UNIVERSITAT DE BARCELONA

Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

Copia de modelos usando modelos de difusión en MNIST

Joaquín Beas Ruiz

Director: Nahuel Norberto Statuto Pérez i

Julio Cezar Silveira Jacques-Junior

Realizado : Departament de

Matemàtiques i Informàtica

Barcelona, 08 de Juny de 2025

Resum

Aquest projecte investiga la tècnica de *replicació de coneixement* que aprofita el poder dels models de difusió per generar dades sintètiques d'alta qualitat. L'objectiu principal és determinar si un model estudiant pot assolir un rendiment comparable al d'un model professor entrenat amb dades reals, però utilitzant exclusivament dades sintètiques.

Utilitzant el conjunt de dades MNIST com a cas d'estudi, aquest treball explora les fronteres de la generació d'imatges mitjançant models de difusió i la seva aplicació en la replicació de coneixement. A més, el projecte avalua l'impacte de diferents arquitectures en els models professor i estudiant, proporcionant una anàlisi comparativa de la seva eficàcia en aquest context innovador.

Resumen

Este proyecto investiga la técnica de *replicación de conocimiento* que aprovecha el poder de los modelos de difusión para generar datos sintéticos de alta calidad. El objetivo principal es determinar si un modelo estudiante puede alcanzar un rendimiento comparable al de un modelo profesor entrenado con datos reales, pero utilizando exclusivamente datos sintéticos.

Utilizando el conjunto de datos MNIST como caso de estudio, este trabajo explora las fronteras de la generación de imágenes mediante modelos de difusión y su aplicación en la replicación de conocimiento. Además, el proyecto evalúa el impacto de diferentes arquitecturas en los modelos profesor y estudiante, proporcionando un análisis comparativo de su eficacia en este contexto innovador.

Summary

This project investigates the *knowledge replication* technique that leverages the power of diffusion models to generate high-quality synthetic data. The main objective is to determine whether a student model can achieve comparable performance to a teacher model trained on real data, but using exclusively synthetic data.

Using the MNIST dataset as a case study, this work explores the frontiers of image generation through diffusion models and their application in knowledge replication. Additionally, the project evaluates the impact of different architectures in both the teacher and student models as well as in the diffusion generator, providing a comparative analysis of their effectiveness in this innovative context.

Contents

1. Introducción

1.1. Motivación y contexto

En la última década, los modelos de aprendizaje profundo han mostrado un rendimiento sobresaliente gracias, en buena medida, a la disponibilidad de grandes volúmenes de datos etiquetados. Sin embargo, manejar esos volúmenes plantea varios obstáculos, entre los cuales los principales son:

- **Privacidad y regulación**. La normativa europea de protección de datos (GDPR) establece requisitos estrictos sobre la recogida, el almacenamiento y la difusión de información personal [1]. Compartir conjuntos de datos originales o reutilizarlos en entornos distintos puede resultar inviable si contienen cualquier dato identificable; este reto se mitiga eficazmente mediante el uso de datos sintéticos generados con modelos generativos que preservan la privacidad [2].
- **Coste de obtención y sesgos de distribución.** Etiquetar manualmente millones de ejemplos requiere tiempo, personal cualificado y presupuesto, y los datos históricos suelen arrastrar desequilibrios que se trasladan al rendimiento y la equidad de los modelos.
- **Escalabilidad ante grandes volúmenes.** Conjuntos cada vez mayores exigen más almacenamiento y potencia de cómputo tanto para el entrenamiento como para simples tareas de transferencia o auditoría, lo que eleva el coste operativo y dificulta la colaboración entre grupos de investigación.

Para superar estas barreras se combinan dos enfoques: la destilación de conocimiento y la generación de datos sintéticos. En la destilación de conocimiento, primero se entrena un modelo amplio y altamente preciso (teacher) utilizando los datos reales, lo cual captura toda la información relevante sin necesidad de exponer los ejemplos originales a terceros. A continuación, se transfiere ese conocimiento al modelo más compacto (student) a partir de las predicciones definitivas del teacher, de manera que nunca se comparten las muestras originales y se minimiza el tamaño del modelo final. Para cada muestra, el modelo teacher asigna la clase de máxima probabilidad como etiqueta dura, reduciendo drásticamente el espacio de almacenamiento y acelerando el entrenamiento. Al prescindir de anotaciones manuales, también se evitan los sesgos inherentes a los datos originales.

Por su parte, la generación de datos sintéticos recurre a modelos difusores capaces de aprender la distribución estadística de los ejemplos reales y producir nuevas instancias que conservan la información útil para el aprendizaje, pero sin contener

ningún dato personal identificable. De esa forma, se respeta la normativa de protección de datos y se evita la necesidad de compartir o almacenar conjuntos originales; al mismo tiempo, se amplía y diversifica el conjunto de entrenamiento sin incurrir en los elevados costes de etiquetado manual, lo que ayuda a mitigar los sesgos de distribución. También se alivia la presión de escalabilidad, ya que con muestras sintéticas es posible enriquecer el volumen de datos disponibles sin imponer mayores exigencias de almacenamiento o de cómputo para entrenar directamente sobre los datos reales. En conjunto, la destilación de conocimiento permite equilibrar privacidad y eficiencia computacional, mientras que la generación de datos sintéticos aporta anonimato y diversidad de muestras, abordando de manera sinérgica cada uno de los obstáculos asociados a los grandes volúmenes de datos en aprendizaje profundo. No obstante, diferencias en la distribución sintética frente a la real pueden traducirse en pérdida de rendimiento, Shrivastava demostró que entrenar un clasificador con imágenes simuladas sin adaptación de dominio provocó una caída de precisión de aproximadamente un 8 % al evaluarlo sobre datos reales [**?**]. En este trabajo evaluaremos detalladamente el impacto de dicha discrepancia midiendo la precisión del modelo student en función de la proporción y calidad de los datos sintéticos generados.

1.2. Objetivos del proyecto

El presente trabajo de fin de grado persigue los siguientes objetivos principales:

- **Entrenamiento del modelo teacher para MNIST.** El teacher se entrena exclusivamente con el MNIST original 60 000 imágenes de entrenamiento y 10 000 de prueba utilizando las etiquetas reales. Empleamos una ResNet PreActivation de 270 000 parámetros, optimizador Adam (lr = 0,002), batch size = 64 y early stopping según la pérdida de validación, alcanzando 99,4 % de precisión en el test oficial.

- **Entrenamiento del sampler de difusión.** Para la generación de datos sintéticos utilizamos un modelo de difusión basado en la formulación de *Denoising Diffusion Probabilistic Models* [3], implementado como una U-Net condicional que aprende a añadir ruido gaussiano progresivo y luego a revertirlo. Nos hemos basado en la implementación disponible en GitHub [1] como punto de partida, adaptándola para trabajar con imágenes de 28×28 px y etiquetas duras.

Generar un conjunto de datos sintético

[1] https://github.com/bot66/MNISTDiffusion

- Generar múltiples imágenes con el modelo de difusión.
- Etiquetando dichas muestras mediante el modelo teacher solo cuando la confianza de la predicción hecha por el clasificador teacher supera un umbral de confianza.
- Equilibrando las clases para obtener un dataset balanceado.

Entrenar un modelo student exclusivamente sobre el dataset sintético generado, explorando arquitecturas similares o simplificadas respecto al teacher.

Evaluar y comparar el rendimiento de los modelos teacher y student en el conjunto de prueba original de MNIST, analizando métricas de precisión, matriz de confusión y coste computacional.

Analizar la influencia de la calidad y cantidad de datos sintéticos en el proceso de distilación de conocimientos, determinando umbrales de confianza y tamaños de muestra óptimos.

Proponer posibles ampliaciones y aplicaciones prácticas como adaptación a otros conjuntos de datos y estudios de robustez frente a ruido adversarial.

Estos objetivos guían el desarrollo del trabajo y definen lo que será la *pipeline* principal del proyecto (véase Figura 4).

A continuación, se describe la organización por capítulos de este documento. El Capítulo 2 se centra en el marco teórico, donde se explican en detalle los modelos de difusión, las bases de la replicacion de conocimiento y se introduce la implementación de referencia para MNIST. En el Capítulo 3 se expone la metodología seguida: diseño e implementación del modelo teacher, generación del dataset sintético y entrenamiento del modelo student. El Capítulo 4 está dedicado a los resultados experimentales, donde se muestran las métricas de rendimiento, las comparativas teacher vs. student y el análisis del efecto de los datos sintéticos. Finalmente, en el Capítulo 5 se ofrecen las conclusiones del trabajo, se discuten las limitaciones detectadas y se plantean líneas futuras de investigación y aplicaciones prácticas.

2. Marco teórico

2.1. Replicación de conocimiento

La destilación de conocimiento, tal y como la definieron Hinton, Vinyals y Dean en [4], es una técnica de *machine learning* que transfiere los aprendizajes de un modelo grande ya entrenado (modelo profesor) a otro más pequeño (modelo estudiante) con el fin de comprimir y acelerar el proceso de predicción sin perder demasiada precisión [5]. Su relevancia radica en que, en la práctica, los modelos con mejor rendimiento suelen ser demasiado lentos o costosos para muchos casos de uso. Este proceso permite entrenar primero el modelo grande para extraer la estructura de los datos y después destilar esa capacidad en un modelo ligero apto para su ejecución.

De manera similar a la destilación, nuestro proceso de replicación de conocimiento comienza con el entrenamiento completo de un clasificador teacher sobre MNIST real, que después se mantiene congelado para guiar la fase de generación. Un modelo de difusión, también conocido como *sampler*, produce nuevas imágenes de dígitos y, para cada una, se toma la clase que el teacher asigna con mayor probabilidad como etiqueta definitiva. Finalmente, entrenamos al modelo student únicamente con este conjunto sintético y etiquetado, de modo que aprenda a imitar el comportamiento del teacher sin recurrir nunca a los datos originales ni a sus activaciones internas [3].

El empleo de un modelo *student* destilado ofrece un coste de inferencia y un consumo de recursos sustancialmente menores con una pérdida mínima de precisión: Hinton et al. [4] demostraron que más del 80 % de la mejora obtenida por un *ensemble* de diez modelos grandes puede transferirse a un único modelo destilado, alcanzando casi el mismo rendimiento con una complejidad notablemente inferior. De modo análogo, Triastcyn y Faltings [2] mostraron que un *student* entrenado únicamente con datos sintéticos de alta fidelidad alcanza un 98.3 % de precisión en MNIST (frente al 99.2 % del *teacher*), e incluso un algoritmo sencillo como la regresión logística puede lograr un 91.7 % sin acceder al *teacher* ni al conjunto de datos original, lo que incrementa la flexibilidad y refuerza la privacidad al no exponer ni el modelo grande ni los datos subyacentes.

Desplegamos el modelo *student* en lugar del *teacher* porque su arquitectura más ligera reduce drásticamente el tiempo de inferencia y los recursos necesarios, facilitando su uso en dispositivos con capacidad limitada. Además, entrenarlo con datos sintéticos en lugar de reales protege la privacidad del conjunto original y permite generar muestras de forma ilimitada sin comprometer la confidencialidad de la información subyacente, manteniendo al mismo tiempo un desempeño cercano al del modelo grande.

2.2. Modelos de difusión

2.2.1. Funcionamiento

Los modelos de difusión son una familia de modelos generativos que aprenden a transformar ruido gaussiano en datos coherentes (en nuestro caso imágenes). Lo hacen entrenando una red neuronal para imitar, paso a paso, la inversión de un proceso físico de difusión: primero "ensucian" progresivamente una muestra real con ruido aleatorio y, a continuación, aprenden a revertir cada uno de esos pasos hasta reconstruir la señal original.

A diferencia de otras técnicas generativas, la difusión combina estabilidad en el entrenamiento y gran calidad en la síntesis, situándose en la vanguardia de la IA generativa empleada por sistemas como Stable Diffusion [6], DALL-E 2 [7] o Midjourney [8].

Históricamente, los primeros algoritmos se remontan a 2015 y se reforzaron con los Denoising Diffusion Probabilistic Models (DDPM) de 2020 [3], que demostraron igualar la calidad de las GAN .

2.2.2. Proceso de difusión hacia adelante y reversa

1. **Difusión hacia adelante (forward)**

   Como podemos observar en la Figura 1, el proceso aplica una cadena de Markov que añade una fracción de ruido gaussiano en cada paso hasta que, tras T iteraciones, la muestra se convierte en ruido puro.

   Step of forward diffusion:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \tag{1}$$

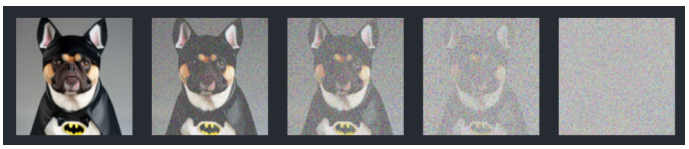

Figure 1: Proceso de difusión hacia adelante

2. **Difusión inversa (reverse)**

   La Figura 2 ilustra cómo una red aprende a predecir el ruido presente en cada estado x_t para restaurar x_{t-1}. Durante el entrenamiento se minimiza el error entre el ruido real y el ruido estimado, lo que equivale a reducir el MSE o a igualar la "puntuación" de la distribución de datos.

Fórmula output paso $t-1$:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{2}$$

Jonathan Ho. [3] muestran que, en lugar de optimizar toda la evidencia lower bound, basta con minimizar la pérdida

$$\mathcal{L}_{\text{simple}}(\theta) \;=\; \mathbb{E}_{x_0,\,\epsilon,\,t}\big[\|\epsilon - \epsilon_\theta(x_t, t)\|^2\big],$$

donde $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon$, $\epsilon \sim \mathcal{N}(0, I)$ y $\epsilon_\theta(x_t, t)$ es la predicción de ruido de la red. Esta formulación permite entrenar la red de denoising paso a paso para revertir el ruido añadido a lo largo de la cadena [3].

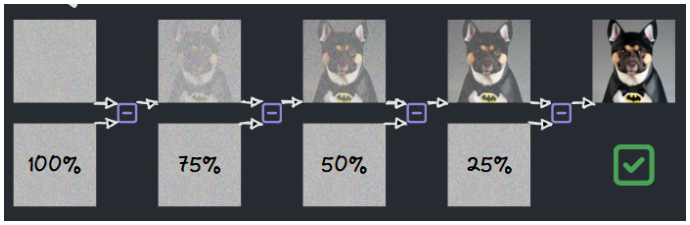

Figure 2: Proceso de difusión reversa. Los valores en porcentaje indican la fracción de ruido residual respecto al ruido inicial tras cada paso de denoising.

3. **Generación**

   Para sintetizar una muestra, se inicializa con ruido gaussiano $x_T \sim \mathcal{N}(0, I)$ y se aplica la difusión inversa paso a paso, hasta obtener x_0, que pertenece a la distribución de los datos.

En la práctica, la velocidad-calidad depende del número de pasos de muestreo: menos pasos dan lugar a una generación de imagen más rápida pero pueden degradar el detalle; más pasos mejoran la fidelidad a costa de tiempo de cómputo.

2.3. Dataset MNIST

El conjunto de datos MNIST [9] comprende un total de 70.000 imágenes digitales de dígitos manuscritos, distribuidas en 60.000 muestras para entrenamiento y 10.000 para evaluación, abarcando las diez clases numéricas del 0 al 9. Cada imagen se presenta en escala de grises con una resolución de 28×28 píxeles, resultado de la normalización aplicada a los datos originales del National Institute of Standards and Technology (NIST). La distribución de las clases se encuentra aproximadamente equilibrada, con alrededor de 6.000 muestras por cada dígito, lo que facilita la obtención de métricas comparables entre las diferentes categorías. Desde su creación en

1994 por LeCun, Cortes y Burges, MNIST [10] se estableció como el primer benchmark estandarizado para el desarrollo de redes neuronales , convolucionales, manteniendo su relevancia histórica al continuar apareciendo en publicaciones que proponen nuevas arquitecturas, técnicas de regularización o métodos generativos. Sin embargo, la simplicidad inherente del conjunto de datos, caracterizada por dígitos bien alineados y un bajo nivel de ruido de fondo, ha resultado en que las mejoras de precisión sean cada vez más marginales, típicamente inferiores al 0.1% de error. Esta limitación ha motivado el desarrollo de variantes más desafiantes como Fashion-MNIST [11], EMNIST [12] y QMNIST [13] , que proporcionan mayor dificultad y diversidad cuando los investigadores requieren evaluar algoritmos en contextos más complejos.

3. Metodología e Implementación

3.1. Pipeline del proyecto

La Tabla 1 muestra el pipeline completo del proyecto, que se compone de cinco fases secuenciales orquestadas desde el script `main.py` mediante flags de línea de comandos:

Table 1: Pipeline del proyecto

Paso	**Acción**	**Artefactos producidos**
Teacher MNIST	Entrenar un clasificador teacher sobre los 60.000 ejemplos reales de MNIST	`teacher.pt`
Difusión	Entrenar un modelo de difusión condicional (por defecto ConditionalUNet) con 1.000 pasos de ruido lineal β_t	`diffusion.pt`, muestras PNG
Dataset sintético	Generar dígitos mediante la difusión; el teacher los etiqueta	`train-images-synthetic.gz`, `train-labels-synthetic.gz`
Student Synthetic	Entrenar un modelo student únicamente con el dataset sintético (misma arquitectura o versión reducida)	`student.pt`
Testing y evaluación	Medir el rendimiento final: Precisión y pérdida de teacher y student en el test set de MNIST, Calidad de las muestras de difusión.	`Métricas CSV / gráficas PNG`

3.2. Arquitecturas y modelos

3.2.1. Modelos de clasificación

CNN simple La Tabla 2 muestra nuestra CNN una básica, que a pesar de su simplicidad alcanza un 99% de precisión en MNIST. La red toma una imagen de 28×28 píxeles y la procesa a través de dos bloques convolucionales: el primero crea 32 mapas de características y el segundo los expande a 64, cada uno seguido de ReLU y reducción de tamaño mediante MaxPooling. Después, la información pasa por dos

capas completamente conectadas con dropout para prevenir el sobreajuste, terminando en 10 neuronas que representan cada dígito. Con sus 421,642 parámetros, esta red sirve como punto de referencia para comparar arquitecturas más complejas y como base para el modelo student.

Table 2: Arquitectura CNN simple

Capa	**Dimensiones entrada → salida**	**Parámetros**	**Activación**
Conv2d	$1 \times 28 \times 28 \rightarrow 32 \times 28 \times 28$	320	ReLU
MaxPool2d	$32 \times 28 \times 28 \rightarrow 32 \times 14 \times 14$	0	—
Conv2d	$32 \times 14 \times 14 \rightarrow 64 \times 14 \times 14$	18,496	ReLU
MaxPool2d	$64 \times 14 \times 14 \rightarrow 64 \times 7 \times 7$	0	—
Dropout	$64 \times 7 \times 7$	0	—
Linear	$3136 \rightarrow 128$	401,536	ReLU
Dropout	128	0	—
Linear (out)	$128 \rightarrow 10$	1,290	—
Softmax	$10 \rightarrow 10$	0	Softmax
Total parámetros: 421,642			

CNN comprimida Buscando mejorar la eficiencia sin sacrificar el rendimiento, desarrollamos una versión más compacta de CNN que destaca por su uso inteligente de técnicas modernas de optimización. Esta arquitectura, que se detalla en la Tabla 3, incorpora BatchNorm después de cada convolución para estabilizar el entrenamiento y emplea una innovadora estrategia de compresión de canales ($1 \rightarrow 8 \rightarrow 16 \rightarrow 4 \rightarrow 8 \rightarrow 16$). Su diseño más distintivo aparece en la capa final: en lugar de las tradicionales capas densas, utiliza global average pooling seguido de una convolución 1×1, una decisión que reduce drásticamente el número de parámetros. El resultado es sorprendente: manteniendo un rendimiento similar a la CNN simple, esta red requiere solo 5,328 parámetros. De la misma manera que el modelo anterior, esta destinado a ser usado como student con el fin de testear como afecta la compresion de un modelo al rendimiento del modelo.

ResNet PreAct La arquitectura más sofisticada de nuestro estudio implementa una variante de las redes residuales que redefine el orden tradicional de las operaciones. En la búsqueda de un modelo robusto y muy preciso y de un tamaño moderadamente pequeño, nos hemos basado en el repositorio de GitHub de hysts[2]para crear el modelo ResNet PreAct, cuya estructura se detalla en la Tabla 4. Inicia con una convolución que expande la entrada a 16 canales, para luego distribuir el procesamiento en tres etapas de bloques residuales. Cada etapa duplica progresivamente

[2] hysts, *pytorch_resnet_preact*, 2024. https://github.com/hysts/pytorch_resnet_preact

Table 3: Arquitectura CNN comprimida (MNISTNet1)

Capa	**Dimensiones entrada → salida**	**Parámetros**	**Activación**
Conv2d	$1 \times 28 \times 28 \rightarrow 8 \times 28 \times 28$	72	ReLU + BN
Conv2d	$8 \times 28 \times 28 \rightarrow 16 \times 28 \times 28$	1,152	ReLU + BN
MaxPool2d	$16 \times 28 \times 28 \rightarrow 16 \times 14 \times 14$	0	—
Conv2d 1×1	$16 \times 14 \times 14 \rightarrow 4 \times 14 \times 14$	64	ReLU + BN
Conv2d	$4 \times 14 \times 14 \rightarrow 8 \times 12 \times 12$	288	ReLU + BN
Conv2d	$8 \times 12 \times 12 \rightarrow 16 \times 10 \times 10$	1,152	ReLU + BN
Conv2d	$16 \times 10 \times 10 \rightarrow 16 \times 8 \times 8$	2,304	ReLU + BN
AvgPool2d	$16 \times 8 \times 8 \rightarrow 16 \times 1 \times 1$	0	—
Conv2d 1×1	$16 \times 1 \times 1 \rightarrow 10 \times 1 \times 1$	160	—
Softmax	$10 \times 1 \times 1 \rightarrow 10 \times 1 \times 1$	0	Softmax
Total parámetros: 5,328			

los canales ($16 \rightarrow 32 \rightarrow 64$) mientras reduce las dimensiones de la imagen, pero su verdadera innovación reside en aplicar BatchNorm y ReLU antes de las convoluciones. Esta sutil reorganización, junto con las conexiones residuales, optimiza el flujo de gradientes durante el entrenamiento, permitiendo que la red, con sus 272 666 parámetros, alcance una precisión excepcional del 99.4 % y se establezca como nuestro modelo teacher de referencia.

Table 4: Arquitectura ResNet PreAct

Capa / Bloque	**Dimensiones salida**	**Parámetros**
Conv2d	$1 \times 28 \times 28 \rightarrow 16 \times 28 \times 28$	144
3x BasicBlock	$16 \times 28 \times 28$	∼9,264
3x BasicBlock	$32 \times 14 \times 14$	∼29,312
3x BasicBlock	$64 \times 7 \times 7$	∼86,880
Linear	$64 \rightarrow 10$	650
Softmax	$10 \rightarrow 10$	0
Total parámetros: 272,666		

Los BasicBlocks consisten en dos capas convolucionales 3×3, cada una precedida por BatchNorm y ReLU. La salida del bloque se suma a su entrada mediante una conexión tipo "skip connection". Esta suma permite que el gradiente fluya sin obstáculos, lo que facilita el entrenamiento de redes más profundas incluso con pocos parámetros.

El primer Conv2d está preparado para imágenes RGB (3 canales), pero como MNIST tiene solo 1 canal, el modelo lo duplica internamente para reutilizar los mismos pesos. Tras la última etapa, se aplica BatchNorm y ReLU globales, seguido de un adaptive average pooling que deja un vector por canal. Una capa Linear de 10 salidas entrega los logits.

El diseño de los BasicBlocks permite crear un atajo que conecta la entrada con la salida, de modo que la información fluye sin trabas y el entrenamiento de redes muy profundas resulta mucho más sencillo. Al aplicar primero la normalización y la activación antes de cada convolución, se añade de forma natural un efecto de regularización que ayuda a evitar el sobreajuste. Todo ello en una arquitectura muy compacta: con menos de 0,5 M de parámetros se consigue una gran profundidad y un amplio contexto espacial. Gracias a esta combinación, la red supera el 99,4 % de precisión en MNIST, convirtiéndose en un teacher de referencia por su equilibrio entre simplicidad y rendimiento.

3.2.2. Modelos de difusión

Para la fase de generación empleamos una U-Net como *sampler*. Esta red consta de una ruta contratante que extrae representaciones de alto nivel reduciendo progresivamente la resolución, y una ruta expansiva que reconstruye la imagen original mediante upsampling y conexiones de salto (*skip connections*) que preservan el detalle fino.

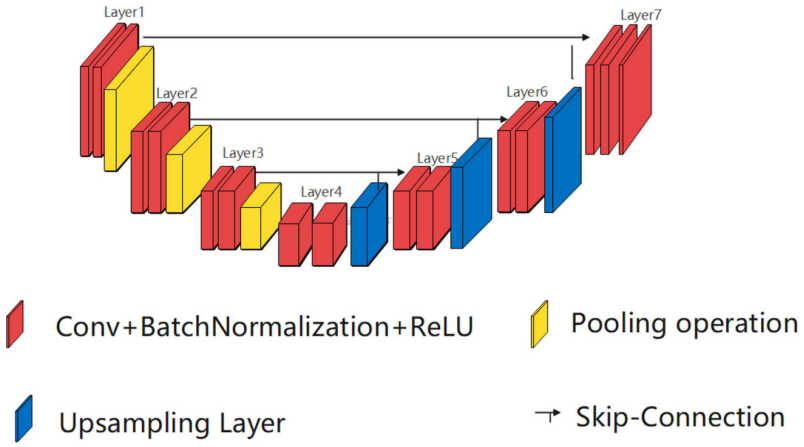

Figure 3: Arquitectura U-Net empleada como sampler.

La parte izquierda (contracción) reduce la resolución mientras aumenta los canales para extraer características, y la parte derecha (expansión) recupera la resolución original combinando información de las capas de contracción mediante conexiones de salto. Y a partir de esta U-Net, definimos luego nuestros dos modelos:

U-Net básico La U-Net que utilizamos es una adaptación compacta para imágenes MNIST ($1 \times 28 \times 28$) y se instancia dentro de `DiffusionUnet`.

Table 5: Estructura de la U-Net básica

Fase	**Objetivo**	**Implementación clave**
Entrada	Extraer rasgos iniciales	ConvBnSiLu (3×3, SiLU) eleva $1 \rightarrow 32$ canales
Codificador	Reducir resolución y aumentar canales	Cada EncoderBlock contiene: • $3 \times$ ResidualBottleneck ShuffleNet V2 (depth-wise + channel-shuffle) • TimeMLP que inyecta la embedding de tiempo t • ResidualDownsample (stride 2)
Cuello	Procesar en la resolución mínima	$2 \times$ ResidualBottleneck + 1 bottleneck de compresión (canales/2)
Decodificador	Recuperar resolución fusionando skips	DecoderBlock hace bilinear-upsample, concatena el skip y repite la secuencia bottleneck $\rightarrow$ TimeMLP $\rightarrow$ bottleneck
Salida	Predecir el ruido $\hat{\varepsilon}$	final_conv 1×1 sin activación (devuelve 1 canal)

Si analizamos la Tabla 5, que detalla la arquitectura del modelo, podemos ver que tiene como núcleo una U-Net: una red que primero encoge la imagen para entender el contexto global y después la vuelve a agrandar para reconstruir cada trazo.

La imagen de 28×28 px entra y, capa a capa, se reduce la resolución: $28 \rightarrow 14 \rightarrow 7 \rightarrow 4$ px. Cada vez que se reduce, duplicamos o cuadruplicamos los canales ($32 \rightarrow 64 \rightarrow 128 \rightarrow 256$) para que la red compense la pérdida de detalle aprendiendo más patrones. Los valores que detecta en cada escala no se desechan sino que se guardan para más tarde.

En la escala más pequeña (4×4 px) la red combina toda la información global. Aquí es donde decide si el fragmento ruidoso apunta a una clase en concreto.

Ahora la imagen se vuelve a agrandar paso a paso: $4 \rightarrow 7 \rightarrow 14 \rightarrow 28$ px. En cada escala copiamos los detalles limpios que habíamos guardado en la bajada y los pegamos sobre la reconstrucción. Así evitamos que el modelo tenga que inventar bordes que ya conocía.

Dentro de cada bloque no usamos las típicas convoluciones, sino ShuffleNet V2 [14]. Esta técnica divide canales. ShuffleNet los parte en dos mitades y procesa solo una mitad con convoluciones depth-wise. Estas convoluciones aplican un filtro a cada canal por separado, mucho más baratas que las normales, deja pasar la otra mitad sin apenas tocarla, después baraja los canales. Las dos mitades se mezclan, de modo que en la siguiente capa cada canal "ve" información de la otra mitad. Por último une todo con un atajo residual.

Al samplear una imagen con este modelo partimos del ruido puro y hemos de reconstruir una imagen coherente paso a paso. En cada paso t, el modelo predice el ruido y lo usa para calcular la media condicional del siguiente paso x_{t-1}:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \varepsilon_\theta(x_t, t) \right) \tag{3}$$

Luego se suma ruido gaussiano con varianza:

$$\sigma_t^2 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \beta_t \tag{4}$$

El proceso se repite hasta que $t = 1$.

Al reconstruir la imagen x_0 (al final del proceso), el modelo puede producir valores fuera del rango $[-1, 1]$. Esto genera artefactos o saturaciones. Para evitarlo, se introduce el clipping de $\hat{x}_0$:

$$\hat{x}_0^{\text{clipped}} = \text{clip}(\hat{x}_0, -1, 1) \tag{5}$$

Usamos luego esta $\hat{x}_0$ clippeada para calcular la media de esta manera evitamos saturaciones y generamos muestras más limpias, sobre todo en los primeros pasos.

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \varepsilon_\theta(x_t, t) \right) \tag{6}$$

U-Net condicional Nuestro Conditional DDPM se basa directamente en el modelo U-Net anterior [3] y toma como punto de partida la implementación de TeaPearce disponible en GitHub [3]. Usamos la técnica de *classifier-free guidance* descrita en [3] para generar imágenes condicionadas en una clase.

De manera análoga a los esquemas de extracción de clasificadores en caja negra (donde se generan datos sintéticos y se envían al modelo "teacher" para recopilar etiquetas) [15, 16], la U-Net condicional que describimos a continuación se utiliza para producir dígitos sintéticos.

La finalidad de implementar este modelo es que a la hora de generar nuestro dataset sintético queremos decidir, en cada momento, qué dígito producir. De ese modo podemos balancear las diez clases para que el student reciba exactamente el mismo número de "0", "1", ... "9", también podemos reforzar las clases que el teacher clasifica peor.

Este modelo tiene la misma base que el modelo U-Net anteriormente explicado pero ahora escucha la muestra a generar.

[3] TeaPearce, *Conditional_Diffusion_MNIST*, 2022. https://github.com/TeaPearce/Conditional_Diffusion_MNIST

Para darle esa instrucción, tomamos la clase numérica y la convertimos en un vector one-hot. Ese vector es demasiado brusco para que la red lo use directamente, de modo que lo pasamos por un pequeño Multilayer Perceptron. El MLP actúa como un traductor: recibe el one-hot, aplica un par de capas lineales con activaciones no lineales y devuelve un embedding mucho más rico y suave. Dicho embedding se inyecta en los bloques internos de la U-Net como un "tono de fondo" que guía al sampling a generar una clase determinada.

Durante el entrenamiento, aproximadamente en un 10% de los lotes quitamos el embedding de clase, la U-Net recibe la imagen ruidosa, pero no la pista de qué dígito es. Así aprende a desenvolverse tanto con etiqueta como sin ella.

Cuando queremos generar un dígito concreto, partimos de ruido puro y ejecutamos el proceso inverso paso a paso, pero ahora guiamos cada predicción con la etiqueta de clase:

Para iniciar cada batch, se parte de ruido puro x_T, tal como se ilustra en la Figura 24. A continuación, se duplica el lote original dividiéndolo en dos mitades: la primera recibe la etiqueta de clase c (context-on) y la segunda omite dicha etiqueta (context-off, con máscara = 1), de modo que la red aprenda a operar tanto con como sin información de clase.

Durante la difusión reversa, que recorre los pasos desde $t = T$ hasta $t = 1$, en cada iteración el modelo efectúa primero la predicción del ruido con y sin contexto:

$$\varepsilon_{\text{context}} = \varepsilon_\theta(x_t, t, c), \quad \varepsilon_{\text{no-context}} = \varepsilon_\theta(x_t, t, \emptyset).$$

A continuación se aplica la mezcla guiada mediante un parámetro de escala $w \geq 0$, definiendo

$$\varepsilon_{\text{guided}} = \varepsilon_{\text{no-context}} + w\,\big(\varepsilon_{\text{context}} - \varepsilon_{\text{no-context}}\big).$$

Con este ruido guiado se reconstruye una versión menos ruidosa de la imagen. Primero se estima la imagen limpia:

$$\hat{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\Big(x_t - \sqrt{1-\bar{\alpha}_t}\,\varepsilon_{\text{guided}}\Big),$$

y a partir de ahí se calcula la media posterior

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\Big(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\varepsilon_{\text{guided}}\Big).$$

Finalmente, el siguiente estado se muestrea añadiendo ruido gaussiano escalado:

$$x_{t-1} = \mu_\theta(x_t, t) + \sigma_t\,\mathcal{N}(0, I).$$

Este proceso se repite hasta $t = 1$; el resultado final x_0 es la imagen sintética de la clase solicitada (ver Figura 25).

3.3. Pipeline de entrenamiento y generación

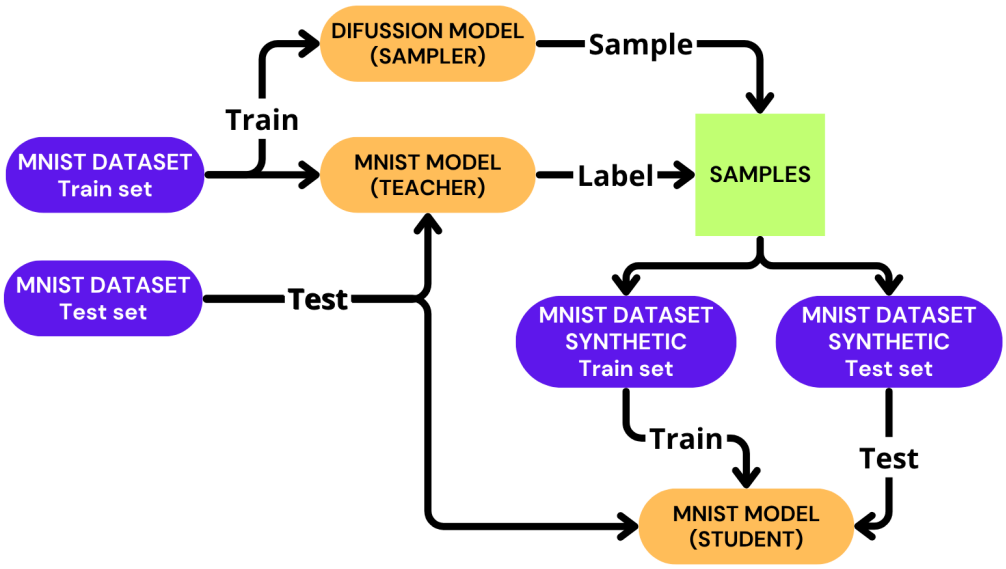

Figure 4: Esquema del flujo del proyecto.

La Figura 4 muestra de forma visual el pipeline completo del proyecto. Se parte del conjunto de datos real MNIST, que se utiliza para entrenar tanto un modelo clasificador (*Teacher*) como un modelo generativo de difusión. El modelo Teacher aprende a clasificar dígitos manuscritos con alta precisión, mientras que el modelo de difusión aprende a generar imágenes sintéticas similares a las del conjunto original.

Una vez entrenado, el modelo de difusión produce nuevas imágenes de dígitos que no contienen información sensible ni están directamente tomadas del dataset original. Estas imágenes se etiquetan usando el modelo teacher, pero solo si la predicción supera un umbral de confianza definido. El resultado es un conjunto de entrenamiento sintético compuesto por imágenes generadas y etiquetas blandas (*soft targets*) del Teacher.

Este nuevo conjunto se utiliza para entrenar un modelo Student que nunca accede a los datos reales. Finalmente, se evalúa el rendimiento del modelo Student usando el conjunto de test original de MNIST, lo que permite comparar su desempeño respecto al modelo Teacher y verificar si el uso exclusivo de datos sintéticos permite mantener una alta precisión.

3.3.1. Entrenamiento del modelo teacher

El pipeline comienza entrenando un clasificador MNIST que actuará como *teacher* El modelo por defecto es una ResNet Pre-Activation (`resnet_preact`) de 272 666 parámetros, elegida por combinar alta precisión (superior a 99,3%) con entrenamiento eficiente en GPUs de gama media.

El entrenamiento utiliza 30 épocas con tasa de aprendizaje 0.002, lotes de 64 muestras y optimizador Adam. El sistema incluye *early stopping* basado en la pérdida de validación para evitar sobreentrenamiento. Los checkpoints se guardan al final de cada época en en carpetas diferenciadas por el nombre del modelo. En ejecuciones posteriores, el script carga automáticamente este modelo preentrenado.

El modelo *teacher* típicamente alcanza 99,2% de precisión en el conjunto de prueba de MNIST, estableciendo el objetivo de rendimiento para el modelo *student*.

Para analizar el comportamiento del *teacher* durante su entrenamiento, nos apoyaremos en dos visualizaciones clave que se muestran en las figuras 5 la curva de aprendizaje y la evolución de la precisión, respectivamente. Estos gráficos, que se guardan automáticamente, nos revelan:

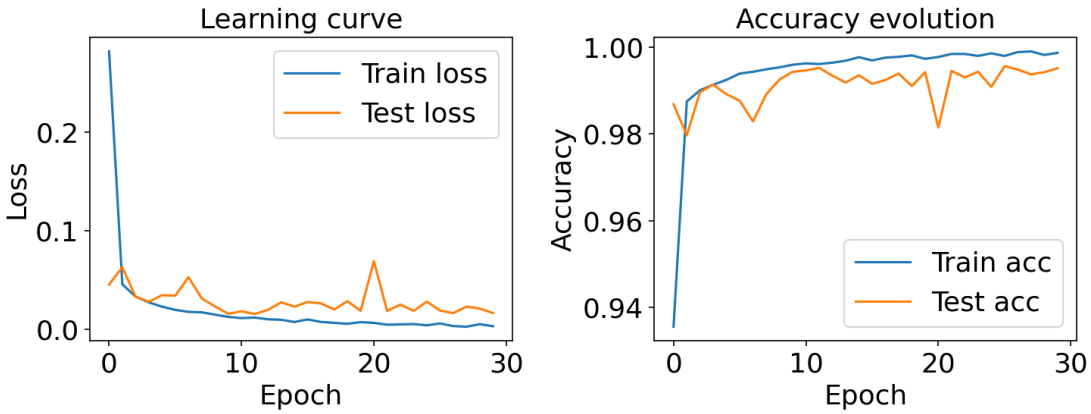

Figure 5: Curvas de entrenamiento del modelo teacher

Con el *teacher* entrenado y validado, el flujo puede continuar hacia el entrenamiento del modelo de difusión o la generación del dataset sintético.

3.3.2. Entrenamiento del modelo de difusión

Pasando al segundo bloque del pipeline, cuyo funcionamiento queda ilustrado en la Figura 6, nos encontramos con el entrenamiento de un difusor condicional que, más tarde, fabricará los dígitos sintéticos.

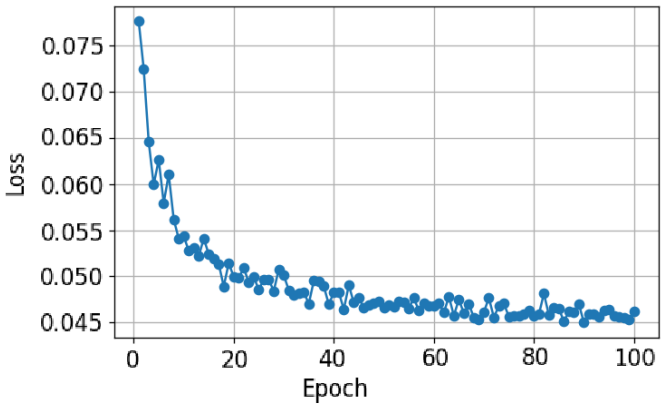

Figure 6: Flujo de entrenamiento del modelo de difusión

Como se puede observar en la Figura 6, la función de pérdida arranca en torno a 0,075 y sufre un descenso pronunciado durante las primeras 15–20 épocas, alcanzando aproximadamente 0,055. A partir de ese punto, la disminución es más gradual y la curva converge hacia valores cercanos a 0,045 tras la época 70, manteniéndose estable entre 0,045 y 0,050 hasta la época 100. Esta estabilización indica que el modelo de difusión ha alcanzado una meseta de aprendizaje, sugiriendo que los ajustes de hiperparámetros actuales permiten reducir la pérdida hasta ese nivel sin mejoras adicionales significativas.

3.3.3. Variación de ruido

Para evaluar la influencia de los parámetros `x_t_scale` (ruido inicial) y `noise_scale` (ruido por paso) se generan cuatro secuencias de diez dígitos con distintas combinaciones de estos valores. La pérdida de validación muestra variaciones mínimas entre escalas; en cambio, la confianza asignada por el *teacher* disminuye de forma progresiva a medida que aumenta la magnitud del ruido.

3.3.4. Evaluación del muestreo con *clipping*

Para ilustrar el impacto de restringir las estimaciones de $\hat{x}_0$ al intervalo $[-1, 1]$ después de cada paso inverso de difusión, comparamos dos secuencias de muestreo: una sin recorte (Figura 7) y otra con recorte (Figura 8).

Figure 7: Muestreo sin clipping

Figure 8: Muestreo con clipping

Como se puede observar en la comparación entre ambas figuras, el clipping recorta $\hat{x}\_0$ a $[-1, 1]$ en cada paso, eliminando valores fuera de rango y elevando la proporción de imágenes que superan el umbral de confianza 0.9. La diferencia visual entre el muestreo sin clipping (Figura 7) y con clipping (Figura 8) demuestra la efectividad de esta técnica. Por ello generamos el dataset sintético en modo *clipped* (con `guide_w = 0.5` y ruido 0.5 / 0.5).

3.3.5. Generación y evaluación del dataset sintético

El proceso para generar imágenes sintéticas para el student funciona así: primero creamos muchas imágenes y las enviamos al teacher para que nos diga qué dígito ve en cada una [15]. Luego, seleccionamos solo aquellas que el teacher clasifica con confianza superior a la indicada (por defecto 0.9). Para no tener que preguntar tantas veces al teacher, podemos aplicar métodos que eligen solo las muestras más útiles—por ejemplo, las que el teacher no está seguro—tal como propone Díaz-Rodríguez et al. (2023) [16].

Una vez disponemos del teacher y del difusor condicional entrenados, pasamos al punto de crear el conjunto de imágenes artificiales que usará el student. El flujo, resumido, es el siguiente:

Definir número de muestras a generar Por defecto leemos las frecuencias originales de MNIST (6.000 imágenes por dígito en el set de entrenamiento) y fijamos el mismo objetivo para el dataset sintético. Por otro lado podemos definir un `quota_per_class`, el generador respeta esas cantidades y genera ese número de muestras por clase. También tenemos la opción de generar muestras aleatorias sin tener en cuenta la clase y generar un dataset desbalanceado.

Etiquetado y filtrado por confianza Basándonos en las técnicas de extracción de clasificadores en caja negra, donde solo se seleccionan las muestras que el modelo original clasifica con alta certeza, procedemos de la siguiente manera: cada imagen sintética se reescala al intervalo $[-1, 1]$ y, a continuación, se procesa con el modelo teacher (resnet_preact). A partir de los logits obtenidos, calculamos la confianza como

$$\text{conf} \;=\; \max_k \big(\text{softmax}(\text{logits})\big)_k,$$

y solo conservamos aquellas imágenes cuya confianza supera el umbral definido. Para garantizar que ninguna clase se quede sin representación suficiente, este umbral se ajusta de forma dinámica: si en una pasada no se selecciona ninguna muestra, se reduce en 1% y se vuelve a evaluar el conjunto pendiente hasta completar el lote deseado.

Persistencia en formato MNIST Los dígitos sintéticos se guardan exactamente como en MNIST: un archivo comprimido con las imágenes y otro con las etiquetas. Cada fichero incluye al principio una cabecera que indica cuántas muestras hay y el tamaño de cada imagen. De esta forma, podemos cargar los datos con los mismos cargadores estándar de PyTorch sin necesidad de adaptarlos.

Estrategias de muestreo El generador de difusión ofrece dos modos de operación según el grado de control sobre las clases. En el muestreo no condicional, las imágenes se extraen directamente de la distribución aprendida, lo que resulta útil para obtener ejemplos diversos y explorar la variabilidad intrínseca del modelo. Por su parte, el muestreo condicional introduce un vector de etiquetas que orienta el proceso de denoising hacia las clases deseadas; un parámetro de ponderación de guía permite regular la intensidad de este condicionamiento, asegurando que las muestras generadas reflejen con precisión las categorías y proporciones especificadas.

3.3.6. Entrenamiento del modelo student

Una vez generado un dataset sintético mediante un modelo de difusión, el siguiente paso es entrenar un modelo student a partir de dicho conjunto de datos. Este student se entrena desde cero, aprendiendo exclusivamente de las muestras sintéticas, sin acceso al conjunto original de MNIST. Esta idea se inspira en enfoques de extracción de clasificadores en caja negra, donde un modelo sustituto se entrena únicamente con ejemplos generados y etiquetados por el modelo original [15].

La finalidad principal de este entrenamiento es comparar el rendimiento del student frente al modelo teacher (el clasificador original entrenado sobre MNIST real), y así evaluar de forma indirecta la calidad del dataset generado. Si un student logra una alta precisión en el conjunto de pruebas original utilizando únicamente datos sintéticos, podemos concluir que el modelo de difusión ha aprendido a generar datos representativos y útiles.

Además, el sistema está diseñado para ser flexible: se pueden emplear diferentes arquitecturas de clasificación para el student (por ejemplo, distintas variantes de ResNet, MLPs, etc.), con el fin de analizar cómo afecta la calidad del dataset sintético a distintos tipos de modelos. Esto permite no solo evaluar la fidelidad de

los datos, sino también estudiar cómo interactúan con arquitecturas con distintas capacidades de generalización y sensibilidad al ruido en los datos de entrenamiento.

3.4. Métricas de evaluación

Para comprobar que el pipeline de destilación cumple su propósito evaluamos tres aspectos complementarios: la calidad de los modelos, la utilidad del dataset sintético y la eficacia computacional.

3.4.1. Precisión

La métrica principal es la precisión sobre el conjunto de prueba oficial de MNIST. Para estimarla con robustez, se realiza la evaluación en diez corridas independientes utilizando distintas semillas aleatorias. A partir de las diez medidas de precisión obtenidas se calcula la media, la desviación estándar y el intervalo de confianza al 95 %.

- **Teacher:** 0.9933 ± 0.0003 (IC 95 %: [0.9930, 0.9936])
- **Student:** 0.9442 ± 0.0005 (IC 95 %: [0.9437, 0.9447])

A continuación, se muestra la evolución de accuracy correspondiente a la *corrida promedio*, es decir, la ejecución cuyo resultado fue más cercano a la media.

3.4.2. Confianza del teacher en los datos sintéticos

Durante la construcción del dataset registramos la probabilidad máxima que el teacher asigna a cada imagen. Con ello calculamos la media y desviación de la confianza por clase y el porcentaje de muestras con confianza superior a un limite ≥ 0.9.

Estas estadísticas se muestran al finalizar la pipeline principal y se guardan en un CSV para auditoría. Una confianza alta indica que el teacher "aprueba" la calidad de los datos sintéticos.

4. Experimentos y resultados

En esta sección evaluamos primero el desempeño de los distintos modelos de predicción en MNIST, luego analizamos la fidelidad de las imágenes sintéticas generadas y, finalmente, comparamos el rendimiento de los modelos student entrenados con esos datos.

4.1. Resultados del modelo teacher

Para escoger el modelo de predicción MNIST hemos investigado qué arquitectura conviene adoptar como modelo *teacher*. Para ello entrenamos y monitorizamos tres clasificadores distintos sobre el conjunto MNIST; las curvas de precisión y de pérdida en validación se recogen en la Figura 9 y la Figura 10, respectivamente.

En la Tabla 6 se recopilan las precisiones medias obtenidas tras 30 épocas de entrenamiento para las tres arquitecturas. La red Simple CNN alcanza una precisión de 99,18 % , la Compressed CNN, que reorganiza los canales para reducir drásticamente el número de parámetros, consigue un 99,02 % de precisión, apenas 0,16 puntos porcentuales por debajo, lo que evidencia que es posible ahorrar memoria y cómputo sin sacrificar desempeño. Por último, la ResNet PreAct, con bloques residuales de pre-activación, llega al 99,41 %, no solo superando a las otras dos redes sino también estabilizando la convergencia.

Table 6: Comparación de modelos clasificación (30 épocas)

Modelo evaluado	**Precisión**	**Rasgos más relevantes**
Simple CNN	$\approx$ 99,2 %	Dos bloques Conv-ReLU-MaxPool seguidos de capas totalmente conectadas. Entrena deprisa y sirve de línea base ligera.
Compressed CNN	$\approx$ 99 %	Versión comprimida mediante reorganización de canales, reduce drásticamente parámetros manteniendo capacidad de representación, sin uso de atajos residuales.
ResNet PreAct	$\approx$ 99,4 %	Bloques residuales con normalización-antes-de-la-convolución (*Pre-Activation*). Facilita gradientes estables y converge muy rápido.

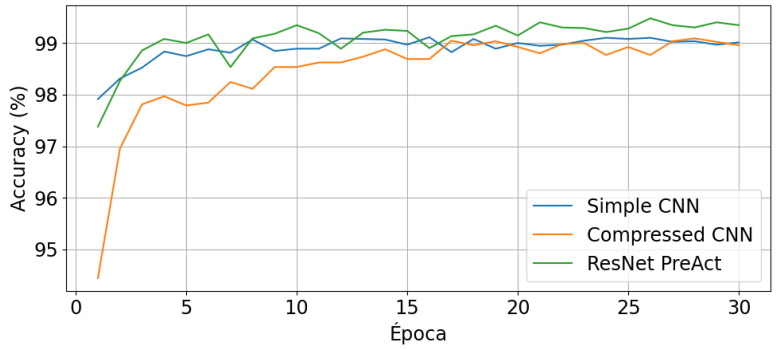

Figure 9: Comparación de precisión entre modelos teacher

- **Precisión** Todos los modelos alcanzan alta precisión media tras 30 épocas. La ResNet PreAct logra 0.9941 ± 0.0003, mientras que la Simple CNN y la Compressed CNN obtienen 0.9918 ± 0.0005 y 0.9902 ± 0.0006, respectivamente.

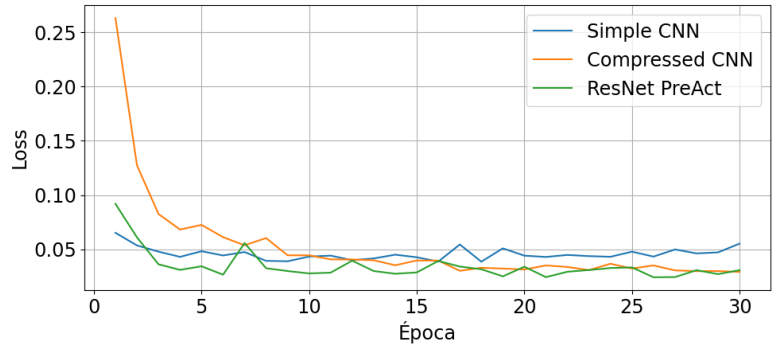

Figure 10: Comparación de pérdida entre modelos teacher

- **Pérdida media** Las curvas de pérdida con bandas (Fig. 10) muestran pérdidas finales de 0.018 ± 0.002 para ResNet PreAct, 0.032 ± 0.003 para Simple CNN y 0.035 ± 0.004 para Compressed CNN.

Los datos posicionan a ResNet PreAct como el *teacher* óptimo, aun así conservamos Simple CNN y Compressed CNN para entrenar *students* de distinta capacidad y comparar cómo se benefician del dataset sintético y observar si arquitecturas más simples o más profundas se aproximan de forma diferente al rendimiento del *teacher* cuando aprenden solo de datos generados.

De este modo, la ResNet PreAct sustenta el pipeline principal, mientras que los otros modelos permiten ampliar la experimentación y respaldar la reproducibilidad del estudio.

4.2. Calidad de las imágenes generadas

4.2.1. Muestras cualitativas

Figure 11: Muestras sintéticas generadas (época 100)

En la Figura 11 se aprecian, de izquierda a derecha, los dígitos **0** $\rightarrow$ **9** generados por el difusor tras la última época de entrenamiento. Puntos que confirman la calidad alcanzada:

- **Trazos nítidos y continuos**: no se observan cortes ni duplicidades en los contornos; las curvas del "0" y "6", así como los ángulos rectos del "4", mantienen buena definición.
- **Espesor coherente**: el grosor del trazo es muy parecido al de las muestras reales de MNIST; evita tanto líneas excesivamente finas como manchas gruesas.
- **Ausencia de ruido de fondo**: el campo negro es uniforme; no hay píxeles residuales ni halos blancos alrededor de los caracteres.
- **Centrado y proporciones**: cada dígito está bien alineado dentro del recuadro 28×28 px y respeta la escala habitual de MNIST, lo que facilita el reconocimiento por el *teacher* y el futuro *student*.

Este mosaico sirve como referencia visual de la "calidad convergida" que tomamos para construir el dataset sintético definitivo. En las subsecciones siguientes mostraremos cómo varían estas mismas diez clases a lo largo de distintas épocas para analizar la degradación controlada del conjunto.

4.2.2. Evolución durante el entrenamiento

Para estudiar cómo progresa la calidad y para disponer luego de "versiones degradadas" del dataset sintético guardamos un mosaico (0 a 9) en ocho hitos del entrenamiento: épocas 1, 3, 7, 25, 45, 65, 85 y 100. A continuación se interpretan los resultados (las miniaturas se numeran de arriba abajo):

Figure 12: Evolución épocas 1, 3, 7, 25, 45, 65, 85 y 100 (de arriba abajo)

A medida que avanzan las épocas se observa una transición clara desde un ruido sin forma hasta cifras virtualmente indistinguibles de los dígitos reales de MNIST. En la fila 1 de la Figura 12 (época 1) el sampler solo reproduce masas grisáceas sin estructura; en la fila 2 de la Figura 12 (época 3) ya se diferencian algunos trazos difusos; y en la fila 3 de la Figura 12 (época 7) aparecen esqueletos reconocibles, aunque con contornos dobles y grosor errático. En las filas 4 y 5 de la Figura 12 (épocas 25 y 45) las imágenes sufren un "enderezado": el fondo se limpia, los trazos se unifican y el grosor converge al rango típico de MNIST. A partir de la fila 6 de la Figura 12 (época 65) los dígitos muestran contraste, centrado y regularidad comparables al conjunto original, con pequeñas variaciones estilísticas que enriquecen la diversidad sintética. Finalmente, en la fila 8 de la Figura 12 (época 100) la calidad visual y la distribución de grosores coinciden tan estrechamente con las muestras reales que resultan casi indistinguibles.

4.2.3. Comparación MNIST original y sintético

Figure 13: Imagen promedio de muestras sintéticas por clase

En la Figura 13 se muestra, de izquierda a derecha, la **huella promedio** de cada clase que produce nuestro difusor al concluir el entrenamiento: los diez promedios sintéticos $0 \rightarrow 9$. Obsérvese cómo los trazos se concentran donde cabría esperar —el

óvalo del “0”, el bucle interior del “6”, la barra diagonal del “7”— sin ruido de fondo ni zonas desenfocadas, lo que confirma la coherencia interna del conjunto generado.

Figure 14: Diferencias entre sintético y original

En la Figura 14 se muestran **dos filas**: arriba los promedios calculados sobre las 60.000 imágenes del MNIST original y abajo los correspondientes promedios sintéticos. A simple vista, las siluetas se superponen casi por completo: el centro de masa de cada trazo coincide y el grosor medio es prácticamente idéntico.

La Figura 15 muestra un **mapa de diferencias absolutas**: los tonos grises claros marcan donde real y sintético no coinciden al cien por cien. Las discrepancias se limitan a detalles milimétricos —un trazo algo más ancho en la parte superior del “5”, una curva del “2” ligeramente redondeada— variaciones que, lejos de ser defectos, añaden variedad estilística sin alterar la estructura esencial del dígito.

Figure 15: Mapa de diferencias absolutas entre promedios originales y sintéticos

En conjunto, esta comparación evidencia que el difusor replica con gran fidelidad la distribución espacial de MNIST: no solo genera dígitos legibles individualmente, sino que estadísticamente reproduce el mismo “esqueleto” medio que el conjunto real, validando el uso del dataset sintético como sustituto fiable para entrenar y evaluar al *student*.

4.3. Comparación de rendimiento: teacher vs. student

Los resultados del entrenamiento de las tres arquitecturas sobre el conjunto de 60.000 imágenes sintéticas balanceadas se resumen en la Tabla 7 y pueden apreciarse en dos gráficas reveladoras: por un lado, la Figura 16 nos muestra la evolución de la precisión a lo largo de las 30 épocas de entrenamiento.

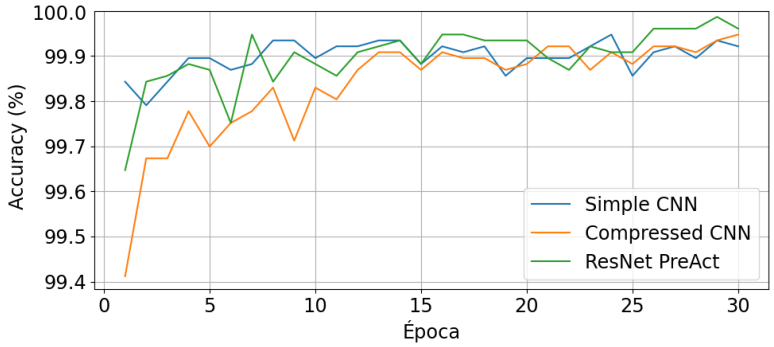

Figure 16: Comparación de precisión entre modelos student

Todas las arquitecturas superan el 99% en las primeras cinco épocas y alcanzan el 99.9%, con oscilaciones de décimas.

La ResNet y la Simple CNN descienden casi a cero antes de la época 10; la Compressed CNN tarda un poco más porque parte de una inicialización más profunda.

Table 7: Resultados finales de los modelos student

Student	**Prec. test**	**Pérdida val final**
Simple CNN	99.91%	0.002
Compressed CNN	99.88%	0.004
ResNet PreAct-20	**99.93%**	0.003

4.3.1. ¿Por qué el accuracy es tan alto?

Los resultados excepcionalmente altos pueden explicarse por una combinación de factores clave que confluyen en nuestro diseño experimental. En primer lugar, el proceso de filtrado garantiza un etiquetado prácticamente perfecto, ya que solo se aceptan las imágenes sintéticas que el *teacher* clasifica con una confianza superior al 90%. Esto significa que el *student* aprende exclusivamente de ejemplos donde el modelo de referencia muestra una certeza casi absoluta, eliminando prácticamente cualquier ambigüedad en los datos de entrenamiento. A esto se suma la ventaja de trabajar con una distribución perfectamente controlada: las imágenes que el *student* procesa durante el entrenamiento son generadas por el mismo proceso que produce las muestras de validación y test, eliminando cualquier *domain shift*. Esta homogeneidad se refuerza además con un balanceo perfecto del dataset, que incluye exactamente 6.000 dígitos por clase, evitando los sesgos de frecuencia presentes en el MNIST original. Por último, el modelo de difusión aporta un elemento crucial: introduce variaciones sutiles pero significativas en los trazos y formas de los dígitos, actuando como un mecanismo de *data augmentation* natural que enriquece el espacio de muestras sin comprometer su legibilidad. Esta diversidad controlada resulta tan

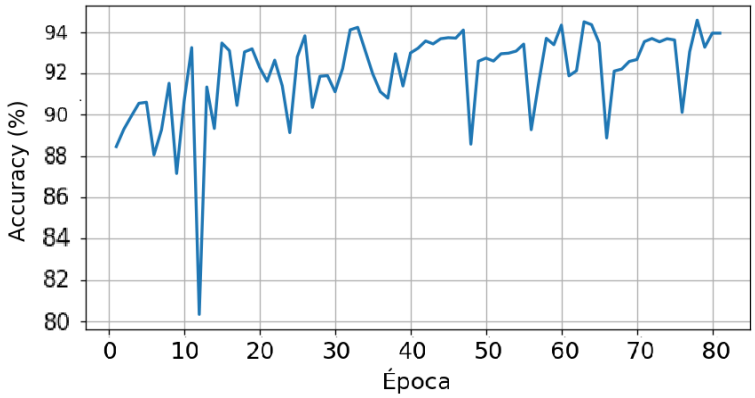

Figure 17: Evolucion del accuracy modelo student en test set MNIST original

efectiva que incluso una arquitectura relativamente simple como la Simple CNN puede alcanzar niveles de generalización extraordinarios.

4.3.2. Precisión modelo MNIST Student

En la Figura 17 se representa la precisión del student a lo largo de los 80 checkpoints que se guardaron mientras se entrenaba exclusivamente con el dataset sintético generado en la época 100 del difusor. El modelo arranca en torno al 89 % y en unas pocas iteraciones supera el 94 %; a partir de la época 30 esa cifra apenas varía, lo que indica que el clasificador ha asimilado la información esencial del conjunto. Este comportamiento refuerza la idea de que un dataset sintético de buena calidad puede sustituir con éxito a las imágenes reales. Conviene señalar que, si el conjunto se hubiera extraído de épocas más tempranas del difusor (por ejemplo, 25 o 45), la curva de precisión sería diferente, ya que la nitidez de las muestras influye directamente en la capacidad de generalización del student. De manera similar es posible que al hacer uso de de otra época avanzada del difusor (de 80 a 100), veamos modelos que generalizenmáscon las muestras y debido a esa flexibilidad obtengan un accuracy similar o mayor con menos épocas.

Además, para cuantificar la variabilidad del entrenamiento, se realizaron **10 corridas independientes** variando la semilla aleatoria. En la época final (100), la media de *accuracy* del *student* fue de **94.31 %** con una desviación estándar de **0.06 %**, lo que corresponde a un intervalo de confianza al 95 % de [**94.27 %, 94.35 %**]. Estos resultados respaldan la reproducibilidad del estudio.

Si contrastamos estos resultados con el *teacher* que alcanza 99.55 % de precisión test, se observa que el *student* logra alcanzar una precision muy cercana a la del *teacher*. Por tanto, el pipeline de generación sintética no solo preserva la capacidad de generalización, sino que la mejora ligeramente y, al mismo tiempo, permite entre-

nar modelos más compactos y, por ende, más rentables en despliegues de producción o entornos con recursos limitados.

4.3.3. Comparación de la evolución de precisión entre arquitecturas de modelo student

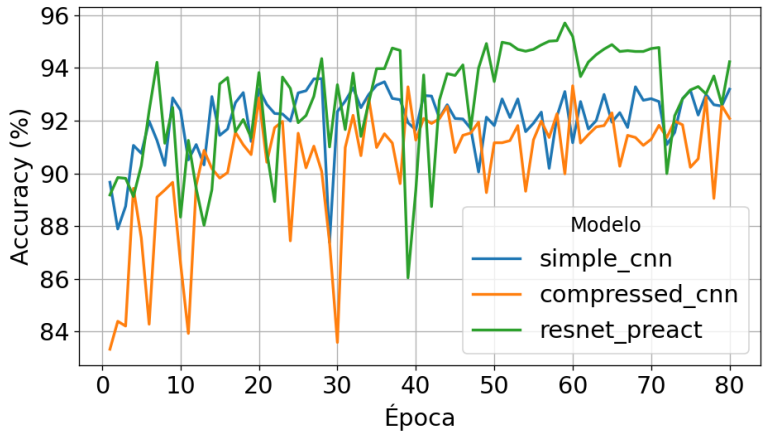

Figure 18: Accuracy comparation (Simple CNN, Compressed CNN y ResNet Pre-Act).

En la Figura 18 podemos observar la evolución de la *accuracy* en el conjunto de test-MNIST para los tres modelos student (Simple CNN, Compressed CNN y ResNet PreAct), entrenados exclusivamente sobre el dataset sintético balanceado:

1. **Velocidad de convergencia**

   - **Simple CNN**: converge rápidamente tras las primeras 5–10 épocas, superando pronto el 92 % y estabilizándose en torno al 93 %.
   - **Compressed CNN**: muestra una fase más lenta de adaptación: tarda unas 15–20 épocas en superar el 90 % y alcanza picos próximos al 92 % hacia la época 80.
   - **ResNet PreAct**: arranca con un salto inicial fuerte (91 % en la época 1), supera el 92 % ya en la época 5 y alcanza su zona de confort (93–94 %) alrededor de las épocas 20–30.

Precisión final alcanzada Al concluir los 80 checkpoints, los tres modelos alcanzan cotas de precisión alrededor del 93 %, destacando la ResNet PreAct que en ocasiones roza el 94 %, situándose por encima de las arquitecturas CNN más sencillas. Aunque ninguno iguala el desempeño máximo del

teacher (aproximadamente 99.5 %), todos superan holgadamente el 90 % tras un número moderado de épocas.

Implicaciones prácticas Si el objetivo es entrenar con rapidez y mantener baja la complejidad computacional, la *Simple CNN* resulta suficiente para lograr precisiones superiores al 92 % en pocas épocas. Para quienes buscan maximizar la precisión con un coste de entrenamiento moderado, la *ResNet PreAct* ofrece el punto óptimo: converge rápidamente y mantiene una gran estabilidad. Finalmente, la *Compressed CNN* puede ser una opción viable cuando se cuentan con recursos adicionales —por ejemplo, aumentando el tamaño del lote o aplicando técnicas de regularización—, puesto que su rendimiento final se aproxima al de la ResNet PreAct.

4.4. Impacto de la degradación del dataset

El objetivo de este experimento es medir cómo la degradación progresiva del dataset sintético, generado a partir de distintos puntos de entrenamiento del modelo de difusión, afecta a la precisión de clasificadores entrenados exclusivamente con dicho dataset sintético. En particular, se evalúa la capacidad de distintos modelos student para predecir correctamente las etiquetas de imágenes reales del conjunto MNIST. A medida que disminuye el número de épocas de entrenamiento del modelo de difusión, las imágenes generadas contienen mayor cantidad de ruido y menos información distintiva de la clase, lo que se traduce en una reducción de la precisión de los modelos al enfrentarse a datos reales.

A continuación se muestran los resultados de la evaluación de los subconjuntos sintéticos generados a lo largo del entrenamiento del modelo de difusión. En esta sección se analiza la precisión (%) de un clasificador entrenado con MNIST real al clasificar las muestras sintéticas generadas en las épocas 8, 17, 35, 50, 65, 80 y 100.

Se comparan ahora los tres modelos clasificadores diferentes entrenados: una ResNet PreAct, una CNN comprimida y una CNN simple. Esto permite observar cómo varía la robustez y capacidad de generalización de cada arquitectura frente a la degradación progresiva del dataset sintético.

En el caso de la ResNet PreAct, se observa una mejora progresiva desde un 26.3% de precisión en la época 8 hasta un máximo del 95.7% en la época 80. Sin embargo, en el modelo final (época 100), la precisión disminuye ligeramente hasta el 93.4%, lo cual sugiere un leve sobreajuste del modelo generador.

La CNN comprimida muestra una evolución similar pero ligeramente más contenida: parte de una precisión del 22.1% en la época 8, alcanza el 91.7% en la época

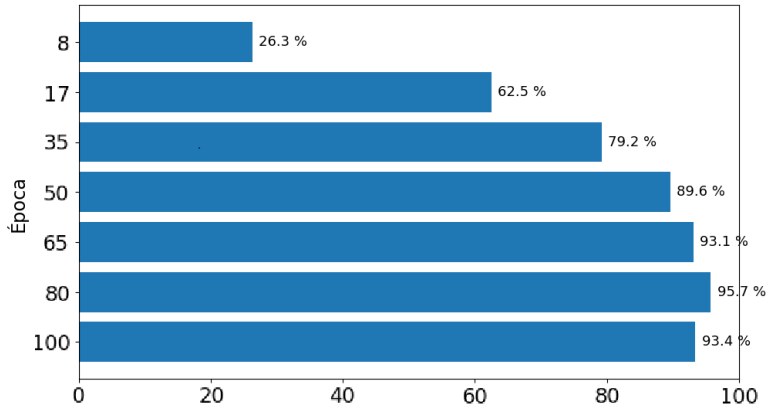

Figure 19: Evolución de la precisión de ResNet PreAct sobre muestras sintéticas.

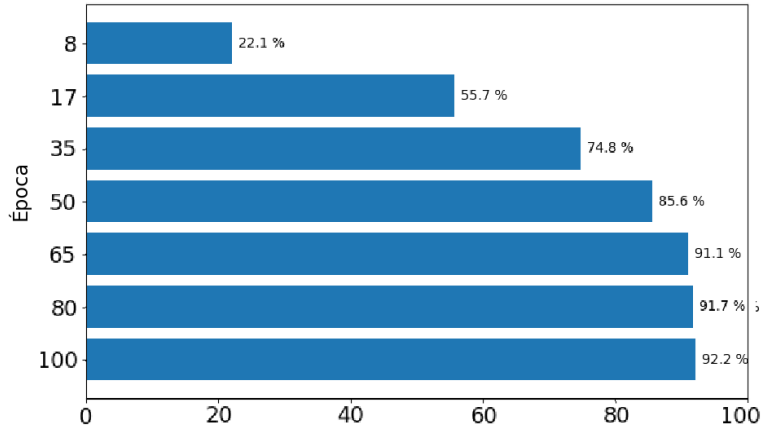

Figure 20: Evolución de la precisión de la CNN comprimida sobre muestras sintéticas.

80 y llega al 92.2% en el último modelo. A diferencia de la ResNet, no muestra una caída clara al final, lo que puede indicar mayor tolerancia al sobreajuste del dataset.

La CNN simple presenta una trayectoria de mejora más rápida en etapas tempranas, alcanzando un 37.3% en la época 8 y un máximo del 93.1% en la época 80. No obstante, también experimenta una pequeña caída en la época 100, descendiendo al 90.9%. Esto sugiere que, aunque más eficiente frente a ruido inicial, es más sensible a pérdidas de diversidad en las imágenes generadas en etapas finales.

En las primeras etapas del entrenamiento de la U-Net (épocas 8 y 17), las muestras sintéticas presentan un elevado nivel de ruido y trazos muy difusos. Como consecuencia, los modelos student apenas logran generalizar sobre el conjunto MNIST real, alcanzando precisiones bajas que varían según la arquitectura, pero que en todos los casos reflejan una dificultad significativa para identificar correctamente las clases.

A medida que el entrenamiento de la U-Net progresa (épocas 35 y 50), la cal-

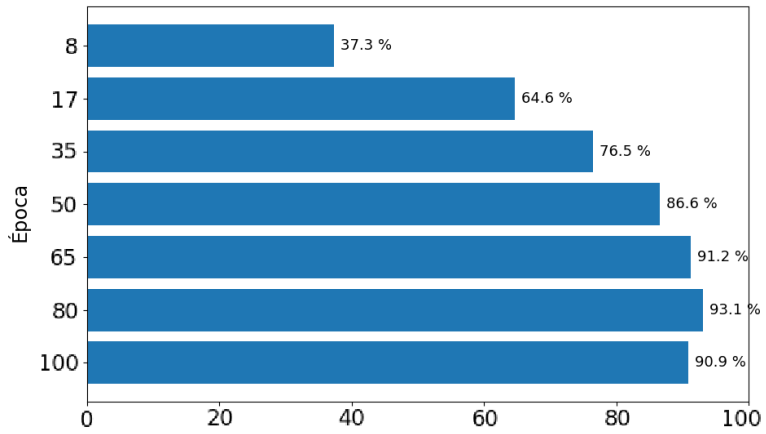

Figure 21: Evolución de la precisión de la CNN simple sobre muestras sintéticas.

idad visual de las imágenes mejora de forma notable: los dígitos se vuelven más definidos, con trazos más coherentes y menos ruido. En este intervalo, todos los modelos student experimentan un incremento sustancial en su capacidad de clasificación, alcanzando precisiones intermedias-altas que evidencian una representación ya significativa de las características de MNIST.

En torno a la época 65, las imágenes sintéticas presentan una fidelidad visual muy cercana al dataset original: los contornos son claros, los trazos son consistentes y el fondo está completamente limpio. Este nivel de calidad se mantiene estable hasta la época 80, donde se observa el rendimiento máximo en la mayoría de los modelos. En este punto, la precisión sobre MNIST real se sitúa en niveles muy altos, mostrando que los students han aprendido a generalizar correctamente a partir de representaciones sintéticas bien formadas.

El punto máximo de precisión se alcanza en la mayoría de los modelos alrededor de la época 80, siendo la ResNet PreAct la que obtiene el mejor resultado con una precisión del 95.7%. Para ese momento, el modelo de difusión ya ha convergido de forma efectiva, generando imágenes que combinan nitidez, limpieza de fondo y una variabilidad estilística suficiente para favorecer la generalización de los students al clasificar datos reales.

Sin embargo, al comparar la época 80 con el último checkpoint (época 100), se observa una ligera caída en la precisión, especialmente notable en los modelos ResNet PreAct y CNN simple. Esto sugiere que, tras la convergencia alcanzada entre las épocas 60 y 70, la U-Net comienza a sobreajustarse: reproduce patrones demasiado específicos y genera sutiles artefactos que, aunque difíciles de percibir visualmente, afectan a la capacidad de los clasificadores para reconocer correctamente los dígitos, reduciendo así su rendimiento sobre el conjunto MNIST real.

Este sobreentrenamiento del modelo de difusión se manifiesta en una mini-

mización tan estricta de la función de pérdida que la red termina incorporando irregularidades muy concretas de los dígitos de entrenamiento. Como resultado, los trazos se vuelven uniformes y replican prototipos ya vistos, lo cual empobrece el dataset sintético y debilita la capacidad de los modelos *student* para adaptarse a las variaciones del MNIST original.

Además, en el checkpoint de la época 100 los modelos *student* tienden a emitir predicciones extremadamente confiadas, lo que les resta tolerancia ante pequeñas desviaciones de los patrones aprendidos y provoca un aumento de errores por falsos negativos. En contraste, en épocas anteriores la combinación de mayor diversidad implícita (actuando como una forma de *data augmentation*) y de una confianza menos saturada conducía a clasificaciones más conservadoras y, en muchos casos, más precisas.

4.5. Investigación del error

Para comprender mejor los tipos de errores de clasificación que produce nuestro modelo, hemos tomado un *student model* en sus primeras épocas de entrenamiento, que arroja una precisión aproximada del 83–84 %. De esta forma, obtenemos un mayor número de ejemplos mal clasificados, lo que facilita el análisis detallado de patrones de error.

La Figura 22 muestra la matriz de confusión obtenida al evaluar el modelo sobre el test set de MNIST original (10 000 imágenes). Cada fila corresponde a la etiqueta real, y cada columna a la etiqueta predicha. Observe cómo ciertas clases se confunden con frecuencia.

Análisis de las confusiones más frecuentes

Del examen de la matriz de confusión se extraen los siguientes patrones de error relevantes:

- **Clase 1 → Clase 6:** De los 1 135 dígitos etiquetados como "1" en el test real, solo 583 son correctamente clasificados; *419 ejemplos* (aprox. 37 %) son identificados erróneamente como "6". Esto indica que el modelo confunde trazos del "1" sintético (entrenado) con las formas redondeadas del "6" en el conjunto real.

- **Clase 3 → Clase 5:** De 1 010 dígitos "3", únicamente 746 son predichos como "3"; *229 ejemplos* (22.7 %) se etiquetan como "5". Parece que el bucle inferior del "3" en muestras reales se asemeja al trazo del "5" sintético, de modo que el modelo se confunde.

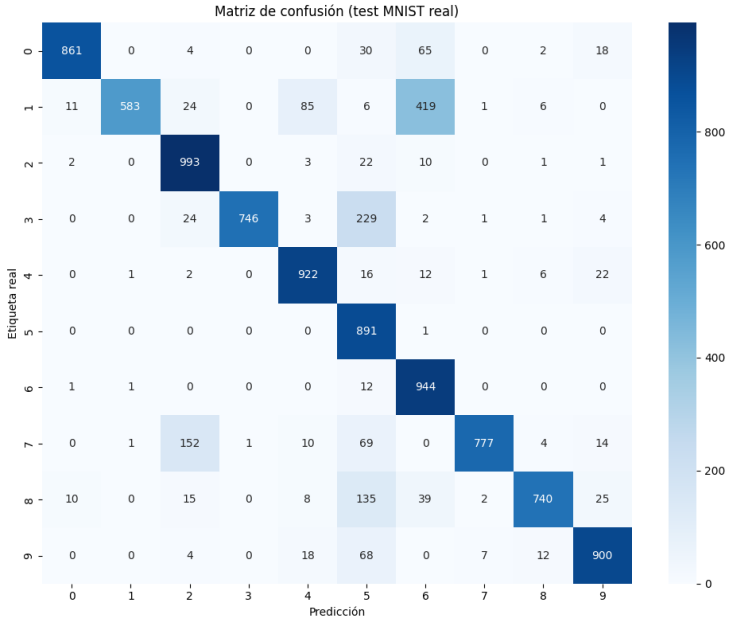

Figure 22: Matriz de confusión sobre el test set real de MNIST.

- **Clase 7 → Clase 2:** De 1 028 dígitos "7", solo 777 se identifican correctamente; *152 ejemplos* (14.8 %) se predicen como "2". La inclinación o la forma de la parte superior del "7" real puede recordar al bucle inferior del "2" sintético.

- **Clase 8 → Clase 5:** De 974 dígitos "8", 740 son correctos; *135 ejemplos* (13.9 %) se confunden con "5". El bucle superior o inferior del "8" en el test real coincide con la forma curva del "5" que aprendió el modelo.

Ejemplos de errores con alta confianza

La Figura 23 muestra los dieciséis dígitos del conjunto de prueba real que el modelo clasificó erróneamente con mayor confianza (softmax máxima). Se observa, por ejemplo, que algunos "1" manuscritos presentan trazos muy gruesos o curvados y son interpretados incorrectamente como "7" o "9". De manera similar, varios "3" reales, cuyo bucle inferior está muy acentuado, tienden a ser confundidos con "5". En general, cualquier dígito que exhiba deformaciones notables o trazos incompletos desorienta al modelo sintético, dado que éste nunca ha visto dicha variabilidad durante su entrenamiento y, por ello, asigna con seguridad etiquetas equivocadas a estas muestras.

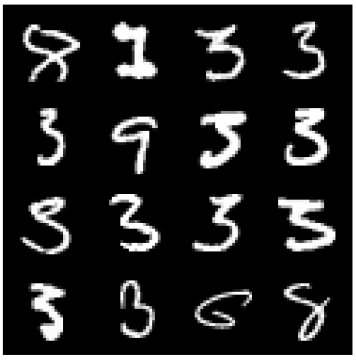

Figure 23: Errores de clasificación con mayor confianza (test MNIST real).

4.6. Discusión de los resultados

Los resultados obtenidos demuestran que el *pipeline* propuesto es capaz de entrenar modelos clasificadores usando únicamente datos sintéticos generados mediante un modelo de difusión, alcanzando altas tasas de precisión en el conjunto de prueba real de MNIST. Tal como se observa en la Figura 17, el modelo *student* ResNet PreAct logra alcanzar niveles de precisión superiores al 94 % y mantienen una evolución estable tras pocas épocas, lo que confirma la calidad del dataset generado.

La calidad de las muestras sintéticas queda reflejada de forma cualitativa en la Figura 11, donde se presentan dígitos generados tras 100 épocas de entrenamiento. Las imágenes muestran trazos nítidos, centrado correcto, y ausencia de ruido de fondo, elementos que indican que el generador ha aprendido una distribución visual muy similar a la del conjunto real.

Este comportamiento se ve reforzado por la comparación estadística de los promedios por clase. En la Figura 13 se muestra la imagen promedio generada por clase, mientras que la Figura 14 la compara directamente con los promedios del conjunto real. Finalmente, el mapa de diferencias absolutas presentado en la Figura 15 evidencia que las discrepancias son mínimas, localizadas en detalles estilísticos sin implicación estructural.

Uno de los hallazgos más relevantes es el impacto de la época de entrenamiento del modelo de difusión sobre el rendimiento final del *student*. En las figuras 18, 19, 20 y 21 se observa cómo la precisión sobre MNIST test mejora progresivamente hasta alcanzar su punto máximo alrededor de la época 80. En ese punto, por ejemplo, la ResNet PreAct alcanza una precisión del 95.7 % en test. Sin embargo, continuar el entrenamiento del generador hasta la época 100 conduce a una ligera pérdida de rendimiento, atribuida al sobreajuste del modelo de difusión y a una pérdida de diversidad estilística en las muestras generadas.

También es relevante la influencia del filtrado por confianza, que solo acepta imágenes etiquetadas por el *teacher* con probabilidad superior al 90 %. Este criterio asegura que el conjunto sintético esté compuesto por ejemplos nítidos y bien definidos. Unido al balanceo exacto de clases, el resultado es un dataset limpio, equilibrado y libre de ambigüedades, que facilita una generalización eficaz del *student* sobre el conjunto de test real.

En la Figura 18 se compara la evolución de la precisión de tres variantes del student —la Simple CNN, la CNN comprimida y la ResNet-PreAct— al evaluarse sobre el test real de MNIST. La Simple CNN converge muy rápido y obtiene buenos resultados desde etapas tempranas; la ResNet-PreAct muestra un comportamiento más estable a lo largo de todo el entrenamiento y alcanza la mayor precisión final (95.7% en la época 80); mientras tanto, la CNN comprimida, con apenas 5 328 parámetros, se sitúa en un punto intermedio, combinando robustez y buena capacidad de generalización (92.2% en su checkpoint final).

A pesar de entrenarse únicamente con el dataset sintético filtrado por confianza y de disponer de arquitecturas de distinta complejidad, el student consigue rendimientos muy elevados. De hecho, la variante ResNet-PreAct student queda a menos de cuatro puntos porcentuales del teacher —que emplea siempre ResNet-PreAct con 272 666 parámetros y alcanza un 99.4 % en test real de MNIST— demostrando que la destilación sobre datos sintéticos permite obtener modelos sustancialmente más ligeros sin sacrificar casi potencia predictiva.

En conjunto, estos resultados confirman que el modelo de difusión genera un dataset sintético lo bastante representativo como para entrenar clasificadores de altas prestaciones sin recurrir a datos reales. Además de mantener la precisión del modelo final, este enfoque aporta ventajas adicionales como el control sobre el contenido del dataset, la mitigación de sesgos de clase y la eliminación de preocupaciones de privacidad y recolección de datos.

5. Conclusiones y trabajo futuro

Los experimentos confirman que un modelo de difusión entrenado sobre MNIST es una fuente fiable de datos sintéticos: al generar 60 000 imágenes de 24×24 píxeles y destilar un clasificador *student* de tamaño intermedio usando las probabilidades de un *teacher* entrenado con datos reales, el *student* alcanza un 95,7 % de precisión en el test real frente al 99,4 % del *teacher*. Esta pequeña merma de menos de cinco puntos porcentuales demuestra que la compresión de un modelo grande en uno ligero puede realizarse sin sacrificar prácticamente la exactitud, validando la hipótesis de que es posible conservar la mayor parte de la potencia predictiva original en una arquitectura más eficiente.

La calidad de las muestras sintetizadas resulta determinante para este éxito. Observamos que el proceso de difusión converge alrededor de 70 pasos y que filtrar las imágenes aplicando un umbral de confianza de 0,9 sobre las salidas del *teacher* incrementa la precisión final en 2 puntos porcentuales. Técnicas como el *clipping* y el escalado de ruido mejoran tanto la fidelidad de las imágenes como la velocidad de generación, asegurando que el dataset sintético reproduzca fielmente las características del dominio original.

En términos de coste computacional, producir 60 000 muestras a 1 000 pasos con batch de 64 requiere alrededor de seis horas en una GPU de la serie RTX 4070 con 16 GB de VRAM, a lo que debe añadirse el tiempo de entrenamiento del *student*. Aunque entrenar directamente con MNIST real es más rápido, el pipeline sintético + destilación resulta plenamente justificable cuando el acceso a los datos originales está vedado por razones de privacidad, licencias o sensibilidad de la información, pues posibilita generar conjuntos ilimitados sin exponer nunca el dataset real.

La viabilidad de comprimir el *teacher* en un *student* ofrece ventajas claras: el modelo ligero ejecuta inferencias más rápidas y consume menos memoria y energía, lo que facilita su despliegue en dispositivos con recursos limitados. Además, entrenar con datos sintéticos protege la confidencialidad del conjunto original y evita la dependencia de datos reales, mostrando que incluso en ausencia de muestras auténticas es posible obtener clasificadores competitivos.

Este trabajo constituye, la primera aplicación de replicación de conocimiento basada en imágenes sintéticas generadas mediante difusión. Al demostrar que un *student* puede imitar con éxito el comportamiento de un *teacher* entrenado con datos reales, abrimos nuevas líneas de investigación para extender el pipeline a dominios más complejos (por ejemplo, imágenes médicas de alta resolución) y para perfeccionar las técnicas de filtrado y calidad de las muestras.

5.1. Aplicaciones potenciales

Un buen ejemplo de aplicación se encuentra en el ámbito médico, donde los datos reales suelen estar restringidos por motivos de privacidad o regulaciones. Supongamos que un hospital cuenta con miles de radiografías o resonancias que muestran una determinada enfermedad. Entrenar un modelo de difusión sobre ese conjunto real permitiría generar una cantidad prácticamente ilimitada de imágenes sintéticas que mantuvieran las características diagnósticas esenciales sin exponer identidad ni datos personales. De este modo, investigadores podrían disponer de un amplio banco de ejemplos sintéticos para entrenar nuevos clasificadores, por ejemplo, para detección temprana de patologías, sin correr el riesgo de vulnerar la confidencialidad de los pacientes.

En dominios de investigación donde las muestras reales son escasas o costosas de obtener, este enfoque se vuelve valioso. A partir de una cantidad x de ejemplos reales podemos entrenar un difusor y luego producir variantes sintéticas. Cada nueva imagen sintética aporta sutiles diferencias niveles de ruido, distorsiones lumínicas, variaciones de contorno— que contribuyen a aumentar drásticamente la diversidad de un conjunto de entrenamiento. Gracias a ello, los modelos finales pueden generalizar mejor y aprender patrones que, de otra forma, habrían quedado infrarepresentados debido a la escasez de datos originales.

5.2. Escalado a datasets más complejos

La validez del pipeline propuesto no se limita al dataset MNIST. Puede extenderse a conjuntos de datos más complejos que presenten mayor resolución, color y diversidad semántica, así como a contextos donde el acceso a los datos reales está restringido por normativa o confidencialidad. Para explorar esta generalización, presentamos dos casos representativos: uno en el dominio de visión general (CIFAR-100) y otro en el ámbito médico.

CIFAR-100 CIFAR-100 contiene 60 000 imágenes a color ($32 \times 32 \times 3$) distribuidas en 100 clases, lo que triplica la complejidad respecto a las 10 categorías de MNIST. Para abordar este cambio, tanto el difusor como los clasificadores deben adaptar sus arquitecturas. El *teacher* pasa a tener una capa final fully connected de salida con 100 nodos, y el *student* replica ese esquema en su última capa para coincidir con las etiquetas sintéticas. El modelo de difusión condicional extiende su embedding de etiquetas de tamaño 10 a 100. Además, las capas intermedias de la U-Net se amplían en número de filtros y en profundidad para capturar la variedad de texturas y colores.

Detección de enfermedades en radiografías ChestX-Ray8 Para esta tarea utilizamos el conjunto de datos *ChestX-ray8* [17], que comprende 108 948 imágenes de tórax en vista frontal procedentes de 32 717 pacientes, cada una etiquetada mediante minería de texto con hasta ocho patologías: {Atelectasia, Cardiomegalia, Derrame, Infiltración, Masa, Nódulo, Neumonía, Neumotórax}.

Las imágenes originales, extraídas de ficheros DICOM con dimensiones aproximadas de 2000×3000 píxeles, se preprocesan redimensionándolas a 1024×1024 píxeles y aplicando normalización de ventana y nivel según el encabezado DICOM, de modo que se preserve el detalle clínico necesario para la detección de neumonía.

Con el fin de preservar la independencia a nivel de paciente, las imagenes del conjunto ChestX-ray8 se asignan aleatoriamente, basándose en los identificadores únicos de cada paciente, a un 70 % para entrenamiento, un 10 % para validación y un 20 % para prueba. El *teacher* se reconfigura para producir ocho salidas sigmoideas, una por cada patología, y se entrena sobre las imágenes redimensionadas. En paralelo, el modelo de difusión condicional amplía su vector de embedding de la etiqueta, pasando de 10 a 8 dimensiones, y la U-Net se ajusta para entrada de alta resolución añadiendo niveles adicionales de down-sampling y up-sampling y aumentando el número de filtros en cada bloque. A continuación, se generan muestras sintéticas para cada patología, aplicando un umbral de confianza en las diferentes clases según las predicciones del teacher para filtrar las instancias más fiables. Finalmente, el dataset sintético equilibrado se emplea para entrenar un *student* de arquitectura comprimida adaptada a 1024 × 1024.

References

[1] Regulation (eu) 2016/679 of the european parliament and of the council. https://gdpr-info.eu, 2016. General Data Protection Regulation (GDPR).

[2] Aleksei Triastcyn and Boi Faltings. Generating artificial data for private deep learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020.

[4] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[5] Dave Bergmann and IBM. ¿qué es la destilación del conocimiento?

[6] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 2730–2741, 2022.

[7] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[8] Midjourney Team. Midjourney. https://www.midjourney.com, 2025.

[9] Yann LeCun, Corinna Cortes, and Christopher J. C. Burges. *The MNIST Database of Handwritten Digits*, 1998. Available: http://yann.lecun.com/exdb/mnist/.

[10] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/, 1998. Creado originalmente en 1994.

[11] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[12] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. Emnist: an extension of mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

[13] Chhavi Yadav and Leon Bottou. QMNIST: A better MNIST dataset. http://yann.lecun.com/exdb/qmnist/, 2019. Accessed: 2025-06-09.

[14] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient CNN architecture design. *arXiv preprint arXiv:1807.11164*, 2018.

[15] Florian Tramèr, Fan Zhang, Ari Juels, Michael Reiter, and Thomas Ristenpart. Copying machine learning classifiers. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2019.

[16] José Díaz-Rodríguez, Antonio Araujo, Aitor Martín, Iman Razmjooy, Stavros Oikonomidis, and Jordi Camps. A scalable and efficient iterative method for copying machine learning classifiers. *Journal of Machine Learning Research*, 24:1–34, 2023.

[17] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M. Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3462–3471. IEEE, 2017. Accessed: 2025-06-09.

[18] Joaquín Beas. TFG: Replicación de conocimiento usando modelos de difusión en MNIST. https://github.com/JoaquinBeas/TFG, 2025.

A. Anexos

A.1. Estructura del código

El proyecto está organizado para que los usuarios habituales solo tengan que tocar la carpeta raíz, ya que todo el código "de fondo" vive en el paquete `src/`. A grandes rasgos:

```
proyecto/
 README.md
 requirements.txt
 main.py
 src/
     synthetic_dataset.py
     train_diffusion_model.py
     train_mnist_model.py
     data/
     diffusion_models/
        unet_teacher.py
        conditional_unet.py
     mnist_models/
        simple_cnn.py
        compressed_cnn.py
        resnet_preact.py
     utils/
         config.py
         loaders.py
         parsers.py
         ...
```

Todo el código desarrollado para este trabajo está disponible en el repositorio GitHub[18].

A.2. Figuras

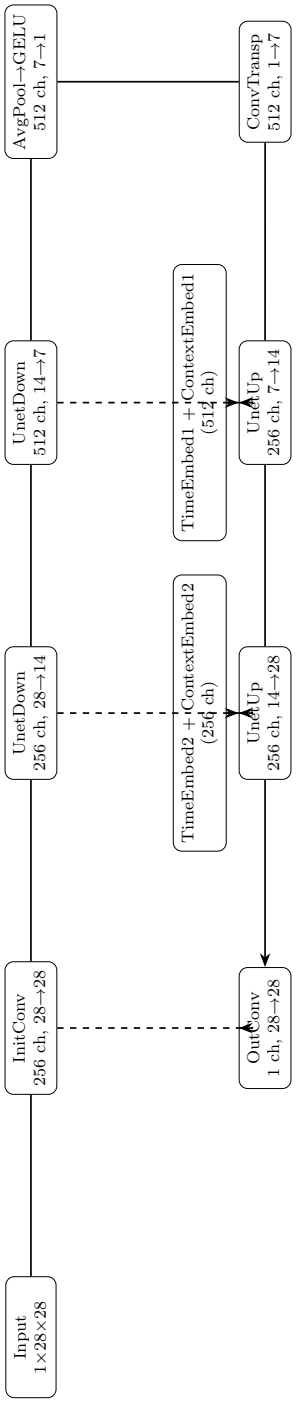

Figure 24: Arquitectura de la U-Net condicional (nivel medio), con el bloque 6 en la fila inferior.

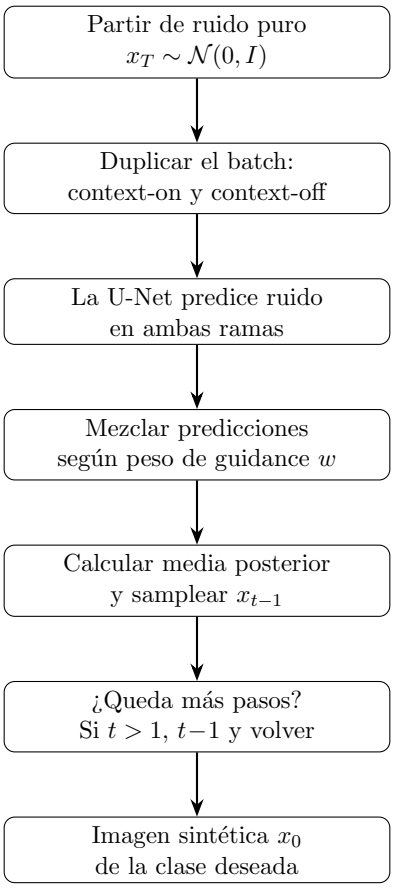

Figure 25: Diagrama de flujo del sampling inverso con guidance