Método para la síntesis de paisajes de optimización bidimensionales basado en redes adversarias generativas

Mariana Medina Muñoz,

Universidad Distrital Francisco José de Caldas
Facultad de Ingeniería
Ingeniería Electrónica
Bogotá D.C
2023

Método para la síntesis de paisajes de optimización bidimensionales basado en redes adversarias generativas

Mariana Medina Muñoz,

Trabajo de grado para optar al título de:
Ingeniera Electrónica en la modalidad de monografía

Director:
Dr.Ing. Miguel Alberto Melgarejo Rey,

Grupo de Investigación:
Laboratorio de Automática e Inteligencia Computacional

Universidad Distrital Francisco José de Caldas
Facultad de Ingeniería
Ingeniería Electrónica
Bogotá D.C
2023

Agradecimientos

A mi familia por todo el apoyo incondicional que han puesto en mi, mis padres que han estado siempre y han deseado verme cumplir la meta de ser ingeniera, mis hermanos por ser mi referente intelectual más cercanos.

A la Universidad Distrital, especialmente al proyecto de Ingeniería Electrónica y al Laboratorio de Automática e Inteligencia Computacional (LAMIC), por acogerme en sus instalaciones y brindarme ese apoyo en este proceso de aprendizaje.

Al profesor Miguel, mi director, mentor y guía en este camino, quien me brindo sus enseñanzas y apoyo. A mis amigos y compañeros por aportarme con su apoyo, por aportarme con sus puntos de vista y palabras de animo.

A todos y cada una de las personas que hicieron parte de este proceso de formación, MUCHAS GRACIAS.

TABLA DE CONTENIDOS

RESUMEN 7

I. GENERALIDADES 8
1.1. Planteamiento del problema . . . 8
1.2. Objetivos . . . 9
1.2.1. Objetivo general . . . 9
1.2.2. Objetivos específicos . . . 9
1.3. Solución propuesta . . . 10

II. MARCO TEÓRICO 11
2.1. Teoría de la optimización . . . 11
2.1.1. Definiciones . . . 11
2.1.2. Métodos de optimización . . . 13
2.2. Funciones de prueba . . . 14
2.3. Antecedentes de funciones de prueba . . . 16
2.3.1. A new class of test functions for global optimization . . . 16
2.3.2. Test problem generator for unconstrained global optimization . . . 17
2.3.3. New hard benchmark functions for global optimization . . . 18
2.4. Redes Neuronales GAN . . . 18
2.4.1. Función de perdidas . . . 22
2.5. Parámetros iniciales para una DCGAN . . . 23
2.6. Equilibrio de Nash . . . 24
2.7. Antecedentes de GANs como generadoras de imágenes . . . 25
2.7.1. Realistic and Textured Terrain Generation using GANs . . . 25
2.7.2. Fast Cosmic Web Simulations with Generative Adversarial Networks . . . 25
2.8. Data Augmentation . . . 27
2.9. Pytorch y herramientas de desarrollo . . . 28
2.9.1. Tensores . . . 28

III. CREACIÓN DE LA BASE DE DATOS 29
3.1. Búsqueda e implementación de los paisajes de optimización . . . 29
3.2. Conformación de las bases de datos . . . 29
3.2.1. Base 1 . . . 30
3.2.2. Base 2 . . . 33
3.2.3. Base 3 . . . 34
3.2.4. Base 4 . . . 35
3.2.5. Aplicación Data Augmentation . . . 36

IV. RED ADVERSARIA GENERATIVA (DCGAN) 37
4.1. Parámetros de entrada: . . . 37
4.2. Procesamiento de la base de datos: . . . 37
4.3. Inicialización de los pesos: . . . 39
4.4. Generador: . . . 39
4.5. Discriminador: . . . 42
4.6. Realimentación de la red: . . . 45
4.7. Entrenamiento de la red: . . . 46

4.7.1. Entrenar al Discriminador . . . 47
4.7.2. Entrenar al Generador . . . 47
4.8. Visualización de los paisajes . . . 48

V. RESULTADOS 49
5.1. Formas de validación . . . 49
5.1.1. Estadísticos de la DCGAN . . . 49
5.1.2. Diagrama de cajas y bigotes . . . 50
5.2. Metodología de experimentación . . . 51
5.3. Parámetros de entrenamiento . . . 51
5.4. Resultados visuales . . . 52
5.4.1. Base 1 . . . 53
5.4.2. Base 2 . . . 54
5.4.3. Base 3 . . . 56
5.4.4. Base 4 . . . 58
5.4.5. Base 5 . . . 60
5.4.6. Base 6 . . . 61
5.4.7. Base 7 . . . 63
5.4.8. Base 8 . . . 64
5.5. Comparación: . . . 66
5.5.1. Perdidas en el Generador(PG) y en el Discriminador (PD) . . . 66
5.5.2. Salida promedio del Discriminador para el lote real($D(x)$) y para el lote falso($D(G(z))$) 67

VI. CONCLUSIONES 71
6.1. Aportes originales . . . 72
6.2. Trabajos Futuros . . . 72

REFERENCIAS 73

Apéndice A: Anexo: Paisajes 76
1.1. Base 1 . . . 76
1.2. Base 2 . . . 77
1.3. Base 3 . . . 78
1.4. Base 4 . . . 79
1.5. Base 5 . . . 80
1.6. Base 6 . . . 81
1.7. Base 7 . . . 82
1.8. Base 8 . . . 83

LISTA DE TABLAS

TABLA. I. Métodos de Optimización 14
TABLA. II. Características de las bases de datos. 30
TABLA. III. Parámetros de entrada. 38
TABLA. IV. Arquitectura Generador 40
TABLA. V. Arquitectura Discriminador 45
TABLA. VI. Parámetros de entrada 52
TABLA. VII. D(x) 53
TABLA. VIII. Perdidas en el Generador y en el Discriminador 67
TABLA. IX. Salidas promedio del Discriminador para el lote falso y el lote real 68
TABLA. X. Coeficiente de variación para el estadístico D(x) 68

LISTA DE FIGURAS

Fig. 1. Diagrama de bloques del proyecto 10
Fig. 2. Funciones BEALE. Tomado de [22] 15
Fig. 3. Funciones SPHERE. Tomado de [22] 15
Fig. 4. Funciones RASTRIGIN. Tomado de [22] 15
Fig. 5. Funciones de componentes unidimensionales. Tomado de[3] 17
Fig. 6. Contornos del problema de nivel 2. Tomado de[21] 17
Fig. 7. Las superficies 3d de las funciones propuestas 3. Tomado de[25] 18
Fig. 8. Arquitectura GAN básica. Tomado de[8] 18
Fig. 9. Arquitectura GAN básica. Tomado de[8] 19
Fig. 10. Arquitectura de una CGAN. Tomado de[29] 20
Fig. 11. Arquitectura del generador de una DCGAN. Tomado de[29] 21
Fig. 12. A la izquierda, un bloque MRU utilizado por la SketchyGAN. A la derecha, arquitectura de la SketchyGAN. Tomado de[29] 21
Fig. 13. a)Región de entrenamiento b) Región generada. Tomado de[37] 25
Fig. 14. Muestras de la simulación de cuerpo N y de GAN para el tamaño de caja de 100 Mpc. Tomado de[38] 26
Fig. 15. Esquema Data Augmentation[41] 27
Fig. 16. Característica Modalidad. Tomado de [22] 30
Fig. 17. Característica Separabilidad. Tomado de [22] 31
Fig. 18. Característica Escalabilidad. Tomado de [22] 31
Fig. 19. Característica Escalabilidad. Tomado de [22] 32
Fig. 20. Característica Continuidad. Tomado de [22] 32
Fig. 21. Ejemplos de paisajes ruidosos 33
Fig. 22. Ejemplos de paisajes para la Base 3 34
Fig. 23. Ejemplos de paisajes para la Base 4 35
Fig. 24. Ejemplo aplicando Data Augmentation 36
Fig. 25. Ejemplo transformación de la base de datos 39
Fig. 26. Arquitectura del Generador. Tomado [42] 40
Fig. 27. Función ReLU. Tomado de [42] 41
Fig. 28. Función Tanh. Tomado de [42] 42
Fig. 29. Arquitectura del Discriminador. Tomado de [46][39] 43
Fig. 30. Función LEAKY RELU. Tomado de [42] 44
Fig. 31. Función Sigmoid. Tomado de [42] 44
Fig. 32. Perdidas durante el entrenamiento Base 1 53
Fig. 33. Paisajes tridimensionales Base 1 54
Fig. 34. Paisajes tridimensionales Base 1 54
Fig. 35. Perdidas durante el entrenamiento Base 2 55
Fig. 36. Paisajes tridimensionales Base 2 55
Fig. 37. Paisajes bidimensionales Base 2 56
Fig. 38. Perdidas durante el entrenamiento Base 3 57
Fig. 39. Paisajes tridimensionales Base 3 57
Fig. 40. Paisajes bidimensionales Base 3 58
Fig. 41. Perdidas durante el entrenamiento Base 4 58
Fig. 42. Paisajes tridimensionales Base 4 59
Fig. 43. Paisajes bidimensionales Base 4 59
Fig. 44. Perdidas durante el entrenamiento Base 5 60
Fig. 45. Paisajes tridimensionales Base 5 60

Fig. 46. Paisajes bidimensionales Base 5 61
Fig. 47. Perdidas durante el entrenamiento Base 6 61
Fig. 48. Paisajes tridimensionales Base 6 62
Fig. 49. Paisajes bidimensionales Base 6 62
Fig. 50. Perdidas durante el entrenamiento Base 7 63
Fig. 51. Paisajes tridimensionales Base 7 63
Fig. 52. Paisajes bidimensionales Base 7 64
Fig. 53. Perdidas durante el entrenamiento Base 8 65
Fig. 54. Paisajes tridimensionales Base 8 65
Fig. 55. Paisajes bidimensionales Base 8 66
Fig. 56. Diagrama de cajas y bigotes para la variable D(x) 69
Fig. 57. Paisajes Tridimensionales Base 1 76
Fig. 58. Paisajes Tridimensionales Base 2 77
Fig. 59. Paisajes Tridimensionales Base 3 78
Fig. 60. Paisajes Tridimensionales Base 4 79
Fig. 61. Paisajes Tridimensionales Base 5 80
Fig. 62. Paisajes Tridimensionales Base 6 81
Fig. 63. Paisajes Tridimensionales Base 7 82
Fig. 64. Paisajes Tridimensionales Base 8 83

RESUMEN

Se realizó la implementación de una red neuronal artificial, específicamente una Deep Convolutional Adversarial Network (DCGAN), con el propósito de generar paisajes bidimensionales de optimización. Se logró aumentar la cantidad de paisajes y su diversidad.

Se inició con la conformación de una base de datos a partir de la literatura, para posteriormente producir otras bases de datos con características específicas. Se conformaron 8 bases de datos, las 4 primeras se dividieron en: Original, Sin ruido, Con un solo mínimo y Con varios mínimos. Las 4 bases de datos posteriores se crearon a partir de la aplicación de un Data Augmentation a las 4 bases de datos nombradas anteriormente.

Luego, se ejecutó el algoritmo varias veces hasta encontrar los parámetros adecuados para el funcionamiento de la red, esto se realizó para cada una de las bases de datos. Una vez escogidos los parámetros para cada una de las bases de datos, se ejecutó nuevamente el algoritmo varias veces para realizar un análisis de los resultados y también se obtuvieron varios paisajes de optimización con diversidad en sus características, con similitud a la base de datos original y a su vez, con presencia de información diferente.

De toda la experimentación se valida que es posible la generación de nuevos paisajes bidimensionales de optimización a partir de una red adversaria generativa, dejando como trabajo futuro el encontrar las funciones o expresiones matemáticas que producirán dichos paisajes.

I. GENERALIDADES

1.1. Planteamiento del problema

La teoría de la optimización se refiere a un cuerpo de resultados matemáticos y métodos numéricos para la identificación del mejor candidato en una colección de alternativas. En la actualidad, es común que la teoría de optimización sea aplicada a ramas de la ciencia, la industria, el comercio, la ingeniería y entre otros[1]. Muchos de los problemas que abarca el mundo de la ingeniería son cada vez más complejos, es por ello que se necesitan algoritmos de optimización que cuenten con un mejor desempeño[2].

Cuando se habla de algoritmos de optimización también se abarcan los problemas de prueba o funciones de prueba, los cuales son tareas importantes y no triviales. Estos problemas deben evidenciar las fortalezas y debilidades de los algoritmos de optimización[3]. Ahora, si se habla del campo de la inteligencia computacional, se podría decir que está demasiado potenciado hacia el análisis empírico. De acuerdo con el teorema NFL(No Free Lunch), ningún algoritmo puede mostrar un desempeño superior con respecto a los otros [4]. Es por esta razón que es importante tener diferentes problemas de prueba para determinar el desempeño general de los algoritmos, y es aquí donde se evidencia lo problemático que es seleccionar el algoritmo adecuado para un problema en específico[5].

En la literatura se encuentran varios trabajos que necesitan de las funciones de referencia para la validación de sus hipótesis, que van desde algoritmos genéticos adaptativos para optimización global en funciones matemáticas[1] hasta algoritmos de búsqueda aleatoria de agua en manadas de elefantes[6]. Pero, aun así, existen trabajos acerca de la deficiencia en la cobertura del espacio del problema en los conjuntos de prueba seleccionados en la literatura[5]; es decir, la elección de los problemas de referencia adecuados varia con el tipo de algoritmo de optimización, haciendo difícil la selección de un conjunto de referencia para la comparación de los algoritmos en cuestiones de desempeño.

La teoría de optimización tiene un papel fundamental en el campo de la IC (Inteligencia Computacional). Esto se debe al desarrollo de los diferentes modelos que esta última contiene. Sin embargo, en trabajos recientes se presenta una forma diferente de colaboración Hombre - Maquina para que los modelos de aprendizaje automático respalden las matemáticas de una manera que nunca antes se había visto y, de está manera, los matemáticos puedan descubrir pistas que podrían haber sido difíciles de encontrar sin ayuda de las maquinas[7].

Dentro del mundo de la IC, la popularidad y potencia de las GANs(Generative Adversarial Networks) se debe a su capacidad de aprender a imitar cualquier distribución de datos[8], logrando lo anterior en varios dominios como lo son las imágenes, la musica, el habla y demás. Los avances de este tipo de red han ido aumentando en los últimos años. Si se habla en el dominio de las imágenes, se podría hablar desde trabajos centrados en la traducción de imágenes a ilustraciones, creando imágenes con estilo pero

conservando el contenido[9], hasta investigaciones con el objetivo de aumentar la resolución de imágenes; un ejemplo es la mejora de la resolución en las resonancias magnéticas[10].

A la hora de hablar de la teoría de la optimización, más concretamente, en los problemas de prueba, se encontraron trabajos donde se crean o construyen nuevas funciones de prueba a partir de las combinaciones lineales de otros problemas o, en otros casos, con la inclusión de osciladores y parámetros que cumplen con ecuaciones especificas (componentes unidimensionales)[3][5], mas a la fecha la autora no ha encontrado trabajos donde el espacio o generador de las funciones de prueba utilice una red neuronal u otro método de IC.

Por lo descrito anteriormente, se formula el siguiente interrogante: ¿cómo las redes neuronales GAN podrían emplearse para generar paisajes bidimensionales de optimización ?

Dado que en la actualidad hay diversidad en los trabajos que tienen como tema central la inteligencia computacional, puede ser que los paisajes encontrados con un modelo basado en GAN brinden un mejor conjunto de prueba para medir el desempeño de los algoritmos de optimización y, de esta manera, disminuir la cantidad de ensayos necesarios para realizar una selección.

1.2. Objetivos

1.2.1) Objetivo general:

Desarrollar un método basado en una red neuronal artificial del tipo adversaria generativa (GAN) que permita obtener paisajes bidimensionales de optimización.

1.2.2) Objetivos específicos:

- Seleccionar los paisajes bidimensionales de optimización necesarios para la creación de la base de datos, la cual contará con un número de funciones de al menos 25 problemas de prueba.[1]

- Diseñar un algoritmo que tenga como base una red neuronal del tipo GAN, para la generación de paisajes bidimensionales a partir de la base de datos construida.

- Seleccionar una batería de estadísticos con los cuales sea posible caracterizar los paisajes obtenidos por el algoritmo.

- Caracterizar un conjunto significativo de paisajes bidimensionales de optimización obtenidos por el algoritmo a partir de la batería de estadísticos seleccionada.

[1]Numero de funciones de prueba tomado de: [5]

1.3. Solución propuesta

El trabajo se divide según el diagrama de bloques que se muestra en la figura 7. Este trabajo permite generar paisajes bidimensionales de optimización y caracterizarlos.

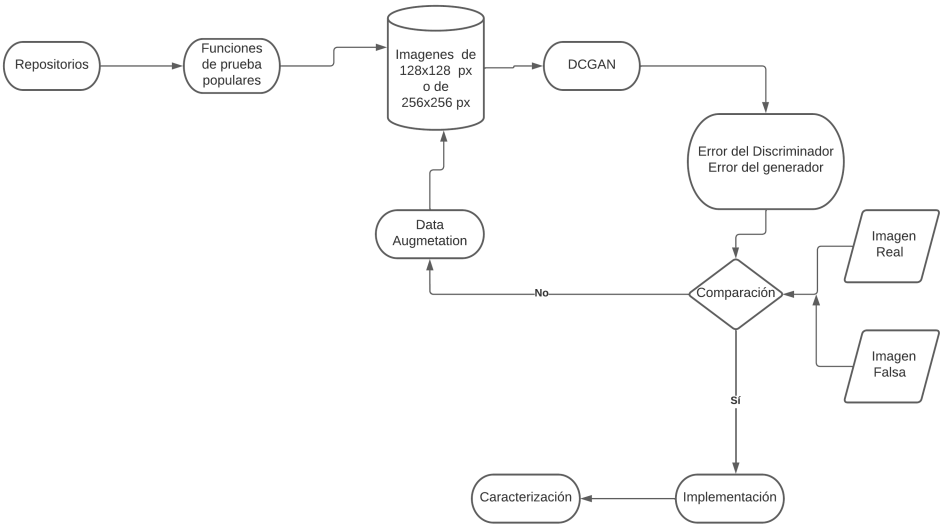


Fig. 1. Diagrama de bloques del proyecto

Como se observa en el diagrama de bloques, se desarrolló un método capaz de generar paisajes bidimensionales de optimización. Este método está compuesto por una base de datos que cuenta con imágenes de 128x128 píxeles que representan los paisajes de optimización. Dichos paisajes serán de funciones conocidas en la mayoría de los casos, y no se tendrán en cuenta las funciones que tengan un parámetro aleatorio en su expresión matemática..

El algoritmo que se desarrolló es una DCGAN. El generador cuenta con 6 capas convolucionales. Las 5 primeras tienen como función de activación LeakyReLU, y la última capa tiene como función de activación una tangente hiperbólica (tanh). El discriminador cuenta con 6 capas convolucionales.

En el capítulo III se detalla la conformación de la base de datos, las características que presentan los paisajes de optimización, cómo se pueden dividir en otras bases de datos con patrones similares y cómo al aplicar técnicas del aprendizaje automático se puede aumentar la información.

En el capítulo IV se encuentra toda la información de manera clara y precisa sobre la arquitectura y los parámetros de la red. Además, se incluye el procesamiento de la información para que pueda ser utilizada por la red. Luego de esto, se encuentra el capítulo V, donde se presentan los resultados. Se muestran diversos y nuevos paisajes bidimensionales de optimización, se detalla la calidad de los resultados y el aprendizaje de la red. En este punto también se realiza un análisis de las posibilidades, complicaciones y limitaciones que presenta la utilización de este método.

II. MARCO TEÓRICO

2.1. *Teoría de la optimización*

La teoría de la optimización se dedica al estudio de los puntos óptimos de una función y las técnicas para determinarlos[11], puede considerarse como la búsqueda de la mejor solución de un problema[12]. También se puede definir como una rama de las matemáticas aplicadas y el análisis numérico que tiene como objetivo encontrar los parámetros óptimos posibles en un espacio de soluciones dependientes de un conjunto de criterios que se expresan como funciones matemáticas[13].

2.1.1) Definiciones:

- Problemas: El objetivo de la optimización es obtener los mejores elementos **x*** a partir de un conjunto *X* según una función en especifico *f(x)* y esta función se define a través de una expresión matemática conocida como función objetivo.
  Si se habla matemáticamente, se trata de obtener los puntos extremos de una función. Un ejemplo de un problema de minimización se establece en la ecuación 1 [14], donde se desea encontrar un vector de x que minimice la función.

$$min_{x \in X} f(\vec{x}) \tag{1}$$

  Donde *X* el dominio de la función y *f(x)* es la función objetivo[14].

- Función Objetivo: Es la función sujeta a optimizar, dicha función que será minimizada o maximizada. También se puede decir que define el problema a optimizar, ya que representa el comportamiento del problema o en algunos casos como se desea que sea el comportamiento de este, dependiendo de sus parámetros la función puede estar sujeta a maximizar o minimizar[14].

  En las ecuaciones 2 y 3 se definen los dos tipos de problemas a los que se enfrenta la teoría de la optimización, el primero es un problema de maximización y el segundo un problema de minimización[15].

$$Max. f(\vec{x}) \qquad s.a \quad g(\vec{x}) \leq b \tag{2}$$

$$Min. f(\vec{x}) \qquad s.a \quad g(\vec{x}) \leq b \tag{3}$$

  Las funciones pueden caracterizarse como continuas, discontinuas, lineales, no lineales, convexas, no convexas, unimodales, multimodales, separables y no separables [16].

- Función Convexa :
  Definición 1: Una función es convexa si para todo punto se puede trazar una recta tangente y la gráfica de la función siempre queda por encima de la recta tangente[17].

  Definición 2: Sea $I \subseteq R$ un intervalo. Se dice que una función $f : I \to R$ es convexa en I si para cualquier t que satisface $0 \leq t \leq 1$ y para dos puntos cualesquiera x_1, x_2 de I se tiene[17]

$$f((1-t)x_1 + tx_2) \leq (1-t)f(x_1) + tf(x_2)$$

- Función multimodal:
  Definición 1: Una función continua $f : I = [a, b] \to I$, es $m - modal$ si existe una sucesión de puntos

$$a = c_0 < c_1 < ... < c_m < c_{m+1} = b,$$

  tal que f es estrictamente monótona en cada intervalo $[c_i, c_{i+1}]$ y alternadamente creciente, decreciente en esos intervalos[18].

  Definición 2: Se dice que una función $m - modal$ $f : I \to I$ es de frontera anclada, si $f(a, b) \subseteq a, b$[18].

- Espacio del problema: Es el conjunto compuesto por todos los elementos posibles del problema de optimización, aunque el espacio puede ser acotado por restricciones[14].

- Restricciones: También se les conoce como limitaciones, se encargan de reducir la cantidad de alternativas posibles y de esta manera definiendo un espacio acotado de soluciones factibles[19].

- Gradiente: El gradiente de un campo escalar $f : R^n$, R es un campo vectorial que indica en cada punto del campo escalar la dirección de máximo incremento del mismo[14].

  Otra definición del gradiente es una función f y es la colección de todas las derivadas parciales en forma de vector, como se muestra en la ecuación 4.

$$\nabla f(x, y, z \dots) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \\ \vdots \end{bmatrix} \tag{4}$$

- Óptimo Local: Es un punto en el espacio de búsqueda que se corresponde con una solución óptima para un vecindario del mismo, tienen como característica ser puntos de atracción para los algoritmos

de búsqueda, pero un buen algoritmo no debería quedarse atascado en un óptimo local[14].

Considerando que D es la intersección entre el dominio de la función f y el conjunto de soluciones factibles.

$$\exists \delta > 0 \quad tq \quad f(\vec{x}) \leq f(\vec{x_0}) \forall \vec{x} \in D \cap B(\vec{x_0}, \delta) \tag{5}$$

$$\exists \delta > 0 \quad tq \quad f(\vec{x}) \geq f(\vec{x_0}) \forall \vec{x} \in D \cap B(\vec{x_0}, \delta) \tag{6}$$

En la ecuación 5 se define que $\vec{x_0}$ es un máximo local de f en D si existe una bola centrada en $\vec{x_0}$, un entorno de $\vec{x_0}$, donde la imagen de $\vec{x_0}$ sea mayor o igual que la de los otros puntos. En la ecuación 6 se define $\vec{x_0}$ como mínimo local.

- Óptimo Global: Es el conjunto de puntos del espacio de búsqueda que corresponden con los valores máximos o mínimos (maximización o minimización) de la función objetivo, siendo la mejor solución que se puede encontrar para el problema que esta sujeto a optimización[14].

$$f(\vec{x}) \leq f(\vec{x_0}) \forall \vec{x} \in D \tag{7}$$

$$f(\vec{x}) \geq f(\vec{x_0}) \forall \vec{x} \in D \tag{8}$$

En la ecuación 7 se define que $\vec{x_0}$ es un máximo global de f en D, si su imagen es mayor o igual que la de cualquier otro punto del dominio. En la ecuación 8 se define $\vec{x_0}$ como mínimo global.

2.1.2) Métodos de optimización: Los métodos de optimización se definen como aquellas técnicas que permiten seleccionar la mejor solución ante un conjunto de alternativas. Los métodos para encontrar la solución óptima varían de acuerdo a la complejidad del problema. Cuando se hablan de problemas triviales, son aquellos que el cerebro humano podría resolver directamente, pero a medida que la complejidad aumenta, es necesario la búsqueda de herramientas adicionales[14].

Las técnicas más básicas son aquellas que se implementan para dar solución a problemas de optimización no lineales de variables reales en un espacio continuo, son restricciones. Este caso es cuando se asume que la función es diferenciable[14].

En la Tabla I se observa una breve clasificación de los diferentes métodos utilizados en la teoría de la optimización.

TABLA. I. Métodos de Optimización

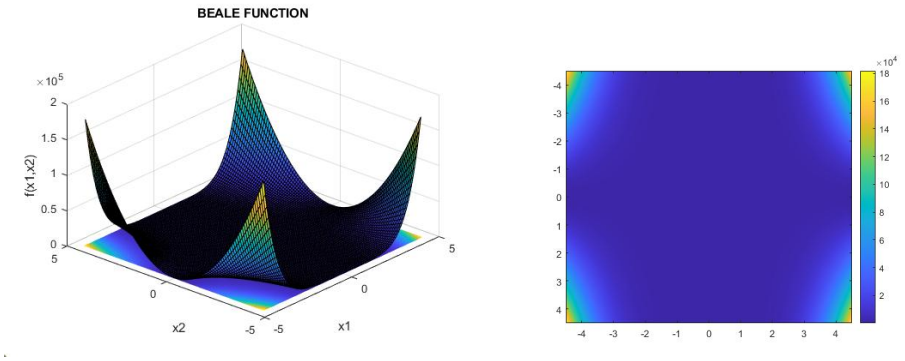
<table>
<tr><th colspan="4">Método</th><th>Ejemplos</th></tr>
<tr><td>Analíticos</td><td colspan="3">Exactos</td><td></td></tr>
<tr><td rowspan="5">Numéricos</td><td colspan="3" rowspan="2">Directos</td><td>Búsqueda Dorada</td></tr>
<tr><td>Búsqueda de Fibonacci</td></tr>
<tr><td colspan="3" rowspan="3">Indirectos</td><td>Descenso de gradiente</td></tr>
<tr><td>Newton</td></tr>
<tr><td>Quasi-Newton</td></tr>
<tr><td colspan="4" rowspan="3">Determinísticos</td><td>Branch and Bound</td></tr>
<tr><td>Divide and Conquer</td></tr>
<tr><td>Geometría Algebraica</td></tr>
<tr><td rowspan="10">Probabilísticos</td><td rowspan="10">Algoritmos Montecarlo</td><td colspan="3">Hill Climbing</td></tr>
<tr><td colspan="3">Random walk</td></tr>
<tr><td colspan="3">Recocido Simulado</td></tr>
<tr><td colspan="3">Búsqueda Tabú</td></tr>
<tr><td rowspan="6">Computación Evolutiva</td><td rowspan="5">Algoritmos Evolutivos</td><td>Algoritmos Genéticos</td></tr>
<tr><td>Programación Genética</td></tr>
<tr><td>Programación Evolutiva</td></tr>
<tr><td>Estrategias Evolutivas</td></tr>
<tr><td>Evolución Diferencial</td></tr>
<tr><td rowspan="2">Inteligencia de Enjambres</td><td>Colonia de Hormigas</td></tr>
<tr><td>Optimización por cúmulo de partículas</td></tr>
</table>

La palabra algoritmo esta definida como un conjunto ordenado de operaciones sistemáticas que permiten hacer un cálculo y así, hallar la solución a algún problema[20]. Ahora si se une esta definición con los métodos nombrados anteriormente tenemos los algoritmos de optimización.

2.2. Funciones de prueba

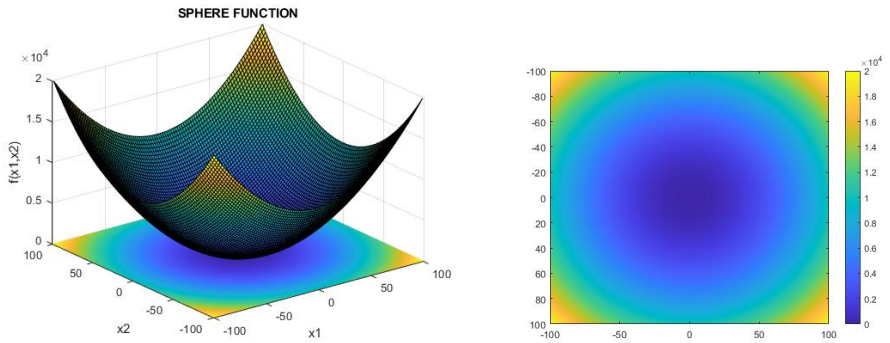
Las funciones de prueba son problemas que se encargan de los algoritmos de optimización en situaciones incomodas o difíciles, como ejemplos se tienen funciones con varios mínimos locales o funciones con un mínimo que es particularmente plano[21].

La creación o construcción de funciones de prueba para la evaluación de algoritmos de optimización es una tarea desafiante. Existen tres tipos principales de funciones de prueba en la literatura: funciones que modelan una variedad de aplicaciones del mundo real, instancias de problemas con ciertas características designas y funciones de prueba generadas aleatoriamente con soluciones conocidas[21].
Se entiende como paisaje bidimensional, la gráfica de la función de prueba donde se tiene en cuenta un ancho y largo, pero no una profundidad.



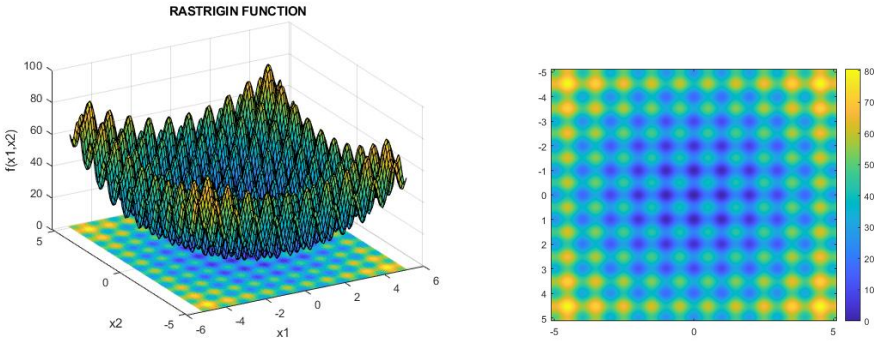
(a) Paisaje Tridimensional (b) Paisaje Bidimensional

Fig. 2. Funciones BEALE. Tomado de [22]



(a) Paisaje Tridimensional (b) Paisaje Bidimensional

Fig. 3. Funciones SPHERE. Tomado de [22]



(a) Paisaje Tridimensional (b) Paisaje Bidimensional

Fig. 4. Funciones RASTRIGIN. Tomado de [22]

En las Figuras 2,3 y 4 se observan tres funciones de prueba con características diferentes, cada una de ellas se presenta con su gráfica tridimensional y bidimensional. La primera es una función BEALE, la cual es una función multimodal, con picos pronunciados en las esquinas del domino de entrada, la segunda es una función SPHERE, es una función continua, convexa y unimodal, y la última es una función RASTRIGIN, la cual presenta varios mínimos locales, es altamente multimmodal con una ubicación de los mínimos

que se distribuyen regularmente[23].

Los problemas de referencia pueden clasificarse en función de características como la modalidad, las cuencas, los valles, la separabilidad y la dimensionalidad[16].

- Modalidad: Corresponde al número de picos ambiguos, los algoritmos de optimización tienden a tener problemas con ellos durante el proceso de búsqueda, al quedar atrapados en estos[16].

- Cuencas: Podría describirse como una meseta en un problema de maximización [24].

- Valles: Aparece cuando una zona angosta con poco cambio se encuentra rodeada por regiones con descensos pronunciados [24][16].

- Separabilidad: La función podrá ser separable o no separable dependiendo de que si sus variables muestran interrelación entre sí o si son independientes[16].

2.3. Antecedentes de funciones de prueba

2.3.1) A new class of test functions for global optimization: En este artículo, se propone una nueva clase de funciones de prueba para problemas de optimización, usando algunos parámetros con los cuales se puede controlar la dificultad de los problemas de prueba. Además, se incluyen algunos experimentos computacionales con dos algoritmos de optimización.
Para la construcción de las funciones de prueba el primer paso fue introducir algunos componentes unidimensionales básicos con múltiples minimizadores en el nivel 1, también se incluye un termino de *Oscilación* basado en la función coseno.

El siguiente paso consiste en la introducción de componentes n-dimensionales básicos con minimizadores locales de 2^m, la forma mas sencilla de obtener estas funciones es sumando componentes unidimensionales con uno o dos minimizadores locales en el nivel 2. Luego, se realiza una combinación lineal de dos componentes básicos. Por ultimo, se definen funciones con minimizadores locales L2 y L3, en el nivel 2 y 3 respectivamente.

En conclusión el trabajo brinda una clase de funciones de prueba para operaciones globales sin restricciones, todo se realiza a partir de unos parámetros en especifico(Osciladores, funciones unidimensionales y n-dimensionales, entre otros)[3].

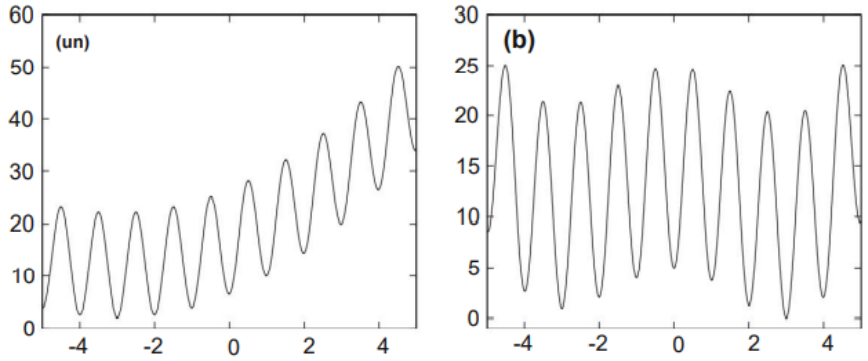


Fig. 5. Funciones de componentes unidimensionales. Tomado de[3]

2.3.2) Test problem generator for unconstrained global optimization: En este trabajo se desarrollo un generador de funciones de prueba de alto desempeño para generar problemas de prueba analíticos y altamente multimodales con el objetivo de comparar algoritmos de optimización sin restricciones. Mas específicamente se propone un procedimiento novedoso y eficiente computacionalmente para generar funciones no restringidas, no separables, no convexas, no lineales con 3 condiciones, la primera funciones de prueba analíticas, la segunda minimizadores locales y la tercera niveles de dificultad controlables.

El trabajo presentó un enfoque computacionalmente eficiente para generar una clase en especifico de problemas de prueba con el objetivo de comparar algoritmos de optimización global sin restricciones. Esta clase de problemas de prueba es analítica R^2 y tiene 2^n minimizadores locales conocidos a priori.

En la Figura 5 el $*$ es un minimizador global, $\times$ un minimizador no global, $\blacktriangle$ un maximizador local y $\bullet$ un punto de silla[21].

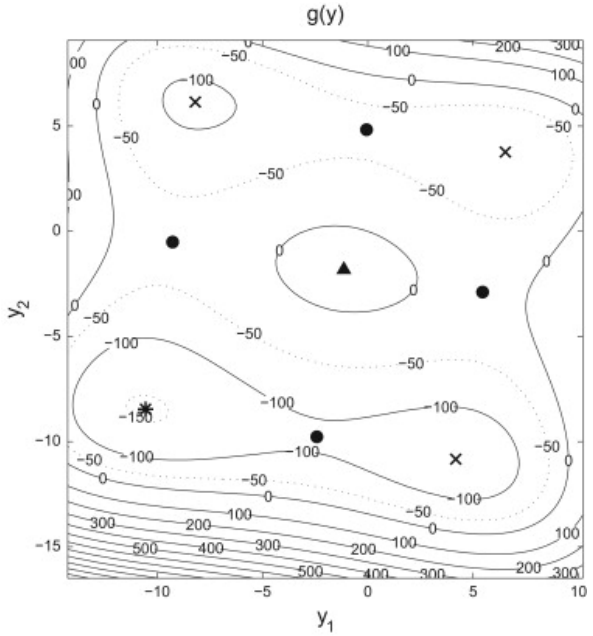


Fig. 6. Contornos del problema de nivel 2. Tomado de[21]

2.3.3) New hard benchmark functions for global optimization: En este articulo se presentan algunas funciones de prueba unimodales, multimodales y de ruido para evaluar el desempeño de los algoritmos de optimización global. Todas las funciones de prueba son funciones multidimensionales.

Para mostrar la dureza de estas funciones, hemos realizado un estudio experimental con algunos algoritmos potentes como los ganadores del concurso CEC, los algoritmos LSHADE, MadDe y LSHADE-SPACMA.

Las funciones de prueba propuestas también validan el teorema NFL que estipula que ningún algoritmo es eficiente para tratar todos los problemas de optimización[25].

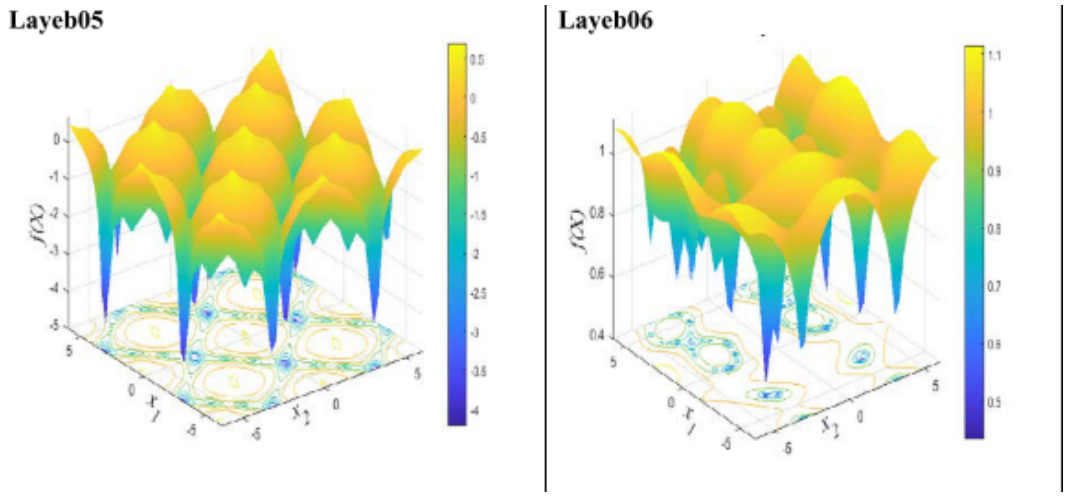


Fig. 7. Las superficies 3d de las funciones propuestas 3. Tomado de[25]

2.4. Redes Neuronales GAN

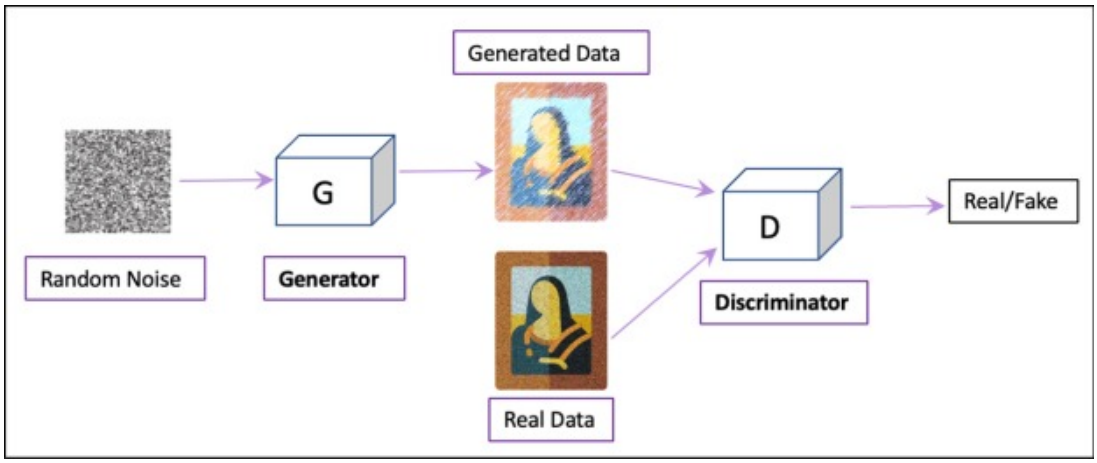


Fig. 8. Arquitectura GAN básica. Tomado de[8]

Las GANs utilizan arquitecturas de aprendizaje profundo. El objetivo de la red es generar nuevos datos o ejemplos de datos que los humanos no podrían distinguir entre los datos provenientes del conjunto real y las datos generados. Esto se logra con un entrenamiento exitoso en el que la red puede distinguir patrones en los datos y aprender las distribuciones de estos. Las GAN consisten principalmente en dos redes de aprendizaje profundo como se observa en la Figura 8, el primero un generador y el segundo un discriminador.El generador cuenta con la capacidad de generar o crear nuevos datos o conjuntos de datos a partir del ruido y el discriminador reconoce los puntos de datos como reales o falsos. Ambos modelos se entrenan juntos en un juego y por ello también se les conoce como red competitiva, donde el objetivo final es que el generador engañe al discriminador aproximadamente la mitad de las veces[8][26].

El generador realiza la apreciación de los datos y el discriminador verifica si son datos reales o no. Por lo tanto, las dos redes compiten entre sí y la pérdida del discriminador se debe minimizar durante el entrenamiento. Si el discriminador no cuenta con la capacidad de saber si los datos son reales o falsos, en función de un umbral predefinido, el entrenamiento se terminará.

Cuando el discriminador no esta en la capacidad de distinguir los datos falsos producidos por el generador de los datos reales, se debe mantener el entrenamiento. Por el contrario, si el discriminador domina, entonces las imágenes generadas se distinguen fácilmente de las imágenes reales, y la tasa de entrenamiento del discriminador se detiene y, en consecuencia, termina el entrenamiento del generador[27].

Anteriormente se hablo de la existencia de 2 redes, la primera, una red Generativa, la cual toma una entrada aleatoria z con densidad $P(z)$ y devuelve una salida $x(g)$ = $G(z)$ que debe seguir la función objetivo después del entrenamiento. La segunda, una red Discriminativa que toma una entrada x, la cual puede ser una entrada verdadera ($x(t)$ cuya densidad se denota $P(t)$) o una entrada creada ($x(g)$ cuya densidad se denota $P(g)$, donde está es la densidad inducida por $P(z)$ a través del generador) y devuelve una probabilidad $D(x)$ de que x sean datos verdaderos.

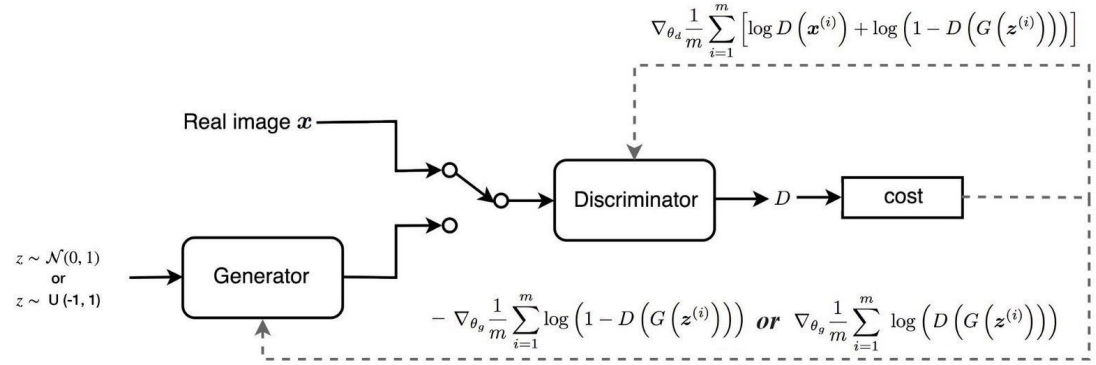


Fig. 9. Arquitectura GAN básica. Tomado de[8]

En la Figura 9 representa una arquitectura GAN básica que consta de un único generador y discriminador. El generador no tiene acceso al conjunto de datos de la base de datos real. Minimiza su error al recibir retroalimentación del discriminador. Mientras tanto, el discriminador tiene acceso a datos reales y a los datos falsos, es decir, si los datos provienen del generador o del conjunto de datos real, y puede usar estos para minimizar su error[8]. A Continuación se definen unas arquitecturas GAN más sofisticadas que pueden utilizar etiquetas para generar puntos de datos en una categoría específica.

- **Conditional Adversarial Network(CGAN):**Es una extensión de la GAN típica, es una arquitectura condicional, donde el generador y el discriminador están sujetos a información auxiliar como etiquetas de clase. La información auxiliar se combina con el ruido de la GAN básica, con esto se desea que el generador controle la producción de los datos según la condición puesta por la información auxiliar. Este tipo de arquitectura es adecuada para la generación multimodal de puntos de datos[28][8]. En la Figura 10 se observa la arquitectura de una GAN condicional, donde la entrada al generador no es solamente el ruido aleatorio z, si no que se alimenta de una segunda entrada y con la información auxiliar[29].

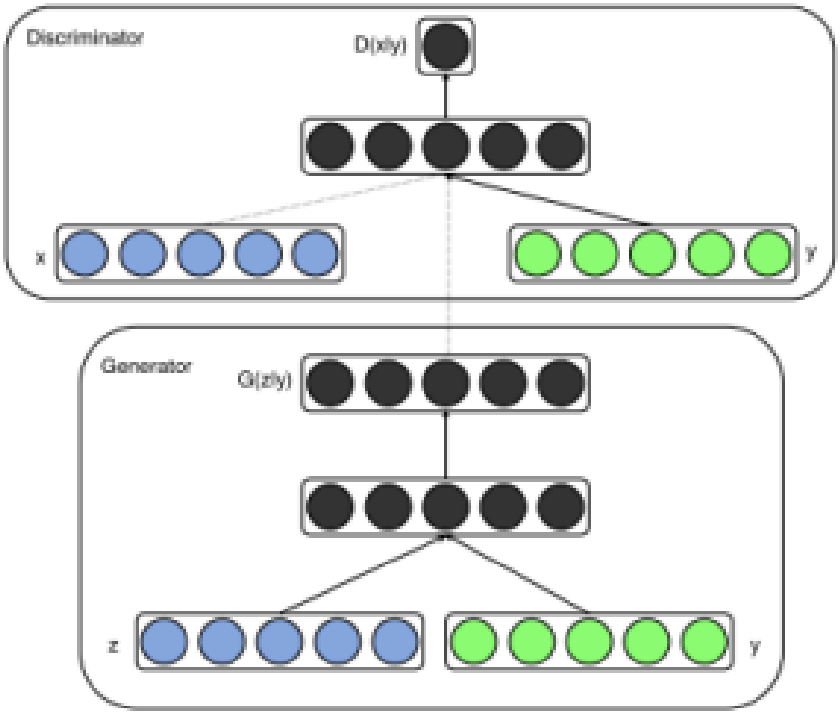


Fig. 10. Arquitectura de una CGAN. Tomado de[29]

- **Deep Convolutional Adversarial Networks(DCGAN):** comúnmente conocida como DCGAN[30], en este tipo de red GAN, el generador y discriminador tienen como arquitectura redes CNNs (Convolutional Neural Networks). Esto facilita el mapeo del espacio de la imagen al espacio latente de menor dimensión, y del espacio de la imagen a un discriminador[31]. Dado el éxito de las CNNs en la clasificación de imágenes y vídeos en los últimos años, DCGAN sigue siendo una arquitectura adecuada para aplicaciones con enfoques en la generación de imágenes[8]. En la Figura 11 se muestran las partes de una red CCN con algunos cambios en especifico de la arquitectura típica de esta. Se remplazan los *max-pooling* por convoluciones por *stride*, se utiliza un *batch normalization*, se eliminan las capas totalmente conectadas y utilizar la función de activación *ReLU* excepto en la salida[29].

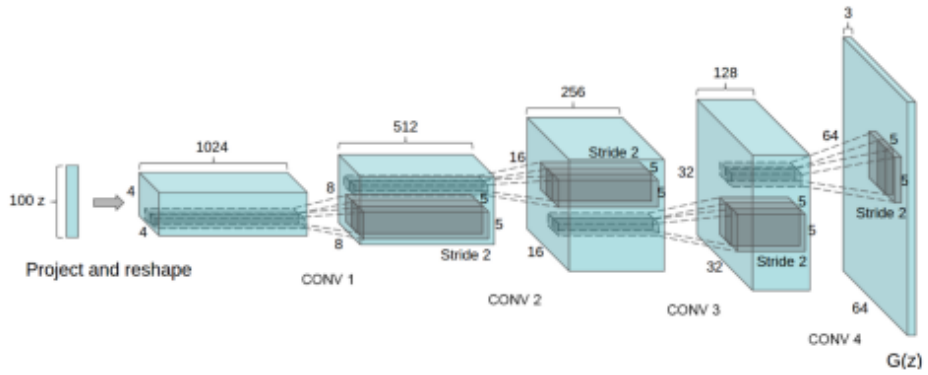


Fig. 11. Arquitectura del generador de una DCGAN. Tomado de[29]

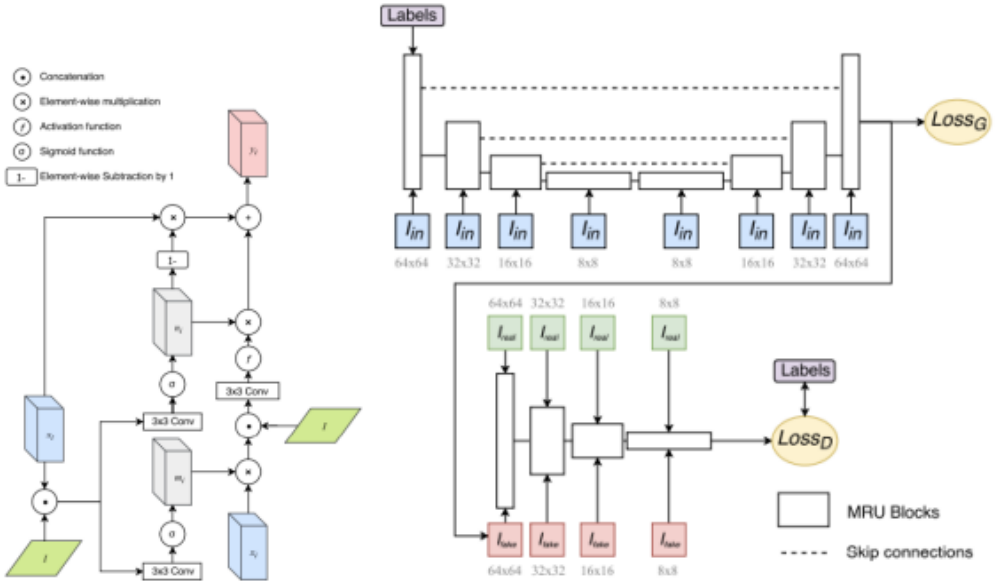


Fig. 12. A la izquierda, un bloque MRU utilizado por la SketchyGAN. A la derecha, arquitectura de la SketchyGAN. Tomado de[29]

- **SketchyGAN:** La idea principal por la que se creo este tipo de arquitectura fue el de sintetizar imágenes arbitrarias a partir de bocetos. La red presenta una arquitectura que fue específicamente diseñada para que la salida dependa fuertemente de la influencia de la entrada y es por ello que se introducen los bloques MRU (Masked Residual Unit) como se observa en la Figura 12. Estos bloques utilizan una mascara interna que es capaz de aprender a extraer características de manera selectiva durante el entrenamiento[29].

2.4.1) Función de perdidas: Para el entrenamiento de la red se necesita tener definida la función de perdidas o función objetivo a maximizar o minimizar, cumpliendo con el juego existente entre el discriminador y generador. El generador trata de ajustar sus parámetros de peso para cada época que finalmente lo llevaría a producir puntos de datos realistas. Mientras tanto, el discriminador intenta ajustar sus parámetros de peso de manera que sea capaz de distinguir entre las muestras reales y las falsas, y es aquí donde aparece la función de perdidas, la cual determina qué tan bien los modelos pueden acercarse al conjunto óptimo de parámetros.

$$E(G,D) = \frac{1}{2}E_{z\sim P_z}[1 - D(G(z))] + \frac{1}{2}E_{x\sim P_D}[D(x)] \tag{9}$$

En la ecuación 9 se expresa el error absoluto esperado por el discriminador. Donde $E_{z\sim P_z}$ hace referencia al error esperado del discriminador con el lote real y $E_{x\sim P_D}$ al error esperado del discriminador con el lote verdadero. La idea al entrenar la red es maximizar el error para el generador mientras se trata de minimizar el del discriminador. Lo cual da

$$f_{minimax} = max_G(min_D E(G,D)) \tag{10}$$

Una de las funciones mas implementadas se conoce como perdida por entropía cruzada y se expresa matemáticamente de la siguiente manera:

$$H(p,q) = -\sum_{\forall x} p(x)\log q(x) \tag{11}$$

Donde x es la entrada del discriminador, $p(x)$ es la distribución verdadera y $q(x)$ es la distribución estimada[8].

La entropía cruzada toma muestras de una zona con mayor probabilidad, lo que genera una nueva distribución de densidad de probabilidad. Como la ocurrencia de la solución óptima de la función objetivo puede verse como un evento de pequeña probabilidad, el concepto de entropía cruzada puede aplicarse al problema de optimización[32][33].

También se define entropía cruzada como una tecnica adaptativa que se basa en la minimización de la distancia *Kullback-Leibler*(KL), que es la distancia medida entre dos *pmfs*(Probability Mass Functions) a y b denotadas por $\mathfrak{D}(a, b)$, como se observa en la ecuación 12. Donde X es el espacio muestral donde están definidas las distribuciones de probabilidad[34].

$$\mathfrak{D}(a, b) = \sum_{X \in x} a(X) \ln \frac{a(X)}{b(X)} \tag{12}$$

En la entropía cruzada se genera un conjunto aleatorio de valores para los parámetros a estimar, y luego se actualiza para generar mejores valores en la siguiente iteración[34].

2.5. Parámetros iniciales para una DCGAN

Existen algunos parámetros que son definidos al comienzo de los algoritmos para el funcionamiento de la red, dichos parámetros varían dependiendo de la base de datos, la red que se usa o el objetivo que se desea. Es por ello que se definen antes de la arquitectura de la red y del pre-procesamiento de la bases de datos.

- **Tamaño del lote:** Es el número de ejemplos en un lote para trabajar antes de actualizar los parámetros del modelo interno. Las estrategias populares para el tamaño del lote están a continuación:

  - Descenso de gradiente estocástico, en el que el tamaño del lote es 1.
  - Lote completo, en donde el tamaño del lote es el número de ejemplos en todo el conjunto de entrenamiento o en toda la base de datos.
  - Mini lote, donde el tamaño del lote toma valores inferiores al total de lo ejemplos, suele estar entre 10 y 1000, suele ser la estrategia más eficiente[35].

- **Epochs:** Se define como el número de veces que el algoritmo de aprendizaje funcionará en todo el conjunto de datos de entrenamiento. En otras palabras una Epoch se refiere a un ciclo a través del conjunto de datos de entrenamiento completo[35].

- **Iteración:** El tamaño del lote determina cuantos ejemplos procesa el modelo en una sola iteración. Ejemplo, si el tamaño del lote es 16 el modelo procesa 16 ejemplos antes de actualizar sus parámetros internos. Una epoch equivale a una cantidad de iteraciones como se observa en la ecuación 13

$$1 \quad epoch = \frac{N}{Tamaño \quad del \quad lote)} \quad Iteraciones \tag{13}$$

  Donde N es el número de ejemplos en la base de datos [35].

- **Tasa de aprendizaje:** Corresponde a un número de punto flotante que le dice al algoritmo de descenso de gradiente con que fuerza debe ajustar los pesos y sesgos en cada iteración[35].

- **Optimizador Adam:** *Adam* o *Estimación Adaptativa del Momento*

$$m_t = \beta_t m_{t-1} + (1 - \beta_1) g_t \tag{14}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{15}$$

Donde t representa la iteración t-ésima y g_t es el gradiente calculado. m_t y v_t son medidas móviles del gradiente y la hessiana, donde β_t es quien regula la "velocidad"del gradiente anterior y del gradiente actual. Adam intenta calcular las tasas de aprendizaje adaptativo para cada parámetro. Esto es muy útil en estructuras de red complejas porque diferentes partes de la red tienen diferente sensibilidad al ajuste de los pesos. Una parte muy sensible generalmente requiere una tasa de aprendizaje más pequeña [35].

2.6. Equilibrio de Nash

El equilibrio de Nash es el concepto de solución mas importante para juegos en forma estratégica. Para poder definir el equilibro de Nash, se debe definir primero el significado de *"Mejor respuesta"*.

Definición (*Mejor respuesta*): *Sea $G = (N, A, u)$ un juego en forma estratégica. Se dice que una estrategia $a_i* \in A_i$ es una mejor respuesta a un vector de estrategia a_{-i} de los otros jugadores si*

$$u_i(a_i*, a_{-i}) \geq u_i(a_i, a_{-i}), \qquad \forall a_i \in A_i$$

Al conjunto de mejores respuestas para el jugador i para cada $a_{-i} \in A_i$ se denota como $MR_i(a_{-i}) := a_i* : u_i(a_i*, a_{-i}) \geq u_i(a_i, a_{-i}) \forall a_i \in A_i$.

Donde $a_{-i} = (a_1, a_2, ..., a_{i-1}, a_{i+1}, ..., a_n)$ el perfil estratégico a sin la estrategia del agente i. Si todos los agentes que no son i (i) se comprometen a jugar a_{-i}, el jugador i que quiere maximizar su utilidad se enfrentará al problema de determinar cuál es su mejor respuesta.

Definición (*Equilibrio de Nash*): *Un perfil estratégico $a* = (a_1*, ..., a_n*)$ es un equilibrio de Nash si:*

$$u_i(a_i*, a_{-i}*) \geq u_i(a_i.a_{-i}*), \qquad \forall a_i \in A_i \quad y \quad \forall i \in N$$

En otras palabras, un equilibrio de Nash en un juego de forma estratégico es un perfil de estrategias en

el cual ningún agente sale ganando si se desvía unilateralmente y cambia de estrategia[36].

2.7. Antecedentes de GANs como generadoras de imágenes

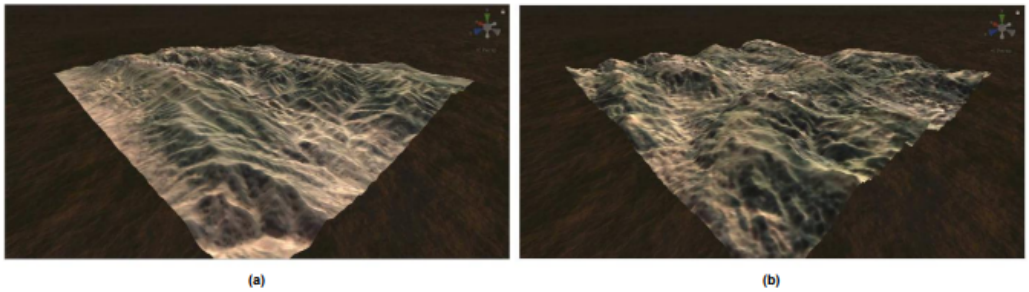


Fig. 13. a)Región de entrenamiento b) Región generada. Tomado de[37]

2.7.1) Realistic and Textured Terrain Generation using GANs: Este trabajo tiene como objetivo entrenar redes del tipo GAN, la cual pueda aprender y generar diversas regiones de terreno texturizado para uso directo dentro de un sistema de visión por computadora. Para lograr el objetivo se utiliza el siguiente método que consiste en entrenar como propuesta una SGAN(Semi-Supervised Generative Adversarial Network) con una imagen de entrada cuidadosamente preparada, se compone de una imagen texturizada que se superpone directamente en la parte superior de un mapa de altura que utiliza coordenadas de longitud y latitud.Además, se implementan otros tipos de redes GANs, como lo son DCGANs (Deep Convolutional Generative Adversarial NetworkS) y el método CNN.

Como conclusión general del trabajo, se tiene que los resultados muestran una exploración de un tipo más específico de red adversaria generativa para la generación de múltiples datos en una instancia. El método propuesto ofrece el uso de aprendizaje profundo generativo para automatizar el proceso de generación de entorno, creando los datos de altura y textura a partir de una entrada singular de la región deseada. En la Figura 13.a) se observa la imagen para el entrenamiento y en la Figura 13.b) se muestra la imagen que fue generada a partir de la GAN[37].

2.7.2) Fast Cosmic Web Simulations with Generative Adversarial Networks: La materia oscura del universo evoluciona a través de la gravedad para formar una compleja red de halos, filamentos, láminas y vacíos, que se conoce como la red cósmica. Los modelos computacionales de los procesos físicos subyacentes, como las simulaciones clásicas de N-cuerpos, son extremadamente de recursos, ya que rastrean la acción de la gravedad en un universo en expansión utilizando miles de millones de partículas como trazadores de la distribución de la materia cósmica.

Las GANs son una solución para el problema de recursos, se utilizan estas redes para aprender modelos que puedan generar de forma eficiente nuevas realizaciones físicamente realistas de la red cósmica. El

conjunto de entrenamiento es una pequeña muestra representativa de instantáneas de imágenes 2D de simulaciones de N-cuerpos de tamaño 500 y 100 Mpc (Megapársec). Demostrando que las muestras generadas por GAN son cualitativa y cuantitativamente muy similares a las originales. Ademas, las GANs requieren una fracción de segundo, en comparación con las técnicas tradicionales de N-cuerpos para cada muestra nueva generada.

En este trabajo se utilizó una versión ligeramente modificada de la arquitectura DCGAN típica, se utilizaron arquitecturas similares para el generador y el discriminador, que consisten en 5 capas convolucionales. El numero total de parámetros entrenables en ambas redes es de $3{,}2 \cdot 10^7$[38].

En la Figura 14 se presenta una muestra de las simulación para la técnica de N-cuerpos y la muestra generada por la GAN, para el tamaño de caja de 100 Mpc. Se puede observar la gran similitud en los resultados de los diferentes métodos para la generación de estas imágenes[38].

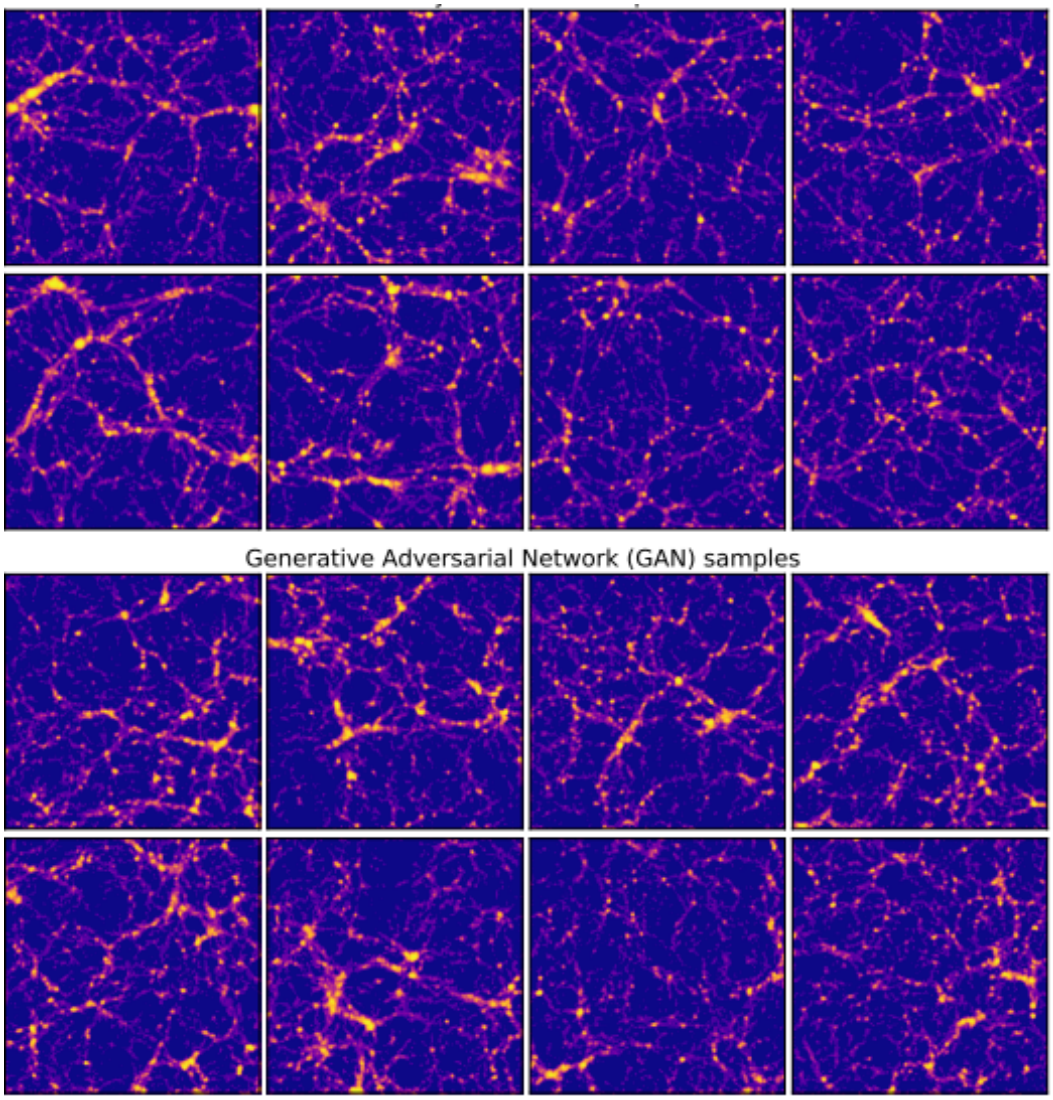


Fig. 14. Muestras de la simulación de cuerpo N y de GAN para el tamaño de caja de 100 Mpc. Tomado de[38]

2.8. *Data Augmentation*

Es una técnica que radica en aplicar distintos tipos de transformaciones a los datos originales con el objetivo de producir nuevos datos[39]. Existen transformaciones geométricas, las transformaciones del espacio de color, los filtros de núcleo, la mezcla de imágenes, el borrado aleatorio, el aumento del espacio de características, el entrenamiento adversario, el aumento basado en GAN, la transferencia de estilo neural y demás [40].

Data augmentation se puede realizar sobre $DataSpace$ o bien sobre $FeatureSpace$, el primero hace referencia al conjunto de valores de la muestra de los datos, mientras que el segundo se refiere a todo el espacio de los datos tanto los muéstrales como los que no. En la Figura 15 se observa un esquema del Data Augmentation [41].

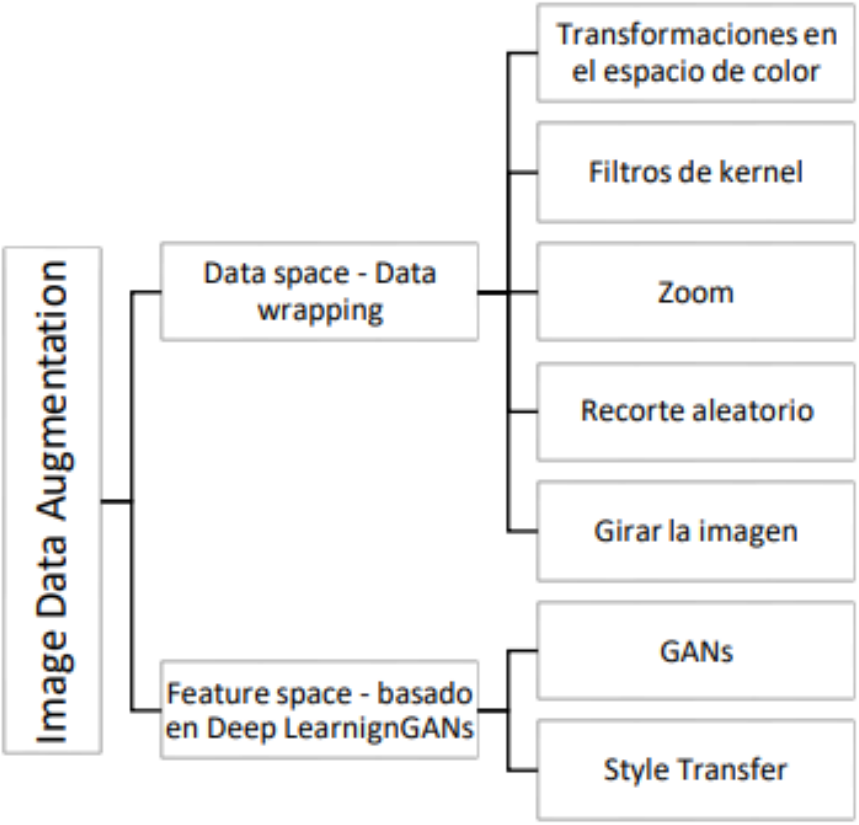


Fig. 15. Esquema Data Augmentation[41]

2.9. Pytorch y herramientas de desarrollo

PyTorch es una biblioteca de tensores optimizada para el aprendizaje profundo mediante GPU y CPU[42]. Es una librería de código abierto, esta librería cuenta con varios módulos interesantes para el desarrollo de redes neuronales. La diferencia con otras librerías es que calcula los gradientes de los pesos de las distintas capas a la vez aplica la fase *forward* a los datos de entrenamiento durante el aprendizaje de la red. Además, permite a crear módulos personalizados, lo cual permite al programador tener una mayor libertad a la hora de diseñar las redes neuronales y así, permitir la creación de estas de manera mas especificas para problemas en concreto[43]. Posee todas las funciones, clases y atributos necesarios para definir las redes neuronales, trabajar con los gradientes de las mismas, entrenarlas, validarlas, testearlas y actualizar sus pesos y sesgos[44].

PyTorch trabaja con tensores, que son matrices multidimensionales con un mismo tipo de datos, que recibe como argumentos una lista o una lista de listas [43].

2.9.1) Tensores: Son empleados para el almacenamiento y la operación de colecciones de números multidimensionales, rectangulares y homogéneos. A continuación se presentan lo módulos principales de esta biblioteca:

- **Autograd:** Proporciona clases y funciones que implementan diferenciación automática de funciones arbitrarias escalares[44].

- **Optim:** Implementa varios algoritmos de optimización utilizados en el aprendizaje de redes neuronales, como Adam[44].

- **nn:** Implementa todas las clases, funciones y atributos necesarios para desarrollar redes neuronales[44].

III. CREACIÓN DE LA BASE DE DATOS

Una parte fundamental del proyecto es la búsqueda de funciones de prueba para problemas de optimización y, a partir de ellas, la producción de los paisajes bidimensionales. En este capítulo se describe la conformación de la base de datos, comenzando por la búsqueda de las funciones de prueba e implementación en Matlab, la conformación de la base de datos principal y la generación de tres bases de datos adicionales. Al final, se realiza un Data Augmentation para conformar otras cuatro bases de datos. Además, se incluyen los criterios para la selección de los paisajes en las diferentes bases de datos

3.1. Búsqueda e implementación de los paisajes de optimización

Al revisar la literatura, se descubre que en [5] no solo se habla de las 25 funciones de prueba iniciales, sino que también se mencionan dos citas [16], [22] con una cantidad considerable de funciones de prueba. En la referencia [16], se especifican las características de las funciones, sus ecuaciones y límites, mientras que en [22] se encuentra la implementación en Matlab(R) de una gran variedad de funciones.

A partir de la implementación especificada anteriormente, se generan los paisajes en 2 y 3 dimensiones. Los paisajes en 3 dimensiones se grafican con la intención de poder caracterizarlos y dividirlos. Por otro lado, los paisajes en 2 dimensiones son los que conforman la base de datos, y muestran imágenes en escala RGB.

En total, se obtienen 161 paisajes que provienen de funciones unimodales y multimodales, separables y no separables, continuas y discontinuas, diferenciables y no diferenciables, escalables y no escalables. La definición de estas categorías se encuentra en [16] y está especificada en el marco teórico.

3.2. Conformación de las bases de datos

Se cuenta con una base de datos como origen o cimiento para las demás bases de datos. Esta base de datos es la Base 1, que cuenta con los 161 paisajes de optimización mencionados en la sección anterior. A continuación, se especifican las características de cada una de las bases de datos.

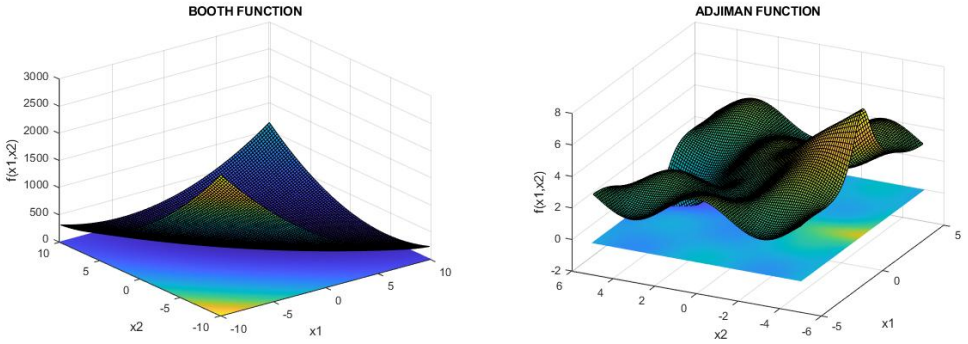
TABLA. II. Características de las bases de datos.

	Con ruido	Con un minimo	Con varios minimos	con Data Augmentation	Total de Paisajes
Base 1	x	x	x		161
Base 2		x	x		124
Base 3		x			89
Base 4			x		71
Base 5	x	x	x	x	644
Base 6		x	x	x	496
Base 7		x		x	356
Base 8			x	x	284

En la Tabla II se especifica que característica visual tiene cada base de datos[2], cuantos paisajes contienen en total y si se le aplica o no el Data Augmentation. A continuación se detallan las 4 primeras bases de datos y luego se precisa la aplicación del Data Augmentation para las otras 4.

3.2.1) Base 1: Como se mencionó anteriormente, esta base de datos cuenta con la mayor diversidad, ya que incluye todos los paisajes seleccionados de la literatura. Se presentan 5 figuras donde se observan las características mencionadas anteriormente y que están especificadas en la sección 3.2. Por cada figura, se presenta una función que incluye esa característica y otra que no la contiene. De esta forma, se ofrece de manera gráfica lo que se describe en palabras en el marco teórico sobre estas características, deseando generar una percepción visual de lo que implica que la función posea o no la característica.

En la Figura 16, se presentan dos funciones de prueba para diferenciar la característica de modalidad. La Figura 16 a) representa una función unimodal, mientras que la Figura 16 b) representa una función multimodal.

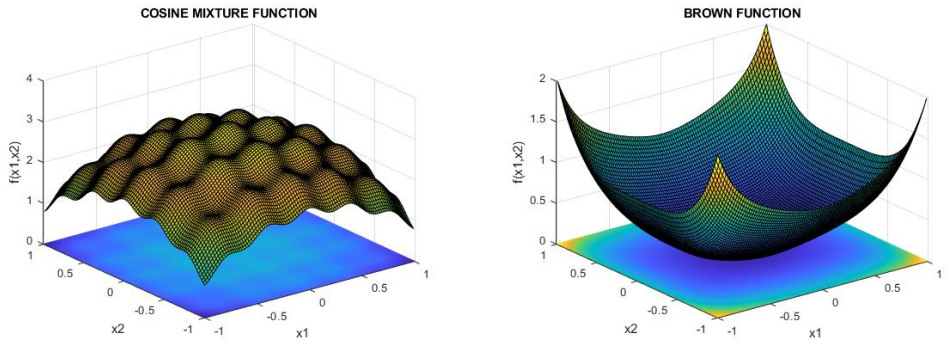


(a) Paisaje Unimodal (b) Paisaje Multimodal

Fig. 16. Característica Modalidad. Tomado de [22]

[2]Cuando se habla de la característica Con ruido hace referencia a los paisajes que visualmente tienen picos repentinos y seguidos. Como se observa en la Figura 21

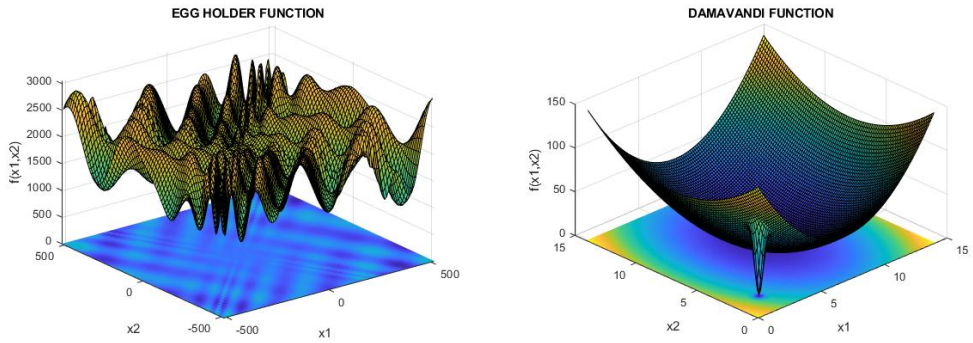
En la Figura 17, se presentan dos funciones de prueba para la característica de separabilidad, que habla de la independencia e interrelación de sus variables. La Figura 17 a) representa una función separable, mientras que la Figura 17 b) representa una función no separable.



(a) Paisaje Separable (b) Paisaje No-Separable

Fig. 17. Característica Separabilidad. Tomado de [22]

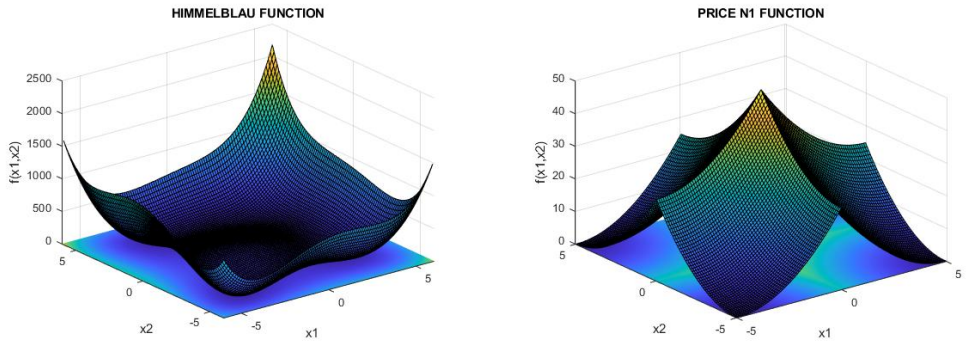
En la Figura 18, se presentan dos funciones de prueba para diferenciar la característica de escalabilidad. La Figura 18 a) representa una función escalable, mientras que la Figura 18 b) representa una función no escalable.



(a) Paisaje Escalable (b) Paisaje No-Escalable

Fig. 18. Característica Escalabilidad. Tomado de [22]

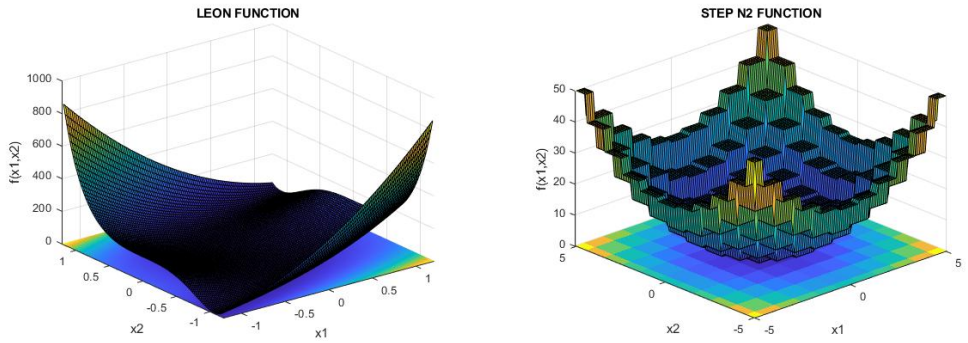
En la Figura 19, se presentan dos funciones de prueba para observar las diferencias existentes entre una función diferenciable y una no diferenciable. La Figura 19 a) representa una función diferenciable, mientras que la Figura 19 b) representa una función no diferenciable.



(a) Paisaje Diferenciable (b) Paisaje No-Diferenciable

Fig. 19. Característica Escalabilidad. Tomado de [22]

La última figura corresponde a la característica de continuidad. En la Figura 20, se presentan dos funciones de prueba. La Figura 20 a) representa una función continua, mientras que la Figura 20 b) representa una función discontinua.

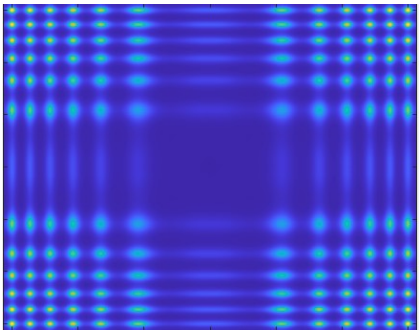


(a) Paisaje Continuo (b) Paisaje Discontinuo

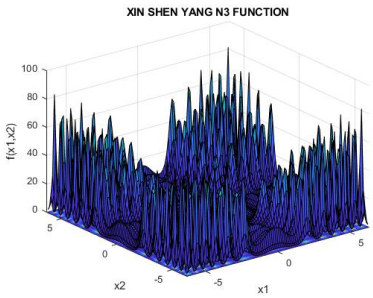
Fig. 20. Característica Continuidad. Tomado de [22]

Las funciones presentadas pueden contener varias características a la vez. Un ejemplo es la Figura 19 a), que muestra la función Himmelblau, la cual es continua, diferenciable, no separable, no escalable y multimodal.

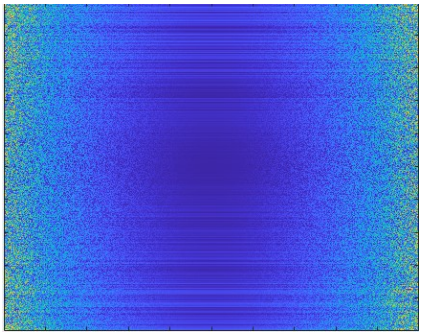
3.2.2) Base 2: En la Base 2 se eliminan todos los paisajes que presentan visualmente ruido en sus gráficas tridimensionales. En la Figura 21, se presentan 4 paisajes en 2 y 3 dimensiones donde se observa la presencia de ruido, y por tanto, este tipo de paisajes son excluidos de la Base 2.



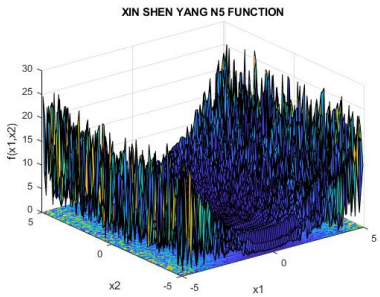
(a) Xin Shen Yan N3 Bidimensional



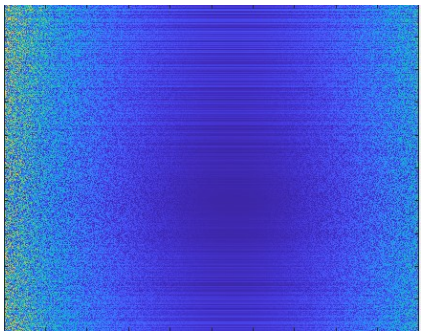
(b) Xin Shen Yan N3 Tridimensional



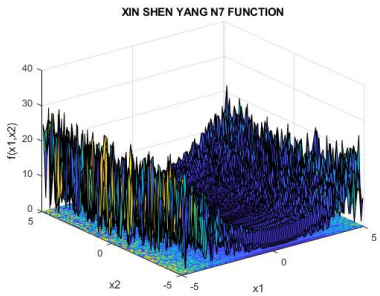
(c) Xin Shen Yan N5 Bidimensional



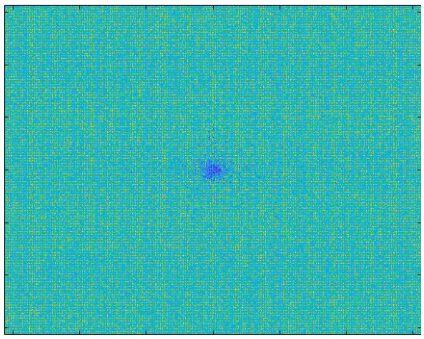
(d) Xin Shen Yan N5 Tridimensional



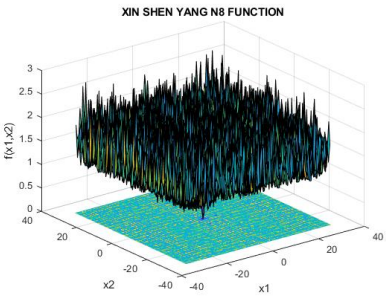
(e) Xin Shen Yan N7 Bidimensional



(f) Xin Shen Yan N7 Tridimensional



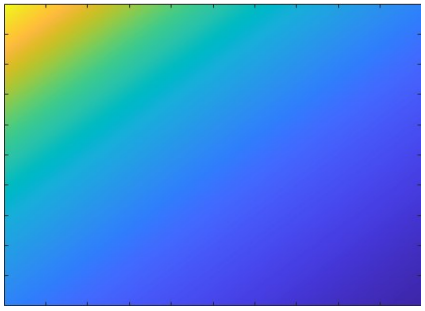
(g) Xin Shen Yan N8 Bidimensional



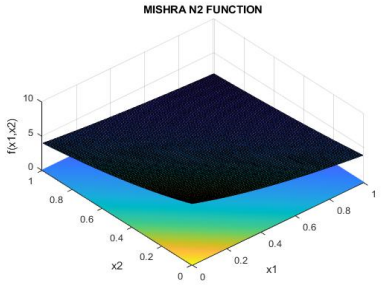
(h) Xin Shen Yan N8 Tridimensional

Fig. 21. Ejemplos de paisajes ruidosos

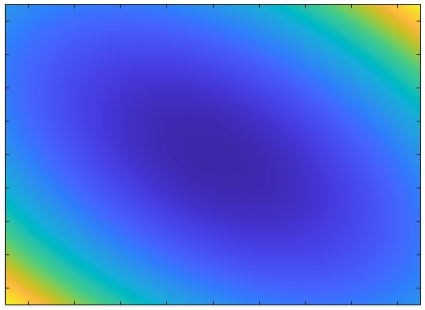
3.2.3) Base 3: La Base 3 parte de la Base 2 y no cuenta con los paisajes ruidosos. Se seleccionan los paisajes con presencia de un solo mínimo o funciones con valles extensos, donde no hay presencia de cambios repentinos e inesperados. En la Figura 22, se exhiben algunos de los paisajes correspondientes a la Base 3.



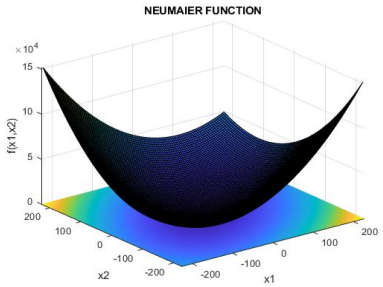
(a) Mishra N2 Bidimensional



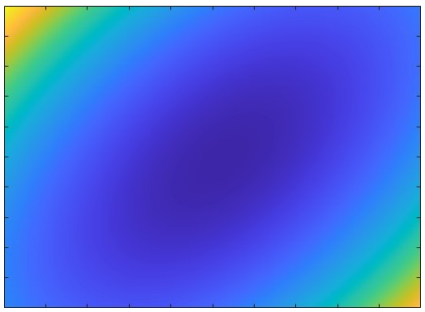
(b) Mishra N2 Tridimensional



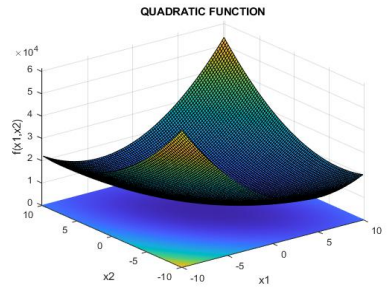
(c) Neumair Bidimensional



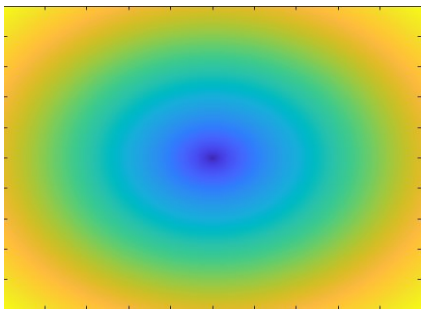
(d) Neumair Tridimensional



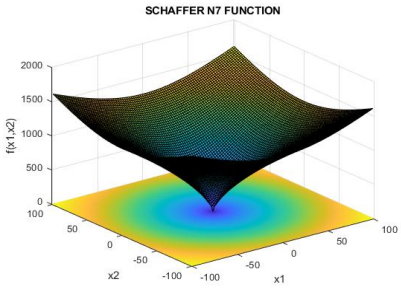
(e) Quadratic Bidimensional



(f) Quadratic Tridimensional



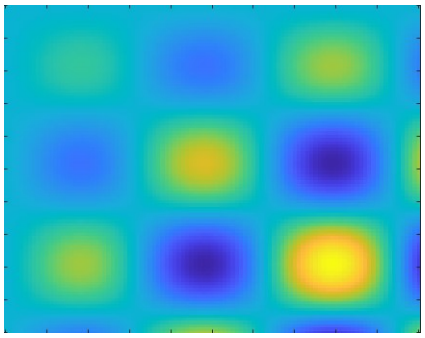
(g) Schaffer N7 Bidimensional



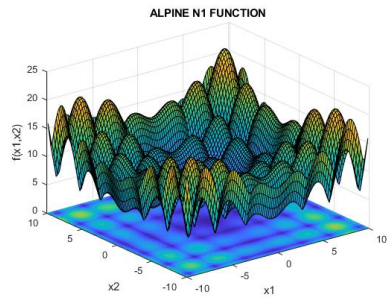
(h) Schaffer N7 Tridimensional

Fig. 22. Ejemplos de paisajes para la Base 3

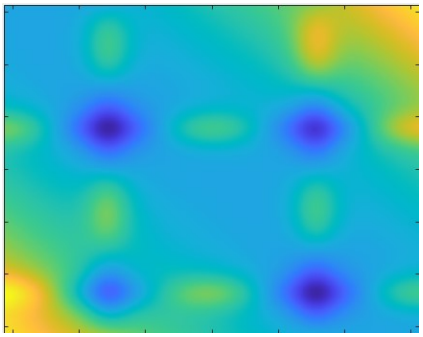
3.2.4) Base 4: La Base 4 parte de la Base 2 y, a diferencia de la Base 3, esta base selecciona los paisajes con cambios repentinos y presencia de varios mínimos, como se observa en la Figura 23.



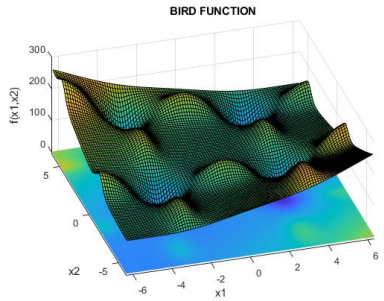
(a) Alpine N1 Bidimensional



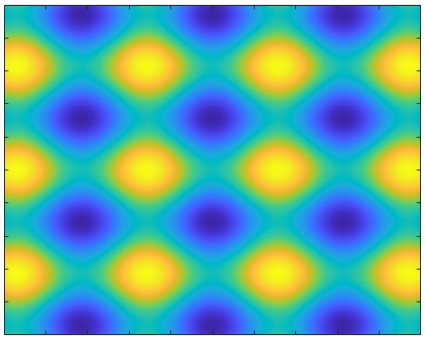
(b) Alpine N1 Tridimensional



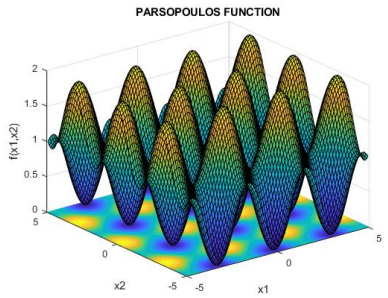
(c) Bird Bidimensional



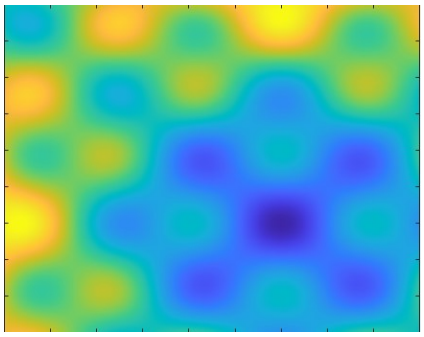
(d) Bird Tridimensional



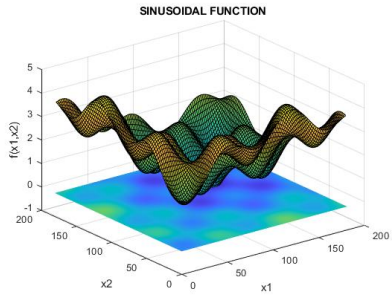
(e) Parsopoulo Bidimensional



(f) Parsopoulo Tridimensional



(g) Sinusoidal Bidimensional

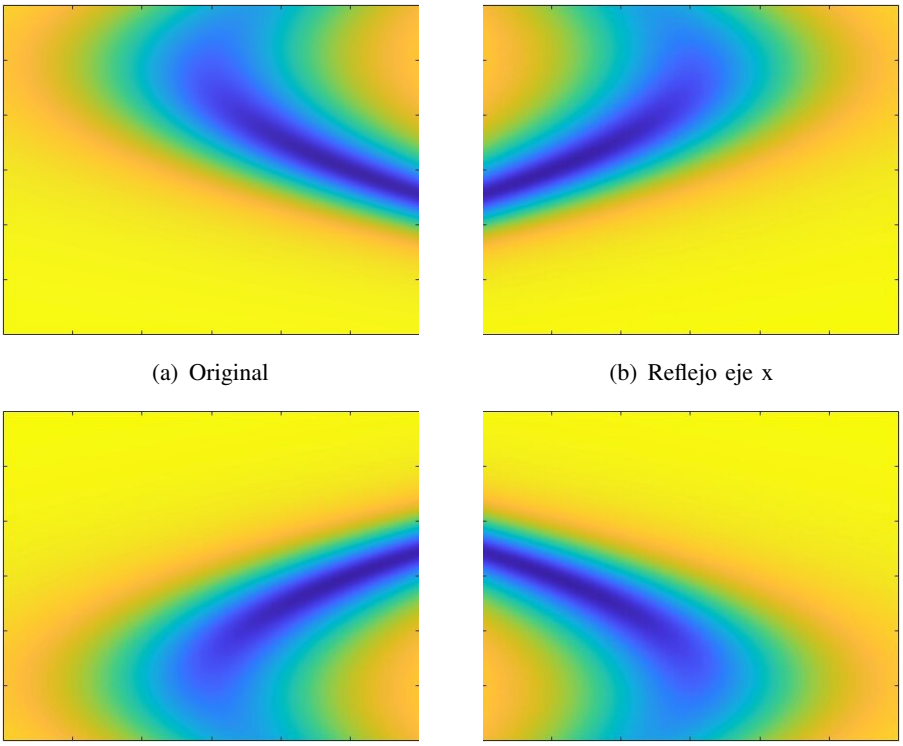


(h) Sinusoidal Tridimensional

Fig. 23. Ejemplos de paisajes para la Base 4

3.2.5) Aplicación Data Augmentation: Las Bases de la 5 a 8 representan las Bases de la 1 a 4, respectivamente, con la diferencia de que se aplica un Data Augmentation. En la Figura 24 a) se muestra el paisaje original; en la Figura 24 b) se presenta el paisaje reflejado en el eje x; en la Figura 24 c) se observa el paisaje reflejado en el eje y; y, por último, en la Figura 24 d) se tiene un paisaje reflejado en ambos ejes.

Se consideró que reflejar los paisajes no conllevaría a una pérdida de información y no cambiaría los límites de las funciones, como podría suceder al utilizar otras técnicas de Data Augmentation. En cambio, sí aumentaría la diversidad en la base de datos.



(a) Original

(b) Reflejo eje x

(c) Reflejo eje y

(d) Reflejo en ambos ejes

Fig. 24. Ejemplo aplicando Data Augmentation

IV. RED ADVERSARIA GENERATIVA (DCGAN)

El tipo de red GAN elegida es una DCGAN (Deep Convolutional Adversarial Networks) [30]. Se seleccionó esta red por contar con redes convolucionales para el generador y el discriminador. El método para la síntesis de paisajes de optimización bidimensional se realiza en 8 partes, como se muestra y especifica a continuación.

4.1. Parámetros de entrada:

Los parámetros que se especifican al comienzo del algoritmo se precisan en la Tabla III. Algunos de estos parámetros son requeridos para cualquier tipo de red neuronal artificial (tamaño del lote, número de epochs, etc.), mientras que otros son específicos para redes convolucionales y redes GAN (mapas de características, vectores, entre otros). El valor mínimo y máximo para el tamaño del lote implementado se encuentra en la tabla, así como el número máximo de epochs y, por último, la tasa de aprendizaje del valor más pequeño al más grande.

Se aclara que el tamaño del lote, el número de epochs y la tasa de aprendizaje se van cambiando dependiendo de la base de datos con la que se esté trabajando. Cada uno de los resultados y dichos parámetros se especifican en el capítulo de resultados.

En la Tabla III, Beta corresponde al hiperparámetro del optimizador Adam y el vector de entrada del generador corresponde a ruido aleatorio. Además, cabe resaltar que la cantidad de píxeles para el tamaño de la imagen se ve limitado a ese valor máximo. La razón de dicho valor es la memoria disponible de los servidores de Google, específicamente en la herramienta de Google Colab.

Solo se utiliza un canal ya que se trabaja en escala de grises, donde cero es negro y uno es blanco, para reducir el tiempo de cómputo y la carga computacional ejercida por el procesamiento de la información en el entrenamiento de red. Se decidió trabajar en escala de grises ya que este cambio no afecta la información original y, por tanto, no afecta el aprendizaje de la red para el cumplimiento del objetivo.

4.2. Procesamiento de la base de datos:

Para el comienzo del procesamiento de la base de datos, se utiliza un cargador de datos genérico *"dset.ImageFolder(root, transform)"*, donde los paisajes se organizan de cierta manera por defecto. Se especifican dos parámetros: **root**, que corresponde a la ruta del directorio raíz, y **transform**, que corresponde a la realización de varias transformaciones en conjunto o especificadas juntas, esto con la ayuda

TABLA. III. Parámetros de entrada.

Parámetro		
Tamaño del lote	16 Paisajes	256 Paisajes
Tamaño de la imagen	128x128 píxeles	
Número de canales	1	
Vector Entrada del Generador	100	
Mapa de características del Generador	64	
Mapa de características del Discriminador	64	
Número de Epochs	5000	
Tasa de aprendizaje	0.0001	0.005
Beta	0.5	

de la clase *"transforms.Compose"*. Dichas transformaciones son necesarias para que al cambiar la base de datos no se vea afectada ni la estructura de la red ni sus parámetros internos. Por ejemplo, no hay necesidad de que los paisajes en la base de datos tengan el mismo tamaño en píxeles, ya que las transformaciones ajustan toda la información a un mismo tamaño de imagen. Las transformaciones que se llevan a cabo son las siguientes:

- *transforms.Resize (tamaño de la imagen):* Se encarga de cambiar el tamaño de las imágenes de la base de datos, por el tamaño especificado en los parámetros iniciales.

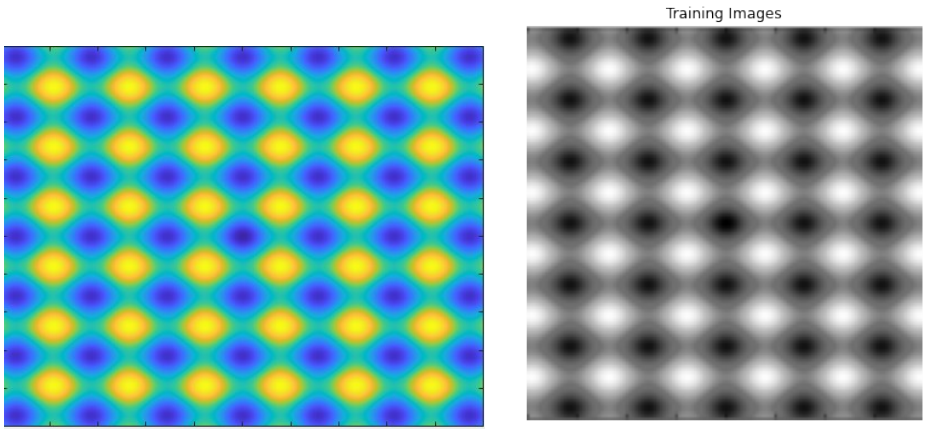
- *transforms.CenterCrop (tamaño de la imagen):* Realiza el cambio de tamaño priorizando el centro.

- *transforms.ToTensor():* Convierte la imagen a un torch.FloatTensor.

- *transforms.Normalize ((media=0.5),(desviaciones estándar=0.5)):* Normaliza una imagen con media y desviación estándar, los dos parámetros son una secuencia para cada canal, como se trabaja en escala de grises especifica solo un valor para la desviación y uno para la media.

- *transforms.Grayscale():* Transforma la imagen de RGB a escala de grises, cambiando de tres canales a uno.

A manera de ejemplo, en la Figura 25, se presenta la función de referencia Price N2. En 25(a) se muestra la función original y en 25(b) la función luego de realizar todo el procesamiento.



(a) Paisaje original (b) Paisaje procesado

Fig. 25. Ejemplo transformación de la base de datos

4.3. Inicialización de los pesos:

TORCH.NN.INIT es un módulo de PyTorch que contiene funciones diseñadas para inicializar los parámetros de la red neuronal. Se utilizan dos funciones de este módulo:
- Torch.nn.init.normal(m.weight.data, mean=0.0, std=0.02): esta función rellena el tensor de entrada con valores extraídos de la distribución normal [42]. Sus parámetros son los pesos, la media y la desviación estándar.
- Torch.nn.init.constant(m.bias.data, 0): se encarga de dar un valor constante a los bias de la red [42].

4.4. Generador:

En la Figura 26 se muestra de manera visual la arquitectura del generador, donde este toma como entrada un vector de ruido z, cuya longitud es definida al comienzo del algoritmo. Luego de una serie de capas, se genera una imagen con dimensiones de 128 x 128 x 1. En cada capa del generador se realizan tres funciones:

- Primero, ConvTranspose2d(), encargada de realizar los filtros y las convoluciones.
- Segundo, BatchNorm2d(), encargada de la normalización por lotes.
- Tercero, la función de activación ReLU.

Solo en la última capa no se hace la normalización y se cambia la función de activación por una tangente hiperbólica, como se observa en la Tabla IV.

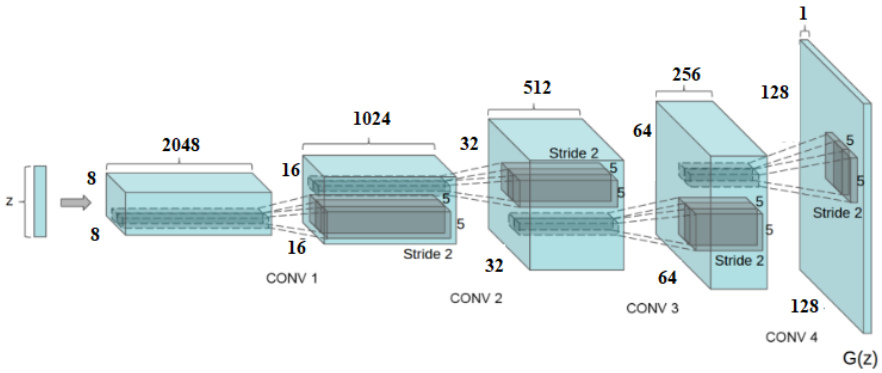


Fig. 26. Arquitectura del Generador. Tomado [42]

TABLA. IV. Arquitectura Generador

Sequential()							
0	**ConvTranspose2d**	in channels = 100	out channels = 2048	kernel = (4,4)	stride = (1,1)	bias = False	
1	**BatchNorm2d**	C = 2048	eps = 1e-05	momentum =0.1	affine = True	track running stats = True	
2	**ReLU**	inplace = True					
3	**ConvTranspose2d**	in channels = 2048	out channels = 1024	kernel = (4,4)	stride = (2,2)	Padding = (1,1)	bias = False
4	**BatchNorm2d**	C = 1024	eps = 1e-05	momentum =0.1	affine = True	track running stats = True	
5	**ReLU**	inplace = True					
6	**ConvTranspose2d**	in channels = 1024	out channels = 512	kernel = (4,4)	stride = (2,2)	Padding = (1,1)	bias = False
7	**BatchNorm2d**	C = 512	eps = 1e-05	momentum =0.1	affine = True	track running stats = True	
8	**ReLU**	inplace = True					
9	**ConvTranspose2d**	in channels = 512	out channels = 256	kernel = (4,4)	stride = (2,2)	Padding = (1,1)	bias = False
10	**BatchNorm2d**	C = 256	eps = 1e-05	momentum =0.1	affine = True	track running stats = True	
11	**ReLU**	inplace = True					
12	**ConvTranspose2d**	in channels = 256	out channels = 128	kernel = (4,4)	stride = (2,2)	Padding = (1,1)	bias = False
13	**BatchNorm2d**	C = 128	eps = 1e-05	momentum =0.1	affine = True	track running stats = True	
14	**ReLU**	inplace = True					
15	**ConvTranspose2d**	in channels = 128	out channels = 1	kernel = (4,4)	stride = (2,2)	Padding = (1,1)	bias = False
16	**Tanh**						

En la Tabla IV se observa que en la primera convolución se tiene una entrada de 100, correspondiente al vector z definido en los parámetros iniciales. Este valor va cambiando sucesivamente en las diferentes convoluciones hasta llegar a la capa de salida con un valor de 128. Así es como se genera una imagen de 128 x 128 x 1 a partir de un vector con longitud igual a 100.

Antes de especificar cada función implementada, es importante definir brevemente las capas utilizadas en las redes convolucionales y tener claridad acerca de por qué se utilizan esas funciones en la estructura de la red.

- Capa convolucional: Es la capa principal de las redes convolucionales. Los parámetros en este tipo de capa hacen referencia a un conjunto de filtros entrenables. Cada filtro ocupa un espacio pequeño a lo largo de las dimensiones de anchura y altura, pero se extiende por toda la profundidad del volumen de entrada al que se aplica[45].
- Capa de agrupación (Pooling layer): Se encarga de agrupar los mapas de características producidos en la capa convolucional en una imagen[45].

- Capa de unidades lineales rectificadas ReLu (Rectified Linear Units Layer): Es una capa muy comúnmente utilizada en redes convolucionales y se utiliza múltiples veces en una misma red, frecuentemente después de cada convolución[45].

Funciones:

- nn.ConvTranspose2d(): Aplica un operador de convolución transpuesta 2D sobre una imagen de entrada. También se puede ver como el gradiente de Conv2d con respecto a su entrada. Cuenta con los siguientes parámetros[42]:

  - ***in channels:*** Número de canales en la imagen de entrada.
  - ***out channels:*** Número de canales producidos por la convolución.
  - ***kernel size:*** Tamaño del núcleo convolutivo.
  - ***Stride:*** Controla el paso para la correlación cruzada.
  - ***Padding:*** Controla la cantidad de relleno cero implícito.

- nn.BatchNorm2d(): Aplica la normalización por lotes sobre una entrada 4D. La media y la desviación estándar se calculan por dimensiones sobre los minilotes, γ y β son vectores de parámetros aprendibles de tamaño c(donde c es el tamaño de la entrada).
- nn.ReLU(): Es la abreviatura para Rectified Linear Unit, aplica la función de unidad rectificada elemento a elemento, como se observa en la Figura 27. La función de esta capa es la de aumentar la propiedad de no linealidad de la función de activación, sin que se modifiquen los cuerpos receptivos de la capa convolucional[45].

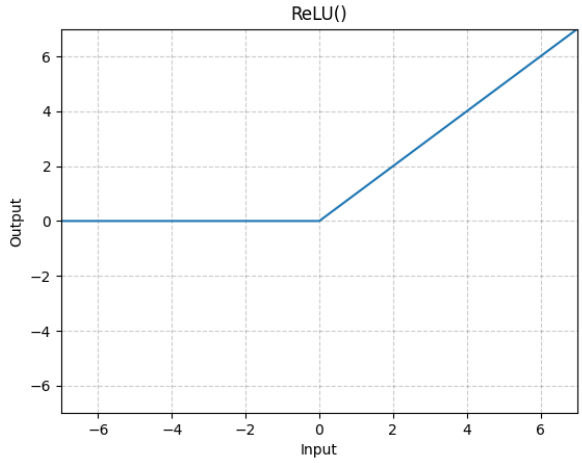


Fig. 27. Función ReLU. Tomado de [42]

- nn.Tanh(): Aplica la función Tangente Hiperbólica (Tanh) por elementos, la cual se observa en la Figura 28.

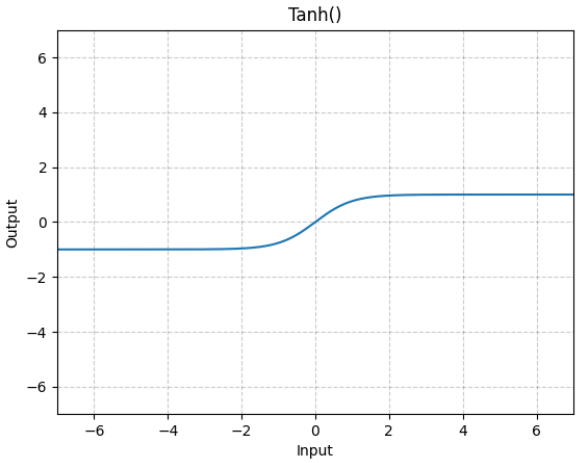


Fig. 28. Función Tanh. Tomado de [42]

Lo importante de la Tabla IV es cómo las entradas y salidas de cada función de convolución van cambiando la cantidad de píxeles de la red. También es importante observar la cantidad de convoluciones, ya que la cantidad de píxeles dependerá de la cantidad de capas de convolución. Por ejemplo, si se trabaja con 64 píxeles, se tendrán 4 capas de convolución.

4.5. Discriminador:

El discriminador es el encargado de diferenciar entre las imágenes falsas producidas por el generador y las imágenes originales. Por tanto, el discriminador tiene como entrada las imágenes de la base de datos procesadas y las imágenes producidas por el generador, las cuales son de tamaño 128 x 128 x 1. Además, se intercambian de manera aleatoria a la hora de pasar por el discriminador. Es decir, se pasa un lote de imágenes falsas, que corresponden a las imágenes producidas por el generador, como entrada del discriminador, y luego se pasa un lote de imágenes reales, que corresponden a la base de datos.

En la Figura 29 se muestra un esquema de la arquitectura del discriminador, donde se pueden ver las dos fuentes de entrada y su salida, la cual es falsa si la imagen es producida por el generador o verdadera si la imagen pertenece a la base de datos. A medida que avanza el entrenamiento, el discriminador comenzará a confundir estas etiquetas. Es decir, la imagen producida por el generador no siempre será clasificada como falsa, sino que comenzará a ser clasificada como verdadera al transcurso de las epochs. Entonces, idealmente se esperaría que el discriminador clasifique todas las imágenes como verdaderas, dando así un error de clasificación del $50\,\%$.

Al igual que en el generador, se tiene una figura de las capas y entradas del discriminador y una tabla que especifica las funciones de la arquitectura de la red. El discriminador comienza con una capa de convolución que tiene como entrada imágenes de 128x128x1. Luego se realizan todas las siguientes capas

de convolución hasta llegar a una salida de etiquetas de verdadero y falso. Se utilizan otras funciones de activación con respecto al generador, dichas funciones han sido probadas en muchos experimentos anteriormente y se establecen como la arquitectura base de una red DCGAN.

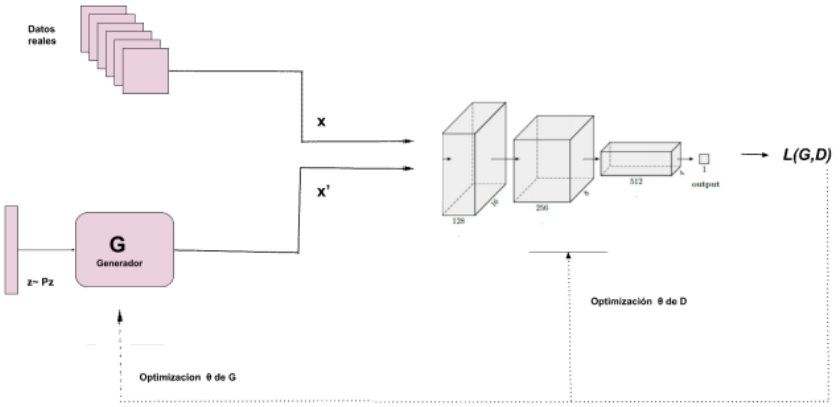


Fig. 29. Arquitectura del Discriminador. Tomado de [46][39]

En la Tabla V se tienen todas las funciones necesarias para la construcción del discriminador, también se especifican todos los parámetros para cada función. El discriminador cuenta con 4 funciones por capa, exceptuando la primera capa, donde solo son 2 funciones. Cuenta con una función más por capa que el Generador debido a que utiliza la funciónn $dropout()$ para evitar el sobreajuste. A continuación se describe el propósito de cada una de las funciones.

Funciones:

- nn.Conv2d(): Aplica una convolución 2D sobre una señal de entrada compuesta por varios planos de entrada.

  - ***in channels:*** Número de canales en la imagen de entrada.
  - ***out channels:*** Número de canales producidos por la convolución.
  - ***kernel size:*** Tamaño del núcleo convolutivo.
  - ***stride:*** Controla el paso para la correlación cruzada.
  - ***padding:*** Controla la cantidad de relleno aplicado a la entrada.

- nn.LeakyReLU(): Aplica la función element-wise. Esta definida por la ecuación 13 y su gráfica se observa en la Figura 30

$$LeakyReLU(x) = max(0, x) + negativeslopemin(0, x) \tag{16}$$

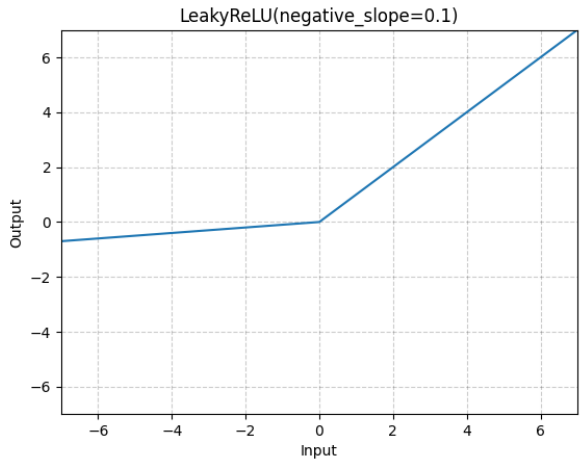


Fig. 30. Función LEAKY RELU. Tomado de [42]

- nn.Dropout(): Durante el entrenamiento, se ponen a cero aleatoriamente algunos de los elementos de la entrada con probabilidad p, utilizando muestras de una distribución de Bernoulli. En otras palabras, es una técnica en la que se ignoran durante el entrenamiento una serie de neuronas seleccionadas al azar. El efecto es que la red se vuelve menos sensible a los pesos específicos de las neuronas, lo que resulta en una red que es capaz de generalizar mejor y que tenga un menor sobreajuste a los datos de entrenamiento[45]. Se ha demostrado ser una técnica eficaz para la regularización y la prevención de la coadaptación de las neuronas.

- nn.Sigmoid(): Aplica la función sigmoid, la cual se observa en la Figura 31.

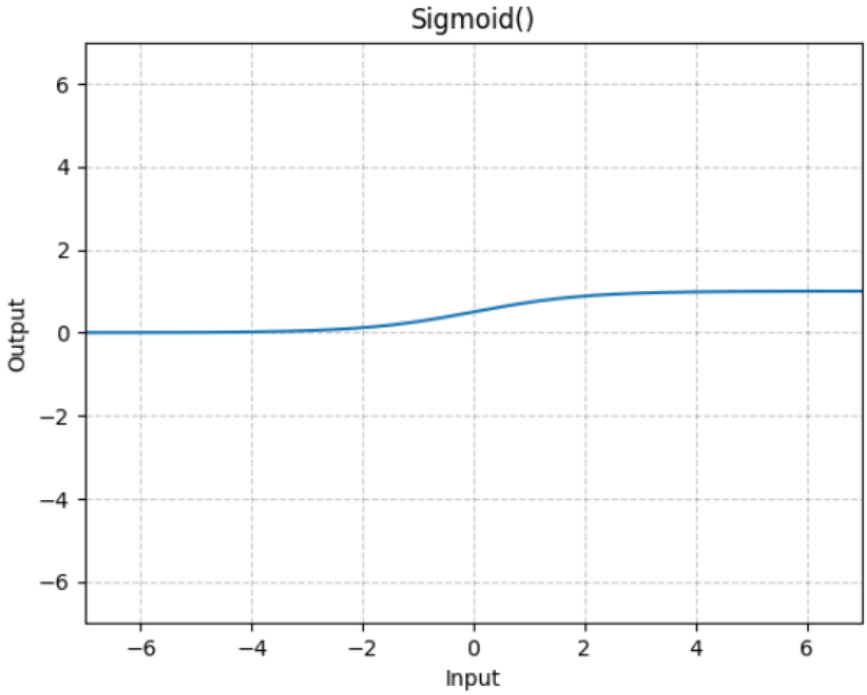


Fig. 31. Función Sigmoid. Tomado de [42]

TABLA. V. Arquitectura Discriminador

Sequential()							
0	**Conv2d**	in channels = 1	out channels = 128	kernel = (4,4)	stride = (2,2)	Padding = (1,1)	bias = False
1	**LeakyReLU**	negative_slope = 0,2			inplace = True		
2	**Conv2d**	in channels = 128	out channels = 256	kernel = (4,4)	stride = (2,2)	Padding = (1,1)	bias = False
3	**Dropout**	p = 0,8			inplace = False		
4	**BatchNorm2d**	C = 256	eps = 1e-05	momentum =0,1	affine = True	track running stats = True	
5	**LeakyReLU**	negative_slope = 0,2			inplace = True		
6	**Conv2d**	in channels = 256	out channels = 512	kernel = (4,4)	stride = (2,2)	Padding = (1,1)	bias = False
7	**Dropout**	p = 0,8			inplace = False		
8	**BatchNorm2d**	C = 512	eps = 1e-05	momentum =0,1	affine = True	track running stats = True	
9	**LeakyReLU**	negative_slope = 0,2			inplace = True		
10	**Conv2d**	in channels = 512	out channels = 1024	kernel = (4,4)	stride = (2,2)	Padding = (1,1)	bias = False
11	**Dropout**	p = 0,8			inplace = False		
12	**BatchNorm2d**	C = 1024	eps = 1e-05	momentum =0,1	affine = True	track running stats = True	
13	**LeakyReLU**	negative_slope = 0,2			inplace = True		
14	**Conv2d**	in channels = 1024	out channels = 2048	kernel = (4,4)	stride = (2,2)	Padding = (1,1)	bias = False
15	**Dropout**	p = 0,8			inplace = False		
16	**BatchNorm2d**	C = 2048	eps = 1e-05	momentum =0,1	affine = True	track running stats = True	
17	**LeakyReLU**	p = 0,8			inplace = False		
18	**Conv2d**	in channels = 2048	out channels = 1	kernel = (4,4)	stride = (2,2)	bias = False	
19	**Sigmoid**						

4.6. Realimentación de la red:

Esta sección está dividida en 4 partes, cada una de las partes es necesaria para lograr la retroalimentación de la red y de esta manera llevar a cabo un entrenamiento adecuado.

Inicialización de la función BCELoss:

Por sus siglas en inglés (Implementing Binary Cross Entropy loss), BCELoss es la función de pérdida que se implementa para la retroalimentación de la red. Crea un criterio que mide la entropía cruzada binaria entre el objetivo y las probabilidades de entrada, se utiliza para medir el error de una reconstrucción. En el marco teórico se especifica la función de pérdida y en uno de sus apartados, la entropía cruzada, y es de lo que se encarga esta función.

Lote de vectores:

La función *"torch.randn(1, nz, 1, 1, device=device)"* devuelve un tensor lleno de números aleatorios de una distribución normal con media y varianza. Donde z es el vector definido anteriormente y aquí corresponde al vector de salida.

Etiquetas:

Es necesario establecer una convención para las etiquetas reales y falsas durante el entrenamiento. Como se dijo en la sección del discriminador, este debe establecer si la imagen corresponde a una falsa (Generador) o una verdadera (Base de datos). Se definen dos variables de la siguiente manera: imágenes falsas (Salida

del Generador) $= 0$ e imágenes reales(Información de la base de datos) $= 1$.

Establecer optimizadores Adam:

Se utiliza el optimizador Adam como algoritmo de optimización para el entrenamiento de la red. Es necesario un optimizador para el generador y otro para el discriminador. Se hace uso de la función *optim.Adam()* y sus parámetros se describen a continuación:

- Generador: optim.Adam(netG.parameters(), lr=lr, betas=(beta1,0.999))

- Discrimnador: optim.Adam(netD.parameters(), lr=lr, betas=(beta1, 0.999))

El primer término de la función corresponde a los tensores de información de cada una de las redes. Luego se tiene la tasa de aprendizaje (lr) y el valor Beta del optimizador.

4.7. Entrenamiento de la red:

El entrenamiento de la red es un proceso repetitivo. Este proceso finaliza cuando se completa el número de epochs seleccionado en los parámetros iniciales. En el Algoritmo 1 se presenta en pseudocódigo el entrenamiento de la red y posteriormente se describe línea por línea.

Algorithm 1 : Entrenamiento de la red

for *Número de Epochs* **do**
  for *k lotes* **do**
    - Entrenar con todos los lotes reales (*Pdata*):
    - Entrenar con los lotes falsos (*Pz*):
    - Actualizar Discriminador (*D*):

$$\nabla \frac{1}{2n} \sum_{i=1}^{n} [\log(D(x_i)) + \log(1 - D(G(z_i)))]$$

  end for
  - Se congelan los parámetros del Discriminador
  - Se obtiene un lote de ruido(*Pz*):
  - Se actualiza el Generados(*G*):

$$\nabla \frac{1}{2n} \sum_{i=1}^{n} [\log(D(G(z_i)))]$$

end for

Antes de comenzar con el bucle (ciclo For) se deben definir 3 variables:

- img list = []: Guarda la información de las imagenes generadas.
- G losses = []: Guarda el error del Generador en cada una de las iteraciones hasta completar el bucle.

- D losses = []: Al igual que el G losses guarda el error del Discriminador.

El entrenamiento comienza con un bucle para cada epoch. A continuación, se realiza otro bucle para la actualización del discriminador y posteriormente se actualiza el generador.

4.7.1) Entrenar al Discriminador: Se realiza la actualización del discriminador a partir de la siguiente función, donde el objetivo de entrenar al discriminador es intentar maximizarla

$$F_{max} = Max[log(D(x)) + log(1 - D(G(z)))] \tag{17}$$

Donde $D(x)$ es la estimación del discriminador de la probabilidad de que la instancia de datos reales x sea real, $G(z)$ es la salida del generador cuando se da el ruido z y $D(G(z))$ es la estimación del discriminador de la probabilidad de que una instancia falsa sea real.

Y siguiendo la literatura, se lleva cabo este procedimiento en tres partes.

1. **Entrenar con todos los lotes reales:**
   Se inicia pasando el lote de muestras reales por el discriminador. Luego se calcula el error de todo el lote con la función BCELoss y, por último, se calcula el gradiente para el discriminador en un paso hacia atrás (backward).

2. **Entrenar con todos los lotes falsos:**
   En este caso se genera un lote de vectores latentes, con ayuda de la función *"torch.randn()"*, la cual devuelve un tensor lleno de números aleatorios para la simulación de ruido. Luego, se produce un lote de imágenes falsas con el generador, este lote se clasifica con el discriminador. Ahora, se debe calcular el error del discriminador con el lote falso y calcular los gradientes de este lote.

3. **Actualizar Discriminador:** Para que posteriormente se unan los gradientes de cada lote y con ello se actualice el Discriminador con la ayuda del optimizador Adam.

4.7.2) Entrenar al Generador: En el discriminador se desea maximizar dicha función, pero en el caso del generador, se desea minimizar una parte de dicha función con el objetivo de proporcionar mejores falsificaciones

$$F_{min} = Min[1 - log(D(G(z)))] \tag{18}$$

Pero, según la literatura, minimizar esa función proporciona gradientes muy pequeños al inicio del entrenamiento debido a que el discriminador puede rechazar muestras con muy alta probabilidad. Esto se debe a que el discriminador mejora más rápido que el generador. Es por ello que se entrena para

maximizar la siguiente función

$$F_g = Max[log(D(G(z)))] \tag{19}$$

Se mantienen los parámetros del discriminador congelados porque el generador necesita la salida del discriminador, ya que los parámetros de este último se ven involucrados en los cálculos de gradientes. Luego, se pasa un lote falso por el discriminador y a partir de la respuesta de este se calcula la pérdida del generador. Posteriormente, se actualiza el generador con el optimizador Adam. Esto se observa en el Algoritmo 1. Se aclara que el proceso se realiza varias veces para el discriminador más que para el generador, y esto sucede para mantener al discriminador cerca de su óptimo.

4.8. Visualización de los paisajes

En la sección 4.7 se definieron 3 variables. La primera corresponde a una matriz de 128x128x1, la cual se encarga de guardar la información píxel por píxel de la imagen generada. Esta variable se va actualizando a medida que va cambiando de epochs.

V. RESULTADOS

En este capítulo se exponen los resultados obtenidos por la red, la cual fue puesta a prueba con diferentes bases de datos. Los resultados gráficos se presentan en secciones, cada sección representa una base de datos específica y contiene tres figuras. La primera es un gráfico del error del generador y el error del discriminador a lo largo de las epochs. La segunda es un paisaje tridimensional que se realiza a partir de la información obtenida por la red para ser representado con la ayuda del software de Matlab. El tercero, último y el objetivo del proyecto, son algunos paisajes bidimensionales de optimización generados por la red.

5.1. Formas de validación

5.1.1) Estadísticos de la DCGAN: Se consideran cuatro estadísticos para determinar el funcionamiento y buen entrenamiento de la DCGAN. Los dos primeros estadísticos se usan para determinar el momento en que el entrenamiento del modelo debe terminarse, esto según el equilibrio de Nash. Los otros dos se usan porque en una red GAN se pretende llegar a ese valor del $50\,\%$ en el error de clasificación del discriminador. A continuación, se define cada uno de ellos con una breve explicación de por qué se escogen.

- Perdidas en el Discriminador(PD): Se calcula como la suma de las pérdidas de todos los lotes reales y falsos. Dicha pérdida penaliza al discriminador por hacer una clasificación errónea de una instancia real como una falsa o de una instancia falsa como una real[35].

$$P.Discriminador = \log[1 - D(G(z))] \tag{20}$$

  Donde z hace referencia al vector de ruido y entrada del generador.

- Perdidas en el Generador(PG): Esta pérdida penaliza al generador por no ser capaz de engañar al discriminador. Como cada pérdida penaliza a su red por no cumplir con el objetivo, la pérdida del generador comienza en un valor alto y comienza a decrecer, mientras que la pérdida del discriminador debe comenzar baja e ir aumentando hasta cierto punto donde debe mantenerse constante o en valores muy cercanos entre las pérdidas pasadas, presentes y futuras.

$$P.Generador = \log[D(G(z))] \tag{21}$$

- D(x): Es la estimación que hace el discriminador de la probabilidad de que la salida sea real con un valor de entrada x real o, en otras palabras, la salida promedio del discriminador para el lote

real[42][35]. D(x) es un estadístico importante para determinar el avance o buen progreso de la red. Esto sucede porque el discriminador debe comenzar en el valor de 1, ya que es sencillo para él clasificar entre un valor generado y uno real. Pero al transcurrir el entrenamiento del generador, este mejora y con ello mejora ese valor que debe ser clasificado. Por lo tanto, el valor de D(x) debe comenzar a decrecer hasta el punto en que no puede distinguir un dato falso de uno real, y esto sucede cuando D(x) está sobre los 0.5.

$$D(x) = \frac{P_{data(x)}}{P_{data(x)} + P_{g(x)}} \tag{22}$$

Donde $P_{data(x)}$ es la distribución de probabilidad de los datos reales, $P_{g(x)}$ es la distribución de probabilidad de los datos producidos por el Generador y x pertenece a una distribución desconocida[47].

- D(G(z)): Este estadístico es similar a D(x) pero ahora con el lote falso. Entonces, al igual que antes, se debe llegar al valor de 0.5, pero no comienza en el valor de 1, sino que empieza en 0 y debe crecer, no decrecer.

$$D(G(z)) = \frac{P_{g(z)}}{P_{data(x)} + P_{g(z)}} \tag{23}$$

Donde $P_{data(x)}$ es la distribución de probabilidad de los datos reales, $P_{g(z)}$ es la distribución de probabilidad de los datos producidos por el Generador y z pertenece a una distribución del vector de ruido[47].

5.1.2) Diagrama de cajas y bigotes: Es un gráfico utilizado comúnmente para distribuciones de grupo o para resumir visualmente un grado de simetría[48]. Las partes importantes de este gráfico son las posiciones del percentil 25, 50 y 75. El percentil 50 es la mediana. Los bigotes se extienden hasta el dato más alejado de 1.5 veces el rango intercuartil desde el respectivo borde de la caja[49].
Las ventajas de utilizar este tipo de gráfico son las siguientes:

- Permite visualizar la existencia de aparentes diferencias entre grupos.

- Facilita la visualización del grado de simetría.

- Favorece la observación de la dispersión de la variable y, por lo tanto, resulta más sencillo generalizar y describir las características correspondientes a ese grupo o correspondientes a esa variable[50].

5.2. Metodología de experimentación

Se realizan diferentes pruebas para cada base de datos con el objetivo de determinar los mejores parámetros para cada una. En cada prueba se van cambiando los valores para los siguientes 3 parámetros: Epochs, Tasa de aprendizaje y Tamaño del lote. Los parámetros se escogen a partir de los 4 estadísticos nombrados anteriormente, de la gráfica de pérdidas y de la visualización de los paisajes en 2 y 3 dimensiones.

Una vez fijados los parámetros, se repite 10 veces el entrenamiento de la red para cada base de datos, sacando de cada prueba los últimos valores para el estadístico D(x) y los paisajes bidimensionales.

Por último, se hace un análisis de los resultados obtenidos con las pruebas anteriormente mencionadas y se grafican en un diagrama de cajas y bigotes para confrontar los resultados de las diferentes bases de datos, mirar la influencia del aumento de información o qué tanto contribuye a los resultados la separación de los paisajes por patrones visuales. También se incluye la Tabla X, donde se encuentran los coeficientes de variación que apoyan la explicación del diagrama de cajas y bigotes.

5.3. Parámetros de entrenamiento

En la Tabla VI se presentan los 3 parámetros iniciales que se seleccionan para cada una de las bases de datos. Dichos parámetros pueden hacer que la red aprenda adecuadamente o no.

El primer parámetro es la tasa de aprendizaje. Este parámetro puede demorar más el proceso si se tiene un valor inferior a $9\text{x}10^{-4}$, ya que las actualizaciones de los pesos se van realizando de a poco. Para el objetivo con que se implementa la red, el rango de la tasa de aprendizaje que da pérdidas en el discriminador y generador similares está entre $1\text{x}10^{-3}$ y $5\text{x}10^{-3}$. Con valores más pequeños, el proceso se demoraba 4 horas más y los resultados eran desfavorables; con valores más grandes, se presenta el problema de gradientes que desaparecen y con ello, las pérdidas del discriminador son cercanas a 0 y las pérdidas en el generador cercanas a 100.

El segundo parámetro es el tamaño del lote. Este parámetro cambia la cantidad de iteraciones por cada epoch y dependiendo de ello, el proceso de aprendizaje puede tardar más tiempo del necesario. El tercer y último parámetro son las epochs.

TABLA. VI. Parámetros de entrada

	Tasa de aprendizaje (lr)	Tamaño del lote	Epochs
Base 1	0,001	64	245
Base 2	0,002	16	2000
Base 3	0,002	16	3000
Base 4	0,001	16	2000
Base 5	0,005	128	900
Base 6	0,005	32	400
Base 7	0,0015	32	2000
Base 8	0,0015	32	600

5.4. Resultados visuales

Para los resultados visuales, se cuenta con una gráfica que representa las pérdidas del generador en color azul y las pérdidas del discriminador en color naranja. En el eje y están las pérdidas y en el eje x las iteraciones.

Se utiliza esta gráfica para determinar el momento en que se debe terminar el entrenamiento. En las GANs, precisar la convergencia es complicado, ya que en estas no existe un punto fijo de optimización, sino que es dinámico. En las GANs no se busca la optimización hacia un mínimo, sino el equilibrio entre las dos redes, muchas veces denominado Equilibrio de Nash. Dicho equilibrio se logra cuando ninguna red puede mejorar por sus propios medios.

El problema con este equilibrio es que no es estable sino fugaz. Esto se debe a que cuando se logra una igualdad en las pérdidas del generador y las pérdidas del discriminador, donde D(x) es igual 0,5 y es el punto deseado, se penaliza ambas redes porque solo se está engañando un $50\,\%$ al discriminador y el discriminador solo está clasificando bien un $50\,\%$. Aquí está la razón por la que es necesario este gráfico.

También se dispone de una visualización en 3D de los paisajes de optimización. Por cada base de datos, se presentan 4 imágenes, las cuales representan diferentes paisajes pero todos obtenidos a partir de la misma red, la misma base y los mismos parámetros. Esto se realiza con la intención de contar con una idea de la dificultad que puede presentar dicho paisaje para un algoritmo de optimización. Asimismo, se espera observar en qué categoría puede estar dicho paisaje, en otras palabras, lograr una clasificación o caracterización a partir de patrones gráficos, como la cantidad de mínimos, el ruido, paisajes suaves, con valles prolongados y demás.

En último lugar, se presentan los 4 paisajes bidimensionales de optimización pertenecientes a la visualización 3D y en RGB.

TABLA. VII. D(x)

	Corrida 0	Corrida 1	Corrida 2	Corrida 3	Corrida 4	Corrida 5	Corrida 6	Corrida 7	Corrida 8	Corrida 9	Corrida 10
Base 1	0,543	0,4759	0,5519	0,5029	0,3076	0,6238	0,4678	0,5344	0,3982	0,5953	0,434
Base 2	0,5422	0,6272	0,5536	0,4539	0,4484	0,5426	0,5348	0,7079	0,5231	0,6068	0,5243
Base 3	0,4631	0,5396	0,6234	0,4614	0,6786	0,5625	0,4878	0,5359	0,3916	0,543	0,578
Base 4	0,5614	0,5851	0,6396	0,4004	0,4903	0,4418	0,3796	0,3004	0,5836	0,5071	0,4132
Base 5	0,3944	0,5769	0,4328	0,4119	0,5909	0,7863	0,4172	0,6643	0,6748	0,4714	0,4117
Base 6	0,5301	0,5465	0,4672	0,447	0,5281	0,5857	0,4601	0,472	0,5117	0,7689	0,4191
Base 7	0,5663	0,5484	0,5612	0,6877	0,5291	0,5685	0,5206	0,4969	0,5654	0,4344	0,5563
Base 8	0,5783	0,5639	0,5932	0,5454	0,498	0,4902	0,6119	0,4524	0,6268	0,5045	0,5606

Se tienen 88 paisajes bidimensionales de optimización los cuales se encuentran en el Apéndice A. Los paisajes expuestos en esta sección hacen referencia a los 4 mejores paisajes de cada base de datos. Dichos paisajes se seleccionan a partir del estadístico D(x), como se observa en la Tabla VII. Cuando se habla de corrida 0 hace referencia al entrenamiento de la red cuando se escogen los parámetros iniciales.

5.4.1) Base 1: En la Figura 32 se observa cómo las pérdidas van disminuyendo al pasar las iteraciones. Como se explica en el marco teórico, las iteraciones están ligadas a las epochs. Se habla en iteraciones ya que siempre la cantidad de iteraciones será igual o mayor que las epochs y, por ende, se tiene un rango más amplio para determinar el comportamiento de la red si las gráficas de pérdidas se trabajan en iteraciones.

En esta base se cuenta con todos los paisajes disponibles. La Figura 33 muestra imágenes 3D algo ruidosas, algunas con un solo mínimo, mientras que otras con varios. Si se observa la Figura 34, encontramos paisajes con algunas similitudes a los originales, pero como se describió anteriormente, son paisajes con presencia de ruido.

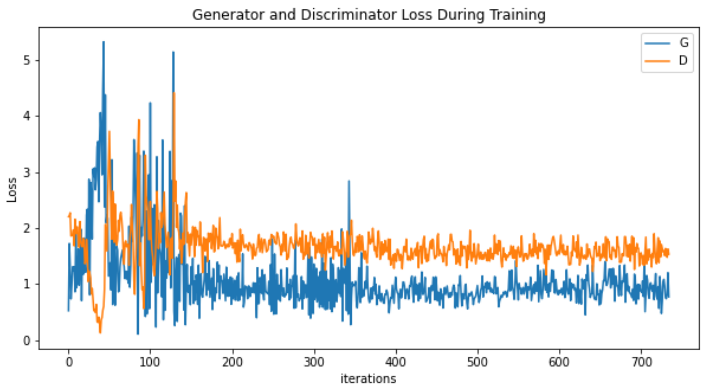
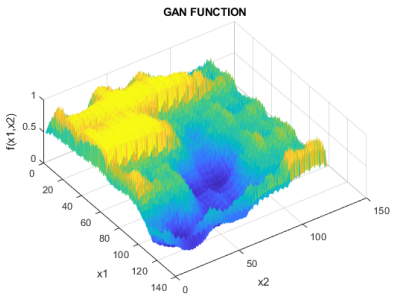
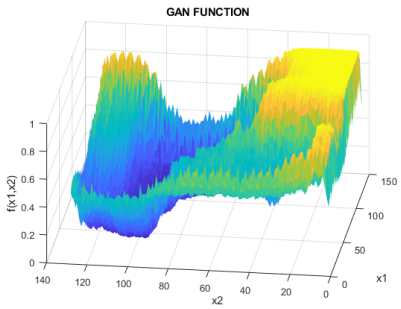


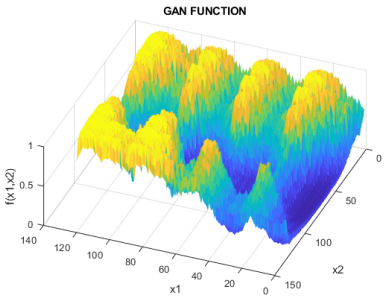
Fig. 32. Perdidas durante el entrenamiento Base 1



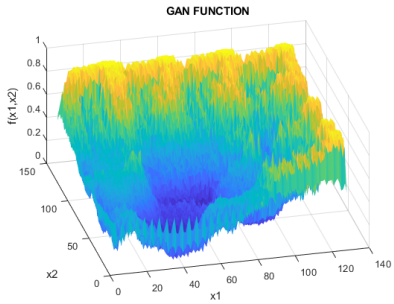
(a) Corrida 1



(b) Corrida 3

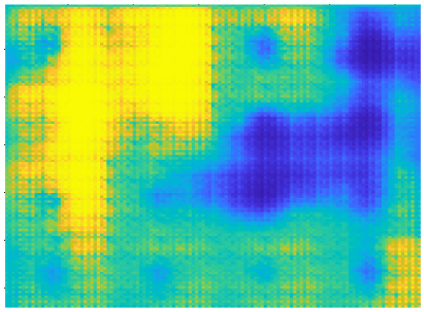


(c) Corrida 6

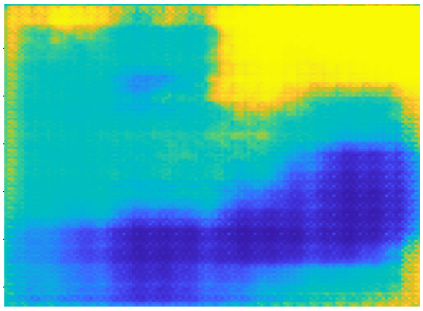


(d) Corrida 7

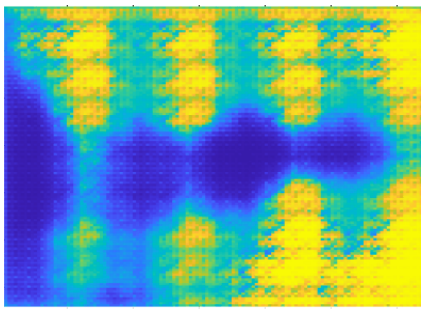
Fig. 33. Paisajes tridimensionales Base 1



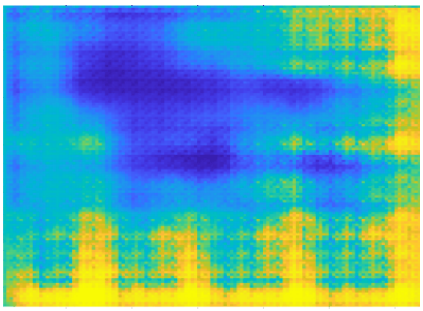
(a) Corrida 1



(b) Corrida 3



(c) Corrida 6



(d) Corrida 7

Fig. 34. Paisajes tridimensionales Base 1

5.4.2) Base 2: Como se mencionó anteriormente, las pérdidas del discriminador deben aumentar y luego disminuir, mientras que las pérdidas del generador comienzan en un valor grande y luego decrecen.

Esto se puede ver en la Figura 35, lo cual indica un progreso positivo al transcurso del entrenamiento.

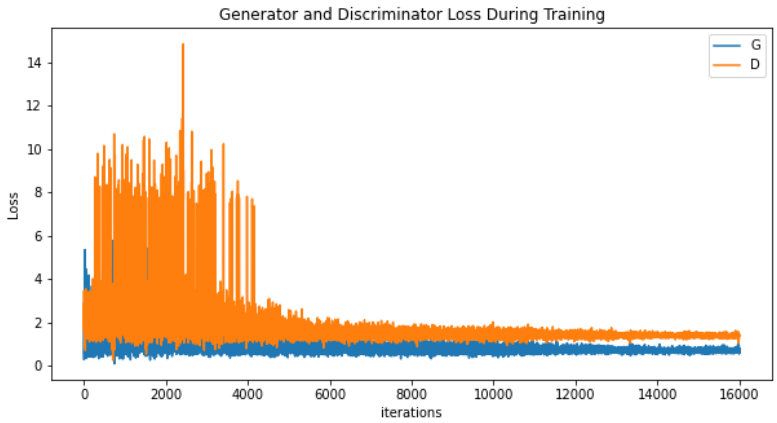
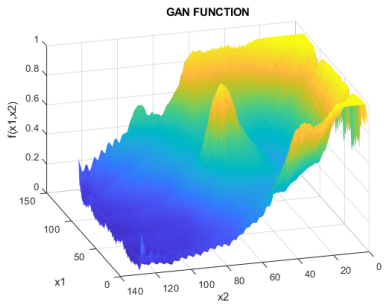
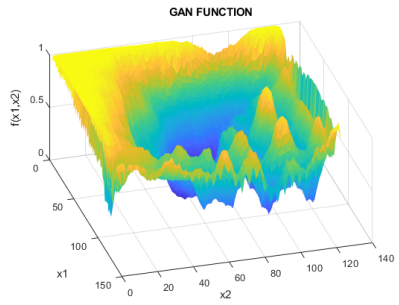


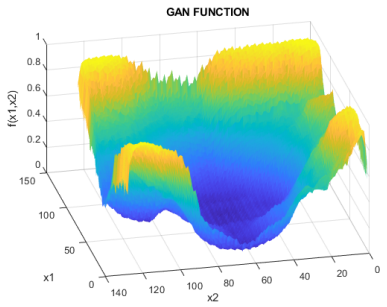
Fig. 35. Perdidas durante el entrenamiento Base 2



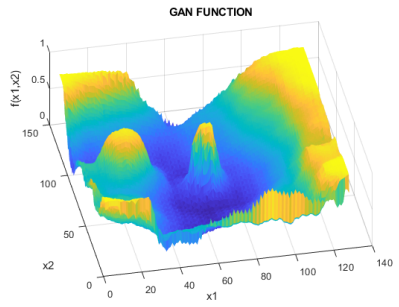
(a) Corrida 0



(b) Corrida 6



(c) Corrida 8



(d) Corrida 10

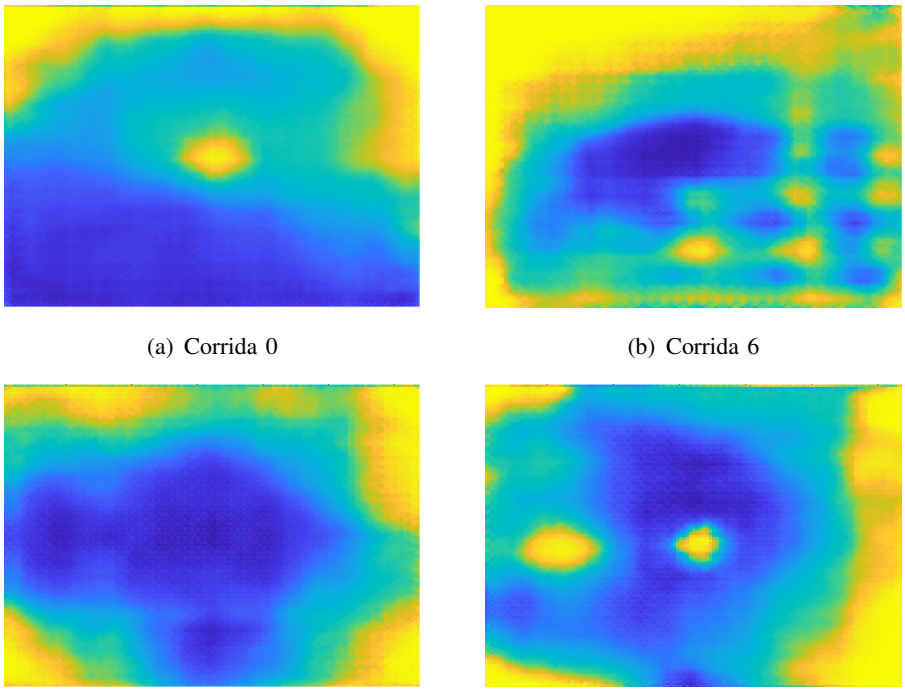
Fig. 36. Paisajes tridimensionales Base 2

Si se comparan los paisajes de la Figura 34 y los paisajes de la Figura 37, es notable la disminución en el ruido, lo que genera paisajes con una mayor similitud a los paisajes originales que con la base de datos anterior.

Al analizar la Figura 36 a), se afirma la idea de poder generar paisajes nuevos. Este paisaje presenta un máximo en el centro de la figura, a mitad de llegar al mínimo global. Normalmente, estos paisajes

presentan una disminución en sus valores de manera lenta hasta llegar al valor del mínimo, pero en este caso, existe la presencia de ese pico, lo que generaría problemas o dificultades a los algoritmos de optimización.

5.4.3) Base 3: Al entrenar una red DCGAN o GAN en general, existe el problema de la no convergencia. Es por ello que en todas las bases se escoge el punto donde las gráficas de pérdidas del generador y el discriminador son cercanas y así dar por finalizado el entrenamiento, como se observa en la Figura 38.



(a) Corrida 0 (b) Corrida 6

(c) Corrida 8 (d) Corrida 10

Fig. 37. Paisajes bidimensionales Base 2

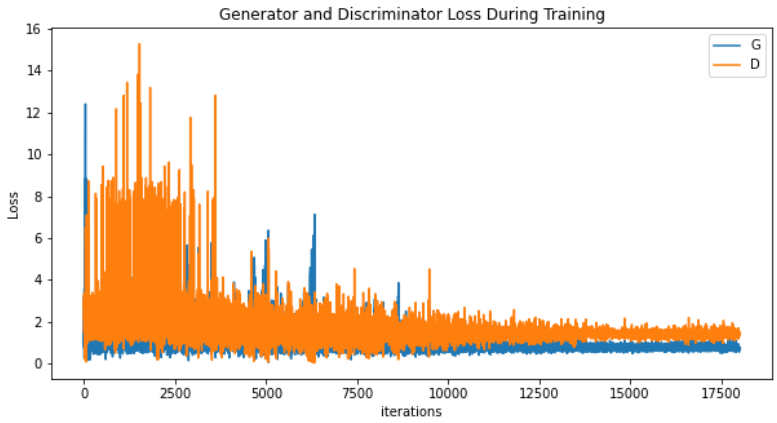
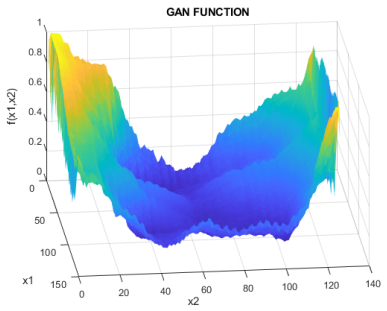
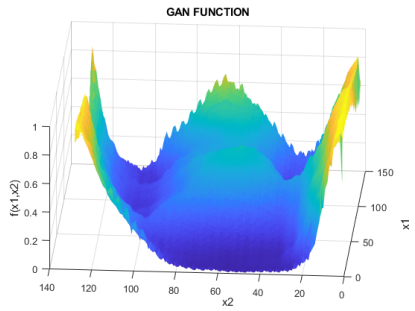


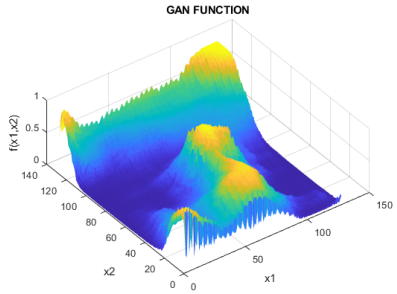
Fig. 38. Perdidas durante el entrenamiento Base 3



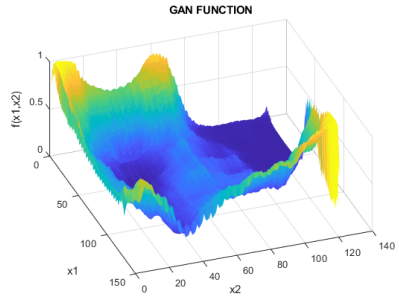
(a) Corrida 0



(b) Corrida 1

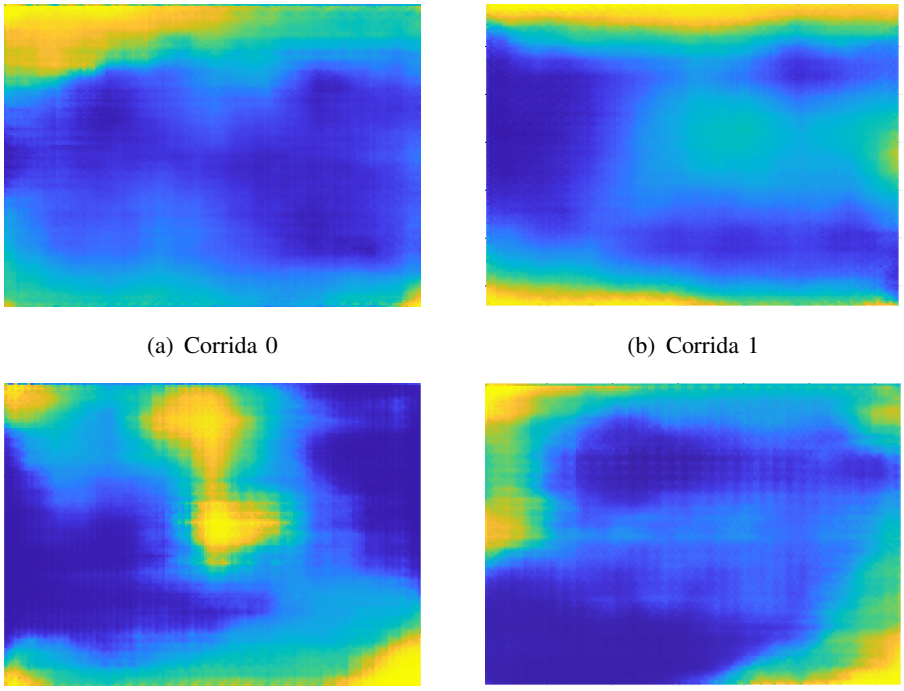


(c) Corrida 6



(d) Corrida 7

Fig. 39. Paisajes tridimensionales Base 3



(a) Corrida 0 (b) Corrida 1

(c) Corrida 6 (d) Corrida 7

Fig. 40. Paisajes bidimensionales Base 3

Esta base de datos cuenta solamente con paisajes de un solo mínimo. Los paisajes generados intentan seguir este patrón, pero no lo cumplen del todo. Aparece información nueva, como lo son los máximos en la Figura 39 c). Aun así, la disminución en presencia de mínimos a comparación de la Figura 36 es muy notoria, como también se puede notar en la Figura 36.

5.4.4) Base 4: En la Figura 41 se tiene una pérdida grande en el generador, pero luego una disminución notoria al finalizar las iteraciones, lo cual indica un buen aprendizaje y el momento correcto para terminar el entrenamiento, ya que las pérdidas de ambas redes son cercanas.

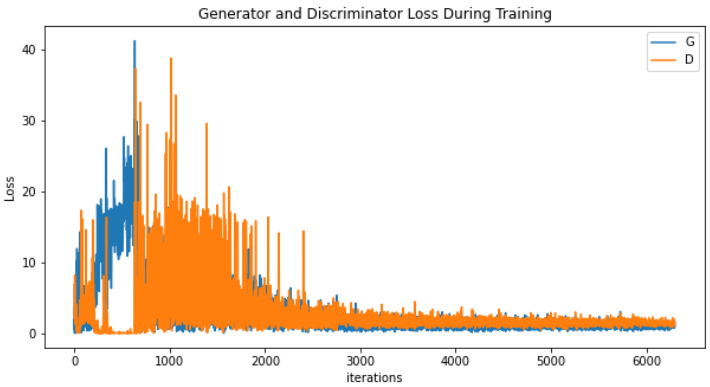
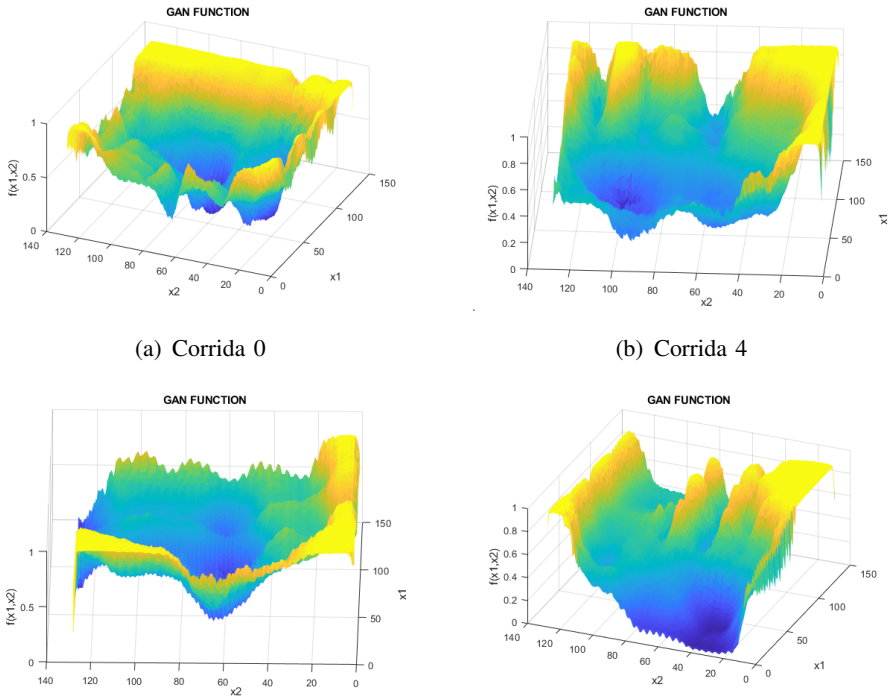


Fig. 41. Perdidas durante el entrenamiento Base 4

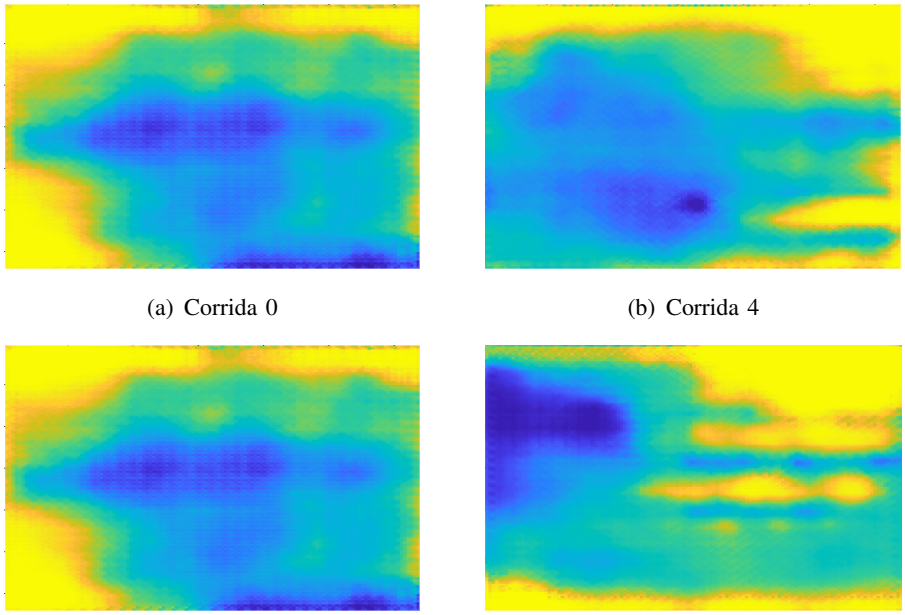


(a) Corrida 0 (b) Corrida 4

(c) Corrida 5 (d) Corrida 9

Fig. 42. Paisajes tridimensionales Base 4

Aunque se disminuye la cantidad de información con la que cuenta esta base de datos, al limitar la información a un patrón específico, se manifiesta un resultado adecuado y sin perder tanta diversidad como podría creerse al reducir la información. La Figura 42 y la Figura 43 muestran resultados diferentes y con menos ruido en comparación con la Figura 34 y la Figura 33.



(a) Corrida 0 (b) Corrida 4

(c) Corrida 5 (d) Corrida 9

Fig. 43. Paisajes bidimensionales Base 4

5.4.5) Base 5: Si se observa la Figura 44, las pérdidas del generador y las pérdidas del discriminador varían bastante aun en la parte final. Aunque estos son los mejores parámetros para la Base 5, existen más cambios repentinos en los valores de las pérdidas con respecto a las otras bases de datos.

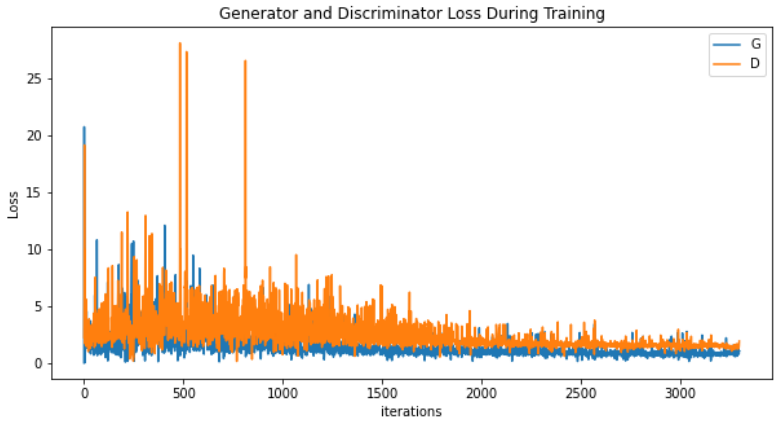
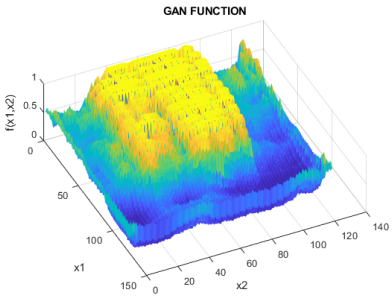
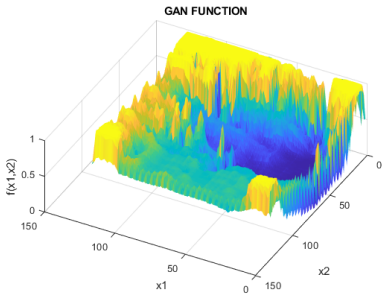


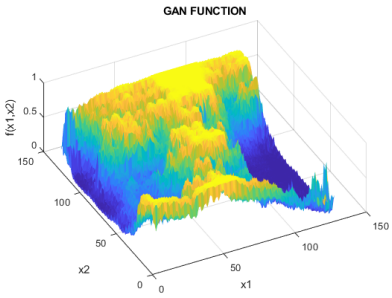
Fig. 44. Perdidas durante el entrenamiento Base 5



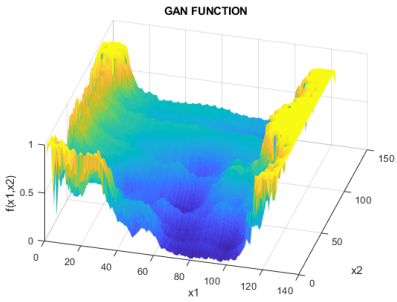
(a) Corrida 1



(b) Corrida 2

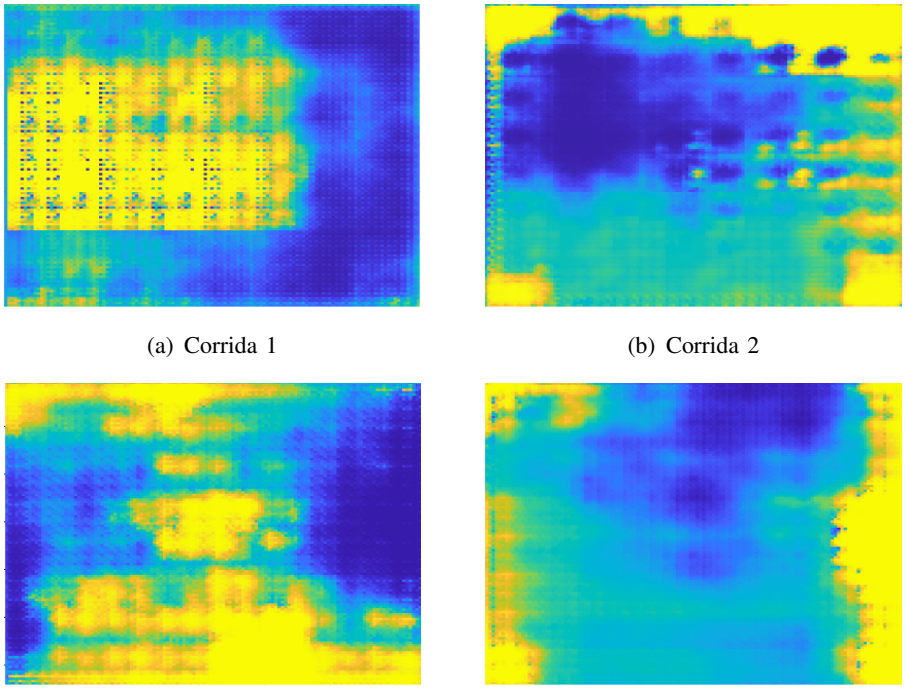


(c) Corrida 6



(d) Corrida 9

Fig. 45. Paisajes tridimensionales Base 5



(a) Corrida 1

(b) Corrida 2

(c) Corrida 6

(d) Corrida 9

Fig. 46. Paisajes bidimensionales Base 5

Esta base de datos presentó paisajes bidimensionales con no muy buena resolución, y al graficar los resultados en 3 dimensiones, la presencia de ruido es notoria. En la Figura 45 se percibe todo ese ruido, y en la Figura 46 se advierte la pérdida en similitud con los paisajes originales.

5.4.6) Base 6: Existe un valor de pérdida del discriminador inmenso al comienzo del entrenamiento de la red, pero como se puede ver en la Figura 47, las pérdidas comienzan a decrecer hasta cierto punto, inferior a 2, y se mantienen a lo largo del entrenamiento.

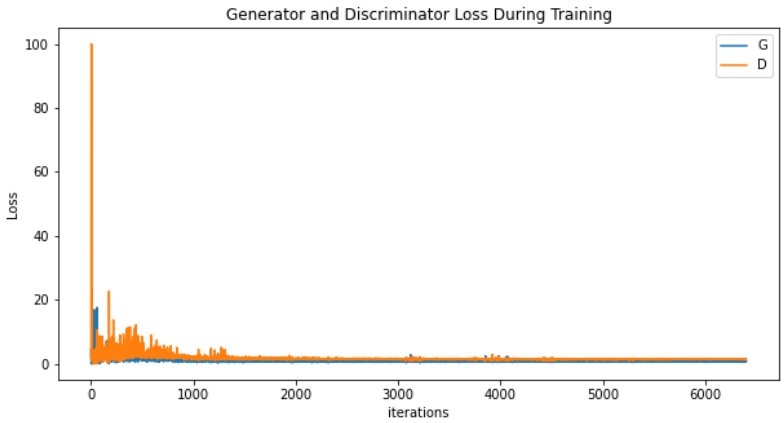
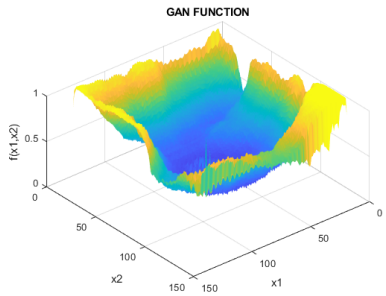
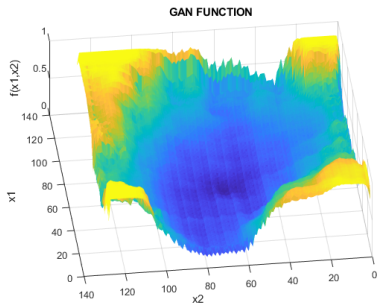


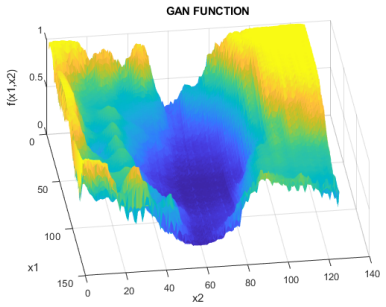
Fig. 47. Perdidas durante el entrenamiento Base 6



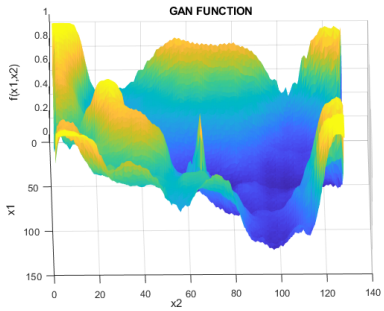
(a) Corrida 2



(b) Corrida 4



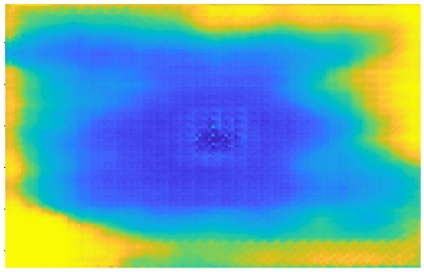
(c) Corrida 7



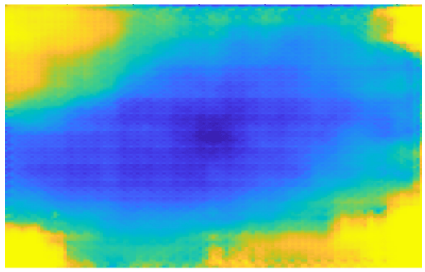
(d) Corrida 8

Fig. 48. Paisajes tridimensionales Base 6

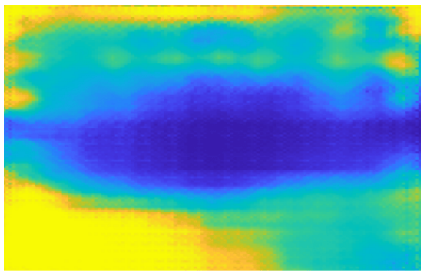
Al igual que en la Base 2, las Figuras 49 y 48 presentan gran diversidad y disminución en los paisajes ruidosos y no presentan picos. La diferencia con los paisajes de la Base 2 no es tan notoria como para ratificar la necesidad de usar un Data Augmentation.



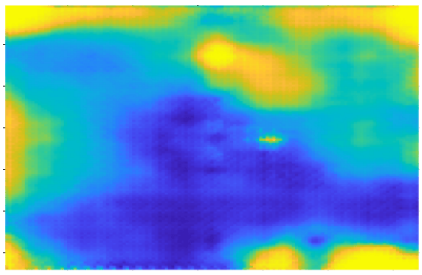
(a) Corrida 2



(b) Corrida 4



(c) Corrida 7



(d) Corrida 8

Fig. 49. Paisajes bidimensionales Base 6

5.4.7) Base 7: Aunque en la Figura 50 se observan unos cambios fuertes en las pérdidas al comienzo del entrenamiento y no se estabilizan tan rápido como en las anteriores figuras, sí muestra resultados visuales adecuados con respecto a la información original en las Figuras 52 y 51, lo que indica que el punto donde se decide finalizar el entrenamiento es el correcto.

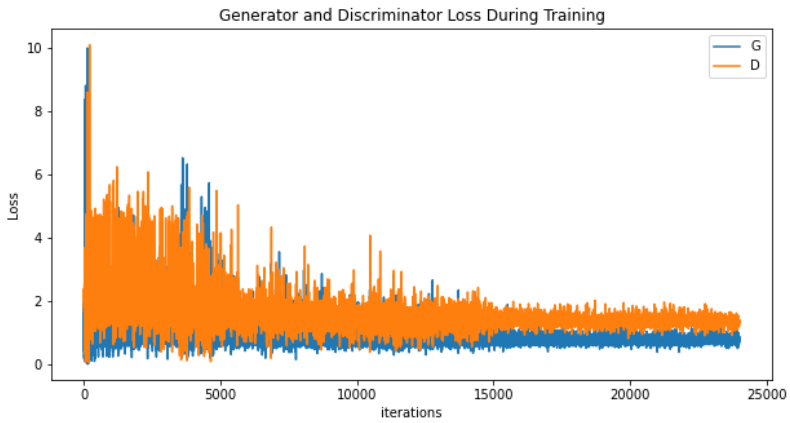
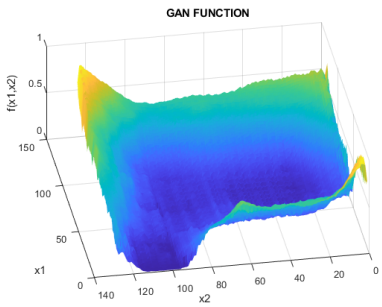
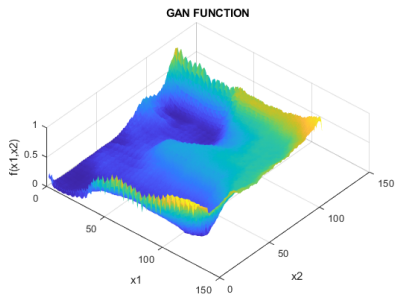


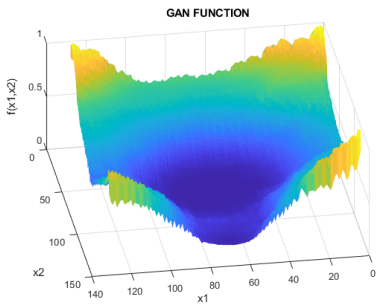
Fig. 50. Perdidas durante el entrenamiento Base 7



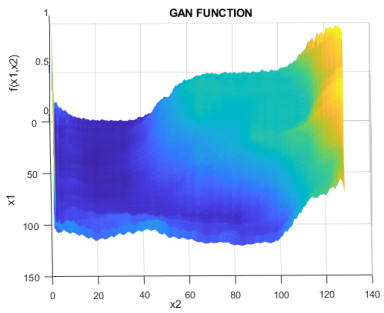
(a) Corrida 1



(b) Corrida 4

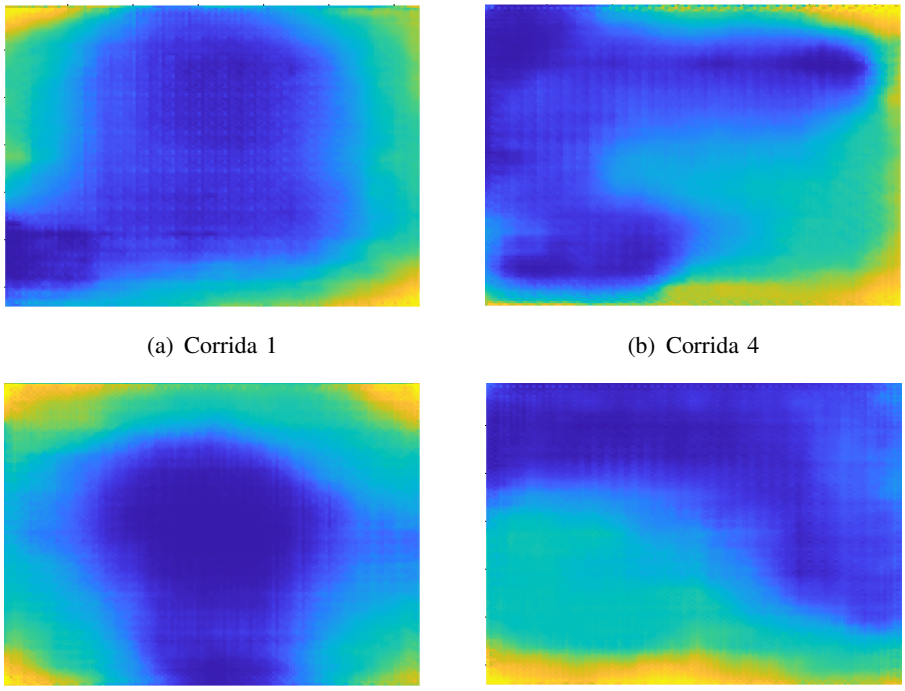


(c) Corrida 6



(d) Corrida 7

Fig. 51. Paisajes tridimensionales Base 7



(a) Corrida 1

(b) Corrida 4

(c) Corrida 6

(d) Corrida 7

Fig. 52. Paisajes bidimensionales Base 7

Al observar los resultados visuales como los resultados de los estadísticos de cada base, se obtiene que esta base de datos presenta resultados visualmente adecuados según la información original. Son paisajes con cambios suaves en los alrededores de los mínimos, lo cual sucede con los paisajes de origen, y además, no presentan ruido.

5.4.8) Base 8: Esta base presenta, en comparación con la Base anterior, unos paisajes tridimensionales más diversos, con unos cambios más fuertes entre mínimos, pero sin agregar ruido por ello y con una presencia de cuencas y máximos. La Figura 54 muestra varios máximos juntos y a la vez presenta un solo mínimo en la Figura 54 b).

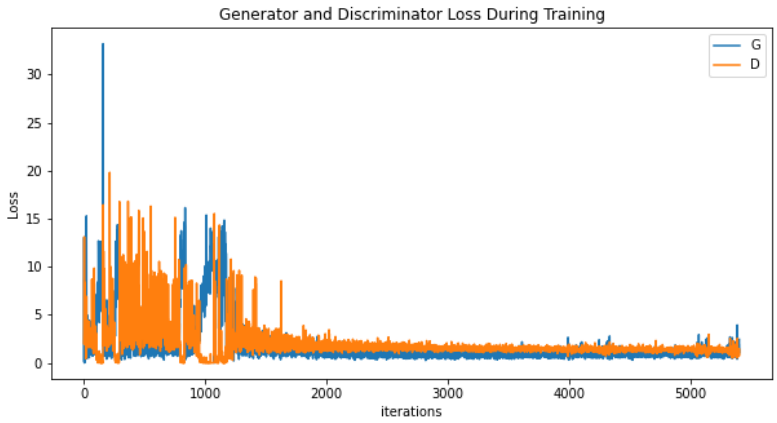
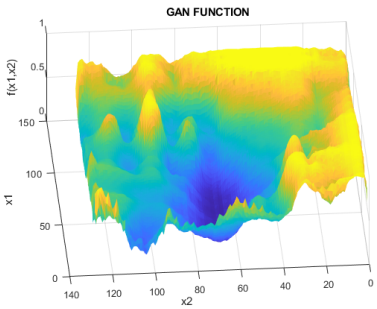
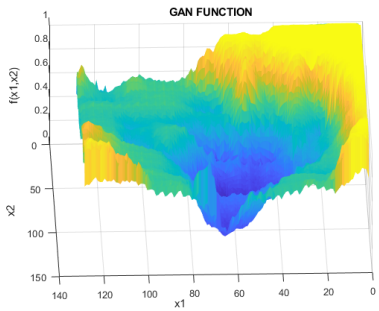


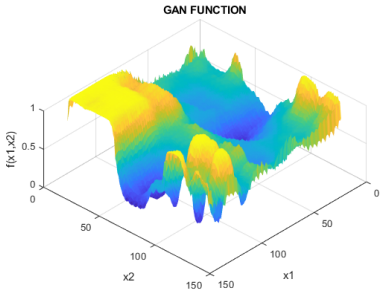
Fig. 53. Perdidas durante el entrenamiento Base 8



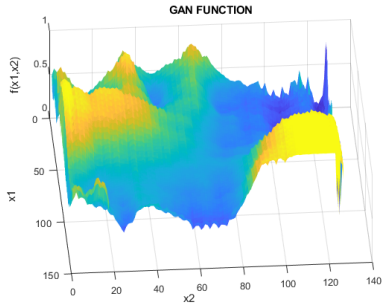
(a) Corrida 4



(b) Corrida 5

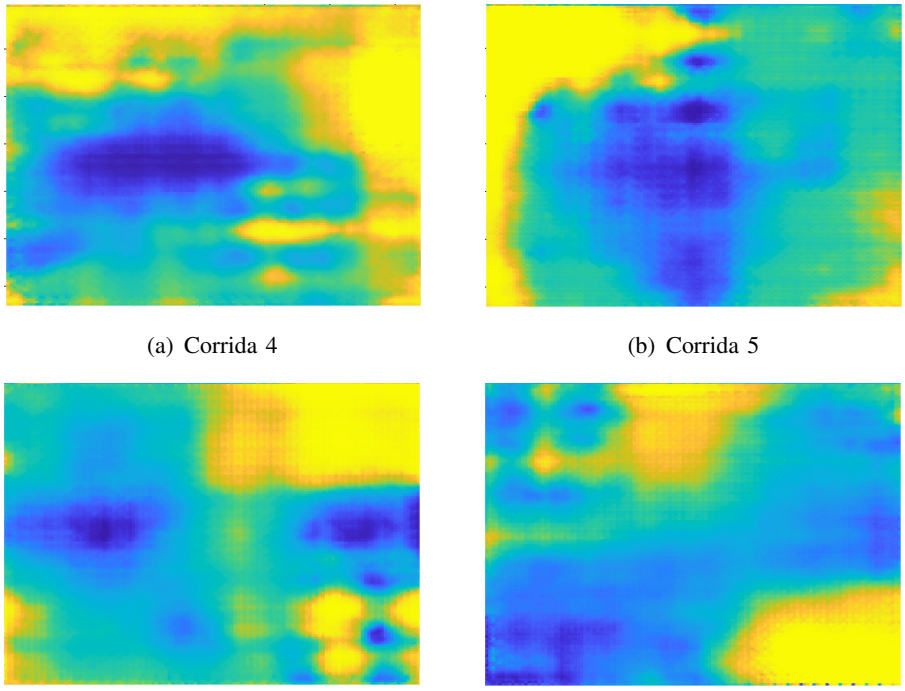


(c) Corrida 7



(d) Corrida 9

Fig. 54. Paisajes tridimensionales Base 8



(a) Corrida 4 (b) Corrida 5

(c) Corrida 7 (d) Corrida 9

Fig. 55. Paisajes bidimensionales Base 8

Una diferencia entre las Base 7 y la Base 8 es la presencia de máximos. En la Figura 55 se muestra una mayor cantidad de valores altos y en diferentes puntos del paisaje, mientras que en la Figura 52 aparecen en los bordes.

5.5. Comparación:

En esta parte se realiza una comparación entre las diferentes bases empleando los estadísticos descritos en la sección 5.1.1. Se divide en dos partes: la primera hace referencia a las pérdidas en ambas redes, y la segunda, a la salida promedio del discriminador para el lote falso y el lote real.

Cuando se realiza la comparación con referencia al estadístico **D(x)**, también se incluyen diagramas de cajas y bigotes para cada base.

5.5.1) Perdidas en el Generador(PG) y en el Discriminador (PD): Lo ideal para finalizar el entrenamiento en una red GAN es que las pérdidas de las dos redes sean iguales, lo que significa que sus distribuciones de probabilidad sean iguales $P_z = P_{data}$.

En la Tabla VIII, se exponen las pérdidas de cada una de las redes y la diferencia entre ellas para las 8 bases de datos. Se evidencia que la diferencia entre las pérdidas de la Base 5 es mayor con respecto a las

otras bases de datos, lo cual corrobora que el aumento de información en dicha base de datos empeoró el aprendizaje del modelo. También se observa que la diferencia entre PD y PG para la Base 8 es muy pequeña, lo cual se acerca al punto deseado.

Todos los valores son inferiores a 2 en ambas redes, lo cual es algo positivo, ya que estas pérdidas pueden llegar a 100 como valor máximo y a 0 como valor mínimo. En algunos casos, las pérdidas del generador llegaron al valor de 100, mientras que las pérdidas del discriminador llegaron a 0. Por lo tanto, todos los valores obtenidos y expuestos en la Tabla VIII son evidencia del adecuado aprendizaje de la red con los parámetros seleccionados.

TABLA. VIII. Perdidas en el Generador y en el Discriminador

	Perdidas en el Generador	Perdidas en el Discriminador	Diferencia entre perdidas
Base 1	0,8237	1,4787	0,655
Base 2	0,6185	1,3488	0,73
Base 3	0,6835	1,2627	0,579
Base 4	1,143	1,3716	0,229
Base 5	0,5964	1,5144	0,918
Base 6	0,6742	1,3905	0,716
Base 7	0,8868	1,3928	0,506
Base 8	1,1768	1,0435	0,133

5.5.2) Salida promedio del Discriminador para el lote real($D(x)$) y para el lote falso($D(G(z))$): Lo ideal en esta parte es tener un $D(x)$ igual a 0.5, al igual que $D(G(z))$. En la Tabla IX, se observa un rango para estos estadísticos que oscila entre 0.38 a 0.58, lo que indica resultados positivos en cuanto al entrenamiento de la red. La Base 2 y la Base 6 comparten la misma información de origen y son las que presentan un valor relativamente cercano entre sí, a comparación de las demás bases. Nuevamente, la Base 5 presenta el valor más lejano del valor esperado, todo debido a los paisajes con presencia de ruido.

La Tabla X muestra los coeficientes de variación para los valores de D(x) de cada una de las bases de datos y ratifica lo mencionado anteriormente. También se observa cómo el Data Augmentation mejora la respuesta para las bases sin ruido y empeora la respuesta para la base con ruido.

TABLA. IX. Salidas promedio del Discriminador para el lote falso y el lote real

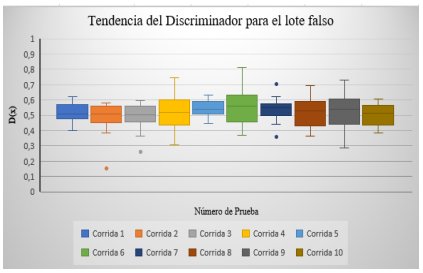
	D(x)	D(G(z))
Base 1	0,543	0,5201
Base 2	0,5422	0,5478
Base 3	0,4631	0,5113
Base 4	0,5614	0,4662
Base 5	0,3944	0,5833
Base 6	0,5301	0,5257
Base 7	0,5163	0,5127
Base 8	0,5593	0,385

TABLA. X. Coeficiente de variación para el estadístico D(x)

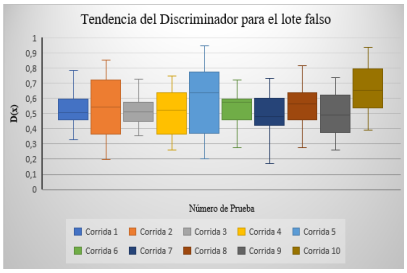
	Corrida 1	Corrida 2	Corrida 3	Corrida 4	Corrida 5	Corrida 6	Corrida 7	Corrida 8	Corrida 9	Corrida 10
Base 1	11,39 %	20,61 %	17,03 %	22,72 %	9,91 %	21,19 %	16,63 %	18,86 %	23,10 %	14,43 %
Base 2	24,22 %	16,79 %	32,45 %	22,64 %	29,83 %	24,45 %	21,29 %	19,20 %	13,95 %	19,04 %
Base 3	11,63 %	10,65 %	17,61 %	12,89 %	14,44 %	11,08 %	17,33 %	15,80 %	9,77 %	17,70 %
Base 4	21,45 %	15,62 %	21,18 %	15,03 %	21,66 %	24,56 %	23,78 %	14,24 %	21,72 %	27,57 %
Base 5	21,29 %	36,64 %	18,41 %	30,21 %	37,05 %	19,59 %	29,03 %	24,58 %	30,20 %	23,43 %
Base 6	8,21 %	11,84 %	7,74 %	9,49 %	13,55 %	15,54 %	10,11 %	24,73 %	22,38 %	13,11 %
Base 7	19,73 %	11,73 %	19,67 %	10,73 %	15,58 %	11,43 %	10,70 %	7,96 %	12,71 %	11,54 %
Base 8	20,38 %	19,80 %	13,00 %	10,80 %	11,22 %	14,64 %	14,61 %	17,39 %	15,27 %	13,41 %

En la Figura 56, se presenta un diagrama de cajas y bigotes correspondiente a cada base de datos. Esto se realiza debido a la presencia de aleatoriedad, ya que podría presentarse la ocasión donde la red aprende muy bien para un caso en específico, sea con un comienzo adecuado para los pesos iniciales o en el vector de ruido que tiene como entrada el generador, lo que podría ocasionar una muy buena respuesta en una sola ocasión. Por lo tanto, se realizan 10 corridas independientes a cada base de datos con las mismas condiciones iniciales y se observa la respuesta del estadístico $D(x)$ para las últimas 20 epochs. Esto se hace para contar con la tendencia de este estadístico en cada base de datos y ver cómo puede variar de una corrida a la otra.

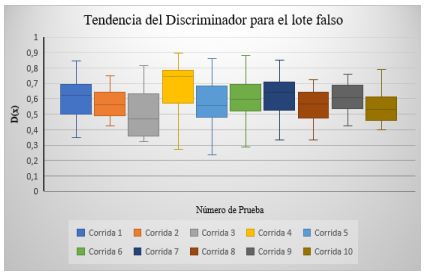
En la Figura 56 b), se observa una mayor oscilación del estadístico en comparación con las demás bases de datos, mientras que en la Figura 56 f), se descubre la mejor base de datos con la que la DCGAN brinda mejores resultados. Esto no solo se verifica en este diagrama, sino que si se observa en los resultados visuales de la sección 5.4.7, se tienen paisajes con mayor similitud a los originales y, a la vez, con presencia de información nueva, poca presencia de ruido, una resolución mejor y una visualización en 3 dimensiones con escasez en picos formados por ruido, como no sucede en la sección 5.4.5.



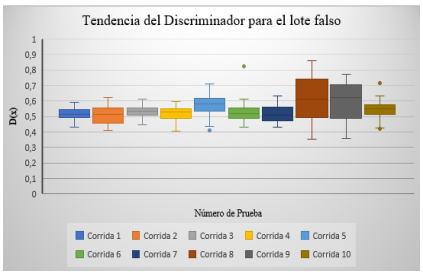
(a) Base 1



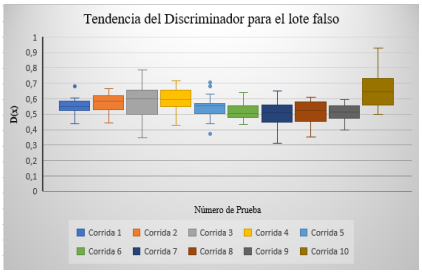
(b) Base 5



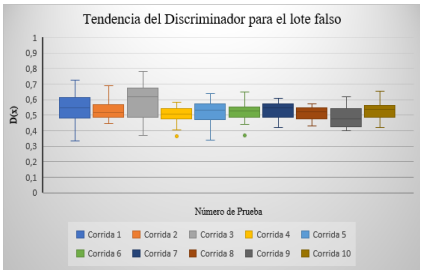
(c) Base 2



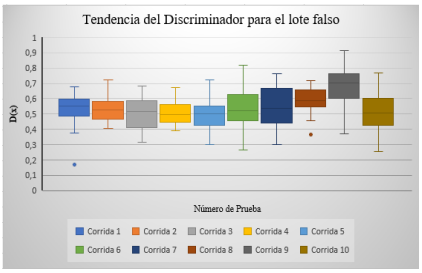
(d) Base 6



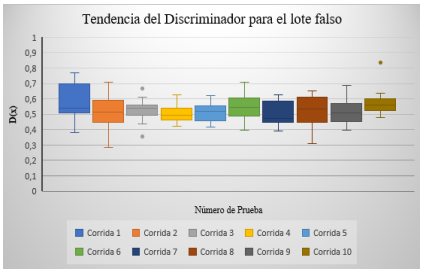
(e) Base 3



(f) Base 7



(g) Base 4



(h) Base 8

Fig. 56. Diagrama de cajas y bigotes para la variable D(x)

Se determina la Base 7 como la mejor, ya que sus estadísticos promedio de la salida del discriminador son los más cercanos a 0.5. Además, si se observa la Tabla X y la Figura 56, se evidencia que la Base 7 presenta los mejores resultados en comparación con las demás bases de datos.

Anteriormente se mencionó la presencia de aleatoriedad. Esta presencia se observa en la Figura 55 d), donde el estadístico $D(x)$ responde con valores cercanos al 0.5 en la mitad de sus corridas independientes, pero en otras tiene unos datos alejados de esa media, presentando valores atípicos que no son deseables. Dichos valores son resultado de tener un paisaje que no confunde muy bien al discriminador en esa epoch,

por lo que la red está teniendo muchos saltos entre valores en esta base.

Por otro lado, la Base 5 muestra que, a diferencia de la Base 7, el Data Augmentation no siempre mejora las respuestas de las redes neuronales artificiales o, en este caso, de una DCGAN.

VI. CONCLUSIONES

Se implementó un método que utiliza redes generativas adversarias, el cual fue capaz de generar nuevos y diversos paisajes bidimensionales de optimización. Los resultados obtenidos presentan características similares y, a su vez, diferentes en comparación con los paisajes existentes y obtenidos de la literatura.

Se utilizó una DCGAN como base del algoritmo para la producción de paisajes bidimensionales, ya que este tipo de red está diseñada a partir de redes convolucionales, las cuales han presentado resultados satisfactorios en el procesamiento de imágenes. Esto demuestra que no es necesario una red GAN que cuente con varios generadores o discriminadores, sino que es suficiente con la arquitectura básica y una pequeña modificación en las redes del generador y el discriminador.

Al agrupar la información en diferentes bases de datos, se confrontan y relacionan entre ellas, logrando de esta manera determinar que:

- La presencia de ruido en los paisajes bidimensionales entorpece el aprendizaje de la red. Los resultados obtenidos de paisajes ruidosos presentan una resolución baja y también presentan un mayor ruido en comparación con los paisajes originales. Al aumentar la cantidad de información, también se aumentó la cantidad de paisajes ruidosos, lo cual corrobora el entorpecimiento por ruido.

- Al variar la información presente en las bases de datos, también se debe realizar un cambio en los parámetros de la red. Se establece que una mala selección de dichos parámetros daña por completo la respuesta de la red y, por tanto, los resultados serán equívocos. Dichas variaciones pueden llegar a ser de valores mínimos y aun así, el cambio en los resultados sería inmenso. No solamente la respuesta estaría mal, sino que también generaría un costo computacional mayor y tiempo de experimentación adicional innecesario.

- Cuando se aplica el Data Augmentation en la base de datos, se esperaba una mejora en los resultados de la red, pero dicha mejora depende de la información en la base de datos a la que se le aplica. Se debe considerar la información que se va a aumentar, ya que esto podría ser causa de una mayor presencia de ruido o, más específicamente, paisajes ruidosos. Lo cual no es deseado ya que lleva a un entorpecimiento en el aprendizaje. Todo se comprueba al confrontar los resultados estadísticos y visuales de la base 1 y 5. Lo anterior hace referencia a un empeoramiento en los resultados a partir del aumento de información. Sin embargo, también se presenta el caso contrario: al aplicar el Data Augmentation a una base homogénea, trae consigo mejoras en los resultados de la red, como se observa en la Tabla X con las últimas 3 bases de datos. Sin embargo, se debe tener en cuenta la pérdida de diversidad en los paisajes.

Además, se concluye que una red DCGAN es capaz de generar paisajes bidimensionales de optimización diversos y nuevos. Dependiendo de la arquitectura de la red, se podrán generar varios paisajes a la vez o, dependiendo de la capacidad de la herramienta utilizada para su implementación, se podrá cambiar la resolución en cantidad de píxeles.

6.1. Aportes originales

- Se conforman 8 bases de datos, compuestas por paisajes bidimensionales de optimización. Donde se presenta una variedad de funciones con características como modalidad, separabilidad, escalabilidad continuidad y diferenciabilidad.

- Se establece un algoritmo que cuenta con pre-procesamiento de información, una red neuronal del tipo DCGAN y una visualización de la información en imágenes de 2 y 3 dimensiones.

6.2. Trabajos Futuros

El trabajo presenta como resultados paisajes bidimensionales de optimización, imágenes de 128x128 píxeles. En un comienzo, se podría mejorar la resolución de los paisajes aumentando la cantidad de píxeles con que se trabaja, lo cual depende de la herramienta con la que se pueda implementar la red. Con ello, sería conveniente una comparación de los resultados para determinar qué tanto influye la resolución en la respuesta de la red.

Como se mencionó anteriormente, el algoritmo entrega imágenes, lo cual lleva al siguiente paso que es la búsqueda de las expresiones matemáticas para las funciones de prueba que dan como resultado dichos paisajes. En esta instancia, se tiene en cuenta la estrecha relación entre las ciencias aplicadas y la ingeniería. El trabajo deja una base que puede ser aplicada en la teoría de la optimización y que posteriormente podría ser utilizada en trabajos de ingeniería.

REFERENCIAS

[1] M. J. Mahmoodabadi y A. R. Nemati, «A novel adaptive genetic algorithm for global optimization of mathematical test functions and real-world problsem,» *Engineering Science and Technology, an International Journal*, vol. 19, n.º 4, págs. 2002-2021, 2016.

[2] H. M. Azamathulla y F. C. Wu, «Support vector machine approach for longitudinal dispersion coefficients in natural streams,» *Applied Soft Computing*, vol. 11, n.º 2, págs. 2902-2905, 2011.

[3] B. Addis y M. Locatelli, «A new class of test functions for global optimization,» *Journal of Global Optimization*, vol. 38, n.º 3, págs. 479-501, 2007.

[4] P. Chen, Z. Shihua, Z. Qiang y N. Kasabov, «A meta-inspired termite queen algorithm for global optimization and engineering design problems,» *Engineering Applications of Artificial Intelligence*, vol. 111, 2022.

[5] R. D. Lang y A. P. Engelbrecht, «An exploratory landscape analysis-based benchmark suite,» *Algorithms*, vol. 14, n.º 3, pág. 78, 2021.

[6] J. Adhikary y S. Acharyya, «Random walk elephant swarm water search algorithm for identifying order-preserving submatrices in gene expression data: a new approach using elephant swarm water search algorithm,» *Recent Trends in Computational Intelligence Enabled Research*, págs. 211-232, 2021.

[7] K. Houston-Edwards, «Machine Learning Becomes a Mathematical Collaborator,» *Quantamagazine*, 2022.

[8] S. Shahriar, «GAN computers generate arts? A survey on visual arts, music, and literary text generation using generative adversarial network,» *Displays*, vol. 73, pág. 102 237, 2022, ISSN: 0141-9382.

[9] N. Barzilay, S. T. Berkovitz y R. Giryes, «MISS GAN: A Multi-IlluStrator style generative adversarial network for image to illustration translation,» *Pattern Recognition Letters*, vol. 151, págs. 140-147, 2021.

[10] M. Jiang, Z. Minghao, L. Wei et al., «FA-GAN: Fused attentive generative adversarial networks for MRI image super-resolution,» *Computerized Medical Imaging and Graphics*, vol. 92, pág. 101 969, 2021.

[11] V. Novo Sanjurjo, *Teoría de la optimización*. UNED. Universidad Nacional de Educación a Distancia, 1999.

[12] G. Jiménez Lozano, *Optimización*. UN, Universidad Nacional de Colombia, Sede Manlizales, 2009.

[13] T. Weise, H. Skubch, M. Zapf y K. Geihs, «Global Optimization Algorithms and their Application to Distributed Systems,» 2008.

[14] J. López, *Optimización Multi-objetivo. Aplicaciones a problemas del mundo real*. Universidad Nacional de la Plata, Facultad de Informática, 2015.

[15] F. J. Martínez Estudillo, *Introducción a las matemáticas para la economía*. Desclée De Brouwer, S.A., 2005.

[16] M. Jamil y X.-S. Yang, «A literature survey of benchmark functions for global optimization problems,» *Journal of Mathematical Modelling and Numerical Optimisation*, vol. 4, págs. 150-194, 2013.

[17] A. Morales Carballo, *El papel que juega el contraejemplo en la construcción de las definiciones en matemáticas: el caso de la función convexa.* Tesis de Maestría, 2008.

[18] D. González Martínez y G. Blé González, «Entropáa topológica de funciones multimodales,» *Basic Sciences*, vol. 2, págs. 1-15, 2016.

[19] E. Baquela y A. Redchuk, *Optimización matemática con R, Volumen 1: Introducción al modelado y resolución de problemas*. Bubok Publishing S.L., 2013.

[20] R. A. Española, *Diccionario de la lengua española*. 23.ª ed., 2014.

[21] C. K. WNg y D. Li, «Test problem generator for unconstrained global optimization,» *Computers and Operations Research*, vol. 51, págs. 338-349, 2014.

[22] A. Al-Roomi, *Unconstrained Single-Objective Benchmark Functions Repository; 2015*. dirección: https://www.al-roomi.org/component/content/article?id=175:generalized-rosenbrock-s-valley-banana-or-2nd-de-jong-s-function.

[23] S. Surjanovic y B. Derek, *Virtual Library of Simulation Experiments: Test Functions and Datasets*, 2013. dirección: https://www.sfu.ca/~ssurjano/index.html/.

[24] A. Corana, M. Marchesi, C. Martini y S. Ridella, «Minimizing Multimodal Functions of Continuous Variables with Simulated Annealing Algorithms,» *ACM Transactions on Mathematical Software*, vol. 13, n.º 3, págs. 262-280, 1987.

[25] A. Layeb, «New hard benchmark functions for global optimization,» *arXiv preprint arXiv:2202.04606*, 2022.

[26] J. Brownlee, «A gentle introduction to generative adversarial networks (GANs),» *Machine Learning Mastery*, vol. 17, 2019.

[27] A. Shokraei Fard, D. C. Reutens y V. Vegh, «From CNNs to GANs for cross-modality medical image estimation,» *Computers in Biology and Medicine*, vol. 146, 2022.

[28] M. Mirza y S. Osindero, «Conditional Generative Adversarial Nets,» *arXiv preprint arXiv:1411.1784*, págs. 1-7, 2014.

[29] E. Mas Candela, *Generación de logotipos mediante redes generativas antagónicas*. Tesis de Maestría , Universidad de Alicante, 2020.

[30] A. Radford, L. Metz y S. Chintala, «Unsupervised representation learning with deep convolutional generative adversarial networkss,» *arXiv preprint arXiv:1511.06434*, 2015.

[31] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta y A. A. Bharath, «Generative Adversarial Networks: An Overview,» *IEEE Signal Processing Magazine*, vol. 35, n.º 1, págs. 53-65, 2018.

[32] R. Rubinstein, «The Cross-Entropy Method for Combinatorial and Continuous Optimization,» *Methodology and Computing in Applied Probability*, vol. 1, págs. 127-190, 1999.

[33] J. Chen, Y. J. Xiong, X. Qiu, D. He, H. Yin y C. Xiao, «A cross entropy based approach to minimum propagation latency for controller placement in Software Defined Network,» *Computer Communications*, vol. 191, págs. 133-144, 2022.

[34] J. Edgar Amaya, E. Camargo, J. Aguilar y M. Tarazona, «A proposal for a cooperative cross-entropy method to tackle the unit commitment problem,» *Computers Industrial Engineering*, vol. 162, pág. 107 764, 2021, ISSN: 0360-8352.

[35] G. Developers, *Google*. dirección: https://developers.google.com/machine-learning/gan/loss.

[36] P. Escoriza Lusilla, *Equilibrio de Nash: teoría y aplicaciones*. 2018.

[37] R. Spick y J. A. Walke, «Realistic and Textured Terrain Generation using GANs,» *European Conference on Visual Media Production*, n.º 3, págs. 1-10, 2019.

[38] A. C. Rodriguez, T. Kacprzak, A. Lucchiand et al., «Fast Cosmic Web Simulations with Generative Adversarial Networks,» *Computational Astrophysics and Cosmology*, vol. 5, n.º 1, págs. 1-11, 2018.

[39] W. Gaggiotti, *Redes GANs como técnica de data augmentation para el reconocimiento de lengua de señas*. Universidad Nacional de La Plata, 2021.

[40] C. Shorten y T. Khoshgoftaar, «A survey on Image Data Augmentation for Deep Learning,» *Big Data*, vol. 6, págs. 1-48, 2019.

[41] S. Redondo Hernández, *Deep Learning para la detección de patologías de cáncer de piel y generación de imágenes de tejidos humanos*. Universidad Politécnica de Cataluña, 2020.

[42] P. Foundation, *PyTorch*. dirección: https://pytorch.org/docs/stable/index.html.

[43] S. Cerezo Sánchez, *Herramientas modernas en Redes Neuronales: la librería PyTorch*. Universidad Autónoma de Madrid, 2020.

[44] I. Belzunegui Gabilondo, *Diseño e implementación de una librería de neuro-evolución para deep-learning con Pytorch*. Universidad del País Vasco, 2020.

[45] C. Bonilla Carrión, *REDES CONVOLUCIONALES*. Universidad de Sevilla, 2020.

[46] G. Cruz, *Generación de señales sintéticas de eventos sismo-volcánicos del volcán Cotopaxi a través de un modelo de red neuronal adversario generativo condicional*. ESPEC, Universidad de las Fuerzas Armadas, 2022.

[47] I. Goodfellow, J. Pouget-Abadie, M. Mirza et al., «Generative adversarial networks,» *Communications of the ACM 63*, n.º 11, págs. 139-144, 2020.

[48] G. Dawson, «Interpretación fácil de la bioestadística: la conexión entre la evidencia y las decisiones médicas.,» *Elsevier Health Sciences*, 2009.

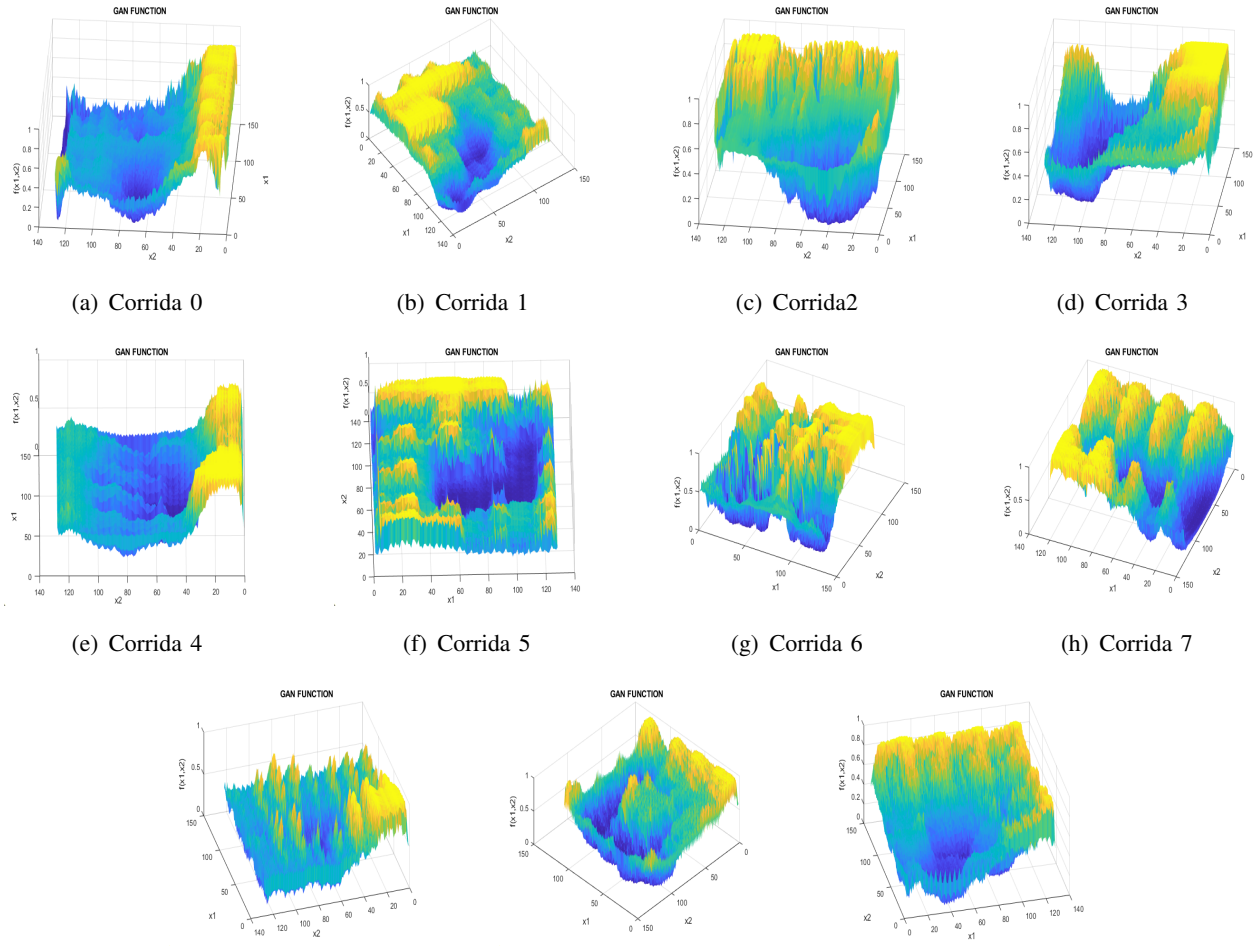
[49] E. Taucher, *Bioestadística*. Editorial Universitaria, 1997.

[50] M. Fuentes y K. Yohannessen, «Cómo leer y generar publicaciones científicas: exploración gráfica de datos cuantitativos: la importancia de mirar la información,» *Neumol. pediátr.(En línea)*, págs. 194-199, 2019.

Apéndice A
Anexo: Paisajes

A continuación se muestran los resultados obtenidos por la red y para cada una de las bases de datos, los paisajes se encuentran graficados en 3 dimensiones para tener una mejor idea visual de su estructura.
La corrida 0 hace referencia a la parte de la metodología de experimentación, donde se seleccionan los parámetros para la red.

1.1. Base 1



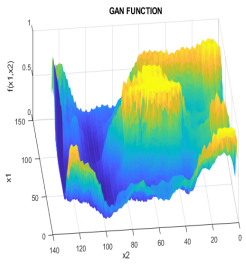
(a) Corrida 0 (b) Corrida 1 (c) Corrida2 (d) Corrida 3

(e) Corrida 4 (f) Corrida 5 (g) Corrida 6 (h) Corrida 7

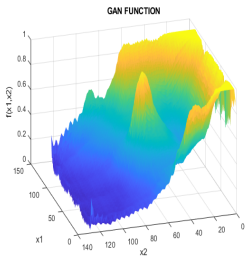
(i) Corrida 8 (j) Corrida 9 (k) Corrida 10

Fig. 57. Paisajes Tridimensionales Base 1

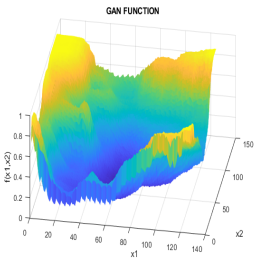
1.2. Base 2



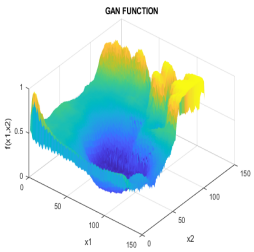
(a) Corrida 0



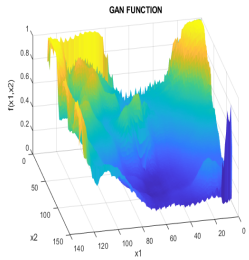
(b) Corrida 1



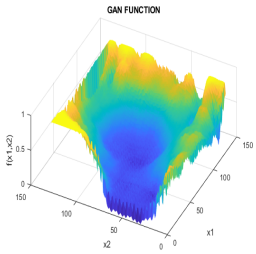
(c) Corrida 2



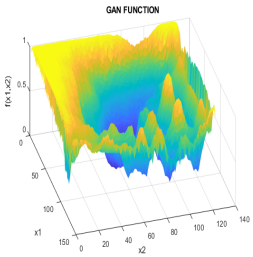
(d) Corrida 3



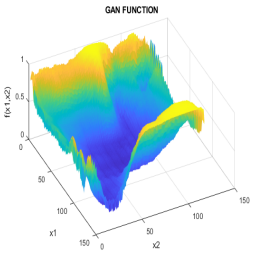
(e) Corrida 4



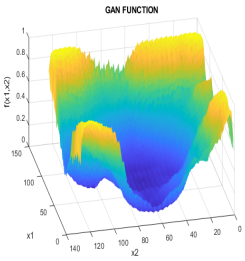
(f) Corrida 5



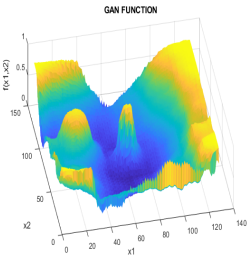
(g) Corrida 6



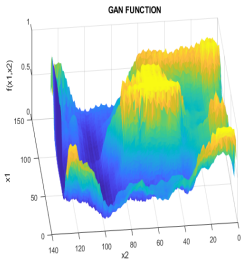
(h) Corrida 7



(i) Corrida 8



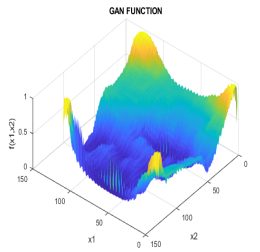
(j) Corrida 9



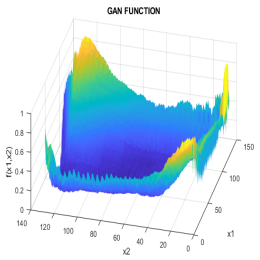
(k) Corrida 10

Fig. 58. Paisajes Tridimensionales Base 2

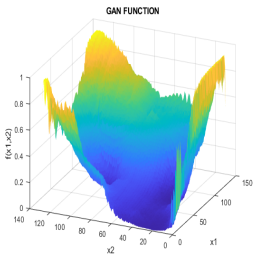
1.3. Base 3



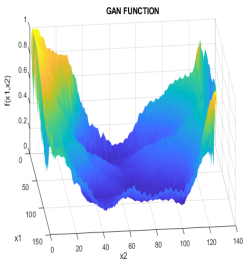
(a) Corrida 0



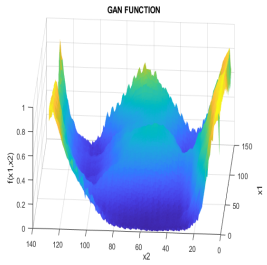
(b) Corrida 1



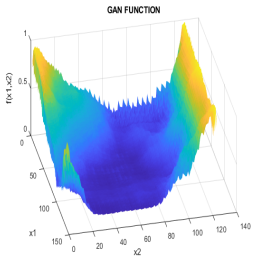
(c) Corrida 2



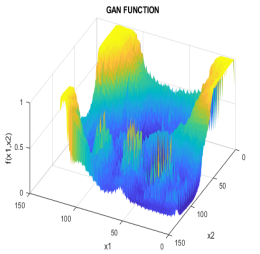
(d) Corrida 3



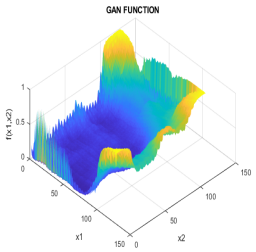
(e) Corrida 4



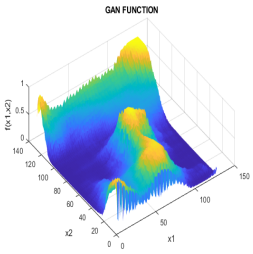
(f) Corrida 5



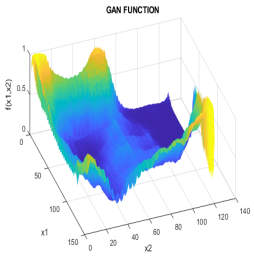
(g) Corrida 6



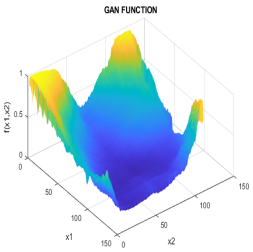
(h) Corrida 7



(i) Corrida 8



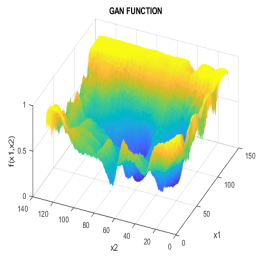
(j) Corrida 9



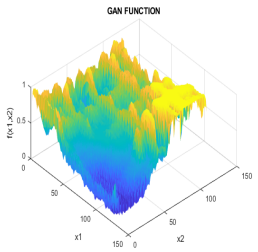
(k) Corrida 10

Fig. 59. Paisajes Tridimensionales Base 3

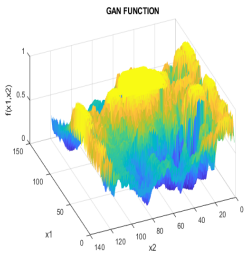
1.4. Base 4



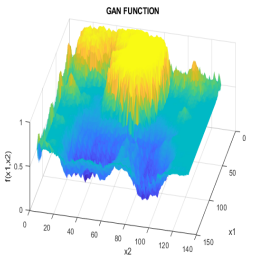
(a) Corrida 0



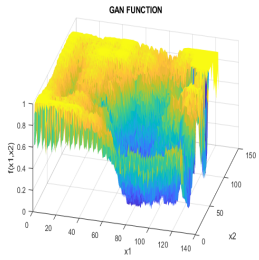
(b) Corrida 1



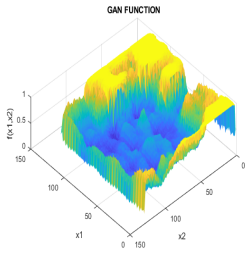
(c) Corrida 2



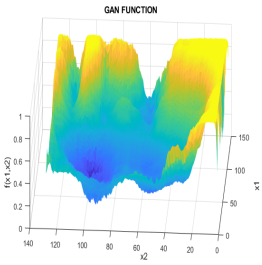
(d) Corrida 3



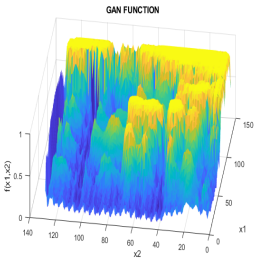
(e) Corrida 4



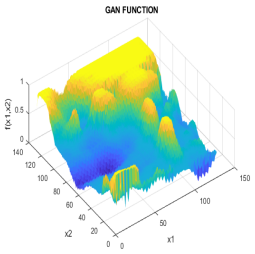
(f) Corrida 5



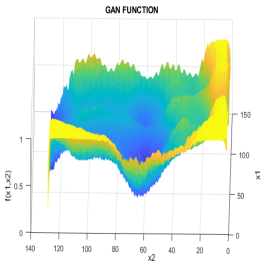
(g) Corrida 6



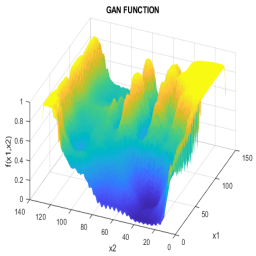
(h) Corrida 7



(i) Corrida 8



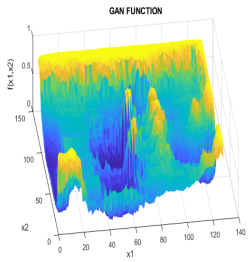
(j) Corrida 9



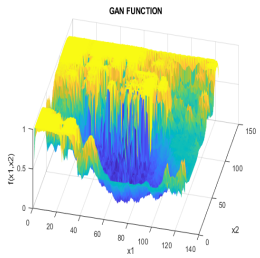
(k) Corrida 10

Fig. 60. Paisajes Tridimensionales Base 4

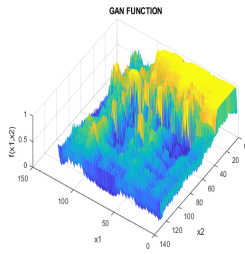
1.5. Base 5



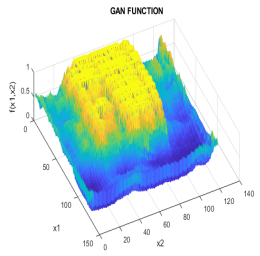
(a) Corrida 0



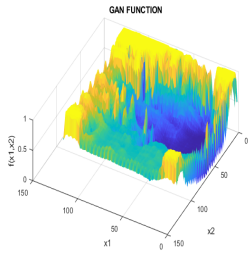
(b) Corrida 1



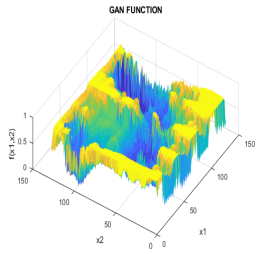
(c) Corrida 2



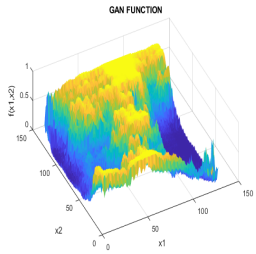
(d) Corrida 3



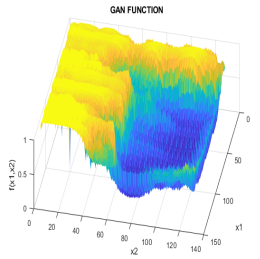
(e) Corrida 4



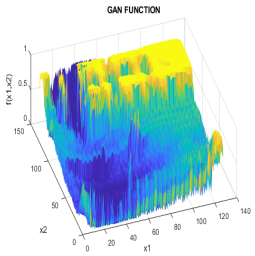
(f) Corrida 5



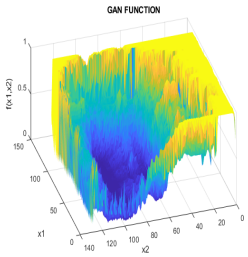
(g) Corrida 6



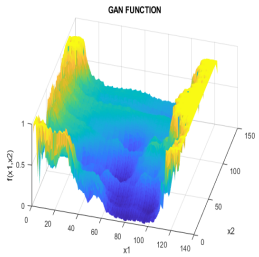
(h) Corrida 7



(i) Corrida 8



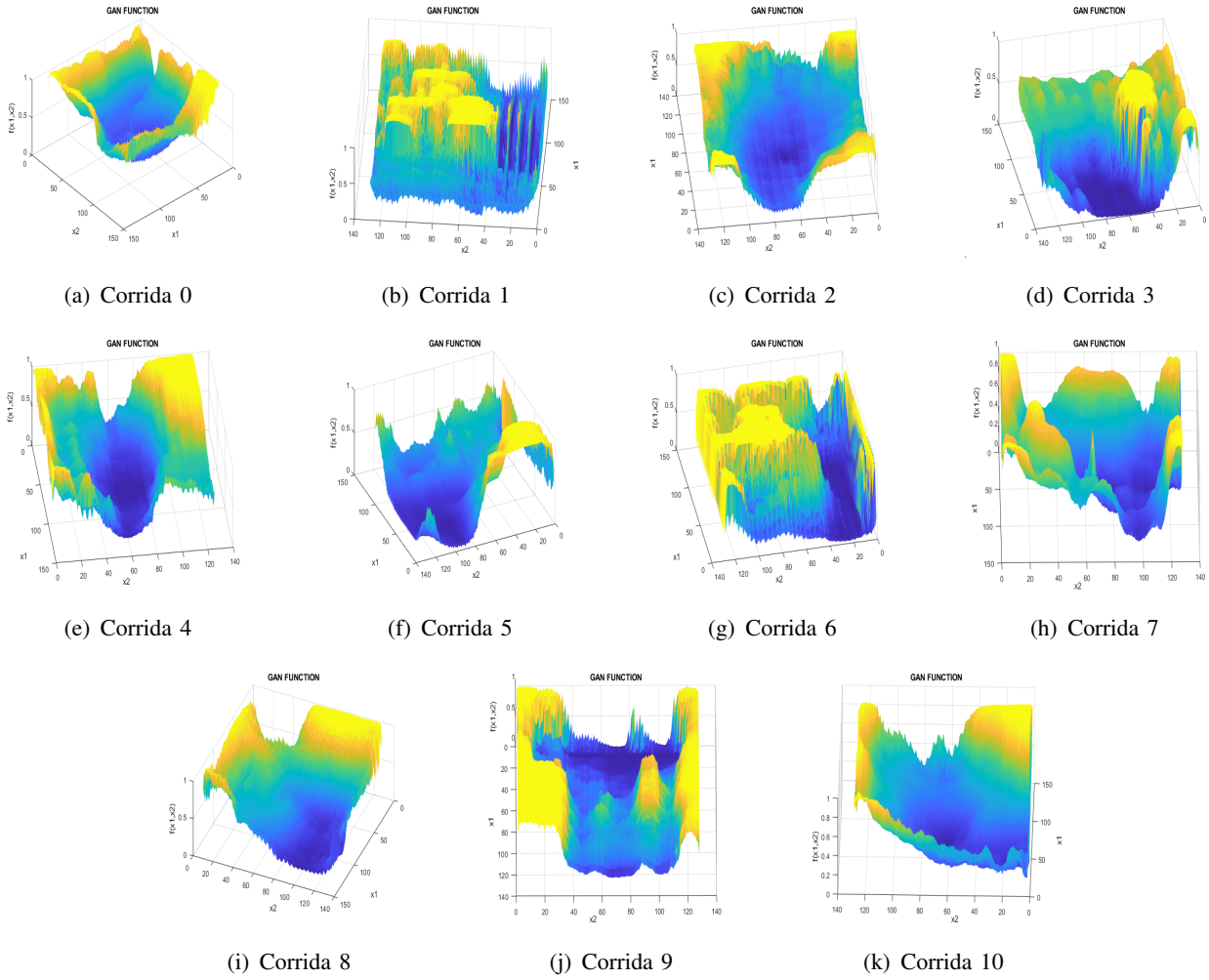
(j) Corrida 9



(k) Corrida 10

Fig. 61. Paisajes Tridimensionales Base 5

1.6. Base 6



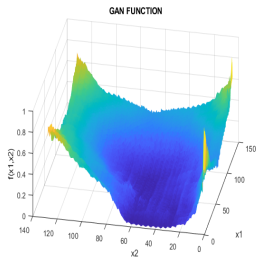
(a) Corrida 0 (b) Corrida 1 (c) Corrida 2 (d) Corrida 3

(e) Corrida 4 (f) Corrida 5 (g) Corrida 6 (h) Corrida 7

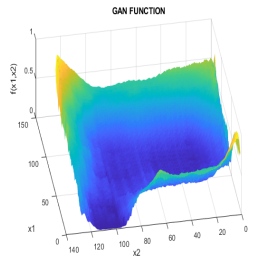
(i) Corrida 8 (j) Corrida 9 (k) Corrida 10

Fig. 62. Paisajes Tridimensionales Base 6

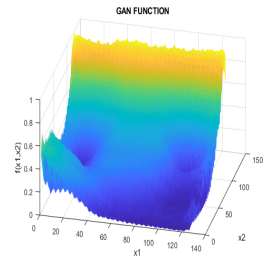
1.7. Base 7



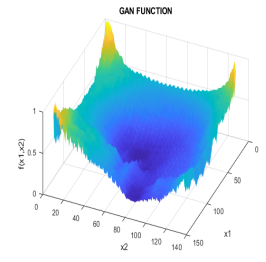
(a) Corrida 0



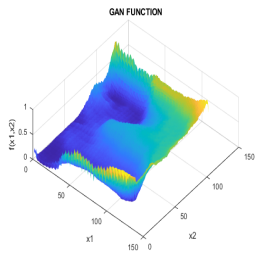
(b) Corrida 1



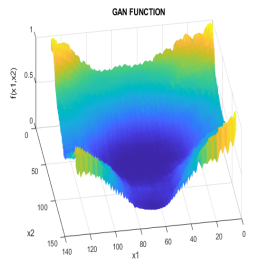
(c) Corrida 2



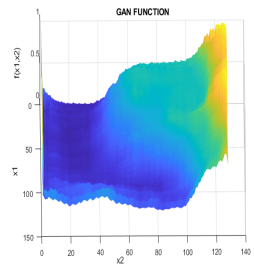
(d) Corrida 3



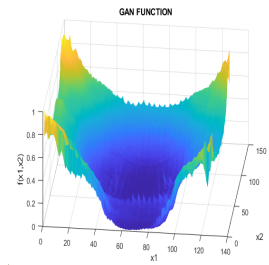
(e) Corrida 4



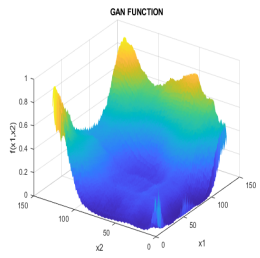
(f) Corrida 5



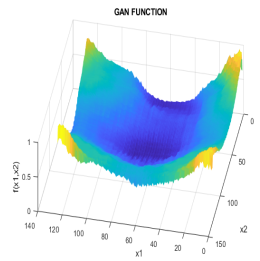
(g) Corrida 6



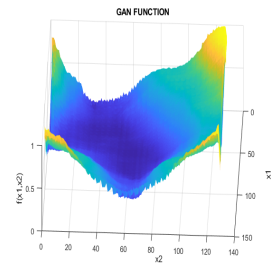
(h) Corrida 7



(i) Corrida 8



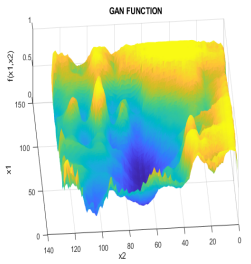
(j) Corrida 9



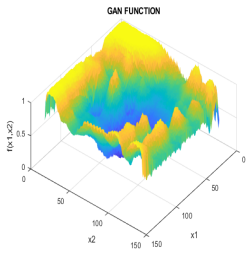
(k) Corrida 10

Fig. 63. Paisajes Tridimensionales Base 7

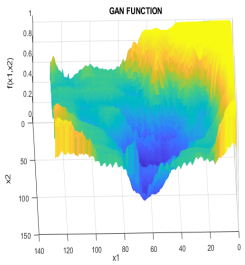
1.8. Base 8



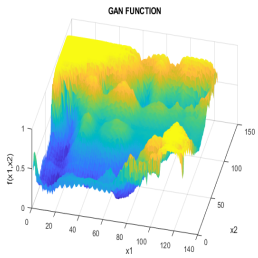
(a) Corrida 0



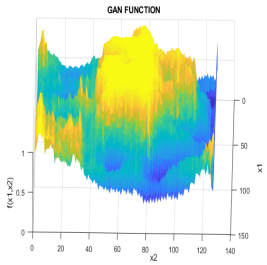
(b) Corrida 1



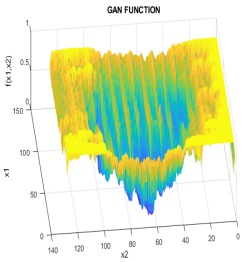
(c) Corrida 2



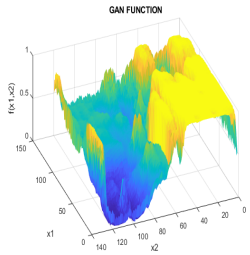
(d) Corrida 3



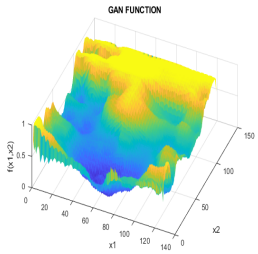
(e) Corrida 4



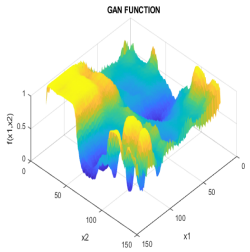
(f) Corrida 5



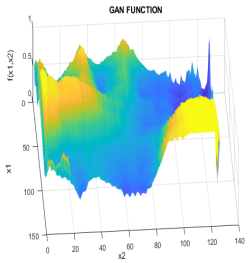
(g) Corrida 6



(h) Corrida 7



(i) Corrida 8



(j) Corrida 9

Fig. 64. Paisajes Tridimensionales Base 8