

# **Generación de Imágenes mediante Ecuaciones Diferenciales Estocásticas y Procesos de Difusión**

Lorena Elena Mohanu y José Arbeláez Nieto

*Grado en Ciencia e Ingenieria de Datos*

*Tutor: Alberto Suarez*

11 de mayo de 2025

## **IMPORTANTE**

Todo el código fuente, así como el contenido detallado de las carpetas y módulos desarrollados como parte de este proyecto, se encuentran disponibles de forma pública en el siguiente repositorio de GitHub:

[https://github.com/trentisiete/image\\_generation\\_difussion\\_project](https://github.com/trentisiete/image_generation_difussion_project)

Debido al considerable tamaño total de estos archivos (aproximadamente 2 GB, sin incluir los checkpoints de los modelos entrenados), no ha sido posible adjuntarlos directamente con la presente entrega de este documento.

Se recomienda encarecidamente consultar dicho repositorio para acceder a la implementación completa, los scripts de experimentación, los notebooks interactivos y todos los recursos generados durante el desarrollo del proyecto.

Este proyecto aborda la generación de imágenes sintéticas de alta fidelidad mediante modelos de difusión basados en Ecuaciones Diferenciales Estocásticas (SDEs). Se desarrolló un marco de software modular en Python y PyTorch que permite la implementación, experimentación y evaluación de diversos componentes de estos modelos, incluyendo distintas formulaciones de SDEs (VE-SDE, VP-SDE, Sub-VP SDE), algoritmos de muestreo (Euler-Maruyama, Predictor-Corrector, ODE de flujo de probabilidad) y arquitecturas de redes de score U-Net. El trabajo investiga el impacto de estos componentes en la calidad de la generación de imágenes, utilizando conjuntos de datos como CIFAR-10 y MNIST. La evaluación cuantitativa (FID, IS, BPD) y cualitativa demuestra la capacidad del sistema para generar imágenes diversas y coherentes. Se destaca el rendimiento del modelo SubVP-SDE Lineal en términos de calidad perceptual (FID/IS) y de los modelos SubVP-SDE con schedule cosenoidal en la modelización de la verosimilitud (BPD), evidenciando un compromiso entre ambas métricas. Adicionalmente, se explora la flexibilidad del marco SDE para la generación condicional por clase, guiada por un clasificador dependiente del tiempo, y para tareas de imputación de regiones faltantes en imágenes. Los resultados subrayan la potencialidad del formalismo SDE para la IA generativa, a la vez que se discuten las limitaciones encontradas y se proponen líneas futuras de investigación, como la optimización de samplers y la exploración de técnicas de guiado más avanzadas.

**Palabras clave:** Generación de Imágenes, Modelos de Difusión, Ecuaciones Diferenciales Estocásticas, Aprendizaje Profundo, IA Generativa.

# Índice general

<b>Índice general</b>	<b>4</b>
<b>Índice de figuras</b>	<b>6</b>
<b>Índice de tablas</b>	<b>8</b>
<b>1 Introducción</b>	<b>9</b>
<b>2 Estado del arte y objetivos</b>	<b>11</b>
2.1 Descripción detallada del problema . . . . .	11
2.2 Modelos generativos y procesos de difusión: Fundamentos teóricos . . . . .	11
2.2.1 Modelos de Difusión y el Marco de las SDEs . . . . .	12
2.2.2 Formulaciones Específicas de SDEs . . . . .	13
2.2.3 Probability Flow ODE . . . . .	13
2.3 Trabajos relacionados y literatura relevante . . . . .	13
2.4 Definiciones, conceptos y notación . . . . .	14
2.5 Objetivos del proyecto . . . . .	15
<b>3 Desarrollo de software</b>	<b>18</b>
3.1 Planificación del trabajo . . . . .	18
3.1.1 Diagrama de Gantt . . . . .	18
3.2 Análisis de requisitos . . . . .	20
3.2.1 Requisitos funcionales . . . . .	20
3.2.2 Requisitos no funcionales . . . . .	21
3.2.3 Casos de uso . . . . .	22
3.3 Diseño del software . . . . .	23
3.3.1 Estructura del paquete . . . . .	24
3.3.2 Diagrama de clases . . . . .	25
3.4 Validación y pruebas . . . . .	25
3.4.1 Pruebas unitarias . . . . .	26
3.4.2 Pruebas de integración y funcionales . . . . .	26
3.5 Desarrollo y garantía de calidad de software . . . . .	27
3.6 Otras cuestiones . . . . .	27
3.6.1 Licencia del software . . . . .	27
3.6.2 Implicaciones éticas . . . . .	28
3.6.3 Cuestiones de género . . . . .	28

<b>4 Ejemplos de uso y experimentación</b>	<b>29</b>
4.1 Generación de imágenes y evaluación mediante notebooks interactivos . . . . .	29
4.1.1 Generación Incondicional de Imágenes con CIFAR-10 . . . . .	29
4.1.2 Generación Condicional de Imágenes con CIFAR-10 . . . . .	35
4.1.3 Imputación de Imágenes con CIFAR-10 y Máscaras MNIST . . . . .	39
4.1.4 Análisis Comparativo de la Convergencia del Entrenamiento entre Modelos SDE . . . . .	40
<b>5 Conclusiones</b>	<b>44</b>
<b>Bibliografía</b>	<b>46</b>

# Índice de figuras

3.1	Fases del Plan de Proyecto de IA Generativa, mostrando la distribución temporal de las fases y tareas principales detalladamente. . . . .	19
3.2	Diagrama de Gantt Visual del Plan de Proyecto de IA Generativa, mostrando la distribución temporal de las fases y tareas principales. . . . .	20
4.1	Evolución de la generación de imágenes con VP-SDE Lineal y sampler Predictor-Corrector sobre CIFAR-10. Se muestran instantáneas del proceso en diferentes pasos (de izquierda a derecha) desde el ruido inicial hasta la imagen final. . . . .	30
4.2	Comparación de imágenes finales (CIFAR-10) generadas con el modelo VP-SDE Lineal utilizando diferentes samplers: Predictor-Corrector, Exponential Integrator, Probability Flow ODE y Euler-Maruyama. . . . .	31
4.3	Distribución de FID Scores. . . . .	33
4.4	Distribución de BPD. . . . .	33
4.5	Gráficos de violín mostrando la distribución de los resultados para las métricas FID y BPD entre los diferentes modelos SDE. . . . .	33
4.6	Distribución de Inception Score (Media). . . . .	33
4.7	Distribución de Inception Score (Desv. Est. Interna). . . . .	33
4.8	Gráficos de violín mostrando la distribución de los resultados para la media del Inception Score y su desviación estándar interna. . . . .	33
4.9	Representación esquemática de la arquitectura del clasificador ‘TimeDependentWideResNet’. . . . .	36
4.10	Evolución de la generación condicional para la clase “Barco” en CIFAR-10 utilizando el ‘ConditionalEulerMaruyamaSampler’. Resultados con modelos de score VE-SDE. . . . .	38
4.11	Comparación de imágenes finales generadas condicionalmente para la clase “Barco” con los modelos VE-SDE, VP-SDE Lineal y SubVP-SDE Lineal. .	38
4.12	Proceso y resultado de la imputación de imágenes CIFAR-10 utilizando máscaras generadas a partir de MNIST y un modelo VE-SDE. Fila superior: Imágenes originales de CIFAR-10 con la máscara aplicada (zonas a imputar). Fila intermedia: Imágenes resultantes tras el proceso de imputación. Fila inferior: Imágenes originales de CIFAR-10 como referencia. .	40
4.13	Comportamiento de la función de pérdida durante las primeras 50 épocas de entrenamiento: (a) Curvas del loss promedio por época para los cinco modelos SDE. (b) Diagrama de violín mostrando la distribución de los valores de loss promedio por época acumulados durante este periodo para cada modelo. . . . .	41



# Índice de tablas

4.1 Resumen de métricas de evaluación para diferentes modelos SDE en generación incondicional sobre CIFAR-10. Los valores se presentan como Media ± Desv. Estándar (Número de ejecuciones). Para FID y BPD, menor es mejor. Para IS, mayor es mejor. . . . .	32
4.2 Precisión promedio de los clasificadores ‘TimeDependentWideResNet’ entrenados para diferentes SDEs base sobre CIFAR-10 ruidoso. . . . .	37

# 1 Introducción

La inteligencia artificial generativa ha emergido como una de las áreas más dinámicas y transformadoras de la investigación contemporánea, con la generación de imágenes sintéticas realistas y diversas como uno de sus exponentes más notables. Esta capacidad tiene profundas implicaciones en dominios que abarcan desde la creación artística y el diseño asistido por computadora hasta la medicina, la simulación científica y el aumento de datos para el entrenamiento robusto de otros modelos de aprendizaje automático. En este panorama, los modelos de difusión han ganado una tracción significativa, estableciéndose como un paradigma poderoso capaz de generar muestras de una calidad y fidelidad sin precedentes. Estos modelos conceptualizan la generación como la inversión de un proceso de difusión que gradualmente corrompe los datos con ruido hasta transformarlos en una distribución prior simple, para luego aprender a revertir este proceso, generando datos a partir de muestras de dicha distribución prior.

El presente proyecto se fundamenta en el marco teórico unificador propuesto por Song et al. [16], que formula los modelos de difusión a través del lenguaje de las Ecuaciones Diferenciales Estocásticas (SDEs). Este enfoque considera la transformación de una distribución de datos compleja a una distribución prior conocida (proceso directo o *forward SDE*) y, crucialmente, el proceso inverso que transforma la distribución prior de nuevo en la distribución de datos (proceso inverso o *reverse-time SDE*). La clave de esta inversión reside en la estimación precisa del gradiente de la densidad de probabilidad logarítmica de los datos perturbados en cada instante de tiempo, conocido como el *score* ( $\nabla_x \log p_t(x)$ ). Al aprovechar los avances en el modelado generativo basado en scores, es posible estimar estos scores con redes neuronales y utilizar solucionadores numéricos de SDEs para generar nuevas muestras. Este formalismo no solo engloba y generaliza enfoques previos como el *Score Matching with Langevin Dynamics (SMLD)* [15] y los *Denoising Diffusion Probabilistic Models (DDPM)* [14, 7], sino que también abre la puerta a nuevos procedimientos de muestreo y capacidades de modelado.

Los objetivos principales de este trabajo son los siguientes:

- Desarrollar e implementar un marco de software modular y extensible en Python, utilizando PyTorch, para la generación de imágenes mediante procesos de difusión basados en SDEs, que facilite la experimentación con sus diversos componentes.
- Investigar, implementar y analizar diferentes formulaciones de SDEs para el proceso de difusión, con especial énfasis en la SDE de Varianza Explosiva (VE-SDE), la SDE de Varianza Preservada (VP-SDE) y la SDE Sub-VP, así como sus correspondientes procesos de discretización temporal.
- Implementar, probar y comparar la efectividad y eficiencia de distintos algoritmos de muestreo (*samplers*) para la SDE inversa. Esto incluye integradores numéricos genéricos como Euler-Maruyama, esquemas más sofisticados como los de Predictor-Corrector (PC), y aquellos derivados de la Ecuación Diferencial Ordinaria (ODE) del flujo de probabilidad (*probability flow ODE*).

- Entrenar modelos de estimación de score (redes de score), principalmente arquitecturas U-Net [13], capaces de aproximar con precisión el *score*  $\nabla_x \log p_t(x)$  para diferentes niveles de ruido y tiempos  $t$ .
- Evaluar de forma sistemática, tanto cuantitativa como cualitativamente, la calidad, diversidad y fidelidad de las imágenes generadas. Para ello, se utilizarán métricas estándar en la literatura, como la Distancia de Inicio de Fréchet (FID), el Inception Score (IS) y los Bits Por Dimensión (BPD), esta última calculable de forma exacta mediante la ODE de flujo de probabilidad.
- Explorar y demostrar la flexibilidad del marco SDE para tareas de generación controlada, como la generación condicional por clase y la imputación de regiones faltantes en imágenes, utilizando un único modelo de score incondicional.

Para la consecución de estos objetivos, se realizará una revisión exhaustiva de los fundamentos teóricos, se procederá al diseño e implementación detallada de los componentes software, y se llevará a cabo una fase de experimentación rigurosa utilizando conjuntos de datos de referencia como CIFAR-10 [3] y MNIST [11], analizando los resultados obtenidos en el contexto de los avances más recientes del campo.

La presente memoria está estructurada de la siguiente manera: El Capítulo 2 presenta una revisión del estado del arte en modelos generativos y procesos de difusión, define los conceptos fundamentales y la notación utilizada, y detalla los objetivos específicos del proyecto. El Capítulo 3 describe en profundidad el proceso de desarrollo del software, incluyendo la planificación, el análisis de requisitos, el diseño arquitectónico, las estrategias de validación y pruebas, y las herramientas de garantía de calidad empleadas. El Capítulo 4 ilustra el uso del sistema desarrollado mediante ejemplos prácticos extraídos de los notebooks de experimentación, presentando los resultados obtenidos y comparativas relevantes. Finalmente, el Capítulo 5 resume las principales conclusiones derivadas del trabajo, discute las limitaciones encontradas y propone posibles líneas de investigación futura. Los Apéndices complementan la memoria con material técnico adicional, ejemplos extendidos y comparaciones exhaustivas.

## 2 Estado del arte y objetivos

### 2.1. Descripción detallada del problema

La generación de datos sintéticos que imiten fielmente distribuciones complejas del mundo real representa uno de los desafíos más significativos y estimulantes en el campo del aprendizaje automático y la inteligencia artificial. Particularmente, la generación de imágenes de alta dimensión, como fotografías o ilustraciones, exige modelos capaces de capturar no solo la apariencia visual y los detalles finos (fidelidad), sino también la vasta variabilidad y riqueza semántica inherentes a los datos naturales (diversidad). La dificultad intrínseca radica en modelar distribuciones de probabilidad sobre espacios de muchísimas dimensiones, donde las dependencias entre variables (píxeles, en el caso de imágenes) son intrincadas y altamente no lineales.

Las aplicaciones de modelos generativos de imágenes son extensas y de gran impacto, abarcando desde la creación de contenido artístico y de entretenimiento, el diseño asistido por computadora, la síntesis de nuevas texturas y materiales, hasta el aumento de datos para mejorar el rendimiento de modelos de aprendizaje supervisado en tareas como la clasificación o la detección de objetos, especialmente en escenarios con datos escasos. Además, tienen un potencial considerable en la simulación de entornos virtuales, la restauración de imágenes, la generación de datos médicos sintéticos para investigación y formación, e incluso en el descubrimiento de fármacos mediante la generación de estructuras moleculares.

A pesar de los avances notables, muchos enfoques generativos tradicionales presentan limitaciones. Algunos modelos pueden sufrir de inestabilidad durante el entrenamiento, generar muestras de baja calidad o diversidad (colapso de modos), o carecer de mecanismos para evaluar la probabilidad de las muestras generadas de forma explícita. El problema específico que este proyecto aborda es la investigación y desarrollo de modelos generativos de imágenes que no solo produzcan muestras de alta fidelidad y diversidad, sino que también ofrezcan un marco teórico sólido, un entrenamiento estable y la capacidad de controlar el proceso de generación. En este contexto, los modelos de difusión basados en Ecuaciones Diferenciales Estocásticas (SDEs) han surgido como una alternativa prometedora y potente.

### 2.2. Modelos generativos y procesos de difusión: Fundamentos teóricos

Los modelos generativos tienen como objetivo aprender la distribución de probabilidad subyacente a un conjunto de datos de entrenamiento  $p_{data}(x)$  para posteriormente generar nuevas muestras  $x \sim p_{model}(x)$  que se asemejen a dicha distribución. A lo largo de los años,

diversas familias de modelos generativos han sido propuestas, cada una con sus propias fortalezas y debilidades.

Entre las más destacadas se encuentran las Redes Generativas Antagónicas (GANs) [6], que emplean un juego minimax entre un generador y un discriminador para producir muestras de alta calidad, aunque su entrenamiento puede ser inestable y son propensas al colapso de modos. Los Autocodificadores Variacionales (VAEs) [10] ofrecen un marco de inferencia variacional bayesiana, permitiendo un entrenamiento estable y la obtención de una cota inferior de la verosimilitud (ELBO), pero a menudo generan imágenes con menor nitidez en comparación con las GANs. Los modelos basados en Flujo Normalizante (*Normalizing Flows*) [5, 9] construyen transformaciones invertibles y diferenciables para mapear la distribución de datos a una distribución simple (e.g., Gaussiana), permitiendo el cálculo exacto de la verosimilitud; sin embargo, la necesidad de invertibilidad puede restringir la expresividad de sus arquitecturas.

### 2.2.1. Modelos de Difusión y el Marco de las SDEs

Los modelos de difusión, también conocidos como modelos probabilísticos de difusión y modelos generativos basados en score, han emergido recientemente como una clase de modelos generativos con un rendimiento excepcional. La idea central, inspirada en la termodinámica de no equilibrio, consiste en dos procesos:

1. **Proceso de difusión directo (forward process):** Se perturba gradualmente una muestra de datos  $x(0) \sim p_{data}(x)$  mediante la adición secuencial de pequeñas cantidades de ruido gaussiano a lo largo de una serie de pasos de tiempo  $t \in [0, T]$ . Este proceso es fijo y no se aprende. A medida que  $t \rightarrow T$ , la distribución de los datos perturbados  $p_t(x)$  se aproxima a una distribución prior simple y conocida, típicamente una Gaussiana isotrópica.
2. **Proceso de difusión inverso (reverse process):** Se aprende a revertir el proceso de adición de ruido. Comenzando con una muestra  $x(T)$  de la distribución prior, el modelo genera iterativamente muestras  $x(t)$  para tiempos decrecientes hasta obtener una muestra final  $x(0)$  que se asemeje a los datos originales.

El trabajo seminal de Song et al. [16] generaliza estos procesos al dominio del tiempo continuo mediante Ecuaciones Diferenciales Estocásticas (SDEs), proporcionando un marco unificado y flexible. El proceso directo se modela mediante una SDE de la forma:

$$dx = f(x, t)dt + g(t)dW(t) \quad (2.1)$$

donde  $x(t)$  es el estado del dato en el tiempo  $t$ ,  $f(x, t)$  es el coeficiente de deriva (*drift*),  $g(t)$  es el coeficiente de difusión, y  $W(t)$  es un proceso de Wiener estándar.

Anderson [1] demostró que, bajo ciertas condiciones, este proceso tiene una SDE inversa correspondiente que también es un proceso de difusión, dado por:

$$dx = [f(x, t) - g(t)^2 \nabla_x \log p_t(x)]dt + g(t)d\bar{W}(t) \quad (2.2)$$

donde  $d\bar{W}(t)$  es un proceso de Wiener estándar en tiempo inverso, y  $dt$  es un paso de tiempo infinitesimal negativo. El término crucial  $\nabla_x \log p_t(x)$  es el *score* de la distribución de datos perturbados  $p_t(x)$  en el tiempo  $t$ .

La tarea del modelo generativo,  $s_\theta(x, t)$ , es estimar este score. Esto se logra típicamente mediante técnicas de *score matching*, como el *denoising score matching* [17], que minimiza

la diferencia esperada entre el score predicho y el score real de la distribución condicional de los datos perturbados dado el dato original. El objetivo de entrenamiento generalizado a tiempo continuo es:

$$\theta^* = \arg \min_{\theta} \mathbb{E}_{t \sim \mathcal{U}(0,T)} \left\{ \lambda(t) \mathbb{E}_{x(0) \sim p_{data}} \mathbb{E}_{x(t) | x(0)} \left[ \| s_{\theta}(x(t), t) - \nabla_{x(t)} \log p_{0t}(x(t) | x(0)) \|_2^2 \right] \right\} \quad (2.3)$$

donde  $p_{0t}(x(t) | x(0))$  es la densidad de transición del proceso directo y  $\lambda(t)$  es una función de ponderación positiva.

## 2.2.2. Formulaciones Específicas de SDEs

Dentro de este marco, Song et al. [16] identifican y analizan varias SDEs importantes, que generalizan trabajos previos:

- **Variance Exploding (VE) SDE:** Dada por  $dx = \sqrt{\frac{d[\sigma^2(t)]}{dt}} dW(t)$ . En este caso,  $f(x, t) = 0$  y  $g(t) = \sqrt{\frac{d[\sigma^2(t)]}{dt}}$ . La varianza de  $x(t)$  crece con el tiempo (explota). Esta SDE generaliza los modelos NCSN/SMLD [15]. La densidad de transición  $p_{0t}(x(t) | x(0))$  es  $\mathcal{N}(x(t); x(0), [\sigma^2(t) - \sigma^2(0)]I)$ .
- **Variance Preserving (VP) SDE:** Dada por  $dx = -\frac{1}{2}\beta(t)x(t)dt + \sqrt{\beta(t)}dW(t)$ . Aquí,  $f(x, t) = -\frac{1}{2}\beta(t)x(t)$  y  $g(t) = \sqrt{\beta(t)}$ . Si  $x(0)$  tiene varianza unitaria,  $x(t)$  mantiene la varianza unitaria para todo  $t$ . Esta SDE generaliza los modelos DDPM [14, 7]. La densidad de transición  $p_{0t}(x(t) | x(0))$  es  $\mathcal{N}(x(t); x(0)e^{-\frac{1}{2}\int_0^t \beta(s)ds}, [1 - e^{-\int_0^t \beta(s)ds}]I)$ .
- **Sub-Variance Preserving (sub-VP) SDE:** Una variante propuesta en [16], dada por  $dx = -\frac{1}{2}\beta(t)x(t)dt + \sqrt{\beta(t)(1 - e^{-2\int_0^t \beta(s)ds})}dW(t)$ . Esta SDE tiende a ofrecer mejores resultados en términos de verosimilitud.

## 2.2.3. Probability Flow ODE

Una propiedad fundamental del marco SDE es que existe una Ecuación Diferencial Ordinaria (ODE) determinista, conocida como la *probability flow ODE*, cuyas trayectorias comparten las mismas densidades marginales  $\{p_t(x)\}_{t=0}^T$  que la SDE. Esta ODE está dada por:

$$dx = \left[ f(x, t) - \frac{1}{2}g(t)^2 \nabla_x \log p_t(x) \right] dt \quad (2.4)$$

Al reemplazar  $\nabla_x \log p_t(x)$  con el modelo de score entrenado  $s_{\theta}(x, t)$ , se obtiene una ODE neuronal [2]. La integración de esta ODE permite la generación determinista de muestras, la manipulación de representaciones latentes y, crucialmente, el cálculo exacto de la verosimilitud de los datos mediante la fórmula instantánea de cambio de variables.

## 2.3. Trabajos relacionados y literatura relevante

La investigación en modelos de difusión y generativos basados en score ha sido muy activa, construyendo sobre varias décadas de trabajo en modelado estadístico y aprendizaje automático.

Los fundamentos de los modelos de difusión se remontan al trabajo de Sohl-Dickstein et al. (2015) [14], quienes introdujeron los "Modelos Probabilísticos de Difusión y Denoising"(DDPMs) inspirados en la termodinámica de no equilibrio. Posteriormente, Song y Ermon (2019) [15] propusieron las Redes de Score Condicionadas al Ruido"(NCSN) y el muestreo mediante "Dinámica de Langevin Recocida"(SMLD), basándose en la estimación de scores. Ho et al. (2020) [7] simplificaron y mejoraron significativamente los DDPMs, logrando una calidad de generación de imágenes que rivalizaba con las GANs.

La estimación del score ( $\nabla_x \log p(x)$ ) es un concepto central. Hyvärinen (2005) [8] introdujo el principio de *score matching* para estimar modelos de probabilidad no normalizados. Vincent (2011) [17] estableció una conexión fundamental entre el *score matching* y los autocodificadores con eliminación de ruido (*denoising autoencoders*), lo que simplificó enormemente el entrenamiento de modelos de score.

El trabajo de Song et al. (2021) [16] representa un hito al unificar los enfoques SMLD y DDPM bajo el formalismo de las SDEs. Este marco no solo proporcionó una comprensión más profunda, sino que también introdujo:

- La distinción explícita entre VE-SDE y VP-SDE, así como la propuesta de la sub-VP SDE.
- Nuevos algoritmos de muestreo, como los muestreadores Predictor-Corrector (PC), que combinan solucionadores numéricos de SDEs con pasos de corrección basados en MCMC (como Langevin MCMC).
- La formulación de la *probability flow ODE*, que habilita el cálculo exacto de la verosimilitud y la generación determinista.
- Métodos para la generación controlada (condicional, imputación, colorización) utilizando un único modelo de score incondicional.

Posteriormente, Nichol y Dhariwal (2021) [12] propusieron mejoras adicionales a los DDPMs, como el uso de un *schedule* de ruido cosenoidal y el aprendizaje de la varianza del proceso de difusión inverso, logrando mejoras en la calidad de las muestras y en la eficiencia del muestreo.

Las arquitecturas de red neuronal utilizadas para modelar el score  $s_\theta(x, t)$  son predominantemente variantes de la U-Net [13], debido a su eficacia en tareas de segmentación de imágenes y su capacidad para capturar información en múltiples escalas. Las implementaciones modernas de modelos de difusión suelen emplear arquitecturas U-Net profundas con mecanismos de atención y embeddings para el tiempo  $t$ .

La investigación continúa activa en áreas como la aceleración del muestreo (que puede ser lento debido al gran número de pasos de discretización), la mejora de la calidad de generación en resoluciones muy altas, la extensión a otros tipos de datos (audio, vídeo, grafos), y la exploración de nuevas aplicaciones y aspectos teóricos.

## 2.4. Definiciones, conceptos y notación

Para asegurar la claridad y consistencia a lo largo de esta memoria, se establece la siguiente notación y se definen los conceptos clave, en gran medida alineados con [16]:

- $x_0$  o  $x(0)$ : Una muestra de la distribución de datos reales  $p_{data}(x)$ .
- $x(t)$ : El estado de la muestra de datos en el tiempo  $t \in [0, T]$  durante el proceso de difusión.

- $p_t(x)$ : La densidad de probabilidad marginal de  $x(t)$ .  $p_0(x) \equiv p_{data}(x)$ , y  $p_T(x)$  es la distribución prior (e.g.,  $\mathcal{N}(0, I)$ ).
  - $p_{st}(x(t)|x(s))$ : La densidad de transición del proceso de difusión de  $x(s)$  a  $x(t)$  para  $s < t$ .
  - $dW(t), d\bar{W}(t)$ : Incrementos infinitesimales de un proceso de Wiener estándar (movimiento Browniano) en tiempo hacia adelante y hacia atrás, respectivamente.
  - $f(x, t)$ : El coeficiente de deriva (drift) de la SDE directa.
  - $g(t)$ : El coeficiente de difusión de la SDE directa.
  - $\nabla_x \log p_t(x)$ : El score (o campo de gradiente) de la densidad logarítmica de  $p_t(x)$ .
  - $s_\theta(x, t)$ : El modelo de score parametrizado por  $\theta$ , que aproxima  $\nabla_x \log p_t(x)$ .
  - $\sigma(t)$ : El nivel de ruido en el tiempo  $t$  para las VE-SDEs.  $\sigma_{min}, \sigma_{max}$  son los niveles de ruido mínimo y máximo.
  - $\beta(t)$ : La función de schedule de ruido para las VP-SDEs y sub-VP SDEs.  $\beta_{min}, \beta_{max}$  son los valores mínimo y máximo de  $\beta(t)$  (o de una función lineal relacionada).
  - $\alpha_t = \exp(-\int_0^t \beta(s)ds)$  y  $\bar{\alpha}_t = \prod_{i=1}^t (1 - \beta_i)$  en contextos discretos (DDPM). La notación puede variar ligeramente si se refiere a la  $\alpha$  acumulada. En el contexto de VP-SDE continuas, el factor de escalamiento de la media es  $e^{-\frac{1}{2} \int_0^t \beta(s)ds}$ .
  - SDE: Ecuación Diferencial Estocástica.
  - ODE: Ecuación Diferencial Ordinaria.
  - VE-SDE: Variance Exploding SDE.
  - VP-SDE: Variance Preserving SDE.
  - sub-VP SDE: Sub-Variance Preserving SDE.
  - Sampler: Algoritmo numérico para resolver la SDE inversa o la ODE de flujo de probabilidad.
    - Euler-Maruyama: Un método numérico básico para SDEs.
    - Predictor-Corrector (PC): Samplers que combinan un paso de predicción (solver SDE) y un paso de corrección (MCMC basado en score).
  - Métricas de Evaluación:
    - FID: Fréchet Inception Distance. Mide la similitud entre las distribuciones de características de imágenes reales y generadas. Menor es mejor.
    - IS: Inception Score. Mide la calidad (nitidez) y diversidad de las imágenes generadas. Mayor es mejor.
    - NLL: Negative Log-Likelihood (Verosimilitud Logarítmica Negativa). Mide qué tan bien el modelo asigna probabilidad a los datos de prueba. Menor es mejor.
    - BPD: Bits Per Dimension. NLL normalizada por la dimensionalidad de los datos;  $BPD = NLL/(\log(2) \times D)$ , donde  $D$  es el número de dimensiones. Menor es mejor.
  - U-Net: Arquitectura de red neuronal convolucional con conexiones skip, comúnmente utilizada para  $s_\theta(x, t)$ .
- Cualquier otra notación específica será introducida en el contexto donde se utilice.

## 2.5. Objetivos del proyecto

Basándose en la descripción del problema y el estado del arte actual, el objetivo fundamental de este proyecto es el estudio, desarrollo, implementación y evaluación de un

sistema de generación de imágenes de alta fidelidad basado en el marco de Ecuaciones Diferenciales Estocásticas (SDEs) para modelos de difusión. Se busca no solo replicar y comprender las técnicas existentes, sino también explorar y analizar sistemáticamente el impacto de sus diferentes componentes.

Los objetivos específicos que se persiguen en este trabajo son los siguientes:

**1. Estudio e Implementación de Formulaciones de SDEs:**

- Investigar en profundidad y desarrollar implementaciones funcionales de las principales SDEs para modelos de difusión: Variance Exploding (VE-SDE), Variance Preserving (VP-SDE) y Sub-Variance Preserving (sub-VP SDE).
- Analizar las propiedades teóricas de cada SDE y sus implicaciones en el proceso de difusión y generación.

**2. Desarrollo y Evaluación de Algoritmos de Muestreo (Samplers):**

- Implementar y validar diversos algoritmos de muestreo para resolver la SDE inversa y la ODE de flujo de probabilidad. Esto incluye:
  - Integradores numéricos genéricos (e.g., Euler-Maruyama).
  - Muestreadores de Predictor-Corrector (PC), explorando diferentes combinaciones de predictores y correctores.
  - Solucionadores para la ODE de flujo de probabilidad.
- Comparar los samplers en términos de calidad de las muestras generadas, velocidad de muestreo y estabilidad numérica.

**3. Entrenamiento de Modelos de Score Eficientes:**

- Implementar y entrenar redes neuronales, específicamente arquitecturas basadas en U-Net, para estimar el score  $\nabla_x \log p_t(x)$ .
- Investigar el impacto de diferentes configuraciones arquitectónicas y estrategias de entrenamiento en la precisión de la estimación del score y, consecuentemente, en la calidad de la generación.

**4. Evaluación Rigurosa de la Calidad de Generación:**

- Evaluar cuantitativamente las imágenes generadas utilizando un conjunto de métricas estándar y reconocidas en la comunidad científica: Fréchet Inception Distance (FID), Inception Score (IS), y Negative Log-Likelihood (NLL) o Bits Per Dimension (BPD), esta última aprovechando la capacidad de cálculo exacto mediante la ODE de flujo de probabilidad.
- Realizar una evaluación cualitativa de la fidelidad visual y diversidad de las muestras generadas.

**5. Exploración de Aplicaciones de Generación Controlada (si el tiempo y los recursos lo permiten):**

- Investigar y, de ser posible, implementar técnicas de generación condicional por clase.
- Explorar la aplicación del marco SDE a tareas de imputación de regiones faltantes en imágenes.

**6. Creación de un Marco de Software Modular y Documentado:**

- Desarrollar una librería de software en Python, utilizando principalmente la biblioteca PyTorch, que sea modular, reutilizable y bien documentada, facilitando la experimentación y la extensión futura del trabajo.
- Asegurar la reproducibilidad de los experimentos mediante una gestión adecuada del código y los entornos.

Se espera que la consecución de estos objetivos contribuya a una comprensión más profunda de los modelos de difusión basados en SDEs y proporcione un conjunto de herramientas robustas para la investigación y aplicación en el campo de la IA generativa.

# 3 Desarrollo de software

Este capítulo detalla el proceso seguido para el desarrollo del software del proyecto, abarcando desde la planificación inicial y el análisis de requisitos, hasta el diseño de la arquitectura, las estrategias de calidad y otras consideraciones relevantes.

## 3.1. Planificación del trabajo

La gestión del proyecto se abordó mediante un enfoque Scrum, lo que permitió flexibilidad para adaptar los objetivos y tareas a medida que se profundizaba en la investigación y la implementación. El desarrollo se estructuró en varias fases principales:

- **Fase 1: Investigación y Fundamentación Teórica:** Revisión exhaustiva de la literatura científica sobre modelos de difusión, Ecuaciones Diferenciales Estocásticas (SDEs), técnicas de *score matching* y arquitecturas de redes neuronales relevantes. Consolidación de la comprensión de los algoritmos y conceptos clave.
- **Fase 2: Diseño y Definición de la Arquitectura del Software:** Establecimiento de la estructura general del proyecto, definición de módulos principales, interfaces entre componentes y selección de las tecnologías y librerías base (Python, PyTorch).
- **Fase 3: Implementación de Componentes Centrales:** Desarrollo de las clases para las SDEs (VE, VP, Sub-VP), los diferentes *samplers* (Euler-Maruyama, Predictor-Corrector, ODE), los modelos de *score* (U-Net) y las métricas de evaluación.
- **Fase 4: Implementación de Funcionalidades Adicionales:** Desarrollo de los *samplers* condicionales y el módulo de imputación de imágenes.
- **Fase 5: Experimentación y Evaluación:** Entrenamiento de los modelos de *score* sobre los datasets seleccionados (e.g., CIFAR-10, MNIST), generación de imágenes y evaluación cuantitativa y cualitativa de los resultados.
- **Fase 6: Documentación y Elaboración de la Memoria:** Redacción de la presente memoria, documentación del código y preparación del material final del proyecto.

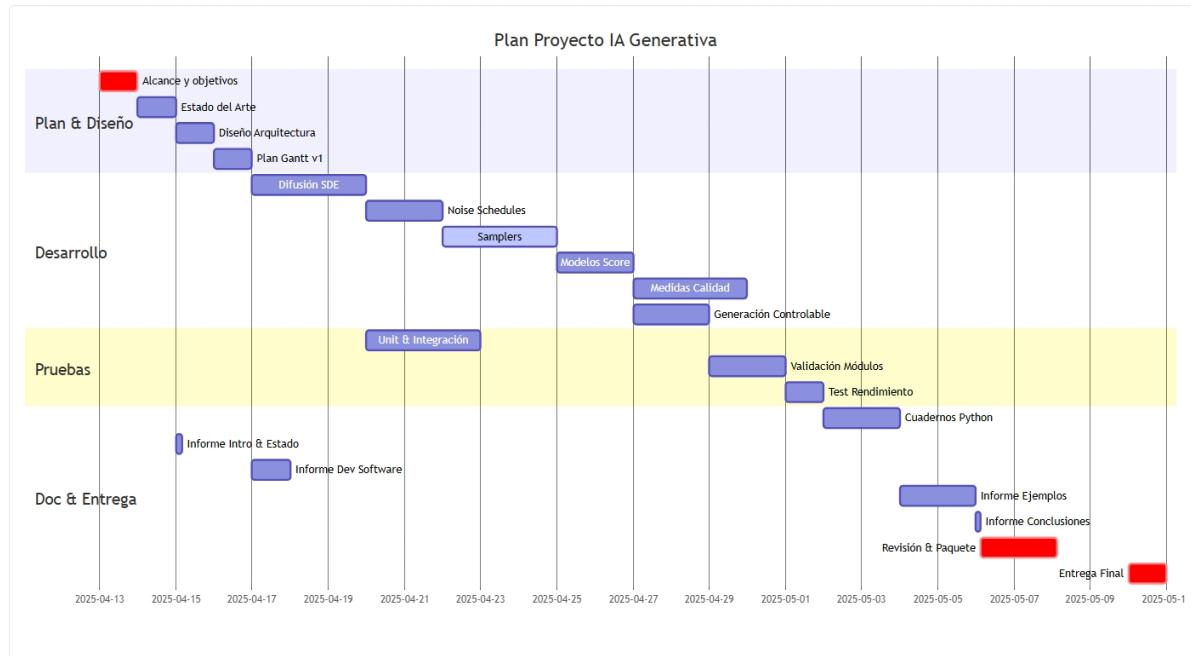
Para el seguimiento de tareas y la organización del trabajo, se utilizó GitHub y WhatsApp.

### 3.1.1. Diagrama de Gantt

Un diagrama de Gantt es una herramienta gráfica cuyo objetivo es exponer el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado. En la Figura 3.2 se presenta el diagrama de Gantt elaborado para la planificación y seguimiento de este proyecto. En él se visualizan las principales fases descritas en la sección anterior (Planificación y Diseño, Desarrollo del Sistema, Pruebas y Validación, y Documentación y Entrega), junto con las tareas más relevantes de cada una, su duración estimada, fechas de inicio y fin, y las interdependencias existentes.

ID Tarea	Descripción de la Tarea	Dependencias	Fecha Inicio Estimada	Duración	Fecha Fin Estimada	Responsable (Ejemplo)	Notas Adicionales
<b>Fase 1: Planificación y Diseño</b>							
obj	Alcance y definición detallada de objetivos	-	2025-04-13	1 día	2025-04-13	Jefe de Proyecto	Definir entregables, KPIs y alcance.
soa	Investigación y Estado del Arte	obj	2025-04-14	1 día	2025-04-14	Equipo de Investigación	Análisis de papers y soluciones existentes.
arch	Diseño de la Arquitectura del Sistema	soa	2025-04-15	1 día	2025-04-15	Arquitecto de Software	Diagrama de clases, componentes.
g1	Planificación detallada (Plan Gantt v1)	arch	2025-04-16	1 día	2025-04-16	Jefe de Proyecto	Ajuste fino de tareas y cronograma.
<b>Fase 2: Desarrollo del Sistema</b>							
diff	Implementación: Procesos de Difusión SDE (VE, VP, Sub-VP)	g1	2025-04-17	3 días	2025-04-19	Desarrollador Principal	Módulos base de difusión.
noise	Implementación: Noise Schedules (Linear, Cosine)	diff	2025-04-20	2 días	2025-04-21	Desarrollador	
samp	Implementación: Samplers (Euler-Maruyama, PC, ODE, Exp)	noise	2025-04-22	3 días	2025-04-24	Desarrollador Principal	
model	Implementación: Modelos de Score (ScoreNet, U-Net)	samp	2025-04-25	2 días	2025-04-26	Desarrollador IA	Entrenamiento y ajuste inicial.
metrics	Implementación: Medidas de Calidad (BPD, FID, IS)	model	2025-04-27	3 días	2025-04-29	Desarrollador IA	Scripts para evaluación.
ctrl	Implementación: Generación Controlable (Colorization, Class-cond, Imputation)	model	2025-04-27	2 días	2025-04-28	Desarrollador IA	Adaptación de modelos para control.
<b>Fase 3: Pruebas y Validación</b>							
test	Pruebas Unitarias y de Integración	diff	2025-04-20	3 días	2025-04-22	Equipo QA	Pruebas continuas post-módulo Difusión SDE.
valid	Validación de resultados de cada módulo	ctrl, metrics	2025-04-30	2 días	2025-05-01	Equipo QA / JP	Verificar calidad y funcionalidad. 2025-05-01 puede ser festivo (Día del Trabajador).
perf	Pruebas de Rendimiento del sistema	valid	2025-05-02	1 día	2025-05-02	Equipo QA	Evaluar eficiencia y tiempos.
<b>Fase 4: Documentación y Entrega</b>							
intro	Redacción Informe: Introducción y Estado del Arte	soa	2025-04-15	4 horas	2025-04-15	Responsable Documentación	Se completa durante Diseño Arq.
dev	Redacción Informe: Desarrollo de Software	g1	2025-04-17	1 día	2025-04-17	Responsable Documentación	Comienza con la fase de desarrollo.
notes	Desarrollo de Cuadernos Python (ejemplos de uso)	perf	2025-05-03	2 días	2025-05-04	Desarrollador Principal	Notebooks ilustrativos.
ex	Redacción Informe: Ejemplos de Uso	notes	2025-05-05	2 días	2025-05-06	Responsable Documentación	
conc	Redacción Informe: Conclusiones y Resumen	ex	2025-05-07	3 horas	2025-05-07	Jefe de Proyecto	Se completa en 1 día.
rev	Revisión final del proyecto y paquete de software	conc	2025-05-08	2 días	2025-05-09	Equipo Completo	Revisión de código, informe y notebooks.
deliver	Entrega Final del Proyecto	-	2025-05-10	1 día	2025-05-10	Jefe de Proyecto	Fecha Hito.

**Figura 3.1:** Fases del Plan de Proyecto de IA Generativa, mostrando la distribución temporal de las fases y tareas principales detalladamente.



**Figura 3.2:** Diagrama de Gantt Visual del Plan de Proyecto de IA Generativa, mostrando la distribución temporal de las fases y tareas principales.

Este diagrama sirvió como hoja de ruta para la ejecución del proyecto, permitiendo una gestión visual del progreso y la identificación temprana de posibles desviaciones respecto a la planificación inicial. Las barras horizontales representan la duración de cada tarea, mientras que el eje temporal permite situar cada actividad en el calendario del proyecto.

## 3.2. Análisis de requisitos

A continuación, se detallan los requisitos funcionales y no funcionales que el software desarrollado debe cumplir, así como los principales casos de uso.

### 3.2.1. Requisitos funcionales

Los requisitos funcionales definen las capacidades específicas que el sistema debe proporcionar al usuario. Basado en la estructura del proyecto y los objetivos planteados, estos son:

- **RF1: Implementación de Procesos de Difusión SDE:**
  - El sistema debe implementar las SDEs de Varianza Explosiva (VE-SDE), Varianza Preservada (VP-SDE) y Sub-Varianza Preservada (Sub-VP SDE), incluyendo sus coeficientes de deriva y difusión.
  - Debe permitir el cálculo de las medias y varianzas de las distribuciones de transición  $p_{0t}(x(t)|x(0))$  para estas SDEs.
  - Debe incluir la lógica para el proceso de SDE inverso (backward SDE) utilizando un modelo de score.
- **RF2: Implementación de Schedules de Ruido:**

- El sistema debe proveer diferentes *schedules* para la variación del ruido/varianza, específicamente *schedules* lineales y cosenoidales para  $\beta(t)$  o  $\sigma(t)$ .
- **RF3: Implementación de Algoritmos de Muestreo (Samplers):**
  - El sistema debe implementar el integrador de Euler-Maruyama para la SDE inversa.
  - El sistema debe implementar muestreadores de Predictor-Corrector (PC), permitiendo configurar el número de pasos de predicción y corrección.
  - El sistema debe implementar un integrador para la ODE de flujo de probabilidad (e.g., basado en Euler exponencial o un solver ODE genérico).
- **RF4: Implementación y Entrenamiento de Modelos de Score:**
  - El sistema debe incluir implementaciones de modelos de score basados en arquitecturas U-Net (e.g., ScoreNet, NCSN++ style).
  - Debe permitir el entrenamiento de estos modelos utilizando una función de pérdida basada en *denoising score matching*.
  - Debe gestionar embeddings de tiempo (e.g., Gaussian Random Fourier Features).
- **RF5: Implementación de Muestreadores Condicionales:**
  - El sistema debe soportar la generación condicional de imágenes mediante el uso de un clasificador auxiliar para guiar el proceso de muestreo con los samplers Euler-Maruyama y Predictor-Corrector, tanto para SDEs VE como VP/Sub-VP.
- **RF6: Evaluación de la Calidad de Imágenes Generadas:**
  - El sistema debe permitir calcular la métrica de Bits Por Dimensión (BPD) y la NLL a través de la integración de la ODE de flujo de probabilidad.
  - El sistema debe permitir calcular la métrica Fréchet Inception Distance (FID) entre un conjunto de imágenes generadas y un conjunto de imágenes reales.
  - El sistema debe permitir calcular el Inception Score (IS) para un conjunto de imágenes generadas.
- **RF7: Funcionalidad de Imputación de Imágenes:**
  - El sistema debe implementar un *sampler* para la imputación de regiones faltantes en una imagen, condicionado a los píxeles conocidos y utilizando un modelo de score entrenado.
- **RF8: Utilidades de Visualización y Soporte:**
  - El sistema debe proporcionar funciones para visualizar imágenes generadas, cuadrículas de imágenes y la evolución del proceso de generación a lo largo del tiempo.

### 3.2.2. Requisitos no funcionales

Los requisitos no funcionales describen los atributos de calidad y las restricciones bajo las cuales el sistema debe operar:

- **RNF1: Rendimiento:** El software debe hacer un uso eficiente de los recursos computacionales, especialmente la GPU para el entrenamiento de modelos y la generación de muestras. Los tiempos de generación deben ser razonables para la experimentación.
- **RNF2: Modularidad:** El código debe estar organizado en módulos cohesivos y

débilmente acoplados, facilitando su comprensión, mantenimiento y reutilización (e.g., separación clara entre SDEs, samplers, modelos, métricas).

- **RNF3: Extensibilidad:** La arquitectura del software debe permitir la fácil incorporación de nuevas SDEs, algoritmos de muestreo, arquitecturas de modelos de score o métricas de evaluación con un impacto mínimo en los componentes existentes.
- **RNF4: Usabilidad:** La API de los componentes principales debe ser clara e intuitiva. Debe ser posible configurar y ejecutar experimentos de forma sencilla, preferiblemente mediante scripts o notebooks interactivos.
- **RNF5: Robustez y Correctitud Numérica:** Las implementaciones de los algoritmos matemáticos y los procesos estocásticos deben ser correctas y numéricamente estables.
- **RNF6: Documentación:** El código debe estar adecuadamente comentado, y se debe proporcionar documentación sobre la estructura del proyecto y el uso de sus componentes.
- **RNF7: Portabilidad:** El software debe ser compatible con entornos de desarrollo comunes (e.g., sistemas operativos Linux) y depender de librerías estándar y ampliamente disponibles en el ecosistema de Python (PyTorch, NumPy, SciPy).

### 3.2.3. Casos de uso

Los casos de uso describen interacciones típicas de un usuario (investigador/desarrollador) con el sistema para alcanzar objetivos específicos:

- **CU1: Generación Incondicional de Imágenes.**
  - *Actor:* Investigador.
  - *Descripción:* El investigador desea generar un lote de imágenes sintéticas a partir de ruido aleatorio utilizando una SDE específica (e.g., VPSDE), un modelo de score pre-entrenado y un algoritmo de muestreo configurado (e.g., Predictor-Corrector con  $N$  pasos).
  - *Flujo básico:*
    1. El investigador configura los parámetros de la SDE, el sampler y carga el modelo de score.
    2. El investigador invoca la función de muestreo especificando las dimensiones deseadas para las imágenes.
    3. El sistema ejecuta el proceso de difusión inverso y devuelve las imágenes generadas.
    4. El investigador visualiza o guarda las imágenes.
- **CU2: Entrenamiento de un Modelo de Score.**
  - *Actor:* Investigador.
  - *Descripción:* El investigador desea entrenar un nuevo modelo de score (e.g., una U-Net) sobre un conjunto de datos de imágenes (e.g., CIFAR-10) para una SDE particular.
  - *Flujo básico:*
    1. El investigador prepara el conjunto de datos y configura los hiperparámetros de entrenamiento (tasa de aprendizaje, tamaño de batch, número de épocas), la SDE a utilizar y la arquitectura del modelo de score.
    2. El investigador inicia el proceso de entrenamiento.

3. El sistema itera sobre los datos, calcula la función de pérdida de *denoising score matching* y actualiza los pesos del modelo.
  4. El sistema guarda los checkpoints del modelo entrenado.
- **CU3: Evaluación de la Calidad de Imágenes Generadas.**
    - *Actor:* Investigador.
    - *Descripción:* El investigador desea evaluar la calidad de un conjunto de imágenes generadas previamente, utilizando métricas como FID, IS o BPD, comparándolas (si aplica) con un conjunto de datos de referencia.
    - *Flujo básico:*
      1. El investigador carga las imágenes generadas y las imágenes de referencia (para FID).
      2. El investigador selecciona la métrica a calcular y configura sus parámetros.
      3. El sistema calcula el valor de la métrica.
      4. El investigador registra el resultado.
  - **CU4: Generación Condicional de Imágenes por Clase.**
    - *Actor:* Investigador.
    - *Descripción:* El investigador desea generar imágenes condicionadas a una etiqueta de clase específica (e.g., generar imágenes de "gatos" de CIFAR-10), utilizando un modelo de score incondicional y un clasificador auxiliar pre-entrenado.
    - *Flujo básico:*
      1. El investigador configura el sampler condicional, especificando la SDE, el modelo de score, el clasificador, la clase objetivo y la escala de guiado.
      2. El sistema ejecuta el muestreo condicional.
      3. El investigador obtiene las imágenes generadas pertenecientes (idealmente) a la clase especificada.
  - **CU5: Imputación de Regiones en Imágenes.**
    - *Actor:* Investigador.
    - *Descripción:* El investigador dispone de una imagen con regiones faltantes (definidas por una máscara) y desea utilizar el sistema para llenar dichas regiones de manera coherente.
    - *Flujo básico:*
      1. El investigador proporciona la imagen conocida, la máscara y carga un modelo de score pre-entrenado.
      2. El investigador configura e invoca el sampler de imputación.
      3. El sistema genera los píxeles para las regiones desconocidas.
      4. El investigador obtiene la imagen completa.

### 3.3. Diseño del software

El diseño del software se ha centrado en la modularidad y la claridad, buscando una implementación que refleje fielmente los conceptos teóricos subyacentes y facilite la experimentación.

### 3.3.1. Estructura del paquete

El proyecto se organiza en los siguientes directorios principales, cada uno encapsulando una funcionalidad específica:

- **diffusion/**: Contiene las implementaciones de los procesos de difusión basados en SDEs y los *schedules* de ruido.
  - **sde.py**: Define la clase base ‘DiffusionProcess’ y las implementaciones específicas para VE-SDE, VP-SDE y Sub-VP SDE (‘VESDE’, ‘VPSDE’, ‘SubVPSDE’), así como la clase ‘GaussianDiffusionProcess’ que añade la forma cerrada para el proceso directo y la función de pérdida.
  - **schedules.py**: Implementa los diferentes *schedules* de ruido (e.g., ‘LinearSchedule’, ‘CosineSchedule’).
- **samplers/**: Alberga los distintos algoritmos de muestreo (integradores) para la SDE inversa y la ODE de flujo de probabilidad.
  - **euler\_maruyama.py**: Integrador de Euler-Maruyama.
  - **exponential\_integrator.py**: Integrador exponencial de Euler para la ODE de flujo de probabilidad.
  - **predictor\_corrector.py**: Muestreador Predictor-Corrector genérico.
  - **probability\_flow\_ode.py**: Integrador genérico para la ODE de flujo de probabilidad.
- **models/**: Contiene las arquitecturas de los modelos de score.
  - **base\_model.py**: Define la interfaz genérica para un modelo de difusión.
  - **score\_model.py**: Implementación de una arquitectura base ScoreNet (U-Net simple).
  - **score\_net.py**: Implementación de una U-Net más completa (estilo NCSN++), con bloques residuales, auto-atención, etc.
- **conditional\_samplers/**: Implementaciones de los muestreadores para generación condicional.
  - **euler\_maruyama\_conditional\_class\_vp\_sub\_vp.py**: Sampler Euler-Maruyama condicional para VP/Sub-VP con guía de clasificador.
  - **euler\_maruyama\_conditional\_class.py**: Sampler Euler-Maruyama condicional genérico (VE/VP/Sub-VP).
  - **predictor\_corrector\_ve\_conditional.py**: Sampler Predictor-Corrector condicional para VE-SDE.
  - **predictor\_corrector\_vp\_sub\_vp\_conditional.py**: Sampler Predictor-Corrector condicional para VP/Sub-VP SDEs.
- **metrics/**: Implementaciones de las métricas para evaluar la calidad de las imágenes generadas.
  - **bpd.py**: Cálculo de Bits Per Dimension (BPD) y NLL vía ODE.
  - **fid.py**: Cálculo de Fréchet Inception Distance (FID).
  - **inception\_score.py**: Cálculo del Inception Score (IS).
- **imputation/**: Contiene la lógica para la imputación de imágenes.
  - **imputation.py**: Implementación del sampler de imputación y utilidades de ejemplo.
- **utils/**: Funciones de utilidad general, especialmente para visualización.
  - **diffusion\_utilities.py**: Herramientas para graficar imágenes, evoluciones del muestreo, etc.

Esta estructura promueve la separación de responsabilidades y facilita la localización y modificación de componentes específicos.

### 3.3.2. Diagrama de clases

El diseño del sistema se basa en un conjunto de clases principales que encapsulan las diferentes responsabilidades identificadas en la arquitectura del software. A continuación, se describen brevemente los grupos de clases más relevantes:

- **Procesos de Difusión (Directorio diffusion):** Clases como ‘DiffusionProcess‘, ‘GaussianDiffusionProcess‘, y las especializaciones ‘VESDE‘, ‘VPSDE‘, ‘SubVPSDE‘ definen la dinámica de los procesos de difusión hacia adelante y hacia atrás, así como los coeficientes de deriva y difusión. La clase ‘NoiseSchedule‘ y sus derivadas gestionan la programación del ruido.
- **Modelos de Score (Directorio models):** La clase base ‘DiffusionModel‘ define la interfaz para los modelos que estiman el score. Las implementaciones concretas, como ‘ScoreNet‘ (basada en U-Net), encapsulan las arquitecturas de redes neuronales que aprenden a estimar  $\nabla_x \log p_t(x)$ . Se incluyen componentes como ‘GaussianRandomFourierFeatures‘ para el embedding de tiempo.
- **Muestreadores (Directorios samplers y conditional\_samplers):** Un conjunto de clases se encarga de los algoritmos de muestreo para generar imágenes resolviendo la SDE inversa o la ODE de flujo de probabilidad.
- **Métricas (Directorio metrics):** Clases y funciones dedicadas al cálculo de las métricas de evaluación.
- **Imputación (Directorio imputation):** Contiene el ‘imputation\_sampler‘ que implementa la lógica específica para la tarea de imputación de imágenes.
- **Utilidades (Directorio utils):** Módulos con funciones auxiliares, principalmente para la visualización de imágenes y trayectorias de generación.

Las interacciones clave implican que un *sampler* utiliza una instancia de una SDE y un modelo de score entrenado para generar iterativamente una imagen. Los modelos condicionales, además, interactúan con un modelo clasificador.

Debido a la complejidad y extensión del diagrama de clases detallado del sistema completo, que incluye todas las clases, sus atributos, métodos y relaciones específicas, este se proporciona como un archivo complementario en formato PDF (o PNG/SVG) denominando `diagrama_clases_completo.svg` y también se encuentra disponible en el repositorio del proyecto en la siguiente ubicación:

[`diagrama_clases_completo.svg`]

Se recomienda consultar dicho archivo para una comprensión exhaustiva de la estructura estática del software.

## 3.4. Validación y pruebas

La validación del software y la verificación de la correctitud de las implementaciones son etapas cruciales en cualquier proyecto de desarrollo, y adquieren especial relevancia en un contexto de investigación donde la fiabilidad de los resultados experimentales depende directamente de la precisión de las herramientas construidas. En este proyecto, se ha

implementado una estrategia de pruebas exhaustiva para asegurar la calidad y el correcto funcionamiento de los componentes desarrollados.

Todas las pruebas se han implementado utilizando el framework ‘**pytest**’ de Python. La ejecución de las mismas se realiza desde el directorio raíz del proyecto mediante el comando ‘**pytest**’, que descubre y ejecuta automáticamente todos los archivos de prueba (nombrados siguiendo la convención ‘`test_*.py`’ o ‘`*_test.py`’).

### 3.4.1. Pruebas unitarias

Se ha puesto un énfasis considerable en las pruebas unitarias para verificar la funcionalidad de los componentes individuales y los algoritmos fundamentales de manera aislada. Estas pruebas cubren los siguientes módulos principales:

- **Módulos de SDEs (diffusion/sde.py):** Se han creado pruebas específicas para cada tipo de SDE implementada (VE-SDE, VP-SDE, Sub-VP SDE).
- **Schedules de Ruido (diffusion/schedules.py):** Se comprueba que los diferentes *schedules* de ruido (lineal, cosenoidal) generen las secuencias  $\beta(t)$  y  $\bar{\alpha}(t)$  conforme a sus definiciones matemáticas, incluyendo la verificación de valores límite y comportamiento monotónico.
- **Algoritmos de Muestreo (Samplers en samplers/ y conditional\_samplers/):** Para cada *sampler* implementado (Euler-Maruyama, Predictor-Corrector, ODE samplers, y sus variantes condicionales) se han diseñado pruebas que:
  - Verifican la ejecución de un paso de muestreo individual con entradas simuladas (mocked SDE y modelo de score).
  - Comprueban la forma y el tipo de datos de las salidas.
  - Aseguran el manejo correcto de los parámetros del *sampler* (e.g., número de pasos, *guidance scale*).
- **Modelos de Score (models/):** Las pruebas para los modelos de score se centran en:
  - La correcta construcción de las arquitecturas de red (e.g., U-Net) con las dimensiones esperadas en cada capa.
  - El paso hacia adelante (*forward pass*) del modelo con tensores de entrada y tiempo de ejemplo, verificando las dimensiones de salida.
  - El funcionamiento de los módulos de embedding de tiempo.
- **Métricas de Evaluación (metrics/):** Para cada métrica implementada (BP-D/NLL, FID, IS), las pruebas unitarias verifican:
  - El correcto preprocessamiento de los datos de entrada.
  - El cálculo de componentes individuales de la métrica con datos sintéticos o valores conocidos.

Estas pruebas unitarias son fundamentales para la detección temprana de errores y para garantizar que las modificaciones o refactorizaciones del código no introduzcan regresiones.

### 3.4.2. Pruebas de integración y funcionales

Además de las pruebas unitarias, se han realizado pruebas de integración y funcionales para verificar la correcta interacción entre los diferentes módulos y la funcionalidad completa de los casos de uso principales. Estas pruebas se ejecutan principalmente a través

de:

- **Scripts de ejecución dedicados:** Se han creado scripts que simulan los pipelines completos, como el entrenamiento de un modelo de score (si aplica dentro del alcance de las pruebas automatizadas) o la generación de un lote de imágenes y el cálculo de una métrica.
- **Notebooks de experimentación:** Los mismos *notebooks* utilizados para la experimentación (descritos en el Capítulo 4) sirven como un entorno de prueba funcional, permitiendo ejecutar flujos completos de trabajo:
  - Generación incondicional y condicional de imágenes.
  - Imputación de imágenes.
  - Cálculo de todas las métricas implementadas sobre las imágenes generadas.

La validación en este nivel a menudo implica la inspección de los resultados intermedios y finales (incluyendo la calidad visual de las imágenes y la plausibilidad de los valores de las métricas) para confirmar que el sistema se comporta según lo esperado.

La ejecución regular del conjunto de pruebas con ‘pytest’ desde el directorio raíz del proyecto asegura que la base de código se mantenga funcional y que los nuevos desarrollos se integren correctamente sin romper la funcionalidad existente.

## 3.5. Desarrollo y garantía de calidad de software

El desarrollo del software se llevó a cabo siguiendo prácticas orientadas a asegurar la calidad, mantenibilidad y reproducibilidad del código. El entorno de desarrollo principal fue Python 3.11.12, utilizando extensivamente la librería PyTorch para la implementación de redes neuronales y operaciones tensoriales aceleradas por GPU. Otras librerías científicas como NumPy, SciPy y Matplotlib fueron empleadas para cálculos numéricos, procesamiento de datos y visualización.

Se utilizó el sistema de control de versiones Git, con un repositorio alojado en GitHub [https://github.com/trentisiete/image\\_generation\\_difussion\\_project](https://github.com/trentisiete/image_generation_difussion_project). Esto facilitó el seguimiento de cambios, la colaboración (si aplica) y la gestión de diferentes versiones del código. Se procuró seguir las convenciones de estilo de código PEP 8 para Python, utilizando herramientas de *linting* y formateo como [Pylint] para mantener la consistencia y legibilidad del código.

La documentación del código se realizó mediante comentarios y *docstrings*, con la intención de facilitar la comprensión de la funcionalidad de las clases y métodos principales.

## 3.6. Otras cuestiones

Finalmente, se abordan algunas cuestiones transversales relacionadas con la distribución del software y sus implicaciones.

### 3.6.1. Licencia del software

El código fuente desarrollado en el marco de este proyecto se distribuye bajo la Licencia [MIT]. Esta es una licencia permisiva que permite la reutilización del código tanto en

proyectos académicos como comerciales, con mínimas restricciones, fomentando así la colaboración y el avance científico.

### 3.6.2. Implicaciones éticas

La generación de imágenes mediante IA, si bien ofrece numerosas aplicaciones beneficiosas, también plantea importantes consideraciones éticas. Entre ellas destacan:

- **Propiedad intelectual y autoría:** La generación de obras artísticas o diseños mediante IA plantea interrogantes sobre los derechos de autor y la originalidad.
- **Impacto ambiental:** El entrenamiento de modelos de aprendizaje profundo a gran escala, como los utilizados en este proyecto, puede requerir una cantidad considerable de recursos computacionales y energía, con el consiguiente impacto ambiental.

Aunque este proyecto se enfoca en los aspectos técnicos y de investigación, es fundamental ser consciente de estas implicaciones y promover un uso responsable de la tecnología.

### 3.6.3. Cuestiones de género

En este proyecto, los principales conjuntos de datos utilizados para la experimentación fueron MNIST [11] y CIFAR-10 [3]. El conjunto de datos MNIST, al estar compuesto por dígitos manuscritos, no presenta problemáticas directas de representación o sesgo de género.

# 4 Ejemplos de uso y experimentación

En este capítulo se presentan los resultados experimentales obtenidos a lo largo del proyecto, ilustrando el funcionamiento del sistema de generación de imágenes mediante procesos de difusión basados en SDEs. Se detallará la configuración experimental general y, posteriormente, se mostrarán ejemplos de uso extraídos de los *notebooks* interactivos desarrollados, que sirvieron como principal herramienta para la experimentación y visualización.

## 4.1. Generación de imágenes y evaluación mediante notebooks interactivos

Una parte fundamental del desarrollo de este proyecto ha sido la creación de *notebooks* de Jupyter que permiten no solo la generación de imágenes bajo diversas configuraciones, sino también la evaluación inmediata de su calidad mediante las métricas implementadas. Estos *notebooks* están diseñados para ser interactivos y configurables, facilitando la exploración de los diferentes componentes del sistema.

Todos los *notebooks* comparten una estructura inicial común que permite al usuario:

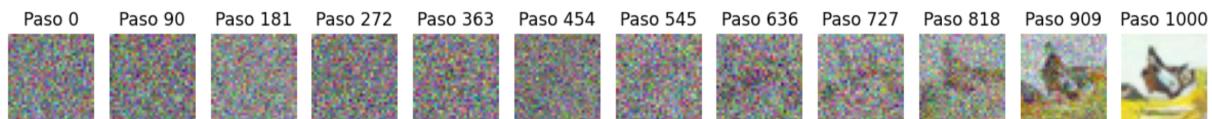
1. **Configuración de hiperparámetros generales y del conjunto de datos:** En las celdas iniciales, el usuario puede ajustar parámetros globales como las dimensiones de las imágenes, el número de muestras a generar, y seleccionar el conjunto de datos (e.g., CIFAR-10, MNIST). Se proporcionan modelos pre-entrenados, dado que el proceso de entrenamiento es computacionalmente costoso; por tanto, la elección del “modelo SDE + arquitectura de red” es una selección a partir de opciones disponibles, mientras que los hiperparámetros del proceso de muestreo son más libremente ajustables.
2. **Elección del *sampler*:** Se ofrece la posibilidad de seleccionar uno o varios de los algoritmos de muestreo implementados (Predictor-Corrector, Euler-Maruyama, Probability Flow ODE, Exponential Integrator) para la generación y comparación. A continuación, se detallan los experimentos realizados para diferentes tareas de generación. Para una exploración exhaustiva de todos los modelos SDE (VE, VP, SubVP con diferentes *schedules* de ruido) y sus resultados, se remite al repositorio de GitHub del proyecto: [https://github.com/trentisiete/image\\_generation\\_difussion\\_project](https://github.com/trentisiete/image_generation_difussion_project). En esta memoria, se presentarán resultados representativos para ilustrar las capacidades del sistema.

### 4.1.1. Generación Incondicional de Imágenes con CIFAR-10

Este primer caso de uso se centra en la generación incondicional de imágenes, es decir, la síntesis de nuevas muestras a partir de ruido sin ninguna etiqueta de clase o condición ex-

terna. Se utiliza el conjunto de datos CIFAR-10 [3] y, a modo ilustrativo, se detallan los resultados obtenidos con una SDE de Varianza Preservada (VP-SDE) y un *schedule* de ruido lineal. Se dispone de un *notebook* específico (`Generacion_Incondicional_CIFAR10.ipynb`) para esta tarea, y otro análogo (`Generacion_Incondicional_MNIST.ipynb`) para experimentación con el dataset MNIST [11].

La función principal `generate_from_sampler` permite seleccionar el *sampler* deseado (Predictor-Corrector, Euler-Maruyama, Probability Flow ODE, Exponential Integrator) y ajustar sus hiperparámetros específicos (e.g., número de pasos, parámetros del corrector). Tras la generación, se visualiza la evolución de un conjunto de muestras desde ruido puro hasta la imagen final. La Figura 4.1 muestra un ejemplo de esta evolución para el modelo VP-SDE Lineal utilizando el *sampler* Predictor-Corrector.



**Figura 4.1:** Evolución de la generación de imágenes con VP-SDE Lineal y sampler Predictor-Corrector sobre CIFAR-10. Se muestran instantáneas del proceso en diferentes pasos (de izquierda a derecha) desde el ruido inicial hasta la imagen final.

Para una comparación directa de los resultados finales de los diferentes *samplers*, se utiliza una función que agrupa las imágenes generadas por cada uno. La Figura 4.2 presenta una comparativa visual de las imágenes finales obtenidas con cada uno de los *samplers* implementados, todos ellos partiendo del mismo ruido inicial (cuando aplica) y utilizando el mismo modelo de score (VP-SDE Lineal).

### Imagenes finales Generadas por Predictor\_corrector



### Imagenes finales Generadas por Exponential Integrator



### Imagenes finales Generadas por Probability Flow ODE



### Imagenes finales Generadas por Euler Maruyama



**Figura 4.2:** Comparación de imágenes finales (CIFAR-10) generadas con el modelo VP-SDE Lineal utilizando diferentes samplers: Predictor-Corrector, Exponential Integrator, Probability Flow ODE y Euler-Maruyama.

Brevemente, sobre el comportamiento de los *samplers* en este contexto:

- **Euler-Maruyama:** Es el integrador estocástico más simple para la SDE inversa (Ecuación 2.2). Discretiza la SDE y añade un término de ruido gaussiano en cada paso. Suele requerir un número considerable de pasos para obtener buenos resultados.
- **Predictor-Corrector (PC):** Estos *samplers*, como los propuestos en [16], combinan un paso de predicción (usualmente un integrador de SDE como Euler-Maruyama) con uno o más pasos de corrección. El corrector utiliza el modelo de score  $s_\theta(x, t)$  para refinar la muestra actual  $x_t$  de acuerdo a  $p_t(x)$ , por ejemplo, mediante pasos de MCMC basados en el score (como la dinámica de Langevin). La sinergia entre el predictor, que avanza en el tiempo (inverso), y el corrector, que mejora la muestra en el mismo instante temporal, a menudo conduce a una mejor calidad de imagen con un

número similar o incluso menor de evaluaciones del modelo de score en comparación con el predictor solo. Los resultados obtenidos con el *sampler* Predictor-Corrector suelen ser de buena calidad visual.

- **Probability Flow ODE (PF-ODE):** Resuelve la ODE determinista (Ecuación 2.4) que comparte las mismas marginales que la SDE. Al ser determinista, una vez fijado el ruido inicial  $x_T$ , la trayectoria y la imagen final  $x_0$  son únicas. Si bien permite el cálculo exacto de la verosimilitud, la calidad visual de las muestras generadas por los solvers ODE de propósito general no siempre iguala a la de los mejores *samplers* estocásticos o PC, especialmente si no se utilizan muchos pasos o solvers adaptativos muy precisos.
- **Exponential Integrator:** Es un tipo específico de solver para la ODE de flujo de probabilidad, particularmente adaptado a la forma de la ODE cuando el coeficiente de deriva es lineal en  $x$  (como en VP-SDE o sub-VP SDE si se reescribe adecuadamente). Su convergencia y calidad pueden depender de la estabilidad de la formulación y del número de pasos; en algunos escenarios, puede no converger adecuadamente o producir artefactos si los pasos son demasiado grandes.

Inmediatamente después de la generación, los *notebooks* permiten calcular las métricas de evaluación sobre las imágenes generadas.

### Evaluación Cuantitativa: Resultados de Métricas

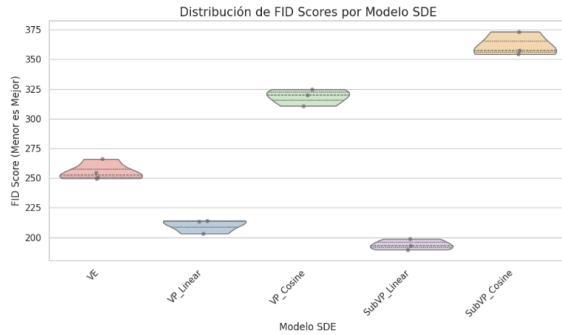
Además de la inspección visual, la calidad de las imágenes generadas incondicionalmente se evaluó cuantitativamente mediante un conjunto de métricas estándar en la literatura: Fréchet Inception Distance (FID), Bits Per Dimension (BPD) e Inception Score (IS). Los *notebooks* desarrollados permiten el cálculo directo de estas métricas tras la generación de las muestras. Los resultados promedio obtenidos para los diferentes modelos SDE base (VE, VP-Lineal, VP-Cosenoidal, SubVP-Lineal y SubVP-Cosenoidal), cada uno evaluado con un conjunto de *samplers* y promediando sobre múltiples ejecuciones (generalmente N=3 o N=4, como se indica), se resumen en la Tabla 4.1.

**Tabla 4.1:** Resumen de métricas de evaluación para diferentes modelos SDE en generación incondicional sobre CIFAR-10. Los valores se presentan como Media ± Desv. Estándar (Número de ejecuciones). Para FID y BPD, menor es mejor. Para IS, mayor es mejor.

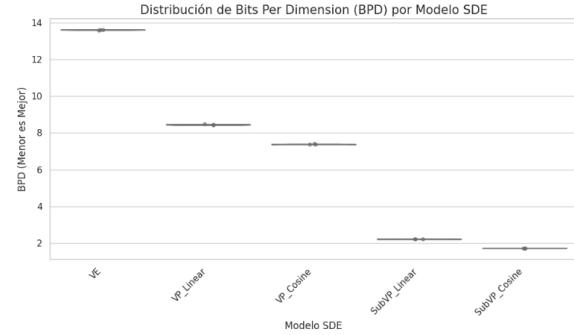
Modelo SDE	FID Score	Bits Per Dimension (BPD)	Inception Score (Media)	IS (Desv. Est. interna)
VE	$255,27 \pm 7,49$ (N=4)	$13,63 \pm 0,02$ (N=3)	$2,70 \pm 0,12$ (N=3)	$0,30 \pm 0,08$ (N=3)
VP-Lineal	$210,28 \pm 6,09$ (N=3)	$8,45 \pm 0,03$ (N=3)	$3,34 \pm 0,13$ (N=3)	$0,41 \pm 0,08$ (N=3)
VP-Cosenoidal	$318,47 \pm 7,05$ (N=3)	$7,38 \pm 0,02$ (N=3)	$1,65 \pm 0,04$ (N=3)	$0,12 \pm 0,02$ (N=3)
SubVP-Lineal	<b><math>193,82 \pm 4,61</math> (N=3)</b>	$2,21 \pm 0,00$ (N=3)	<b><math>3,72 \pm 0,15</math> (N=3)</b>	$0,40 \pm 0,12$ (N=3)
SubVP-Cosenoidal	$361,72 \pm 10,15$ (N=3)	<b><math>1,71 \pm 0,00</math> (N=3)</b>	$1,42 \pm 0,02$ (N=3)	$0,09 \pm 0,01$ (N=3)

Para una mejor visualización de la distribución de estos resultados a través de las ejecuciones, se generaron gráficos de violín para cada métrica principal, los cuales se presentan en las Figuras 4.3, 4.4, e 4.6. Estos gráficos permiten apreciar no solo la media, sino también la dispersión y forma de la distribución de los resultados para cada modelo.

**Análisis de los Resultados Cuantitativos.** De la Tabla 4.1 y los gráficos de violín, se desprenden varias observaciones:

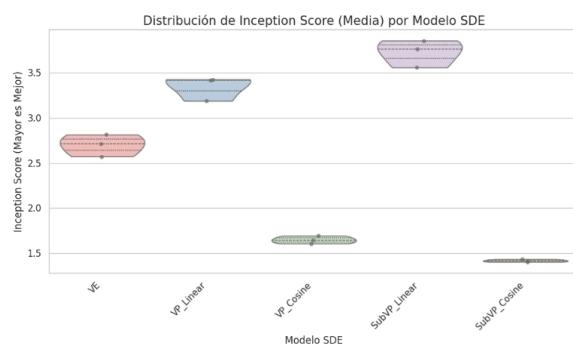


**Figura 4.3:** Distribución de FID Scores.

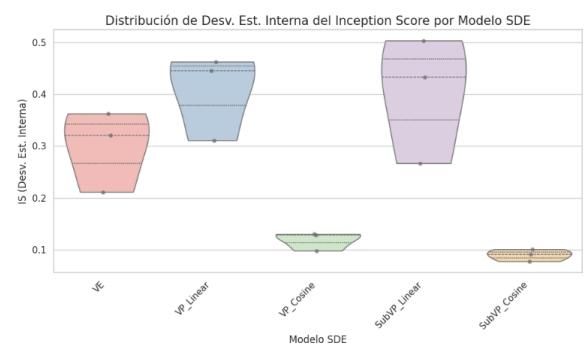


**Figura 4.4:** Distribución de BPD.

**Figura 4.5:** Gráficos de violín mostrando la distribución de los resultados para las métricas FID y BPD entre los diferentes modelos SDE.



**Figura 4.6:** Distribución de Inception Score (Media).



**Figura 4.7:** Distribución de Inception Score (Desv. Est. Interna).

**Figura 4.8:** Gráficos de violín mostrando la distribución de los resultados para la media del Inception Score y su desviación estándar interna.

- **Fréchet Inception Distance (FID):** Esta métrica compara la distribución de las características extraídas por un modelo InceptionV3 de las imágenes generadas con las de las imágenes reales. Un valor FID más bajo indica una mayor similitud y, por tanto, mejor calidad y realismo perceptual. En nuestros experimentos, el modelo **SubVP-SDE Lineal** obtiene el mejor (menor) FID promedio ( $193,82 \pm 4,61$ ), sugiriendo la mejor calidad perceptual entre los modelos probados. Le sigue el modelo VP-SDE Lineal ( $210,28 \pm 6,09$ ). Los modelos con *schedule* cosenoidal (VP-Cosenoidal y SubVP-Cosenoidal) muestran un FID considerablemente más alto, indicando una peor calidad perceptual. El modelo VE-SDE se sitúa en una posición intermedia. Es importante notar que, debido a limitaciones computacionales, el cálculo de FID se realizó sobre un subconjunto del dataset de prueba de CIFAR-10 (controlado por el parámetro ‘partition\_subset’ en el código, que divide el número total de datos del dataset). Si bien esto permite una comparación relativa válida entre nuestros modelos, los valores absolutos de FID podrían variar si se utilizara el conjunto completo estándar de 50k imágenes generadas y 50k reales.
- **Bits Per Dimension (BPD):** BPD es una medida de la verosimilitud logarítmica negativa (NLL) del modelo, normalizada por la dimensionalidad de los datos. Un BPD más bajo indica que el modelo asigna una mayor probabilidad a los datos de prueba, lo que sugiere un mejor ajuste del modelo a la distribución de los datos. Aquí, los modelos con *schedule* cosenoidal destacan notablemente: **SubVP-SDE Cosenoidal** ( $1,71 \pm 0,00$ ) y **VP-SDE Cosenoidal** ( $7,38 \pm 0,02$ ) obtienen los mejores (menores) valores de BPD, con el SubVP-SDE Cosenoidal mostrando un rendimiento excepcional en esta métrica. El modelo SubVP-SDE Lineal ( $2,21 \pm 0,00$ ) también presenta un excelente BPD. En contraste, el modelo VE-SDE ( $13,63 \pm 0,02$ ) obtiene el BPD más alto, indicando una peor modelización de la verosimilitud de los datos.
- **Inception Score (IS):** El IS intenta medir simultáneamente la calidad (imágenes reconocibles y nítidas) y la diversidad de las muestras generadas. Un IS más alto es mejor. Los resultados del IS (Media) se correlacionan en cierta medida con los del FID. El modelo **SubVP-SDE Lineal** ( $3,72 \pm 0,15$ ) vuelve a ser el de mejor rendimiento, seguido por VP-SDE Lineal ( $3,34 \pm 0,13$ ). Los modelos con *schedule* cosenoidal (VP-Cosenoidal y SubVP-Cosenoidal) obtienen los IS más bajos, lo que concuerda con su pobre rendimiento en FID. La desviación estándar interna del IS (cuanto menor, más consistentes son las predicciones para cada *split* de imágenes) también se reporta.

**Discusión sobre la Fiabilidad y Conclusiones de las Métricas.** La evaluación cuantitativa de modelos generativos es un desafío, y cada métrica tiene sus propias fortalezas y limitaciones:

Es importante abordar la evaluación cuantitativa con una perspectiva crítica. Las métricas estándar como FID, BPD e IS, aunque ampliamente utilizadas, presentan una volatilidad considerable y no siempre son indicadores infalibles del “mejor” modelo generativo de forma global. En ocasiones, pueden incluso clasificar en primer lugar a modelos que visualmente no son los más convincentes, debido a que sus optimizaciones pueden estar alineadas con aspectos particulares que la métrica favorece, pero que no necesariamente se correlacionan con todos los atributos deseados de la generación (como la coherencia

semántica a largo plazo o la creatividad). Los valores obtenidos pueden depender de múltiples factores ajenos a la calidad intrínseca del generador, como se detalla a continuación al analizar cada métrica:

- Los resultados presentados se basan en un número limitado de ejecuciones ( $N=3$  o  $N=4$ ), como se indica. Las desviaciones estándar proporcionan una idea de la variabilidad de los resultados. Por ejemplo, los valores de BPD para los modelos SubVP muestran una desviación estándar muy baja, lo que sugiere una alta consistencia en esta métrica. Los FID e IS presentan una variabilidad mayor, lo cual es común.
- **FID** es ampliamente utilizado y se considera que se correlaciona razonablemente bien con la percepción humana de la calidad de imagen. Sin embargo, es sensible al número de muestras utilizadas, a la capa de Inception de la que se extraen las características y puede no capturar todos los aspectos de la calidad de imagen o el contenido semántico.
- **BPD** (o NLL) es una métrica rigurosa desde el punto de vista probabilístico, pero no siempre se alinea directamente con la calidad perceptual. Como se observa en nuestros resultados, los modelos con mejor BPD (SubVP-Cosenoidal, VP-Cosenoidal) no son los que mejor FID o IS obtienen. Esto sugiere que estos modelos son excelentes para capturar la distribución de probabilidad de los datos, pero esto no se traduce necesariamente en imágenes visualmente más atractivas o diversas según los criterios de FID/IS.
- **IS** también tiene limitaciones conocidas, como su sensibilidad a los trucos (*gaming*) y su dependencia de un clasificador InceptionV3 pre-entrenado en ImageNet, que puede no ser óptimo para datasets pequeños o muy diferentes como CIFAR-10. Tiende a favorecer modelos que generan imágenes fácilmente clasificables en una de las 1000 clases de ImageNet.

En conclusión, a partir de los resultados cuantitativos de este estudio sobre CIFAR-10:

- El modelo **SubVP-SDE Lineal** emerge como el más prometedor en términos de calidad perceptual y diversidad, según las métricas FID e IS.
- Los modelos basados en **SubVP-SDE**, particularmente con *schedule* cosenoidal, demuestran una capacidad superior para modelar la verosimilitud de los datos (mejor BPD).
- Se observa un claro *trade-off* entre la optimización de la verosimilitud (BPD) y la calidad perceptual (FID/IS). Los *schedules* cosenoidales, si bien excelentes para BPD, resultan en un peor rendimiento en FID e IS en comparación con los *schedules* lineales dentro de las familias VP y SubVP.
- El modelo VE-SDE, en la configuración probada, no destaca en ninguna de las métricas evaluadas en comparación con las mejores variantes de VP o SubVP.

Estos resultados subrayan la importancia de utilizar un conjunto diverso de métricas para una evaluación comprensiva de los modelos generativos. La elección del "mejor" modelo puede depender del objetivo final de la aplicación (e.g., máxima fidelidad visual vs. mejor modelado probabilístico).

#### 4.1.2. Generación Condicional de Imágenes con CIFAR-10

Además de la generación incondicional, este proyecto explora la capacidad de los modelos de difusión basados en SDEs para la generación condicional de imágenes, es decir, la

síntesis de muestras que pertenezcan a una clase específica  $y$ . Este enfoque, conocido como guiado por clasificador (*classifier guidance*) [4], permite dirigir el proceso de generación utilizando un modelo de score incondicional  $s_\theta(x(t), t)$  junto con un clasificador adicional  $p_\phi(y|x(t), t)$  entrenado para predecir la clase de una imagen ruidosa  $x(t)$  en el instante  $t$ .

El principio se basa en modificar el score efectivo en la SDE inversa. Según la regla de Bayes, el score condicional  $\nabla_x \log p_t(x|y)$  se puede descomponer como:

$$\nabla_x \log p_t(x|y) = \nabla_x \log p_t(x) + \nabla_x \log p_t(y|x) \quad (4.1)$$

donde  $\nabla_x \log p_t(x)$  es el score incondicional estimado por  $s_\theta(x(t), t)$ , y  $\nabla_x \log p_t(y|x)$  es el gradiente de la probabilidad logarítmica de la clase  $y$  dada la imagen ruidosa  $x(t)$ , que proporciona el clasificador dependiente del tiempo. Este último término "guía.<sup>el</sup> muestreo hacia la clase deseada. La SDE inversa condicional se convierte entonces en:

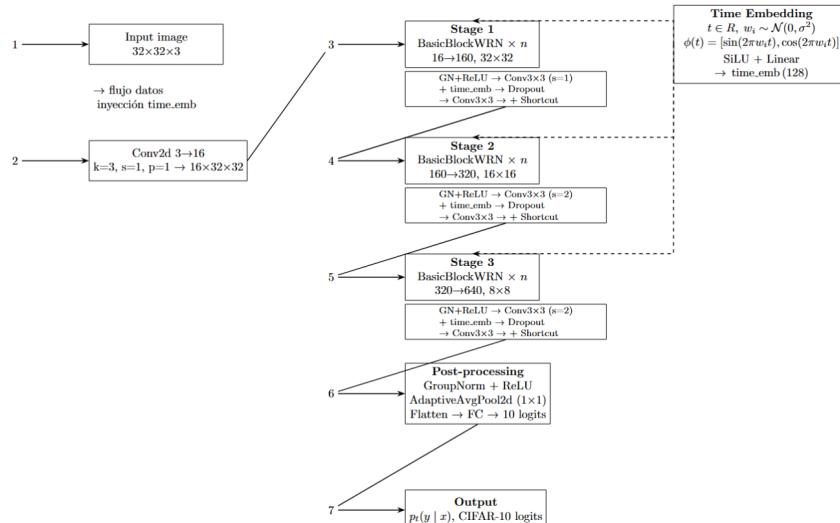
$$dx = \left[ f(x, t) - g(t)^2 (s_\theta(x(t), t) + w \nabla_x \log p_\phi(y|x(t), t)) \right] dt + g(t) d\bar{W}(t) \quad (4.2)$$

donde  $w$  es un factor de escala para la guía del clasificador (*guidance scale*).

## Componentes del Sistema de Generación Condicional

Para esta tarea, se emplean dos modelos principales:

- Clasificador Dependiente del Tiempo ( $p_\phi(y|x(t), t)$ ):** Se ha implementado un modelo ‘TimeDependentWideResNet’, basado en la arquitectura Wide ResNet (WRN) [18], modificado para procesar imágenes ruidosas  $x(t)$  y el instante de tiempo  $t$  como entradas. El tiempo  $t$  se codifica mediante Características Aleatorias de Fourier Gaussianas (*Gaussian Random Fourier Features*) y se inyecta en los bloques residuales de la red. Esta arquitectura permite al clasificador aprender a identificar la clase de una imagen a diferentes niveles de ruido. La Figura 4.9 ilustra esquemáticamente esta arquitectura.



**Figura 4.9:** Representación esquemática de la arquitectura del clasificador ‘TimeDependentWideResNet’.

2. **Modelo de Score Incondicional ( $s_\theta(x(t), t)$ ):** Se utilizan los mismos modelos de score (arquitecturas U-Net) entrenados para la generación incondicional.

Por consideraciones de coste computacional y para simplificar la demostración, en los experimentos presentados en esta memoria se ha utilizado principalmente el muestreador ‘ConditionalEulerMaruyamaSampler’. Este sampler implementa el método de Euler-Maruyama para la Ecuación 4.2, calculando el gradiente del clasificador  $\nabla_x \log p_\phi(y|x(t), t)$  en cada paso y combinándolo con el score incondicional. Aunque se han implementado otros samplers condicionales (e.g., basados en Predictor-Corrector), el ‘ConditionalEulerMaruyamaSampler’ ha demostrado un buen equilibrio entre simplicidad, calidad de resultados y eficiencia para esta tarea en nuestras pruebas preliminares.

## Experimentación y Resultados en CIFAR-10

Los experimentos de generación condicional se realizaron sobre el conjunto de datos CIFAR-10. El *notebook* correspondiente (`Generacion_Condicional_CIFAR10.ipynb`) permite al usuario seleccionar una de las 10 clases de CIFAR-10 (Avión, Automóvil, Pájaro, Gato, Ciervo, Perro, Rana, Caballo, Barco, Camión) y el número de muestras a generar.

Debido al coste computacional asociado al entrenamiento de un clasificador dependiente del tiempo para cada tipo de SDE, las pruebas se centraron en los tres modelos de score incondicional que mejores resultados preliminares ofrecieron en la generación incondicional:

- VE-SDE (Variance Exploding SDE)
- VP-SDE Lineal (Variance Preserving SDE con schedule de ruido lineal)
- SubVP-SDE Lineal (Sub-Variance Preserving SDE con schedule de ruido lineal)

Para cada uno de estos modelos SDE, se entrenó un clasificador ‘TimeDependentWideResNet’ específico. Las precisiones (*accuracy*) promedio de estos clasificadores sobre un conjunto de prueba de imágenes ruidosas de CIFAR-10 (evaluadas a través de diferentes niveles de ruido  $t$ ) se resumen en la Tabla 4.2.

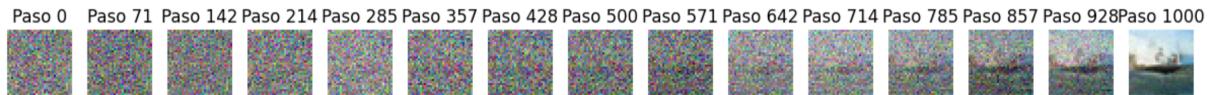
**Tabla 4.2:** Precisión promedio de los clasificadores ‘TimeDependentWideResNet’ entrenados para diferentes SDEs base sobre CIFAR-10 ruidoso.

Modelo SDE Base del Clasificador	Precisión Promedio (Accuracy)	Nota:
VE-SDE	0.55	
VP-SDE Lineal	0.35	
SubVP-SDE Lineal	0.44	

La precisión se promedia sobre múltiples niveles de ruido  $t$ .

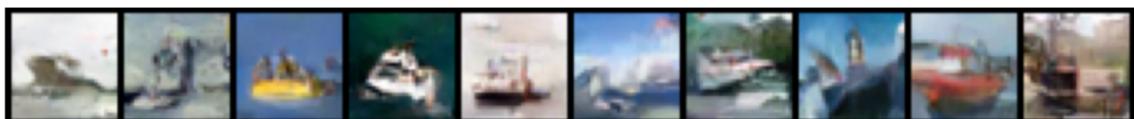
Se observa que el clasificador entrenado con la perturbación de la VE-SDE obtiene la mayor precisión, mientras que el de VP-SDE obtiene la menor. Esto puede influir en la calidad de la guía del clasificador.

A continuación, se muestra un ejemplo de la generación condicional para la clase “Barco” ( $y = 8$ ). La Figura 4.10 ilustra la evolución de las muestras desde el ruido inicial hasta las imágenes finales para los tres modelos SDE mencionados, utilizando el ‘ConditionalEulerMaruyamaSampler’.



**Figura 4.10:** Evolución de la generación condicional para la clase “Barco” en CIFAR-10 utilizando el ‘ConditionalEulerMaruyamaSampler’. Resultados con modelos de score VE-SDE.

Generacion de 10 imágenes de Barco con SUB-VP:



Generacion de 10 imágenes de Barco con VP:



Generacion de 10 imágenes de Barco con VE:



**Figura 4.11:** Comparación de imágenes finales generadas condicionalmente para la clase “Barco” con los modelos VE-SDE, VP-SDE Lineal y SubVP-SDE Lineal.

Los resultados visuales (Figura 4.10) muestran cómo el proceso de muestreo, guiado por el clasificador, es capaz de generar imágenes que se asemejan a la clase objetivo. La calidad y coherencia de las imágenes pueden variar entre los diferentes modelos SDE base, influenciadas tanto por la calidad del modelo de score incondicional como por la efectividad del clasificador asociado.

A partir de la inspección visual de la Figura 4.5, se comparan las imágenes finales generadas condicionalmente para la clase Barco utilizando los modelos VE-SDE, VP-SDE Lineal y SubVP-SDE Lineal, se pueden extraer las siguientes conclusiones cualitativas:

- SubVP-SDE Lineal (fila superior): Este modelo, además de ser el mejor, produce consistentemente las imágenes más coherentes y visualmente reconocibles como “barcos”. Las formas generales de las embarcaciones son, en su mayoría, claras y distinguibles, y se aprecia cierta diversidad en los tipos y perspectivas de los barcos generados. La calidad, dentro de las limitaciones de la resolución de CIFAR-10, es notable para esta clase.
- VP-SDE Lineal (fila central): Las imágenes generadas por este modelo tienden a ser las más abstractas o con mayores artefactos visuales del conjunto, esto es porque es

peor modelo entre los tres. Si bien algunas formas pueden sugerir vagamente una embarcación, la fidelidad a la clase "barco" parece ser inferior en comparación con los otros dos modelos, y la coherencia visual entre las distintas muestras es menor.

- VE-SDE (fila inferior): Este modelo también logra generar imágenes que se identifican como barcos, con algunas muestras que presentan siluetas bastante definidas. La calidad visual general podría considerarse intermedia o, al menos, con un estilo de artefactos diferente al de VP-SDE. La variabilidad en la calidad entre las muestras parece ser mayor que en Sub-VP, pero con resultados a menudo más definidos que con VP-SDE.

### 4.1.3. Imputación de Imágenes con CIFAR-10 y Máscaras MNIST

Además de la generación incondicional y condicional, los modelos de difusión basados en SDEs ofrecen una notable flexibilidad para tareas de edición y manipulación de imágenes, como la imputación. La imputación de imágenes (o *inpaiting*) consiste en llenar regiones faltantes o corruptas de una imagen de manera que el contenido generado sea coherente con el contexto visual circundante. Esta capacidad es crucial para la restauración de fotografías antiguas, la eliminación de objetos no deseados o la reconstrucción de datos incompletos.

El enfoque para la imputación con modelos de score SDE, tal como se explora en [16], se basa en guiar el proceso de muestreo de la SDE inversa (Ecuación 2.2) utilizando la información de los píxeles conocidos. Sea  $x_0$  la imagen original parcialmente observada y  $m$  una máscara binaria donde los valores 1 indican los píxeles conocidos y 0 los píxeles a imputar. Durante el proceso de difusión inverso, en cada paso de tiempo  $t_i$  a  $t_{i-1}$ :

1. Se predice una muestra  $x'_{t_{i-1}}$  a partir de  $x_{t_i}$  usando el modelo de score  $s_\theta(x_{t_i}, t_i)$  y un *sampler* de SDE (e.g., Euler-Maruyama).
2. Simultáneamente, los píxeles conocidos de la imagen original  $x_0$  se perturban con ruido según la SDE directa hasta el tiempo  $t_{i-1}$ , obteniendo  $\tilde{x}_{t_{i-1}}(\text{conocidos}) \sim p_{0,t_{i-1}}(x(t_{i-1})|x_0(\text{conocidos}))$ .
3. La muestra  $x'_{t_{i-1}}$  se actualiza reemplazando sus regiones conocidas por las de  $\tilde{x}_{t_{i-1}}(\text{conocidos})$ :

$$x'_{t_{i-1}} = m \odot \tilde{x}_{t_{i-1}}(\text{conocidos}) + (1 - m) \odot x_{t_{i-1}}. \quad (4.3)$$

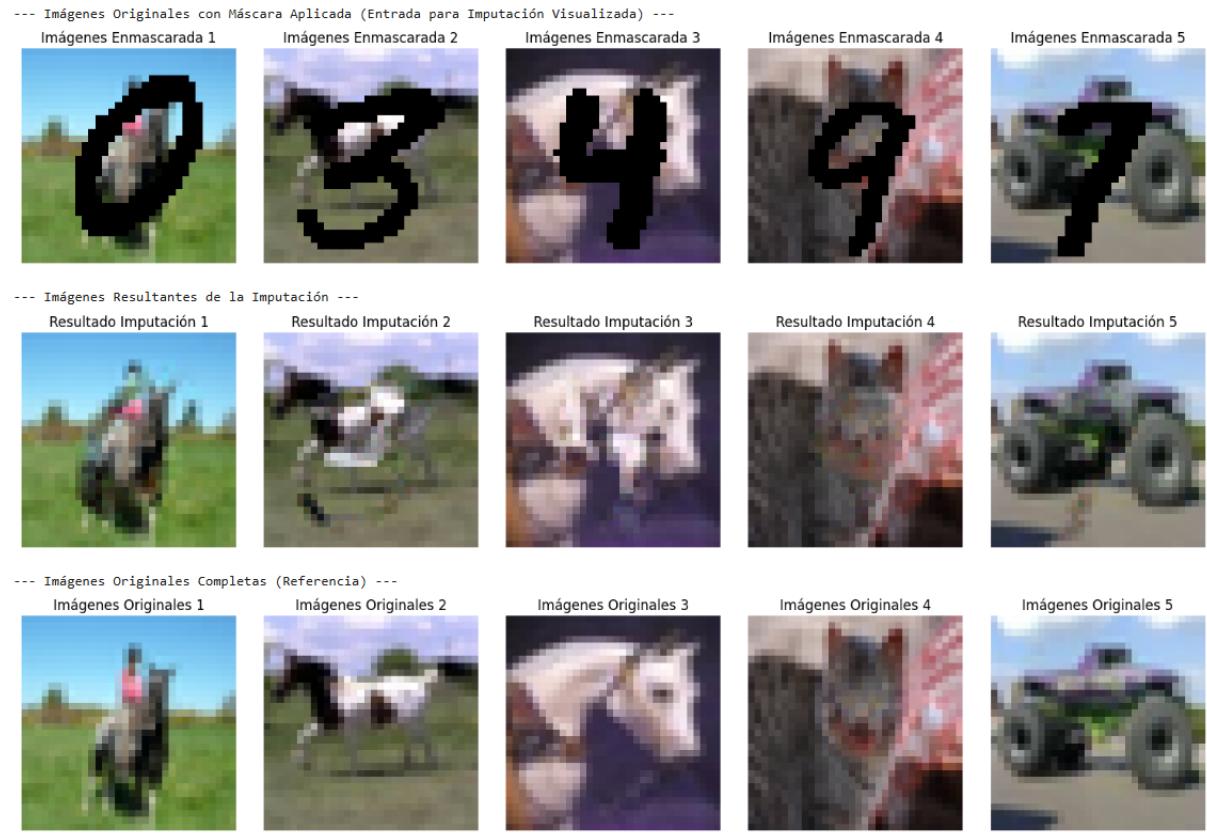
Esta  $x'_{t_{i-1}}$  sirve como entrada para el siguiente paso.

Este mecanismo asegura que la generación en las regiones faltantes esté condicionada y sea consistente con la información presente en el resto de la imagen.

Para demostrar esta funcionalidad, se ha desarrollado el `notebook demo_Imputation.ipynb`. En este, se utiliza el conjunto de datos CIFAR-10 para las imágenes base y el conjunto de datos MNIST para generar máscaras. Específicamente, un dígito de MNIST redimensionado se utiliza para definir la región que será enmascarada (y posteriormente imputada) en una imagen de CIFAR-10.

La Figura 4.12 ilustra el proceso y los resultados. Se utiliza un modelo de score pre-entrenado con una SDE de Varianza Explosiva (VE-SDE) y el *sampler* de imputación implementado (`imputation_sampler`), ejecutado durante 1000 pasos de discretización. La primera fila muestra las imágenes originales de CIFAR-10 con la máscara derivada de un dígito de MNIST aplicada (las áreas negras representan las regiones a imputar).

La segunda fila presenta el resultado de la imputación: el modelo de difusión ha generado contenido plausible para llenar las áreas enmascaradas. Para una evaluación visual directa, la tercera fila muestra las imágenes originales de CIFAR-10 completas.



**Figura 4.12:** Proceso y resultado de la imputación de imágenes CIFAR-10 utilizando máscaras generadas a partir de MNIST y un modelo VE-SDE. Fila superior: Imágenes originales de CIFAR-10 con la máscara aplicada (zonas a imputar). Fila intermedia: Imágenes resultantes tras el proceso de imputación. Fila inferior: Imágenes originales de CIFAR-10 como referencia.

Se observa que el contenido generado en las regiones imputadas busca integrarse de manera natural con las texturas, colores y estructuras presentes en las porciones conocidas de la imagen. La calidad y coherencia de la imputación dependen de la capacidad del modelo de score para capturar la distribución de los datos, la información contextual disponible en la imagen enmascarada, la SDE elegida y los hiperparámetros del *sampler* de imputación, como el número de pasos.

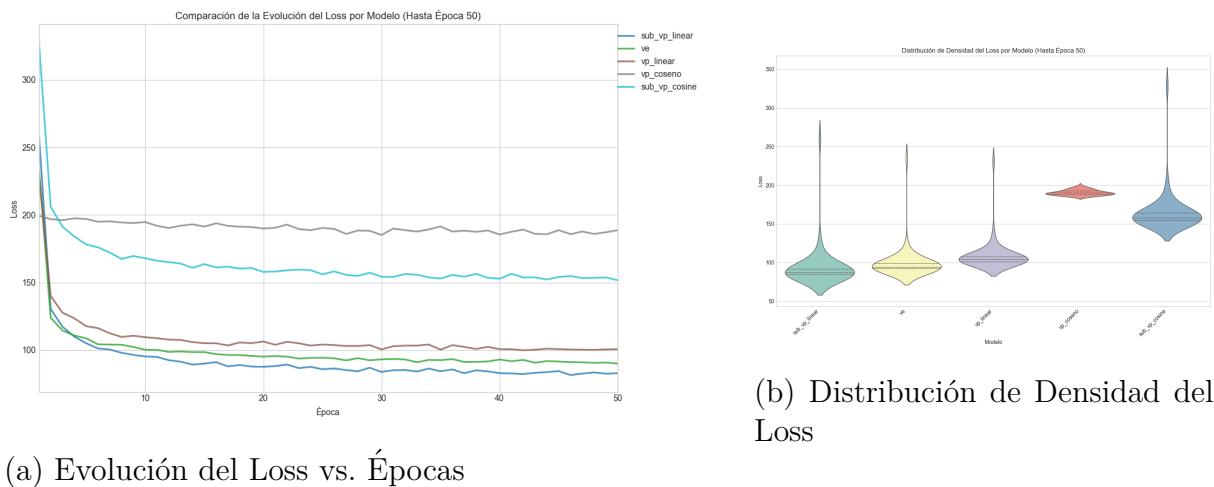
#### 4.1.4. Análisis Comparativo de la Convergencia del Entrenamiento entre Modelos SDE

La evaluación del rendimiento durante el entrenamiento de los diferentes modelos de score implementados, considerando las cinco combinaciones de Ecuaciones Diferenciales Estocásticas (SDEs) y *schedules* de ruido (VE-SDE, VP-Lineal, VP-Coseno, SubVP-Lineal y SubVP-Coseno), se ha centrado en el análisis de la función de pérdida (Loss) durante

las primeras 50 épocas. Este periodo inicial es a menudo indicativo de la estabilidad y la eficiencia de aprendizaje de cada configuración. Las visualizaciones presentadas a continuación ofrecen una visión complementaria y detallada del comportamiento de cada modelo en el conjunto de datos CIFAR-10.

### Evolución Temporal y Distribución del Loss en las Primeras 50 Épocas

Para comprender la dinámica de aprendizaje y la consistencia del rendimiento, la Figura 4.13 muestra: (a) la evolución temporal del loss promedio por época y (b) un diagrama de violín que representa la distribución de estos valores de loss a lo largo de las primeras 50 épocas de entrenamiento para cada una de las configuraciones SDE.



**Figura 4.13:** Comportamiento de la función de pérdida durante las primeras 50 épocas de entrenamiento: (a) Curvas del loss promedio por época para los cinco modelos SDE. (b) Diagrama de violín mostrando la distribución de los valores de loss promedio por época acumulados durante este periodo para cada modelo.

### Observaciones y Conclusiones (Curvas y Diagrama de Violín del Loss en 50 Épocas):

La gráfica de evolución temporal del loss (Figura 4.13a) ilustra la dinámica de aprendizaje de cada modelo. De manera general, todos los modelos exhiben una tendencia a la convergencia, manifestada por una disminución progresiva del loss a medida que aumentan las épocas de entrenamiento. Este comportamiento es indicativo de que los modelos están, en efecto, aprendiendo a estimar el *score* de la distribución de datos perturbados. Sin embargo, emergen diferencias significativas tanto en la velocidad de convergencia como en el nivel de loss asintótico alcanzado durante este periodo inicial. Notablemente, los modelos configurados como `sub_vp_linear` y `ve` se destacan por alcanzar los valores de loss más bajos de forma más temprana y mantenerlos de manera relativamente estable. Les sigue de cerca el modelo `vp_linear`, que también demuestra una convergencia robusta, aunque estabilizándose en un nivel de loss ligeramente superior en comparación con los dos primeros. En contraste, el modelo `sub_vp_cosine` se sitúa en un rango intermedio de loss, mientras que la configuración `vp_coseno` registra consistentemente los valores de

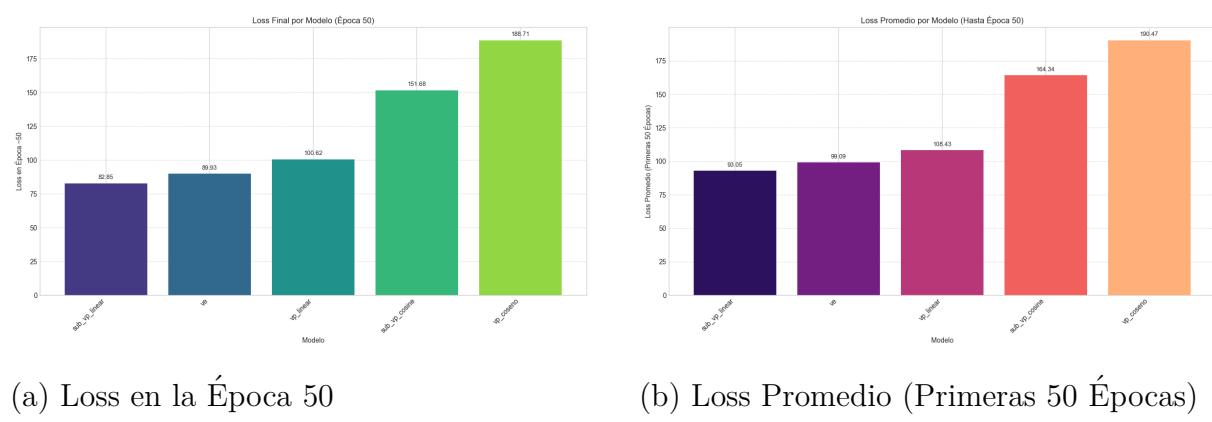
loss más elevados durante estas primeras 50 épocas, sugiriendo una menor eficacia en la minimización de la función de pérdida bajo esta configuración específica en la fase inicial.

El diagrama de violín (Figura 4.13b) enriquece este análisis al proporcionar una perspectiva sobre la distribución completa de los valores de loss promedio por época para cada modelo, acumulados a lo largo de las 50 épocas. Esta representación no solo corrobora las observaciones sobre los niveles medios de loss (visibles en la concentración de la densidad y la posición de las medianas implícitas), sino que también revela la variabilidad y la dispersión de dichos valores. Así, se confirma que `sub_vp_linear` y `ve` concentran la mayor parte de su distribución de loss en la región inferior, con violines más estrechos en esa zona, indicando una minimización más efectiva y consistente de la función de pérdida. El modelo `vp_linear` muestra una distribución similar, pero con su densidad principal desplazada ligeramente hacia valores superiores. Por otro lado, `vp_coseno` y, en menor medida, `sub_vp_cosine`, presentan distribuciones centradas en valores de loss notablemente más altos y con una mayor dispersión, lo que es coherente con su evolución temporal y sugiere una mayor variabilidad en el rendimiento entre épocas o una convergencia menos decidida hacia valores bajos durante este periodo.

En conjunto, estas dos primeras visualizaciones sugieren que las combinaciones de SDE y *schedule* de ruido representadas por los modelos `sub_vp_linear` y `ve` son las más prometedoras en términos de optimización de la función de pérdida durante la fase inicial del entrenamiento.

### Análisis Cuantitativo del Loss Final y Promedio (50 Épocas)

Para complementar el análisis visual de las curvas de convergencia, la Figura 4.14 presenta diagramas de barras que resumen cuantitativamente el rendimiento del entrenamiento al cabo de las 50 primeras épocas. Específicamente, se muestra: (a) el valor de la función de pérdida alcanzado en la época 50 y (b) el valor promedio de la función de pérdida calculado sobre la totalidad de estas 50 épocas para cada uno de los cinco modelos SDE.



(a) Loss en la Época 50

(b) Loss Promedio (Primeras 50 Épocas)

**Figura 4.14:** Resumen cuantitativo del rendimiento del entrenamiento tras 50 épocas:  
 (a) Valor del loss en la época 50. (b) Valor del loss promedio acumulado durante las primeras 50 épocas.

**Observaciones y Conclusiones (Diagramas de Barras del Loss a 50 Épocas):**

Ambas gráficas de barras (Figura 4.14) confirman la jerarquía de rendimiento de los modelos observada en la evolución temporal y las distribuciones de densidad. El modelo `sub_vp_linear` consistentemente logra el loss más bajo, tanto al considerar el valor final en la época 50 (Figura 4.14a) como el promedio acumulado durante las 50 épocas (Figura 4.14b). Le siguen de cerca, en términos de rendimiento, los modelos `ve` y `vp_linear`, los cuales también presentan valores de loss competitivos y significativamente inferiores a las configuraciones basadas en *schedules* cosenoidales en este tramo inicial.

Los modelos `sub_vp_cosine` y, de manera más acentuada, `vp_coseno`, muestran un loss considerablemente mayor en ambas métricas resumidas. Esto indica una menor efectividad en la optimización de la función de objetivo durante estas primeras 50 épocas para estas configuraciones específicas. El análisis del loss final en la época 50 refuerza las tendencias observadas en el promedio, destacando la eficiencia superior de las configuraciones `sub_vp_linear` y `ve` en la fase temprana del entrenamiento.

Este desempeño superior en la minimización del loss por parte de `sub_vp_linear` y `ve` en las etapas iniciales podría ser un indicativo de una capacidad más rápida para aprender las características fundamentales de la distribución de datos o una mayor adecuación de estas formulaciones SDE para la arquitectura de red y el conjunto de datos utilizados. No obstante, es crucial extender este análisis a la totalidad del proceso de entrenamiento y complementarlo con métricas de calidad de imagen para una evaluación exhaustiva del rendimiento global de cada modelo.

# 5 Conclusiones

Este proyecto se ha centrado en el estudio, desarrollo, implementación y evaluación de un sistema para la generación de imágenes de alta fidelidad, utilizando el marco de las Ecuaciones Diferenciales Estocásticas (SDEs) para modelos de difusión. Los principales logros y contribuciones se pueden resumir de la siguiente manera:

Se ha desarrollado un marco de software modular y extensible en Python, con PyTorch como librería principal. Este marco facilita la experimentación con diversos componentes de los modelos de difusión basados en SDEs, incluyendo diferentes formulaciones de SDEs (VE-SDE, VP-SDE, Sub-VP SDE), algoritmos de muestreo (Euler-Maruyama, Predictor-Corrector, ODE de flujo de probabilidad), y arquitecturas de modelos de score (basadas en U-Net).

La investigación ha permitido implementar y analizar distintas formulaciones de SDEs y sus procesos de discretización temporal, así como comparar la efectividad y eficiencia de varios algoritmos de muestreo. Se han entrenado modelos de estimación de score capaces de aproximar el gradiente de la densidad de probabilidad logarítmica de los datos perturbados.

La evaluación sistemática de las imágenes generadas, tanto cuantitativa (FID, IS, BPD) como cualitativa, ha sido un pilar fundamental. Los resultados obtenidos con conjuntos de datos como CIFAR-10 y MNIST han permitido analizar la calidad, diversidad y fidelidad de las imágenes. Destaca que el modelo **SubVP-SDE Lineal** ha emergido como el más prometedor en términos de calidad perceptual y diversidad (mejores FID e IS), mientras que los modelos **SubVP-SDE con schedule cosenoidal** han demostrado una capacidad superior para modelar la verosimilitud de los datos (mejor BPD). Esto evidencia un *trade-off* entre la optimización de la verosimilitud y la calidad perceptual.

Además, se ha explorado la flexibilidad del marco SDE para tareas de generación controlada. Se implementó la generación condicional por clase mediante el guiado por clasificador, utilizando un clasificador dependiente del tiempo (arquitectura `TimeDependentWideResNet`) junto con los modelos de score incondicionales. Los experimentos con CIFAR-10 demostraron la capacidad del sistema para generar imágenes coherentes con la clase objetivo. El modelo **SubVP-SDE Lineal** también ofreció resultados visualmente superiores en esta tarea. También se abordó la imputación de regiones faltantes en imágenes, donde el sistema fue capaz de llenar máscaras (derivadas de MNIST) aplicadas a imágenes de CIFAR-10 de manera coherente con el contexto visual.

El análisis de la convergencia del entrenamiento reveló que los modelos **SubVP-SDE Lineal** y **VE-SDE** mostraron una optimización más eficiente de la función de pérdida en las etapas iniciales del entrenamiento en comparación con otras configuraciones, particularmente aquellas con *schedules* cosenoidales, que aunque resultaron en mejores BPD, tuvieron un rendimiento inferior en FID e IS.

## Limitaciones y Líneas Futuras

A pesar de los logros, el proyecto presenta ciertas limitaciones. El entrenamiento de los modelos de score y los clasificadores dependientes del tiempo es computacionalmente costoso, lo que ha restringido el número de épocas de entrenamiento y la exhaustividad de algunas comparativas. La evaluación de métricas como FID también se realizó sobre subconjuntos de datos debido a estas limitaciones, lo que implica que los valores absolutos podrían variar. La volatilidad inherente a las métricas de evaluación de modelos generativos también es un factor a considerar, y ninguna métrica por sí sola captura todos los aspectos deseados de la generación.

Como líneas de trabajo futuras, se podrían explorar:

- La optimización y aceleración de los algoritmos de muestreo para reducir el tiempo de generación.
- El entrenamiento de los modelos con mayor cantidad de datos y durante más épocas para mejorar la calidad de las imágenes.
- La extensión del marco a imágenes de mayor resolución y a otros tipos de datos.
- Una investigación más profunda sobre el *trade-off* entre métricas de verosimilitud y de calidad perceptual.
- La implementación de técnicas de guiado sin clasificador (*classifier-free guidance*) para la generación condicional, que han demostrado ser muy efectivas.
- Una exploración más exhaustiva de diferentes arquitecturas para los modelos de score y los clasificadores.
- La aplicación del sistema a dominios específicos con necesidades particulares de generación de imágenes.

En definitiva, este trabajo ha permitido no solo replicar y comprender técnicas avanzadas en el campo de la IA generativa, sino también desarrollar un conjunto de herramientas robustas y realizar un análisis sistemático de sus componentes, sentando las bases para futuras investigaciones y aplicaciones en la generación de imágenes mediante modelos de difusión basados en SDEs.

# Bibliografía

- [1] B. D. O. Anderson. «Reverse-time diffusion equation models». En: *Stochastic Processes and their Applications* 12.3 (1982), págs. 313-326. DOI: [10.1016/0304-4149\(82\)90052-3](https://doi.org/10.1016/0304-4149(82)90052-3).
- [2] T. Q. Chen et al. «Neural Ordinary Differential Equations». En: *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*. Ed. por S. Bengio et al. 2018, págs. 6571-6583.
- [3] A. Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. Inf. téc. Master's thesis. Available at <http://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>. University of Toronto, 2009.
- [4] P. Dhariwal y A. Q. Nichol. «Diffusion models beat GANs on image synthesis». En: *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*. 2021, págs. 8780-8794.
- [5] L. Dinh, J. Sohl-Dickstein y S. Bengio. «Density estimation using Real NVP». En: *International Conference on Learning Representations (ICLR)*. arXiv preprint arXiv:1605.08803 (2016). 2017.
- [6] I. J. Goodfellow et al. «Generative Adversarial Nets». En: *Advances in Neural Information Processing Systems 27 (NIPS 2014)*. Ed. por Z. Ghahramani et al. Curran Associates, Inc., 2014, págs. 2672-2680.
- [7] J. Ho, A. Jain y P. Abbeel. «Denoising Diffusion Probabilistic Models». En: *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*. Ed. por H. Larochelle et al. 2020, págs. 6840-6851.
- [8] A. Hyvärinen. «Estimation of Non-Normalized Statistical Models by Score Matching». En: *Journal of Machine Learning Research* 6 (2005), págs. 695-709.
- [9] D. P. Kingma y P. Dhariwal. «Glow: Generative Flow with Invertible 1x1 Convolutions». En: *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*. Ed. por S. Bengio et al. Curran Associates, Inc., 2018, págs. 10215-10224.
- [10] D. P. Kingma y M. Welling. «Auto-Encoding Variational Bayes». En: *arXiv preprint arXiv:1312.6114* (2013). arXiv: [1312.6114](https://arxiv.org/abs/1312.6114).
- [11] Y. LeCun et al. «Gradient-Based Learning Applied to Document Recognition». En: *Proceedings of the IEEE* 86.11 (1998), págs. 2278-2324. DOI: [10.1109/5.726791](https://doi.org/10.1109/5.726791).
- [12] A. Q. Nichol y P. Dhariwal. «Improved Denoising Diffusion Probabilistic Models». En: *Proceedings of the 38th International Conference on Machine Learning (ICML 2021)*. Ed. por M. Meila y T. Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, págs. 8162-8171.

- [13] O. Ronneberger, P. Fischer y T. Brox. «U-Net: Convolutional Networks for Biomedical Image Segmentation». En: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Vol. 9351. Lecture Notes in Computer Science. Springer International Publishing, 2015, págs. 234-241. DOI: [10.1007/978-3-319-24574-4\\_28](https://doi.org/10.1007/978-3-319-24574-4_28).
- [14] J. Sohl-Dickstein et al. «Deep Unsupervised Learning using Nonequilibrium Thermodynamics». En: *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*. Ed. por F. Bach y D. Blei. Vol. 37. JMLR Workshop and Conference Proceedings. JMLR.org, 2015, págs. 2256-2265.
- [15] Y. Song y S. Ermon. «Generative Modeling by Estimating Gradients of the Data Distribution». En: *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*. Ed. por H. M. Wallach et al. 2019, págs. 11917-11928.
- [16] Y. Song et al. «Score-Based Generative Modeling through Stochastic Differential Equations». En: *International Conference on Learning Representations (ICLR)*. 2021.
- [17] P. Vincent. «A Connection Between Score Matching and Denoising Autoencoders». En: *Neural Computation* 23.7 (2011), págs. 1661-1674. DOI: [10.1162/NECO\\_a\\_00142](https://doi.org/10.1162/NECO_a_00142).
- [18] S. Zagoruyko y N. Komodakis. «Wide Residual Networks». En: *arXiv preprint arXiv:1605.07146* (2016).