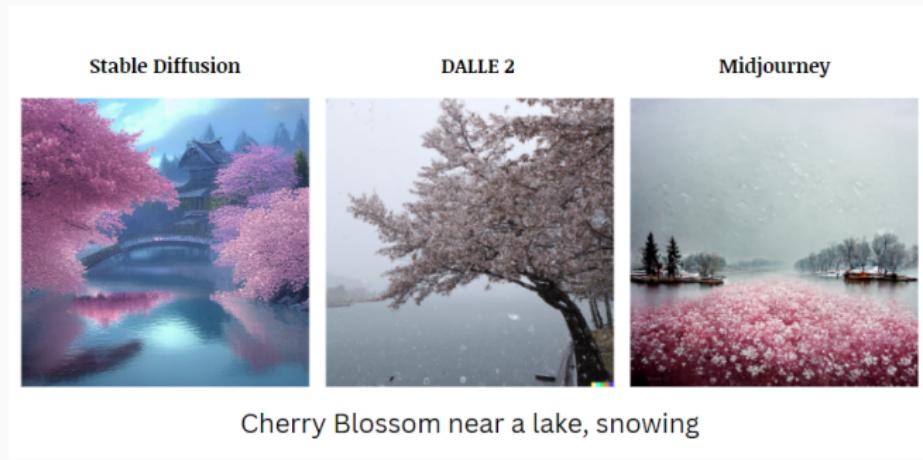
De esta manera se elimina la **aleatoriedad** de las cadenas de Markov.

El algoritmo DDIM llega a ser **10 veces** más eficiente que el DDPM.

Ejemplos y aplicaciones

Generación de imágenes

Quizás las aplicaciones más conocidas de los **diffusion models** son **Stable Diffusion** o **Dall-e 2**.



[10]

Stable Diffusion utiliza una arquitectura llamada **Latent Diffusion Model (LDM)**. Por su parte tanto Midjourney como Dall-e 2 usan tecnologías propietario desconocidas.

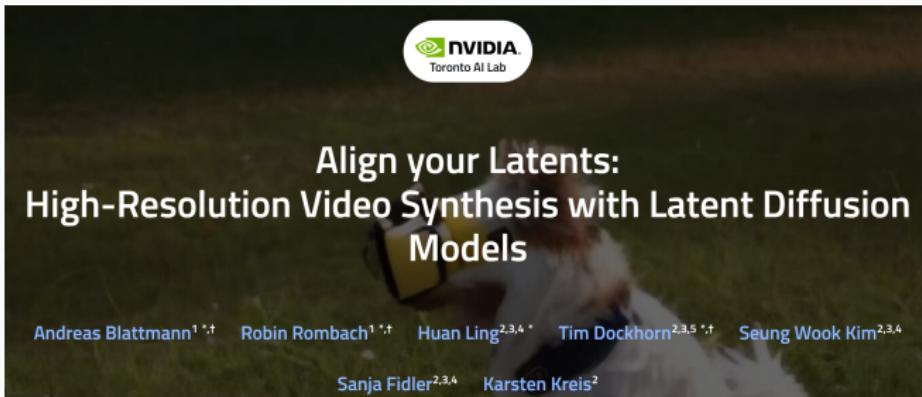
La arquitectura LDM [8] se basa en **comprimir la información** en un espacio latente para realizar con este el proceso de difusión.

De esta manera la **dimensión** de la información tratada se reduce drásticamente. Con ello se permiten mejorar los tiempos de inferencia y entrenamiento.

Generación de video: Align your latents

A parte de generar imágenes, existen alternativas capaces de generar vídeo. Uno de los trabajos más importantes fue publicado por Andreas Blattmann et al., de NVIDIA [11] en el año 2023.

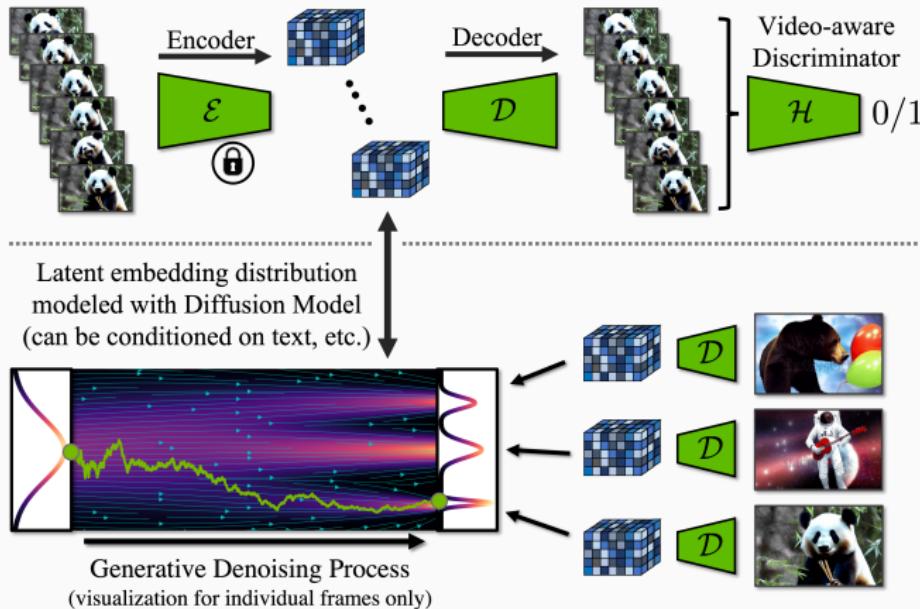
Este trabajo usa la arquitectura LDM introduciendo un **batch de imágenes** que corresponden con un vídeo de 4,7 segundos.



Ejemplos del resultado en la **Web de NVIDIA**

Align your latents

En la arquitectura propuesta se usan ideas también de GAN, como el uso de un **discriminador** para que los videos mantengan fotorealismo.

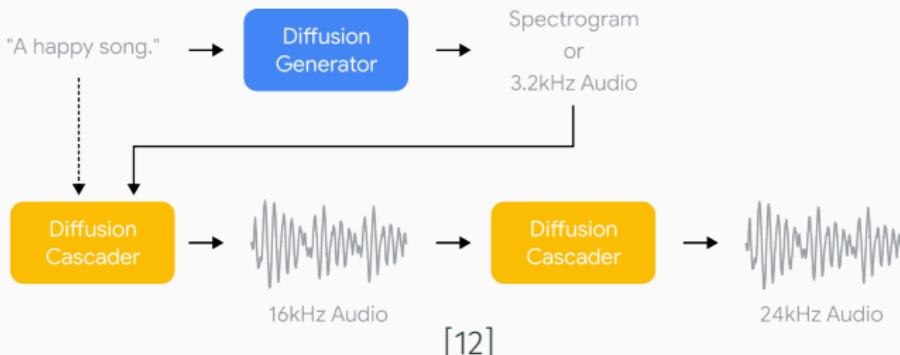


Generación de música: Noise2Music

El trabajo de **Noise2Music** [12] propone el uso de modelos de difusión para generar música a partir de un **texto**.

Para ello se proponen 2 componentes principales:

- El **diffusion generator** se encarga de generar la **base de la música** a baja calidad a partir de la **query de texto**.
- Una serie de **diffusion cascader** que se encargan de mejorar la **calidad** iterativamente.



En el trabajo original proponen dos alternativas para tratar el audio:

- **Ondas sonoras:** Tratan la música como una serie temporal. Para ello se usa una **1D U-Net** en la que las convoluciones observan una única dimensión.
- **Espectrograma:** Una alternativa bastante común para tratar con audio es transformarlo a imagen usando su **espectograma**.

Referencias i

- [1] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli.
Deep unsupervised learning using nonequilibrium thermodynamics.
In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.
- [2] A Vahdat and Karsten Kreis.
Improving diffusion models as an alternative to gans, part 1. nvidia technical blog.
NVIDIA Developer, 2022.

Referencias ii

- [3] J. Rafid Siddiqui (towards data science).
Diffusion model basic scheme.
<https://towardsdatascience.com/diffusion-models-made-easy-8414298ce4da>.
[Online; accessed August, 2023].
- [4] Ayan Das.
Forward diffusion image.
<https://ayandas.me/blog-tut/2021/12/04/diffusion-prob-models.html>.
[Online; accessed August, 2023].
- [5] Jonathan Ho, Ajay Jain, and Pieter Abbeel.
Denoising diffusion probabilistic models.
Advances in neural information processing systems,
33:6840–6851, 2020.

Referencias iii

- [6] Sharon Zhou.
How diffusion models work.
<https://learn.deeplearning.ai/diffusion-models>.
[Online; accessed August, 2023].
- [7] Vaibhav Singh (LearnOpenCV).
U-net architecture image.
<https://learnopencv.com/denoising-diffusion-probabilistic-models/>.
[Online; accessed August, 2023].
- [8] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer.
High-resolution image synthesis with latent diffusion models.
In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

- [9] Jiaming Song, Chenlin Meng, and Stefano Ermon.
Denoising diffusion implicit models.
arXiv preprint arXiv:2010.02502, 2020.
- [10] Bootcamp AI.
Stable diffusion, dall-e 2 midjourney comparison image.
<https://bootcampai.medium.com/c>
[Online; accessed August, 2023].
- [11] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn,
Seung Wook Kim, Sanja Fidler, and Karsten Kreis.
**Align your latents: High-resolution video synthesis with latent
diffusion models.**
In *Proceedings of the IEEE/CVF Conference on Computer Vision
and Pattern Recognition*, pages 22563–22575, 2023.

- [12] Qingqing Huang, Daniel S Park, Tao Wang, Timo I Denk, Andy Ly, Nanxin Chen, Zhengdong Zhang, Zhishuai Zhang, Jiahui Yu, Christian Frank, et al.
Noise2music: Text-conditioned music generation with diffusion models.
arXiv preprint arXiv:2302.03917, 2023.

Contribuciones de las diapositivas

- Autor original de las diapositivas: Guillermo Iglesias Hernández
- Diapositivas basadas en el curso: *How Diffusion Models Work*, de Sharon Zhou.
<https://learn.deeplearning.ai/diffusion-models> [6].