



UNIVERSIDAD  
DE LA REPÚBLICA  
URUGUAY



FACULTAD DE  
INGENIERÍA

# **Hibridación de algoritmos evolutivos, redes neuronales generativas antagónicas y aprendizaje semisupervisado para mejorar clasificación de conjuntos de datos parcialmente etiquetados**

Informe de Proyecto de Grado presentado por

Franco Ferrari

en cumplimiento parcial de los requerimientos para la graduación de la carrera  
de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de  
la República

Supervisores

Sergio Nesmachnow  
Jamal Toutouh

Montevideo, 11 de noviembre de 2025



Hibridación de algoritmos evolutivos, redes neuronales generativas antagónicas y aprendizaje semisupervisado para mejorar clasificación de conjuntos de datos parcialmente etiquetados por Franco Ferrari tiene licencia [CC Atribución 4.0](#).

# Agradecimientos

En esta etapa de cierre de mi carrera de grado, quisiera expresar mi sincero agradecimiento a todas las personas que, de una u otra forma, contribuyeron a la realización de esta tesis. A cada uno de ustedes, muchas gracias.

En particular, deseo expresar mi más profundo agradecimiento a Sergio Neschmachnow y Jamal Toutouh por su generosidad al compartir su conocimiento, por su paciencia infinita y por su apoyo constante a lo largo de la realización de este proyecto.

A mi familia y amigos, quienes fueron y siempre serán mi mayor soporte y motivación. Su acompañamiento, comprensión y aliento me impulsan a seguir adelante y a crecer tanto personal como profesionalmente. Este logro es tanto mío como de ustedes.

A todos los que de alguna forma fueron parte de este camino, gracias.



# Resumen

Este proyecto tiene como objetivo entrenar redes generativas antagónicas (generative adversarial networks, GANs) mediante un enfoque semisupervisado, en un escenario caracterizado por una cantidad muy reducida de datos etiquetados y una proporción considerablemente mayor de datos sin etiquetar, una situación frecuente en aplicaciones reales donde la obtención de etiquetas es costosa o inviable.

El entrenamiento semisupervisado de GANs utilizando datasets con una cantidad limitada de etiquetas suele presentar inestabilidad, discriminadores de baja precisión y generadores que producen imágenes de calidad o diversidad insuficiente. Este proyecto aborda estos problemas trabajando sobre el dataset MNIST, utilizando entre 60 y 1000 datos etiquetados del conjunto de entrenamiento (que contiene 60000 imágenes en total) y tratando el resto como datos no etiquetados.

En una primera etapa, se establece una arquitectura para la GAN capaz de ser entrenada correctamente sin emplear algoritmos evolutivos (AE), a pesar de los pocos datos etiquetados disponibles. Posteriormente, esta arquitectura se optimiza utilizando AE que manejan en paralelo una población de generadores y otra de discriminadores, bajo esquemas mono y multi objetivo.

Los resultados muestran que, incluso sin AE, es posible entrenar con pocos datos etiquetados obteniendo resultados competitivos. La incorporación de AE mejora la estabilidad del entrenamiento, genera resultados con menor dispersión y más cercanos a una distribución normal, logrando además un rendimiento superior en comparación con el enfoque base. Este trabajo aporta evidencia experimental sobre la eficacia de técnicas evolutivas para mejorar GANs en contextos semisupervisados con datos escasos.

**Palabras clave:** redes generativas antagónicas, aprendizaje semisupervisado, algoritmos evolutivos



# Índice general

<b>1. Introducción</b>	<b>1</b>
<b>2. Marco teórico</b>	<b>5</b>
2.1. Redes neuronales . . . . .	5
2.1.1. Arquitectura . . . . .	5
2.1.2. Funcionamiento básico . . . . .	6
2.2. Redes neuronales convolucionales . . . . .	8
2.2.1. Funcionamiento y capas convolucionales . . . . .	8
2.2.2. Capas de convolución transpuesta . . . . .	10
2.3. Funciones de activación . . . . .	12
2.3.1. Leaky ReLU . . . . .	12
2.3.2. Softmax . . . . .	12
2.4. Batch normalization . . . . .	13
2.5. Capas de dropout . . . . .	14
2.6. Label smoothing . . . . .	15
2.7. Redes generativas antagónicas . . . . .	16
2.7.1. Funcionamiento y entrenamiento . . . . .	17
2.7.2. Equilibrio teórico . . . . .	18
2.8. Algoritmos evolutivos . . . . .	18
2.8.1. Algoritmos genéticos . . . . .	18
2.8.2. Selección por torneo . . . . .	20
2.8.3. Reemplazo elitista . . . . .	20
2.8.4. Algoritmo multi objetivo NSGA-II . . . . .	20
<b>3. GAN convolucional</b>	<b>23</b>
3.1. Dataset MNIST . . . . .	23
3.2. Arquitectura e implementación . . . . .	24
3.2.1. Discriminador . . . . .	24
3.2.2. Generador . . . . .	26
3.3. Entrenamiento . . . . .	26
3.3.1. Distribución de datos . . . . .	26
3.3.2. Losses . . . . .	28
3.3.3. Algoritmo de entrenamiento . . . . .	32

<b>4. GAN con AE mono objetivo</b>	<b>35</b>
4.1. Esquema general del algoritmo . . . . .	35
4.2. Evaluación de individuos . . . . .	36
4.3. Algoritmo evolutivo planteado . . . . .	38
4.3.1. Inicialización . . . . .	38
4.3.2. Operador de selección . . . . .	39
4.3.3. Operador de cruzamiento . . . . .	39
4.3.4. Algoritmo de reemplazo . . . . .	40
4.3.5. Algoritmo evolutivo completo . . . . .	40
<b>5. GAN con AE multi objetivo</b>	<b>43</b>
5.1. Evaluación de individuos . . . . .	43
5.1.1. Fitness de los generadores . . . . .	43
5.1.2. Fitness de los discriminadores . . . . .	44
5.2. Algoritmo evolutivo planteado . . . . .	44
5.2.1. Selección y reemplazo . . . . .	45
5.2.2. Algoritmo evolutivo completo . . . . .	45
<b>6. Evaluación experimental</b>	<b>47</b>
6.1. Metodología . . . . .	47
6.1.1. Metodología para el análisis del estado del arte . . . . .	47
6.1.2. Metodología para evaluar las GANs . . . . .	48
6.1.3. Metodología para evaluar ejecuciones múltiples en la etapa de configuración . . . . .	49
6.1.4. Metodología para la configuración de los parámetros de la GAN . . . . .	49
6.1.5. Metodología para la configuración de los parámetros de los AE . . . . .	50
6.1.6. Metodología para la validación de resultados . . . . .	51
6.2. Evaluación del enfoque sin AE . . . . .	52
6.2.1. Configuración paramétrica . . . . .	52
6.2.2. Validación . . . . .	58
6.3. Evaluación del enfoque con AE mono objetivo . . . . .	61
6.3.1. Configuración paramétrica . . . . .	62
6.3.2. Validación . . . . .	66
6.4. Evaluación del enfoque con AE multi objetivo . . . . .	70
6.4.1. Configuración paramétrica . . . . .	70
6.4.2. Validación . . . . .	74
6.5. Análisis de resultados . . . . .	81
6.5.1. Precisión . . . . .	81
6.5.2. FID e imágenes generadas . . . . .	84
6.5.3. Tiempos de ejecución . . . . .	86
6.5.4. Comparación con arquitecturas del estado del arte . . . . .	88

<b>7. Conclusiones y trabajo futuro</b>	<b>91</b>
7.1. Conclusiones	91
7.2. Trabajo futuro	93
7.2.1. Integración con arquitecturas del estado del arte	93
7.2.2. Extensión a otros datasets	93
7.2.3. Funciones de fitness de los AE	94
7.2.4. Paralelización del entrenamiento	94
<b>Referencias</b>	<b>95</b>
<b>A. Anexo 1</b>	<b>97</b>
A.1. Configuración paramétrica de la GAN	97
A.1.1. Estadísticas de precisión y FID para la configuración de la arquitectura de la GAN	97
A.1.2. Hipervolúmenes de las configuraciones finales de la arquitectura	99
A.1.3. Imágenes generadas en la configuración paramétrica para configuraciones sin uso de batch normalization	101
A.1.4. Hipervolúmenes de la configuración final del entrenamiento con 1000 datos etiquetados	102
A.1.5. Hipervolúmenes de la configuración de los datos etiquetados	103
A.2. Evaluación del enfoque sin AE	106
A.2.1. Imágenes generadas	106
A.3. Evaluación del enfoque con AE mono objetivo	107
A.3.1. Imágenes generadas	107
A.4. Evaluación del enfoque con AE multi objetivo	109
A.4.1. Imágenes generadas	109
A.4.2. Composición de frentes de Pareto de los discriminadores	110
A.5. Análisis de resultados	111
A.5.1. Boxplots de precisiones	111
A.5.2. Boxplots de FID	112



# Capítulo 1

## Introducción

Desde su creación en 2014, las redes generativas antagónicas (generative adversarial networks, GANs) han sido ampliamente utilizadas debido a su efectividad para resolver problemas de clasificación y generación de imágenes realistas. Durante la última década se han logrado numerosos avances en relación con la arquitectura y los métodos de entrenamiento de las GANs. Sin embargo, estas redes aún presentan limitaciones importantes al ser entrenadas con una cantidad escasa de datos etiquetados.

En contextos donde las etiquetas son escasas pero se cuenta con una mayor proporción de datos sin etiquetar, el entrenamiento semisupervisado surge como una alternativa viable. No obstante, su aplicación en GANs presenta dificultades: los discriminadores suelen alcanzar precisiones bajas, tanto globales como por clase, y los generadores tienden a producir imágenes de calidad deficiente, con poca diversidad, o incluso a colapsar, generando únicamente un conjunto reducido de ejemplos muy similares. Este proyecto aborda dichos problemas utilizando el dataset MNIST. Si bien este dataset cuenta con 60000 imágenes etiquetadas, se simularon situaciones en las que solo se dispone de una cantidad muy limitada de datos etiquetados (entre 60 y 1000) y una mayor proporción de datos sin etiquetar (el resto del dataset).

En primer lugar, se definió una arquitectura para la GAN que soporte el entrenamiento con muy pocos datos etiquetados sin caer en los problemas típicos, mencionados en el párrafo anterior. La arquitectura fue diseñada siguiendo las mejores prácticas y técnicas reportadas en el estado del arte de las GANs. Para su desarrollo, se emplearon técnicas como batch normalization (sección 2.4), dropout (sección 2.5), label smoothing (sección 2.6) y funciones de activación específicas como leaky ReLU (subsección 2.3.1). Además, se implementó una estrategia no tradicional respecto al uso de los datos etiquetados, en la cual estos datos se reutilizan en todas las iteraciones del entrenamiento.

Al evaluar el modelo, los resultados indicaron que es posible entrenar correctamente una GAN en contextos de escasez de datos etiquetados y obtener buenos resultados. No obstante, este modelo presentó resultados muy dispersos, presentando ejecuciones con resultados muy buenos y otras con resultados

claramente deficientes.

Trabajos previos, como el artículo ([Sedeño, Toutouh, y Chicano, 2025](#)) o la tesis ([Nalluru, 2023](#)), integran algoritmos evolutivos (AE) en enfoques de entrenamiento semisupervisado de GANs, obteniendo resultados prometedores. En consecuencia, en este proyecto se propuso hibridar la arquitectura previamente obtenida con un AE que trabaje de forma paralela con una población de generadores y otra de discriminadores, siguiendo enfoques mono y multi objetivo. Los AE utilizados fueron implementados desde cero para tener mayor control y flexibilidad.

El procedimiento planteado es el mismo para el AE mono objetivo y el multi objetivo: se itera sobre cada población, produciendo una nueva generación en cada paso, con el objetivo de obtener mejores individuos respecto a la generación anterior. En cada iteración se emparejan generadores y discriminadores, se entranan simultáneamente y se seleccionan los individuos de mayor aptitud para formar la siguiente generación. Así, un generador puede ser entrenado con distintos discriminadores a lo largo del proceso y viceversa.

Las funciones objetivo utilizadas por cada población fueron construidas en base a las funciones de loss empleadas durante el entrenamiento. Para los discriminadores, se consideran losses supervisados y no supervisados, descritos en ([Salimans y cols., 2016](#)). Para los generadores se utilizan dos losses descritos en ([Wang, Xu, Yao, y Tao, 2018](#)): uno asociado a la calidad de las imágenes generadas y otro asociado a la diversidad de las mismas.

Los resultados muestran que la incorporación de AE permite un entrenamiento más robusto, logrando distribuciones de resultados más estables y con menor variabilidad en comparación con el enfoque no evolutivo.

En cuanto al desempeño de los enfoques, en el caso del AE mono objetivo, las precisiones obtenidas fueron más bajas que en el resto de los enfoques. Esto se debió a que los losses supervisado y no supervisado, de distinta magnitud, fueron evaluados en conjunto. Como resultado, el loss no supervisado dominó la función de fitness, generando discriminadores más débiles. Sin embargo, esta debilidad en los discriminadores permitió que los generadores alcanzaran mejores resultados en ejecuciones específicas, produciendo las imágenes de mayor calidad para todos los escenarios estudiados.

En cuanto al AE multi objetivo, el cual evalúa de forma independiente los componentes supervisado y no supervisado, este obtuvo mejoras significativas en las precisiones obtenidas por los discriminadores, logrando medias considerablemente más altas que las obtenidas por el resto de los enfoques, independientemente de la cantidad de datos etiquetados. Sin embargo, este AE no evidenció mejoras sustanciales en la calidad de las imágenes generadas respecto a los otros enfoques.

En síntesis, este proyecto demuestra que es posible mejorar la estabilidad y el rendimiento de GANs en contextos semisupervisados mediante AE personalizados. La comparación entre enfoques mono y multi objetivo evidencia fortalezas y limitaciones específicas, aportando información valiosa para investigaciones futuras.

Este informe está estructurado en capítulos. En el capítulo [2](#) se presenta el

marco teórico requerido para comprender el proyecto en detalle, incluyendo temas relevantes como redes neuronales, GANs y AE. En el capítulo [3](#) se presenta la arquitectura utilizada para generar una GAN según el estado del arte que no utiliza AE. Los capítulos [4](#) y [5](#) detallan los enfoques con AE mono objetivo y multi objetivo, respectivamente. La experimentación y resultados se presentan en el capítulo [6](#), y finalmente, las conclusiones y trabajo futuro en el capítulo [7](#).



# Capítulo 2

## Marco teórico

En este capítulo se abordan en detalle los conceptos necesarios para la comprensión de este proyecto. Se presentan las redes neuronales, las redes neuronales convolucionales, las GANs y los AE. En el caso de los AE, se los aborda de forma general, complementando con técnicas y operadores específicos empleados a lo largo del proyecto.

### 2.1. Redes neuronales

Las redes neuronales constituyen un modelo de aprendizaje automático inspirado en el funcionamiento del cerebro humano. Su propósito es transformar datos de entrada en representaciones útiles que faciliten la extracción de información significativa. Estas representaciones se aplican en un amplio conjunto de tareas, que incluyen reconocimiento de patrones, predicción de variables y generación de contenido sintético, como imágenes, audio o texto. Esto posiciona a las redes neuronales como una herramienta fundamental en diversos ámbitos de clasificación y generación de datos.

En esta sección se abordan las redes neuronales totalmente conectadas (también conocidas como redes densas o fully connected), una de las arquitecturas más tradicionales y simples dentro del campo.

#### 2.1.1. Arquitectura

La arquitectura de una red neuronal está organizada en capas, cada una de las cuales contiene un conjunto de neuronas que reciben información, la procesan y producen una salida que se transmite a la capa siguiente.

Como se ilustra en la figura 2.1, estas redes se componen de tres tipos de capas: una capa de entrada, que recibe los datos a analizar; un conjunto de capas intermedias (también llamadas capas ocultas), en las que cada neurona transforma la información que recibe y la envía, en el caso específico de redes

totalmente conectadas, a todas las neuronas de la siguiente capa; y una capa de salida, que entrega el resultado final del procesamiento.

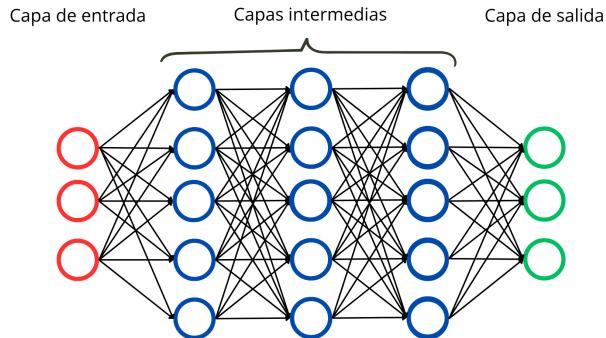


Figura 2.1: Arquitectura de una red neuronal con una capa de entrada, tres capas intermedias y una capa de salida.

El entrenamiento de la red consiste en ajustar las transformaciones realizadas por las neuronas de las capas intermedias con el fin de lograr un objetivo determinado. Para ello se utilizan datos de un conjunto de entrenamiento (dataset), que puede ser total o parcialmente etiquetado, siendo las etiquetas un elemento esencial para que la red pueda aprender de sus errores.

Como ejemplo, supóngase que se busca entrenar una red neuronal para distinguir entre imágenes reales y falsas. El dataset debe contener ambas clases de imágenes, acompañadas de una etiqueta que indique si la imagen es real o no. Con base en estas etiquetas, la red evalúa su desempeño y ajusta progresivamente sus parámetros internos para reducir el error de clasificación en iteraciones posteriores.

### 2.1.2. Funcionamiento básico

Como fue mencionado previamente en esta sección, las redes neuronales se componen de capas, las cuales están formadas por neuronas que aplican transformaciones sobre la información. Cada arista que une dos neuronas posee un peso, y cada neurona, un sesgo. Estos parámetros definen la transformación realizada por la neurona y deben ser modificados a lo largo del entrenamiento para que la red reduzca sus errores.

Cada neurona realiza un proceso compuesto por varios pasos. En primer lugar, la neurona toma las salidas de todos las neuronas de la capa anterior (o la entrada en caso de la primera capa) en conjunto con los pesos de las aristas que vinculan cada neurona de la capa anterior con la neurona actual y realiza una combinación lineal de dichos datos. Posteriormente, se añade el sesgo de la neurona actual al resultado de la combinación lineal. Finalmente, sobre el resultado obtenido se aplica una función de activación, usualmente no lineal, que introduce la capacidad de modelar relaciones complejas y facilita la

convergencia del entrenamiento. Estas funciones se abordarán en detalle en la sección 2.3. El resultado obtenido tras aplicar la función de activación constituye la salida de la neurona, y se transmite como entrada a las neuronas de la capa siguiente.

Para expresar matemáticamente la salida de la  $i$ -ésima neurona de la capa  $L$ , denominada  $a_i^L$ , se denotan:

- $N$  la cantidad de neuronas de la capa  $L - 1$ .
- $a_j^{L-1}$  la salida de la  $j$ -ésima neurona de la capa  $L - 1$
- $w_{ji}^L$  el peso de la arista que conecta a la  $j$ -ésima neurona de la capa  $L - 1$  con la  $i$ -ésima neurona de la capa  $L$
- $b_i^L$  el sesgo que tiene la  $i$ -ésima neurona de la capa  $L$
- $\sigma$  la función de activación utilizada.

Entonces,  $a_i^L$  se calcula utilizando la ecuación 2.1.

$$a_i^L = \sigma \left( \sum_{j=1}^N w_{ji}^L a_j^{L-1} + b_i^L \right) \quad (2.1)$$

Una vez conocidas la función de cada neurona y la estructura de la red, solo resta entender cómo a partir de las entradas y las transformaciones sucesivas se obtiene la salida deseada. Esta salida, que puede ser un valor o un vector de valores dependiendo de cuántas neuronas componen la última capa, representa una inferencia sobre la entrada, como por ejemplo determinar si una imagen es real o falsa en un caso práctico sencillo.

Durante el entrenamiento de la red, se calcula un error en función del desempeño, a partir del cual se actualizan los parámetros de toda la red (pesos y sesgos). El error se obtiene de aplicar una función de loss (o pérdida) que compara la salida de la red con el resultado esperado y devuelve un valor que cuantifica el error. El resultado esperado se obtiene típicamente de las etiquetas en el caso de datos etiquetados, pero puede llegar a ser más complejo de calcular en otros casos.

Una vez obtenido el error de la red, se deben actualizar todos sus parámetros. Para lograr este objetivo se aplica un algoritmo de descenso por gradiente, el cual ajusta los pesos y sesgos en la dirección que minimiza la función de loss. En este proceso es necesario calcular las derivadas parciales de dicha función con respecto a cada parámetro de la red, lo que puede resultar costoso si se realiza de forma directa. Para hacerlo de manera eficiente e iterativa se emplea el algoritmo de backpropagation. Este algoritmo aplica la regla de la cadena, reutilizando los gradientes calculados en capas posteriores para obtener los de las capas anteriores. De esta forma, logra propagar el error desde la salida hasta la entrada de la red y actualizar todos los parámetros de forma eficiente.

## 2.2. Redes neuronales convolucionales

Las redes neuronales convolucionales (convolutional neural networks, CNN) constituyen un tipo específico de arquitectura de red que, a diferencia de las redes totalmente conectadas, está especializada en el procesamiento de datos con estructura espacial, como imágenes y secuencias de video. Las CNN se destacan por sus resultados superiores en tareas como clasificación de imágenes, detección de objetos y reconocimiento de patrones visuales. En esta sección se presenta el funcionamiento de las CNN, al igual que sus diferencias con las redes neuronales totalmente conectadas (presentadas en la sección anterior).

### 2.2.1. Funcionamiento y capas convolucionales

La característica distintiva de una CNN es que, en lugar de emplear capas totalmente conectadas, donde cada nodo está conectado con todos los de la capa anterior y posterior, utiliza capas convolucionales. Cada capa convolucional está formada por un conjunto de filtros, que son pequeños tensores tridimensionales con pesos entrenables.

Los filtros permiten detectar patrones locales en los datos, utilizando muchos menos parámetros que una capa totalmente conectada y preservando las relaciones espaciales. El resultado de aplicar un filtro sobre una entrada es un mapa de características, una matriz que indica las regiones donde el patrón aprendido por el filtro aparece con mayor intensidad. La figura 2.2 ilustra este proceso.

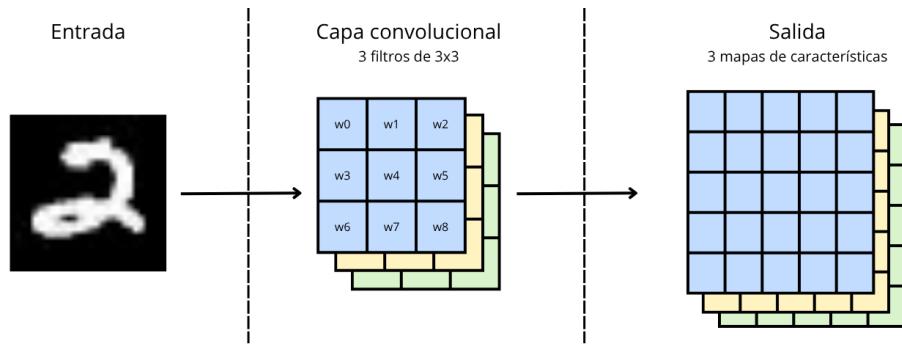


Figura 2.2: Funcionamiento de una capa convolucional con tres filtros.

Se debe considerar que tanto la entrada de una CNN como la entrada de cada capa convolucional es un tensor. En el caso de una imagen, este tensor es tridimensional: las dos primeras dimensiones corresponden al alto y ancho (tamaño espacial), y la tercera dimensión corresponde a los canales de color. Por ejemplo, una imagen RGB posee tres canales, mientras que una imagen en escala de grises solo uno.

Para comprender cómo una entrada es afectada al pasar por una capa convolucional, es necesario entender en mayor profundidad cómo funcionan los filtros.

El funcionamiento de un filtro, ilustrado en la figura 2.3, se basa en un desplazamiento de una grilla virtual sobre toda la entrada, realizando en cada posición un producto escalar entre los valores seleccionados del tensor de entrada y los pesos del filtro. El resultado de esta operación se asigna a la posición correspondiente en el mapa de características. A medida que el filtro aprende a identificar un patrón concreto, el mapa reflejará con mayor intensidad las regiones donde ese patrón aparece.

En la práctica, la entrada de una capa convolucional suele tener múltiples canales. En ese caso, cada filtro tiene la misma cantidad de canales que la entrada, por lo que está compuesto por un conjunto de matrices (una por cada canal). Durante la convolución, el filtro se aplica a cada canal de la entrada por separado, obteniendo una matriz intermedia para cada canal. Luego, los resultados obtenidos para todos los canales se suman y se asigna este valor a la posición correspondiente en el mapa de características. Este proceso permite que cada filtro detecte patrones que dependen de combinaciones específicas de información entre distintos canales de la entrada.

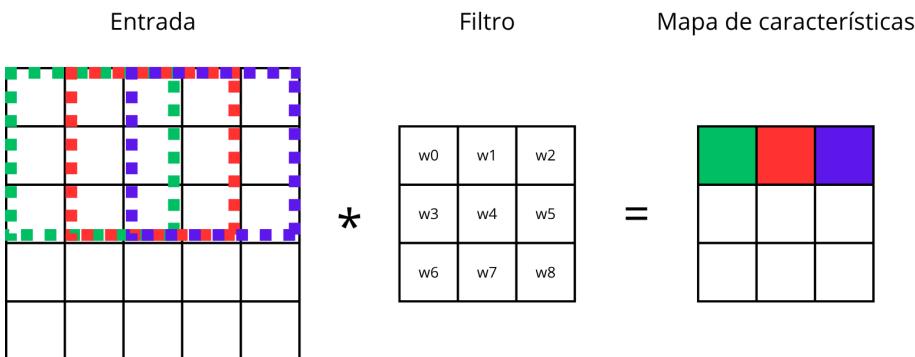


Figura 2.3: Aplicación de filtro a una entrada para obtener un mapa de características.

Dado que una capa convolucional utiliza múltiples filtros y obtiene un único mapa de características por cada uno, el número de canales en el tensor de salida es igual al número de filtros empleados.

A medida que se avanza desde las capas iniciales hacia las más profundas, el tamaño espacial de los tensores intermedios suele reducirse, mientras que el número de canales tiende a aumentar debido al uso de más filtros. Este proceso construye una jerarquía de representaciones: las capas iniciales capturan características simples como bordes o texturas, las capas intermedias combinan estas características para reconocer formas más complejas, mientras que las capas profundas identifican objetos completos. De esta forma, las capas convolucionales transforman progresivamente la entrada para producir una representación rica en información, basada en los patrones aprendidos por los filtros.

Además de capas convolucionales, una CNN suele incluir otros tipos de capas que mejoran el rendimiento o la estabilidad del entrenamiento. Algunos ejemplos

son la reducción de la dimensionalidad espacial mediante capas de pooling y la normalización de activaciones (batch normalization).

Finalmente, cuando la CNN es entrenada para algunas tareas específicas como clasificación o regresión, es habitual que la arquitectura concluya con una o varias capas totalmente conectadas, encargadas de integrar la información extraída por las capas anteriores y generar la salida final.

En la figura 2.4 se muestra un ejemplo de arquitectura para la clasificación de dígitos manuscritos del dataset MNIST, la cual presenta una reducción progresiva del tamaño espacial y un incremento en el número de canales a medida que se avanza en la red, hasta lograr un vector de largo diez que representa las probabilidades de que la entrada pertenezca a cada clase.

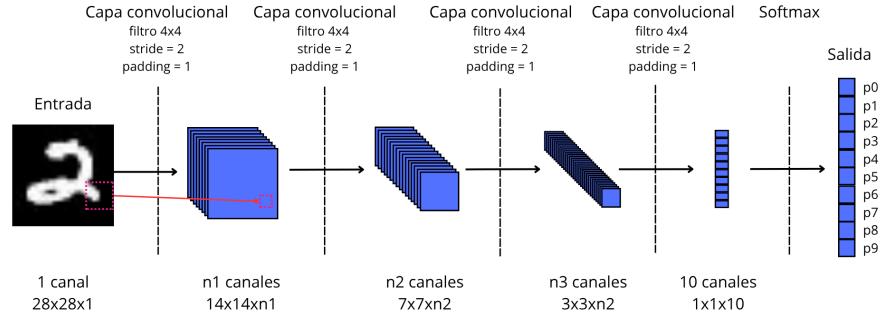


Figura 2.4: Arquitectura de una CNN con cuatro capas convolucionales para clasificar imágenes del dataset MNIST.

### 2.2.2. Capas de convolución transpuesta

En la subsección anterior se presentó el funcionamiento de las capas convolucionales, las cuales toman un tensor de cierto tamaño como entrada y generalmente reducen su tamaño espacial (primeras dos dimensiones) a la vez que aumentan su cantidad de canales (su profundidad). Este proceso basado en el uso de filtros es clave para la resolución de problemas como la clasificación de imágenes. En esta subsección se desarrollan las capas convolucionales transpuestas, las cuales producen un efecto opuesto sobre el tamaño espacial. Estas capas trabajan, generalmente, expandiendo la entrada en su tamaño espacial y reduciendo la cantidad de canales. Las capas convolucionales transpuestas son útiles en tareas como la generación de imágenes realistas a partir de un vector aleatorio o el aumento de la resolución de una imagen.

Al igual que en una convolución normal, una capa convolucional transpuesta está formada por filtros con pesos entrenables, pero el proceso de aplicación de estos filtros difiere. En lugar de recorrer la entrada aplicando el filtro y reduciendo el tamaño espacial, la convolución transpuesta expande la información: cada valor de la entrada se multiplica por todos los valores del filtro, generando una contribución que se coloca en posiciones específicas de la salida.

Al multiplicar un valor de la entrada por un filtro, las posiciones en las que se colocan los resultados en la salida están determinadas por el stride. El stride indica cuántos lugares deben desplazarse en la salida por cada desplazamiento en la entrada. Cuando las salidas de este proceso se solapan en alguna posición de la matriz de salida, sus valores se suman, permitiendo reconstruir patrones más suaves y continuos. En la figura 2.5 se desglosa este proceso utilizando dos strides diferentes.

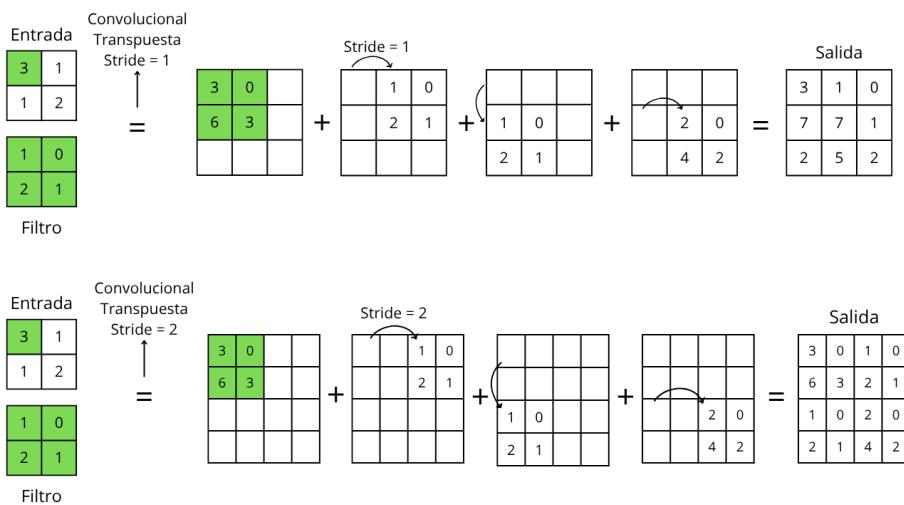


Figura 2.5: Aplicación de filtro de  $2 \times 2$  en una capa convolucional transpuesta con distintos strides sobre una misma entrada de  $2 \times 2$ .

En casos prácticos reales, la entrada suele tener más de un canal. En dichos casos cada filtro de la capa convolucional transpuesta tiene la misma cantidad de canales que la entrada y genera un único mapa de características bidimensional. En consecuencia, la cantidad de canales en la salida corresponde a la cantidad de filtros utilizados en la capa.

En cuanto a lo que aprenden los filtros, las capas de convolución transpuesta iniciales tienden a generar estructuras globales como formas generales o distribución de color. A medida que la información avanza hacia capas más profundas, los filtros refinan la estructura y añaden detalles progresivamente más finos, hasta llegar a las últimas capas, que capturan bordes, texturas y detalles. De esta forma, se logra aumentar el tamaño espacial de una entrada utilizando filtros que reconstruyen patrones locales aprendidos, cuyos pesos se optimizan durante el entrenamiento mediante el algoritmo de backpropagation, al igual que en las redes totalmente conectadas.

## 2.3. Funciones de activación

Las funciones de activación son funciones matemáticas aplicadas sobre las salidas de las neuronas en cada capa de una red neuronal. Su función principal es introducir no linealidad en el modelo, lo que permite que la red aprenda y represente relaciones complejas entre los datos. Sin estas funciones, la red sería una composición lineal de transformaciones, limitando severamente su capacidad para modelar problemas reales.

Estas funciones se utilizan tanto en redes totalmente conectadas, donde se aplican directamente a las salidas de cada neurona, como en redes convolucionales, donde se suelen implementar como capas independientes que aplican la función de activación a cada canal de la salida. En esta sección se describen dos funciones de activación relevantes para este proyecto: la función leaky ReLU, comúnmente empleada en capas intermedias, y la función softmax, utilizada como normalizadora de probabilidades en la capa de salida para tareas de clasificación.

### 2.3.1. Leaky ReLU

La función leaky ReLU (leaky rectified linear unit) es una variante de la función ReLU, diseñada para mitigar ciertas limitaciones de esta última. Estas funciones se expresan en las ecuaciones 2.3 y 2.2, respectivamente.

$$ReLU(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases} \quad (2.2)$$

$$Leaky\_ReLU(x) = \begin{cases} x & x \geq 0 \\ \alpha x & x < 0 \end{cases} \quad (2.3)$$

La función ReLU introduce no linealidad al establecer en cero los valores negativos, lo que facilita el aprendizaje de relaciones complejas y mejora la eficiencia computacional al evitar la saturación de gradientes, un problema frecuente en funciones tradicionales como sigmoid o tanh. Sin embargo, ReLU puede provocar el problema conocido como "dying ReLU", donde neuronas dejan de activarse si reciben entradas negativas constantes, lo que impide la actualización de sus parámetros. La función leaky ReLU, ilustrada en la figura 2.6, soluciona este inconveniente agregando un coeficiente  $\alpha$ , generalmente pequeño y positivo, para valores negativos. Esto garantiza que la neurona nunca quede inactiva completamente durante el entrenamiento, facilitando la propagación del gradiente y mejorando la convergencia. En este proyecto se utilizó la función de leaky ReLU con  $\alpha = 0.2$ , un valor comúnmente empleado.

### 2.3.2. Softmax

La función softmax es ampliamente utilizada en problemas de clasificación multi-clase. Su propósito es transformar un vector de valores reales arbitrarios en un vector de probabilidades, donde cada elemento está en el intervalo  $[0, 1]$

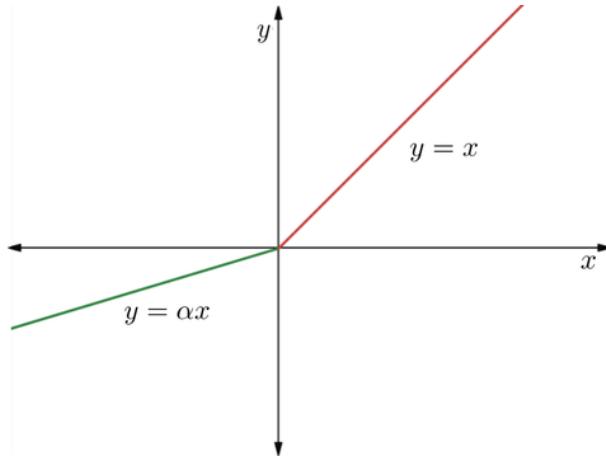


Figura 2.6: Función leaky ReLU.

y la suma total es igual a uno. Esto permite interpretar la salida como una distribución de probabilidad sobre las posibles clases.

Matemáticamente, la función de softmax se define en la ecuación 2.4, donde  $z_i$  representa el valor pre-activación de la salida de la  $i$ -ésima neurona antes de aplicar softmax,  $\sigma(z_i)$  es la probabilidad normalizada resultante, y  $K$  es la cantidad de neuronas de la capa de salida.

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad \text{para } i = 1, 2, \dots, K \quad (2.4)$$

En el contexto de esta tesis, el discriminador de la red generativa antagónica utiliza softmax en su capa de salida para calcular la probabilidad de que una entrada pertenezca a cada una de las clases consideradas.

## 2.4. Batch normalization

La capa de batch normalization (Ioffe y Szegedy, 2015) es un tipo de capa utilizado en redes neuronales que permite un entrenamiento más estable y rápido mediante la normalización de datos.

Las capas de batch normalization cumplen la función de normalizar las entradas, restando la media del mini-batch y dividiendo por su desviación estándar para cada dimensión. Posteriormente, los datos normalizados se multiplican por un valor  $\gamma$  y se suma un desvío  $\beta$ . Estos parámetros se aprenden durante el entrenamiento mediante descenso por gradiente y brindan mayor libertad a la red, ya que permiten ajustar la escala y el desplazamiento de las activaciones normalizadas. En consecuencia, el uso de capas de batch normalization permite que los datos ingresen a la siguiente capa manteniendo una distribución más estable, lo que facilita un descenso por gradiente más controlado y soluciona el

problema conocido como “internal covariate shift”. Este problema se produce cuando pequeñas variaciones en los pesos de una capa provocan cambios significativos en la distribución de las entradas de capas posteriores, dificultando el entrenamiento. Dado que los pesos se actualizan constantemente durante el entrenamiento, la distribución de los datos puede variar considerablemente. Por ello, la normalización mediante batch normalization ayuda a estabilizar estas variaciones y acelera la convergencia del modelo.

La función matemática utilizada por las capas de batch normalization, así como los cálculos de media y varianza, se presentan en el algoritmo 1.

---

**Algoritmo 1** Batch normalization

---

Entrada: mini-batch:  $\mathcal{B} = \{x_1 \dots m\}$ ;

Parámetros a aprender:  $\gamma, \beta$

$$\begin{aligned}
 \mu_{\mathcal{B}} &\leftarrow \frac{1}{m} \sum_{i=1}^m x_i && \text{media del mini-batch} \\
 \sigma_{\mathcal{B}}^2 &\leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 && \text{varianza del mini-batch} \\
 \hat{x}_i &\leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} && \text{normalizar datos} \\
 y_i &\leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) && \text{escalar y desplazar}
 \end{aligned}$$


---

## 2.5. Capas de dropout

El dropout (Srivastava, Hinton, Krizhevsky, Sutskever, y Salakhutdinov, 2014) es una técnica utilizada en redes neuronales para reducir el sobreajuste durante el entrenamiento. Esta técnica consiste en desactivar neuronas de manera aleatoria e independiente con una probabilidad  $p$  (denominada coeficiente de dropout) en cada iteración del entrenamiento. Cuando una neurona se desactiva, no participa en la propagación hacia adelante ni en la retropropagación, y sus conexiones entrantes y salientes se eliminan temporalmente. A modo de ejemplo, en la figura 2.7 se muestra una red neuronal sin dropout y dos ejemplos de cómo se desactivan distintas neuronas en diferentes iteraciones.

Durante el entrenamiento, esta desactivación aleatoria fuerza a que las neuronas aprendan a no depender excesivamente de otras neuronas específicas, ya que no pueden “confiar” en que estarán activas en otra iteración. Esto fomenta la independencia y robustez de las características aprendidas, llevando a representaciones más generales y menos propensas al sobreajuste.

En la fase de inferencia, el dropout no se aplica; en su lugar, las activaciones o pesos se escalan para compensar la reducción que hubo durante el entrenamiento, manteniendo coherente la magnitud de las salidas.

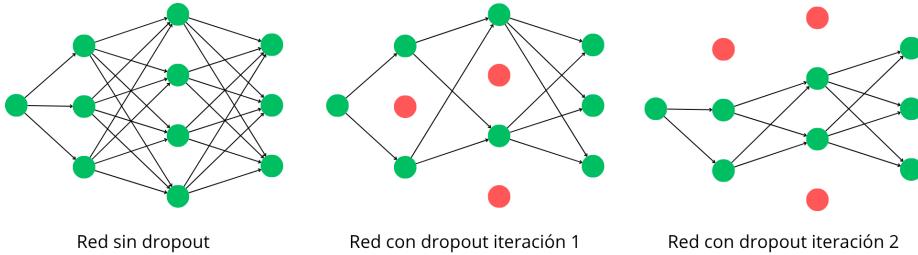


Figura 2.7: Variaciones en la arquitectura de una red neuronal como resultado del uso de dropout

Matemáticamente, el dropout puede interpretarse como un entrenamiento conjunto de un gran número de sub-redes con pesos compartidos, donde cada iteración utiliza una combinación distinta de neuronas activas. El resultado final es similar a promediar los modelos producidos por estas sub-redes, mejorando la capacidad de generalización.

En CNNs, se utilizan capas de dropout que desactivan canales completos de los mapas de características en lugar de neuronas individuales, preservando así la coherencia espacial. Al apagar canales completos, se eliminan temporalmente las respuestas de determinados filtros, lo que fomenta que la red no dependa en exceso de detectores de patrones específicos y favorece una mayor independencia y diversidad entre filtros.

## 2.6. Label smoothing

Label smoothing (Szegedy, Vanhoucke, Ioffe, Shlens, y Wojna, 2015) es una técnica utilizada durante el entrenamiento de redes neuronales para reducir el sobreajuste y mejorar la generalización del modelo. Cuando se entrena una red neuronal para clasificación supervisada, el objetivo clásico es que la probabilidad asignada a la clase correcta sea exactamente 1 (o 100 %), es decir, que el clasificador esté completamente seguro de la etiqueta verdadera. Sin embargo, este enfoque puede llevar a un sobreajuste y a una mayor confianza que no siempre es deseable.

La técnica de label smoothing consiste en suavizar las etiquetas objetivo, asignando a la clase correcta un valor ligeramente inferior a 1, por ejemplo,  $1 - \epsilon$ . Esto evita que la red se vuelva demasiado confiada en sus predicciones, fomentando una mejor calibración y robustez.

Esta técnica suele mejorar los resultados porque reduce la probabilidad de que el modelo asigne una certeza absoluta a una sola clase, lo cual puede hacer que el modelo sea más propenso a sobreajustar los datos de entrenamiento. De esta forma, se introduce una penalización suave contra la certeza extrema, lo que lleva a que el modelo aprenda representaciones más generales y sea menos sensible a ruido o ejemplos atípicos.

En este proyecto se utilizó un valor de suavizado  $\epsilon = 0.1$ , por lo que la etiqueta objetivo para la clase correcta será 0.9 en lugar de 1. De este modo, la red aprende a no sobreajustar tan estrictamente los ejemplos correctamente clasificados durante el entrenamiento.

## 2.7. Redes generativas antagónicas

Las GANs, introducidas en 2014 (Goodfellow y cols., 2014), constituyen una arquitectura en la que dos redes neuronales compiten entre sí: un generador  $G$  y un discriminador  $D$ . El objetivo de  $G$  es producir datos sintéticos que imiten la distribución de los datos reales, mientras que  $D$  intenta distinguir entre datos reales y generados. Este esquema puede implementarse tanto con redes densas como con redes convolucionales. En el caso de redes convolucionales,  $D$  suele emplear capas de convolución para clasificar imágenes, mientras que  $G$  utiliza capas de convolución transpuesta para generar imágenes a partir de un vector de ruido (con valores aleatorios).

En la figura 2.8 se ilustra la interacción entre  $G$  y  $D$  en un entrenamiento sobre el dataset MNIST, mostrando cómo  $G$  intenta engañar a  $D$ , mientras  $D$  intenta aprender a distinguir imágenes reales de generadas.

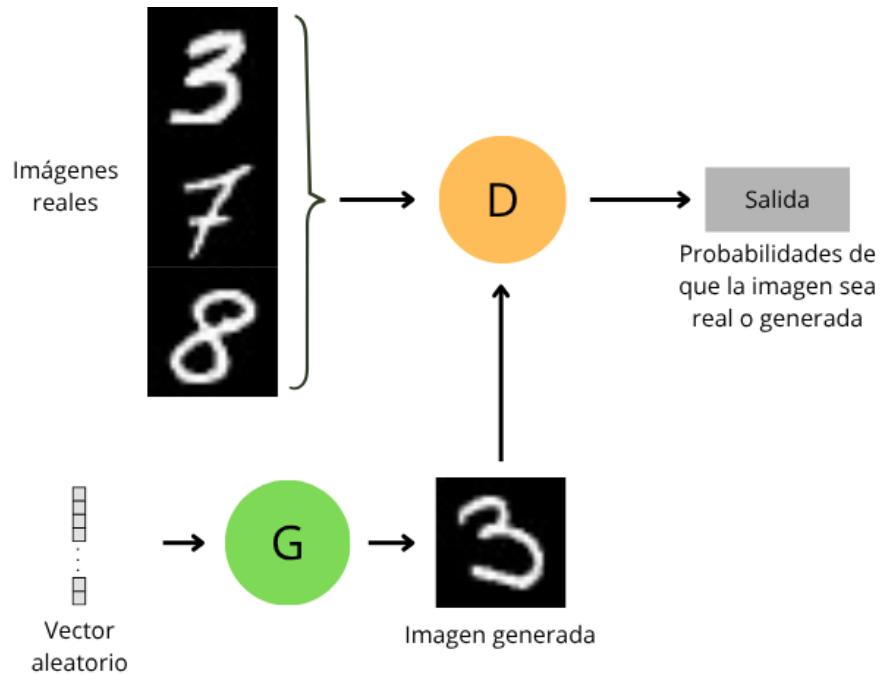


Figura 2.8: Arquitectura básica de una GAN entrenada sobre el dataset MNIST.

En algunos casos,  $D$  no solo discrimina entre muestras reales y generadas, sino que también clasifica las muestras reales en ciertas clases específicas. Para ello, su salida debe ser un vector de dimensión  $K + 1$ : las primeras  $K$  entradas corresponden a las probabilidades de pertenencia a cada clase real, mientras que la última representa la probabilidad de que la imagen sea generada por  $G$ .

### 2.7.1. Funcionamiento y entrenamiento

El funcionamiento de una GAN se basa en la interacción competitiva entre  $G$  y  $D$ . Al inicio, ambas redes tienen pesos aleatorios:  $G$  produce ruido sin sentido y  $D$  clasifica de manera prácticamente arbitraria. A medida que avanza el entrenamiento,  $D$  mejora su habilidad para reconocer correctamente las muestras reales y rechazar las falsas, mientras que  $G$  aprende a generar datos cada vez más realistas que logren confundir a  $D$ .

Matemáticamente hablando, el objetivo del entrenamiento de una GAN es aprender dos conjuntos de parámetros,  $u$  y  $v$ , que representan los valores de todos los parámetros de la red para  $G$  y  $D$  respectivamente. El aprendizaje de estos parámetros tiene como objetivo que  $G$  produzca muestras que se acerquen a la distribución de datos reales, denotada  $p_{\text{datos}}$ , al mismo tiempo que  $D$  logra discriminar correctamente.

Dado que los parámetros  $u$  y  $v$  representan configuraciones específicas de cada red, teóricamente se pueden definir dos conjuntos  $\mathcal{U}, \mathcal{V} \subseteq \mathbb{R}^p$  que contienen todas las configuraciones posibles de generadores y discriminadores, respectivamente. De manera análoga, se define una clase de generadores  $\mathcal{G}$  parametrizados por  $u \in \mathcal{U}$  y una clase de discriminadores  $\mathcal{D}$  parametrizados por  $v \in \mathcal{V}$ :  $\mathcal{G} = \{G_u \mid u \in \mathcal{U}\}$  y  $\mathcal{D} = \{D_v \mid v \in \mathcal{V}\}$ .

Ambas redes se entrena de manera adversarial, donde cada una busca mejorar su desempeño para “convencer” a la otra. El loss combinado se expresa mediante la ecuación 2.5, donde  $z$  es un vector de ruido de dimensión latente que sirve como entrada para  $G$ ,  $p_z$  es su distribución (comúnmente una distribución gaussiana o uniforme), y  $\phi : [0, 1] \rightarrow \mathbb{R}$  es una función cóncava utilizada para medir la calidad de la clasificación. En la formulación original, se toma  $\phi(t) = \log(t)$ , lo que lleva al loss denominado binary cross entropy. Además, en la implementación práctica, los valores esperados  $\mathbb{E}$  equivalen a la media obtenida sobre todos los datos utilizados en una iteración de entrenamiento.

$$L(u, v) = \mathbb{E}_{x \sim p_{\text{datos}}} [\phi(D_v(x))] + \mathbb{E}_{z \sim p_z} [\phi(1 - D_v(G_u(z)))] \quad (2.5)$$

El primer término de  $L(u, v)$  corresponde al valor esperado de la función de penalización aplicada a la probabilidad de que  $D$  clasifique correctamente como reales los datos provenientes de la distribución verdadera  $p_{\text{datos}}$ . El segundo término corresponde al valor esperado de la función de penalización aplicada a la probabilidad de que  $D$  identifique correctamente como falsas las muestras generadas por  $G$ . Por lo tanto, un mayor valor de  $L(u, v)$  refleja una mejor capacidad de  $D$  para clasificar correctamente. Por otra parte, un mayor valor de  $L(u, v)$  puede significar un mal desempeño de  $G$ , siempre que el segundo

componente, relacionado a la clasificación de datos generados, aumente. En consecuencia, el objetivo del entrenamiento se puede formalizar como un problema de minimax como se presenta en la ecuación 2.6. En este problema,  $G$  busca una configuración de parámetros  $u$  que minimice la probabilidad de que sus muestras sean identificadas como falsas, mientras que  $D$  busca una configuración de parámetros  $v$  que maximice su capacidad de distinguir datos reales de falsos. El entrenamiento se realiza de forma iterativa y alternada, utilizando los errores de la contraparte para ajustar los parámetros de cada red. Con el tiempo, esta competencia impulsa el aprendizaje conjunto, dando lugar a generadores capaces de producir datos de alta calidad y discriminadores con gran precisión.

$$\min_{u \in \mathcal{U}} \max_{v \in \mathcal{V}} L(u, v), \quad (2.6)$$

### 2.7.2. Equilibrio teórico

En el equilibrio teórico de una GAN,  $G$  ha aprendido a generar muestras tan realistas que  $D$  no puede diferenciarlas de las reales mejor que al azar, aproximando su precisión al 50 % para ambas clases (real y falsa). En este escenario ideal,  $G$  ha capturado perfectamente la distribución de datos reales  $p_{\text{datos}}$  y, desde la perspectiva de  $D$ , las muestras reales y sintéticas son indistinguibles. Sin embargo, en la práctica, alcanzar este equilibrio perfecto es difícil y suele lograrse solo de forma aproximada.

## 2.8. Algoritmos evolutivos

Los AE son técnicas de optimización inspiradas en los procesos naturales de evolución biológica. Estas técnicas son muy utilizadas para la exploración de espacios de soluciones complejos, donde otras metodologías resultan ineficientes o inviables en tiempo real. Aunque existen muchas variantes de AE, en este proyecto se trabajó específicamente con algoritmos genéticos. En consecuencia, en esta sección se introducen los algoritmos genéticos y se presentan algunos operadores evolutivos específicos utilizados a lo largo de este proyecto.

### 2.8.1. Algoritmos genéticos

Los algoritmos genéticos son un tipo de AE específico caracterizado por el uso de poblaciones de individuos sobre los cuales se aplican operadores evolutivos. Cada individuo representa una solución factible del problema a resolver, y es evaluado mediante una o múltiples funciones de fitness (también denominadas funciones objetivo) que determinan su aptitud para la supervivencia.

Estos algoritmos comienzan con una población inicial y, mediante operadores evolutivos, buscan generar nuevas generaciones de dicha población con individuos de mayor aptitud. El proceso se repite iterativamente hasta cumplirse una condición de parada, esperando encontrar soluciones óptimas o cercanas a óptimas dentro del espacio de soluciones.

El esquema general utilizado en los algoritmos genéticos es siempre el mismo: en primer lugar, se inicializa aleatoriamente una población. Luego, mediante un proceso de selección, ciertos individuos son escogidos para reproducirse, usualmente con probabilidad proporcional a su fitness. Sobre los individuos seleccionados se aplica un cruzamiento para generar nuevos individuos descendientes. Posteriormente, se introducen mutaciones aleatorias en los descendientes para aumentar la diversidad genética. Finalmente, mediante un algoritmo de reemplazo, se forma una nueva generación de la población, procurando que su aptitud general sea superior a la anterior. Este ciclo de selección, cruzamiento, mutación y reemplazo se repite hasta alcanzar el criterio de parada, obteniéndose idealmente una población final con individuos con valores de fitness altos.

El criterio para seleccionar los mejores individuos depende de la cantidad de funciones de fitness a optimizar. En problemas mono objetivo se busca maximizar o minimizar una única función de fitness, mientras que en problemas multi objetivo se consideran múltiples funciones a optimizar simultáneamente.

En el caso de algoritmos mono objetivo encontrar la mejor solución es equivalente a encontrar el individuo que tenga el mejor resultado para la única función de fitness evaluada.

En el caso multi objetivo, pueden existir soluciones que sean mejores en una función pero peores en otra. Para manejar esta situación se utiliza el concepto de frente de Pareto, que es el conjunto de soluciones no dominadas por ninguna otra. Una solución A se dice dominada por otra solución B si B es igual o mejor que A en todas las funciones de fitness y estrictamente mejor en al menos una. A modo de ejemplo, y suponiendo que se tienen únicamente dos funciones de fitness a minimizar, en la figura 2.9 se representa el frente de Pareto para un conjunto de soluciones.

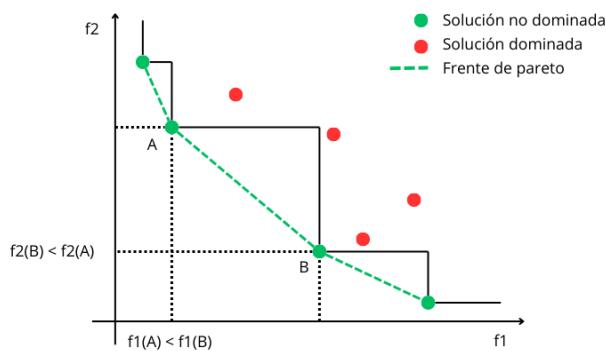


Figura 2.9: Frente de Pareto para un conjunto de soluciones que minimizan dos funciones de fitness,  $f_1$  y  $f_2$

### 2.8.2. Selección por torneo

La selección por torneo es un método ampliamente utilizado en la etapa de selección de los AE. Este procedimiento consiste en fijar un parámetro que define el tamaño del torneo y, posteriormente, seleccionar aleatoriamente un conjunto de individuos de la población con dicho tamaño. Entre ellos, se evalúa el valor de fitness de cada participante y sobrevive únicamente aquel con el valor más alto. El proceso se repite hasta obtener el número deseado de individuos.

El tamaño del torneo es un parámetro importante: a mayor tamaño aumenta el elitismo, ya que las soluciones de bajo fitness tienen más probabilidades de enfrentarse a soluciones de fitness superior. Por el contrario, cuanto más pequeño sea el tamaño del torneo, mayor será la probabilidad de que sobrevivan individuos con fitness reducido. La supervivencia de individuos menos aptos no siempre resulta perjudicial, ya que pueden contener información genética relevante para alcanzar el óptimo y que no esté presente en el resto de la población, contribuyendo así a mantener la diversidad genética. La elección del tamaño del torneo dependerá de los objetivos y del equilibrio buscado entre elitismo y diversidad en cada implementación específica.

### 2.8.3. Reemplazo elitista

En el proceso de un AE, una vez obtenidos los nuevos individuos generados mediante cruzamiento y mutación, es necesario conformar una nueva generación de la población a partir de los individuos originales y sus descendientes, para lo cual se emplea un algoritmo de reemplazo.

El reemplazo elitista consiste en seleccionar y conservar únicamente a los individuos con mejor valor de fitness hasta alcanzar el tamaño de población deseado, descartando los demás. Esta estrategia favorece la preservación de soluciones de alta calidad, pero puede llegar a reducir la diversidad genética, lo que puede aumentar el riesgo de convergencia prematura. La eliminación de individuos con fitness inferior puede privar al algoritmo de características únicas que, aunque inicialmente sean poco competitivas, resulten útiles en etapas posteriores para explorar nuevas regiones del espacio de búsqueda.

### 2.8.4. Algoritmo multi objetivo NSGA-II

El algoritmo Non-dominated Sorting Genetic Algorithm II o NSGA-II ([Deb, Pratap, Agarwal, y Meyarivan, 2002](#)) es un método ampliamente utilizado en AE multi objetivo para priorizar individuos de acuerdo con criterios de no dominancia y diversidad. NSGA-II puede emplearse en diferentes etapas del AE, como la selección o el cruzamiento.

El algoritmo trabaja sobre distintos frentes de Pareto (introducidos en la subsección [2.8.1](#)), clasificando los individuos de la población en distintas clases de no dominancia ( $F_1, F_2, \dots, F_N$ ) y posteriormente ordenando los individuos dentro de cada clase según su crowding distance. Finalmente, el algoritmo toma los individuos de las clases con menor índice (mayor prioridad) hasta completar

el tamaño de población deseado. Si el último frente necesario para completar la población contiene más individuos que los requeridos, se utiliza el crowding distance para desempatar, priorizando a los individuos que presentan un mayor valor de esta métrica. Esta decisión se basa en que los individuos con mayores valores de esta métrica se encuentran más alejados de sus vecinos, brindando mayor diversidad. El proceso de NSGA-II, en conjunto con la selección por crowding distance, se ilustran en la figura 2.10.

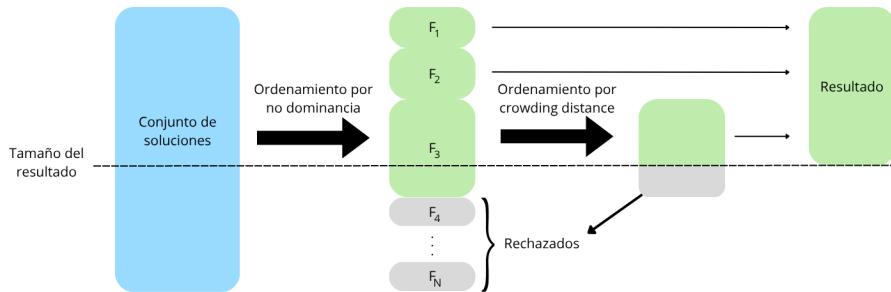


Figura 2.10: Selección aplicada por el algoritmo NSGA-II con desempate por crowding distance.

El crowding distance de un individuo  $i$  se calcula como el volumen (o, en dos dimensiones, el perímetro) del hipercuboide definido por sus dos vecinos más cercanos (uno con menor valor y otro con mayor valor) en cada función objetivo, como se muestra en la figura 2.11.

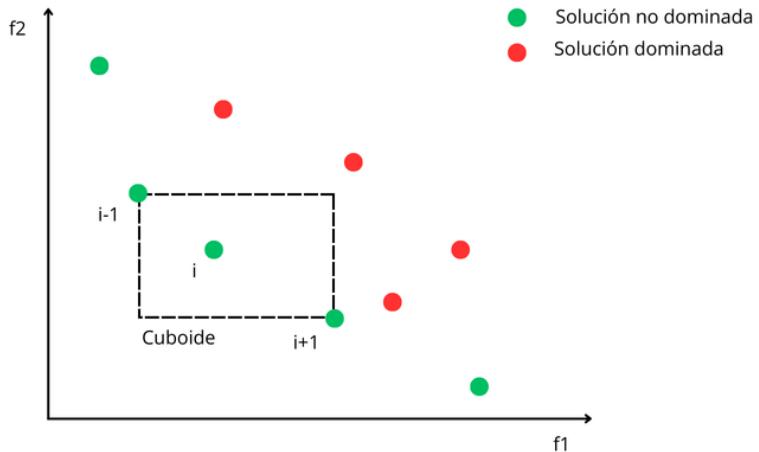


Figura 2.11: Cuboide obtenido en base al individuo  $i$  para el cálculo de crowding distance.



# Capítulo 3

## GAN convolucional

En este capítulo se presenta la arquitectura desarrollada para una GAN entrenada con una cantidad limitada de datos etiquetados del dataset MNIST, alcanzando discriminadores de alta precisión y generadores capaces de producir imágenes con buena calidad y diversidad. Primero se describe brevemente el dataset utilizado, dado que ciertas características de la arquitectura dependen directamente de él. Posteriormente, se detalla la arquitectura del discriminador y del generador, y finalmente se presentan las funciones de loss utilizadas por cada componente, en conjunto con el algoritmo de entrenamiento propuesto.

### 3.1. Dataset MNIST

El entrenamiento de la GAN se realizó sobre el dataset MNIST. Este conjunto de datos se compone de imágenes de dígitos manuscritos, cada una de  $28 \times 28$  píxeles en escala de grises. A modo de ejemplo, en la figura 3.1 se muestran 200 imágenes del dataset. En total, MNIST contiene 60000 imágenes de entrenamiento y 10000 de prueba, todas etiquetadas. Cada etiqueta corresponde al dígito representado en la imagen, abarcando diez clases (del 0 al 9).

Con el objetivo de realizar un entrenamiento semisupervisado, se simularon distintos escenarios con datos etiquetados limitados, variando su cantidad. El resto de los datos se consideró como no etiquetado. Este enfoque reproduce la situación semisupervisada en la que los datos etiquetados son escasos y existe una mayor proporción de datos sin etiquetar. Las imágenes etiquetadas se seleccionaron de manera equitativa entre las clases. Por ejemplo, si se emplean 60 imágenes, se eligen seis de cada una de las diez clases. Este balance garantiza que todas las clases estén representadas en el conjunto de entrenamiento etiquetado, lo cual favorece el aprendizaje del discriminador y mejora la estabilidad del entrenamiento.

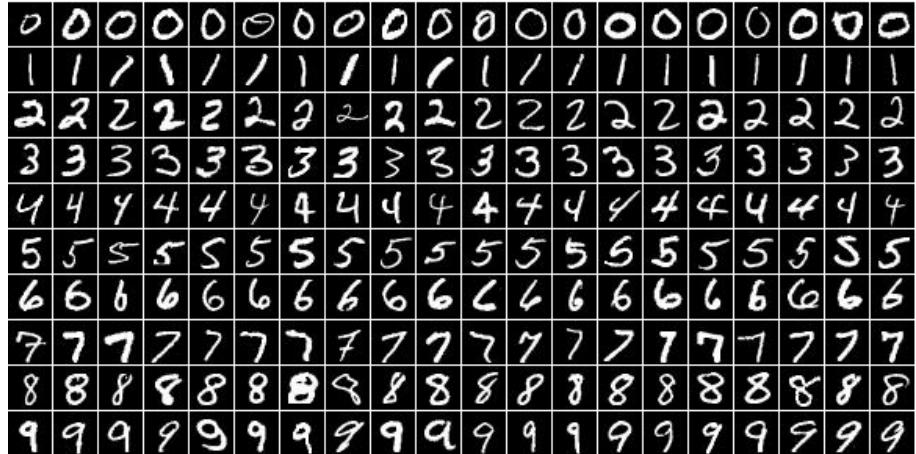


Figura 3.1: Muestra de 200 imágenes del dataset MNIST

## 3.2. Arquitectura e implementación

La implementación de la GAN fue realizada en Python utilizando la librería PyTorch. Tanto el generador como el discriminador son CNNs que se entrena de manera conjunta y competitiva. El generador  $G$  aprende a producir imágenes que imitan las reales, mientras que el discriminador  $D$  debe distinguir entre imágenes reales y sintéticas, actuando además como clasificador de dígitos. En esta sección se detallan las arquitecturas específicas del discriminador y del generador.

### 3.2.1. Discriminador

Para comprender la arquitectura de  $D$ , en esta subsección se detallan sus componentes en el orden en el que son procesados por dicha red. En primer lugar se presenta la entrada, luego las capas intermedias, y finalmente la salida.

La entrada de  $D$  consiste en imágenes de  $28 \times 28$  píxeles en escala de grises, correspondientes al dataset MNIST o generadas por  $G$ . Estas se representan como tensores de tamaño  $28 \times 28 \times 1$ .

Las imágenes utilizadas como entrada son procesadas por las capas intermedias previo a obtener la salida. La arquitectura consta de cuatro bloques convolucionales secuenciales, cada uno compuesto por múltiples capas intermedias. Los bloques están diseñados para extraer patrones y representar características jerárquicas de las imágenes. Cada bloque reduce progresivamente la dimensión espacial y aumenta el número de canales, hasta obtener finalmente un tensor de tamaño  $1 \times 1 \times 11$ , donde el valor 11 corresponde a la cantidad de clases reales ( $K = 10$ ), más una salida extra para clasificar imágenes falsas. Cada bloque combina:

- Capa convolucional (sección 2.2): extrae características y patrones locales. Reduce la dimensión espacial de la entrada y, generalmente, aumenta la profundidad de los canales.
- Batch normalization (sección 2.4): estabiliza y acelera el entrenamiento, normalizando la salida de la capa convolucional.
- Dropout (sección 2.5): aplica regularización, entrenando sub-redes dentro de  $D$  para evitar sobreajuste.
- Leaky ReLU (subsección 2.3.1): permite aprender relaciones no lineales y evita neuronas muertas.

Al final de la secuencia de bloques, la representación obtenida se proyecta a un vector de once valores, sobre el cual se aplica softmax (presentada en la subsección 2.3.2), permitiendo obtener el vector de salida que contiene las probabilidades finales. Los primeros diez valores ( $K$ ) de la salida indican la probabilidad de que la imagen sea clasificada con cada uno de los dígitos del 0 al 9. El undécimo valor ( $K + 1$ ) corresponde a la probabilidad de que la imagen sea generada por  $G$ . Formalmente, si  $S$  es el vector de salida, entonces  $S[0] \dots S[9]$  representan las probabilidades de las clases reales (dígitos del 0 al 9), y  $S[10]$  indica la probabilidad de que la imagen sea falsa. Esto permite clasificar imágenes reales y distinguir las producidas por  $G$ .

Este diseño de la salida, que utiliza únicamente softmax sobre las  $K + 1$  salidas de la red, se basa en (Nalluru, 2023), en el cual se estudiaron distintas alternativas para escenarios semisupervisados, resultando la utilizada en este proyecto la de mayor efectividad.

Finalmente, la figura 3.2 ilustra la arquitectura completa de  $D$  y su funcionamiento, según fue descrito a lo largo de esta subsección. Según la arquitectura propuesta,  $D$  toma la imagen de entrada, aplica una serie de bloques convolucionales, y finalmente utiliza la función softmax para obtener un vector de probabilidades que representa la salida.

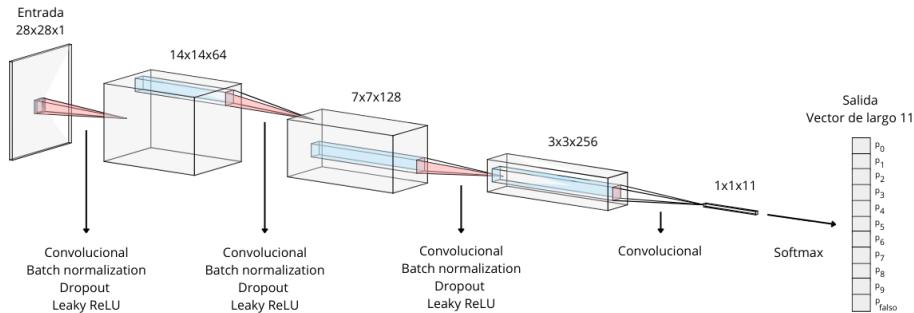


Figura 3.2: Arquitectura propuesta del discriminador para entrenamiento semi-supervisado sobre el dataset MNIST

### 3.2.2. Generador

En esta subsección se presenta la arquitectura propuesta para el generador  $G$ , explicando como se compone su entrada, sus capas intermedias y finalmente su salida.

En primer lugar  $G$  recibe como entrada un vector de ruido de dimensión 100, que introduce variabilidad, lo cual constituye una condición necesaria (aunque no suficiente) para lograr diversidad en las imágenes generadas. Dicho vector se representa como un tensor de tamaño  $1 \times 1 \times 100$  y se transforma progresivamente al pasar por las distintas capas hasta obtener un tensor de tamaño  $28 \times 28 \times 1$  que representa la imagen final. Este proceso se realiza mediante capas de convolución transpuesta (presentadas en la subsección 2.2.2) que incrementan la resolución espacial y reducen el número de canales, reconstruyendo los patrones aprendidos durante el entrenamiento.

La arquitectura de  $G$  consta de cuatro bloques de capas. Cada bloque combina:

- Capa de convolución transpuesta (subsección 2.2.2): regenera patrones aprendidos aumentando la dimensión espacial y, generalmente, reduciendo la profundidad de los canales.
- Batch normalization (sección 2.4): estabiliza y acelera el entrenamiento, normalizando la salida de la capa convolucional transpuesta.
- Leaky ReLU (subsección 2.3.1): permite aprender relaciones no lineales y evita neuronas muertas.

Finalmente,  $G$  aplica la función de activación tanh, que normaliza los valores de los píxeles en el rango  $[-1, 1]$ , de modo que puedan interpretarse como intensidades en escala de grises.

En la figura 3.3 se ilustra la arquitectura completa de  $G$ , mostrando cómo se incrementan las dimensiones espaciales y se reducen los canales hasta obtener la imagen final.

## 3.3. Entrenamiento

El entrenamiento de una GAN (introducido en la subsección 2.7.1) consiste en un juego competitivo entre dos redes neuronales: el generador  $G$  y el discriminador  $D$ . Este proceso se desarrolla a lo largo de múltiples épocas, en las cuales  $G$  y  $D$  actualizan sus parámetros en base a los errores cometidos, con el objetivo de mejorar sus desempeños. En esta sección se detallan las estrategias implementadas para la distribución de los datos, las funciones de loss utilizadas por cada red, y finalmente, el algoritmo de entrenamiento completo.

### 3.3.1. Distribución de datos

Previo a describir las estrategias de distribución de datos utilizadas, es necesario comprender la estructura iterativa del entrenamiento. El entrenamiento de

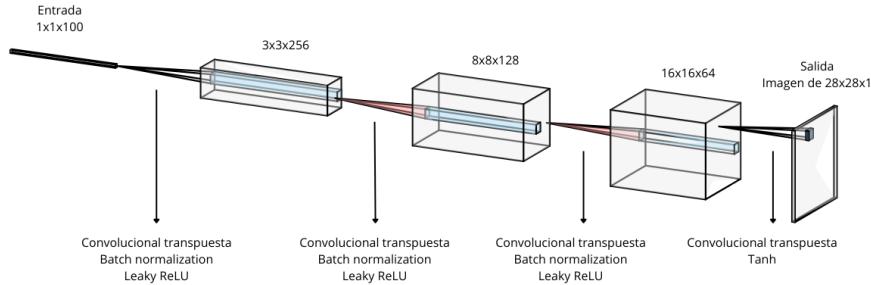


Figura 3.3: Arquitectura propuesta del generador para entrenamiento semisupervisado sobre el dataset MNIST

una GAN se realiza por épocas: una época es una iteración del entrenamiento en la cual se emplea el conjunto completo de datos reales. El entrenamiento dentro de cada época es a su vez dividido en iteraciones, las cuales son definidas por lotes de tamaño fijo (batches). El tamaño del batch determina cuántos ejemplos se procesan antes de cada actualización de parámetros y, en consecuencia, cuántas iteraciones componen una época. Por ejemplo, con 15000 datos y un tamaño de batch de 100, se realizan 150 iteraciones por época (dado que todos los datos deben ser utilizados en cada época).

En este trabajo, el entrenamiento es de tipo semisupervisado. Por lo tanto, el discriminador  $D$  recibe tres tipos de entradas:

- Imágenes reales etiquetadas
- Imágenes reales sin etiquetar
- Imágenes sintéticas generadas por  $G$

La cantidad de datos etiquetados es un parámetro del programa. Experimentalmente, en este proyecto se trabajó con distintos valores, contemplando casos de 60, 100, 500 y 1000 datos etiquetados. El resto de los datos del conjunto de entrenamiento (60000 en total) se emplean como no etiquetados. En consecuencia, durante este proyecto la cantidad de iteraciones se determinó a partir de los datos no etiquetados, que constituyen la mayor proporción del conjunto de entrenamiento.

### Uso de datos etiquetados durante el entrenamiento

Dada la escasez de datos etiquetados, se presenta el desafío de asegurar su distribución adecuada a lo largo de las iteraciones de entrenamiento. A diferencia de los datos no etiquetados, cuya abundancia permite cubrir todas las iteraciones sin restricciones, los datos etiquetados requieren una estrategia específica de utilización. A modo de ejemplo, si se utiliza un tamaño de batch de 100, se

tendrían 599 iteraciones por época (basándonos en las imágenes no etiquetadas). Si los datos se distribuyen, por ejemplo, en 100 datos etiquetados y 59900 no etiquetados, no es posible asignar un único ejemplo etiquetado por iteración (100 datos para 599 iteraciones) sin recurrir a la reutilización repetida de los mismos. Para abordar este problema, se consideraron dos estrategias de distribución de los datos etiquetados durante el entrenamiento:

1. Reparto uniforme: los datos etiquetados se distribuyen a lo largo de todas las iteraciones de una misma época, repitiéndose en caso de que no alcancen para cubrir la cantidad de iteraciones, de manera análoga a un entrenamiento convencional.
2. Uso completo por iteración: en cada iteración de la época se trabaja con el conjunto completo de datos etiquetados. Esto implica una reutilización intensiva de ejemplos, lo que en principio podría favorecer el sobreajuste del discriminador. Sin embargo, en la práctica no se presentó tal efecto, probablemente debido a la arquitectura establecida. En este caso las iteraciones de cada época quedan definidas por los datos no etiquetados. En consecuencia, en cada iteración se tiene una cantidad de datos no etiquetados igual al batch size, y todos los datos etiquetados.

### 3.3.2. Losses

En esta subsección se describe el cálculo de los losses del generador  $G$  y del discriminador  $D$ , los cuales permiten actualizar los parámetros de la red durante el entrenamiento.

Como fue establecido en la subsección 3.2.1, el discriminador  $D$  posee  $K + 1$  salidas: las primeras  $K$  corresponden a la probabilidad de clasificar una imagen en cada una de las clases reales, mientras que la salida  $K + 1$  indica la probabilidad de que la imagen haya sido generada por  $G$ . A partir de esta estructura, se definen las siguientes notaciones:

- $p(y|x)$ : probabilidad de que  $x$  pertenezca a la clase  $y$ .
- $D(x) = \sum_{y=1}^K p(y|x)$ : probabilidad de que  $x$  sea real, independientemente de la clase.
- $\phi$ : función de penalización aplicada a las probabilidades predichas.

Dado que  $D$  utiliza softmax en su salida, se cumple que la suma de todas las probabilidades es igual a uno. En consecuencia, siendo la salida  $K + 1$  la probabilidad de que  $x$  sea falso, se cumple la ecuación 3.1.

$$D(x) = 1 - p(y = K + 1|x) \quad (3.1)$$

En todos los losses se utilizará como función de penalización  $\phi(t) = -\log(t)$ . Cuando esta función se aplica en un problema de clasificación multi-clase, da lugar al cross entropy loss (CE), mientras que en el caso binario corresponde al binary cross entropy loss (BCE).

## Losses del generador

Para el entrenamiento de  $G$  se utilizan dos losses parciales que reflejan la calidad y la diversidad de las muestras generadas. Estos losses están inspirados en los índices de calidad y diversidad presentados en (Wang y cols., 2018). Los índices sobre los cuales se construyen dichos losses son:

1. Índice de calidad ( $F_q$ ): evalúa qué tan realistas son las imágenes generadas según  $D$ . El resultado esperado por  $G$  para estas imágenes es que  $D$  las clasifique como verdaderas, sin importar en qué clases específicas tuvo mayores o menores probabilidades. Según el artículo,  $F_q$  se expresa mediante la ecuación 3.2, donde  $D(G(z))$  representa la probabilidad de que  $D$  haya clasificado como verdaderas las imágenes producidas por  $G$ .

$$F_q = \mathbb{E}_{z \sim p_z} [D(G(z))] \quad (3.2)$$

2. Índice de diversidad ( $F_d$ ): mide qué tan variadas son las imágenes generadas y qué tan confiable es el discriminador sobre estas. Este índice está formado por tres componentes principales:

- La probabilidad de que  $D$  clasifique como reales las imágenes reales ( $D(x)$ ).
- La probabilidad de que  $D$  clasifique como falsas imágenes generadas ( $1 - D(G(z))$ ).
- La magnitud del gradiente de  $D$  respecto a las imágenes generadas ( $\nabla D$ ).

Con esta notación en cuenta, el fitness de diversidad ( $F_d$ ) propuesto en el artículo se presenta en la ecuación 3.3.

$$F_d = -\log \|\nabla D - \mathbb{E}_{x \sim p_{\text{datos}}} [\log D(x)] - \mathbb{E}_{z \sim p_z} [\log(1 - D(G(z)))]\| \quad (3.3)$$

Si se toma en cuenta que el loss estándar de  $D$  es el presentado en la ecuación 3.4, entonces al sustituir en la ecuación 3.3, se obtiene la ecuación 3.5, la cual indica que  $F_d$  depende directamente de  $\nabla D$  y del loss estándar  $L_D$ .

$$L_D = -[\mathbb{E}_{x \sim p_{\text{datos}}} \log D(x) + \mathbb{E}_{z \sim p_z} \log(1 - D(G(z)))] \quad (3.4)$$

$$F_d = -\log \|\nabla D + L_D\| \quad (3.5)$$

La magnitud del gradiente de  $D$  ( $\nabla D$ ) representa la intensidad de las actualizaciones necesarias en los parámetros del discriminador. Un valor

bajo de  $\nabla D$  luego de clasificar las imágenes generadas por  $G$  indica que  $D$  está seguro en sus clasificaciones, sin requerir grandes ajustes.

En la métrica  $F_d$ , este valor se combina con el loss estándar de  $D$  ( $L_D$ ), que mide qué tan bien distingue entre imágenes reales y generadas. Cuanto  $L_D$  es bajo, significa que  $D$  clasifica correctamente con confianza. Si, además,  $\nabla D$  es pequeño, el término  $\nabla D + L_D$  resulta reducido, lo que al ser invertido por el logaritmo eleva el valor de  $F_d$ .

Por lo tanto, un valor elevado de  $F_d$  indica que el discriminador no requiere actualizaciones intensivas, lo que sugiere que el generador está produciendo muestras suficientemente variadas y no colapsadas. En este sentido,  $F_d$  funciona como un indicador indirecto de diversidad en las imágenes generadas por  $G$ .

Con ambos índices parciales definidos ( $F_q$  y  $F_d$ ), el índice total de  $G$  se calcula utilizando la ecuación 3.6, donde  $\gamma \geq 0$  controla la importancia relativa del fitness de diversidad frente al fitness de calidad.

$$F = F_q + \gamma F_d, \quad (3.6)$$

Como fue mencionado anteriormente en esta sección, los valores  $F_q$  y  $F_d$  son índices de desempeño de  $G$ : un mayor índice indica mejor desempeño. Sin embargo, para establecer un loss para  $G$ , estos índices deben manipularse de modo que un mayor índice se convierta en un menor loss para  $G$ , asegurando que sea premiado cuando genera imágenes de alta calidad y diversidad.

En este trabajo, para representar  $F_q$ , se implementó un loss de calidad denominado  $L_{calidad}$ , que se ajusta más a los típicos losses utilizados para  $G$ , agregando una función de penalización  $\phi$ . Matemáticamente, este loss se expresa mediante la ecuación 3.7. Al reemplazar  $D(G(z))$  y  $\phi$  por sus equivalentes, se obtiene la ecuación 3.8.

$$L_{calidad} = \mathbb{E}_{z \sim p_z} [\phi(D(G(z)))] \quad (3.7)$$

$$L_{calidad} = -\mathbb{E}_{z \sim p_z} [\log (1 - p(y = K + 1 | G(z)))] \quad (3.8)$$

Este loss equivale a un BCE dado que se calcula sobre una sola probabilidad. El mismo simula el opuesto de  $F_q$ , por lo que es menor cuando  $D$  clasifica las muestras generadas como reales y viceversa.

En el caso de  $F_d$ , se utilizó como loss de diversidad ( $L_{diversidad}$ ) directamente el opuesto del índice asociado, como se presenta en la ecuación 3.9.

$$L_{diversidad} = -F_d \quad (3.9)$$

Finalmente, el loss total del generador ( $L_G$ ) utilizado en el entrenamiento simula el comportamiento de  $F$ , utilizando un coeficiente  $\gamma$  para el loss de diversidad. Este loss se presenta en la ecuación 3.10.

$$L_G = L_{calidad} + \gamma L_{diversidad}, \quad (3.10)$$

## Losses del discriminador

El loss total utilizado para  $D$ , denotado  $L_D$ , también consta de dos losses parciales, un loss supervisado y un loss no supervisado, como es sugerido en (Salimans y cols., 2016). El loss total  $L_D$  es la suma de ambos losses parciales, como se presenta en la ecuación 3.11.

$$L_D = L_{\text{supervisado}} + L_{\text{no-supervisado}} \quad (3.11)$$

Los losses parciales de  $D$  son los siguientes:

- Loss supervisado: se calcula únicamente sobre los datos reales etiquetados, considerando solo las clases reales ( $K$  clases), sin incluir la clase extra de “falso” ( $K + 1$ ). Este loss se expresa en la ecuación 3.12, la cual es equivalente, reemplazando  $\phi$ , a la ecuación 3.13.

$$L_{\text{supervisado}} = \mathbb{E}_{(x,y) \sim p_{\text{datos}}} [\phi(p(y | x; y < K + 1))] \quad (3.12)$$

$$L_{\text{supervisado}} = -\mathbb{E}_{(x,y) \sim p_{\text{datos}}} [\log(p(y | x; y < K + 1))] \quad (3.13)$$

Este loss representa el cross entropy loss (CE), dado que se comparan los valores de todas las clases reales. El mismo se calcula utilizando como valor obtenido el vector de probabilidades de la salida, omitiendo la entrada  $K + 1$ , y como salida esperada un vector one-hot suavizado, utilizando 0.9 en la clase correcta y 0 en el resto. El valor de 0.9 se debe al uso de la técnica de label smoothing (presentada en la sección 2.6), la cual tiene como objetivo reducir el sobreajuste y aumentar la generalización del modelo.

El valor de  $L_{\text{supervisado}}$  será menor cuando  $D$  este clasificando los datos etiquetados correctamente en sus clases, y viceversa.

- Loss no supervisado: El loss no supervisado se obtiene como la suma de dos losses diferentes, el loss no etiquetado ( $L_{\text{no-etiquetado}}$ ) y el loss falso ( $L_{\text{falso}}$ ), como se presenta en la ecuación 3.14.

$$L_{\text{no-supervisado}} = L_{\text{no-etiquetado}} + L_{\text{falso}} \quad (3.14)$$

1. Loss no etiquetado: Surge de la clasificación de imágenes reales no etiquetadas. Se espera que  $D$  clasifique las mismas como reales, sin necesidad de asignar una clase concreta. Este loss se expresa mediante la ecuación 3.15, la cual es equivalente a la ecuación 3.16, tras reemplazar  $\phi$  y  $D(x)$ .

$$L_{\text{no-etiquetado}} = \mathbb{E}_{x \sim p_{\text{datos}}} [\phi(D(x))] \quad (3.15)$$

$$L_{\text{no-etiquetado}} = -\mathbb{E}_{x \sim p_{\text{datos}}} [\log(1 - p(y = K + 1 | x))] \quad (3.16)$$

Este loss es un BCE dado que solo se utiliza la probabilidad de falsoedad. Para su cálculo se utiliza como valor obtenido  $1 - p(y = K+1|x)$ , que representa la probabilidad de que  $D$  haya clasificado la entrada en cualquier clase real. Nuevamente se utiliza como valor esperado 0.9 (debido a label smoothing).

2. Loss falso: Surge de la clasificación de imágenes generadas por  $G$ . Se espera que  $D$  clasifique estas imágenes como falsas, es decir, que la salida  $K + 1$  sea lo más alta posible. Este loss se expresa mediante la ecuación 3.17, la cual, al reemplazar  $\phi$  y  $D(G(z))$  por sus equivalentes, da lugar a la ecuación 3.18.

$$L_{\text{falso}} = \mathbb{E}_{z \sim p_z} [\phi(1 - D(G(z)))] \quad (3.17)$$

$$L_{\text{falso}} = -\mathbb{E}_{z \sim p_z} [\log(p(y = K + 1 | G(z)))] \quad (3.18)$$

Este loss es equivalente al BCE, utilizando como salida únicamente la probabilidad de falsoedad y como valor esperado 0.9 (debido al uso label smoothing).

### 3.3.3. Algoritmo de entrenamiento

El entrenamiento de la GAN se desarrolla a lo largo de varias épocas, repitiendo el mismo procedimiento en cada una de ellas. En cada época, los datos de entrenamiento se dividen en lotes (batches) de igual tamaño, lo que determina la cantidad de iteraciones por época. En cada iteración se calculan los losses de cada red y se actualizan sus parámetros.

En el algoritmo 2 se desarrolla el proceso de entrenamiento de la GAN utilizando todos los datos etiquetados en cada iteración y generando los batches a partir de los datos no etiquetados (como fue presentado en la subsección 3.3.1).

El proceso de entrenar a  $D$  y  $G$  se repite durante todas las épocas e iteraciones. Finalizado el entrenamiento, se evalúa el modelo sobre el conjunto de test, que contiene 10000 datos etiquetados, calculando las métricas establecidas para cada red.

---

**Algoritmo 2** Algoritmo de entrenamiento semisupervisado propuesto para la GAN

---

**Parámetros:**  $E$ : número de épocas  $B$ : tamaño del batch  
 $N$ : cantidad de datos no etiquetados  $G$ : Generador  
 $D$ : Discriminador

```
1: para  $i = 1 \dots E$  hacer
2:   para  $j = 1 \dots (N/B)$  hacer
3:     Cálculo del loss del discriminador
4:     Tomar todos los datos etiquetados
5:     Calcular  $L_{\text{supervisado}}$  (ecuación 3.13)
6:     Tomar  $B$  datos no etiquetados
7:     Calcular  $L_{\text{no\_etiquetado}}$  (ecuación 3.16)
8:     Generar un batch de  $B$  vectores de ruido
9:     Generar  $B$  imágenes falsas
10:    Calcular  $L_{\text{falso}}$  (ecuación 3.18)
11:     $L_D = L_{\text{supervisado}} + L_{\text{no\_supervisado}}$ 
12:    Realizar el descenso por gradiente sobre los parámetros de  $D$  con  $L_D$ 
13:    Cálculo del loss del generador
14:    Obtener  $B$  vectores de ruido
15:    Generar  $B$  imágenes falsas
16:    Calcular  $L_{\text{calidad}}$  (ecuación 3.8)
17:    Calcular  $L_{\text{diversidad}}$  (ecuación 3.9)
18:     $L_G = L_{\text{calidad}} + \gamma L_{\text{diversidad}}$ 
19:    Realizar el descenso por gradiente sobre los parámetros de  $G$  con  $L_G$ 
20:  fin para
21: fin para
```

---



## Capítulo 4

# GAN con AE mono objetivo

En el capítulo 3 se presentó la implementación de una GAN convolucional compuesta por un generador y un discriminador, entrenada con un conjunto reducido de datos etiquetados.

En este capítulo se propone una extensión de la arquitectura de la GAN mediante la integración con AE. A diferencia del esquema tradicional, que emplea un único par generador-discriminador, en este capítulo se plantea un enfoque coevolutivo basado en dos poblaciones independientes: una de generadores, denotada  $P_G$ , y otra de discriminadores, denotada  $P_D$ . Cada población está conformada por  $P$  individuos y tiene como objetivo optimizar una única función de fitness (mono objetivo). Este enfoque busca fomentar la exploración de soluciones diversas y, al mismo tiempo, mejorar el desempeño global mediante operadores evolutivos.

### 4.1. Esquema general del algoritmo

Como fue descrito en la sección 2.8, un AE típico comprende los operadores evolutivos de inicialización, selección, cruzamiento, mutación y reemplazo. El algoritmo propuesto en este trabajo conserva la mayoría de estos operadores, con la particularidad de que no se aplica el operador de mutación por simplicidad.

El proceso evolutivo se organiza en generaciones sucesivas de individuos. En cada generación, se seleccionan  $K$  individuos de cada población ( $P_G$  y  $P_D$ ), sobre los cuales se generan copias (descendientes). Posteriormente, los generadores descendientes se emparejan aleatoriamente con los discriminadores descendientes, y cada pareja se entrena independientemente durante un número fijo de épocas.

Al finalizar el entrenamiento, cada población contiene  $P + K$  individuos: los  $P$  individuos originales (padres) más los  $K$  descendientes entrenados. A continuación, se aplica un algoritmo de reemplazo que evalúa el fitness de todos los

individuos, sin distinción entre padres y descendientes, reduciendo nuevamente el tamaño de cada población a  $P$  individuos, que representan la nueva generación de individuos. Este ciclo se repite hasta alcanzar el criterio de parada, definido en este trabajo como un número fijo de generaciones.

El esquema general implementado para el AE mono objetivo se presenta en el algoritmo 3.

---

**Algoritmo 3** Esquema reducido del AE mono objetivo

---

**Parámetros:**  $E$ : número de épocas por generación  
 $I$ : número total de generaciones

**Inicialización**

- 1: Inicializar  $P_G$  aleatoriamente ( $P$  generadores)
- 2: Inicializar  $P_D$  aleatoriamente ( $P$  discriminadores)
- 3: **para**  $i = 1 \dots I$  **hacer**

**Selección**

- 4: Seleccionar  $K$  individuos de  $P_G$  y  $P_D$

**Cruzamiento**

- 5: Clonar los individuos seleccionados para generar  $K$  descendientes en cada población ( $P + K$  individuos por población)

- 6: Formar  $K$  parejas generador–discriminador a partir de los descendientes

- 7: Entrenar cada pareja durante  $E$  épocas

**Reemplazo**

- 8: Evaluar el fitness de cada individuo

- 9: Reducir cada población a  $P$  individuos con el algoritmo de reemplazo

- 10: **fin para**
- 

## 4.2. Evaluación de individuos

La aplicación de un AE requiere definir una función de fitness para cada población que permita evaluar la aptitud de sus individuos. En el caso mono objetivo, se tomó la decisión de construir estas funciones partiendo de los losses utilizados durante el entrenamiento, lo cual resulta intuitivo dado que miden el desempeño de cada red. En consecuencia, el objetivo del AE será minimizar estas funciones de fitness.

Dado que los generadores y discriminadores se entranan de a pares, el loss de un individuo depende del adversario enfrentado, lo que introduce un sesgo: una misma red podría obtener un desempeño distinto al ser evaluada frente a adversarios de diferente capacidad. Por ello, las funciones de fitness implementadas no corresponden directamente a los losses, sino que se definen a partir de ellos. Para evitar sesgos, el fitness de cada red se define considerando los losses obtenidos al enfrentar a cada uno de los individuos de la población opuesta, garantizando una evaluación más estable y menos sensible a adversarios individuales.

## Fitness de los generadores

Para cada generador  $G$ , el loss total ( $L_G$ ) y sus componentes parciales ( $L_{calidad}$  y  $L_{diversidad}$ ) dependen del discriminador enfrentado. Si se usara como fitness el valor de  $L_G$  obtenido del entrenamiento contra un único discriminador, el mismo resultaría dependiente del adversario específico. En la solución propuesta, el fitness de cada generador (denotado  $F_G$ ) se calcula como el promedio de los losses obtenidos al evaluar dicho generador contra todos los discriminadores de la población  $P_D$ .

Si bien este procedimiento implica un mayor costo computacional que evaluar frente a un único adversario, solo se realiza en operadores puntuales del ciclo evolutivo (selección y reemplazo), por lo que el impacto computacional es poco significativo.

En la figura 4.1 se ilustra el esquema de obtención de los múltiples losses para un generador  $G_i$ , donde  $L_G(G_i, D_j)$  denota el loss total de dicho generador al enfrentarse al discriminador  $D_j$ .

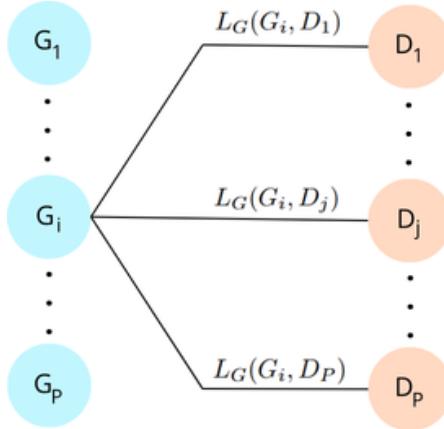


Figura 4.1: Cálculo de los distintos  $L_G$  al enfrentar un generador  $G_i$  con todos los discriminadores  $D_j$  de la población  $P_D$  para el AE mono objetivo.

Sea  $|P_D|$  el número de discriminadores en la población  $P_D$ , entonces el fitness de un generador  $G_i$  se define en la ecuación 4.1.

$$F_G(G_i) = \frac{1}{|P_D|} \left( \sum_{j=1}^{|P_D|} L_G(G_i, D_j) \right) \quad (4.1)$$

## Fitness de los discriminadores

El loss total de un discriminador ( $L_D$ ) se compone de tres términos:  $L_{supervisado}$ ,  $L_{no\_etiquetado}$  y  $L_{falso}$ . Cada componente presenta diferentes dependencias que deben mitigarse para definir una función de fitness objetiva:

- $L_{supervisado}$ : depende de las imágenes etiquetadas utilizadas como entrada.
- $L_{no\_etiquetado}$ : depende de las imágenes no etiquetadas utilizadas como entrada.
- $L_{falso}$ : depende directamente del generador enfrentado.

Por lo tanto, para construir una función de fitness objetiva para  $D$  es necesario controlar estas dependencias.

En el caso de  $L_{supervisado}$ , su cálculo se realiza siempre utilizando el conjunto completo de datos etiquetados, garantizando condiciones equivalentes para cada discriminador de  $P_D$ .

El cálculo de  $L_{no\_etiquetado}$  se efectúa empleando el mismo batch de datos no etiquetados para cada discriminador de  $P_D$ .

Finalmente, se calcula un  $F_{falso}$  promediando los valores de  $L_{falso}$  obtenidos al enfrentar al discriminador con todos los generadores de  $P_G$ . Este procedimiento es análogo al utilizado para el cálculo de fitness de los generadores (presentado en la sección 4.2) y garantiza que este componente del fitness sea independiente del adversario específico. El cálculo de  $F_{falso}$  para un discriminador específico  $D_i$  al enfrentar todos los generadores  $G_j$  pertenecientes a  $P_G$  se presenta en la ecuación 4.2, siendo  $L_{falso}(G_j, D_i)$  el loss calculado mediante la ecuación 3.18.

$$F_{falso}(D_i) = \frac{1}{|P_G|} \left( \sum_{j=1}^{|P_G|} L_{falso}(G_j, D_i) \right) \quad (4.2)$$

A diferencia de  $L_{falso}$ , los componentes  $L_{supervisado}$  y  $L_{no\_etiquetado}$  son calculados directamente sobre el discriminador evaluado, sin promediación, dado que no dependen del generador adversario sino exclusivamente de las imágenes reales (etiquetadas y no etiquetadas) empleadas. En consecuencia, la función de fitness utilizada para un discriminador  $D_i$ , denotada  $F_D$ , se presenta en la ecuación 4.3.

$$F_D(D_i) = L_{supervisado}(D_i) + L_{no\_etiquetado}(D_i) + F_{falso}(D_i) \quad (4.3)$$

### 4.3. Algoritmo evolutivo planteado

En esta sección se profundiza en aspectos técnicos del AE mono objetivo implementado. Se detallará el algoritmo completo, incluyendo la inicialización de las poblaciones y los operadores de selección, cruzamiento y reemplazo.

#### 4.3.1. Inicialización

Para el correcto funcionamiento del algoritmo, es necesario ejecutar un proceso de inicialización previo al bloque evolutivo. Sea  $P$  el tamaño inicial de cada

población, entonces el procedimiento inicia generando aleatoriamente  $P$  generadores y  $P$  discriminadores que conforman las poblaciones iniciales  $P_G$  y  $P_D$ , respectivamente. Esta población inicial es sometida a cruzamientos aleatorios, donde se empareja cada uno de los  $P$  generadores con cada uno de los  $P$  discriminadores. Luego, cada pareja se entrena durante un número fijo de épocas. Este proceso permite inicializar los valores de fitness de cada red siguiendo el procedimiento propuesto en la sección 4.2.

Debido a que en el proceso de inicialización todas las redes de cada población son entrenadas, en este proyecto se buscó minimizar la cantidad de épocas de inicialización utilizadas en el AE con el objetivo de reducir los tiempos de cómputo. Además, incluso utilizando valores bajos de épocas de inicialización, las redes lograron generar valores de fitness representativos dada la rápida convergencia que presentan las redes neuronales en sus primeras épocas de entrenamiento.

### 4.3.2. Operador de selección

La selección es el primer operador aplicado una vez inicializado el algoritmo. En este proyecto, se decidió implementar un operador de selección similar a la selección por torneo (presentada en la sección 2.8.1), el cual es aplicado sobre cada una de las poblaciones con el objetivo de seleccionar  $K$  de los  $P$  individuos para su posterior cruzamiento.

El algoritmo de selección planteado se basa en utilizar aleatoriedad y elitismo al igual que la selección por torneo. El mismo consiste en seleccionar  $M \leq P$  individuos aleatoriamente y quedarse con los  $K$  individuos con mejor valor de fitness, siendo  $K \leq M$ .

Los valores de  $M$  y  $K$  serán parámetros a configurar, permitiendo que el torneo sea más aleatorio o más elitista según estos valores. En casos límite, si  $M = K$ , la selección es totalmente aleatoria, mientras que si  $M = P$ , la selección es completamente elitista.

### 4.3.3. Operador de cruzamiento

En la etapa de cruzamiento, se trabaja sobre los individuos obtenidos del proceso de selección (padres), generando nuevos descendientes.

Al finalizar el proceso de selección, se obtienen  $K$  individuos padres por población, los cuales son clonados para generar  $K$  descendientes. Inicialmente, los descendientes son réplicas exactas de sus padres. Los  $K$  generadores descendientes y los  $K$  discriminadores descendientes son emparejados aleatoriamente generando  $K$  parejas de generador-discriminador. Estas parejas son entrenadas de forma independiente durante cierta cantidad de épocas, siguiendo el mismo algoritmo de entrenamiento utilizado en el enfoque sin AE (siguiendo el algoritmo 2). Durante el entrenamiento, las redes actualizan sus parámetros en función de los errores cometidos contra el adversario seleccionado en el emparejamiento, de modo que sus parámetros difieren de los de sus padres y dejan de ser copias exactas.

Este procedimiento permite que un generador sea entrenado frente a cualquier discriminador. En caso de que el discriminador presente un desempeño bajo, el generador podrá adaptarse y superar fácilmente su capacidad de clasificación. En una siguiente generación,  $G$  podría enfrentarse a un discriminador mejor y verse forzado a mejorar. El proceso es análogo para los discriminadores. En consecuencia, este proceso permite que la red cambie su rival durante el entrenamiento, generando un entorno más competitivo para todas las redes.

#### 4.3.4. Algoritmo de reemplazo

Luego de finalizado el entrenamiento de las  $K$  parejas de redes descendientes, se tienen dos poblaciones de  $P + K$  individuos, compuestas por los padres y los descendientes, las cuales deben ser reducidas nuevamente a  $P$  individuos mediante el algoritmo de reemplazo.

Antes de aplicar el algoritmo de reemplazo, se re-calculan los valores de fitness de todos los individuos, de modo que cada uno se enfrente a los  $P + K$  adversarios posibles, siguiendo el procedimiento descrito en la sección 4.2. Esta etapa asegura que los valores de fitness reflejen el desempeño frente a todos los rivales disponibles en la población ampliada.

En este punto, se debe aplicar un algoritmo de reemplazo que permita definir cuáles serán los individuos que conformarán cada población en la próxima generación. Para este proyecto se decidió utilizar el algoritmo de reemplazo elitista (introducido en la sección 2.8.3), dada su simplicidad y efectividad en la práctica. El mismo consiste en seleccionar, entre los  $P + K$  individuos, los  $P$  con mejor valor de fitness.

#### 4.3.5. Algoritmo evolutivo completo

En esta subsección se presenta el algoritmo implementado en detalle, agrupando todo contenido explicado previamente durante este capítulo. Se incluye la estructura iterativa del AE, al igual los operadores evolutivos desarrollados. El pseudo-código del mismo se presenta en el algoritmo 4.

Al finalizar el entrenamiento, el algoritmo retorna dos poblaciones finales que idealmente estarán compuestas por los individuos más aptos, los cuales serán evaluados según métricas específicas para cada población.

---

**Algoritmo 4** Algoritmo evolutivo mono objetivo

---

**Parámetros:**  $E_G$ : número de épocas por generación

$E_I$ : número de épocas de inicialización

$I$ : número total de generaciones

$P$ : tamaño inicial de la población

$M$ : cantidad de individuos seleccionados aleatoriamente en la selección

$K$ : cantidad de parejas entrenadas simultáneamente

**Inicialización**

- 1: Inicializar  $P_G$  con  $P$  generadores aleatorios
- 2: Inicializar  $P_D$  con  $P$  discriminadores aleatorios
- 3: Emparejar generadores y discriminadores de forma aleatoria
- 4: **para**  $(G, D)$  in parejas **hacer**
- 5:     Entrenar  $G$  y  $D$  durante  $E_I$  épocas
- 6: **fin para**
- 7: Actualizar fitness de cada red

**Bloque evolutivo**

- 8: **para**  $i = 1 \dots I$  **hacer**
  - 9:     **Selección**
  - 10:     Seleccionar  $M \leq P$  generadores y discriminadores aleatoriamente
  - 11:     Padres  $\leftarrow$  seleccionar  $K \leq M$  generadores y discriminadores por elitismo
  - 12:     **Cruzamiento**
  - 13:     Clonar padres para generar  $K$  descendientes
  - 14:     Emparejar aleatoriamente los  $K$  descendientes
  - 15:     **para**  $(G, D)$  in parejas **hacer**
  - 16:         Entrenar  $G$  y  $D$  durante  $E_G$  épocas
  - 17:     **fin para**
  - 18:     **Reemplazo**
  - 19:     Actualizar fitness de todas las redes
  - 20:     Reducir las poblaciones a  $P$  individuos mediante elitismo
- 18: **fin para**
  - 19: Evaluar las redes obtenidas en  $P_G$  y  $P_D$
-



## Capítulo 5

# GAN con AE multi objetivo

El capítulo anterior presentó un enfoque que combina GANs con AE, obteniendo una arquitectura capaz de entrenarse de forma semisupervisada sobre el dataset MNIST utilizando una cantidad reducida de datos etiquetados. Dicha arquitectura, compuesta por dos poblaciones, una de generadores ( $P_G$ ) y otra de discriminadores ( $P_D$ ), tuvo como objetivo optimizar una única función de fitness por población (enfoque mono objetivo).

En este capítulo se presenta la extensión del AE presentado en el capítulo 4 para optimizar múltiples funciones de fitness en paralelo (enfoque multi objetivo). En particular, cada población optimiza dos funciones objetivo simultáneamente.

Si bien la estructura general del algoritmo es idéntica en ambos casos, existen diferencias importantes en el cálculo del fitness y en los operadores de selección y reemplazo, las cuales se detallan a lo largo de este capítulo.

### 5.1. Evaluación de individuos

El AE multi objetivo mantiene las dos poblaciones  $P_G$  y  $P_D$ , pero cada una optimiza ahora dos funciones de fitness en lugar de una. En esta sección se presentan dichas funciones, las cuales se definen a partir de los losses parciales utilizados durante el entrenamiento de las redes.

#### 5.1.1. Fitness de los generadores

En el caso de los generadores, los dos objetivos se inspiran en los losses parciales que componen el loss total  $L_G$ : el loss de calidad ( $L_{calidad}$ ) y el loss de diversidad ( $L_{diversidad}$ ), previamente introducidos en la sección 3.3.2.

Siguiendo la lógica del AE mono objetivo (sección 4.2), cada loss parcial da lugar a una función de fitness, calculada promediando los valores obtenidos al enfrentar un generador con todos los discriminadores de la población  $P_D$ . Esto garantiza que el fitness no dependa de un adversario en particular.

Los cálculos del fitness de calidad ( $F_{calidad}$ ) y del fitness de diversidad ( $F_{diversidad}$ ) se muestran en las ecuaciones 5.1 y 5.2, donde  $L_{calidad}(G_i, D_j)$  y  $L_{diversidad}(G_i, D_j)$  se calculan según las ecuaciones 3.8 y 3.9, respectivamente.

$$F_{calidad}(G_i) = \frac{1}{|P_D|} \left( \sum_{j=1}^{|P_D|} L_{calidad}(G_i, D_j) \right) \quad (5.1)$$

$$F_{diversidad}(G_i) = \frac{1}{|P_D|} \left( \sum_{j=1}^{|P_D|} L_{diversidad}(G_i, D_j) \right) \quad (5.2)$$

### 5.1.2. Fitness de los discriminadores

En los discriminadores se utilizaron dos funciones de fitness inspiradas en los dos losses parciales que componen el loss total  $L_D$ :  $L_{supervisado}$  y  $L_{no\_supervisado}$ , definidos en las ecuaciones 3.13 y 3.14, respectivamente.

La función de fitness asociada al componente supervisado fue directamente el valor de  $L_{supervisado}$ , siendo este calculado utilizando todos los datos etiquetados disponibles, lo que asegura igualdad de condiciones en las evaluaciones por parte de los distintos discriminadores.

Por otra parte, la segunda función de fitness, denominada  $F_{no\_supervisado}$ , se basó en el componente no supervisado, el cual se compone de dos términos parciales:  $L_{no\_etiquetado}$  y  $L_{falso}$ . Respecto a  $L_{no\_etiquetado}$ , este fue utilizado directamente como componente parcial del fitness, siendo calculado siempre a partir del mismo batch de datos no etiquetados para asegurar comparabilidad. Además, se construyó un  $F_{falso}$  como componente parcial de  $F_{no\_supervisado}$ , el cual promedia los  $L_{falso}$  obtenidos al enfrentar un discriminador con todos los generadores de  $P_G$ , tal como se definió en la ecuación 4.2. En consecuencia, el fitness no supervisado  $F_{no\_supervisado}$  sigue la ecuación 5.3.

$$F_{no\_supervisado}(D_i) = L_{no\_etiquetado}(D_i) + F_{falso}(D_i) \quad (5.3)$$

De este modo, los dos objetivos a optimizar por cada discriminador son  $L_{supervisado}$  y  $F_{no\_supervisado}$ , que reflejan su desempeño sobre datos etiquetados y no etiquetados (reales y generados), respectivamente.

## 5.2. Algoritmo evolutivo planteado

El AE multi objetivo propuesto se basa en la misma estructura del AE mono objetivo (presentado en la sección 4.3), manteniendo los procedimientos de inicialización y cruzamiento. La principal diferencia radica en los métodos de selección y reemplazo, ya que en el enfoque multi objetivo se deben considerar dos funciones de fitness por población.

### 5.2.1. Selección y reemplazo

En el AE multi objetivo se mantienen los mismos procesos de selección y reemplazo que en el algoritmo mono objetivo, con la diferencia de que los mejores individuos se determinan ahora mediante un criterio multi objetivo.

El algoritmo de selección es el mismo que fue presentado en la subsección 4.3.2, donde se toman  $M \leq P$  individuos aleatoriamente y se conservan los  $K \leq M$  mejores por elitismo. El algoritmo de reemplazo es el algoritmo de reemplazo elitista.

La diferencia en estos operadores surge a la hora de seleccionar los mejores individuos dado que se poseen dos funciones objetivo por población. Por ello, para seleccionar los mejores individuos se utiliza el algoritmo non-dominated sorting genetic algorithm II (NSGA-II), introducido en la subsección 2.8.4. Este algoritmo organiza a los individuos en frentes de Pareto, priorizando aquellos no dominados. En caso de empate dentro de un mismo frente, se utiliza la métrica de crowding distance para favorecer la diversidad poblacional. De este modo, se garantiza que las soluciones seleccionadas representen un compromiso entre los distintos objetivos optimizados.

A modo de síntesis, los algoritmos de selección y reemplazo son equivalentes a los del esquema mono objetivo, con la salvedad de que la elección de los mejores individuos se realiza mediante NSGA-II, considerando de manera simultánea los dos objetivos.

### 5.2.2. Algoritmo evolutivo completo

El algoritmo multi objetivo se construye a partir del AE mono objetivo (algoritmo 4), reemplazando los operadores de selección y reemplazo por versiones basadas en NSGA-II. Además, cada individuo es evaluado ahora sobre dos funciones de fitness (definidas en la sección 5.1). El pseudo-código completo se presenta en el Algoritmo 5.

El AE multi objetivo constituye una extensión natural del esquema mono objetivo, al permitir la optimización simultánea de múltiples criterios en cada población. La integración de NSGA-II introduce un balance entre explotación y exploración, ya que preserva tanto a los individuos más aptos como a aquellos que aportan diversidad en los frentes de Pareto.

De este modo, la población de generadores  $P_G$  tiende a contener individuos que alcanzan alta calidad, alta diversidad, o un equilibrio entre ambos, mientras que la población de discriminadores  $P_D$  conserva individuos con buen desempeño sobre datos etiquetados, no etiquetados o en ambos escenarios. En síntesis, este esquema ofrece un marco más flexible y robusto para el entrenamiento coevolutivo de GANs mediante AE, permitiendo evaluar y optimizar de manera independiente múltiples funciones de fitness en cada población.

---

**Algoritmo 5** Algoritmo evolutivo multi objetivo

---

**Parámetros:**  $E_G$ : número de épocas por generación  
 $E_I$ : número de épocas de inicialización  
 $I$ : número total de generaciones  
 $P$ : tamaño inicial de la población  
 $M$ : cantidad de individuos seleccionados aleatoriamente en la selección  
 $K$ : cantidad de parejas entrenadas simultáneamente

**Inicialización**

- 1: Inicializar  $P_G$  con  $P$  generadores aleatorios
- 2: Inicializar  $P_D$  con  $P$  discriminadores aleatorios
- 3: Emparejar generadores y discriminadores de forma aleatoria
- 4: **para**  $(G, D)$  in parejas **hacer**
- 5:     Entrenar  $G$  y  $D$  durante  $E_I$  épocas
- 6: **fin para**
- 7: Actualizar fitness de cada red

**Bloque evolutivo**

- 8: **para**  $i = 1 \dots I$  **hacer**
- 9:     **Selección**
- 10:     Seleccionar  $M \leq P$  generadores y discriminadores aleatoriamente
- 11:     Padres  $\leftarrow$  seleccionar  $K \leq M$  generadores y discriminadores mediante NSGA-II

**Cruzamiento**

- 12:     Clonar padres para generar  $K$  descendientes
- 13:     Emparejar aleatoriamente los  $K$  descendientes
- 14:     **para**  $(G, D)$  in parejas **hacer**
- 15:         Entrenar  $G$  y  $D$  durante  $E_G$  épocas
- 16:     **fin para**

**Reemplazo**

- 17:     Actualizar fitness de todas las redes
- 18:     Reducir las poblaciones a  $P$  individuos mediante NSGA-II
- 19: **fin para**

- 19:     Evaluar las redes obtenidas en  $P_G$  y  $P_D$
-

# Capítulo 6

## Evaluación experimental

Este capítulo presenta la metodología aplicada y los resultados obtenidos durante la experimentación, incluyendo la configuración paramétrica y la validación final para cada enfoque estudiado. En la sección final se presenta un análisis comparativo de todos los enfoques, resaltando sus similitudes y diferencias. En todas las etapas relativas a la experimentación se reportan métricas estadísticas, imágenes generadas y gráficas que respaldan las conclusiones.

### 6.1. Metodología

En esta sección se describe la metodología empleada a lo largo del proyecto. En primer lugar, se llevó a cabo una revisión del estado del arte de las GANs, con énfasis en enfoques de entrenamiento semisupervisado. Asimismo, se estudiaron las tecnologías y bibliotecas disponibles para su implementación. Posteriormente, se analizaron los fundamentos de los AE y sus posibles integraciones con GANs. Finalmente, en la etapa de experimentación, se configuraron parámetros de cada algoritmo para optimizar su desempeño antes de la validación.

#### 6.1.1. Metodología para el análisis del estado del arte

El primer paso consistió en un estudio detallado de las redes neuronales, CNNs y GANs, con el objetivo de comprender en profundidad sus fundamentos teóricos y prácticos. A continuación, se revisaron trabajos académicos de relevancia en el área, analizando específicamente el estado del arte de las GANs en enfoques semisupervisados, técnicas propuestas y sus impactos en el rendimiento. Una vez implementada la GAN, se profundizó en el estudio de los AE, considerando las arquitecturas reportadas en la literatura y las alternativas disponibles para el diseño de operadores evolutivos, evaluando su complejidad y rendimiento. Finalmente, se analizaron artículos que integran AE con GANs, con el fin de identificar antecedentes y establecer comparaciones con los resultados obtenidos en este proyecto.

### 6.1.2. Metodología para evaluar las GANs

Para evaluar el desempeño de las GANs se emplearon dos métricas ampliamente aceptadas en la literatura: precisión y Fréchet Inception Distance (FID). Estas métricas fueron seleccionadas por su capacidad de medir aspectos complementarios: la precisión cuantifica el rendimiento del discriminador, mientras que FID refleja la calidad y diversidad de las imágenes generadas. Además, para la evaluación conjunta de precisión y FID sobre múltiples ejecuciones del algoritmo se utilizó la métrica de hipervolumen. En esta subsección se presentan detalladamente las métricas de precisión, FID e hipervolumen.

#### Precisión

La precisión mide la proporción de resultados exitosos ( $N_{exitosos}$ ) sobre el total de casos ( $N$ ), y se expresa como porcentaje de acuerdo con la ecuación 6.1.

$$\text{Precisión} = \frac{N_{exitosos}}{N} \times 100 \% \quad (6.1)$$

En este proyecto, la precisión se emplea para cuantificar la proporción de clasificaciones correctas realizadas por  $D$  sobre el total de casos evaluados.

#### FID

FID se calcula a partir de un conjunto de prueba (conformado por imágenes reales) y otro conjunto de imágenes generadas. Ambos conjuntos son procesados por una red preentrenada (Inception V3), de la cual se extraen vectores de características. Finalmente, FID se define como la distancia entre las distribuciones de características de ambos conjuntos. Por lo tanto, valores bajos de FID indican mejores resultados, ya que reflejan una mayor similitud entre imágenes reales y generadas.

#### Hipervolumen

El hipervolumen es una métrica ampliamente utilizada en problemas multiobjetivo para evaluar la calidad de un conjunto de soluciones en relación con un punto de referencia, resumiendo el desempeño de múltiples ejecuciones sobre varios objetivos en un solo índice de calidad. El punto de referencia, cuya dimensión coincide con la cantidad de funciones a optimizar, está compuesto por los peores valores posibles de cada función.

En este proyecto, el hipervolumen se utiliza como métrica central para comparar configuraciones en todos los enfoques (sin AE, AE mono objetivo y AE multi objetivo). El cálculo se realizó con los objetivos de maximizar la precisión y minimizar FID. De esta forma, el hipervolumen resume el desempeño conjunto de ambas métricas en un único valor de calidad, independientemente del enfoque empleado. Al tener dos objetivos, el cálculo del hipervolumen corresponde

al área comprendida entre el frente de Pareto de las soluciones y el punto de referencia, como se ilustra en la figura 6.1.

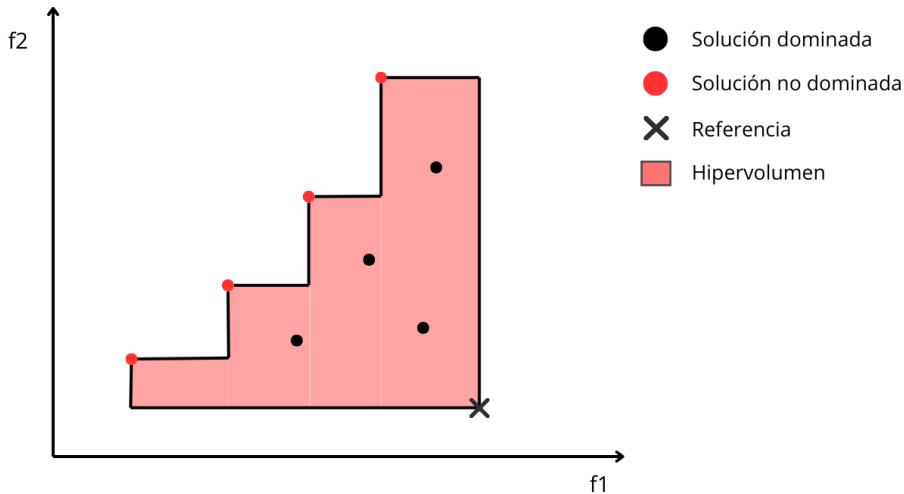


Figura 6.1: Hipervolumen de un conjunto de datos con el objetivo de minimizar  $f_1$  y maximizar  $f_2$

Dado que los valores de FID y precisión emplean escalas diferentes, en todos los casos se realizó una normalización de los ejes antes del cálculo del hipervolumen, evitando así otorgar mayor peso a una función sobre la otra.

### 6.1.3. Metodología para evaluar ejecuciones múltiples en la etapa de configuración

Debido a la naturaleza estocástica de las GANs y de los AEs, una única ejecución no resulta suficiente para evaluar el desempeño de forma confiable. Dado que intervienen factores aleatorios y un proceso de entrenamiento dinámico, los resultados pueden variar significativamente entre ejecuciones. Por lo tanto, la evaluación se realizó a partir de múltiples ejecuciones (detalladas en cada análisis específico) de cada configuración estudiada. Sobre todas las ejecuciones realizadas para una configuración, se calculó la métrica de hipervolumen. Finalmente, la configuración que obtuvo el mayor valor de hipervolumen fue seleccionada como configuración final.

### 6.1.4. Metodología para la configuración de los parámetros de la GAN

Con el fin de establecer una arquitectura final para la GAN, fue necesario configurar múltiples parámetros para optimizar su desempeño. En este proyecto se consideraron distintos escenarios con diferentes cantidades de datos

etiquetados. Para cada escenario, se estableció una configuración específica e independiente.

Para cada cantidad de datos etiquetados, los parámetros a configurar fueron los mismos cinco. Estos fueron distribuidos en tres etapas sucesivas:

- Configuración del modelo: Se evaluaron el coeficiente de dropout utilizado por  $D$  y el coeficiente de diversidad  $\gamma$  utilizado en  $L_G$  (presentado en la ecuación 3.10).
- Configuración del entrenamiento: Se analizaron el tamaño del batch y la utilización de batch normalization (presente o ausente).
- Configuración del uso de datos etiquetados: Se compararon los dos enfoques descritos en la subsección 3.3.1. El primero utiliza todos los datos etiquetados en cada iteración, mientras que el segundo los distribuye uniformemente a lo largo de las iteraciones de una época, reutilizando datos en caso de ser necesario.

Dada una cantidad fija de datos etiquetados y una configuración específica, se realizaron múltiples ejecuciones del algoritmo. Sobre estas ejecuciones se calculó el hipervolumen a partir de los valores de precisión y FID obtenidos. Para una cantidad fija de datos etiquetados, la configuración final seleccionada fue aquella que alcanzó el mayor valor de hipervolumen.

#### 6.1.5. Metodología para la configuración de los parámetros de los AE

En la etapa de configuración de los AE se respetaron los valores de los parámetros obtenidos para la GAN, cuya metodología de configuración fue descrita en la subsección anterior. En esta etapa el objetivo no fue modificar la arquitectura de la GAN ni sus parámetros, sino integrarla con los AE. Por lo tanto, únicamente se configuraron parámetros propios del AE. Dado el alto costo computacional del proceso y la cantidad de parámetros involucrados, se decidió trabajar únicamente con el escenario de 100 datos etiquetados, dado que fue el más restrictivo de los tres escenarios inicialmente planificados (100, 500 y 1000 datos).

Ambos algoritmos, el mono objetivo y el multi objetivo, comparten los mismos parámetros. Esto se debe a que sus diferencias radican en dos aspectos: en primer lugar, el multi objetivo emplea NSGA-II en lugar de elitismo; ambos enfoques no dependen de parámetros adicionales. En segundo lugar, uno optimiza una única función de fitness mientras que el otro optimiza dos, por lo que tampoco se tienen nuevos parámetros por configurar. A pesar de utilizar los mismos parámetros, y con el objetivo de obtener mejores desempeños en ambos algoritmos, la configuración se realizó por separado, obteniendo una configuración específica para el AE mono objetivo y otra para el multi objetivo.

La configuración paramétrica de los AE fue separada en dos etapas secuenciales dada la cantidad de parámetros a configurar. Estas etapas fueron:

- Configuración de  $P$ ,  $M$  y  $K$ : Se analizaron el tamaño inicial de la población ( $P$ ), la cantidad de individuos seleccionados aleatoriamente en el proceso de selección ( $M$ ) y el tamaño del resultado del torneo ( $K$ ), que determina cuántos individuos participarán en el cruzamiento.
- Configuración de épocas: Se analizó la cantidad de épocas de entrenamiento por generación. Para este análisis se utilizó un número de épocas totales que se mantiene entre todas las configuraciones de modo que ninguna configuración obtenga ventaja sobre otra.

En ambos algoritmos, el criterio para seleccionar la configuración final fue el valor de hipervolumen obtenido a partir de las múltiples ejecuciones de cada configuración. Si bien en los AE una única ejecución del algoritmo tiene como resultado una población final de generadores y otra de discriminadores, para el cálculo de hipervolumen se utilizó, por cada ejecución, la pareja de los mejores valores de FID y precisión obtenidos entre todos los individuos finales.

#### 6.1.6. Metodología para la validación de resultados

La etapa de validación tuvo como objetivo evaluar el desempeño de los algoritmos desarrollados mediante entrenamientos más extensos, utilizando los valores de los parámetros obtenidos en la etapa de configuración paramétrica. Este proceso permitió comprobar la estabilidad de los resultados, comparar los distintos enfoques propuestos y analizar el impacto de la cantidad de datos etiquetados disponibles. Aunque inicialmente se planificó trabajar con 100, 500 y 1000 datos etiquetados (al igual que en la etapa de configuración), para la etapa de validación se incorporó un escenario adicional, aún más restrictivo, con solo 60 datos. Este escenario fue incorporado con el objetivo de estudiar el desempeño de los distintos enfoques en un escenario aún más critico. Para este escenario, se optó por reutilizar la configuración obtenida para el escenario de 100 datos etiquetados con el fin de simplificar el análisis.

Para cada combinación de cantidad de datos etiquetados y enfoque (sin AE, AE mono objetivo y AE multi objetivo), se ejecutaron 30 ejecuciones utilizando más épocas que en la etapa de configuración. En todos los casos, la validación se realizó con 100 épocas de entrenamiento. Este valor se fijó dado que, a partir de esa cantidad, los valores de loss comenzaron a oscilar sin mostrar mejoras significativas de desempeño. En el caso del algoritmo sin AE, el entrenamiento consta de 100 épocas secuenciales, mientras que en los algoritmos con AE, las épocas se distribuyeron entre la fase de inicialización y el entrenamiento de cada generación de individuos.

De las ejecuciones se obtuvieron dos conjuntos de valores: uno de FIDs y otro de precisiones. Sobre dichos conjuntos se calcularon estadísticos descriptivos comúnmente utilizados: media, mediana, desviación ( $\sigma$ ), rango intercuartílico (IQR), mínimo y máximo. Además, se aplicaron tests estadísticos para garantizar la validez de las conclusiones:

- Kolmogorov-Smirnov (K-S): utilizado para evaluar la normalidad de las distribuciones. Se consideró  $p > 0.05$  como indicador de normalidad.
- ANOVA: utilizado para comparar medias entre los tres algoritmos en simultáneo y detectar diferencias significativas. Fue utilizado cuando todos los conjuntos fueron normales.
- Levene: utilizado para comparar las varianzas de los algoritmos de a pares tras diferencias detectadas por ANOVA.
- Kruskal-Wallis: utilizado para comparar las medianas de los tres algoritmos en simultáneo. Fue utilizado cuando al menos una distribución fue no normal.
- Wilcoxon: utilizado para comparar los algoritmos de a pares tras diferencias detectadas por Kruskal-Wallis.

## 6.2. Evaluación del enfoque sin AE

En esta sección se presenta la etapa de configuración paramétrica y los resultados obtenidos en la etapa de validación para el enfoque sin AE.

Todas las ejecuciones del algoritmo fueron realizadas en el Centro Nacional de Supercomputación, ClusterUY ([Nesmachnow y Iturriaga, 2019](#)).

### 6.2.1. Configuración paramétrica

Siguiendo la metodología descrita en la subsección 6.1.4, se configuraron cinco parámetros para la GAN. Este proceso fue repetido para cada cantidad de datos etiquetados utilizados (100, 500 y 1000), obteniendo una configuración independiente por cada escenario. Estas configuraciones fueron distribuidas en tres etapas diferentes: configuración de la arquitectura, configuración del entrenamiento y configuración del uso de datos etiquetados. Las etapas se presentan detalladamente en esta subsección.

#### Configuración de la arquitectura

En la primera etapa se configuraron los valores del coeficiente de dropout utilizado por  $D$  y el coeficiente de diversidad  $\gamma$  utilizado en  $L_G$  (ecuación 3.10). Para los coeficientes de dropout se analizaron los valores de 0.2 y 0.5. Para el coeficiente  $\gamma$ , utilizado en  $L_G$ , se analizaron los valores de 0.2, 0.5 y 0.8. Por lo tanto, considerando las variaciones dentro de los parámetros a analizar, se obtuvieron seis configuraciones distintas resultado de todas las combinaciones posibles, las cuales se presentan en el cuadro 6.1.

Dado que la configuración del entrenamiento y del uso de datos etiquetados se realizó después de la configuración de la arquitectura, en esta etapa inicial aún no se disponía de valores formalmente definidos para sus parámetros. Por

Configuración	Dropout	$\gamma$ en $L_G$
A1	0.2	0.2
A2	0.2	0.5
A3	0.2	0.8
A4	0.5	0.2
A5	0.5	0.5
A6	0.5	0.8

Cuadro 6.1: Configuraciones de parámetros de la arquitectura de la GAN

ello, se asignaron valores provisionales basados en pruebas iniciales que indicaron mejores resultados al emplear batch normalization, un tamaño de batch de 100 y todos los datos etiquetados en cada iteración. En consecuencia, para la configuración de la arquitectura se fijaron dichos valores, los cuales fueron ajustados formalmente en etapas posteriores de la configuración paramétrica.

Con los parámetros de entrenamiento y del uso de datos etiquetados fijados, se procedió al análisis de las configuraciones de la arquitectura. En esta etapa se realizaron 30 ejecuciones independientes de 20 épocas por cada configuración. Este análisis fue repetido variando la cantidad de datos etiquetados entre 100, 500 y 1000. Las estadísticas detalladas sobre FIDs y precisiones obtenidas a partir de estas ejecuciones se encuentran en el anexo [A.1.1](#).

En el cuadro [6.2](#) se presentan los hipervolúmenes resultantes de analizar cada configuración específica. Las configuraciones finales fueron aquellas que obtuvieron un mayor valor de hipervolumen. Dados los resultados obtenidos, en los escenarios de 100 y 500 datos etiquetados se utilizó la configuración A3, correspondiente a un dropout de 0.2 y un  $\gamma$  en  $L_G$  de 0.8. En el escenario de 1000 datos etiquetados se utilizó la configuración A2, correspondiente a un coeficiente de dropout de 0.2 y un  $\gamma$  en  $L_G$  de 0.5. Para cada configuración final obtenida, en el anexo [A.1.2](#) se ilustra la representación utilizada para el cálculo del hipervolumen.

Configuración	Cantidad de datos etiquetados		
	100	500	1000
A1	0.4940	0.4544	0.4990
A2	0.4751	0.4395	0.5159
A3	0.4969	0.4766	0.4985
A4	0.4654	0.4513	0.5109
A5	0.4308	0.4460	0.4748
A6	0.4119	0.4713	0.4723

Cuadro 6.2: Hipervolúmenes obtenidos para las configuraciones de la arquitectura variando la cantidad de datos etiquetados

## Configuración del entrenamiento

Una vez establecidos los parámetros de la arquitectura, se procedió a configurar los parámetros de entrenamiento de la GAN. En esta etapa se analizaron el tamaño del batch y el uso de capas de batch normalization. Para el tamaño del batch se evaluaron los valores de 64, 100 y 500, mientras que para las capas de batch normalization se consideró utilizarlas o no. Las configuraciones evaluadas se presentan en el cuadro 6.3.

Configuración	Usa batch normalization	Tamaño de batch
E1	No	64
E2	No	100
E3	No	500
E4	Sí	64
E5	Sí	100
E6	Sí	500

Cuadro 6.3: Configuraciones de parámetros de entrenamiento de la GAN

Para el análisis de los parámetros de entrenamiento fue necesario fijar previamente los parámetros correspondientes a la arquitectura y al uso de datos etiquetados. Los parámetros de la arquitectura se definieron según los resultados previamente obtenidos en la etapa de configuración de la misma, mientras que para el uso de datos etiquetados, cuya configuración ocurre después de la etapa de configuración del entrenamiento, se utilizaron todos los datos etiquetados por iteración, siguiendo la configuración provisional obtenida en la etapa de configuración de la arquitectura.

La configuración de los parámetros de entrenamiento se llevó a cabo en dos fases:

1. Análisis con 1000 datos etiquetados: el primer análisis tuvo por objetivo establecer una configuración final fijando la cantidad de datos etiquetados en 1000. En este caso, se ejecutaron 30 ejecuciones independientes de 20 épocas por cada configuración.
2. Análisis reducido para 100 y 500 datos: el segundo análisis tuvo como finalidad validar la configuración final obtenida en el primero, considerando los escenarios de 100 y 500 datos etiquetados. Este análisis se denominó reducido, dado que para cada configuración se realizaron 10 ejecuciones de 15 épocas cada una.

La separación de los experimentos permitió reducir los tiempos de cómputo asociados a la configuración del entrenamiento, necesidad que surgió a partir de los extensos tiempos de cómputo requeridos para configurar la arquitectura (que también contempló seis configuraciones).

Las estadísticas de FID y precisión obtenidas en el primer análisis (1000 datos etiquetados) se presentan en los cuadros 6.4 y 6.5, respectivamente. En las

configuraciones que no utilizaron batch normalization, la GAN no pudo entrenarse de manera adecuada. Las precisiones resultaron prácticamente nulas y los valores de FID fueron elevados, produciendo imágenes que consistían únicamente en ruido (como se ilustra en el anexo A.1.3). Estos resultados evidencian que el uso de batch normalization es indispensable para estabilizar el entrenamiento de la GAN en este contexto. Debido a las notorias fallas de entrenamiento en las configuraciones  $E1$ ,  $E2$  y  $E3$ , estas fueron descartadas como candidatas. Además, en este experimento, se descartó una ejecución correspondiente a  $E5$  debido a un valor de FID atípico (176.88), que también indica una falla en el entrenamiento. En cuanto a las distribuciones de las métricas, todos los conjuntos de FIDs presentaron distribuciones normales según el test de K-S, mientras que todos los conjuntos de precisiones, salvo el correspondiente a  $E6$ , resultaron en distribuciones no normales.

Configuración	Mínimo	Media	$\sigma$	Mediana	IQR	p-valor
$E1$	300.62	358.66	29.16	359.28	28.22	0.8639
$E2$	261.86	359.16	40.46	360.48	37.01	0.2075
$E3$	307.80	356.74	23.53	357.26	31.26	0.9880
$E4$	10.16	12.27	1.41	12.24	1.70	0.9811
$E5$	9.68	11.70	1.01	11.69	1.15	0.8567
$E6$	10.03	11.54	1.00	11.34	1.04	0.5828

Cuadro 6.4: Estadísticas de FID obtenidas para las distintas configuraciones del entrenamiento utilizando 1000 datos etiquetados

Configuración	Máximo	Media	$\sigma$	Mediana	IQR	p-valor
$E1$	11.35	6.32	4.92	9.33	10.10	0.0017
$E2$	19.63	7.28	5.60	10.10	10.31	0.0061
$E3$	36.02	12.21	8.65	10.10	1.62	0.0054
$E4$	97.08	86.61	15.83	92.68	8.17	0.0066
$E5$	96.78	90.31	12.06	94.87	4.72	0.0022
$E6$	90.01	69.58	8.93	68.61	11.97	0.8853

Cuadro 6.5: Estadísticas de los valores de precisión obtenidos para las distintas configuraciones del entrenamiento utilizando 1000 datos etiquetados

Con las configuraciones  $E1$ ,  $E2$  y  $E3$  descartadas, se procedió al cálculo de hipervolumen para las configuraciones restantes. Los hipervolúmenes de las configuraciones estudiadas resultaron en 0.3545 para  $E4$ , 0.3834 para  $E5$  y 0.3135 para  $E6$ . Como consecuencia, la configuración final obtenida para el escenario de 1000 datos etiquetados fue  $E5$ , cuyo hipervolumen se ilustra en el anexo A.1.4. Esta configuración corresponde a un tamaño de batch de 100 y al uso de capas de batch normalization.

Finalizado el primer análisis, se procedió a la realización del segundo, el cual buscó validar la configuración  $E5$  para los escenarios de 100 y 500 datos etiquetados. Los resultados obtenidos para el análisis reducido confirmaron que las configuraciones sin batch normalization continuaban siendo deficientes. Por este motivo, únicamente se calcularon los hipervolúmenes de  $E4$ ,  $E5$  y  $E6$ , cuyos resultados se muestran en el cuadro 6.6.

Configuración	Datos etiquetados	
	100	500
$E4$	0.3801	0.3398
$E5$	0.3803	0.3044
$E6$	0.2143	0.1806

Cuadro 6.6: Hipervolúmenes obtenidos en el análisis reducido de las configuraciones del entrenamiento

Los resultados del segundo análisis indicaron que la mejor configuración fue  $E4$  en el escenario de 500 datos etiquetados y  $E5$  en el escenario con 100 datos etiquetados. Dado que  $E5$  resultó óptima en dos de tres escenarios (100 y 1000 datos) y presentó diferencias despreciables (menores a 0.04) respecto a  $E4$  en el escenario de 500 datos, se adoptó  $E5$  como configuración única para todos los escenarios. Este compromiso simplifica la experimentación manteniendo un desempeño cercano al óptimo en todos los casos. En consecuencia, para todos los escenarios se utilizaron capas de batch normalization y un tamaño de batch de 100.

### Configuración paramétrica del uso de datos etiquetados

Como se presentó en la subsección 6.1.4, en este proyecto se experimentó con dos modalidades respecto al uso de los datos etiquetados durante el entrenamiento de la GAN. En este análisis se estudiaron los resultados obtenidos para ambas configuraciones  $L1$  y  $L2$ , donde  $L1$  es la configuración que utiliza todos los datos etiquetados por iteración, mientras que  $L2$  es la configuración que utiliza los datos etiquetados repartidos entre todas las iteraciones de una misma época.

Para cada configuración se llevaron a cabo 30 ejecuciones independientes, cada una de 20 épocas, nuevamente variando la cantidad de datos etiquetados entre los valores de 100, 500 y 1000. Además, se respetaron los valores de los parámetros obtenidos previamente en esta subsección correspondientes a la arquitectura y el entrenamiento.

En los cuadros 6.7 y 6.8 se presentan las estadísticas de FID y precisión obtenidas para cada configuración, variando la cantidad de datos etiquetados. En esta etapa se omite una ejecución atípica para el caso de 500 datos etiquetados, la cual reportó un FID de 203.90, lo cual indica una falla en el entrenamiento. Todos los conjuntos de FID presentaron distribuciones normales según el test

de K-S. Además, todos los conjuntos de precisiones, salvo los correspondientes a  $L1$  para los casos de 100 y 1000 datos etiquetados, también presentaron distribuciones normales.

Datos	Configuración	Mínimo	Media	$\sigma$	Mediana	IQR	p-valor
100	$L1$	9.49	11.76	1.27	11.92	1.79	0.9439
	$L2$	18.38	22.98	3.01	22.86	3.73	0.9675
500	$L1$	9.37	12.57	2.14	12.17	1.80	0.1972
	$L2$	9.98	11.69	0.96	11.86	1.50	0.8827
1000	$L1$	9.04	11.87	1.67	11.44	2.01	0.4975
	$L2$	9.84	11.94	1.48	11.56	2.22	0.6088

Cuadro 6.7: Estadísticas de FID obtenidas para las configuraciones  $L1$  y  $L2$ , variando la cantidad de datos etiquetados utilizados

Datos	Configuración	Máximo	Media	$\sigma$	Mediana	IQR	p-valor
100	$L1$	96.24	85.52	16.46	94.12	14.95	0.0058
	$L2$	81.30	64.01	9.83	64.47	15.19	0.9022
500	$L1$	94.09	77.40	18.44	85.18	18.77	0.1483
	$L2$	95.12	75.77	11.24	76.35	18.43	0.7239
1000	$L1$	96.66	89.62	12.79	95.09	6.98	0.0095
	$L2$	89.86	70.55	13.39	72.04	17.22	0.7509

Cuadro 6.8: Estadísticas de precisión obtenidas para las configuraciones  $L1$  y  $L2$ , variando la cantidad de datos etiquetados utilizados

Los valores de hipervolumen obtenidos para cada cantidad de datos etiquetados, cuyas gráficas se ilustran en el anexo A.1.5, se presentan en el cuadro 6.9. Independientemente de la cantidad de datos etiquetados, la configuración  $L1$  obtuvo un mayor hipervolumen, con diferencias notorias en el caso más restrictivo de 100 datos etiquetados. Por lo tanto, para todos los escenarios la configuración final utilizada fue  $L1$ , correspondiente a utilizar todos los datos etiquetados por cada iteración.

Configuración	Cantidad de datos etiquetados		
	100	500	1000
$L1$	0.6444	0.6196	0.6588
$L2$	0.2723	0.6167	0.5718

Cuadro 6.9: Hipervolúmenes obtenidos para las configuraciones del uso de datos etiquetados variando su cantidad

## Configuración final de la GAN

Una vez completadas las tres etapas de configuración (arquitectura, entrenamiento y uso de datos etiquetados), se consolidaron los parámetros finales para cada escenario, los cuales se presentan en el cuadro 6.10. Adicionalmente, para la etapa de validación se incorporó un escenario aún más restrictivo, con solo 60 datos etiquetados. Este último tuvo como propósito evaluar el desempeño de la arquitectura en condiciones más limitadas, reutilizando la configuración final obtenida para el caso de 100 datos etiquetados.

Parámetro	Valor			
	Cantidad de datos etiquetados			
	60	100	500	1000
Dropout	0.2	0.2	0.2	0.2
$\gamma$ en $L_G$	0.8	0.8	0.8	0.5
Batch normalization	Sí	Sí	Sí	Sí
Tamaño del batch	100	100	100	100
Todos los datos por iteración	Sí	Sí	Sí	Sí

Cuadro 6.10: Configuraciones finales de la GAN para cada escenario de datos etiquetados

### 6.2.2. Validación

Una vez fijados los parámetros en la etapa de configuración (presentados en el cuadro 6.10), se procedió a validar la GAN bajo dichos valores. Cada escenario fue evaluado mediante 30 ejecuciones independientes de 100 épocas cada una, de las cuales se extrajeron conjuntos de FIDs, precisiones y tiempos de ejecución. Esta subsección presenta los resultados para cada métrica obtenida, en conjunto con cuadros y figuras auxiliares.

#### FID e imágenes generadas

En el cuadro 6.11 se muestran los valores de FID obtenidos para cada cantidad de datos etiquetados. En el escenario con 500 datos se descartó una ejecución atípica con FID de 420.67, considerada una falla de entrenamiento. El criterio de exclusión se basó en su desviación respecto a la media y en evidencia visual de imágenes generadas incorrectamente. Las medias se mantuvieron cercanas a 10 en todos los escenarios. Además, las distribuciones de FID para los escenarios de 60, 100 y 1000 datos etiquetados resultaron normales según el test de K-S (p-valor > 0.05), con desviaciones estándar inferiores al 11 % de la media, evidenciando estabilidad en el entrenamiento de los generadores.

Respecto a la evolución de FID a lo largo del entrenamiento, todos los generadores presentaron un comportamiento similar. Este comportamiento se carac-

Datos	Media	$\sigma$	Mediana	IQR	Mínimo	Máximo	p-valor
60	9.82	0.98	9.64	1.04	8.06	12.37	0.3383
100	9.56	1.01	9.43	1.37	7.91	11.83	0.9810
500	10.80	2.24	10.64	2.16	8.20	18.91	0.2572
1000	9.83	1.06	9.54	1.54	8.14	12.22	0.5024

Cuadro 6.11: Estadísticas de FID sobre el enfoque sin AE en la etapa de validación, variando la cantidad de datos etiquetados

terizó por un descenso pronunciado de FID en las primeras épocas, seguido de una estabilización próxima al valor final, lo cual indica estabilidad en el entrenamiento de los generadores. A modo de ejemplo, en la figura 6.2 se ilustra la evolución de FID a lo largo de las épocas para la ejecución con mejor desempeño en el escenario de 100 datos etiquetados.

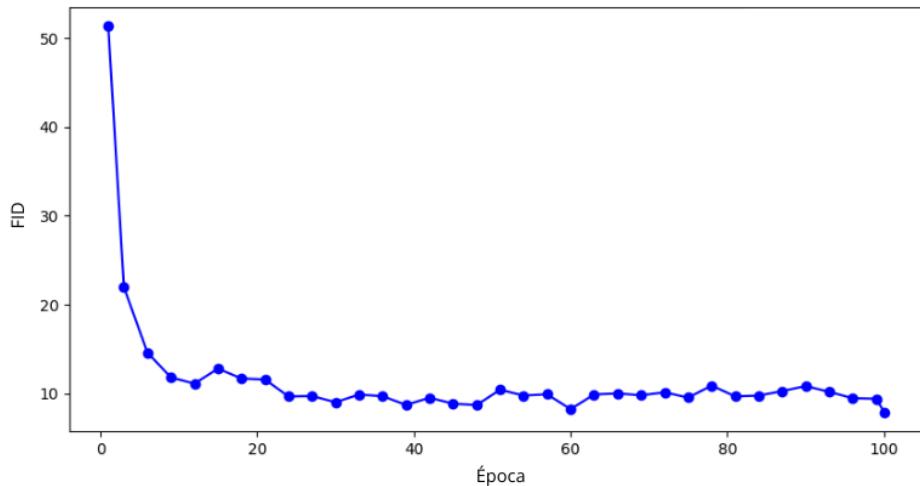


Figura 6.2: Descenso de FID durante las épocas de entrenamiento para el mejor caso que utiliza 100 datos etiquetados en la etapa de validación del enfoque sin AE

Los valores de FID obtenidos se corresponden con la calidad de las imágenes generadas. En la figura 6.3 se muestran ejemplos obtenidos por las ejecuciones con el mejor y el peor valor de FID utilizando 100 datos etiquetados. Las imágenes análogas para el resto de los escenarios se presentan en el anexo A.2.1. En general, las imágenes presentaron dígitos reconocibles y definidos, habiendo algunos casos atípicos que no fueron dígitos reales.



Figura 6.3: Ejemplos de imágenes generadas en las ejecuciones con mejor (izquierda) y peor (derecha) valor de FID utilizando 100 datos etiquetados para el enfoque sin AE en la etapa de validación

### Precisión

Además de la calidad de las imágenes, se evaluó la precisión del discriminador al clasificar, cuyas métricas estadísticas obtenidas para cada escenario se presentan en el cuadro 6.12. El único escenario que presentó una distribución normal según el test de K-S fue el correspondiente a 60 datos etiquetados. En este caso, la media de las precisiones fue inferior a la de los demás escenarios y la desviación estándar alcanzó un valor relativamente alto (17.73, equivalente al 23.5 % de la media), lo que refleja una dispersión considerable. En los escenarios restantes, donde las distribuciones no resultaron normales, se registraron medianas elevadas (superiores al 90 %) acompañadas de valores de IQR que, en relación con la mediana, resultaron amplios (entre un 8 % y un 20 %). Esto indica que, aunque las precisiones tienden a ser altas, existe una dispersión notable entre ejecuciones.

Datos	Media	$\sigma$	Mediana	IQR	Mínimo	Máximo	p-valor
60	75.34	17.73	83.59	31.55	34.40	92.83	0.1351
100	82.45	19.44	92.41	17.92	20.64	94.88	0.0033
500	83.98	21.61	93.12	7.47	8.92	96.07	0.0020
1000	86.94	19.01	94.44	11.05	13.16	97.30	0.0089

Cuadro 6.12: Estadísticas de precisión del enfoque sin AE para la etapa de validación, variando la cantidad de datos etiquetados

En la figura 6.4 se presentan los boxplots de las precisiones para cada escenario. Los resultados muestran que, al incrementar la cantidad de datos etiquetados, se obtuvieron mayores valores de precisión, con diferencias claras en

los escenarios con menor disponibilidad de datos. No obstante, incluso en los escenarios con más datos se registraron ejecuciones con resultados atípicos negativos, alcanzándose precisiones inferiores al 15 % en casos puntuales para los escenarios de 500 y 1000 datos etiquetados.

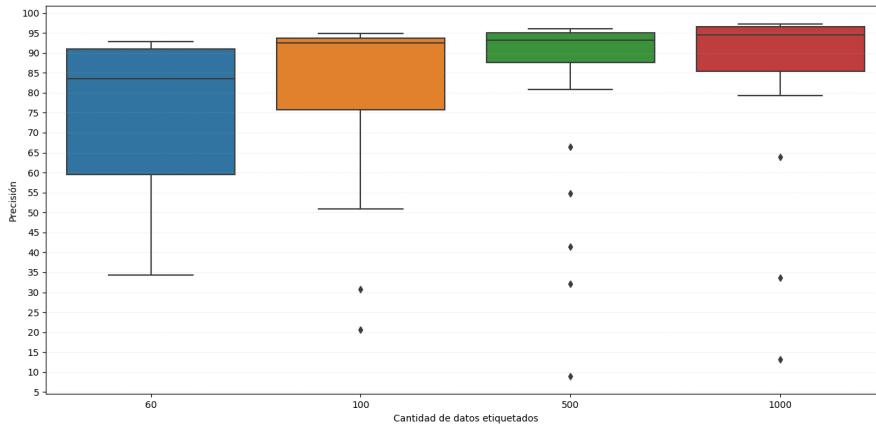


Figura 6.4: Boxplots de precisiones obtenidas para cada escenario en la etapa de validación del enfoque sin AE

### Tiempos de ejecución

En los escenarios con 60 y 1000 datos etiquetados, los tiempos de ejecución siguieron una distribución normal según el test de K-S. Para 60 datos etiquetados, la media fue de 3807 segundos (63 minutos), mientras que para 1000 datos fue de 7236 segundos (121 minutos). En cambio, en los escenarios con 100 y 500 datos etiquetados los tiempos no siguieron una distribución normal, presentando medianas de 4089 segundos (68 minutos) para el escenario de 100 datos y 5190 segundos (87 minutos) para el de 500 datos. El incremento del tiempo de cómputo (de 63 a 121 minutos entre 60 y 1000 datos) refleja el costo adicional del cálculo del loss supervisado al utilizar una mayor cantidad de datos etiquetados, debido a la configuración establecida en la cual se utilizan todos los datos etiquetados en cada iteración de cada época.

### 6.3. Evaluación del enfoque con AE mono objetivo

En esta sección se presenta la etapa de configuración paramétrica y los resultados obtenidos en la etapa de validación para el enfoque que utiliza AE mono objetivo.

### 6.3.1. Configuración paramétrica

En esta subsección se presenta la configuración realizada para los parámetros específicos del AE mono objetivo. Como fue presentado en la subsección 6.1.5, con el objetivo de simplificar el proceso de configuración, en esta etapa se fijó la cantidad de datos etiquetados en 100 dado que es el caso más restrictivo. Además, se utilizaron los parámetros de la GAN obtenidos en la sección anterior (presentados en el cuadro 6.10) para dicho escenario.

La configuración paramétrica fue realizada en dos etapas secuenciales. En primer lugar se analizaron los parámetros relativos a la estructura del AE, mientras que en un segundo análisis se estudiaron parámetros propios del entrenamiento. Para cada etapa se realizaron 20 ejecuciones independientes por cada configuración. Sobre los resultados obtenidos para cada configuración se realizó el cálculo de hipervolumen con el fin de determinar la mejor configuración.

#### Parámetros de la estructura del AE

En la etapa de configuración paramétrica de la estructura del AE se analizaron el tamaño inicial de la población ( $P$ ) y dos parámetros específicos de la selección: la cantidad de individuos elegidos aleatoriamente para el torneo ( $M$ ) y el número de individuos que resultan del torneo ( $K$ ), los cuales participan posteriormente en el cruceamiento.

Con el objetivo de acotar los valores a evaluar para  $P$ ,  $M$  y  $K$ , se realizaron dos experimentos reducidos. El primer experimento tuvo como objetivo estudiar el impacto de  $K$  sobre los tiempos de cómputo. El mismo consistió en realizar ejecuciones variando el valor del parámetro  $K$  entre los valores 3, 6 y 9, dejando fijos los valores de  $P$  y  $M$ . Para cada valor de  $K$  estudiado se realizaron 10 ejecuciones independientes de 22 épocas. Los resultados obtenidos para dichas ejecuciones mostraron una relación lineal entre el valor de  $K$  y los tiempos de ejecución del algoritmo. En consecuencia, y con el objetivo de evitar tiempos de cómputo excesivos, se decidió analizar valores pequeños para  $K$ , los cuales fueron 3 y 5.

En el segundo experimento, se evaluó el efecto del tamaño de la población  $P$ , fijando el resto de los parámetros. Se realizaron 10 ejecuciones de 22 épocas sobre cada valor de  $P$  considerado (3, 6 y 9). Utilizando  $K = 3$ , los resultados no presentaron diferencias significativas entre usar valores de  $P$  cercanos a  $K$  o mayores. En consecuencia, se decidió utilizar valores de  $P$  cercanos a  $K$ , evaluando los ubicados en el rango  $[K, K + 2]$ .

En cuanto al parámetro  $M$ , se utilizaron todos los valores posibles, considerando que el mismo debe estar en el rango  $[K, P]$  dado que define cuántos individuos se toman aleatoriamente de la población (debe ser menor o igual a  $P$ ) para realizar la selección elitista de  $K$  individuos (debe ser mayor o igual a  $K$ ). Con estas restricciones, las configuraciones evaluadas para la estructura del AE mono objetivo se presentan en el cuadro 6.13.

Para cada configuración se realizaron 20 ejecuciones independientes, cada una entrenada durante 22 épocas. Estas épocas fueron distribuidas en 2 épocas

Configuración	<i>K</i>	<i>M</i>	<i>P</i>
<i>A1</i>	3	3	3
<i>A2</i>	3	3	4
<i>A3</i>	3	3	5
<i>A4</i>	3	4	4
<i>A5</i>	3	4	5
<i>A6</i>	3	5	5
<i>A7</i>	5	5	5
<i>A8</i>	5	5	6
<i>A9</i>	5	5	7
<i>A10</i>	5	6	6
<i>A11</i>	5	6	7
<i>A12</i>	5	7	7

Cuadro 6.13: Configuraciones de parámetros relativos al AE mono objetivo

de inicialización y 5 generaciones entrenadas durante 4 épocas cada una. Los resultados estadísticos de FIDs y precisiones se presentan en los cuadros 6.14 y 6.15, respectivamente. A diferencia del enfoque sin AE, todas las distribuciones de FID y precisiones resultaron normales, indicando una mayor estabilidad en el algoritmo. No obstante, los valores de precisión fueron inferiores a los obtenidos en el algoritmo sin AE.

Configuración	Mínimo	Media	$\sigma$	Mediana	IQR	p-valor
<i>A1</i>	9.77	10.91	0.77	10.84	1.05	0.8722
<i>A2</i>	9.67	11.20	1.01	11.00	1.51	0.8278
<i>A3</i>	9.74	11.33	1.11	11.12	1.22	0.5114
<i>A4</i>	9.06	11.42	1.08	11.14	1.21	0.5946
<i>A5</i>	9.29	11.26	0.91	11.08	0.76	0.7277
<i>A6</i>	9.24	11.33	1.22	11.48	1.51	0.9826
<i>A7</i>	8.87	10.71	1.07	10.81	1.12	0.9565
<i>A8</i>	8.77	10.90	1.06	10.88	1.22	0.9970
<i>A9</i>	8.92	11.17	1.12	11.00	1.43	0.9554
<i>A10</i>	9.09	10.71	1.24	10.55	1.41	0.6339
<i>A11</i>	8.62	10.77	1.23	10.80	1.37	0.4891
<i>A12</i>	9.79	11.44	1.01	11.54	1.79	0.4082

Cuadro 6.14: Estadísticas de FID obtenidas para las distintas configuraciones de la estructura del AE mono objetivo

Con el objetivo de determinar la configuración final a utilizar, se calcularon los hipervolúmenes correspondientes a cada configuración. Los resultados, presentados en el cuadro 6.16, muestran que la configuración con mayor hipervolumen, y por lo tanto seleccionada como configuración final, fue *A4*. Esta

Configuración	Máximo	Media	$\sigma$	Mediana	IQR	p-valor
$A1$	90.50	77.32	9.47	79.27	8.24	0.5002
$A2$	88.63	76.68	8.48	78.80	12.98	0.7390
$A3$	91.85	77.13	12.56	80.64	19.53	0.6868
$A4$	92.63	76.84	8.48	77.23	10.12	0.8604
$A5$	92.01	78.40	8.34	79.82	12.74	0.8186
$A6$	90.47	71.87	9.89	73.49	10.99	0.8191
$A7$	86.56	74.53	8.78	75.50	13.56	0.9615
$A8$	90.24	72.35	11.37	74.10	17.12	0.5944
$A9$	90.07	76.69	10.65	79.05	12.21	0.6717
$A10$	89.50	75.01	9.68	73.50	16.58	0.8307
$A11$	91.18	76.24	8.44	74.66	10.55	0.7292
$A12$	93.85	73.10	10.57	72.14	18.06	0.8409

Cuadro 6.15: Estadísticas de precisión obtenidas para las distintas configuraciones de la estructura del AE mono objetivo

configuración corresponde a utilizar un tamaño de población inicial ( $P$ ) de 4, un tamaño de selección aleatoria dentro del torneo ( $M$ ) de 4 y un tamaño del resultado del torneo ( $K$ ) de 3. Dado que  $P = M$ , el proceso de selección fue puramente elitista, sin aleatoriedad. En consecuencia, para el AE mono objetivo, el proceso de selección consistió en tomar los 3 individuos con mejor valor de fitness de cada población,  $P_D$  y  $P_G$ .

Configuración	Hipervolumen
$A1$	0.2514
$A2$	0.2513
$A3$	0.2369
$A4$	0.3013
$A5$	0.2704
$A6$	0.2519
$A7$	0.2579
$A8$	0.2868
$A9$	0.2631
$A10$	0.2800
$A11$	0.2816
$A12$	0.2405

Cuadro 6.16: Hipervolúmenes obtenidos para las distintas configuraciones de la estructura del AE mono objetivo

## Parámetros del entrenamiento

Posteriormente a la configuración de la estructura del AE mono objetivo, se procedió a la configuración de un único parámetro relativo al entrenamiento del mismo: la cantidad de épocas de entrenamiento por generación (denotada  $E_G$ ). Los valores a analizar para este parámetro fueron 3, 4, 5, 6 y 8.

Para poder evaluar las distintas configuraciones en igualdad de condiciones, se fijó en 33 la cantidad total de épocas de entrenamiento (denotada  $E_T$ ). Sean  $I$  la cantidad de generaciones y  $E_I$  la cantidad de épocas de inicialización, entonces  $E_T$  se calcula mediante la ecuación 6.2.

$$E_T = I \times E_G + E_I \quad (6.2)$$

Dada la ecuación 6.2, resultó necesario definir valores adecuados para  $I$  y  $E_I$  para cada valor de  $E_G$  estudiado, de forma que todas las configuraciones mantuvieran el mismo  $E_T$ . En consecuencia, las configuraciones evaluadas para el entrenamiento, las cuales comparten el mismo valor de  $E_T$ , se presentan en el cuadro 6.17. Respecto al criterio utilizado para distribuir las épocas, estas configuraciones fueron construidas de modo que minimizan la cantidad de épocas de inicialización, las cuales son más costosas computacionalmente dado que entrenan todas las parejas disponibles en cada población.

Configuración	Generaciones	$E_G$	$E_I$
$E1$	10	3	3
$E2$	8	4	1
$E3$	6	5	3
$E4$	5	6	3
$E5$	4	8	1

Cuadro 6.17: Configuraciones de parámetros relativos al entrenamiento del AE mono objetivo

Para cada configuración se realizaron 20 ejecuciones independientes. Las estadísticas obtenidas para FIDs y precisiones de cada configuración se encuentran en los cuadros 6.18 y 6.19, respectivamente. Al igual que en la configuración de los parámetros de la estructura del AE mono objetivo, los conjuntos de precisiones y FIDs de todas las configuraciones presentaron distribuciones normales, mostrando mayor estabilidad respecto al enfoque no evolutivo.

Posteriormente, se realizó el cálculo del hipervolumen de cada configuración para determinar la configuración final utilizada. Los resultados obtenidos para el hipervolumen, presentados en el cuadro 6.20, indicaron que la configuración final fue  $E1$ . Por lo tanto, para el AE mono objetivo se utilizaron 3 épocas de entrenamiento por generación.

Configuración	Mínimo	Media	$\sigma$	Mediana	IQR	p-valor
$E1$	7.73	10.40	1.47	10.22	1.54	0.9891
$E2$	7.99	10.40	1.35	10.14	1.62	0.8136
$E3$	8.44	10.32	1.28	10.25	2.04	0.9604
$E4$	7.87	10.70	1.64	10.68	1.30	0.4658
$E5$	9.01	10.52	1.03	10.44	0.88	0.4480

Cuadro 6.18: Estadísticas de FID obtenidas para las distintas configuraciones del entrenamiento del AE mono objetivo

Configuración	Máximo	Media	$\sigma$	Mediana	IQR	p-valor
$E1$	89.47	73.78	11.45	74.56	13.38	0.9317
$E2$	94.07	76.41	10.89	79.06	12.03	0.4423
$E3$	94.23	79.99	9.47	80.34	16.29	0.7132
$E4$	93.51	77.71	13.55	81.59	14.79	0.4337
$E5$	93.61	75.54	16.14	79.82	19.47	0.2826

Cuadro 6.19: Estadísticas de precisión obtenidas para las distintas configuraciones del entrenamiento del AE mono objetivo

Configuración	Hipervolumen
$E1$	0.4163
$E2$	0.3905
$E3$	0.4132
$E4$	0.4161
$E5$	0.3717

Cuadro 6.20: Hipervolúmenes obtenidos para las distintas configuraciones del entrenamiento del AE mono objetivo

### 6.3.2. Validación

Una vez configurados los parámetros del AE mono objetivo, se procedió a la etapa de validación del mismo. En esta etapa se respetaron los valores de los parámetros obtenidos para la GAN (presentados en el cuadro 6.10) y los obtenidos para el AE, configurados en la subsección previa. Para la validación se realizaron 30 ejecuciones independientes del algoritmo, cada una entrenada sobre un total de 100 épocas de entrenamiento. Dado que se utilizaron 3 épocas por generación, y con el objetivo de minimizar las épocas de inicialización, en la etapa de validación las 100 épocas se distribuyeron en 1 época de inicialización y 33 generaciones de 3 épocas cada una. Esta subsección presenta los resultados obtenidos, reportando estadísticas de FID, precisión y tiempo de cómputo, en conjunto con gráficas e imágenes auxiliares.

## FID e imágenes generadas

A partir de las ejecuciones realizadas se estudiaron los valores de FID obtenidos. Las métricas estadísticas obtenidas para dichos valores se presentan en el cuadro 6.21. Todos los conjuntos de FIDs presentaron distribuciones normales según el test de K-S, al igual que en el enfoque sin AE. Los resultados fueron muy similares a los del enfoque sin AE, con medias cercanas a 10 y bajas desviaciones estándar, independientemente de la cantidad de datos etiquetados utilizados. Los resultados indicaron que el AE mono objetivo obtuvo los valores mínimos de FID para todos los escenarios. Una menor cantidad de datos etiquetados se asoció con valores mínimos más bajos, alcanzando ejecuciones con FID de hasta 5.75 en el escenario de 60 datos etiquetados.

Datos	Media	$\sigma$	Mediana	IQR	Mínimo	Máximo	p-valor
60	10.24	2.13	10.50	3.08	5.75	14.45	0.9425
100	9.50	1.23	9.28	1.67	6.85	12.06	0.8866
500	9.70	1.17	9.48	1.56	7.29	12.22	0.7597
1000	9.57	1.08	9.62	1.72	7.71	12.41	0.6028

Cuadro 6.21: Estadísticas de FID para el AE mono objetivo en la etapa de validación, variando la cantidad de datos etiquetados

En cuanto a las imágenes producidas, en la figura 6.5 se muestran ejemplos obtenidos por las ejecuciones con el mejor y el peor valor de FID utilizando 60 datos etiquetados. El mejor caso correspondió también al menor valor de FID alcanzado en todo el proyecto, cuyo descenso a lo largo de las épocas de entrenamiento se ilustra en la figura 6.6. Las imágenes que representan el mejor y el peor caso para el resto de los escenarios (100, 500 y 1000 datos etiquetados) se presentan en el anexo A.3.1.

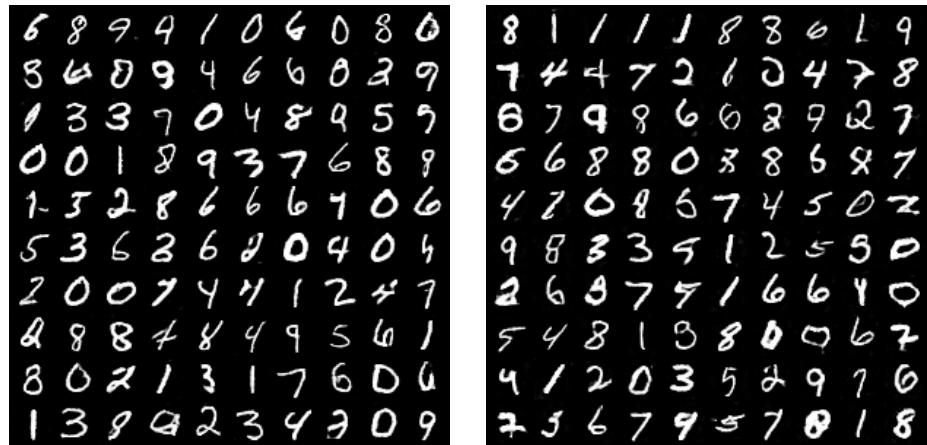


Figura 6.5: Ejemplos de imágenes generadas en las ejecuciones con mejor (izquierda) y peor (derecha) valor de FID utilizando 60 datos etiquetados para el AE mono objetivo en la etapa de validación

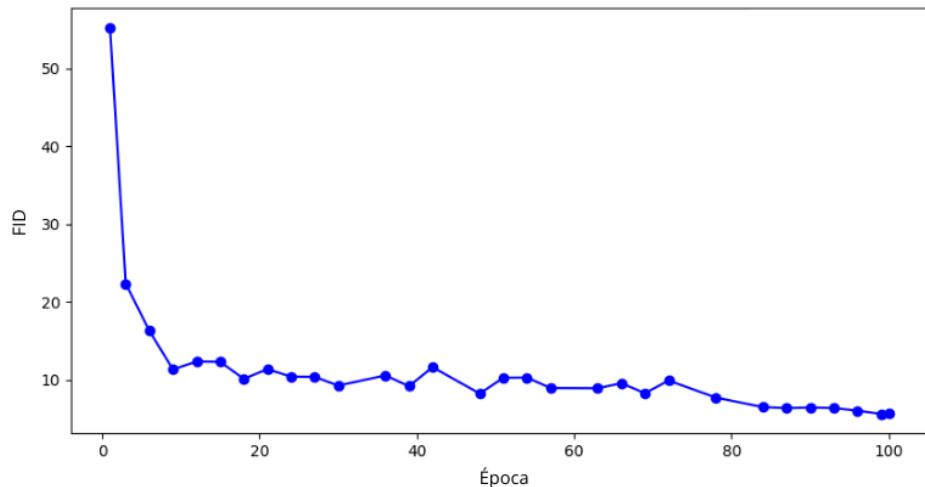


Figura 6.6: Descenso de FID durante las épocas de entrenamiento para el mejor caso que utiliza 60 datos etiquetados en la etapa de validación del AE mono objetivo

### Precisión

Para complementar el análisis de FID realizado para los generadores, se analizaron las precisiones obtenidas por los discriminadores. Se analizaron las precisiones obtenidas a partir de las 30 ejecuciones realizadas, cuyas estadísticas se muestran en el cuadro 6.22. Todos los conjuntos de precisiones resultaron

normales según el test de K-S. En comparación con el enfoque sin AE, el AE mono objetivo produjo distribuciones más compactas y con menor dispersión. Sin embargo, tanto los valores máximos como las medianas fueron inferiores a los obtenidos por el enfoque sin AE.

Datos	Media	$\sigma$	Mediana	IQR	Mínimo	Máximo	p-valor
60	75.39	11.49	78.43	19.09	52.71	89.07	0.4943
100	71.78	11.65	71.90	16.12	41.95	92.58	0.9286
500	72.24	12.73	75.01	16.08	40.14	89.05	0.3528
1000	78.39	7.75	79.40	7.64	60.80	91.54	0.7054

Cuadro 6.22: Estadísticas de precisión para el AE mono objetivo en la etapa de validación, variando la cantidad de datos etiquetados

En cuanto a las distribuciones de las precisiones obtenidas para los distintos escenarios, en la figura 6.7 se presentan boxplots representativos. Los resultados muestran que, en comparación con el enfoque sin AE (boxplots en la figura 6.4), se presentaron menos valores atípicos, al mismo tiempo que las distribuciones son más compactas.

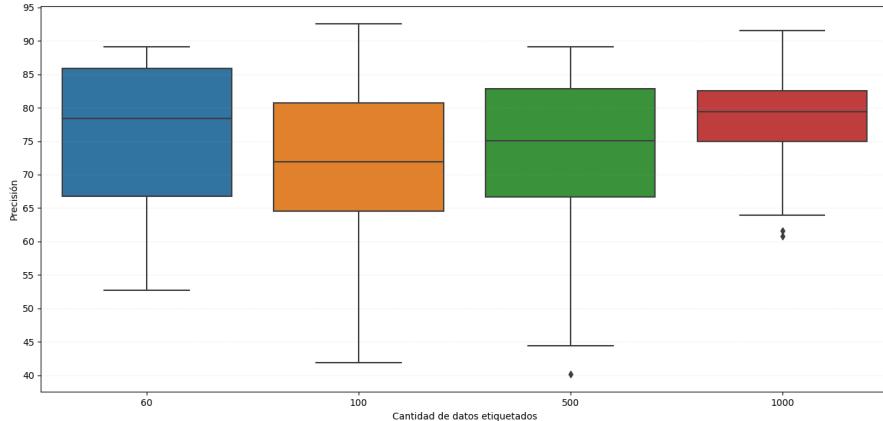


Figura 6.7: Boxplots de precisiones obtenidas en la etapa de validación del AE mono objetivo variando la cantidad de datos etiquetados

### Tiempos de ejecución

Respecto a los tiempos de ejecución, únicamente el escenario con 1000 datos etiquetados presentó una distribución normal según el test de K-S, con una media de 20336 segundos (338 minutos). En los demás escenarios, las distribuciones de los tiempos de ejecución resultaron no normales según el test de K-S. En estos escenarios, las medianas de los tiempos de ejecución fueron 11417 segundos

(190 minutos) para 60 datos etiquetados, 11793 segundos (197 minutos) para 100 datos y 14022 segundos (234 minutos) para 500 datos. En comparación con el enfoque sin AE (63 a 121 minutos), los tiempos de ejecución del AE mono objetivo fueron aproximadamente tres veces superiores, en concordancia con el mayor número de parejas entrenadas en cada época bajo este enfoque.

## 6.4. Evaluación del enfoque con AE multi objetivo

En esta sección se presenta la etapa de configuración paramétrica y los resultados obtenidos en la etapa de validación para el enfoque que utiliza AE multi objetivo.

### 6.4.1. Configuración paramétrica

En esta subsección se presenta la configuración realizada para los parámetros específicos del AE multi objetivo, los cuales fueron presentados en la subsección 6.1.5. Esta subsección es análoga a la subsección 6.3.1, correspondiente a la configuración del AE mono objetivo. La configuración de los parámetros del AE multi objetivo fue separada en dos análisis. En un primer análisis se configuraron los parámetros relativos a la estructura del AE y en el segundo se configuró un único parámetro relativo al entrenamiento del AE, el cual es la cantidad de épocas por generación. Para cada análisis se realizaron 20 ejecuciones independientes para cada configuración, de las cuales se estudiaron FIDs y precisiones obtenidas.

Al igual que en el AE mono objetivo, la cantidad de datos etiquetados fue fijada en 100. Esta decisión se basó en que dicha cantidad representa el caso más restrictivo en la etapa de configuración paramétrica. En consecuencia, la GAN fue configurada con los valores de los parámetros obtenidos previamente para dicho escenario (disponibles en el cuadro 6.10).

#### Parámetros de la estructura del AE

La configuración de la estructura del AE multi objetivo fue análoga a la realizada en la subsección 6.3.1 para el AE mono objetivo. Se evaluaron el tamaño inicial de la población ( $P$ ), la cantidad de individuos seleccionados de forma aleatoria en el torneo ( $M$ ) y la cantidad de individuos resultantes del torneo ( $K$ ). Además, se realizaron los mismos dos experimentos reducidos con el objetivo de acotar los valores a estudiar para dichos parámetros, obteniendo resultados similares. Por lo tanto, las configuraciones a analizar para la estructura del AE multi objetivo fueron exactamente las mismas que las utilizadas por el AE mono objetivo. Estas configuraciones se presentan en el cuadro 6.23.

Al igual que en el enfoque con AE mono objetivo, para cada configuración se realizaron 20 ejecuciones independientes, cada una entrenada durante 22 épocas. Estas épocas fueron distribuidas en 2 épocas de inicialización y 5 generaciones

Configuración	<i>K</i>	<i>M</i>	<i>P</i>
<i>A1</i>	3	3	3
<i>A2</i>	3	3	4
<i>A3</i>	3	3	5
<i>A4</i>	3	4	4
<i>A5</i>	3	4	5
<i>A6</i>	3	5	5
<i>A7</i>	5	5	5
<i>A8</i>	5	5	6
<i>A9</i>	5	5	7
<i>A10</i>	5	6	6
<i>A11</i>	5	6	7
<i>A12</i>	5	7	7

Cuadro 6.23: Configuraciones de parámetros relativos a la estructura del AE multi objetivo

entrenadas durante 4 épocas cada una. Las estadísticas obtenidas de FID y precisión se presentan en los cuadros 6.24 y 6.25, respectivamente. Todos los conjuntos de precisiones y FIDs de todas las configuraciones, salvo los conjuntos de precisiones obtenidos para *A5* y *A10*, resultaron en distribuciones normales.

Configuración	Mínimo	Media	$\sigma$	Mediana	IQR	p-valor
<i>A1</i>	8.98	11.03	1.30	10.96	1.99	0.9163
<i>A2</i>	8.74	11.06	1.13	11.17	1.53	0.9959
<i>A3</i>	9.21	10.87	1.00	10.78	1.60	0.9674
<i>A4</i>	9.31	11.23	1.33	11.32	1.73	0.9669
<i>A5</i>	9.69	11.50	1.45	11.50	1.75	0.8437
<i>A6</i>	9.22	11.03	1.10	10.94	1.50	0.5565
<i>A7</i>	9.51	11.30	1.35	11.20	1.80	0.7192
<i>A8</i>	9.49	10.95	1.04	10.81	1.10	0.8687
<i>A9</i>	9.63	11.07	0.99	11.12	1.38	0.9759
<i>A10</i>	8.98	10.73	0.96	10.52	1.29	0.8696
<i>A11</i>	9.43	11.12	1.17	11.06	1.45	0.8763
<i>A12</i>	8.84	10.97	1.30	10.69	1.46	0.6156

Cuadro 6.24: Estadísticas de FID obtenidas para las distintas configuraciones de la estructura AE multi objetivo

Para establecer la configuración final a utilizar, se calcularon los hipervolúmenes correspondientes a cada configuración. Los resultados, presentados en el cuadro 6.26, muestran que la configuración con mayor hipervolumen, y por lo tanto la configuración final, fue *A2*. Esta configuración corresponde a utilizar un tamaño de población inicial (*P*) de 4, un tamaño de selección aleatoria den-

Configuración	Máximo	Media	$\sigma$	Mediana	IQR	p-valor
<i>A1</i>	96.08	94.14	1.36	94.30	1.43	0.5648
<i>A2</i>	95.80	93.65	1.85	94.32	2.54	0.4259
<i>A3</i>	96.07	94.17	1.36	94.14	1.54	0.7858
<i>A4</i>	95.96	92.91	2.74	93.50	2.02	0.0897
<i>A5</i>	95.69	93.86	2.65	94.92	1.60	0.0301
<i>A6</i>	95.66	93.12	2.76	94.30	2.98	0.2687
<i>A7</i>	95.96	93.65	2.20	94.03	1.78	0.3145
<i>A8</i>	96.18	94.04	1.67	94.32	2.64	0.3947
<i>A9</i>	96.00	94.76	1.06	95.06	0.76	0.2307
<i>A10</i>	96.05	93.65	2.70	94.85	2.86	0.0472
<i>A11</i>	95.81	93.73	2.03	94.38	1.33	0.1508
<i>A12</i>	95.75	93.43	2.36	93.85	2.34	0.3450

Cuadro 6.25: Estadísticas de precisión obtenidas para las distintas configuraciones de la estructura del AE multi objetivo

tro del torneo ( $M$ ) de 3 y un tamaño del resultado del torneo ( $K$ ) de 3. Dado que  $K = M$ , el proceso de selección fue aleatorio, sin un componente elitista. El mismo consistió en seleccionar tres de los cuatro individuos de la población de forma aleatoria.

Configuración	Hipervolumen
<i>A1</i>	0.2773
<i>A2</i>	0.2876
<i>A3</i>	0.2697
<i>A4</i>	0.2589
<i>A5</i>	0.2421
<i>A6</i>	0.2448
<i>A7</i>	0.2478
<i>A8</i>	0.2649
<i>A9</i>	0.2521
<i>A10</i>	0.2801
<i>A11</i>	0.2529
<i>A12</i>	0.2865

Cuadro 6.26: Hipervolúmenes obtenidos para las distintas configuraciones de la estructura del AE multi objetivo

## Parámetros del entrenamiento

Una vez configurados los parámetros de la estructura del AE multi objetivo, se procedió a configurar un único parámetro relativo a su entrenamiento: la cantidad de épocas por generación  $E_G$ . En esta etapa, todos los valores previamente configurados fueron respetados.

Esta subsección presenta un proceso de configuración análogo al realizado en la subsección 6.3.1 para los parámetros del entrenamiento del AE mono objetivo. Por lo tanto, las configuraciones analizadas (presentadas en el cuadro 6.27) fueron exactamente las mismas. Las estadísticas de FID y precisión obtenidas a partir de las ejecuciones realizadas para cada configuración se presentan en los cuadros 6.28 y 6.29, respectivamente. Los conjuntos de precisiones obtenidos para  $E3$  y  $E5$ , y los conjuntos de FIDs de todas las configuraciones presentaron distribuciones normales según el test de K-S.

Configuración	Generaciones	$E_G$	$E_I$
$E1$	10	3	3
$E2$	8	4	1
$E3$	6	5	3
$E4$	5	6	3
$E5$	4	8	1

Cuadro 6.27: Configuraciones de parámetros relativos al entrenamiento del AE multi objetivo

Configuración	Mínimo	Media	$\sigma$	Mediana	IQR	p-valor
$E1$	7.40	10.48	1.48	10.19	1.70	0.8897
$E2$	7.66	10.82	1.58	10.62	1.90	0.8329
$E3$	8.37	10.68	1.47	10.32	1.93	0.8181
$E4$	8.47	10.38	1.38	10.58	2.19	0.6861
$E5$	7.84	10.74	1.70	10.43	1.88	0.9629

Cuadro 6.28: Estadísticas de FID obtenidas para las distintas configuraciones del entrenamiento del AE multi objetivo

Para obtener la configuración final se realizó el cálculo del hipervolumen para cada configuración, cuyos resultados se presentan en el cuadro 6.30. La configuración que obtuvo el mayor valor, y por lo tanto la configuración final utilizada, fue  $E1$ . En consecuencia, para el AE multi objetivo se utilizaron 3 épocas de entrenamiento por generación, al igual que en el AE mono objetivo.

Configuración	Máximo	Media	$\sigma$	Mediana	IQR	p-valor
$E1$	96.11	93.60	3.52	94.98	1.57	0.0408
$E2$	96.26	94.08	3.05	95.16	0.96	0.0119
$E3$	96.14	94.14	2.30	94.85	1.79	0.1997
$E4$	95.87	94.07	2.44	94.88	1.31	0.0314
$E5$	96.03	93.14	3.44	93.76	2.13	0.0513

Cuadro 6.29: Estadísticas de precisión obtenidas para las distintas configuraciones del entrenamiento del AE multi objetivo

Configuración	Hipervolumen
$E1$	0.3872
$E2$	0.3633
$E3$	0.3451
$E4$	0.3386
$E5$	0.3734

Cuadro 6.30: Hipervolúmenes obtenidos para las distintas configuraciones del entrenamiento del AE multi objetivo

#### 6.4.2. Validación

Tras la configuración de los parámetros del AE multi objetivo, se procedió a la etapa de validación. En esta instancia se mantuvieron los valores de los parámetros definidos previamente para la GAN (presentados en el cuadro 6.10) y los obtenidos para el AE multi objetivo en la subsección anterior. Para cada escenario de datos etiquetados se realizaron 30 ejecuciones independientes del algoritmo, entrenadas durante 100 épocas. Dado que cada generación utilizó 3 épocas, estas 100 épocas se distribuyeron en 1 de inicialización y 33 generaciones de 3 épocas cada una, de forma análoga al AE mono objetivo. Esta subsección presenta los resultados obtenidos para el AE multi objetivo, que incluyen métricas estadísticas, gráficas e imágenes ilustrativas.

##### FID e imágenes generadas

Se analizaron FIDs en conjunto con las imágenes producidas por los distintos generadores. Dado que el AE multi objetivo puede tener múltiples generadores en su población final, para cada ejecución se consideró siempre el menor valor de FID obtenido por los distintos generadores. A partir de las ejecuciones realizadas se calcularon las estadísticas de FID, las cuales se presentan en el cuadro 6.31. Los resultados son muy similares a los presentados por los otros enfoques: todas las distribuciones de FID fueron normales según el test de K-S, con medias cercanas a 10 y bajas desviaciones estándar, lo que evidencia estabilidad en el entrenamiento.

Datos	Media	$\sigma$	Mediana	IQR	Mínimo	Máximo	p-valor
60	10.26	1.68	10.16	1.94	7.43	15.67	0.8475
100	9.96	1.85	9.63	1.99	7.40	15.48	0.6094
500	10.26	1.44	10.42	2.09	7.31	13.17	0.9582
1000	10.27	0.95	10.28	1.56	8.72	12.30	0.9497

Cuadro 6.31: Estadísticas de FID del AE multi objetivo para la etapa de validación, variando la cantidad de datos etiquetados

Respecto a las imágenes generadas, en la figura 6.8 se muestran ejemplos producidos por los generadores que obtuvieron el mejor y el peor valor de FID utilizando 500 datos etiquetados. Las imágenes análogas para el resto de los escenarios se presentan en el anexo A.4.1. Los resultados presentan en su mayoría dígitos reconocibles, habiendo algunos casos atípicos que, si bien siguen patrones presentes en los dígitos, no representan dígitos reales.

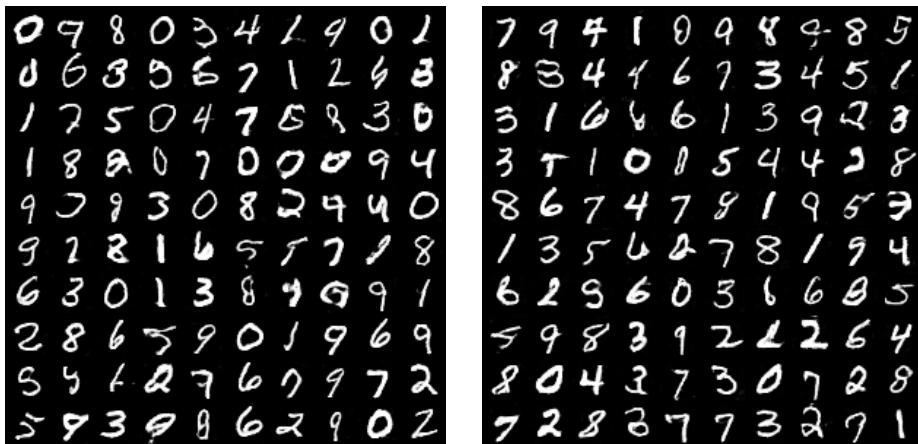


Figura 6.8: Ejemplos de imágenes generadas en las ejecuciones con mejor (izquierda) y peor (derecha) valor de FID utilizando 500 datos etiquetados para el AE multi objetivo en la etapa de validación

En cuanto al comportamiento de FID durante el entrenamiento, en la figura 6.9 se ilustran FIDs obtenidas a lo largo de las épocas para la ejecución que obtuvo el valor mínimo de dicha métrica. Al igual que en el resto de los enfoques, se presenta un descenso pronunciado al inicio y más estable sobre las épocas finales, lo cual indica estabilidad en el entrenamiento de los generadores.

Los frentes de Pareto de los generadores, evaluados según  $F_{calidad}$  y  $F_{diversidad}$ , estuvieron predominantemente compuestos por un único individuo. Este comportamiento se dio en 111 de las 120 ejecuciones, indicando convergencia hacia soluciones dominantes. En cuanto al resto de las ejecuciones, 8 tuvieron 2 generadores en sus frentes de Pareto y tan solo una ejecución logró que su frente de

Pareto estuviera compuesto por 3 generadores (ilustrado en la figura 6.10). La convergencia hacia un único individuo dominante, presentada por la mayoría de los casos, se debe a que, en general, los valores de  $F_{calidad}$  y  $F_{diversidad}$  tuvieron un comportamiento similar, de forma que cuando uno aumenta el otro también lo hace y viceversa. Este comportamiento en los valores de fitness surge a partir de los losses del generador, que son quienes se promedian para dar lugar a dichos fitness, como se describe en las ecuaciones 5.1 y 5.2. En consecuencia, la convergencia hacia un único individuo dominante se debe a la similitud en los comportamientos de ambos losses,  $L_{calidad}$  y  $L_{diversidad}$ , cuyo comportamiento típico se ilustra en la figura 6.11 para un caso representativo.

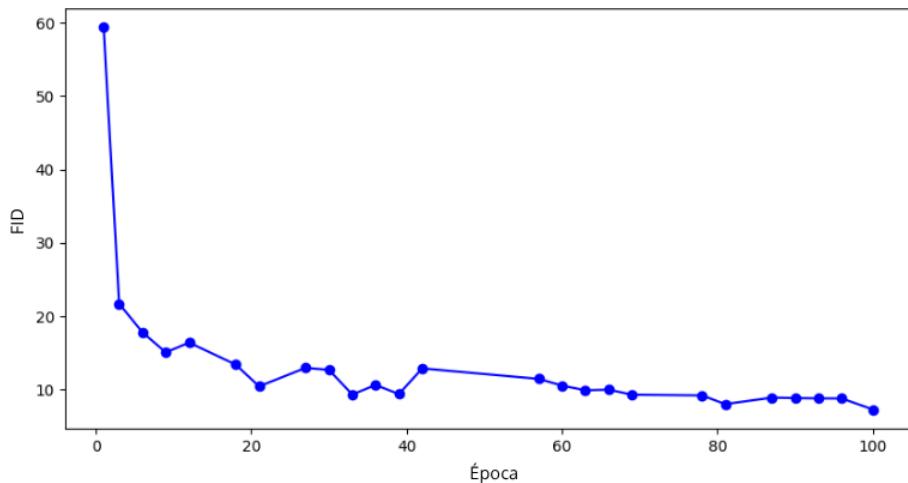


Figura 6.9: Descenso de FID durante las épocas de entrenamiento para el mejor caso que utiliza 500 datos etiquetados en la validación del AE multi objetivo

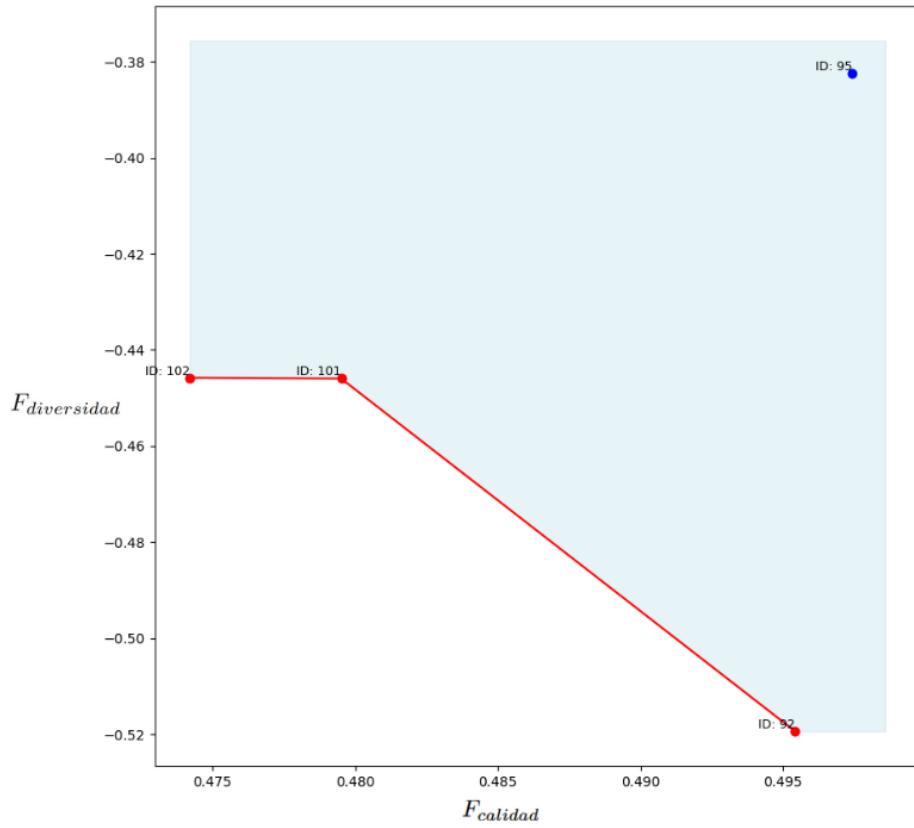


Figura 6.10: Frente de Pareto (rojo) de la población final de generadores para la única ejecución en la cual el mismo estuvo compuesto por tres individuos

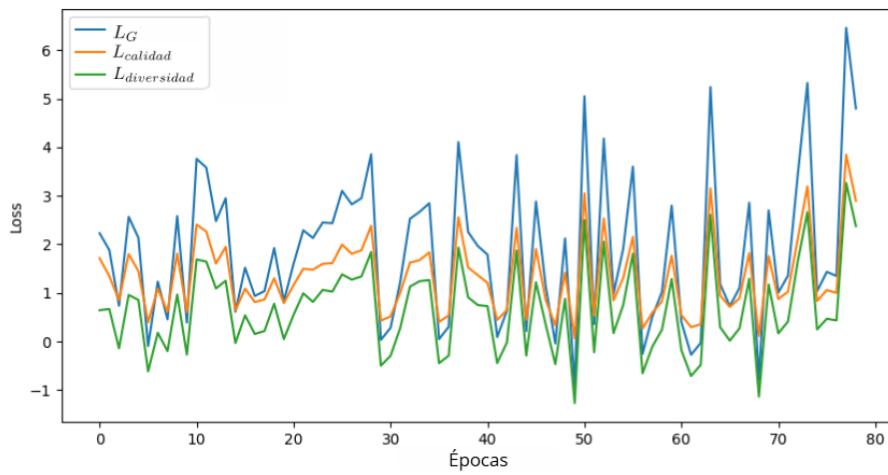


Figura 6.11: Losses obtenidos para un generador durante las épocas de entrenamiento para el AE multi objetivo en su etapa de validación

## Precisión

Complementando el análisis de calidad de generación, y de forma análoga al resto de los enfoques, se procedió a analizar los valores de precisión obtenidos a partir de las 30 ejecuciones realizadas. Para cada ejecución se utilizó como valor de precisión el máximo obtenido sobre todos los discriminadores de la población final. Con esto en cuenta, las estadísticas de precisión para los distintos escenarios se presentan en el cuadro 6.32. Todos los conjuntos de precisiones, salvo el correspondiente al escenario de 100 datos etiquetados, resultaron normales según el test de K-S. Además, se obtuvieron bajos valores de dispersión, que representan entre el 0.5 % y el 8.8 % de las medias obtenidas, indicando estabilidad en los desempeños de los discriminadores.

Los resultados indican que al utilizar una mayor cantidad de datos etiquetados, las distribuciones se volvieron más compactas, al mismo tiempo que alcanzaron valores superiores, como se presenta en los boxplots de la figura 6.12. El escenario de mayor cantidad de datos etiquetados (1000), logró precisiones superiores al 95 % para todas sus ejecuciones.

Datos	Media	$\sigma$	Mediana	IQR	Mínimo	Máximo	p-valor
60	87.58	7.65	89.02	6.38	59.37	95.74	0.1755
100	92.56	4.03	94.16	2.97	80.53	96.13	0.0358
500	94.17	2.46	95.02	1.73	85.80	96.34	0.0554
1000	96.68	0.49	96.74	0.68	95.47	97.49	0.7725

Cuadro 6.32: Estadísticas de precisión del AE multi objetivo para la etapa de validación, variando la cantidad de datos etiquetados

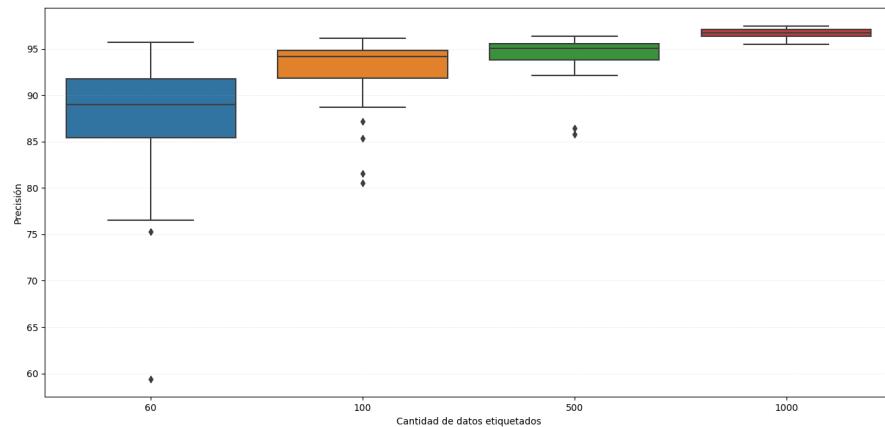


Figura 6.12: Boxplots de precisiones obtenidas en la etapa de validación del AE multi objetivo variando la cantidad de datos etiquetados

Respecto a los frentes de Pareto obtenidos para las poblaciones finales de discriminadores, estuvieron compuestos por un mínimo de 1 discriminador y un máximo de 4, que se corresponde con el total de la población. A diferencia de los frentes de Pareto presentados por los generadores, los frentes de los discriminadores estuvieron en general compuestos por más de un individuo, indicando mayor diversidad en los individuos entrenados. La distribución exacta de los frentes de Pareto para los discriminadores fue de 10 ejecuciones con 1 solo individuo, 30 ejecuciones con 2, 44 ejecuciones con 3 y 36 ejecuciones con 4. Salvo una excepción para el escenario de 500 datos, en todos los escenarios las ejecuciones que presentaron frentes de Pareto compuestos por más individuos obtuvieron mejores medias en sus precisiones (como se presenta en el anexo [A.4.2](#)), remarcando que la diversidad en la población fue un factor determinante para lograr mejoras en el desempeño de los discriminadores. En concordancia con este punto, la precisión máxima obtenida en este proyecto (%97.49) fue obtenida para un frente de Pareto compuesto por todos los discriminadores de la población final. En la figura [6.13](#) se ilustra dicho frente de Pareto, siendo el discriminador 101 el que obtuvo la precisión máxima.

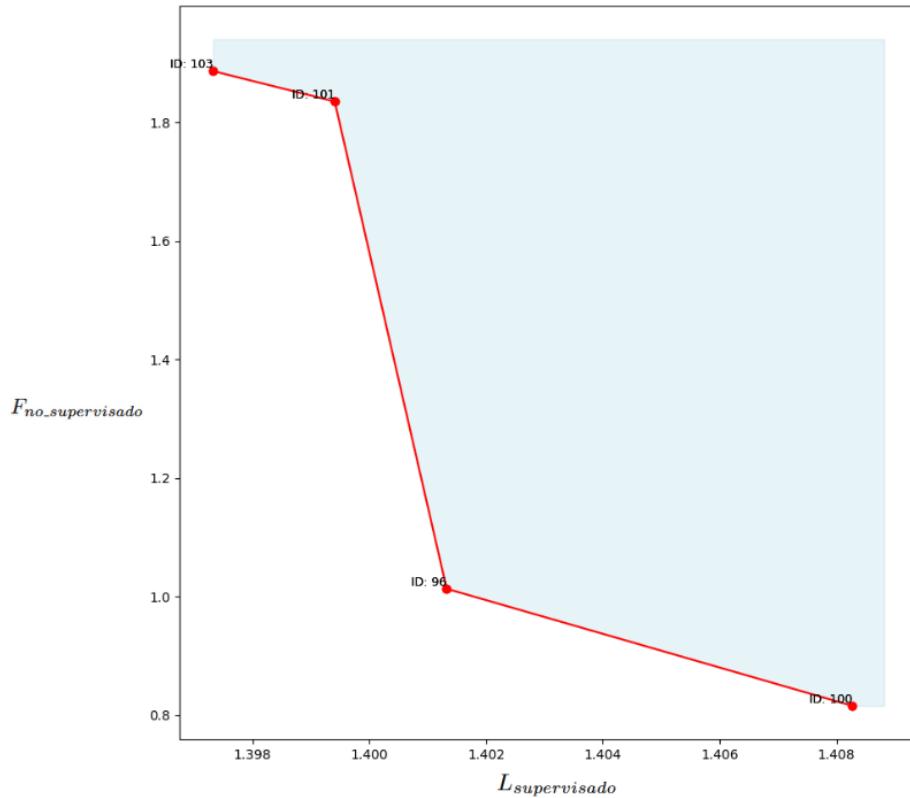


Figura 6.13: Frente de Pareto (rojo) compuesto por todos los individuos de la población final de discriminadores para la ejecución con el mejor valor de precisión

### Tiempos de ejecución

En el enfoque con AE multi objetivo todos los conjuntos de tiempos de ejecución, excepto el correspondiente al escenario de 60 datos etiquetados, resultaron en distribuciones normales según el test de K-S. Las medias obtenidas para los escenarios que presentaron distribuciones normales fueron 13765 segundos (229 minutos) para 100 datos etiquetados, 18058 segundos (301 minutos) para 500 datos y 20336 segundos (339 minutos) para 1000 datos. Además, el escenario de 60 datos etiquetados presentó una mediana de 12019 segundos (200 minutos). Los tiempos de ejecución presentados por el AE multi objetivo fueron similares a los obtenidos por el AE mono objetivo.

## 6.5. Análisis de resultados

En esta sección se analizan los resultados obtenidos en la etapa de validación para cada enfoque (sin AE, AE mono objetivo y AE multi objetivo), destacando sus similitudes y diferencias. Asimismo, se presentan los resultados obtenidos para los tests estadísticos (introducidos en la subsección 6.1.6), los cuales permiten identificar si existen diferencias significativas entre las distribuciones de precisiones y FID de los distintos enfoques. Los tests fueron realizados de forma independiente para cada escenario de datos etiquetados. Finalmente, se presenta un análisis comparativo entre los resultados obtenidos en este proyecto con los del estado del arte, y además, con un proyecto que tuvo un enfoque similar, utilizando AE.

### 6.5.1. Precisión

La métrica de precisión fue la que evidenció las mayores diferencias entre los distintos enfoques. Tal como se planteó a lo largo del proyecto, esta métrica muestra un menor desempeño a medida que se reduce la cantidad de datos etiquetados. Esto se debe a que, si bien se dispone de un gran volumen de imágenes sin etiquetar, el aprendizaje de las clases reales durante el entrenamiento depende exclusivamente de los datos etiquetados.

En el cuadro 6.33 se presentan las estadísticas descriptivas de las precisiones obtenidas para cada escenario y algoritmo. Para ilustrar gráficamente las distribuciones, la figura 6.14 muestra los boxplots de las precisiones correspondientes al escenario de 1000 datos etiquetados para los distintos enfoques, mientras que los boxplots análogos para el resto de los escenarios se incluyen en el anexo A.5.1. Al comparar los enfoques implementados, los que utilizaron AE lograron un entrenamiento más robusto y estable, lo cual se refleja en las reducciones sustanciales de las desviaciones estándar e IQR, en comparación con el enfoque sin AE. Al comparar ambos AE entre ellos, el multi objetivo obtuvo mejores valores de desviación e IQR.

La robustez del entrenamiento en los enfoques que utilizaron AE también se puede inferir de los resultados de los tests de K-S. En estos tests todas las distribuciones correspondientes a los enfoques que utilizan AE resultaron normales, salvo por la correspondiente al AE multi objetivo en el escenario de 100 datos etiquetados. En contraparte, para el enfoque sin AE tres de los cuatro escenarios resultaron en distribuciones no normales.

Los resultados indican que el AE multi objetivo alcanzó los mejores desempeños en todos los escenarios, con medias aproximadamente un 10 % superiores respecto al enfoque sin AE y medianas entre 1,75 % y 5,43 % más altas. Sumando a esto, el AE multi objetivo logró reducir considerablemente la desviación estándar respecto al enfoque sin AE, con reducciones de entre un 56 % y un 97 %, presentándose mayores reducciones al aumentar la cantidad de datos etiquetados. Por lo tanto, el AE multi objetivo no solo alcanza mayores valores de precisión, sino que también proporciona resultados más reproducibles y robustos.

Datos	Algoritmo	Media $\pm \sigma$	Mediana $\pm$ IQR	Mínimo-Máximo	p-valor
60	sin AE	75.34 $\pm$ 17.73	83.59 $\pm$ 31.55	34.40 - 92.83	0.1351
	mono objetivo	75.39 $\pm$ 11.49	78.43 $\pm$ 19.09	52.71 - 89.07	0.4943
	multi objetivo	87.58 $\pm$ 7.65	89.02 $\pm$ 6.38	59.37 - 95.74	0.1755
100	sin AE	82.45 $\pm$ 19.44	92.41 $\pm$ 17.92	20.64 - 94.88	0.0033
	mono objetivo	71.78 $\pm$ 11.65	71.90 $\pm$ 16.12	41.95 - 92.58	0.9286
	multi objetivo	92.56 $\pm$ 4.03	94.16 $\pm$ 2.97	80.53 - 96.13	0.0358
500	sin AE	83.98 $\pm$ 21.61	93.12 $\pm$ 7.47	8.92 - 96.07	0.0020
	mono objetivo	72.24 $\pm$ 12.73	75.01 $\pm$ 16.08	40.14 - 89.05	0.3528
	multi objetivo	94.17 $\pm$ 2.46	95.02 $\pm$ 1.73	85.80 - 96.34	0.0554
1000	sin AE	86.94 $\pm$ 19.01	94.44 $\pm$ 11.05	13.16 - 97.30	0.0089
	mono objetivo	78.39 $\pm$ 7.75	79.40 $\pm$ 7.64	60.80 - 91.54	0.7054
	multi objetivo	96.68 $\pm$ 0.49	96.74 $\pm$ 0.68	95.47 - 97.49	0.7725

Cuadro 6.33: Estadísticas de precisión obtenidas en la etapa de validación para los distintos escenarios y algoritmos utilizados

En cuanto a los resultados obtenidos por el AE mono objetivo, si bien resultó más robusto en su entrenamiento que el enfoque sin AE, el mismo presentó medias y medianas de precisión inferiores a las obtenidas por el resto de los enfoques. En otras palabras, el AE mono objetivo mejoró la consistencia del entrenamiento pero no tradujo esta estabilidad en mejores valores de precisión. Esta reducción en las precisiones es atribuible a la dominancia del componente no supervisado en la función de fitness única, que tuvo mayor variabilidad respecto al componente supervisado. Esto plantea una limitación importante para el AE mono objetivo: el mismo termina priorizando individuos en base a su fitness total, cuya variación puede estar dominada por un fitness parcial. Por su parte, el AE multi objetivo evaluó de forma separada los componentes supervisado y no supervisado, evitando la dominancia presentada por el esquema mono objetivo. Esto permitió que el entrenamiento optimizara ambos criterios de manera equilibrada, logrando mejores valores de precisión.

Para demostrar formalmente las diferencias entre las distribuciones de precisiones para cada enfoque, se realizaron tests estadísticos. Para el escenario de 60 datos etiquetados, el cual fue el único que presentó distribuciones normales para todos los enfoques, se realizaron tests ANOVA y Levene, cuyos resultados se presentan en el cuadro 6.34. Tanto el test ANOVA como los diferentes tests de Levene de a pares indicaron diferencias significativas, con la excepción del test de Levene correspondiente a la comparación entre el enfoque sin AE y el enfoque con AE mono objetivo. Para los escenarios de 100, 500 y 1000 datos etiquetados, los cuales presentaron al menos un enfoque con una distribución de precisiones no normal, se realizaron tests de Kruskal-Wallis y Wilcoxon, cuyos resultados se presentan en el cuadro 6.35. Ambos tests indicaron diferencias significativas entre todos los enfoques para todos los escenarios. En resumen, todas las distribuciones de precisiones presentaron diferencias significativas entre sí in-

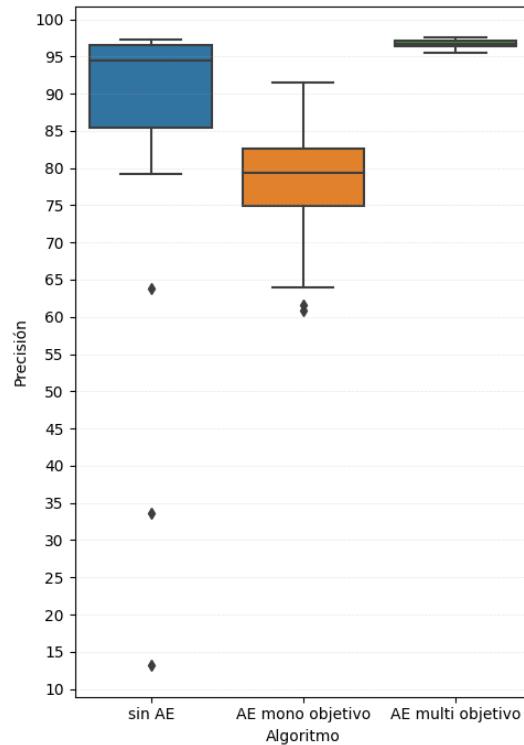


Figura 6.14: Boxplots de las precisiones obtenidas en la etapa de validación para los distintos algoritmos estudiados utilizando 1000 datos etiquetados

dependientemente del enfoque y la cantidad de datos etiquetados, salvo por la comparación entre el enfoque sin AE y el enfoque con AE mono objetivo para el escenario de 60 datos etiquetados, el cual no presentó diferencias significativas.

ANOVA		Levene	
Diferencias	p-valor	Algoritmos comparados	Diferencias
Sí	0.0003	sin AE - AE mono objetivo sin AE - AE multi objetivo AE mono - AE multi objetivo	No Sí Sí

Cuadro 6.34: Tests estadísticos sobre las distribuciones de precisiones obtenidas para los distintos enfoques en la etapa de validación para el escenario de 60 datos etiquetados

Datos	Kruskal-Wallis			Wilcoxon		
	Diferencias	p-valor	Algoritmos comparados	Diferencias	p-valor	
100	Sí	0.0000	sin AE - AE mono objetivo	Sí	0.0157	
			sin AE - AE multi objetivo	Sí	0.0300	
			AE mono - AE multi objetivo	Sí	0.0000	
500	Sí	0.0000	sin AE - AE mono objetivo	Sí	0.0026	
			sin AE - AE multi objetivo	Sí	0.0010	
			AE mono - AE multi objetivo	Sí	0.0000	
1000	Sí	0.0000	sin AE - AE mono objetivo	Sí	0.0026	
			sin AE - AE multi objetivo	Sí	0.0001	
			AE mono - AE multi objetivo	Sí	0.0000	

Cuadro 6.35: Tests estadísticos sobre las distribuciones de precisiones obtenidas para los distintos enfoques en la etapa de validación para los escenarios que presentaron al menos un enfoque con una distribución no normal

### 6.5.2. FID e imágenes generadas

Los resultados de FID y sus distribuciones fueron muy similares entre todos los enfoques y escenarios. En todos los casos, los conjuntos obtenidos en la etapa de validación resultaron en distribuciones normales según el test de K-S, con excepción del enfoque sin AE en el escenario de 500 datos etiquetados. Las medias se ubicaron alrededor de 10, con desviaciones pequeñas y muy pocos datos atípicos. En consecuencia, el entrenamiento de los generadores fue estable y robusto, independientemente del enfoque o escenario considerado. Sin embargo, el AE mono objetivo alcanzó los valores mínimos de FID en todos los escenarios. Este resultado puede atribuirse a que discriminadores más débiles, como los producidos por el AE mono objetivo (discutido previamente en la subsección 6.5.1), permiten que los generadores los superen más fácilmente, optimizando su FID a costa de la utilidad del discriminador para clasificación.

Para formalizar las similitudes mencionadas en el párrafo anterior respecto a las distribuciones, se aplicaron tests estadísticos comparando los tres enfoques para cada escenario de datos etiquetados. En los escenarios de 60, 100 y 1000 datos etiquetados, se realizaron tests ANOVA y Levene, cuyos resultados se presentan en el cuadro 6.36. Los resultados del ANOVA mostraron que no existían diferencias significativas en los escenarios de 60 y 100 datos. En el caso de 1000 datos, aunque el ANOVA identificó diferencias globales, las comparaciones por pares realizadas con el test de Levene no evidenciaron diferencias significativas entre ninguno de los enfoques. En el escenario de 500 datos etiquetados, donde el enfoque sin AE presentó una distribución no normal, se aplicó un test de Kruskal-Wallis sobre las distribuciones obtenidas para todos los enfoques. Este test indicó que no existían diferencias significativas entre ninguna pareja de distribuciones, obteniendo un p-valor de 0.0610. En síntesis, los análisis confirmaron que en todos los escenarios las distribuciones de FID fueron similares, sin diferencias significativas para ningún enfoque específico.

Para comparar visualmente las distribuciones de los distintos enfoques, en la figura 6.15 se ilustran los boxplots de FIDs obtenidas para el escenario de 60

Datos	ANOVA		Levene		
	Diferencias	p-valor	Enfoques comparados	Diferencias	p-valor
60	No	0.5124			
100	No	0.3839			
1000	Sí	0.0337	sin AE - AE mono objetivo sin AE - AE multi objetivo AE mono - AE multi objetivo	No No No	0.7610 0.7887 0.5156

Cuadro 6.36: Tests estadísticos sobre las distribuciones de FID obtenidos para los distintos enfoques en la etapa de validación para los escenarios en los cuales todos los enfoques presentaron distribuciones normales

datos etiquetados sobre los distintos algoritmos utilizados. Los boxplots análogos se encuentra en el anexo A.5.2. En todos los escenarios, los resultados indicaron que, si bien las medias y medianas son muy similares, el AE mono objetivo fue capaz de lograr ejecuciones puntuales que lograron los mínimos más bajos, es decir, los mejores valores de FID para cada escenario.

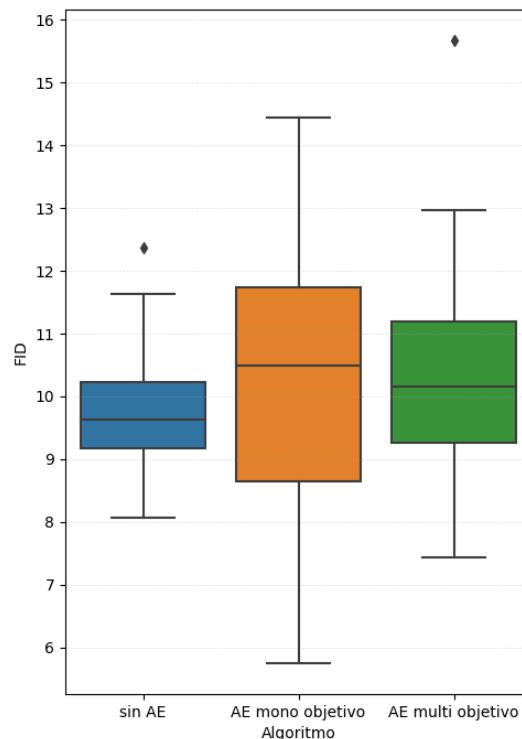


Figura 6.15: Boxplots de FIDs obtenidas en la etapa de validación para los distintos algoritmos estudiados utilizando 60 datos etiquetados

Respecto a las imágenes generadas, presentadas en la etapa de validación de cada enfoque, las mismas no presentaron diferencias visuales relevantes independientemente del escenario, lo cual es coherente con la similitud en los valores y las distribuciones de FID. En todos los casos, se visualizan gran cantidad de dígitos reconocibles, habiendo algunas imágenes atípicas que, si bien contienen patrones presentes en los dígitos como líneas y curvas, no representaron dígitos reales.

### 6.5.3. Tiempos de ejecución

El costo computacional constituye un aspecto relevante al evaluar la viabilidad práctica de los diferentes enfoques propuestos. En esta subsección se analizan los tiempos de ejecución obtenidos durante la etapa de validación, donde cada enfoque fue entrenado durante 100 épocas totales.

#### Resultados obtenidos

El cuadro 6.37 presenta las estadísticas de tiempo de ejecución para cada combinación de enfoque y cantidad de datos etiquetados. Los tiempos se expresan en minutos para facilitar la interpretación.

Datos	Algoritmo	Media	$\sigma$	Mediana	IQR
60	sin AE	63.45	10.88	68.51	8.47
	mono objetivo	181.78	27.57	190.28	17.91
	multi objetivo	220.91	83.47	200.31	21.11
100	sin AE	88.05	68.73	68.15	28.09
	mono objetivo	223.70	70.19	196.55	41.04
	multi objetivo	229.41	36.66	213.98	55.15
500	sin AE	99.23	19.36	86.50	37.90
	mono objetivo	281.23	94.03	233.70	81.64
	multi objetivo	301.45	106.99	252.84	85.08
1000	sin AE	120.60	31.73	120.35	33.53
	mono objetivo	347.41	61.77	323.67	95.38
	multi objetivo	338.93	66.12	316.08	88.92

Cuadro 6.37: Estadísticas de tiempos de ejecución (en minutos) obtenidas en la etapa de validación para los distintos escenarios y algoritmos utilizados

#### Comparación entre enfoques

Los enfoques con AE presentaron tiempos de ejecución consistentemente superiores al enfoque sin AE, de alrededor del triple de los obtenidos para el enfoque no evolutivo. Este incremento es atribuible a la estructura poblacional de los AE: mientras el enfoque sin AE entrena una única pareja generador-discriminador, los enfoques evolutivos entrenaron 3 parejas en simultáneo. Para el escenario de 100 datos etiquetados, por ejemplo, el enfoque sin AE presentó una mediana de 68 minutos, mientras que los enfoques con AE mono objetivo y multi objetivo reportaron medianas de 197 y 229 minutos, respectivamente.

Esta diferencia representa un costo computacional adicional de  $2.9 \times$  a  $3.4 \times$  respecto al enfoque base. Entre ambos AE, los tiempos fueron similares (diferencia menor al 10 %), lo cual es esperable dado que comparten la misma estructura poblacional y difieren principalmente en los operadores de selección y reemplazo, cuyo costo es despreciable respecto al entrenamiento de las redes.

### Escalamiento con datos etiquetados

El tiempo de ejecución aumentó con la cantidad de datos etiquetados. Este comportamiento se explica por la configuración adoptada, donde todos los datos etiquetados se utilizan en cada iteración para calcular el loss supervisado. Como consecuencia, el costo computacional del discriminador escala linealmente con el número de ejemplos etiquetados. En el enfoque sin AE, los tiempos incrementaron de 63 minutos (60 datos) a 121 minutos (1000 datos), representando un aumento del 92 %. Para los enfoques evolutivos, el incremento fue similar: en el AE multi objetivo, los tiempos pasaron de aproximadamente 200 minutos (60-100 datos) a 339 minutos (1000 datos), correspondiente a un aumento del 70 %.

### Implicaciones prácticas

Los resultados de eficiencia computacional deben considerarse en conjunto con las mejoras en desempeño reportadas en las secciones anteriores:

- El enfoque sin AE es el más eficiente computacionalmente pero presenta alta variabilidad en precisión ( $\sigma$  entre 17.73 y 21.61, según escenario) y valores atípicos frecuentes.
- El AE mono objetivo requiere  $3 \times$  más tiempo pero no logra mejoras en sus medias de precisión respecto al enfoque sin AE, aunque sí reduce la variabilidad ( $\sigma$  entre 7.75 y 12.73). Esta relación costo-beneficio lo hace poco competitivo.
- El AE multi objetivo también requiere  $3 \times$  más tiempo, pero logra mejoras sustanciales en precisión (incrementos de 10-20 % en media según escenario) y reduce significativamente la variabilidad ( $\sigma$  entre 0.49 y 7.65). Para aplicaciones donde la precisión del discriminador es crítica y se dispone de recursos computacionales adecuados, este enfoque justifica su costo adicional.

En escenarios críticos donde se requiera entrenar múltiples modelos hasta obtener uno satisfactorio, el AE multi objetivo podría resultar más eficiente globalmente: aunque cada ejecución toma  $3 \times$  más tiempo, la mayor consistencia de resultados reduce la necesidad de re-entrenamientos.

#### 6.5.4. Comparación con arquitecturas del estado del arte

En esta subsección se comparan los resultados obtenidos en este proyecto con los siguientes trabajos correspondientes al estado del arte: ([Salimans y cols., 2016](#)), ([Dong y Lin, 2019](#)), ([Zieba y Wang, 2017](#)) y ([Li y cols., 2019](#)). La comparación con dichos trabajos debe considerarse con precaución debido a diferencias metodológicas significativas. Estos trabajos:

- Utilizan arquitecturas con mayor número de capas
- Entrenan durante 400-500 épocas, mientras que en este proyecto se utilizaron 100 épocas
- Optimizan exclusivamente para precisión dejando de lado la calidad y diversidad de las imágenes generadas

El cuadro [6.38](#) muestra que el AE multi objetivo, el cual presentó los mejores valores de precisión en este proyecto, obtuvo precisiones por debajo de las obtenidas en el resto de los trabajos analizados. Por ejemplo, en el escenario de 100 datos etiquetados, se obtuvo una mediana de 94.16 %, la cual está por debajo de las obtenidas en otros proyectos (entre 96.50 % y 99.21 %). En cuanto al escenario de 60 datos, el cual recorta casi a la mitad los datos etiquetados en comparación con el escenario de 100 datos etiquetados, el resto de los trabajos no reportan desempeños por lo que es imposible determinar si las precisiones obtenidas en este proyecto son superiores para casos de extrema escasez de datos.

Modelo	Cantidad de datos etiquetados				
	60	100	500	600	1000
UGAN ( <a href="#">Li y cols., 2019</a> )		99.21			
FM-GAN ( <a href="#">Salimans y cols., 2016</a> )		99.07			
Triplet Loss ( <a href="#">Zieba y Wang, 2017</a> )		97.61		98.64	
MarginGAN ( <a href="#">Dong y Lin, 2019</a> )		96.47		97.13	
AE multi objetivo	89.02	94.16	95.02		96.74

Cuadro 6.38: Comparación de precisiones obtenidas en los distintos trabajos del estado del arte con las obtenidas en este proyecto

Este proyecto no alcanzó las precisiones del estado del arte, en parte debido al uso de arquitecturas más simples y menor cantidad de épocas de entrenamiento. Sin embargo, a diferencia de los trabajos analizados, que optimizan exclusivamente la precisión, la solución propuesta mantuvo un equilibrio con la calidad de generación, como evidencian los valores de FID consistentemente bajos (medias de alrededor de 10). Sin embargo, es imposible realizar una comparación cuantitativa directa dado que los trabajos analizados no reportan métricas de calidad de generación (FID). Aún así, la figura [6.16](#) muestra ejemplos visuales

de imágenes generadas por los diferentes métodos. Si bien se considera que las imágenes generadas en este proyecto presentan mejoras de calidad comparables, se reconoce que esta comparación visual es subjetiva y no puede reemplazar una evaluación métrica rigurosa.

Finalmente, se compararon las precisiones obtenidas en este proyecto con las presentadas en el artículo ([Sedeño y cols., 2025](#)), el cual presenta un enfoque similar donde se entrena poblaciones de generadores y discriminadores. Para los escenarios con mayor disponibilidad de datos etiquetados, los resultados presentados en dicho artículo son similares a los obtenidos en este proyecto con medianas de entre 95 % y 97 % y valores de IQR menores a 2 %. En los escenarios con menor disponibilidad de datos, se presentaron las mayores diferencias. El artículo presenta un escenario de 100 datos etiquetados que obtuvo una mediana de 88.00 % con un IQR de 1.30 %, mientras que en este proyecto se obtuvo una mediana de 94.15 % y un IQR de 2.97 % para dicho escenario, siendo la mediana obtenida en este proyecto 6.15 % mayor. Además, el escenario de 60 datos etiquetados evaluado en este proyecto presentó una mediana mayor (89.02 %) a la obtenida para el escenario de 100 datos etiquetados en dicho artículo.



(a) UGAN



(b) FM-GAN



(c) MarginGAN



(d) GAN sin AE



(e) AE mono objetivo



(f) AE multi objetivo

Figura 6.16: Imágenes generadas por distintos métodos del estado del arte (a-c) y los enfoques de este proyecto (d-f) utilizando 100 datos etiquetados

## Capítulo 7

# Conclusiones y trabajo futuro

En este capítulo se presentan las conclusiones derivadas del análisis de los resultados obtenidos. Además, se plantean posibles líneas de mejora y proyecciones de investigación orientadas a optimizar y ampliar el alcance del trabajo desarrollado.

### 7.1. Conclusiones

Este proyecto abordó el entrenamiento semisupervisado de GANs en escenarios de extrema escasez de datos etiquetados, utilizando entre 60 y 1000 ejemplos sobre el dataset MNIST. El objetivo principal fue desarrollar una arquitectura capaz de entrenarse correctamente en estas condiciones restrictivas y evaluar si la hibridación con AE podía mejorar tanto la estabilidad como el desempeño del entrenamiento.

La arquitectura base de la GAN, diseñada siguiendo las mejores prácticas del estado del arte, logró entrenar correctamente incluso con solo 60 datos etiquetados, produciendo generadores con valores de FID consistentemente bajos (medias cercanas a 10) e imágenes de buena calidad visual. Sin embargo, los discriminadores presentaron alta variabilidad en sus precisiones, con desviaciones estándar de entre 22 % y 25 % de las medias obtenidas, evidenciando un entrenamiento poco robusto. Esta dispersión se tradujo en resultados impredecibles, con ejecuciones que alcanzaron precisiones superiores al 90 % mientras que otras caían por debajo del 15 %.

La introducción de AE logró mejoras significativas en comparación con el enfoque base. Ambos enfoques evolutivos lograron estabilizar el entrenamiento mediante la estructura poblacional, que permite eliminar soluciones débiles y reemplazarlas por mejores alternativas, además de evitar el estancamiento al emparejar dinámicamente generadores y discriminadores a lo largo de las generaciones. Todas las distribuciones de precisión obtenidas en los enfoques

evolutivos resultaron normales, salvo una excepción. En contraparte, para el enfoque no evolutivo tres de los cuatro escenarios presentaron distribuciones de precisión no normales.

Más allá de su robustez en el entrenamiento, ambos enfoques evolutivos presentaron diferencias sustanciales en sus resultados finales. El AE mono objetivo, si bien mejoró la consistencia del entrenamiento, presentó precisiones con medianas entre un 6 % y un 22 % inferiores a las presentadas por el enfoque sin AE. Esta limitación se debe a que la función de fitness única combina los losses supervisado y no supervisado, permitiendo que el componente no supervisado, de mayor variabilidad, domine la selección de individuos. Esto resultó en discriminadores optimizados principalmente para distinguir imágenes reales de falsas, pero menos efectivos en clasificación por clases. Como efecto secundario, esta debilidad permitió que los generadores alcanzaran los mejores valores individuales de FID, llegando a mínimos de 5.75 en casos específicos.

Por el contrario, el AE multi objetivo demostró ser consistentemente superior en términos de precisión y estabilidad, constituyendo la contribución principal de este proyecto. Al evaluar de forma independiente los componentes supervisado y no supervisado mediante NSGA-II, se evitó el problema de dominancia del esquema mono objetivo, permitiendo que el entrenamiento optimizara ambos criterios de manera equilibrada. Los resultados fueron contundentes: medias de precisión 10-12 % superiores al enfoque sin AE, medianas con incrementos de hasta 5.43 %, y reducciones de desviación estándar de hasta 97 %. En el escenario con mayor disponibilidad de datos, correspondiente a 1000 datos etiquetados, todas las ejecuciones superaron el 95 % de precisión, demostrando alta reproducibilidad. Además, el análisis de los frentes de Pareto reveló que las ejecuciones con mayor diversidad poblacional en discriminadores obtuvieron mejores desempeños, confirmando que la evaluación independiente de múltiples objetivos es un factor determinante para el éxito del entrenamiento.

Por su parte, los valores de FID se mantuvieron similares entre todos los enfoques sin diferencias estadísticamente significativas, indicando que la estabilidad lograda por los AE no comprometió la calidad de generación.

Aunque las precisiones obtenidas son inferiores a las del estado del arte (96.5 %-99.2 % versus 89 %-96.7 %), estos trabajos no reportan métricas de calidad de generación y fueron entrenados con arquitecturas más complejas durante 400-500 épocas frente a las 100 épocas utilizadas en este proyecto. En este sentido, la solución propuesta mantiene un balance explícito entre ambos objetivos, mientras que los enfoques utilizados en el estado del arte están orientados a maximizar la precisión.

En comparación con ([Sedeño y cols., 2025](#)), un trabajo comparable que emplea AE con GANs, los resultados de este proyecto para escenarios de máxima escasez de datos fueron superiores, con una mediana de 94.15 % versus el 88.00 % obtenido en dicho proyecto, para el escenario de 100 datos etiquetados.

Si bien los resultados obtenidos al integrar AE son superiores a los obtenidos por la arquitectura base, el aumento en el costo computacional al utilizar los enfoques evolutivos, respecto al enfoque base, es proporcional a la cantidad de parejas entrenadas en simultáneo. Este costo adicional se justifica únicamente

para el AE multi objetivo, especialmente en aplicaciones donde la precisión y reproducibilidad son críticas.

Más allá de los resultados específicos sobre MNIST, este proyecto establece un marco de trabajo general y replicable para la hibridación de GANs con AE. El enfoque es independiente de la arquitectura base y puede aplicarse a cualquier GAN existente, incluyendo soluciones del estado del arte. Si bien en este proyecto solo se utilizan dos funciones de fitness a optimizar por cada población, la metodología admite la incorporación de múltiples funciones objetivo según las métricas relevantes del dominio, y es directamente adaptable a otros datasets y tareas.

En síntesis, la hibridación de GANs con AE constituye una estrategia efectiva para el entrenamiento semisupervisado con datos etiquetados extremadamente limitados. El enfoque multi objetivo logró mejoras sustanciales en precisión y estabilidad respecto al enfoque base, manteniendo la calidad de generación y eliminando prácticamente las fallas de entrenamiento. La optimización simultánea de múltiples objetivos mediante NSGA-II demostró ser fundamental, permitiendo balancear diferentes aspectos del entrenamiento y mantener diversidad poblacional. El marco establecido es general, extensible y aplicable a cualquier arquitectura de GANs, ofreciendo un camino claro para mejorar la robustez del entrenamiento semisupervisado a costa de un factor de tiempo adicional.

## 7.2. Trabajo futuro

En esta sección se presentan los puntos relevantes que podrían potenciar este trabajo, los cuales se enfocan en expandir y optimizar la metodología propuesta para la hibridación entre GANs y AE.

### 7.2.1. Integración con arquitecturas del estado del arte

Como fue mencionado en las conclusiones, este proyecto presentó un marco de trabajo que integra el uso de AE con GANs. Este marco de trabajo puede ser aplicado sobre cualquier arquitectura base de GANs, por lo que podría integrarse con cualquier arquitectura del estado del arte para verificar si, aún en escenarios con resultados superiores a los obtenidos en este proyecto, el uso de AE logra una mejora en la estabilidad del entrenamiento y el desempeño de las métricas evaluadas.

### 7.2.2. Extensión a otros datasets

En este proyecto se trabajó sobre el dataset MNIST, el cual es uno de los dataset más simples, utilizado para evaluar la viabilidad de proyectos. Este dataset es simple en el sentido de que, los dígitos representados son similares entre sí, solo hay diez clases y solamente trabaja sobre un único canal (imágenes en blanco y negro). Se sugiere llevar la solución propuesta a diferentes datasets reconocidos para evaluar las mejoras de desempeño posibles.

### 7.2.3. Funciones de fitness de los AE

En este proyecto se trabajó únicamente sobre dos funciones de fitness por población. Aún así, el enfoque evolutivo propuesto admite el uso de tantas funciones de fitness por población como sean necesarias. El uso de más y mejores funciones de fitness, que representen correctamente los objetivos a optimizar por las distintas poblaciones, podría lograr mejoras significativas en el desempeño. Un claro ejemplo de funciones de fitness que podrían mejorarse son las utilizadas por la población de generadores en este proyecto. Ambas funciones de fitness utilizadas para los generadores en este proyecto,  $F_{calidad}$  y  $F_{diversidad}$  presentaron una gran correlación en sus valores, no permitiendo explotar la capacidad del AE multi objetivo. Esto se debe a que, por lo general, se obtuvo un único individuo dominante en cada población de generadores. En contraparte, para la población de discriminadores, donde los objetivos presentaron mayor independencia, los frentes de Pareto estuvieron más poblados, obteniéndose mejores valores de precisión para los frentes compuestos por una mayor cantidad de individuos.

### 7.2.4. Paralelización del entrenamiento

El AE propuesto en este proyecto entrena múltiples parejas de generador y discriminador de forma secuencial, entrenando una pareja por vez. Este proceso consume la mayor cantidad de tiempo de cómputo, siendo el tiempo empleado por los operadores evolutivos despreciable. En consecuencia, los enfoques evolutivos presentaron aumentos en sus tiempos de cómputo proporcionales a la cantidad de parejas entrenadas simultáneamente. Dado que las parejas entrenadas son independientes, el entrenamiento podría realizarse de forma paralela, reduciendo así los tiempos de cómputo asociados al entrenamiento de las múltiples parejas.

# Referencias

- Deb, K., Pratap, A., Agarwal, S., y Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2), 182-197. doi: 10.1109/4235.996017
- Dong, J., y Lin, T. (2019). MarginGAN: Adversarial training in semi-supervised learning. En H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, y R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 32). Curran Associates, Inc. Descargado de [https://proceedings.neurips.cc/paper\\_files/paper/2019/file/517f24c02e620d5a4dac1db388664a63-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2019/file/517f24c02e620d5a4dac1db388664a63-Paper.pdf)
- Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). *Generative adversarial networks*. Descargado de <https://arxiv.org/abs/1406.2661>
- Ioffe, S., y Szegedy, C. (2015). *Batch normalization: Accelerating deep network training by reducing internal covariate shift*. Descargado de <https://arxiv.org/abs/1502.03167>
- Li, W., Wang, Z., Yue, Y., Li, J., Speier, W., Zhou, M., y Arnold, C. W. (2019). *Semi-supervised learning using adversarial training with good and bad samples*. Descargado de <https://arxiv.org/abs/1910.08540>
- Nalluru, N. S. S. (2023). *Semi-supervised generative adversarial networks coevolutionary distributed training* (Tesis Doctoral no publicada). Imperial College London.
- Nesmachnow, S., y Iturriaga, S. (2019). Cluster-uy: Collaborative scientific high performance computing in uruguay.. [https://doi.org/10.1007/978-3-030-38043-4\\_16](https://doi.org/10.1007/978-3-030-38043-4_16)
- Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., y Chen, X. (2016). *Improved techniques for training gans*. Descargado de <https://arxiv.org/abs/1606.03498>
- Sedeño, F., Toutouh, J., y Chicano, F. (2025). *Generate more than one child in your co-evolutionary semi-supervised learning gan*. Descargado de <https://arxiv.org/abs/2504.20560>
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., y Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15, 1929–1958. Descargado de <http://jmlr.org/papers/v15/srivastava14a.html>

- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., y Wojna, Z. (2015). *Rethinking the inception architecture for computer vision*. Descargado de <https://arxiv.org/abs/1512.00567>
- Wang, C., Xu, C., Yao, X., y Tao, D. (2018). *Evolutionary generative adversarial networks*. Descargado de <https://arxiv.org/abs/1803.00657>
- Zieba, M., y Wang, L. (2017). *Training triplet networks with gan*. Descargado de <https://arxiv.org/abs/1704.02227>

# Anexo A

## Anexo 1

### A.1. Configuración paramétrica de la GAN

#### A.1.1. Estadísticas de precisión y FID para la configuración de la arquitectura de la GAN

100 datos etiquetados

	Normal	Mínimo	Media	$\sigma$	Mediana	IQR	p-valor
A1	Sí	9.24	11.39	1.06	11.22	1.41	0.9266
A2	Sí	9.84	11.51	1.09	11.53	1.51	0.8856
A3	Sí	9.49	11.76	1.27	11.92	1.79	0.9439
A4	Sí	9.35	13.05	2.56	12.34	2.70	0.3630
A5	Sí	10.16	12.27	0.73	12.28	0.66	0.6474
A6	Sí	10.60	12.77	1.35	12.62	1.77	0.8999

Cuadro A.1: Estadísticas de FID obtenidas para las distintas configuraciones de la arquitectura utilizando 100 datos etiquetados

	Normal	Máximo	Media	$\sigma$	Mediana	IQR	p-valor
A1	No	95.71	74.60	23.85	86.70	40.79	0.0486
A2	No	95.77	85.34	15.49	93.32	12.16	0.0263
A3	No	96.24	85.52	16.46	94.12	14.95	0.0058
A4	Sí	91.31	72.19	19.88	76.74	18.41	0.1290
A5	No	90.21	65.30	24.73	79.64	42.92	0.0195
A6	No	91.61	72.83	19.37	80.22	23.28	0.0276

Cuadro A.2: Estadísticas de precisión obtenidas para las distintas configuraciones de la arquitectura utilizando 100 datos etiquetados

### 500 datos etiquetados

		Normal	Mínimo	Media	$\sigma$	Mediana	IQR	p-valor
<i>A1</i>	No	10.31	75.27	128.28	13.36	33.19	0.0008	
<i>A2</i>	No	10.55	59.37	105.79	12.24	37.23	0.0004	
<i>A3</i>	No	9.37	18.95	34.99	12.32	2.16	0.0000	
<i>A4</i>	Sí	9.92	12.55	1.31	12.46	1.61	0.9972	
<i>A5</i>	Sí	9.95	12.18	1.32	11.96	1.78	0.7459	
<i>A6</i>	Sí	9.13	12.24	1.37	12.26	1.12	0.6190	

Cuadro A.3: Estadísticas de FID obtenidas para las distintas configuraciones de la arquitectura utilizando 500 datos etiquetados

		Normal	Máximo	Media	$\sigma$	Mediana	IQR	p-valor
<i>A1</i>	Sí	95.43	62.77	34.34	78.41	63.19	0.0718	
<i>A2</i>	Sí	94.96	65.17	32.64	81.57	53.00	0.0534	
<i>A3</i>	Sí	94.09	77.40	18.44	85.18	18.77	0.1483	
<i>A4</i>	No	93.95	85.36	11.45	90.22	6.79	0.0195	
<i>A5</i>	Sí	93.59	85.83	8.87	88.78	8.68	0.1976	
<i>A6</i>	No	94.30	81.18	19.44	90.23	10.14	0.0037	

Cuadro A.4: Estadísticas de precisión obtenidas para las distintas configuraciones de la arquitectura utilizando 500 datos etiquetados

### 1000 datos etiquetados

		Normal	Mínimo	Media	$\sigma$	Mediana	IQR	p-valor
<i>A1</i>	No	9.59	19.64	45.73	11.56	1.64	0.0000	
<i>A2</i>	Sí	9.04	11.87	1.67	11.44	2.01	0.4975	
<i>A3</i>	Sí	9.58	11.84	1.14	11.72	1.48	0.9417	
<i>A4</i>	Sí	9.08	11.88	1.74	11.77	2.24	0.8369	
<i>A5</i>	Sí	9.93	11.86	1.14	11.75	1.37	0.6649	
<i>A6</i>	Sí	9.80	12.54	1.01	12.66	1.03	0.9241	

Cuadro A.5: Estadísticas de FID obtenidas para las distintas configuraciones de la arquitectura utilizando 1000 datos etiquetados

		Normal	Máximo	Media	$\sigma$	Mediana	IQR	p-valor
<i>A1</i>	No	96.94	85.66	17.43	92.67	13.68	0.0292	
<i>A2</i>	No	96.66	89.62	12.79	95.09	6.98	0.0095	
<i>A3</i>	No	96.66	87.53	13.54	94.98	13.56	0.0235	
<i>A4</i>	No	95.39	84.88	15.44	92.52	11.45	0.0258	
<i>A5</i>	No	95.54	85.86	13.95	91.47	6.59	0.0043	
<i>A6</i>	Sí	95.29	81.27	17.54	90.81	21.88	0.0521	

Cuadro A.6: Estadísticas de precisión obtenidas para las distintas configuraciones de la arquitectura utilizando 1000 datos etiquetados

### A.1.2. Hipervolúmenes de las configuraciones finales de la arquitectura

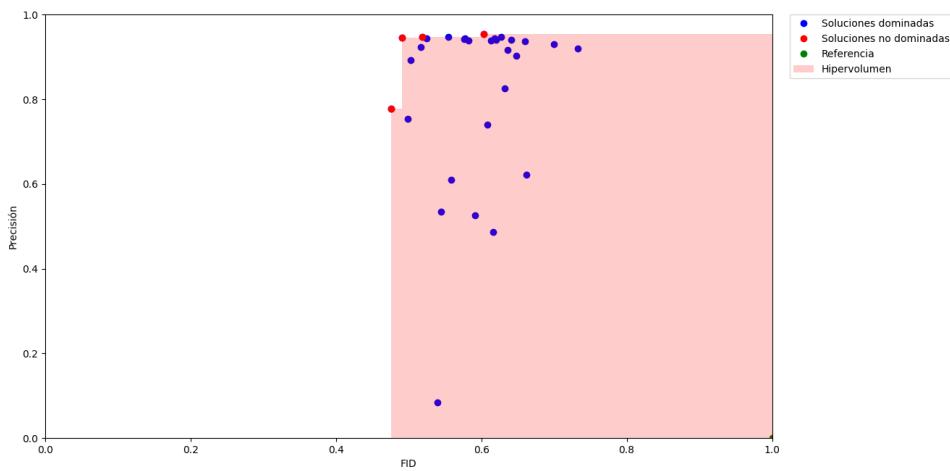


Figura A.1: Hipervolumen y frente de pareto normalizado para la configuración A3 de la arquitectura utilizando 100 datos etiquetados

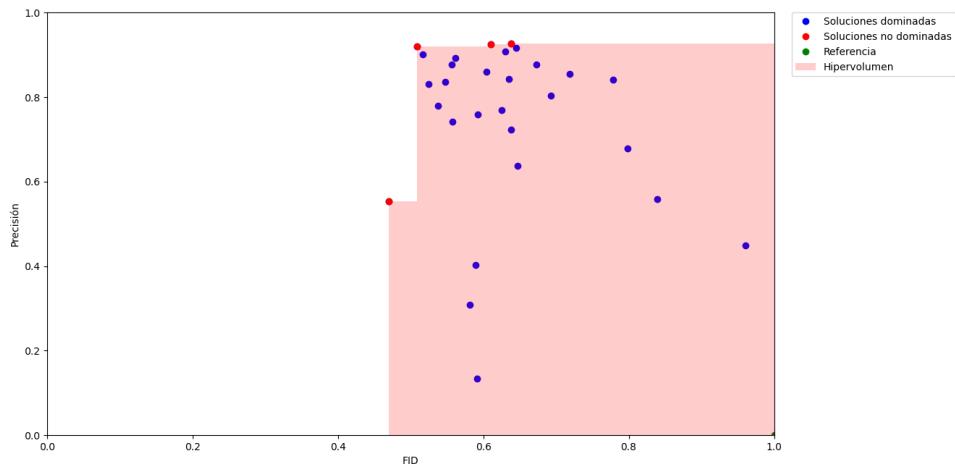


Figura A.2: Hipervolumen y frente de pareto normalizado para la configuración A3 de la arquitectura utilizando 500 datos etiquetados

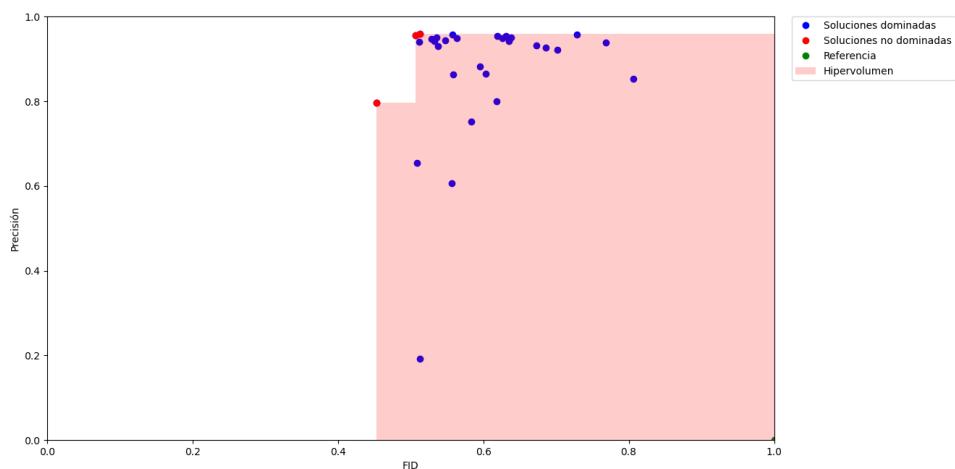


Figura A.3: Hipervolumen y frente de pareto normalizado para la configuración A2 de la arquitectura utilizando 1000 datos etiquetados

### A.1.3. Imágenes generadas en la configuración paramétrica para configuraciones sin uso de batch normalization

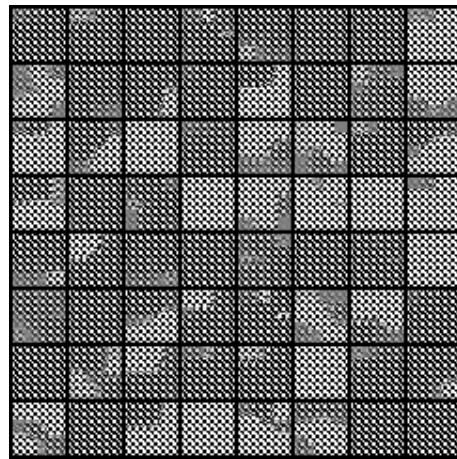


Figura A.4: 64 imágenes generadas por un generador entrenado sin batch normalization utilizando tamaño de batch de 64

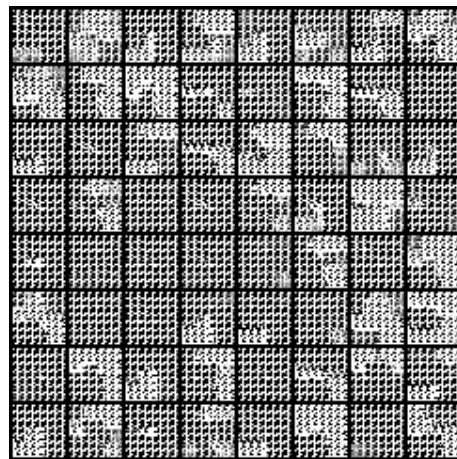


Figura A.5: 64 imágenes generadas por un generador entrenado sin batch normalization utilizando tamaño de batch de 100

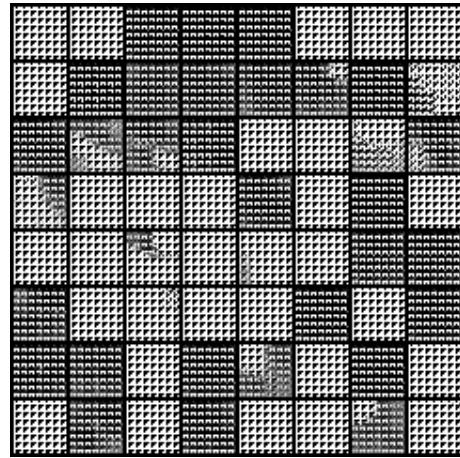


Figura A.6: 64 imágenes generadas por un generador entrenado sin batch normalization utilizando tamaño de batch de 500

#### A.1.4. Hipervolúmenes de la configuración final del entrenamiento con 1000 datos etiquetados

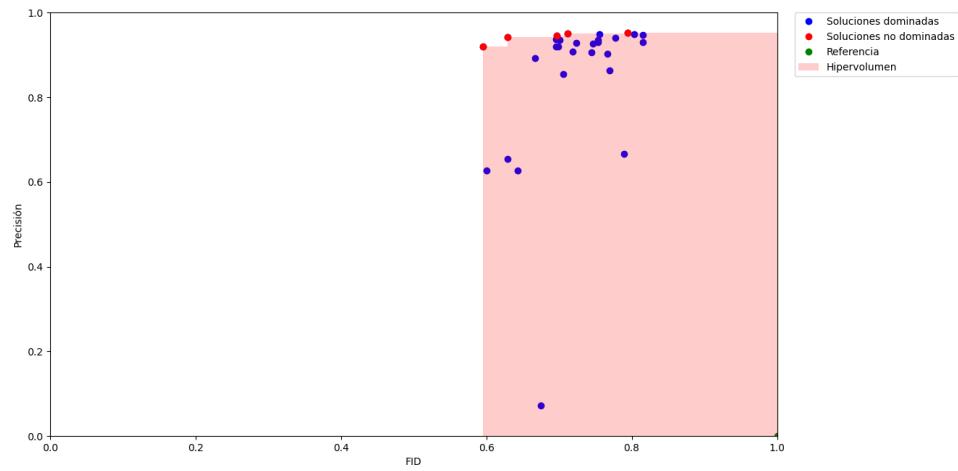


Figura A.7: Hipervolumen y frente de pareto normalizado para la configuración E5 del entrenamiento

### A.1.5. Hipervolúmenes de la configuración de los datos etiquetados

#### 100 datos etiquetados

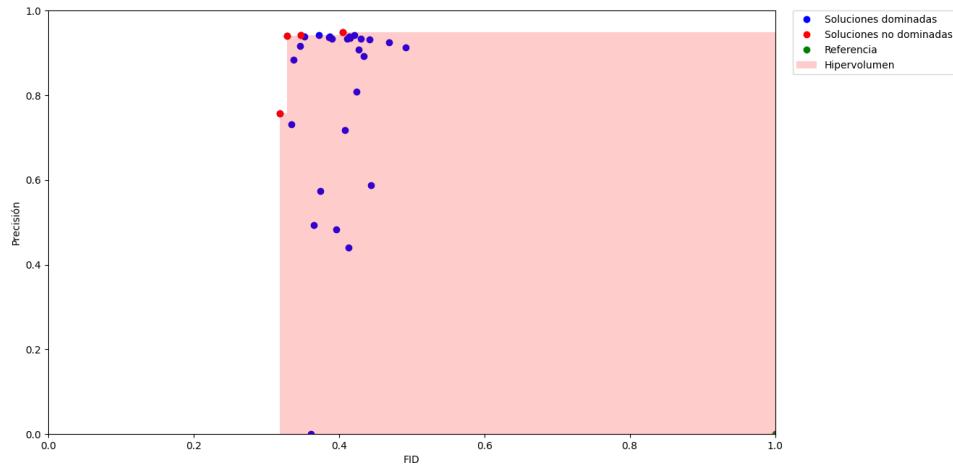


Figura A.8: Hipervolumen y frente de pareto normalizado para la configuración L1 del uso de datos etiquetados con 100 datos

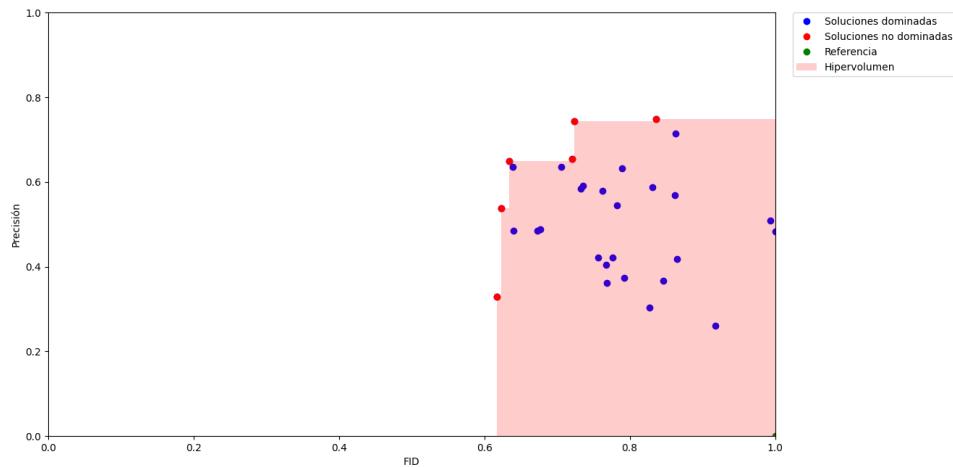


Figura A.9: Hipervolumen y frente de pareto normalizado para la configuración L2 del uso de datos etiquetados con 100 datos

## 500 datos etiquetados

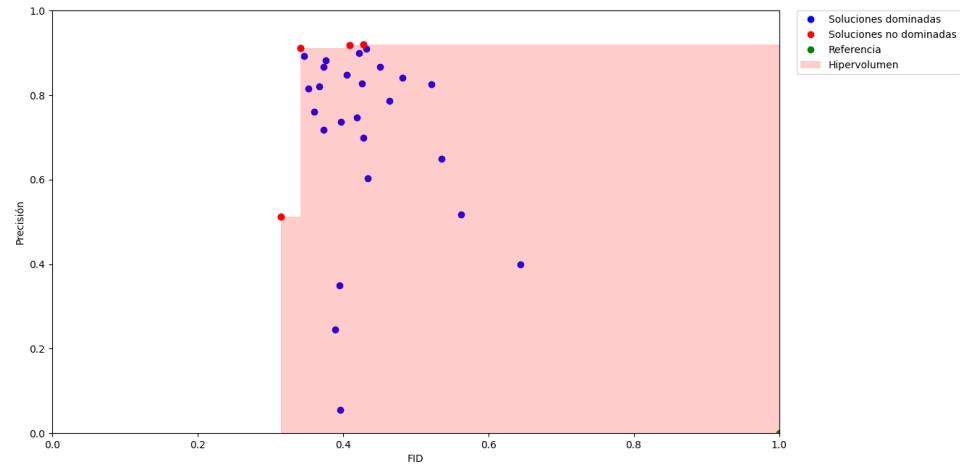


Figura A.10: Hipervolumen y frente de pareto normalizado para la configuración L1 del uso de datos etiquetados con 500 datos

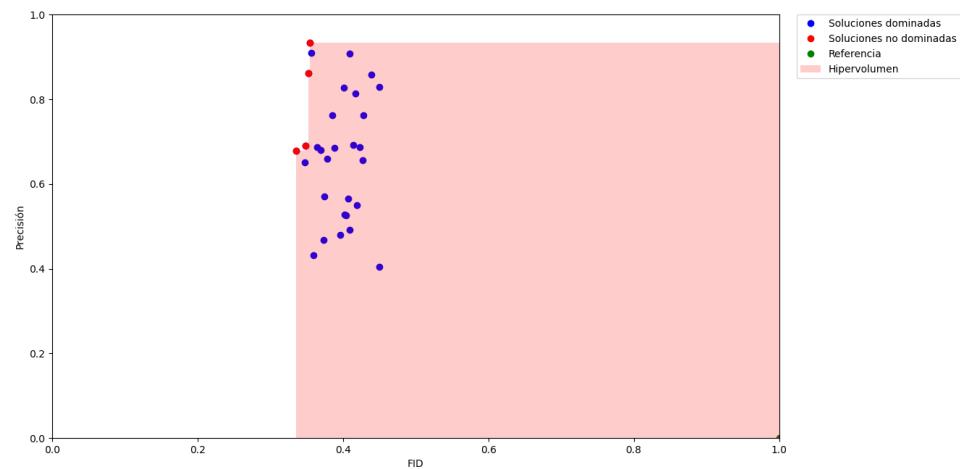


Figura A.11: Hipervolumen y frente de pareto normalizado para la configuración L2 del uso de datos etiquetados con 500 datos

## 1000 datos etiquetados

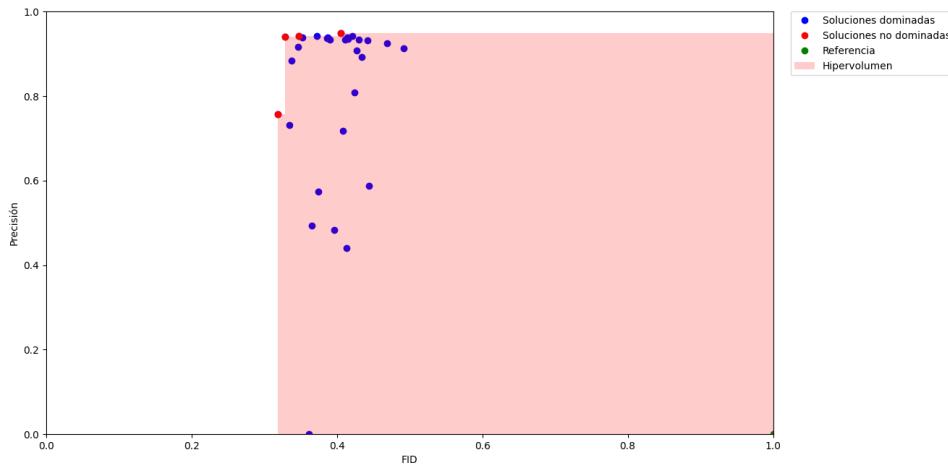


Figura A.12: Hipervolumen y frente de pareto normalizado para la configuración L1 del uso de datos etiquetados con 100 datos

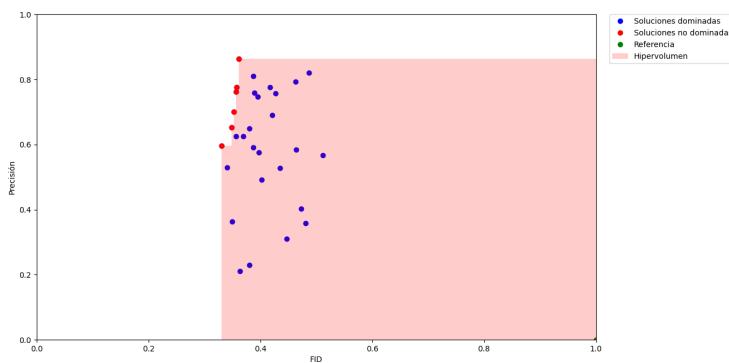


Figura A.13: Hipervolumen y frente de pareto normalizado para la configuración L2 del uso de datos etiquetados con 1000 datos

## A.2. Evaluación del enfoque sin AE

### A.2.1. Imágenes generadas

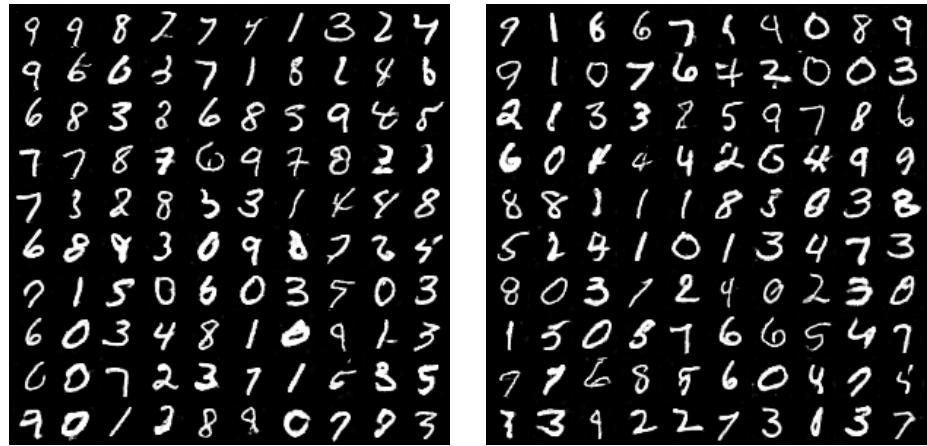


Figura A.14: Ejemplos de imágenes generadas en las ejecuciones con mejor (izquierda) y peor (derecha) valor de FID utilizando 60 datos etiquetados para el enfoque sin AE en la etapa de validación

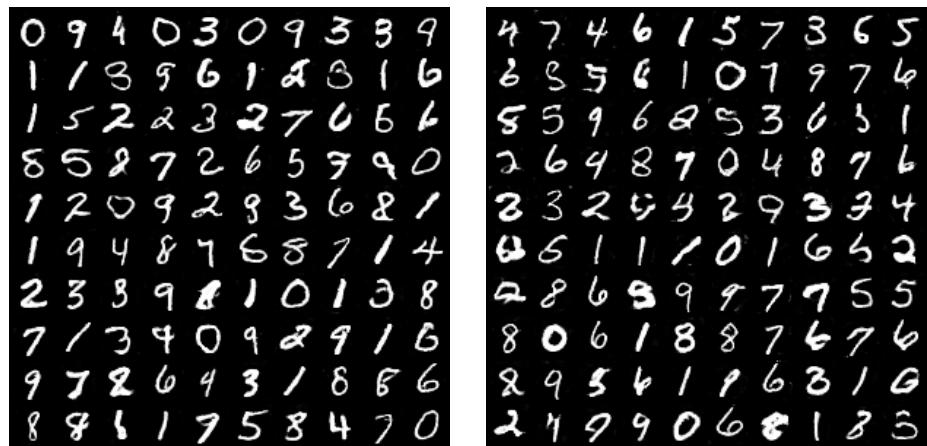


Figura A.15: Ejemplos de imágenes generadas en las ejecuciones con mejor (izquierda) y peor (derecha) valor de FID utilizando 500 datos etiquetados para el enfoque sin AE en la etapa de validación

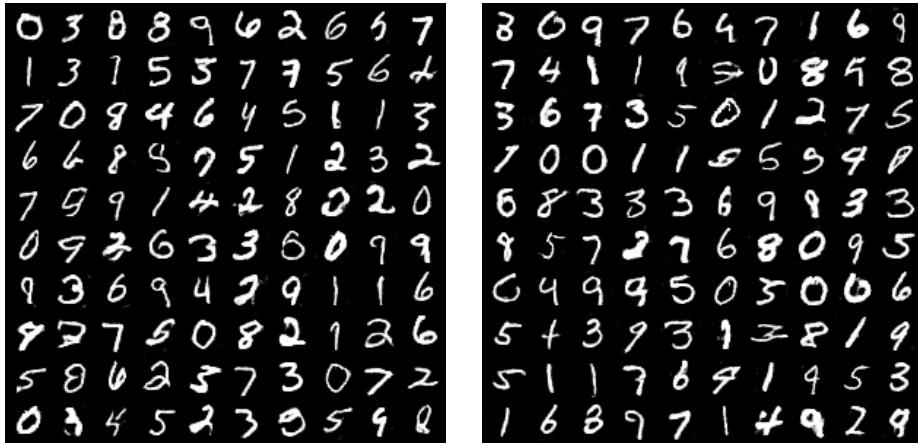


Figura A.16: Ejemplos de imágenes generadas en las ejecuciones con mejor (izquierda) y peor (derecha) valor de FID utilizando 1000 datos etiquetados para el enfoque sin AE en la etapa de validación

### A.3. Evaluación del enfoque con AE mono objetivo

#### A.3.1. Imágenes generadas

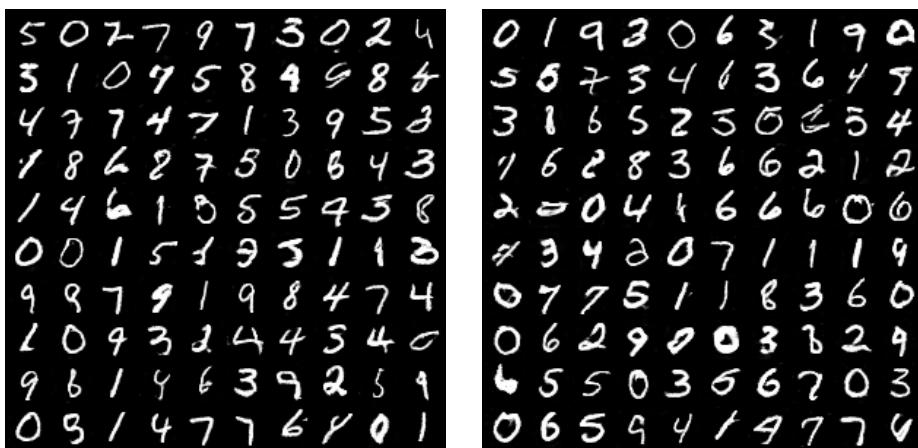


Figura A.17: Ejemplos de imágenes generadas en las ejecuciones con mejor (izquierda) y peor (derecha) valor de FID utilizando 100 datos etiquetados para el AE mono objetivo en la etapa de validación

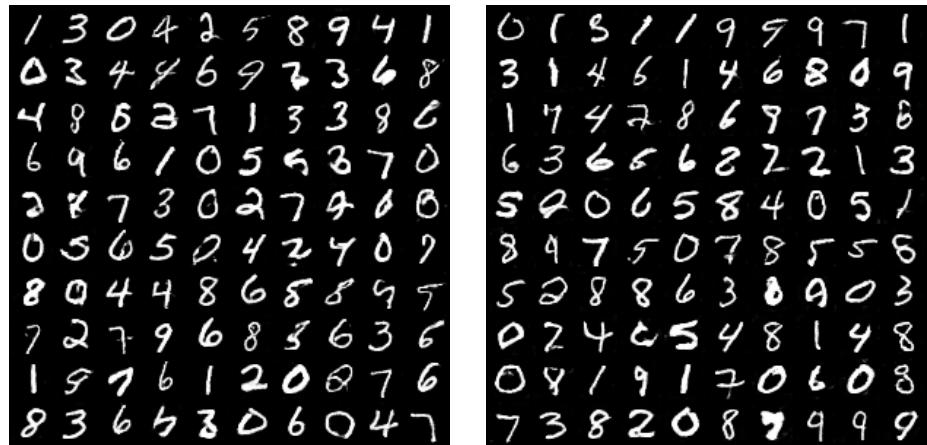


Figura A.18: Ejemplos de imágenes generadas en las ejecuciones con mejor (izquierdo) y peor (derecho) valor de FID utilizando 500 datos etiquetados para el AE mono objetivo en la etapa de validación

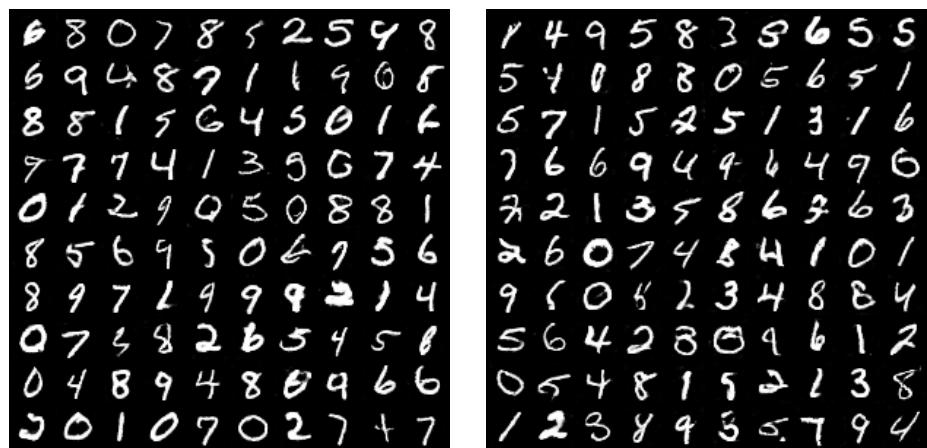


Figura A.19: Ejemplos de imágenes generadas en las ejecuciones con mejor (izquierdo) y peor (derecho) valor de FID utilizando 1000 datos etiquetados para el AE mono objetivo en la etapa de validación

## A.4. Evaluación del enfoque con AE multi objetivo

### A.4.1. Imágenes generadas

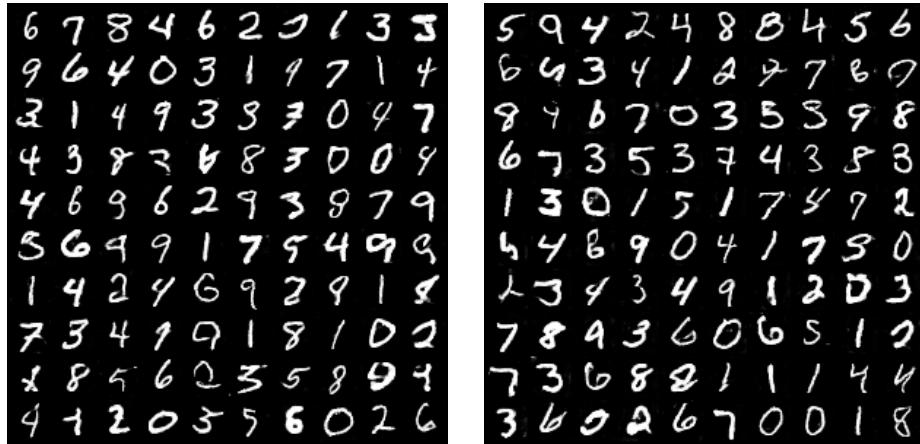


Figura A.20: Ejemplos de imágenes generadas en las ejecuciones con mejor (izquierda) y peor (derecha) valor de FID utilizando 60 datos etiquetados para el AE multi objetivo en la etapa de validación

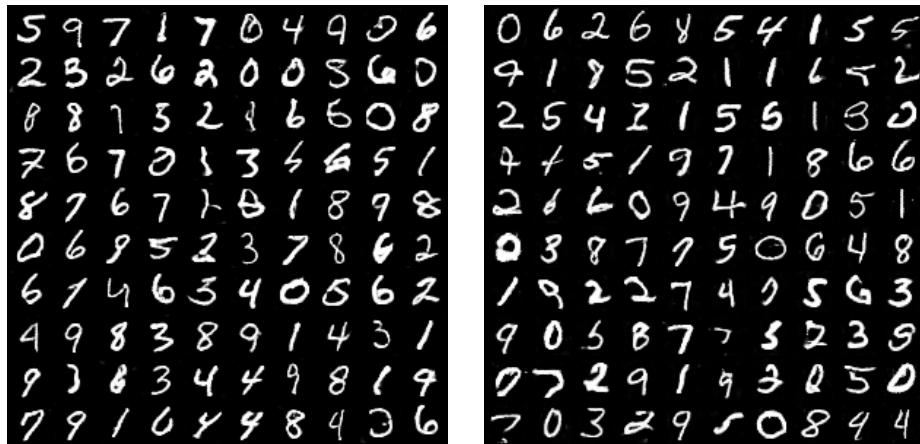


Figura A.21: Ejemplos de imágenes generadas en las ejecuciones con mejor (izquierda) y peor (derecha) valor de FID utilizando 100 datos etiquetados para el AE multi objetivo en la etapa de validación



Figura A.22: Ejemplos de imágenes generadas en las ejecuciones con mejor (izquierda) y peor (derecha) valor de FID utilizando 1000 datos etiquetados para el AE multi objetivo en la etapa de validación

#### A.4.2. Composición de frentes de Pareto de los discriminadores

#Discriminadores	Cantidad de datos etiquetados			
	60	100	500	1000
1	79.54	86.50	92.13	-
2	87.68	92.19	90.21	95.84
3	90.27	94.05	94.81	96.54
4	92.67	94.73	95.31	96.89

Cuadro A.7: Medias de precisiones obtenidas por el AE multi objetivo según cantidad de discriminadores que componen el frente de Pareto final para cada escenario. No se obtuvieron ejecuciones con un solo discriminador para el escenario de 1000 datos

## A.5. Análisis de resultados

### A.5.1. Boxplots de precisiones

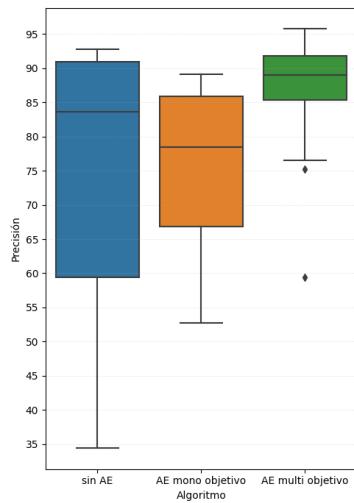


Figura A.23: Boxplots de las precisiones obtenidas en la etapa de validación para los distintos algoritmos estudiados utilizando 60 datos etiquetados

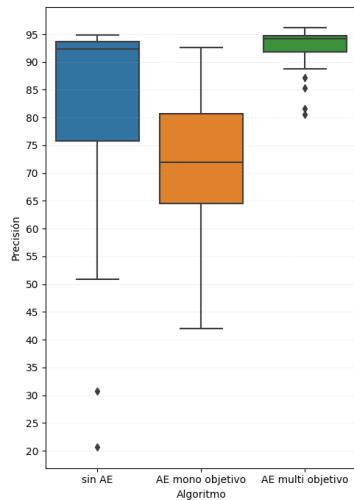


Figura A.24: Boxplots de las precisiones obtenidas en la etapa de validación para los distintos algoritmos estudiados utilizando 100 datos etiquetados

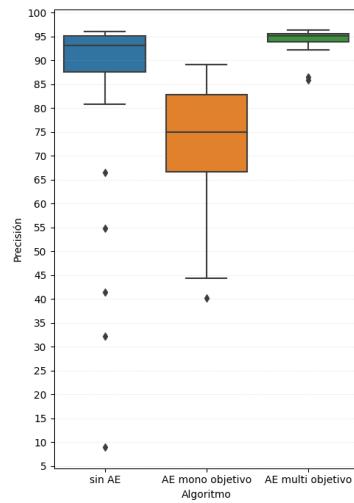


Figura A.25: Boxplots de las precisiones obtenidas en la etapa de validación para los distintos algoritmos estudiados utilizando 500 datos etiquetados

### A.5.2. Boxplots de FID

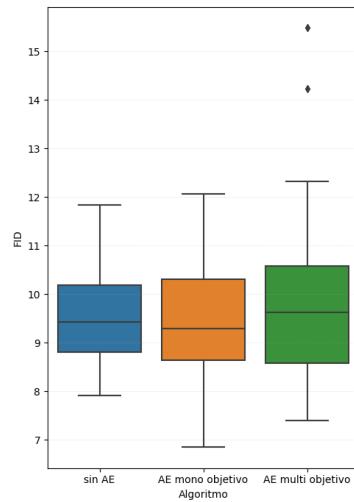


Figura A.26: Boxplots de FIDs obtenidas en la etapa de validación para los distintos algoritmos estudiados utilizando 100 datos etiquetados

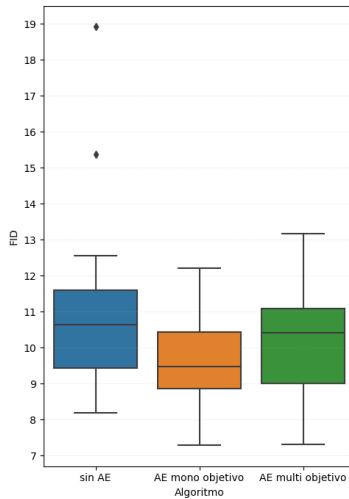


Figura A.27: Boxplots de FIDs obtenidas en la etapa de validación para los distintos algoritmos estudiados utilizando 500 datos etiquetados

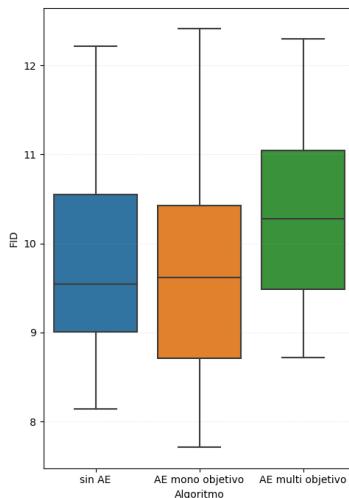


Figura A.28: Boxplots de FIDs obtenidas en la etapa de validación para los distintos algoritmos estudiados utilizando 1000 datos etiquetados