Mecanismo de clasificación de paisajes de optimización basado en muestreo multiescala y aprendizaje automático

Angie Patricia Rodríguez Hernández

Universidad Distrital Francisco José de Caldas

Facultad de Ingeniería

Proyecto Curricular de Maestría en Ciencias de la Información y las Comunicaciones

Bogotá D.C.

2024

Mecanismo de clasificación de paisajes de optimización basado en muestreo multiescala y aprendizaje automático

Angie Patricia Rodríguez Hernández

Trabajo de grado para optar al título de:
Maestría en Ciencias de la Información y las Comunicaciones

Director:
Doctor Ing. Miguel Alberto Melgarejo Rey

Grupo de Investigación:
Laboratorio de Automática e Inteligencia Computacional.

Universidad Distrital Francisco José de Caldas
Facultad de Ingeniería
Proyecto Curricular de Maestría en Ciencias de la Información y las Comunicaciones
Bogotá D.C.
2024

Resumen

Este trabajo propone un enfoque para clasificar la modalidad en paisajes de optimización, combinando un muestreo a múltiples escalas con técnicas de aprendizaje automático.

En primer lugar, se seleccionó un conjunto de funciones de optimización [1], las cuales fueron etiquetadas de acuerdo con la definición propuesta por Kanemitsu et al. [2]. Para reducir el sesgo en la muestra, se desarrolló un algoritmo de muestreo a múltiples escalas que aprovecha el comportamiento de una caminata guiada por una ley de potencias, complementada con mecanismos de explotación a escala fina, para explorar de manera diferente los paisajes de optimización.

Estas muestras forman una imagen que se utiliza como entrada para una red neuronal convolucional, encargada de clasificar la modalidad del paisaje. Los resultados experimentales muestran que el enfoque propuesto ofrece un rendimiento competitivo en la clasificación de paisajes no observados previamente. Además, los resultados sugieren que la estrategia multiescala proporciona información más confiable en comparación con el muestreo aleatorio, que es la técnica estándar utilizada en el análisis de paisajes de optimización.

Contenido

Lista de figuras **9**

Lista de tablas **10**

1. INTRODUCCIÓN **12**
1.1. Planteamiento del problema 12
1.1.1. Formulación del Problema 16
1.1.2. Sistematización del Problema 16
1.2. Objetivos 16
1.2.1. Objetivo general 16
1.2.2. Objetivos específicos 16
1.2.3. Solución propuesta 17
1.2.4. Contenido del libro 17

2. MARCO REFERENCIAL **19**
2.1. MARCO CONCEPTUAL 19
2.1.1. Problema de optimización 19
2.1.2. Paisaje de optimización 19
2.1.3. Muestreo imparcial de un problema de optimización 20
2.1.4. Muestreo aleatorio 20
2.1.5. Muestreo uniforme 21
2.1.6. Propiedades (de alto nivel) de un paisaje de optimización 21
2.1.7. Características (o propiedades de bajo nivel) de un paisaje de optimización 21
2.2. MARCO TEÓRICO 22
2.2.1. Fitness Landscape Analisys (FLA) 22
2.2.2. Exploratory Landscape Analisys (ELA) 23
2.2.3. Propiedad de modalidad en los paisajes de optimización 26
2.2.4. Caminatas aleatorias y caminatas que siguen una ley de potencias . . 28
2.2.5. Algoritmos de búsqueda local 30
2.2.5.1. Comparación de Algoritmos de Búsqueda Local 31

2.2.6. Redes Neuronales Convolucionales 32
2.2.6.1. Función de Activación 34
2.2.6.2. Optimizador 35
2.2.6.3. Transfer Learning 36
2.2.7. Índices de rendimiento 39
2.2.7.1. Matriz de Confusión 39
2.2.7.2. Algunos índices de rendimiento en clasificación 40
2.2.7.3. Curva ROC y AUC 41
2.2.7.4. Intervalo de Confianza con Prueba t-Student 42
2.3. Validación Cruzada 43
2.4. Cuantificación de la Incertidumbre entre Imágenes 43
2.4.1. Error Cuadrático Medio (MSE) 43
2.4.2. Correlación de Pearson 44
2.4.3. Información Mutua Ajustada (AMI) 44
2.4.4. Índice de Similitud Estructural (SSIM) 44
2.5. Trabajos relacionados 45

3. METODOLOGÍA DE CLASIFICACIÓN PROPUESTA 48
3.1. Construcción de la base de datos de problemas de optimización 48
3.2. Algoritmo de muestreo exploratorio multiescala (Multiscale Scouting Samplig Algorithm - MUSSA) 52
3.2.1. Construcción de la base de datos de imágenes a partir del muestreo multiescala 52
3.2.2. Caminata de Lévy 55
3.2.3. Algoritmos de búsqueda local 56
3.2.3.1. Algoritmo de búsqueda local de Powell: 57
3.2.3.2. Algoritmo de Búsqueda de Nelder-Mead (NMSA): 58
3.3. Clasificación de la base de datos de imágenes con Redes Neuronales Convolucionales (CNN) 58

4. RESULTADOS 61
4.1. Marco experimental 61
4.1.1. Primer experimento 61
4.1.1.1. Funciones de prueba 62
4.1.2. Segundo experimento 62
4.1.3. Tercer experimento 62
4.1.3.1. Tercer experimento: base de datos de imágenes de muestras aleatorias uniformes (URSA) 62
4.2. Resultados 64
4.2.1. Resultados primer y segundo experimento 64

4.2.2. Resultados tercer experimento 69
4.3. Discusión 70

Conclusiones y trabajos futuros **72**
4.4. Aportes originales 73
4.5. Trabajo futuro 73

A. Clasificación de funciones de optimización generadas por aprendizaje automático **75**
A.1. Funciones generadas por una red generativa adversaria profunda (DCGAN) . 75
A.2. Funciones generadas por autocodificadores variacionales (VAE) 77

Índice de figuras

1-1. Función Griewank. 14
1-2. Modelo Conceptual. 17

2-1. Diagrama de capas de una CNN. Tomada de [3]. 34
2-2. Estructura general de la función de activación (imagen de elaboración propia). 35
2-3. Ejemplos de curvas ROC (imagen de elaboración propia). 42

3-1. Ejemplo de mínimos en paisajes de optimización. Elaboración propia. 48
3-2. Funcionamiento de MUSSA. 52
3-3. Modelo de transferencia de Aprendizaje con MobileNet V2. 60

4-1. Comparación de validación utilizando PSA y NMSA y solo NMSA 65
4-2. Comparación pos validación utilizando PSA y NMSA y solo NMSA. 66
4-3. Curvas ROC del primer y segundo experimento para diferentes pasos en las funciones de prueba. 68
4-4. Comparación de las curvas ROC entre MUSSA y URSA 70

Índice de tablas

2-1. Comparación de algunos algoritmos de optimización de búsqueda local [4, 5]. 32
2-2. Arquitectura de la red basada en MobileNet. 38
2-3. Arquitectura de la red basada en MobileNet V2. 39
2-4. Índices de rendimiento en clasificación [6, 7, 8, 9, 10]. 41

3-1. Caracterización de las funciones. 51
3-2. Obtención de imágenes con MUSSA de algunos problemas de optimización. . 54
3-3. Parámetros de MUSSA (parámetros de los paquetes del lenguaje de programación Python: [11] y [12]). 55

4-1. Construcción de la base de datos para el entrenamiento, validación y prueba de los modelos. 61
4-2. Caracterización de las funciones de prueba. 63
4-3. Contribución de los algoritmos de búsqueda local al muestreo. 64
4-4. Configuración del muestreo aleatorio uniforme para cada representación de pasos de Lévy. 64
4-5. Intervalos de confianza para la exactitud en validación y prueba. 66
4-6. Comparación integral de la prueba entre experimentos. 67
4-7. Tiempo de ejecución aproximado en minutos para el primer experimento. . . 69
4-8. Tiempo de ejecución aproximado en minutos para el segundo experimento. . 69
4-9. Comparación de pruebas entre MUSSA y URSA. 69

A-1. Clasificación de las funciones generadas por una red neuronal generativa (GAN). 76
A-2. Clasificación de las funciones generadas por autocodificadores variacionales (VAE). 77

Agradecimientos

Quiero comenzar expresando mi más profundo agradecimiento a mi familia, cuyo apoyo constante ha sido el cimiento sobre el que he podido construir este proyecto. Ellos representan la tierra fértil de la que surgió este árbol que, hoy en día, puede ofrecer sus frutos gracias a su dedicación y cuidado incondicional.

Mi sincero agradecimiento al profesor Miguel Melgarejo, cuya mente brillante y sus enriquecedoras charlas fueron fundamentales para dar vida a este trabajo. Sin su orientación, este proyecto no habría sido posible, ni habría encontrado el rumbo necesario para desarrollarse.

Finalmente, agradezco a todas las personas que me han acompañado a lo largo de este proceso, tanto en los momentos de éxito como en los de dificultad. El desarrollo de este trabajo no fue un camino secuencial, pero sin duda ha sido una experiencia enormemente gratificante.

Capítulo 1

INTRODUCCIÓN

1.1. Planteamiento del problema

Los problemas de optimización y su resolución a través de algoritmos se encuentran en el centro de los procesos de toma de decisiones en diversos campos, su objetivo es identificar la solución óptima entre una multitud de alternativas, mientras se adhieren a los objetivos y restricciones específicos del problema [13]. Sin embargo, el teorema No Free Lunch (NFL) nos recuerda que el rendimiento de cualquier algoritmo dentro de una clase específica de problemas inevitablemente compensa su eficacia en el espectro más amplio de los demás problemas existentes [14]. Por lo tanto, la búsqueda de un algoritmo de optimización universalmente eficaz sigue siendo un reto fundamental [15].

El problema de selección de algoritmos de optimización contiene dos partes. La primera es la identificación o caracterización del problema de optimización, y la segunda es la búsqueda del algoritmo que mejor se adapte a dicho problema [16]. La identificación del problema de optimización se ha tratado de realizar de diversas maneras, incluyendo el análisis de las características matemáticas del mismo, como la convexidad, la diferenciabilidad y las restricciones presentes, sin embargo, esto implica un análisis profundo en la complejidad matemática del problema [17]. También se han utilizado técnicas como la clasificación de problemas según su tamaño, número de variables o la estructura de las funciones objetivo y de restricción. Otra estrategia usada es identificar patrones en problemas similares previamente resueltos, sin embargo, estas técnicas son limitadas cuando se enfrentan a problemas más complejos o de alta dimensionalidad, donde las características no son fácilmente discernibles o clasificables. Además, su efectividad puede depender de la calidad y cantidad de problemas previos almacenados, lo que plantea desafíos en términos de generalización y escalabilidad para nuevos tipos de problemas [18, 19].

Para entender los problemas de optimización es fundamental el concepto de paisaje de optimización, el cual consiste en la representación gráfica de todas las soluciones concebibles para

un problema dado [20, 21]. Numerosos métodos matemáticos, como el descenso de gradiente y los métodos cuasi-newtonianos, han intentado navegar por estos paisajes en busca de soluciones óptimas [22, 13]. Sin embargo, independientemente del método empleado, persisten las limitaciones inherentes que plantea el NFL.

Para mitigar las implicaciones del teorema NFL y con el fin de dar resolución al problema de selección de algoritmos de optimización, se introdujeron: el Análisis del Paisaje de Aptitud (u optimización - FLA) y el Análisis Exploratorio de Paisajes de optimización (ELA) [23, 1]. FLA y ELA proponen el uso de técnicas de muestreo de datos imparciales para revelar las características o propiedades subyacentes de los paisajes de optimización, ayudando así en la selección de algoritmos adaptados a las características específicas del problema [23, 24]. Sin embargo, lograr un muestreo verdaderamente imparcial sigue siendo un reto importante. Se ha sugerido la caminata aleatoria como solución, pero está contiene algunos problemas como del sesgo del tamaño del paso y el número de muestras necesarias para capturar la información [25, 26, 27, 21]. En [28] desarrollaron un método para evaluar la fiabilidad de las técnicas de análisis del paisaje, arrojando luz sobre la volatilidad de determinadas características y el impacto de la "maldición de la modalidad" en los requisitos del tamaño de la muestra. Este fenómeno sugiere que, a medida que aumenta la complejidad del paisaje a analizar, se necesitan muestras de mayor tamaño.

Además del muestreo, la selección de características plantea un problema importante en el ELA, ya que su eficacia depende en gran medida de la representación de las mismas. Si la representación elegida no capta adecuadamente la estructura y la complejidad del espacio de características del problema, los resultados de ELA pueden estar sesgados o incompletos. Adicionalmente, la interpretación de los resultados de ELA puede ser un reto [29, 21, 28]. Existe una falta de consenso en cuanto a las características más relevantes y las necesidades de cálculo de nuevas características para cada paisaje examinado o algoritmo considerado. Este cómputo recurrente de características hace que la aplicación de ELA sea computacionalmente costosa [28]. Asimismo, la visualización de los paisajes de características es compleja y exige un análisis meticuloso para obtener información significativa sobre la importancia de las características y su interpretación [29]. Asimismo, es importante mencionar que la extracción de propiedades de los paisajes de optimización usando FLA puede ser un proceso bastante complejo ya que se requiere conocer las propiedades matemáticas del problema [17].

Un análisis más complejo implica considerar las escalas integradas en los paisajes de optimización. Cuando una función presenta múltiples máximos y mínimos locales, estos valores pueden darse en varias escalas o rangos dentro del dominio de la función. En consecuencia, se pueden obtener diferentes perspectivas del problema en función de la escala de observación [30].

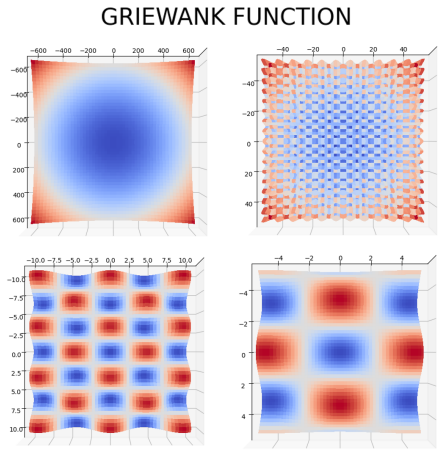


Figura **1-1**: Función Griewank.

Por ejemplo, consideremos la función Griewank representada en la Fig. **1-1**, a escala gruesa, la función puede parecer que muestra un comportamiento convexo. Sin embargo, a medida que la escala se hace más fina, la oscilación de la función aumenta, lo que hace que la función deje de ser convexa. En consecuencia, cuando se muestrea el paisaje a través de técnicas como las caminatas aleatorias para discernir sus propiedades, las muestras obtenidas pueden estar influenciadas por la información local y pueden no capturar los comportamientos que se producen a través de escalas gruesas y finas del paisaje de optimización [25, 28].

El muestreo en línea, que deriva muestras de las soluciones encontradas por un algoritmo durante la búsqueda, y ha surgido como una alternativa popular al muestreo aleatorio para el análisis del paisaje [31, 32]. Aunque estos métodos suelen arrojar resultados favorables, incurren en costos computacionales significativos y tienden a adaptarse y resolver el problema explotando el paisaje, desviándose del objetivo principal del análisis de paisajes de optimización. Sin embargo, la utilización adaptativa de algoritmos de búsqueda local de bajo coste computacional, que se dedican a la explotación adaptativa del paisaje a escala fina sin resolución explícita, presenta una intrigante vía de exploración.

El problema de selección de algoritmos ha sido abordado recientemente mediante el uso de sistemas de recomendación para metaheurísticas, proporcionando resultados interesantes [33]. Este enfoque utiliza redes neuronales profundas que correlacionan el rendimiento de varias metaheurísticas con las posibles expresiones algebraicas detrás de los problemas de optimización. No obstante, además de ser extremadamente costoso computacionalmente, este enfoque carece de valor analítico, ya que no arroja luz sobre la conexión entre las características del problema de optimización y el algoritmo correspondiente para resolver dicho problema.

En este estudio, seguimos un enfoque diferente, tratando el problema de selección de algoritmos como una tarea analítica que requiere la clasificación del problema de optimización para establecer evidencia útil antes de seleccionar un algoritmo. Para abordar esta clasificación, proponemos un enfoque multiescala que aborda el problema del muestreo imparcial, capitalizando una caminata de ley de potencias combinada con mecanismos de explotación para navegar por el paisaje de optimización en varias escalas y construir una imagen representativa que contenga la mayoría de sus características relevantes. Inspirados por la efectividad de los sistemas de ley de potencias en el modelado de comportamientos dinámicos, el objetivo es desvelar propiedades ocultas del paisaje explorando dinámicas a gran escala complementadas por algoritmos de búsqueda local a escala fina [34].

Estudios recientes sobre el Análisis Exploratorio de Paisajes (ELA) han propuesto el uso de la extracción de características basada en imágenes, que ofrece un enfoque competitivo al tiempo que reduce los costos computacionales en comparación con los algoritmos que realizan el cálculo de características específicas del problema [35]. Al emplear la clasificación basada en imágenes en lugar de centrarse en la adquisición de vectores de características para discernir las propiedades de los paisajes de optimización de caja negra, se puede lograr un ahorro significativo en términos de selección de características y gastos computacionales.

Las redes neuronales convolucionales (CNN) se han convertido en la piedra angular de la clasificación de imágenes y son conocidas por su gran precisión, eficacia y robustez. Estas redes tienen la capacidad de aprender automáticamente características relevantes a partir de datos de entrenamiento, lo que las hace ideales para tareas de clasificación de imágenes [36, 37, 38]. El campo de la visión por ordenador con redes neuronales está experimentando un rápido crecimiento impulsado por los avances tecnológicos, el aumento de la demanda en diversos sectores y la mejora de la accesibilidad a herramientas y recursos [39].

Por lo tanto, en este trabajo se utiliza un enfoque de clasificación basado en redes neuronales convolucionales (CNNs) para aprender y detectar características extraídas de las muestras. Este enfoque se centra en la construcción de imágenes mediante la exploración y explotación del paisaje de optimización en múltiples escalas, con el objetivo principal de clasificar los paisajes de optimización según su modalidad (unimodal o multimodal) como una herramienta analítica para ayudar en el problema de selección de algoritmos en optimización.

La contribución de este trabajo es doble: i) Se propone una estrategia de navegación para la caracterización de problemas de optimización que utiliza tanto exploración como explotación. Esta estrategia proporciona una representación del paisaje que contiene información en múltiples escalas, lo que reduciría el sesgo de muestreo; ii) Se entrena un sistema de aprendizaje automático de manera efectiva utilizando esta representación para clasificar la modalidad de problemas de optimización.

1.1.1. Formulación del Problema

De acuerdo con el planteamiento del problema, surge la siguiente pregunta de investigación: **¿Cómo sería un mecanismo de clasificación basado en el análisis exploratorio de los paisajes de optimización usando un algoritmo de muestreo en múltiples escalas y aprendizaje automático?**

1.1.2. Sistematización del Problema

La pregunta de investigación se puede descomponer con respecto a las siguientes preguntas:

- ¿Cómo seleccionar una base de datos de funciones de optimización?
- ¿Cómo etiquetar las funciones de optimización empleando una propiedad teórica?
- ¿Cómo realizar el muestreo de los paisajes de optimización de las funciones seleccionadas en múltiples escalas para generar una base de datos de imágenes?
- ¿Cómo entrenar una red neuronal que a partir de una base de datos de imágenes, clasifique las funciones de optimización de acuerdo con la propiedad teórica seleccionada?
- ¿Cómo evaluar si el mecanismo de clasificación propuesto está clasificando adecuadamente los paisajes de optimización?

1.2. Objetivos

1.2.1. Objetivo general

Diseñar un mecanismo de clasificación de paisajes de optimización conectando un algoritmo de muestreo en múltiples escalas y redes neuronales convolucionales.

1.2.2. Objetivos específicos

- Seleccionar una base de datos de funciones de optimización.
- Etiquetar las funciones de optimización empleando una propiedad teórica.
- Proponer un algoritmo de muestreo de los paisajes de optimización de las funciones seleccionadas en múltiples escalas con el fin de generar una base de datos de imágenes.
- Entrenar una red neuronal convolucional a partir de la base de datos de imágenes construida que clasifique las funciones de optimización con respecto a la propiedad teórica seleccionada.
- Evaluar el desempeño del mecanismo de clasificación propuesto en la clasificación de los paisajes de optimización en un conjunto de problemas.

1.2.3. Solución propuesta

La Figura **1-2** ilustra el mecanismo de clasificación propuesto, el cual consta de dos componentes principales. El primer componente es el Algoritmo de Muestreo Exploratorio Multiescala, que integra una caminata que sigue una ley de potencia con algoritmos de búsqueda local. Este algoritmo tiene como objetivo utilizar los pasos proporcionados por la caminata como una ruta en una escala gruesa, complementada por algoritmos de búsqueda local en una escala más fina, en un esfuerzo por minimizar el sesgo en los datos muestreados. A partir de esta muestra, se construirá una imagen que servirá como espacio de características, según lo designado por la metodología ELA, facilitando la clasificación del paisaje.

El segundo componente implica la utilización de una Red Neuronal Convolucional para la clasificación de la base de imágenes brindada en el paso anterior, culminando con la etiquetación final de las funciones con respecto a la propiedad unimodal o multimodal. Se medió el porcentaje de exactitud (accuracy) de la clasificación del mecanismo junto con otras medidas, las cuales permitieron obtener un modelo de clasificación final.

1.2.4. Contenido del libro

Este trabajo se organiza de la siguiente manera: El capítulo dos presenta el marco de referencia que sustenta este estudio, proporcionando las definiciones clave e investigaciones previas relacionadas con el tema, y culmina con la explicación de la hipótesis detrás de la metodología de clasificación propuesta. El capítulo tres describe el desarrollo de la metodología de

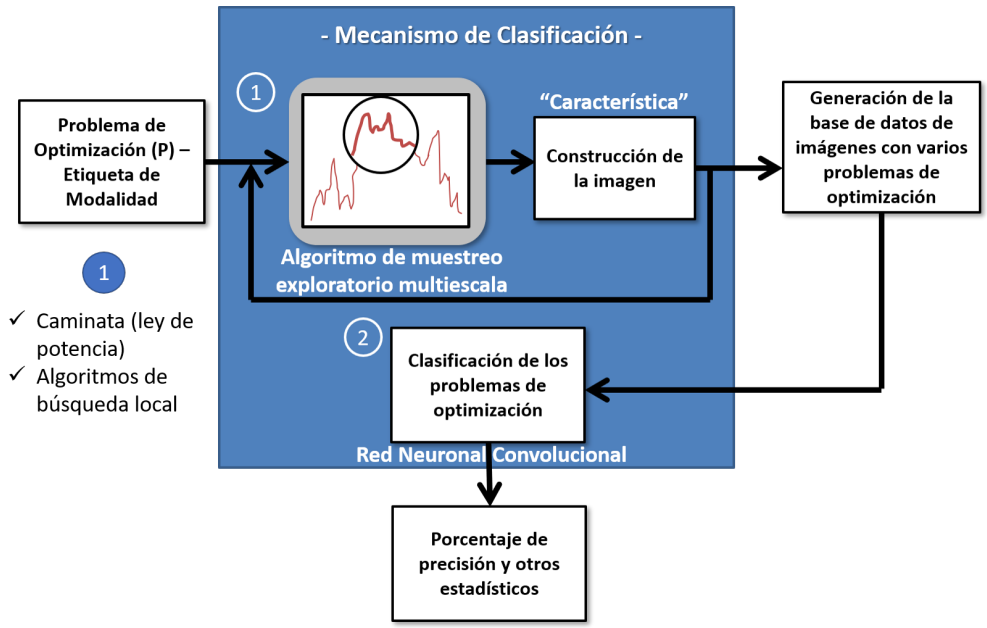


Figura **1-2**: Modelo Conceptual.

clasificación, la cual consta de tres fases: la primera corresponde a la selección y etiquetado de los paisajes de optimización utilizados en este estudio; la segunda, al planteamiento del algoritmo multiescala propuesto, denominado Algoritmo de Muestreo Exploratorio Multiescala (MUSSA); y la tercera, a la selección y configuración de la red neuronal convolucional. El capítulo cuatro detalla el marco experimental y los resultados obtenidos del método de clasificación propuesto. Finalmente, el capítulo cinco ofrece una discusión sobre los resultados, terminando con las conclusiones, los aportes originales y las posibles direcciones para trabajos futuros.

Capítulo 2

MARCO REFERENCIAL

2.1. MARCO CONCEPTUAL

2.1.1. Problema de optimización

Informalmente se llama así a un problema que busca minimizar o maximizar el valor de una función o de una variable con el fin de encontrar la mejor solución de todas las soluciones factibles [40].

Formalmente, según [41], un problema de optimización (maximización o minimización) puede definirse mediante un espacio de búsqueda S y una función objetivo $f_{obj} : S \mapsto \mathbb{R}$. Un elemento $x \in S$ se llamará una solución y una solución óptima es una solución $x^* \in S$ tal que $\forall x \in S, f_{obj}(x) \leq f_{obj}(x^*)$.

El propósito de un algoritmo de optimización es calcular soluciones óptimas a partir de S. Algunas técnicas se centran en técnicas locales de búsqueda que exploran S utilizando una función de vecindario N para moverse a través de S y una función de aptitud (fitness) f para evaluar las soluciones (que podrían ser diferentes de f_{obj}). La dinámica de la búsqueda local depende principalmente de las propiedades estructurales del triplete (S, N, f) denotado como **paisaje de optimización** o paisaje de aptitud (fitness) y más precisamente la compatibilidad entre las relaciones N y f [41].

2.1.2. Paisaje de optimización

Un paisaje de optimización (o aptitud) es una representación metafórica utilizada en la optimización y en la computación evolutiva para describir la relación entre las posibles soluciones a un problema y sus correspondientes valores de aptitud. Formalmente, un fitness landscape se define como una función matemática $f : X \mapsto S$ donde X es el conjunto de todas las posibles soluciones (o configuraciones) y S representa los valores de aptitud (u objetivos)

asociados a cada solución [42].

Este concepto se utiliza ampliamente para analizar la complejidad de los problemas de optimización, estudiando características como el número de óptimos locales, la rugosidad del paisaje, y cómo estos factores afectan la capacidad de los algoritmos de búsqueda para encontrar óptimos globales [17].

2.1.3. Muestreo imparcial de un problema de optimización

El muestreo imparcial en problemas de optimización es un proceso que busca seleccionar un subconjunto de puntos o muestras del espacio de búsqueda que represente con precisión las características del paisaje de optimización (como su topología y variabilidad). Este enfoque permite una evaluación más justa y generalizable de algoritmos de optimización al asegurar que no se favorezcan ciertas regiones del espacio de búsqueda en detrimento de otras [25].
Para realizar un muestreo imparcial en paisajes de optimización, es necesario considerar los siguientes principios [43]:

1. Cobertura de características topológicas: La muestra debe incluir puntos que reflejen la diversidad del paisaje, como áreas de máxima, mínima y puntos de silla. Esto asegura que se capture una visión completa de la estructura y la dificultad del espacio de búsqueda.

2. Uniformidad en la distribución: El muestreo debe ser uniforme o estratificado para evitar la concentración en una sola área del paisaje, ya que esto podría sesgar los resultados del algoritmo hacia comportamientos específicos de esa región.

3. Adaptabilidad al problema: En función del tipo de optimización, las técnicas de muestreo pueden incluir enfoques como el análisis exploratorio del paisaje (ELA) o el análisis del paisaje fitness (FLA), que extraen características relevantes del espacio de búsqueda (p. ej., el número de óptimos locales, gradientes y forma de la función objetivo) y guían el muestreo en base a estas.

4. Reducción de redundancia: La selección de puntos debe evitar la inclusión de instancias muy similares o redundantes, para maximizar la información obtenida sobre la variabilidad del paisaje y minimizar el sesgo por puntos duplicados.

2.1.4. Muestreo aleatorio

El muestreo aleatorio es un proceso de selección de una muestra a partir de una población en el que cada elemento tiene una probabilidad conocida y no cero de ser seleccionado. Formalmente, si N es el tamaño de la población y n es el tamaño de la muestra, el muestreo aleatorio puede ser descrito como:

$$P(X_i) = \frac{1}{N} \quad \forall i \in \{1, 2, \ldots, N\} \tag{2-1}$$

donde $P(X_i)$ es la probabilidad de seleccionar el elemento X_i de la población. En el caso de muestreo aleatorio simple, la selección de elementos es independiente, lo que implica que la probabilidad de seleccionar cualquier subconjunto de elementos es proporcional a su tamaño [44, 45].

2.1.5. Muestreo uniforme

El muestreo uniforme es un tipo específico de muestreo en el que todos los elementos de la población tienen la misma probabilidad de ser seleccionados. Este concepto es importante en estadísticas y teoría de probabilidades. Matemáticamente, se puede expresar como:

$$P(X_i) = \frac{1}{N} \quad \forall i \in \{1, 2, \ldots, N\} \tag{2-2}$$

En este contexto, si se selecciona una muestra de n elementos, la probabilidad de seleccionar cualquier conjunto específico de elementos es igualmente probable, y no hay diferenciación en la selección [45].

2.1.6. Propiedades (de alto nivel) de un paisaje de optimización

Las propiedades de alto nivel de un paisaje de optimización se refieren a los atributos del paisaje formado por la función objetivo en un problema de optimización. Estas propiedades son utilizadas en el análisis de paisajes fitness (FLA) y juegan un papel crucial en determinar el comportamiento y la eficiencia de los algoritmos de optimización en la búsqueda de la solución óptima ya que ayudan a entender la estructura del problema, algunas propiedades son: Modalidad, cuencas, valles, separabilidad, dimensión, etc [46].

2.1.7. Características (o propiedades de bajo nivel) de un paisaje de optimización

Las características o propiedades de bajo nivel de un paisaje de optimización son métricas cuantitativas y cualitativas que describen las propiedades fundamentales de un espacio de búsqueda en problemas de optimización. Estas características se utilizan en análisis exploratorio de paisajes de optimización para comprender la dificultad del problema y predecir el rendimiento de diferentes algoritmos. Entre las principales propiedades de bajo nivel se incluyen: Contenido de información parcial, número de óptimos locales, rugosidad, curvatura, correlación entre variables, etc [23].

2.2. MARCO TEÓRICO

El Teorema No Free Lunch (NFL) establece que no existe un algoritmo que funcione bien en todos los problemas [14]. Por lo tanto, al abordar un problema de optimización, es necesario experimentar con varios algoritmos para identificar cuál es el más adecuado. Si bien existen metodologías racionales que relacionan los dominios del problema con los algoritmos a utilizar, esto no garantiza que el algoritmo seleccionado sea el óptimo para resolver el problema [14, 15]. A partir de esta incertidumbre surge el análisis del paisaje de aptitud (FLA) y el análisis exploratorio de paisajes de optimización (ELA).

Históricamente, las técnicas para cuantificar las características de los paisajes de optimización se conocían bajo el nombre de análisis de paisajes de aptitud (FLA). Sin embargo, estudios más recientes, como el de Mersmann et al. [23], introdujeron el análisis exploratorio de paisajes de optimización (ELA), que es un conjunto de métodos automatizados para extraer características del paisaje. El término análisis de paisajes (LA) engloba tanto las medidas de FLA como las de ELA [17].

En general, una medida FLA está asociada con una característica de alto nivel del paisaje de optimización, mientras que una medida ELA tiende a estar relacionada con varias características de este tipo [23, 17]. Las medidas FLA individuales son interpretables, ya que pueden describir una propiedad de alto nivel de un paisaje de optimización de manera independiente. No obstante, su cálculo es complejo, ya que implica conocer o derivar propiedades matemáticas del paisaje. Por otro lado, las medidas ELA suelen ser menos interpretables de manera individual y están diseñadas para ser utilizadas colectivamente como insumos en modelos de aprendizaje automático, con aplicaciones como la selección y configuración automática de algoritmos. A pesar de su utilidad, las medidas ELA también presentan varios desafíos subyacentes. La distinción entre ELA y FLA no siempre es clara, y con frecuencia, medidas originalmente clasificadas como FLA se utilizan en el contexto de ELA [17].

2.2.1. Fitness Landscape Analisys (FLA)

La función de aptitud u optimización (fitness) se refiere a los valores utilizados para evaluar la calidad de una solución en los procesos iterativos de optimización, los cuales se comparan según el principio de maximización o minimización. Estos valores se calculan en función del problema de optimización del problema en cuestión. Por lo tanto, comprender a fondo las propiedades de un problema de optimización es fundamental para entender cómo este se relaciona con el comportamiento de un algoritmo, lo cual a su vez mejora la capacidad del algoritmo para resolver problemas. En este contexto, el análisis del paisaje de aptitud u optimización (FLA) se ha consolidado como una herramienta analítica poderosa para caracterizar el paisaje de optimización de un problema, y ha ganado una creciente atención en

diversas áreas de optimización [47].

Usualmente, la dificultad de un problema de optimización se evalúa a partir de su función objetivo. En esencia, el paisaje de optimización está compuesto por los valores objetivos de todas las variables en el espacio de decisiones. El análisis del paisaje de optimización, al ser una herramienta analítica eficaz, permite caracterizar de manera detallada dicho paisaje. Por tanto, la dificultad de un problema de optimización se evalúa utilizando el análisis del paisaje de optimización [47]. Algunas propiedades que se puede analizar del paisaje son:

Óptimos locales y modalidad: Al evaluar la dificultad de un problema de optimización, el número de óptimos locales es una característica clave del paisaje de optimización. Un problema de optimización con un solo óptimo local se considera unimodal, mientras que aquellos con múltiples óptimos locales se catalogan como multimodales. La distribución de los óptimos locales pueden proporcionar información más relevante sobre las características del paisaje de optimización, lo que permite una mejor comprensión de la dificultad inherente al problema de optimización. La modalidad del paisaje de optimización es un concepto derivado de la estadística y refleja la distribución de los óptimos locales dentro de los espacios de decisiones y objetivos [47, 48].

Cuenca de atracción y embudo: La cuenca de atracción de los óptimos locales es otra característica relevante del paisaje de optimización. Las cuencas de atracción pueden clasificarse en dos tipos [47, 49]: la cuenca de atracción fuerte, donde todos los individuos dentro de ella convergen exclusivamente hacia un óptimo específico, y la cuenca de atracción débil, donde algunos individuos pueden acercarse a otros óptimos. El análisis de las cuencas de atracción puede revelar información importante sobre el tamaño, la forma, la estabilidad y la distribución de los óptimos locales, lo cual es esencial para evaluar la dificultad de un problema de optimización. El embudo de un paisaje de optimización, por su parte, representa una cuenca global que engloba varios óptimos locales agrupados [47].

2.2.2. Exploratory Landscape Analisys (ELA)

En análisis exploratorio de paisajes de optimización (ELA) es un conjunto de técnicas que son empleadas para obtener conocimiento por medio de (no necesariamente comprensibles) características numéricas sobre las propiedades de un paisaje de optimización de un problema desconocido [23, 50, 25]. El objetivo de esta técnica es explotar el conocimiento adquirido sobre el paisaje para poder dar recomendaciones del algoritmo de optimización más adecuado para la resolución de dicho problema [51]. Estas características halladas en el paisaje podrían usarse para guiar a un conjunto de algoritmos de optimización hacia el óptimo de un problema o simplemente para hacer un reconocimiento y previa reconstrucción del paisaje de optimización[23, 50].

La relevancia del análisis exploratorio de paisajes de optimización radica en que si se usa en problemas de caja negra (desconocidos), donde las evaluaciones de las funciones son consideradas a menudo costosas, disminuye el costo computacional a raíz de medir características que se basan en muestras del paisaje [50]. Una de las características que se puede obtener de los paisajes es:

Contenido de información parcial: Se han establecido algunas posibles clasificaciones de los paisajes dependiendo del contenido de información. El *contenido de información* de un objeto es una medida de lo difícil que es describir dicho objeto [25]. Aplicando este concepto a los paisajes de optimización, se clasifica como un *paisaje neutral* a un objeto de baja información porque se puede describir completamente mediante el uso de los límites del espacio de entrada y la aptitud de una observación. En contraste, se clasifica como un *paisaje rugoso* a un objeto de alta información porque el número de observaciones requeridas para describirlo depende del número de óptimos locales, que pueden ser altos [25, 52]. Propiamente, relacionando las propiedades de alto nivel, se considera que un paisaje de una función con un sólo óptimo (unimodal) es un paisaje neutral, mientras que un paisaje con varios óptimos (multimodal) es un paisaje rugoso [23].

Sea $f : \chi \mapsto Y$ una función donde χ es el espacio de entrada, Y el espacio de salida y $x_i \in \chi$ una observación. Para medir el *contenido de información* de un **paisaje de optimización** f según el método presentado por [27], en un primer instante se analiza la secuencia de los valores de fitness $S = y_1, ..., y_n$ resultado de una caminata aleatoria sobre el paisaje, donde $y_i = f(x_i)$, i = 1,...,n [27]. Sea $\Phi(\varepsilon) = \phi_1, ..., \phi_{n-1}$ una secuencia de símbolos sonde $\phi \in \left\{\bar{1}, 0, 1\right\}$. La secuencia de los valores fitness son convertidos a una secuencia de símbolos siguiendo la siguiente regla:

$$\phi_i = \begin{cases} \bar{1} & y_{i+1} - y_i & < -\varepsilon \\ 0 & \left|y_{i+1} - y_i\right| & \leq \varepsilon \\ 1 & y_{i+1} - y_i & > \varepsilon \end{cases} \tag{2-3}$$

Donde $\varepsilon \geq 0$ es la *sensibilidad de la información*, la cual actúa como un parámetro de precisión de la secuencia de símbolos. Dos símbolos consecutivos componen un **bloque**. Cada bloque representa un objeto en el paisaje de optimización, es decir: una pendiente, un pico o un área neutral. El *contenido de información* de la secuencia se define como:

$$H(\varepsilon) = -\sum_{a \neq b} p_{ab} \log_6 p_{ab} \tag{2-4}$$

Donde $a, b \in \left\{\bar{1}, 0, 1\right\}$ y p_{ab} es la probabilidad de encontrar un bloque ab en la secuencia de símbolos. El logaritmo en base seis es porque ese es el número posible de bloques donde

$a \neq b$. El *contenido de información* se encuentra entre $[0, 1]$ y $0 \log_6 0 \equiv 0$.

Según [27] el *contenido de información* calculado con la ecuación 2-4 mide la variedad de objetos que se encuentran dentro del paisaje de optimización, pero no mide explícitamente el nivel de suavidad del paisaje. Para estimar el nivel de suavidad, se crea una nueva secuencia a partir de $\Phi(\varepsilon)$, $\Phi'(\varepsilon)$, quitando todos los ceros y los valores repetidos. La nueva secuencia tiene la forma "$...\bar{1}, 1, \bar{1}, 1, \bar{1}...$" y representa los cambios de concavidad encontrados durante la caminata aleatoria, por lo tanto $|\Phi'(\varepsilon)|$ es la cantidad de cambios de concavidad encontrados en la secuencia. Se define como el *contenido de información parcial* $M(\varepsilon)$, a la medida de suavidad del paisaje de optimización, y es calculado como:

$$M(\varepsilon) = \frac{\left|\Phi'\right|}{n-1} \tag{2-5}$$

Con $n \gg 1$. El método supone que el paisaje es estadísticamente isotrópico, es decir, que la distribución de probabilidad de los valores fitness obtenidos durante una caminata aleatoria no cambia entre caminatas. Esta es una suposición importante, ya que implica que el método requiere una **muestra imparcial** del espacio de entrada [25].

En el método para un **paisaje de optimización continuo** presentado por [27, 52], la caminata aleatoria es construida muestreando la primera observación (x_1) al azar y las siguientes $n-1$ usando la regla:

$$x_{i+1} = x_i + \rho \cdot \delta \tag{2-6}$$

Donde δ es un vector de variables aleatorias con $\left\|\delta\right\| = 1$, las cuales controlan la dirección y ρ es una variable aleatoria que controla el tamaño del paso y se distribuye uniformemente entre $[0, r]$, donde r es el tamaño de paso máximo [25, 52].

Algunos de los métodos conocidos usan la caminata aleatoria para muestrear el paisaje de entrada, como el que se explicó anteriormente [27], suponiendo que una caminata aleatoria captura cambios en los valores de fitness en el paisaje porque genera una secuencia de observaciones vecinas en el espacio de entrada [27, 25].

En espacios continuos, esta metodología de muestreo presenta limitaciones: en primer lugar, al utilizar una caminata aleatoria con un tamaño de paso ajustable como método de muestreo, se requiere un gran tamaño de muestra para garantizar una muestra imparcial, por lo tanto, el método es costoso en términos de evaluaciones de funciones. Con un tamaño de muestra limitado, la muestra tiene una alta concentración de observaciones en algunas áreas del espacio de entrada, mientras que otras áreas no se muestrean. Por lo tanto, la muestra y los resultados están sesgados. En segundo lugar, no se tiene en cuenta la distancia entre dos observaciones en el análisis [25]. Por lo tanto, aunque ya han habido aproximaciones con el

fin de tomar muestras imparciales de los valores fitness de un paisaje de optimización, este sigue siendo un punto clave dentro del análisis exploratorio de paisaje de optimización.

2.2.3. Propiedad de modalidad en los paisajes de optimización

Uno de los atributos principales investigados en los paisajes de optimización es la modalidad, ya que indica el nivel de dificultad que varios algoritmos pueden encontrar al resolverlos, como se señala en la literatura existente [28, 21]. Si bien numerosos trabajos y bases de datos han intentado categorizar los paisajes de optimización en función de propiedades como la modalidad, la separabilidad, la escalabilidad y la diferenciabilidad, el consenso sigue siendo esquivo. Algunas referencias destacadas en esta área incluyen a Jamil y Yang [46], el repositorio de Al-Roomi [53] y la base de datos AMPGO [54]. Sin embargo, faltan definiciones formales de modalidad en estas referencias. En consecuencia, la siguiente sección resume una definición matemática de unimodalidad débil y multimodalidad.

Unimodalidad y Multimodalidad débil: Teniendo en cuenta problemas de optimización continuos bidimensionales sin restricciones. A continuación se presenta la definición de un problema de optimización continuo descrito en [2]:

$$P \begin{cases} min & \tilde{f}(x) = \tilde{f}(x_1, x_2, ..., x_n) \\ s.t. & x \,\epsilon\, S \subset R^n \end{cases} \tag{2-7}$$

Se hacen dos suposiciones sobre esta definición, la primera es que $S \subset R^n$ es compacto, y la segunda es que la función f es semicontinua inferior. El problema P no está restringido en R^n, por lo que es necesario proporcionar la siguiente función objetivo. Una función extendida $f : R^n \to (-\infty, +\infty]$ se define en la ecuación 2-8.

$$f(x) = \begin{cases} \tilde{f}(x), & x \in S; \\ +\infty, & x \notin S. \end{cases} \tag{2-8}$$

Para comprender el concepto de unimodalidad y multimodalidad débil, es necesario tener en cuenta dos conceptos bastante importantes, el primero es la definición de los diferentes tipos de mínimos que se pueden encontrar en un problema de optimización y el segundo el concepto de conjuntos de nivel. Un conjunto de nivel es un elemento utilizado para poder conectar las definiciones de mínimos con la definición final de modalidad.

$$\begin{array}{ll} \textit{Mínimos globales}: & \exists x^{**} R^n;\ \ f(x^{**}) \leq f(x), \\ \textit{Conjunto}: X^{**} & \\ \textit{Mínimos locales}: & \exists x^*, \exists \delta_1 > 0, \forall x \in B(x^*, \delta_1);\ \ f(x^*) \leq f(x), \\ \textit{Conjunto}: X^* & \end{array} \tag{2-9}$$

En la ecuación 2-9 se aprecian las definiciones de dos tipos de mínimos que se pueden encontrar en un paisaje de optimización. Los mínimos globales se definen como ese punto o

conjunto de puntos que alcanzan el valor más bajo de la función objetivo en todo el paisaje, y los mínimos locales están centrados en una región $B(x^*, \delta_1)$.

Un conjunto de nivel $L^{\leq}(\alpha) \subset R^n$, un conjunto de nivel estricto $L^{<}(\alpha) \subset R^n$ y un conjunto de nivel igual $L^{=}(\alpha) \subset R^n$ en un nivel $\alpha = f(x)$ se definen en la ecuación 2-10.

$$\begin{array}{cc} Conjunto\,de\,nivel & L^{\leq}(\alpha) = \left\{x \in R^n |\, f(x) \leq \alpha\right\} \\ Conjunto\,de\,nivel\,estricto & L^{<}(\alpha) = \left\{x \in R^n |\, f(x) < \alpha\right\} \\ Conjunto\,de\,nivel\,igual & L^{=}(\alpha) = \left\{x \in R^n |\, f(x) = \alpha\right\} \end{array} \tag{2-10}$$

La componente conectada de $L_c^{\leq}(f(x))$ tal que la componente incluye a x se llama conjunto de nivel conectado. Un conjunto de nivel conectado $L_c^{\leq}(\alpha; x)$, conjunto de nivel conectado estrictamente $L_c^{<}(\alpha; x)$ y un conjunto de nivel igual $L_c^{=}(\alpha; x)$ en un nivel $\alpha = f(x)$ se definen en 2-11.

$$\begin{array}{cc} Conjunto\,de\,nivel\,conectado: & L_c^{\leq}(f(x)) = L_c^{\leq}(f(x); x) = L_c^{\leq}(\alpha; x) \\ Conjunto\,de\,nivel\,estricto\,conectado: & L_c^{<}(f(x)) = L_c^{<}(f(x); x) = L_c^{<}(f(x); x) \backslash L_c^{=}(f(x)) \\ Conjunto\,de\,nivel\,igual\,conectado: & L_c^{=}(f(x)) = L_c^{=}(f(x); x) \end{array} \tag{2-11}$$

Reuniendo los conceptos de mínimos y conjuntos de nivel, las expresiones mostradas en la ecuación 2-12 denotan un conjunto de nivel de mínimos globales.

$$\begin{array}{cc} Conjunto\,de\,mínimos\,globales: & \cdot) L^{\leq}(f(x^{**}) - \varepsilon) = 0 \\ (\varepsilon > 0) & \cdot) L^{\leq}(f(x^{**})) = L^{=}(f(x^{**})) \\ & \cdot) L^{<}(f(x^{**})) = 0 \end{array} \tag{2-12}$$

Para cualquier $\varepsilon > 0$, si existe $x \in R^n$ y la ecuación 2-13 se cumple, entonces el conjunto $L_c^{\leq}(f(x))$ se llama conjunto de valores mínimos locales que incluye a x, y se denota como $E_c(f(x^*))$.

$$L^{<}(f(x) - \varepsilon) \cap L_c^{\leq}(f(x)) = 0 \tag{2-13}$$

Cada conjunto de x que satisface la condición anterior se llama Conjunto de Valores Mínimos Locales (l.m.s.v) o E^*, y se define en la ecuación 2-14 [2].

$$E^* = \left\{x^* | \forall \varepsilon > 0; L^{<}(f(x^*) - \varepsilon) \cap L_c^{\leq}(f(x^*)) = 0\right\} \tag{2-14}$$

Permita que el número de componentes conectadas para todos los l.m.s.v (E^*) se llame por un número (inferior) modal, y se denota por la expresión $\# \left|E^*\right|_c$. A partir de la definición, las funciones unimodales (inferiores) débiles y las funciones multimodales (inferiores) se definen en la ecuación 2-15 [2].

$$\begin{cases} \# \left|E^*\right|_c = 1, & funciones\,unimodales\,bajas\,débiles; \\ \# \left|E^*\right|_c \geq 2, & funciones\,multimodales\,bajas\,débiles. \end{cases} \tag{2-15}$$

En esencia, la naturaleza de la multimodalidad o unimodalidad de una función puede determinarse examinando el conjunto de valores, denotado como X^*, que satisfacen la Ecuación 2-14 y pertenecen al conjunto de mínimos locales de la función. Si existe más de un valor en X^*, lo que indica múltiples mínimos locales, la función se considera multimodal. Por el contrario, si solo existe un valor en X^*, que representa un único mínimo global (X^{**}), la función se considera unimodal. Expandiendo este concepto, una función débilmente unimodal posee un único mínimo global sin ningún mínimo local, mientras que una función débilmente multimodal alberga múltiples mínimos locales, posiblemente ocultando un mínimo global dentro de este conjunto.

2.2.4. Caminatas aleatorias y caminatas que siguen una ley de potencias

Las caminatas aleatorias son procesos estocásticos fundamentales en el estudio de la probabilidad y la física estadística. Este tipo de modelos describe trayectorias formadas por una serie de pasos que ocurren de manera aleatoria, y su estudio ha permitido modelar una variedad de fenómenos físicos, biológicos y económicos. Una caminata aleatoria simple puede pensarse como el recorrido de un partícula que se mueve paso a paso, con cada movimiento decidido por una distribución de probabilidad uniforme. En términos matemáticos, cada paso de la caminata aleatoria es independiente del anterior y sigue una distribución de probabilidad predefinida, usualmente gaussiana [44].

Matemáticamente, una caminata aleatoria en una dimensión puede expresarse como:

$$X(t) = \sum_{i=1}^{t} \xi_i, \tag{2-16}$$

donde cada ξ_i es una variable aleatoria que representa un desplazamiento en el tiempo i y se asume típicamente que ξ_i es independiente e idénticamente distribuida, siguiendo una distribución normal de media cero y varianza finita. Este modelo produce una distribución gaussiana en la posición $X(t)$ para tiempos grandes, con una varianza que crece linealmente en el tiempo, $\langle X^2(t) \rangle \propto t$ [44].

En contraste, existen caminatas que exhiben patrones de desplazamiento más complejos y que se ajustan a una ley de potencias. Este tipo de movimiento se caracteriza por la presencia de "colas largas" en la distribución de probabilidad de los saltos, lo cual indica que existe una probabilidad considerable de realizar desplazamientos largos. Las caminatas que siguen una ley de potencias se observan en fenómenos donde ocurren movimientos o eventos excepcionales con frecuencia notable, como el movimiento de animales en búsqueda de alimento o la propagación de epidemias en redes sociales [55]. A continuación se describen algunos de los tipos principales de caminatas que siguen una ley de potencias.

1. **Caminatas de Levy:** Las caminatas de Lévy son un tipo de proceso de movimiento estocástico caracterizado por una distribución de saltos de la forma:

$$P(l) \sim l^{-\mu},\ con\ \mu \in (1,3] \tag{2-17}$$

donde l es la magnitud del salto y μ es un parámetro que determina el "peso" de los saltos largos. En este caso, si $X(t)$ representa la posición de la partícula en el tiempo t, se puede demostrar que para $\mu < 3$, el segundo momento de la distribución es infinito, es decir:

$$\langle X^2(t) \rangle \to \infty \tag{2-18}$$

lo cual conduce a un comportamiento de súper-difusión. La caminata de Lévy permite modelar trayectorias en las que los saltos largos tienen una probabilidad significativa, como en el caso del movimiento de algunos animales o de partículas en medios turbulentos [56, 57].

2. **Caminatas de Lévy Truncadas:** Para algunas aplicaciones, los desplazamientos extremadamente largos no son físicamente plausibles o están limitados por restricciones naturales. En estos casos, se utiliza una caminata de Lévy truncada, en la cual la distribución de saltos se ajusta a una ley de potencias solo hasta una longitud máxima L_{max}:

$$P(l) \sim l^{-\mu}\ para\ l < L_{max} \tag{2-19}$$

y $P(l) = 0$ para $l \geq L_{max}$. Esta distribución permite que el proceso siga una ley de potencias en una escala limitada, y luego decae rápidamente para evitar saltos mayores a L_{max}. Este tipo de caminata es particularmente útil en modelos de difusión limitada, como el movimiento animal en hábitats finitos o en procesos de transporte en materiales con estructura restringida [58].

3. **Caminatas Auto-Afines o con Memoria:** Las caminatas auto-afines introducen dependencia entre los pasos, de manera que cada desplazamiento depende del historial de la caminata. Matemáticamente, esto se expresa introduciendo una correlación entre los desplazamientos. Si $X(t)$ representa la posición en el tiempo t, entonces:

$$X(t) = X(t-1) + \eta(t), \tag{2-20}$$

donde $\eta(t)$ está correlacionado con los valores previos $X(t-k)$ para $k = 1, 2, \ldots$. Este tipo de proceso es útil en modelación de sistemas de búsqueda, donde un agente ajusta su dirección de movimiento en función de su trayectoria anterior. Estos modelos auto-afines son ampliamente aplicados en estudios de eficiencia de búsqueda en entornos donde los recursos son escasos y dispersos [59].

4. **Procesos de Difusión Anómala con Ley de Potencias en los Tiempos de Espera:** En estos procesos, no solo se considera la distribución de saltos, sino también una distribución de tiempos de espera τ entre cada desplazamiento, que sigue una ley de potencias:

$$\psi(\tau) \propto \tau^{-\beta}, \quad 1 < \beta < 2. \tag{2-21}$$

Esto implica que el proceso puede detenerse en una posición durante períodos largos antes de realizar un nuevo desplazamiento, resultando en una difusión anómala. En términos de desplazamiento medio cuadrático, para este tipo de proceso se observa un comportamiento de la forma:

$$\langle X^2(t) \rangle \propto t^{\gamma}, \tag{2-22}$$

donde $\gamma < 1$ para difusión sub-difusiva o $\gamma > 1$ para súper-difusión. Este modelo es útil para describir el transporte en medios con heterogeneidades estructurales, donde la partícula encuentra obstáculos o retrasos en su movimiento [60].

5. **Caminatas Sesgadas o con Preferencia:** Las caminatas con preferencia, conocidas también como "biased random walks", consideran un sesgo direccional o un peso en los movimientos. En un contexto de redes, la probabilidad de que un nodo i se conecte con otro depende del grado de conectividad de este último. Matemáticamente, la probabilidad de que un nodo i se conecte con otro nodo j es proporcional a:

$$P(i \to j) \propto k_j^{\delta}, \tag{2-23}$$

donde k_j es el grado de conectividad de j y δ determina la intensidad de la preferencia. Este modelo genera redes con distribuciones de conectividad que siguen una ley de potencias, característica de redes complejas como las sociales y de comunicación [61].

2.2.5. Algoritmos de búsqueda local

Un algoritmo de búsqueda local es un procedimiento de optimización que busca el mínimo (o máximo) de una función objetivo $f : \mathbb{R}^n \to \mathbb{R}$ mediante la exploración iterativa de un espacio de soluciones, utilizando únicamente evaluaciones de la función f en puntos locales [4, 5].

Formalización

1. **Espacio de Soluciones**: Sea $S \subseteq \mathbb{R}^n$ un conjunto no vacío que representa el espacio de soluciones del problema de optimización.

2. **Función Objetivo**: Sea $f : S \to \mathbb{R}$ una función continua y diferenciable en el espacio de soluciones S.

3. **Estado Actual**: Definimos un estado actual $x_k \in S$ en la iteración k.

4. **Vecindad**: Se define una función de vecindad $N(x_k) \subseteq S$ que retorna un conjunto de soluciones cercanas a x_k. Por lo general, $N(x_k)$ incluye soluciones que se obtienen a partir de x_k aplicando perturbaciones o movimientos locales.

5. **Criterio de Mejora**: Un algoritmo de búsqueda local aplica un criterio de mejora para seleccionar la siguiente solución x_{k+1}:

$$x_{k+1} = \arg \min_{y \in N(x_k)} f(y). \tag{2-24}$$

   Esto implica que se elige el mejor candidato y dentro de la vecindad $N(x_k)$.

6. **Condición de Terminación**: El algoritmo termina cuando se satisface una condición de convergencia, que puede ser:

   - Un número máximo de iteraciones K.
   - La ausencia de mejora en el valor de la función durante un número específico de iteraciones consecutivas.
   - El valor de $f(x_k)$ se encuentra dentro de una tolerancia ϵ respecto al mínimo global.

Notación

En términos formales, el algoritmo de búsqueda local se puede expresar como:

$$x^* = \lim_{k \to \infty} x_k, \tag{2-25}$$

donde x^* es una solución óptima (mínima o máxima) que se aproxima a través del proceso iterativo.

2.2.5.1. Comparación de Algoritmos de Búsqueda Local

A continuación se presenta una tabla que compara diferentes algoritmos de búsqueda local en función de su formalismo matemático, ventajas, desventajas y aplicaciones:

Algoritmo	Formalismo Matemático	Ventajas	Desventajas	Aplicaciones
Nelder Mead [62]	$x_{k+1} =$ $argmin_{y \epsilon N(x_k x k)}$ $f(y)N(x_k)$ (vencindad de x_k)	- No requiere derivadas. - Buena en baja dimensión	- No garantiza convergencia en alta dimensión.	- Ajuste de parámetros. - Optimización de funciones no lineales
Powell [63]	$x_{k+1} =$ $x_k + \lambda_i d_i$, $d_i =$ $x_n - x_0$ (dirección actual)	- Convergencia rápida en funciones cuadráticas. - No requiere derivadas.	- Dependiente de la elección de direcciones iniciales.	- Optimización en ingeniería. - Finanzas
Recocido Simulado [64]	x' (nuevo punto) $P(x) =$ $exp\big(-\frac{f(x')-f(x)}{T}\big)$	- Capacidad para escapar de mínimos locales.	- Lento en la fase final de enfriamiento.	- Problemas de optimización global. - Diseño de redes
Descenso de Gradiente Estocástico [4]	$x_{k+1} =$ $x_k - \alpha \nabla f(x_k)$ α (tasa de aprendizaje)	- Eficiencia en grandes conjuntos de datos.	- Puede quedar atrapado en mínimos locales.	- Entrenamiento de redes neuronales. - Regresión logística

Tabla **2-1**: Comparación de algunos algoritmos de optimización de búsqueda local [4, 5].

2.2.6. Redes Neuronales Convolucionales

Las redes neuronales convolucionales (CNN) se han consolidado como uno de los modelos más destacados en el ámbito del aprendizaje profundo. La visión por computadora basada en CNN ha permitido logros anteriormente impensables, como el reconocimiento facial, los vehículos autónomos, los supermercados autoservicio y los tratamientos médicos inteligentes [65].

Las CNN son un tipo de red neuronal feedforward que puede extraer características de los datos mediante estructuras de convolución. A diferencia de los métodos tradicionales de extracción de características [36, 37, 38], las CNN no requieren extracción manual. Su arquitectura se inspira en la percepción visual [66]. Una neurona biológica se corresponde con una neurona artificial, los núcleos de las CNN actúan como receptores que responden a diversas características y las funciones de activación simulan la función neuronal de transmitir señales eléctricas solo cuando superan un cierto umbral [65]. Las funciones de activación y los optimizadores son herramientas desarrolladas para enseñar a todo el sistema de la CNN a aprender lo que se espera.

Las Redes Neuronales Convolucionales (CNN) presentan varias ventajas en comparación con las Redes Neuronales Fully Connected (FC). Estas ventajas se pueden resumir como sigue [65]:

1. **Conexiones Locales**: Cada neurona en una capa convolucional se conecta únicamente a un subconjunto local de neuronas en la capa anterior, lo que permite reducir significativamente la cantidad de parámetros y acelerar la convergencia del algoritmo. Matemáticamente, esto se puede expresar como:

$$\mathbf{y} = f(\mathbf{W} * \mathbf{x} + \mathbf{b}), \tag{2-26}$$

   donde $\mathbf{y}$ es la salida de la neurona, f es la función de activación, $\mathbf{W}$ representa los pesos de la conexión, $\mathbf{x}$ es la entrada, y $\mathbf{b}$ es el sesgo.

2. **Compartición de Pesos**: En las CNN, un grupo de conexiones puede compartir los mismos pesos, lo que reduce aún más el número total de parámetros a optimizar. Esto se representa mediante el operador de convolución $*$ aplicado a la entrada $\mathbf{x}$, donde el núcleo de convolución $\mathbf{K}$ se aplica repetidamente sobre diferentes regiones de $\mathbf{x}$.

3. **Reducción de Dimensión a través de Submuestreo**: Una capa de pooling se utiliza para submuestrear los mapas de características generados, aprovechando la correlación local en la imagen. Este proceso se formaliza como:

$$\mathbf{z} = \mathcal{P}(\mathbf{y}), \tag{2-27}$$

   donde $\mathcal{P}$ denota la operación de pooling (ya sea max pooling o average pooling) y $\mathbf{z}$ es el resultado del submuestreo. Esto permite reducir la cantidad de datos procesados mientras se conserva la información relevante y se eliminan características triviales. Estas propiedades hacen de las CNN algoritmos altamente representativos en el ámbito del aprendizaje profundo.

Para la construcción de un modelo de CNN, generalmente se requieren cuatro componentes fundamentales. La **convolución** es crucial para la extracción de características y produce mapas de características. Con un núcleo de convolución de tamaño fijo, se pierde información en los bordes de la imagen; para mitigar este efecto, se introduce el **padding**, que expande la entrada con ceros, ajustando indirectamente el tamaño de la salida:

$$\tilde{\mathbf{x}} = \mathrm{pad}(\mathbf{x}), \tag{2-28}$$

donde $\tilde{\mathbf{x}}$ es la entrada modificada. Para controlar la densidad de la operación de convolución, se utiliza el **stride** s. Un valor de stride mayor resulta en una menor densidad de conexiones, que puede expresarse como:

$$\mathbf{y}_{i,j} = f\left(\sum_{m=0}^{k-1}\sum_{n=0}^{k-1} \mathbf{K}_{m,n} \cdot \mathbf{x}_{i+m,j+n}\right), \tag{2-29}$$

donde k es el tamaño del núcleo de convolución y (i, j) representa las coordenadas de la salida.

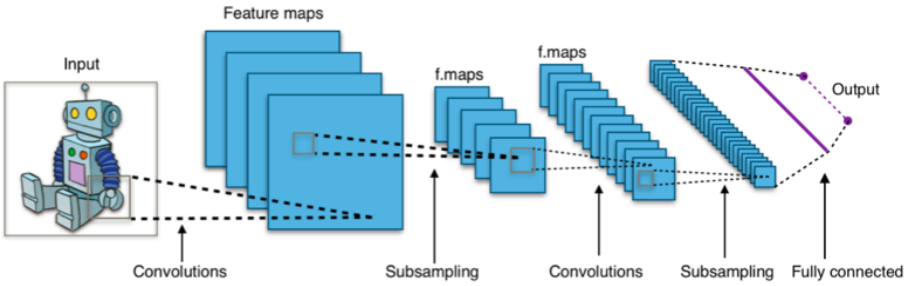


Figura **2-1**: Diagrama de capas de una CNN. Tomada de [3].

Tras la operación de convolución, los mapas de características pueden contener una gran cantidad de información, lo que puede dar lugar a un problema de sobreajuste [67]. Para evitar este fenómeno, se propone el uso de técnicas de pooling (también conocido como submuestreo) [68], que eliminan la redundancia en los datos y permiten conservar las características más significativas. La figura **2-1** ilustra el procedimiento general de una CNN [65].

La visión artificial, especialmente a través de técnicas como las redes neuronales convolucionales (CNN), proporciona numerosas ventajas para tareas de clasificación de imágenes a gran escala en comparación con métodos tradicionales, tales como Redes Neuronales Recurrentes (RNN), Máquinas de Vectores de Soporte (SVM) y algoritmos basados en árboles. Estas ventajas incluyen una mayor precisión, adaptabilidad a diversas tareas de clasificación y la capacidad de realizar la extracción automática de características relevantes de las imágenes, eliminando la necesidad de ingeniería de características manual [36, 37, 38].

2.2.6.1. Función de Activación

La función de activación en una red neuronal artificial actúa análogamente a la función de activación de una neurona en el cerebro humano. En este contexto, cada neurona artificial recibe como entrada los valores de salida de las neuronas de la capa anterior y decide, mediante una función matemática, qué información debe ser transmitida a la siguiente neurona. Formalmente, si se denota el conjunto de entradas de la neurona j como $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, la función de activación puede definirse como:

$$y_j = f\left(\sum_{i=1}^{n} w_{ij} x_i + b_j\right), \tag{2-30}$$

donde Y_j es la salida de la neurona j, w_{ij} representa el peso de la conexión entre la entrada x_i y la neurona j, b_j es el sesgo o estado interno de la neurona j, y $F(\cdot)$ es la función de activación. Esta función puede tomar diversas formas, tales como la función sigmoide, la función tangente hiperbólica (tanh) y la función lineal rectificada (ReLU), entre otras. La estructura de la función de activación se ilustra en la Fig. **2-2** [65].

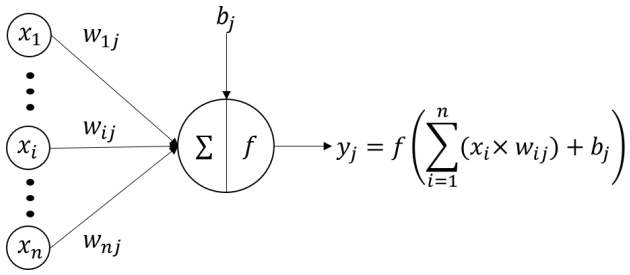


Figura **2-2**: Estructura general de la función de activación (imagen de elaboración propia).

En la Fig. **2-2**, el valor x_i representa una característica de entrada, donde se ingresan simultáneamente n características a la neurona j. La función de activación $f(\cdot)$ juega un papel crucial en la introducción de no linealidades en el modelo, lo que permite a la red neuronal aprender representaciones complejas y realizar tareas de clasificación y regresión efectivas [65, 69].

2.2.6.2. Optimizador

En el contexto de redes neuronales, la optimización de funciones no convexas es una tarea común y desafiante. Los métodos matemáticos convencionales requieren una capacidad computacional considerable, lo que motiva el uso de algoritmos de optimización durante el proceso de entrenamiento. El objetivo principal es minimizar una función de pérdida $L(\theta)$ respecto a los parámetros θ de la red neuronal. Esto se busca con el fin de obtener los parámetros "óptimos" en un tiempo razonable. Entre los algoritmos de optimización más utilizados se encuentran Momentum, Root Mean Square Propagation (RMSprop), y el algoritmo de optimización adaptativa (Adam) [65].

Optimizador Adam: El optimizador Adam (Adaptive Moment Estimation) es un algoritmo que utiliza el gradiente de primer orden para la optimización de funciones objetivo estocásticas. Este método se basa en estimaciones adaptativas de los momentos de primer y segundo orden de los gradientes. Formalmente, el algoritmo actualiza los parámetros θ_t en el tiempo t como sigue:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t, \tag{2-31}$$

donde α es la tasa de aprendizaje, $\hat{m}_t$ es la estimación del primer momento (la media del gradiente), $\hat{v}_t$ es la estimación del segundo momento (la varianza del gradiente), y ϵ es un término de regularización para evitar la división por cero. El pseudocódigo del algoritmo Adam se presenta en el algoritmo 1 según lo descrito en [70]. Para cada parámetro del modelo, se definen dos conjuntos de variables al inicio del entrenamiento. La variable m_t guarda la media de los gradientes, mientras que v_t registra la media de los gradientes elevados al

Algoritmo 1: Algoritmo Adam. Tomado de [70].

Data: α: Tasa de aprendizaje, β_1, $\beta_2 \in [0, 1)$: Tasas de decaimiento exponencial para las estimaciones del momento, $f(\theta)$: Función objetivo estocástica con parámetros θ, θ_0: Vector de parámetros inicial, Momento: m, Segundo momento: v, Iteración: t

Result: Parámetros actualizados: θ_{t+1}

1 Inicialización: $m_0 = 0$, $v_0 = 0$, $t = 0$;
2 **while** *No se alcance el criterio de parada* **do**
3 $\quad t = t + 1$;
4 $\quad g_t = \nabla_\theta J(\theta_{t-1})$;
5 $\quad m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$;
6 $\quad v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$;
7 $\quad \hat{m}_t = \frac{m_t}{1-\beta_1^t}$;
8 $\quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$;
9 $\quad \theta_t = \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}$;
10 **end**

cuadrado. Ambas variables se inicializan en cero. Los coeficientes β_1 y β_2 representan factores de decaimiento exponencial, aplicados respectivamente a la media y a la varianza de los gradientes en el proceso de actualización. En cada iteración del entrenamiento, se calculan los gradientes de los parámetros con respecto a la función de pérdida, y estos se emplean para actualizar m_t y v_t. Debido a la inicialización en cero de m_t y v_t, pueden aparecer sesgos hacia cero, especialmente en las iteraciones iniciales. Para mitigar este sesgo, se aplican correcciones durante las primeras iteraciones. Finalmente, los parámetros del modelo se actualizan usando m_t y v_t, adaptando la tasa de aprendizaje de forma individual para cada parámetro. [70]

Este método es fácil de implementar y computacionalmente eficiente. Además, presenta un bajo requerimiento de memoria, es invariante al reescalado diagonal de los gradientes y es especialmente adecuado para problemas de gran escala en términos de datos y/o parámetros. Asimismo, Adam es ideal para abordar objetivos no estacionarios y problemas que involucran gradientes ruidosos y/o dispersos [70, 65].

2.2.6.3. Transfer Learning

Las redes neuronales convolucionales (CNN) pueden aprovechar el aprendizaje por transferencia para mejorar el rendimiento y acelerar el entrenamiento en nuevos problemas. Formalmente, el aprendizaje por transferencia puede considerarse un conjunto de métodos que facilitan la reutilización de un modelo M_S, preentrenado en una tarea o dominio fuente D_S,

para aplicarlo a una tarea o dominio objetivo D_T [71]. En otras palabras, un modelo preentrenado en un conjunto de datos grande y diverso, como D_S, permite ajustar de manera eficiente los parámetros del modelo M_T para una tarea T_T sobre un conjunto de datos diferente D_T, optimizando la función de pérdida $L_T(\theta)$ mediante una combinación de parámetros θ obtenidos previamente y reajustes específicos.

La arquitectura de algunos modelos preentrenados como MobileNet, ResNet, Xception y VGG son populares en la literatura para aplicar aprendizaje por transferencia [72, 71]. A continuación, se detallan las características matemáticas y arquitectónicas de MobileNet V1 y MobileNet V2.

MobileNet V1: MobileNet V1 es una red neuronal convolucional diseñada para aplicaciones en dispositivos móviles y sistemas integrados, optimizada mediante el uso de convoluciones separables en profundidad. Su estructura minimiza el número de parámetros, y la salida de cada capa convolucional es de la forma:

$$Y = F(W \times X + b), \tag{2-32}$$

donde Y es el mapa de características de salida, W y b son los pesos y el sesgo aprendidos, X representa el mapa de entrada, $F(\cdot)$ es la función de activación, y $\times$ denota la operación de convolución. En la tabla **2-2** se muestra la arquitectura de esta red, que sigue el diseño de MobileNet v1. La red se compone de varias capas fundamentales: una capa convolucional inicial, seguida por una capa de normalización por batch (BatchNormalization) para estabilizar la distribución de los datos y mejorar la convergencia, y una capa de activación ReLU para introducir no linealidad.

A continuación, se emplean capas convolucionales de profundidad (DepthwiseConv2D) y capas convolucionales de punto (PointwiseConv2D o Conv2D), seguidas cada una de sus propias capas BatchNormalization y ReLU. Las capas Depthwise separan la convolución espacial de la convolución en profundidad, lo que reduce significativamente el número de parámetros, mientras que las capas Pointwise aplican una convolución 1×1 para combinar la salida de la capa Depthwise.

También se observa una capa de relleno (ZeroPadding2D) para ajustar las dimensiones de entrada en la siguiente capa de convolución. Este enfoque modular de capas convolucionales y de normalización con activaciones permite una arquitectura eficiente en cuanto a parámetros, lo que hace que MobileNet sea especialmente adecuada para aplicaciones en dispositivos móviles y otros entornos con recursos limitados. En total, la red cuenta con 3,228,864 parámetros (12.32 MB), de los cuales 3,206,976 (12.23 MB) son entrenables y 21,888 (85.50 KB) son no entrenables [72, 71].

Layer (type)	Output Shape	Param #
Input layer	(None, 128, 128, 3)	0
conv# (Conv2D)	(None, 64, 64, 32)	864
conv#_bn (BatchNormalization)	(None, 64, 64, 32)	128
conv#_relu (ReLU)	(None, 64, 64, 32)	0
conv_dw_# (DepthwiseConv2D)	(None, 64, 64, 32)	288
conv_dw_#_bn (BatchNormalization)	(None, 64, 64, 32)	128
conv_dw_#_relu (ReLU)	(None, 64, 64, 32)	0
conv_pw_# (Conv2D)	(None, 64, 64, 64)	2048
conv_pw_#_bn (BatchNormalization)	(None, 64, 64, 64)	256
conv_pw_#_relu (ReLU)	(None, 64, 64, 64)	0
conv_pad_#+1 (ZeroPadding2D)	(None, 65, 65, 65)	0
conv_dw_#+1 (DepthwiseConv2D)	(None, 32, 32, 64)	576
conv_dw_#+1_bn (BatchNormalization)	(None, 32, 32, 64)	256
conv_dw_#+1_relu (ReLU)	(None, 32, 32, 64)	0

Tabla **2-2**: Arquitectura de la red basada en MobileNet.

MobileNet V2: MobileNet V2 es una versión mejorada de MobileNet V1, optimizada mediante la reducción drástica del número de parámetros y la introducción de bloques residuales invertidos. Formalmente, cada bloque residual invertido incluye un cuello de botella lineal de la forma:

$$Y = X + F_{\text{residual}}(X), \tag{2-33}$$

donde $F_{\text{residual}}(X)$ representa la función residual que incorpora convoluciones profundas y convoluciones 1×1, lo cual permite mantener información importante y reducir la redundancia. Los cuellos de botella lineales minimizan la pérdida de información importante, mejorando la precisión y reduciendo las operaciones necesarias para el entrenamiento y la inferencia.

En la tabla **2-3** se muestra la arquitectura de esta red basada en MobileNet V2, una mejora sobre la versión anterior que introduce capas adicionales y optimiza aún más el número de parámetros para mejorar la eficiencia y precisión. La arquitectura utiliza bloques inversos residuales, donde la convolución se expande primero para aumentar el número de canales, y luego se aplica una convolución Depthwise (DepthwiseConv2D) seguida de una convolución de reducción con una operación 1×1 (PointwiseConv2D). Este enfoque permite capturar relaciones de alto nivel con un número reducido de parámetros.

Cada bloque contiene una capa de normalización por batch (BatchNormalization) y una función de activación ReLU6 en lugar de ReLU, que ayuda a prevenir la saturación de activaciones en redes profundas. MobileNet V2 también emplea conexiones residuales entre bloques, lo cual preserva la información de entrada y facilita el flujo de gradientes, mejoran-

Layer (type)	Output Shape	Param #
Input layer	(None, 128, 128, 3)	0
conv1 (Conv2D)	(None, 64, 64, 32)	864
bn_conv1 (BatchNormalization)	(None, 64, 64, 32)	128
conv1_relu6 (ReLU6)	(None, 64, 64, 32)	0
expanded_conv_depthwise (DepthwiseConv2D)	(None, 64, 64, 32)	288
expanded_conv_depthwise_bn (BatchNormalization)	(None, 64, 64, 32)	128
expanded_conv_depthwise_relu6 (ReLU6)	(None, 64, 64, 32)	0
expanded_conv_project (Conv2D)	(None, 64, 64, 16)	512
expanded_conv_project_bn (BatchNormalization)	(None, 64, 64, 16)	64
block_#_expand (Conv2D)	(None, 64, 64, 96)	1536
block_#_expand_bn (BatchNormalization)	(None, 64, 64, 96)	384
block_#_expand_relu6 (ReLU6)	(None, 64, 64, 96)	0
block_#_pad (ZeroPadding2D)	(None, 64, 64, 96)	0
block_#_depthwise (DepthwiseConv2D)	(None, 64, 64, 96)	864
block_#_depthwise_bn (BatchNormalization)	(None, 64, 64, 96)	384
block_#_depthwise_relu6 (ReLU6)	(None, 64, 64, 96)	0
block_#_project (Conv2D)	(None, 32, 32, 24)	2304
block_#_project_bn (BatchNormalization)	(None, 32, 32, 24)	96

Tabla **2-3**: Arquitectura de la red basada en MobileNet V2.

do la eficiencia del entrenamiento en redes profundas.

Este diseño modular con bloques expansivos y residuales hace que MobileNet V2 sea ideal para aplicaciones en dispositivos de recursos limitados al reducir tanto el número de parámetros como la carga computacional sin comprometer la precisión del modelo. En total, la red tiene 2,257,984 parámetros (8.61 MB), de los cuales 2,223,872 (8.48 MB) son entrenables y 34,112 (133.25 KB) son no entrenables [73, 71].

2.2.7. Índices de rendimiento

2.2.7.1. Matriz de Confusión

La matriz de confusión es una herramienta de análisis estadístico ampliamente utilizada para evaluar el rendimiento de modelos de clasificación, especialmente en problemas de clasificación binaria. Esta matriz resume la relación entre las clases reales y las clases predichas, organizando los resultados en cuatro métricas principales que definen el comportamiento del modelo respecto a la asignación de etiquetas de clase. Formalmente, para un conjunto de datos con N muestras y una función de clasificación $f : X \to \{0, 1\}$, la matriz de confusión se define por los siguientes elementos:

- **Verdaderos Positivos (TP)**: Número de muestras correctamente clasificadas como positivas, es decir, el caso donde $f(x) = 1$ y la clase verdadera es positiva.

- **Falsos Negativos (FN)**: Número de muestras clasificadas incorrectamente como negativas, siendo la clase verdadera positiva. Es decir, $f(x) = 0$ cuando la clase real es 1.
- **Verdaderos Negativos (TN)**: Número de muestras correctamente clasificadas como negativas, donde $f(x) = 0$ y la clase real es negativa.
- **Falsos Positivos (FP)**: Número de muestras clasificadas incorrectamente como positivas, cuando la clase real es negativa; esto ocurre si $f(x) = 1$ y la clase real es 0.

Estos valores conforman una matriz 2×2, denominada **matriz de confusión**, que tiene la forma:

$$\text{Matriz de Confusión} = \begin{pmatrix} TP & FN \\ FP & TN \end{pmatrix}$$

La matriz de confusión permite derivar diversas métricas de rendimiento del modelo, tales como la exactitud, la precisión y el recall. Formalmente, se define:

- Exactitud (Accuracy): Proporción de predicciones correctas sobre el total de predicciones:
$$\text{Exactitud} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2-34}$$
- Precisión (Precision): Proporción de predicciones positivas correctas sobre todas las predicciones positivas:
$$\text{Precisión} = \frac{TP}{TP + FP} \tag{2-35}$$
- Sensibilidad (Recall): Proporción de verdaderos positivos sobre el total de casos positivos reales:
$$\text{Sensibilidad} = \frac{TP}{TP + FN} \tag{2-36}$$

Para que un clasificador sea considerado eficiente, los valores de FN y FP deben minimizarse, maximizando así TP y TN [74].

2.2.7.2. Algunos índices de rendimiento en clasificación

La tabla **2-4** resume algunas de las métricas más utilizadas para evaluar el rendimiento de modelos de clasificación de aprendizaje automático. Cada métrica ofrece una perspectiva diferente sobre el comportamiento del modelo, desde la pérdida de entropía cruzada [6, 9], que mide la disimilitud entre distribuciones de probabilidad, hasta el F1-Score, que equilibra la precisión y el recall [7]. Además, se incluyen versiones promediadas de estas métricas para situaciones de clasificación multiclase, donde la evaluación sobre todas las clases debe ser

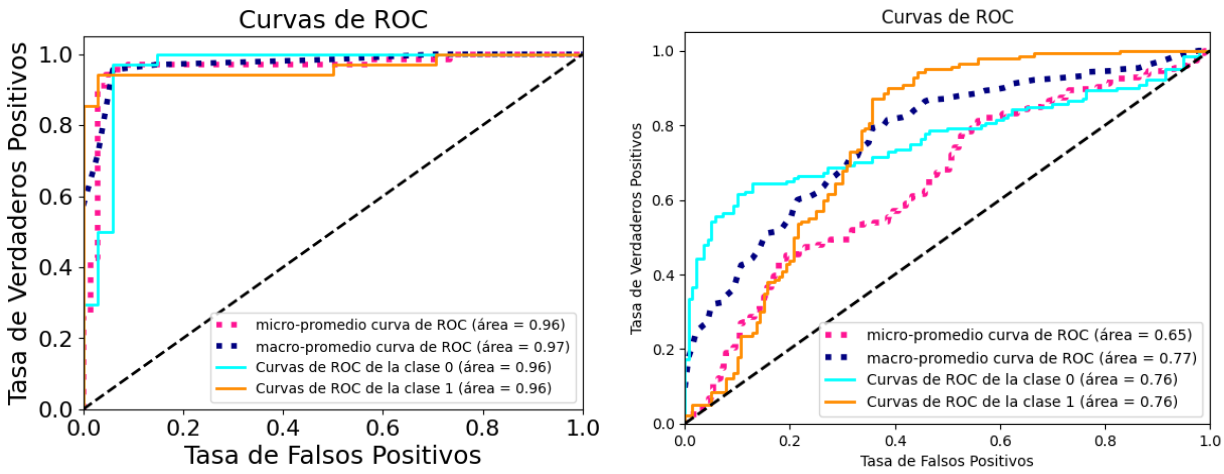
Métricas	Fórmula	Definición
Pérdida: Entropía Cruzada [6, 9]	$L = -\sum_{i=1}^{n} [p(x_i) \log q(x_i)]$	Entre dos distribuciones de probabilidad, mide el número medio de bits necesarios para identificar un evento de un conjunto de posibilidades, si un esquema de codificación está basado en una distribución de probabilidad dada q, más que en la verdadera distribución p.
Exactitud (Accuracy) [7]	$\frac{tp+tn}{tp+fp+tn+fn}$	En general, la métrica de exactitud mide el ratio de predicciones correctas sobre el total de instancias evaluadas.
Exactitud promedio [7]	$\frac{\sum_{i=1}^{l} \frac{tp_i+tn_i}{tp_i+fp_i+tn_i+fn_i}}{l}$	La exactitud promedio de todas las clases.
Índice Kappa de Cohen [8]	$\kappa = \frac{p_o-p_e}{1-p_e}$	Es una medida estadística que ajusta el efecto del azar en la proporción de concordancia observada, donde p_o es la concordancia observada y p_e es la concordancia esperada por azar.
Precisión [7]	$\frac{tp}{tp+fp}$	La precisión se utiliza para medir los patrones positivos que se predicen correctamente a partir del total de patrones predichos como positivos.
Precisión promedio [7]	$\frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i+fp_i}}{l}$	El promedio de precisión por clase.
Índice Recall [7]	$\frac{tp}{tp+fn}$	El índice recall se utiliza para medir la fracción de patrones positivos que se clasifican correctamente.
Recall Promedio [7]	$\frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i+fn_i}}{l}$	El promedio de recuperación por clase.
Índice F1-Score [7]	$\frac{2 \cdot p \cdot r}{p+r}$	Esta métrica representa la media armónica entre los valores de recuperación (recall) y precisión (precision).
F1-Score promedio [7]	$\frac{2 \cdot p_M \cdot r_M}{p_M+r_M}$	El promedio de la medida F por clase.

Tabla **2-4**: Índices de rendimiento en clasificación [6, 7, 8, 9, 10].

justa y equilibrada. También se incluyen métricas como la precisión, el recall y la exactitud, ampliamente utilizadas en tareas de clasificación binaria y multicategórica [7, 10]. La métrica kappa de Cohen [8] se destaca por su capacidad de ajustar los resultados teniendo en cuenta las concordancias obtenidas por azar.

2.2.7.3. Curva ROC y AUC

La Curva Característica Operativa del Receptor (ROC, *Receiver Operating Characteristic*) es un gráfico que visualiza la capacidad de un clasificador binario para distinguir entre clases



(a) Buen desempeño. (b) Bajo desempeño.

Figura **2-3**: Ejemplos de curvas ROC (imagen de elaboración propia).

a medida que se ajusta el umbral de decisión. La curva ROC se construye graficando la *Tasa de Verdaderos Positivos* (TPR) frente a la *Tasa de Falsos Positivos* (FPR) en varios umbrales, donde estas tasas se definen como:

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN}, \qquad FPR = \frac{FP}{N} = \frac{FP}{FP + TN} \tag{2-37}$$

El área bajo la curva ROC, denominada AUC (por sus siglas en inglés, *Area Under the Curve*), es una métrica que cuantifica la efectividad del clasificador; un valor de AUC cercano a 1 indica un alto poder de discriminación entre clases. En la Figura **2-3**a, se muestra un ejemplo de una curva ROC de alto rendimiento, mientras que la Figura **2-3**b representa un modelo con menor capacidad de clasificación, con AUC más cercana a 0.5 [75].

2.2.7.4. Intervalo de Confianza con Prueba t-Student

Para evaluar la fiabilidad de las métricas de desempeño obtenidas, es común calcular un intervalo de confianza basado en la distribución t de Student. En el caso de una muestra de tamaño n y varianza de las mediciones de error σ^2, el intervalo de confianza del 95 % para la media de precisión $\bar{X}$ se define como:

$$IC = \bar{X} \pm t_{\alpha/2,n-1} \frac{\sigma}{\sqrt{n}} \tag{2-38}$$

donde $t_{\alpha/2,n-1}$ es el valor crítico de la t de Student para $n - 1$ grados de libertad y un nivel de significancia del 5 % (95 % de confianza) [45]. Este intervalo proporciona una estimación de la precisión con un margen de error aceptado, contribuyendo a una interpretación robusta del rendimiento del modelo.

2.3. Validación Cruzada

La validación cruzada es una técnica utilizada para evaluar el rendimiento de modelos de aprendizaje automático y se considera un enfoque fundamental para evitar el sobreajuste. Consiste en dividir el conjunto de datos en varios subconjuntos, o "folds", para asegurar que cada parte de los datos sea utilizada tanto para entrenar como para validar el modelo. Este método permite obtener una estimación más precisa de la eficacia del modelo en datos no vistos [30, 76].

El método más común de validación cruzada es la **validación cruzada k-fold**. En este enfoque, el conjunto de datos se divide aleatoriamente en k subconjuntos de igual tamaño. A continuación, el modelo se entrena k veces, cada vez utilizando $k-1$ subconjuntos para el entrenamiento y el subconjunto restante para la validación. Esta técnica garantiza que cada observación del conjunto de datos sea utilizada para la validación exactamente una vez, y se promedia el rendimiento a través de las k iteraciones. La fórmula para la precisión promedio de la validación cruzada k-fold se puede expresar como:

$$\text{Precisión} = \frac{1}{k}\sum_{i=1}^{k} \text{Precisión}_i \tag{2-39}$$

donde Precisión_i es la precisión del modelo en la i-ésima iteración.

Además de k-fold, existen otras variantes como la **validación cruzada estratificada**, que garantiza que cada fold represente adecuadamente la distribución de clases en el conjunto de datos, lo que es particularmente importante en problemas de clasificación con clases desbalanceadas [30, 76].

2.4. Cuantificación de la Incertidumbre entre Imágenes

Para medir la incertidumbre de una imagen generada en comparación con una imagen original, se pueden utilizar diferentes métricas de similitud y error. A continuación se presentan algunas de estas métricas:

2.4.1. Error Cuadrático Medio (MSE)

El Error Cuadrático Medio mide el promedio de los cuadrados de las diferencias entre los píxeles correspondientes de las dos imágenes. Cuanto menor sea el MSE, más similares serán las imágenes [77].

$$\text{MSE} = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(I_{\text{original}}(i,j) - I_{\text{generada}}(i,j)\right)^2 \tag{2-40}$$

donde m y n son las dimensiones de la imagen, e I_{original} e I_{generada} son las intensidades de píxeles en la imagen original y generada, respectivamente [77].

2.4.2. Correlación de Pearson

La Correlación de Pearson mide la correlación lineal entre los píxeles correspondientes de las dos imágenes. Es una métrica de similitud que toma valores entre -1 y 1, donde 1 significa una correlación positiva perfecta, 0 indica que no hay correlación, y -1 es una correlación negativa perfecta [78].

$$r = \frac{\sum \left(I_{\text{original}} - \overline{I_{\text{original}}}\right)\left(I_{\text{generada}} - \overline{I_{\text{generada}}}\right)}{\sqrt{\sum \left(I_{\text{original}} - \overline{I_{\text{original}}}\right)^2 \sum \left(I_{\text{generada}} - \overline{I_{\text{generada}}}\right)^2}} \tag{2-41}$$

2.4.3. Información Mutua Ajustada (AMI)

La Información Mutua Ajustada mide el grado de similitud entre dos imágenes en términos de la cantidad de información compartida. En este caso, AMI evalúa qué tan ïnformativamente" cercana es la imagen generada respecto a la original [79].

$$\text{AMI} = \frac{\text{MI}\left(I_{\text{original}}, I_{\text{generada}}\right) - \mathbb{E}\left[\text{MI}\left(I_{\text{original}}, I_{\text{generada}}\right)\right]}{\text{máx}\left(\text{H}(I_{\text{original}}), \text{H}(I_{\text{generada}})\right)} \tag{2-42}$$

donde:

- MI es la Información Mutua entre las dos imágenes.
- H es la entropía de cada imagen.

2.4.4. Índice de Similitud Estructural (SSIM)

El Índice de Similitud Estructural (SSIM) mide las similitudes en la estructura visual entre las dos imágenes, teniendo en cuenta la luminancia, el contraste y la estructura. SSIM varía entre 0 y 1, siendo 1 la coincidencia estructural completa [80].

$$\text{SSIM}\left(I_{\text{original}}, I_{\text{generada}}\right) = \frac{\left(2\mu_{I_{\text{original}}}\mu_{I_{\text{generada}}} + C_1\right)\left(2\sigma_{I_{\text{original}}, I_{\text{generada}}} + C_2\right)}{\left(\mu^2_{I_{\text{original}}} + \mu^2_{I_{\text{generada}}} + C_1\right)\left(\sigma^2_{I_{\text{original}}} + \sigma^2_{I_{\text{generada}}} + C_2\right)} \tag{2-43}$$

donde:

- μ representa la media de cada imagen.
- σ es la desviación estándar.
- C_1 y C_2 son constantes para evitar divisiones por cero.

2.5. Trabajos relacionados

El problema de selección de algoritmos de optimización es un área crítica dentro del campo de la inteligencia artificial y la investigación operativa. La premisa fundamental es que no existe un algoritmo de optimización que funcione mejor en todas las instancias de problemas [14]. Por ello, surge la necesidad de seleccionar el algoritmo adecuado para un problema específico en función de sus características o propiedades.

En los últimos años, el estudio de ELA y FLA ha sido objeto de investigación para proporcionar una caracterización de los paisajes de optimización y con esto, información para la selección del algoritmo [17]. Sin embargo, aunque estas técnicas han mostrado resultados prometedores, también presentan limitaciones. Una de las limitaciones más visibles, es que tienden a confiar demasiado en la calidad del muestreo del paisaje, lo que puede llevar a errores si las regiones clave del espacio de búsqueda no son suficientemente exploradas [25, 28].

En estudios más recientes, el aprendizaje automático ha desempeñado un papel clave para abordar el problema de selección de algoritmos usando las técnicas que caracterización del problema. Un estudio preliminar subrayó la importancia de los modelos de dificultad empírica, que utilizan características de los problemas para predecir el esfuerzo computacional requerido por diferentes algoritmos [19]. A raíz de esto, se han desarrollado marcos como Auto-WEKA [81], donde la selección de algoritmos y la optimización de hiperparámetros se automatizan mediante técnicas de meta-aprendizaje.

El meta-aprendizaje ha sido particularmente efectivo para seleccionar algoritmos, ya que aprende de instancias previas del problema, permitiendo recomendar algoritmos adecuados para nuevos problemas no vistos, sin embargo, su efectividad depende de la calidad del conjunto de datos de entrenamiento, lo que puede llevar a un sobreajuste y un rendimiento subóptimo en problemas nuevos [82]. Además, la búsqueda exhaustiva en el espacio de algoritmos y sus hiperparámetros puede resultar en altos costos computacionales, limitando su escalabilidad, especialmente en conjuntos de datos grandes [83]. También es sensible a la calidad de los modelos base utilizados, lo que afecta la capacidad de selección del sistema. Por último, no se aborda de manera exhaustiva la interpretabilidad de los modelos complejos resultantes, lo que puede ser problemático en aplicaciones que requieren una comprensión clara de las decisiones del modelo [81].

Aunque las investigaciones en optimización de caja negra y los marcos basados en aprendizaje automático [84, 85, 86, 87] han ampliado el alcance de la selección de algoritmos al permitir una evaluación continua de las características del problema y la comparación de múltiples algoritmos en tiempo real, este enfoque presenta varias limitaciones. Primero, la flexibilidad prometida puede traducirse en un aumento significativo de la complejidad compu-

tacional, ya que la evaluación constante de algoritmos puede resultar costosa en términos de tiempo y recursos, especialmente para problemas grandes o de alta dimensión [83]. Además, la eficacia de estos sistemas depende en gran medida de la calidad y representatividad de las características extraídas, lo que puede llevar a un rendimiento subóptimo si las características no capturan adecuadamente la naturaleza del problema [88]. Por último, la necesidad de adaptarse continuamente a diferentes problemas puede dificultar la generalización del modelo, planteando dudas sobre su efectividad en situaciones del mundo real donde los problemas presentan dinámicas complejas y cambiantes [89].

Nuevos enfoques han abordado recientemente el problema de selección de algoritmos mediante el uso de sistemas de recomendación para metaheurísticas, proporcionando resultados interesantes [33, 90, 29, 91]. Este enfoque utiliza redes neuronales profundas que correlacionan el rendimiento de varias metaheurísticas con las posibles expresiones algebraicas detrás de los problemas de optimización. No obstante, además de ser extremadamente costoso computacionalmente, este enfoque carece de valor analítico, ya que no arroja luz sobre la conexión entre las características del problema de optimización y el algoritmo correspondiente para resolver dicho problema.

Así mismo, estudios recientes han abordado la selección de algoritmos mediante la extracción de características específicas del problema, como conjuntos particulares de propiedades descriptivas del espacio de búsqueda [92], atributos específicos del dominio del problema [93] o incluso características topológicas del paisaje de optimización [94]. Estos enfoques segmentan la caracterización tradicional de Análisis Exploratorio del Paisaje (ELA) y se enfocan en propiedades de alto nivel, tal como lo sugiere el Análisis del Paisaje de Fitness (FLA), para obtener un conocimiento más detallado del paisaje de optimización antes de abordar el problema de selección de algoritmos.

Sin embargo, la eficacia de estos métodos depende de la forma en que se muestrean las características del paisaje. Un muestreo insuficiente o sesgado puede llevar a una representación incompleta del espacio de búsqueda, lo cual afecta la capacidad del modelo para capturar patrones generales del problema y reduce la confiabilidad de las características extraídas. Por ejemplo, una muestra limitada en zonas críticas del espacio de búsqueda podría pasar por alto características importantes, mientras que una cobertura excesiva de otras regiones irrelevantes puede aumentar los costos computacionales sin añadir valor. Además, debido a la alta dimensionalidad de muchos problemas de optimización, el muestreo exhaustivo es impráctico, por lo que los métodos actuales se ven forzados a tomar decisiones con datos potencialmente insuficientes [28, 95].

Asimismo, estos enfoques suelen asumir que las características muestreadas en una pequeña porción del espacio de búsqueda son representativas de todo el paisaje, una suposición que

puede no ser válida en problemas de gran escala o con paisajes irregulares. Esta limitación cuestiona la capacidad de generalización de los métodos basados en características, ya que una caracterización incompleta puede llevar a seleccionar algoritmos subóptimos para resolver el problema en su totalidad [28, 95].

Estos estudios muestran la amplitud de enfoques utilizados para resolver el problema de la selección de algoritmos, con un hilo común de identificar y aprovechar las características específicas de los problemas para guiar la elección del algoritmo, así mismo se pueden apreciar y estudiar sus limitaciones las cuales pueden llevar a avances posteriores.

Capítulo 3

METODOLOGÍA DE CLASIFICACIÓN PROPUESTA

3.1. Construcción de la base de datos de problemas de optimización

La base de datos utilizada proviene de [1], y abarca funciones de referencia presentadas en el artículo de Jamil y Yang [46], el repositorio de Al-Roomi [53], el paquete R: Smoof [96], el repositorio de benchmarks de CI-lib [97], así como las competiciones CEC de 2013, 2014, 2015 y 2017, y GECCO BBOB [98]. Esta compilación resulta en una colección exhaustiva de 24 funciones objetivo de prueba, que abarca problemas de un solo objetivo y acotados. Es notable que los criterios de selección, tal como se describen en [1], aseguran una cobertura de todas las características del paisaje medidas en las bases usadas.

Cada función se somete a una inspección visual para asignarle la etiqueta correspondiente, basada en la definición descrita en la sección 2.2.3 brindada por el trabajo de Kanemitsu

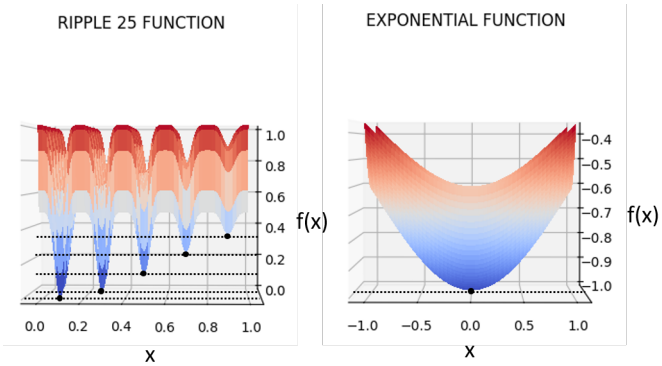


Figura **3-1**: Ejemplo de mínimos en paisajes de optimización. Elaboración propia.

[2]. La Fig. **3-1** ilustra un ejemplo con dos funciones de optimización y la ilustración de los diferentes mínimos que se pueden observar en los paisajes. El objetivo es llevar a cabo el proceso de etiquetación para determinar su propiedad de modalidad. Al examinar la función Ripple 25, es evidente que contiene varios mínimos locales distribuidos en diferentes regiones del paisaje de optimización. En el paisaje pueden delinearse conjuntos de nivel conectados, que encapsulan los valores de las variables X^* correspondientes a estos mínimos locales.

Tomando los puntos de referencia de $f(x) = [0{,}13, 0{,}21, 0{,}38, 0{,}41]$ en la función Ripple 25, se puede observar que retomando la ecuación 2-14 (sección 2.2.3):

$$E^* = \left\{x^* | \forall \varepsilon > 0; L^{<}(f(x^*) - \varepsilon) \cap L_c^{\leq}(f(x^*)) = 0\right\} \tag{3-1}$$

el l.m.v.s es $E = [0{,}1, 0{,}25, 0{,}46, 0{,}68, 0{,}9]$, con el mínimo global en $X^{**} = [0{,}1]$. Por lo tanto, el número de componentes conectados es $\#\left|E\right| c = 5$, lo que indica que esta función es débilmente multimodal según la Ecuación 2-15.

Por otro lado, en la función exponencial, se observa que solo existe un mínimo global en $X^{**} = [0]$. En consecuencia, el conjunto de valores de los extremos locales sería $E = [0]$, con el número de componentes conectados siendo $\#\left|E^*\right| c = 1$, lo que sugiere que esta función es débilmente unimodal.

En la Tabla **3-1** se muestra la caracterización de las funciones utilizadas en la base de datos mostrando con su mínimo global, límites de las variables dentro de los cuales se evalúa la función, el número de dimensiones típicamente utilizado en experimentos de optimización y por último su clasificación en unimodal o multimodal dependiendo del número de mínimos locales.

La tercera columna de la Tabla **3-1** muestra mapas de calor de las funciones propuestas como multimodales y unimodales, donde el color azul indica valores más bajos y el rojo valores más altos. Al observar estos mapas de calor, se observa que las funciones clasificadas como multimodales exhiben áreas con puntos de diferentes escalas, lo que sugiere la presencia de múltiples mínimos locales posibles dentro de los paisajes. Por el contrario, las funciones unimodales muestran áreas de color azul uniforme, lo que indica la convergencia hacia un único valor mínimo. Es notable destacar algunas funciones: entre las multimodales, la función Boyandi-Michalewicz muestra fluctuaciones continuas dentro de su área predominantemente azul, lo que indica la presencia de mínimos locales. Por otro lado, entre las funciones unimodales, destacan las funciones Needle Eye y Step N.3. La primera presenta un descenso abrupto que conduce a un mínimo global, mientras que la segunda, aunque tiene texturas, se caracteriza por rasgos discretos que indican la presencia de un único mínimo global.

Función	**Paisaje de optimización**	**Mapas de calor**	**Mínimo Global**	**Límites de las variables**	**Dimensiones**	**Número de Mínimos Locales**
Schwefel 1			$f(x) = 0$ en $x = 0$	$[-500, 500]$	Variable, comúnmente $n = 10$	1 (Unimodal)
Exponential			$f(x) = 1 - e^{-n}$ en $x = 0$	$[-1, 1]$	Variable	1 (Unimodal)
Needle Eye			$f(x) = 0$ en $x = 0$	$[-5, 5]$	Variable	1 (Unimodal)
Step N3			$f(x) = 0$ en $x = 0$	$[-5{,}12, 5{,}12]$	Variable	1 (Unimodal)
Generalized Paviani			$f(x) \approx -45{,}7785$ en $x \approx 9{,}7$	$[2{,}001, 10]$	10	1 (Unimodal)
Brown			$f(x) = 0$ en $x = 0$	$[-4, 4]$	Variable	1 (Unimodal)
Mishra 7			$f(x) = 0$ en $x = (2, \ldots, 2)$	$[-10, 10]$	Variable	1 (Unimodal)
Mishra 1			$f(x) = 2$ en $x = 1$	$[-10, 10]$	Variable	1 (Unimodal)
Rosenbrock			$f(x) = 0$ en $x = (1, \ldots, 1)$	$[-5, 10]$	Variable	1 (Unimodal)
BBOB 2 (Bent Cigar)			$f(x) = 0$ en $x = 0$	$[-5, 5]$	Variable	1 (Unimodal)
BBOB 6 (Attractive Sector)			$f(x) = 0$ en $x = 0$	$[-5, 5]$	Variable	1 (Unimodal)
Discus			$f(x) = 0$ en $x = 0$	$[-5, 5]$	Variable	1 (Unimodal)

Función	**Paisaje de optimización**	**Mapas de calor**	**Mínimo Global**	**Límites de las variables**	**Dimensiones**	**Número de Mínimos Locales**
Elliptic			$f(x) = 0$ en $x = 0$	$[-5, 5]$	Variable	1 (Unimodal)
Qing			$f(x) = 0$ en $x_i = \pm\sqrt{i}$	$[-500, 500]$	Variable	> 1 (Multimodal)
Ripple 25			$f(x) = 0$ en $x = 0$	$[-5, 5]$	Variable	> 1 (Multimodal)
Generalized Giunta			$f(x) = 0{,}0644704$ en $x = 0{,}46732$	$[-1, 1]$	Variable	> 1 (Multimodal)
Cosine Mixture			$f(x) = -0{,}1n$ en $x = 0$	$[-1, 1]$	Variable	> 1 (Multimodal)
Generalized Price 2			$f(x) = 0$ en $x = (0, \ldots, 0)$	$[-500, 500]$	Variable	> 1 (Multimodal)
Generalized Egg Crate			$f(x) = 0$ en $x = 0$	$[-5, 5]$	Variable	> 1 (Multimodal)
Pinter 2			$f(x) = 0$ en $x = 0$	$[-10, 10]$	Variable	> 1 (Multimodal)
BBOB 16 (Weierstrass)			$f(x) = -0{,}5n$ en $x = 0$	$[-0{,}5, 0{,}5]$	Variable	> 1 (Multimodal)
BBOB 17 (Schaffer F7)			$f(x) = 0$ en $x = 0$	$[-5, 5]$	Variable	> 1 (Multimodal)
Generalized Drop Wave			$f(x) = -1$ en $x = 0$	$[-5{,}12, 5{,}12]$	Variable	> 1 (Multimodal)
Bonyadi-Michalewicz			$f(x) = 0$ en $x_i = -1$ para cada i	$[-1, 1]$	Variable	> 1 (Multimodal)

Tabla **3-1**: Caracterización de las funciones.

3.2. Algoritmo de muestreo exploratorio multiescala (Multiscale Scouting Samplig Algorithm - MUSSA)

El algoritmo de muestro multiescala busca ser una forma de muestreo que emplee herramientas que le permitan extraer información del paisaje tanto globalmente como localmente. El término "scouting sampling" se emplea para reflejar la capacidad del algoritmo de realizar una caracterización multiescala del paisaje al combinar exploración y explotación. En este algoritmo, propiamente en este trabajo, una caminata de Lévy se utiliza para recorrer la escala gruesa del paisaje, mientras que dos algoritmos de búsqueda local complementan el muestreo en la escala fina: el algoritmo de búsqueda local de Powell y el algoritmo de Nelder-Mead como se muestra en la Figura **3-2**.

3.2.1. Construcción de la base de datos de imágenes a partir del muestreo multiescala

El Algoritmo 2 describe la integración de la caminata de Lévy con los algoritmos de búsqueda local descritos en las secciones anteriores para generar las imágenes.

Los datos de entrada requeridos por los problemas incluyen: la función objetivo, los límites de la función, el número de pasos de Lévy, el número de experimentos (o imágenes) y el tipo de función, ya sea unimodal o multimodal, que es esencial para formar la base de datos.

La condición inicial para la primera iteración del vuelo de Lévy se establece como $X_{LS}(0) = 0$. Luego, se determina el primer paso de Lévy. Posteriormente, el paso de Lévy calculado sirve como condición inicial para ejecutar tanto el algoritmo de búsqueda local de Powell como el algoritmo de Nelder-Mead. Las soluciones obtenidas por estos algoritmos, junto con sus correspondientes evaluaciones de la función objetivo, se almacenan en una matriz X_t (que

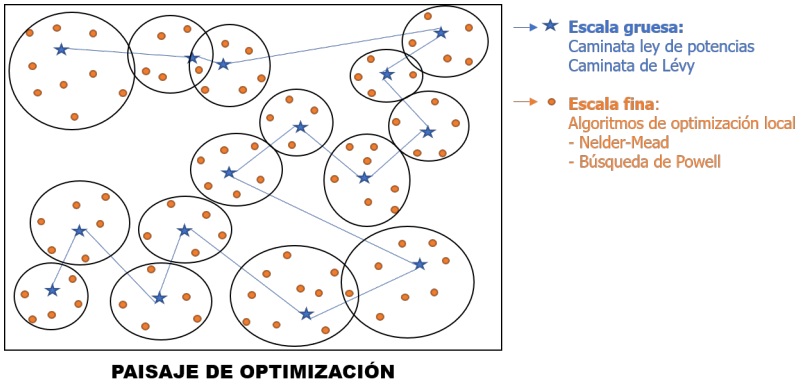


Figura **3-2**: Funcionamiento de MUSSA.

Algoritmo 2: Construcción de imagen

Data: $f()$ - Función objetivo, (lb, ub) - Límite inferior y límite superior de la función, $p(l)$ - Número de pasos de Lévy, exp - Número de experimentos, etiqueta de la función objetivo (Unimodal o Multimodal).

Result: $Im_s(X_t, F_t)$ Imagen de la gráfica de dispersión del muestreo.

```
1  for Experimentos ← 1 hasta exp do
2      X_LS(0) ← 0; for Pasos de Lévy (LS) ← 1 hasta p(l) do
3          X_LS+1 ← X_LS + l; if X_LS+1 < lb then
4              X ← lb
5          else
6              if X_LS+1 > ub then
7                  X ← ub
8          X_g ← x_g, : x_0 = X_LS+1 ;  /* Cálculo del primer algoritmo: Búsqueda de
            Powell. */
9          F_g ← f(X_g) ;  /* Cálculo de la función objetivo con las soluciones.
            */
10         X_i ← x_i, : x_0 = X_LS+1 ;  /* Cálculo del segundo algoritmo: Búsqueda de
            Nelder-Mead. */
11         F_i ← f(X_i) ;  /* Cálculo de la función objetivo con las soluciones.
            */
12         X_t ← (X_g, X_i); F_t ← (F_g, F_i);
13     Im_s(X_t, F_t) ;    /* Construcción de la imagen del gráfico de dispersión
        por experimento. */
   ;   /* Construcción de la base de datos: Inclusión de la imagen según su
       etiqueta. */
```

depende del número de variables, en este caso, dos) y en un vector F_t, respectivamente.

El paso final consiste en generar la imagen creando un gráfico de dispersión de los puntos que representan las soluciones trazadas contra sus correspondientes evaluaciones de la función. Es importante destacar que la imagen resultante incorpora colores para distinguir entre los puntos mínimos y máximos de la función, representados en una escala de azul a rojo.

Ejemplos de las imágenes obtenidas tras la ejecución del algoritmo usando los parámetros descritos en la Tabla **3-3**, junto con su comparación con el paisaje de optimización original usando algunas métricas de similitud (sección 2.4) como un primer acercamiento a la medición de semejanza de las imágenes, se ilustran en la Tabla **3-2**. Se puede observar que, en problemas multimodales, el algoritmo captura eficazmente tanto las regiones altas como

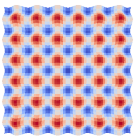
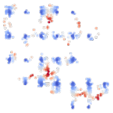
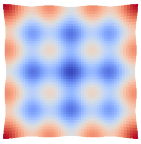
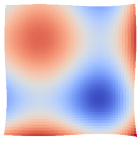
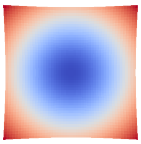
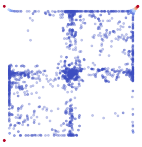
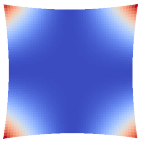
Problema de optimización	**Paisaje de optimización**	**Imagen del paisaje muestreado**	MSE	Correlación de Pearson	AMI	SSIM
Generalized price 2			0,031	0,284	0,647	0,652
Generalized egg crate			0,017	0,497	0,703	0,74
Generalized Giunta			0,019	0,543	0,815	0,746
Exponencial			0,022	0,528	0,816	0,725
Brown			0,071	0,474	0,509	0,687
Mishra 7			0,055	0,471	0,687	0,711

Tabla **3-2**: Obtención de imágenes con MUSSA de algunos problemas de optimización.

Parámetros	**Valor**
p(l) (Pasos Lévy)	128
Tolerancia para el cambio en las variables de decisión (xtol - Powell Search y Nelder Mead)	1^{-4}
Tolerancia para el cambio de la función objetivo (ftol - Powell Search y Nelder Mead)	1^{-4}

Tabla **3-3**: Parámetros de MUSSA (parámetros de los paquetes del lenguaje de programación Python: [11] y [12]).

bajas del paisaje, intentando representar su textura. Por otro lado, en problemas unimodales, la imagen muestra predominantemente los puntos mínimos alcanzados por los algoritmos de búsqueda local, aunque también se evidencia la representación de los máximos dentro del paisaje. En términos de las métricas, el muestreo parece captar la información del paisaje original.

La complejidad del algoritmo surge de su estructura iterativa, bucles anidados y el número de evaluaciones de la función. Incluye dos bucles principales (para los experimentos y los pasos de Lévy) y emplea dos técnicas de optimización: los algoritmos de Powell y Nelder-Mead, los cuales tienen diferentes tasas de convergencia y exigencias computacionales. Además, se aplican condiciones de frontera (límites inferior y superior) para garantizar la viabilidad, lo que añade complejidad lógica al algoritmo. A medida que aumenta el tamaño del problema (por ejemplo, más experimentos o un espacio de búsqueda más amplio), el tiempo de ejecución y el esfuerzo computacional también crecen, lo que hace que el equilibrio entre precisión y eficiencia sea crucial.

3.2.2. Caminata de Lévy

Inspirados por la efectividad de las caminatas basadas en leyes de potencia para modelar comportamientos dinámicos, el objetivo es descubrir propiedades ocultas de los paisajes examinando sus características a gran escala. Las caminatas de Lévy, son comunes en movimientos biológicos, algunos trabajos han proporcionado evidencia numérica de que esta estrategia de caminata facilita los procesos de difusión eficientes en redes y mejora significativamente la tasa de entropía en comparación con las estrategias de caminatas aleatorias [99].

La caminata de Lévy es un tipo de caminata con comportamiento similar al caos. Es una caminata de l pasos que sigue una ley de potencias $P(l) \sim l^{-\mu}$, donde $\mu \in (1, 3]$ y representa el exponente de la ley de potencias. La caminata de Lévy se observa en una variedad de movimientos de agentes biológicos, que van desde células hasta mamíferos, y se caracteriza porque permite al agente tomar decisiones flexibles entre explotar o explorar su entorno (subsección 2.2.4). Esta característica conduce a una mejora en la eficiencia de búsqueda en comparación con una caminata aleatoria normal u otros movimientos como caminatas brownianas [57, 100].

La generación de una caminata X_{LS} con paso l de tamaño $p(l)$ no es trivial. Un esquema simple discutido en [100] y sus trabajos anteriores utilizados para el algoritmo de búsqueda de cuco multiobjetivo se puede resumir de la siguiente manera.

Una versión simple de una caminata de Lévy se puede definir como:

$$\begin{gathered} X_{LS}(i+1) = X_{LS}(i) + l,\ X_{LS}(0) = 0 \\ (for\ iterations\ i = 0, 1, ..., p(l)) \end{gathered} \tag{3-2}$$

Un paso de Lévy l se define como:

$$l = \frac{u}{(|v|)^{\frac{1}{\beta}}} \tag{3-3}$$

donde u y v se extraen de distribuciones normales. Es decir:

$$u \sim N(0, \sigma_u^2),\ v \sim N(0, \sigma_v^2) \tag{3-4}$$

con:

$$\sigma_u = \left(\frac{\gamma(1+\beta)sin(\frac{\pi\beta}{2})}{\gamma(\frac{1+\beta}{2})\beta 2^{\frac{\beta-1}{2}}}\right)^{\frac{1}{\beta}},\ \sigma_v = 1 \tag{3-5}$$

Aquí Γ es la función Gamma estándar, y β es una distribución uniforme entre 1 y 2. Este modelo fue implementado en [101], y presenta una facilidad para su implementación en diferentes lenguajes de programación.

3.2.3. Algoritmos de búsqueda local

La idea principal detrás del uso de algoritmos de búsqueda local no es necesariamente emplear algoritmos computacionalmente costosos, sino más bien aquellos algoritmos tradicionales que faciliten la adquisición simple de datos y proporcionen información sobre el problema. Entre los algoritmos tradicionales, se eligieron el algoritmo de búsqueda de Powell y el algoritmo de Nelder-Mead porque no requieren el cálculo de las derivadas de la función, lo que los

hace más rápidos en comparación con otros algoritmos de búsqueda local. Estos algoritmos son simples de implementar y tienen un bajo costo computacional por iteración. Además, ambos algoritmos son flexibles y efectivos para adaptarse a las diversas funciones objetivo encontradas en la literatura (subsección 2.2.5) [102].

3.2.3.1. Algoritmo de búsqueda local de Powell:

El método de búsqueda de Powell es un algoritmo propuesto por Michael J. D. Powell para encontrar un mínimo local de una función utilizando el método de direcciones conjugadas. La ventaja de este método es que la estructura del algoritmo es simple, fácil de implementar y computacionalmente eficiente [103, 104]. A continuación se presenta el resumen presentado en [103] del algoritmo de búsqueda de Powell.

La dirección de búsqueda para la l-ésima coordenada de la g-ésima dimensión del espacio de búsqueda de N dimensiones (f) se puede definir como:

$$f_l^g = \begin{cases} 1; & g = l \\ 0; & g \neq l \end{cases} \; (g = 1, 2, ...N; l = 1, 2, ...N) \tag{3-6}$$

La longitud del paso λ_g^* para la g-ésima variable de decisión se puede determinar como:

$$\lambda_g^* = \lambda_g^{min} + rand \; \times (\lambda_g^{max} - \lambda_g^{min}) \; (g = 1, 2, ...N) \tag{3-7}$$

donde $\lambda^{min}g$ y $\lambda^{max}g$ son la longitud mínima y máxima del paso para la g-ésima variable de decisión, respectivamente.

El vector de la variable de decisión x_g se actualiza una vez en la dirección de la coordenada (l) como:

$$x_g = x_0 + \lambda_g^* \; \times f_l^g \; (g = 1, 2, ...N) \tag{3-8}$$

donde x_0 representa la condición inicial.

El vector de las variables de control se modifica sobre la base del valor mínimo de la función objetivo. Para todas las "N" coordenadas, este proceso continúa. La dirección de búsqueda del patrón se obtiene para el próximo ciclo de optimización de la siguiente forma:

$$f^g l = xg - L_g \tag{3-9}$$

donde L_g es el valor inicial de la variable de decisión x_g. Además, una de las direcciones de coordenadas se descartó en la dirección del patrón "M" como:

$$f^g m = f^g l \tag{3-10}$$

El proceso continúa hasta que se descarte toda la dirección de la coordenada y todas las operaciones se reinicien en una de las direcciones de la coordenada nuevamente. Finalmente, hasta que el método de Powell alcance el máximo de iteraciones, el proceso de actualización continúa. La implementación se hizo usando el paquete del lenguajes de programación Pyhton [11].

3.2.3.2. Algoritmo de Búsqueda de Nelder-Mead (NMSA):

El Algoritmo de Nelder-Mead [62] fue introducido como un método para minimizar una función de valor real $f(x)$ para $x \in R^n$. Es el método de búsqueda directa más utilizado para la minimización de funciones reales sin restricciones. El algoritmo opera comparando los valores de la función en los $V+1$ vértices x_i de un simplex [105]. A continuación, se presenta un resumen del proceso tal como se describe en [105]:

Se inicializa un simplex de tamaño a en x_0 con base en la siguiente regla:

$$x_i = x_0 + pe_i + \sum_{k=1,k\neq i}^{V} qe_k,\ i = 1, \ldots, V \tag{3-11}$$

donde e_i son los vectores base unitarios y:

$$p = \frac{a}{V\sqrt{2}}(\sqrt{V+1} + V - 1),\ q = \frac{a}{V\sqrt{2}}(\sqrt{V+1} - 1) \tag{3-12}$$

Los vértices del simplex se ajustan mediante operaciones de reflexión, expansión y contracción para encontrar un punto de mejora. El algoritmo termina cuando los valores de la función en los vértices convergen, lo que se mide mediante la desigualdad:

$$\sqrt{\sum_{i=1}^{V+1} \frac{(f_i - \bar{f})^2}{V}} < \varepsilon,\ \bar{f} = \frac{1}{V+1}\sum_{i=1}^{V+1} f_i \tag{3-13}$$

donde ε es un umbral positivo pequeño. El efecto acumulado de estas operaciones en el simplex es estirar su forma a lo largo de las direcciones de descenso y hacer zoom en los óptimos locales [105]. La implementación se hizo usando el paquete del lenguajes de programación Pyhton [12].

3.3. Clasificación de la base de datos de imágenes con Redes Neuronales Convolucionales (CNN)

La visión artificial, particularmente con técnicas como las redes neuronales convolucionales (CNN), ofrece numerosas ventajas para tareas de clasificación de imágenes a gran escala

en comparación con los métodos tradicionales como RNNs, SVMs o algoritmos basados en árboles. Estas ventajas incluyen mayor precisión, adaptabilidad a diversas tareas de clasificación y la extracción automática de características relevantes de las imágenes, eliminando la necesidad de realizar ingeniería manual de características [36, 37, 38].

El aprendizaje por transferencia abarca un conjunto de métodos que permiten traspasar el conocimiento adquirido al abordar un problema para resolver otros, lo que facilita el desarrollo rápido de modelos efectivos para abordar tareas complejas de visión por computadora o procesamiento de lenguaje natural. Este enfoque implica utilizar un modelo previamente entrenado (inicialmente entrenado en un conjunto de datos) para posteriormente entrenarlo con la base de datos propia y realizar predicciones en un conjunto de datos diferente [71].

En este trabajo, se aprovechó el modelo preentrenado MobileNet V2 debido a su sencilla implementación y eficiencia. MobileNet está diseñado meticulosamente como una arquitectura de red simplificada con un conjunto mínimo de hiperparámetros, lo que permite la creación de modelos compactos y de baja latencia que pueden personalizarse fácilmente para diversas aplicaciones de visión por computadora. Además, la literatura ha demostrado que MobileNet ofrece resultados prometedores en tareas de detección de objetos (o patrones) [72, 71].

La Fig. **3-3** ilustra la implementación de TensorFlow de MobileNet V2 utilizada en este estudio. Empleando la biblioteca ImageDataGenerator de TensorFlow, las imágenes se reescalan y se introducen en la red con dimensiones de 128x128x3[1] para mejorar la velocidad de procesamiento. Para el aprendizaje por transferencia, las capas superiores del modelo preentrenado se omiten para permitir que la red realice la extracción de características y evitar que los pesos de las capas convolucionales se actualicen antes de ensamblar y entrenar el modelo. Dado que el modelo preentrenado está entrenado para clasificar en 1000 clases, las capas de salida se ajustan manualmente para adaptarse al problema de clasificación propuesto. La tarea final de clasificación implica una clasificación categórica entre 'multimodal' y 'unimodal', lo que requiere dos salidas. Por lo tanto, las capas finales incluyen una capa de GlobalAveragePooling2D para convertir las características en un vector de 1280 elementos por imagen, seguida de una capa densa con dos salidas que emplea una función de activación softmax.

Posteriormente, el modelo se ajusta utilizando el optimizador Adam [70], la función de pérdida de entropía cruzada categórica [9, 6], y la exactitud (accuracy) como índice de rendimiento [10, 7]. Además, se eligió un tamaño de lote de 32, el cual es comúnmente utilizado en el entrenamiento de CNN debido a diversos factores. Los tamaños de lote pequeños tienden a mejorar la generalización del modelo al proporcionar una estimación más estocástica del gradiente, lo que conduce a una exploración más amplia en el espacio de parámetros y poten-

[1]Este tamaño se escoge por costos computacionales.

cialmente previene la convergencia a mínimos locales. Además, al utilizar una potencia de dos, como 32, se pueden aprovechar optimizaciones de hardware en la GPU, lo que resulta en un entrenamiento más rápido [106].

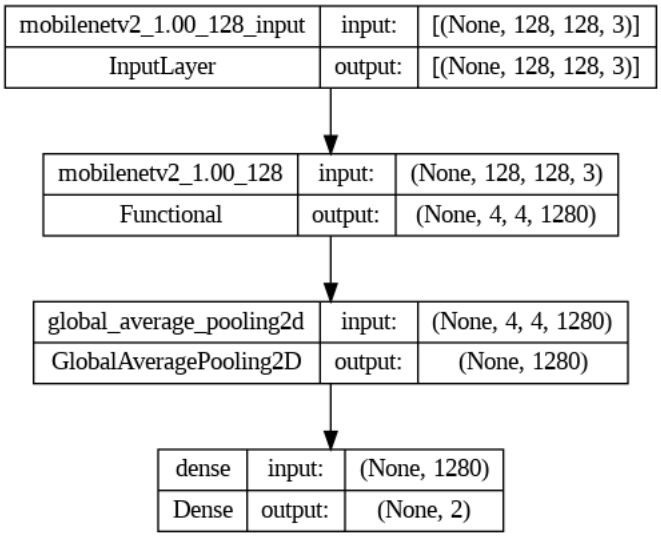


Figura **3-3**: Modelo de transferencia de Aprendizaje con MobileNet V2.

Capítulo 4

RESULTADOS

4.1. Marco experimental

4.1.1. Primer experimento

Inicialmente, se construyó una base de datos de imágenes utilizando el algoritmo de muestreo exploratorio multiescala (MUSSA) para evaluar el Mecanismo de Clasificación descrito, como se ilustra en la Tabla **4-1**. Dado el número limitado de funciones, se realizaron múltiples experimentos del algoritmo para cada función, lo que resultó en imágenes distintas para cada experimento independiente. La configuración experimental involucró la variación en el número de pasos de Lévy, incluyendo 8, 16, 32, 64 y 128 pasos, con bases de datos separadas creadas para cada caso. Para cada una de estas bases de datos, se entrenaron 30 modelos CNN utilizando aprendizaje por transferencia y validación cruzada (sección 2.3), siguiendo los porcentajes descritos en la Tabla **4-1**, lo que resultó en un total de 150 modelos CNN.

Después de ejecutar 15 modelos de CNN por paso, se observó que los modelos podían alcanzar un 100 % de exactitud (accuracy) en el entrenamiento después de 50 épocas, sin mostrar mejoras en la validación, aunque tardaban más en ejecutarse. Por lo tanto, se fijó el parámetro

Tabla **4-1**: Construcción de la base de datos para el entrenamiento, validación y prueba de los modelos.

Distribución de la base de datos	**Etiquetado del problema**	**Número de funciones**	**Número de experimentos (imágenes independientes)**	**Imágenes totales**	**%**
Entrenamiento	Unimodales	14	50	700	71.43 %
	Multimodales	10	70	700	
Validación	Unimodales	14	10	140	14.29 %
	Multimodales	10	14	140	
Prueba	Unimodales	7	20	140	14.29 %
	Multimodales	5	28	140	

para el número de épocas durante el entrenamiento y la validación en 50 para todos los modelos de CNN.

4.1.1.1. Funciones de prueba

La Tabla **4-2** ilustra las funciones empleadas para la prueba de los modelos y su caracterización, las cuales serán utilizadas con el objetivo de identificar el mejor modelo para cada experimento. Estas funciones son comúnmente encontradas en la literatura [46, 53, 97, 54]. Fueron clasificadas de acuerdo con el procedimiento descrito en la sección 3.1, siguiendo la misma metodología aplicada a las funciones de entrenamiento y validación. Es importante destacar que estas funciones no se utilizaron durante las fases de entrenamiento ni de validación, garantizando que los modelos nunca han estado expuestos a ellas.

4.1.2. Segundo experimento

Se realizó un examen de la contribución de las muestras generadas por el primer y segundo algoritmo de búsqueda local (Powell search algorithm (PSA) y Nelder mead search algorithm (NMSA)), el cual se describe en la Tabla **4-3**. Se calculó el número promedio de muestras generadas en cada paso de Lévy y se redondeó al número entero más cercano. Se observó que aproximadamente el 90 % de las muestras fueron producidas por el segundo algoritmo. En consecuencia, se diseñó una prueba posterior utilizando únicamente las muestras obtenidas con el segundo algoritmo. Se aplicó el mismo marco experimental descrito en la Tabla **4-1**, lo que resultó en 30 modelos adicionales de CNN entrenados por paso de Lévy, sumando un total de otros 150 modelos de CNN.

4.1.3. Tercer experimento

4.1.3.1. Tercer experimento: base de datos de imágenes de muestras aleatorias uniformes (URSA)

Para garantizar la ausencia de sesgo en el modelo seleccionado como el mejor después de la pos validación, se realizó un análisis comparativo utilizando una base de datos de imágenes muestreadas aleatoriamente. Esta base de datos se construyó mediante un muestreo uniforme (subseccion 2.1.5) en las funciones de prueba y esta base de imágenes fue llamada URSA. El número promedio proporciona un tamaño de muestra aproximado para cada base de datos correspondiente (ver Tabla **4-4**). Es importante destacar que, debido a la metodología de muestreo establecida, el tamaño de las muestras posteriores se duplicó aproximadamente con respecto a las anteriores. Siguiendo esta progresión, se crearon bases de datos para 2 y 4 pasos, cada una con sus respectivos tamaños de muestra aproximados. Posteriormente, se construyeron bases de datos que comprenden 280 imágenes para cada representación de paso, manteniendo la consistencia con la estructura descrita en la Tabla **4-1**.

Función	Paisaje de optimización	Mapa de calor	Mínimo Global	Límites de las Variables	Dimensiones	Número de Mínimos Locales
Alpine 01			$f(x) = 0$ en $x = (0, \ldots, 0)$	$[-10, 10]$	Variable	> 1 (Multimodal)
Griewank			$f(x) = 0$ en $x = (0, \ldots, 0)$	$[-600, 600]$	Variable	> 1 (Multimodal)
Parsopoulos			$f(x) = 0$ en $x = (0, \ldots, 0)$	$[-1, 1]$	Variable	> 1 (Multimodal)
Bohachevsky			$f(x) = 0$ en $x = (0, 0)$	$[-100, 100]$	2	> 1 (Multimodal)
Holder Table			$f(x) = 0$ en $x = (1, 1)$	$[-10, 10]$	Variable	> 1 (Multimodal)
Sphere			$f(x) = 0$ en $x = (0, \ldots, 0)$	$[-5{,}12, 5{,}12]$	Variable	1 (Unimodal)
Sum Squares			$f(x) = 0$ en $x = (0, \ldots, 0)$	$[-5, 5]$	Variable	1 (Unimodal)
Adjiman			$f(x) = 0$ en $x = 0{,}1$	$[-10, 10]$	Variable	1 (Unimodal)
Bartelsconn			$f(x) = 0$ en $x = 0$	$[-2, 2]$	Variable	1 (Unimodal)
Brent			$f(x) = 0$ en $x = 1$	$[-10, 10]$	Variable	1 (Unimodal)
Schwefel 20			$f(x) = 0$ en $x = 0$	$[-500, 500]$	Variable	1 (Unimodal)
Price 3			$f(x) = 0$ en $x = 0$	$[-10, 10]$	Variable	1 (Unimodal)

Tabla **4-2**: Caracterización de las funciones de prueba.

Pasos Lévy	Promedio de muestras en PSA	Desviación estandar en las muestras de PSA	Promedio de muestras en NMSA	Desviación estandar en las muestras de NMSA	Promedio de muestras en PSA y NMSA	Desviación estandar en las muestras PSA + NMSA
8	30	13	339	140	369	147
16	59	26	668	282	727	297
32	120	56	1302	575	1422	608
64	241	112	2601	1185	2842	1251
128	479	213	5196	2447	5675	2570

Tabla **4-3**: Contribución de los algoritmos de búsqueda local al muestreo.

Pasos Lévy	**Promedio de muestras**	**Desviación estándar**	**Aproximación de muestras aleatorias**
2	-	-	92
4	-	-	184
8	369	147	368
16	727	297	736
32	1422	608	1472
64	2842	1251	2944
128	5675	2570	5888

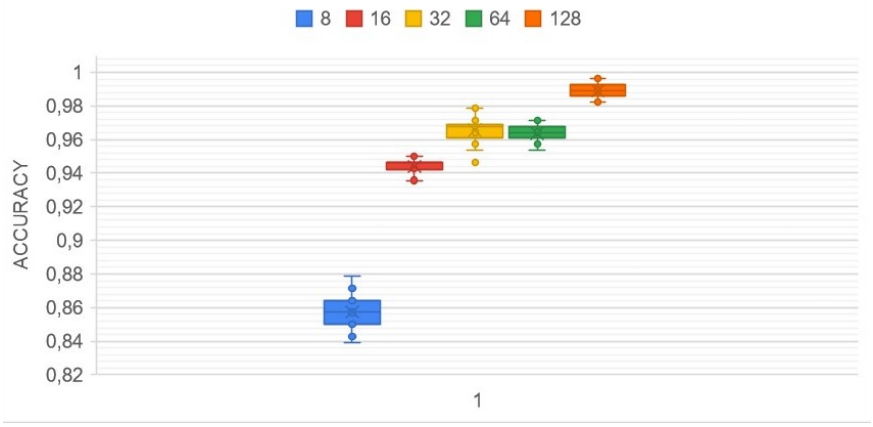
Tabla **4-4**: Configuración del muestreo aleatorio uniforme para cada representación de pasos de Lévy.

4.2. Resultados

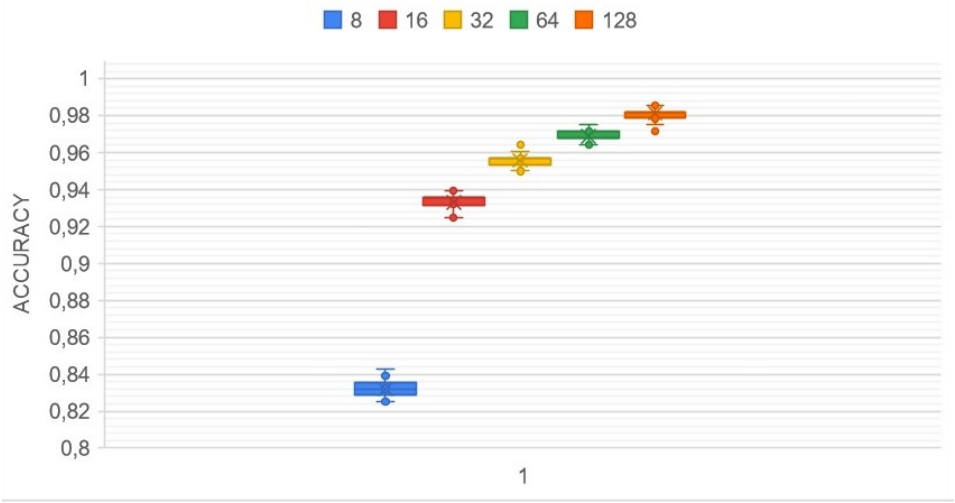
4.2.1. Resultados primer y segundo experimento

La Figura **4-1** presenta el diagrama de cajas que muestra la exactitud (accuracy) en validación de los 30 modelos CNN a través de cada base de datos correspondiente a los pasos de Lévy. El gráfico revela una tendencia de aumento en la exactitud (accuracy) conforme aumentan los pasos de Lévy, lo que sugiere que los modelos muestran un rendimiento mejorado en la validación a medida que el número de muestras en las imágenes del paisaje crece, lo que concuerda con los hallazgos en [1]. Investigaciones previas concluyen que al aumentar el número de muestras del paisaje se logra una exploración más amplia del mismo. Cabe destacar que el primer experimento generalmente arroja mejores resultados de exactitud (accuracy), mientras que el segundo experimento muestra menos variabilidad en los datos de exactitud (accuracy) a través de los 30 modelos CNN.

Por otro lado, la Figura **4-2** ilustra los resultados de la prueba de los 30 modelos CNN en términos de exactitud (accuracy). A pesar de la tendencia creciente observada en la exactitud (accuracy) de validación con respecto a los pasos de Lévy, se hace evidente que, para el primer experimento, se produce una caída en la exactitud (accuracy) en 64 y 128 pasos de Lévy, mientras que para el segundo experimento, esta caída es notoria a partir de los 128 pasos de Lévy. Este fenómeno sugiere que el sobremuestreo del paisaje podría llevar al sobre-



(a) Validación para PSA + NMSA.



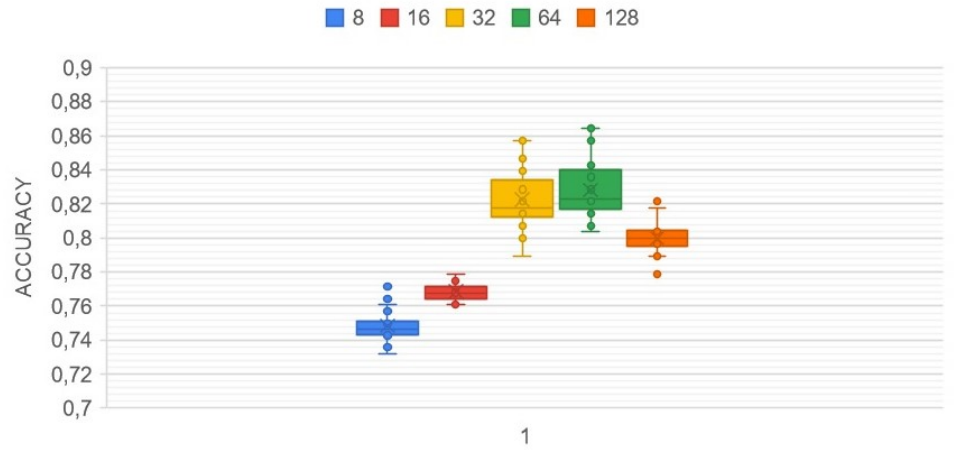
(b) Validación sólo con NMSA.

Figura **4-1**: Comparación de validación utilizando PSA y NMSA y solo NMSA

ajuste, lo que resulta en que los modelos no logren un rendimiento adecuado de clasificación en las funciones de pos validación. Esto también se puede observar en la Tabla **4-5**, donde se nota que los intervalos de confianza (IC) (subsección 2.2.7.4) en la prueba para el primer experimento son mayores para 32 y 64 pasos Levy, obteniendo en promedio el mayor valor de exactirud en 32 pasos Levy, pero presentando una disminución en el intervalo de confianza para 128 pasos Levy. Mientras que, para el segundo experimento, el mayor valor promedio en exactirud se obtiene para 64 pasos Lévy, y decae para 128 pasos Levy.

Los modelos de CNN con la mayor exactitud en validación para cada paso de Lévy fueron seleccionados para una prueba extensa, considerando seis métricas adicionales: matriz de confusión, curvas ROC, coeficiente kappa de Cohen, precisión, recall y puntaje F1.

(a) Prueba para PSA + NMSA.



(b) Prueba sólo para NMSA.

Figura **4-2**: Comparación pos validación utilizando PSA y NMSA y solo NMSA.

Pasos Levy	Promedio de muestras	Desviación estándar de las muestras	IC en exactitud (accuracy) en validación	IC en exactitud (accuracy) en prueba	Promedio en exactitud (accuracy) en prueba	Desviación en exactitud (accuracy) en prueba
Primer experimento						
8	369	147	(0,85,0,864)	(0,751,0,765)	0,758	0,008
16	727	297	(0,937,0,95)	(0,815,0,829)	0.822	0,007
32	1422	608	(0,958,0,972)	**(0,83,0,843)**	**0,837**	0,012
64	2842	1251	(0,957,0,97)	(0,828,0,841)	0,835	0,007
128	5675	2570	(0,982,0,995)	(0,72,0,734)	0,727	0,02
Segundo experimento						
8	339	140	(0,83,0,833)	(0,744,0,751)	0,748	0,009
16	668	282	(0,931,0,934)	(0,766,0,77)	0,768	0,004
32	1302	575	(0,954,0957)	(0,816,0,828)	0,822	0,016
64	2601	1185	(0,967,0969)	**(0,822,0,833)**	**0,828**	0,016
128	5196	2447	(0,979,0,982)	(0,795,0,804)	0,8	0,012

Tabla **4-5**: Intervalos de confianza para la exactitud en validación y prueba.

La Figura **4-3** muestra una comparación de las curvas ROC obtenidas en los dos experimentos. Se puede apreciar que en general todos los modelos obtienen un área bajo la curva mayor a 0,8 siendo el modelo de 16 pasos Lévy obtenido con el primer experimento, el que mejor resultados obtiene en términos de este índice (Figura **4-3**b).

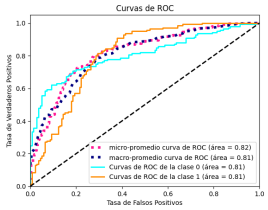
Una comparación de los resultados de los mejores modelos en términos de exactitud en validación, agrupados por experimento y paso de Lévy, se presenta en la Tabla **4-6**. Se puede apreciar que excepto para el caso de 128 pasos Levy, los demás modelos de CNN derivados del primer experimento superaron a los del segundo experimento en las estadísticas de prueba. Por el contrario, el segundo experimento arrojó mejores estadísticas en comparación con el primero para 128 pasos.

Analizando la Tabla **4-6**, aunque el área bajo la curva ROC del modelo de 32 pasos del primer experimento es inferior en comparación con el modelo de 16 pasos del mismo experimento, supera a otros modelos de CNN en las cinco métricas restantes. En consecuencia, se selecciona el modelo de 32 pasos de Lévy del primer experimento, que alcanzó una precisión del 85.3 %, como el mejor modelo en términos de la prueba.

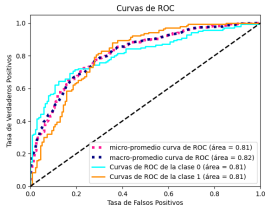
Las Tablas **4-7** y **4-8** muestran el tiempo de ejecución aproximado en minutos para el primer y segundo experimento. Se puede apreciar que en general el segundo experimento es menos demorado, debido a que las imágenes tienen menos muestras. En general el experimento más demorado fue con la base de 128 pasos, sin embargo, es importante notar que el mejor modelo obtenido fue con la base de 32 pasos usando ambos algoritmos de muestreo, y el tiempo para obtener una etiqueta de funciones que no han sido vistas por la red (funciones de prueba) es de 2,24 minutos.

Experimentos		**PSA + NMSA**	**NMSA**	**PSA + NMSA**	**NMSA**	**PSA + NMSA**	**NMSA**	**PSA + NMSA**	**NMSA**	**PSA + NMSA**	**NMSA**
Validación	**Pasos Lévy**	**8**		**16**		**32**		**64**		**128**	
	Pérdida: Entropía cruzada	0,342	0,372	0,166	0,213	0,087	0,144	0,078	0,089	0,028	0,055
	Exactitud	0,879	0,843	0,950	0,939	0,854	0,964	0,971	0,975	0,996	0,989
Prueba: Matriz de Confusión	**Etiqueta verdadera Multimodal**	89	101	**107**	99	105	**107**	106	**107**	88	97
	Porcentaje	64 %	72 %	**76 %**	71 %	75 %	**76 %**	76 %	**76 %**	63 %	69 %
	Etiqueta verdadera Unimodal	119	105	125	119	**134**	122	125	122	110	132
	Porcentaje	85 %	75 %	89 %	85 %	**96 %**	87 %	89 %	87 %	79 %	94 %
Prueba: ROC	**Clase 0 (Multi) ÁREA**	0,81	0,81	**0,88**	0,85	0,87	0,86	0,86	0,85	0,81	0,87
	Clase 1 (Uni) ÁREA	0,81	0,81	**0,88**	0,85	0,87	0,86	0,86	0,85	0,81	0,87
Métricas de prueba	**índice Kappa de Cohens**	0,485	0,471	0,657	0,557	**0,707**	0,63571	0,65	0,63571	0,414	0,63571
	Exactitud	0,742	0,735	0,828	0,778	**0,853**	0,81786	0,825	0,81786	0,707	0,81786
	Precisión	0,754	0,735	0,834	0,784	**0,869**	0,82155	0,831	0,82155	0,712	0,83905
	Recall	0,742	0,735	0,828	0,778	**0,853**	0,81786	0,825	0,81786	0,707	0,81786
	F1 Score	0,739	0,735	0,827	0,777	**0,851**	0,81733	0,824	0,81733	0,705	0,81497

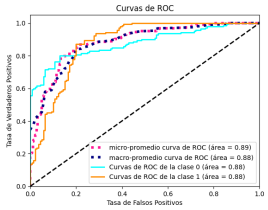
Tabla **4-6**: Comparación integral de la prueba entre experimentos.



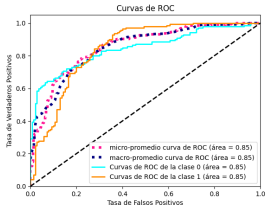
(a) Curva ROC para 8 pasos (Exp 1.).



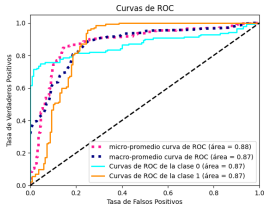
(b) Curva ROC para 8 pasos (Exp 2.).



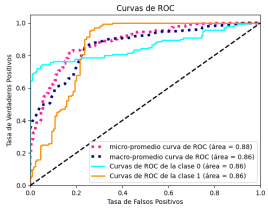
(c) Curva ROC para 16 pasos (Exp 1.).



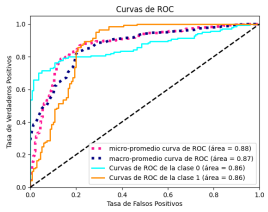
(d) Curva ROC para 16 pasos (Exp 2.).



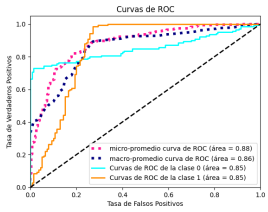
(e) Curva ROC para 32 pasos (Exp 1.).



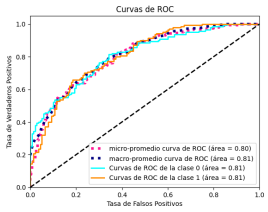
(f) Curva ROC para 32 pasos (Exp 2.).



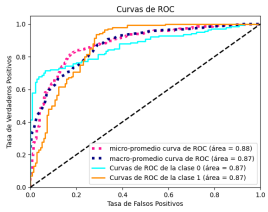
(g) Curva ROC para 64 pasos (Exp 1.).



(h) Curva ROC para 64 pasos (Exp 2.).



(i) Curva ROC para 128 pasos (Exp 1.).



(j) Curva ROC para 128 pasos (Exp 2.).

Figura **4-3**: Curvas ROC del primer y segundo experimento para diferentes pasos en las funciones de prueba.

PSA + NMSA				
Tiempo de ejecución aproximado (min)				
Pasos	**Entrenamiento**	**Validacion**	**Prueba**	**Total**
8	21	3,59	6,53	31,12
16	34,32	6,66	3,73	44,71
32	97,34	19,1	6,55	122,99
64	138,43	28,65	10,91	177,99
128	300	55,84	21,32	377,16

Tabla **4-7**: Tiempo de ejecución aproximado en minutos para el primer experimento.

NMSA				
Tiempo de ejecución aproximado (min)				
Pasos	**Entrenamiento**	**Validacion**	**Prueba**	**Total**
8	9,36	1,57	0,74	11,67
16	13,54	2,62	0,98	17,14
32	23,38	4,42	2,24	30,04
64	41,73	8,22	3,53	53,48
128	79,78	17,18	6,6	103,56

Tabla **4-8**: Tiempo de ejecución aproximado en minutos para el segundo experimento.

4.2.2. Resultados tercer experimento

La Tabla **4-9** presenta los resultados del experimento de prueba realizado con la base de datos de imágenes de muestras aleatorias uniformes, propuesta en la subsección 4.1.3.1. En cada representación de los pasos, el mejor modelo seleccionado anteriormente, obtenido con el mecanismo de clasificación propuesto, logra una exactitud 20 % mayor en comparación con la base de datos de muestreo aleatorio uniforme. Cabe destacar que el modelo alcanza una precisión máxima del 85.3 % en la base de datos de 32 pasos de Lévy.

La Figura **4-4** ilustra una comparación de las curvas ROC, mostrando que la curva ROC obtenida con URSA presenta una caída más pronunciada en comparación con la generada utilizando MUSSA. Esto sugiere que el modelo entrenado con la base de datos muestreada

Comparación de imágenes de la base de datos	**Test ROC**		**Test indices**				
Base de datos de imágenes	***Clase 0 (Multi) ÁREA***	***Clase 1 (Uni) ÁREA***	***Exactitud (accuracy)***	***Cohens kappa***	***Precisión***	***Recall***	***f1***
MUSSA	**0,877**	**0,872**	**0,853**	**0,707**	**0,869**	**0,853**	**0,851**
URSA	0,648	0,768	0,585	0,171	0,773	0,585	0,499
Diferencia	0,229	0,104	0,268	0,536	0,096	0,268	0,352

Tabla **4-9**: Comparación de pruebas entre MUSSA y URSA.

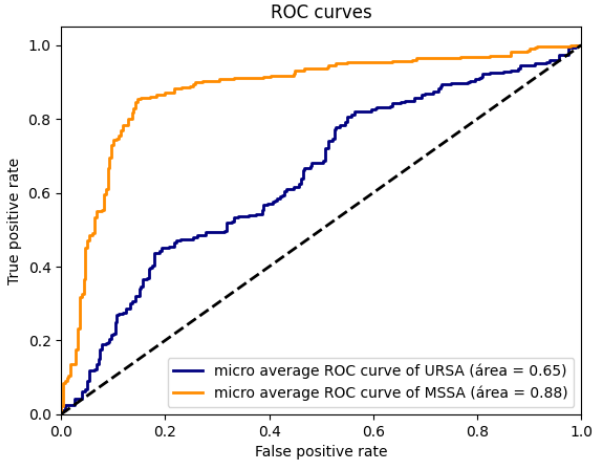


Figura **4-4**: Comparación de las curvas ROC entre MUSSA y URSA

aleatoriamente muestra un rendimiento menos estable o es más propenso a errores en ciertas regiones del espacio de características, mientras que el modelo entrenado con el método de muestreo multiescala mantiene una mayor robustez y precisión en la clasificación.

4.3. Discusión

La clasificación de los paisajes de optimización en este estudio se basa en la definición expuesta en [2]. Sin embargo, es crucial reconocer que el proceso de clasificación es subjetivo y depende de la interpretación de dicha definición. En consecuencia, la clasificación de las funciones de problemas de referencia para el entrenamiento de modelos sigue siendo un área que requiere más investigación en futuros trabajos. En investigaciones futuras, esta misma definición podría utilizarse para realizar una segmentación de problemas de optimización y una clasificación más directa.

Los resultados de los primeros dos experimentos muestran que aunque en la validación la exactitud aumenta a medida que aumenta el número de pasos Lévy, en la prueba, este valor disminuye para 64 y 128 pasos Lévy en el primer experimento y 128 pasos Lévy en el segundo. Este codo podría deberse a varias razones, una posible razón podría indicar que para más pasos puede haber un sobre muestreo que lleve al sobre ajuste del modelo lo que podría implicar que el modelo está generalizando con menos muestras, otra hipótesis es que, podría darse debido a la compresión de la imagen a la hora de entrenar la red neuronal convolucional. En cuanto a las métricas de prueba, se pudo apreciar que exceptuando por la base de datos de 128 pasos Lévy, el primer experimento obtuvo mejores resultados. El mejor modelo con el primer experimento se obtuvo con la base de datos de 32 pasos Lévy, mientras que para el segundo experimento, se obtuvo con la base de 64 pasos Lévy.

El mejor modelo de clasificación propuesto alcanzó una precisión del 85.3 %, lo que sugiere que la imagen extraída mediante el muestreo multiescala captura efectivamente la modalidad de los problemas de optimización, y que la red neuronal elegida generaliza exitosamente esta información. Esto se evidencia también en el tercer experimento, donde las imágenes de la base de datos original obtienen mejores resultados que las imágenes muestreadas uniformemente, indicando que el modelo extrae información relevante del muestreo para la clasificación. Estos resultados podrían mejorarse ajustando variables de entrenamiento como el número de épocas y el tamaño de lote, o experimentando con otros modelos de clasificación. Para estudios futuros, sería útil entrenar una CNN con una base de datos de problemas de optimización muestreados uniformemente para comparar su desempeño.

MUSSA está diseñado para tratar de obtener una muestra menos sesgada, generando una imagen que encapsula las propiedades subyacentes del paisaje de optimización. En este estudio, se implementa un enfoque de muestreo multiescala que utiliza una caminata que sigue una ley de potencia en la escala gruesa y algoritmos de búsqueda local en la escala fina. Este método representa una propuesta inicial de muestreo multiescala, abierta a exploraciones futuras que examinen alternativas para mejorar la cobertura y precisión a diferentes escalas. Adicionalmente, es una propuesta innovadora como algoritmo de reconocimiento de patrones como forma de entender, en este caso, los problemas de optimización.

La red neuronal utilizada en este estudio empleó transfer learning para aprovechar la arquitectura y el conocimiento adquirido en tareas de clasificación previas. Sin embargo, este enfoque también podría introducir sesgos derivados del modelo original. Por lo tanto, el mecanismo de clasificación puede beneficiarse de una adaptación que explore herramientas alternativas, como distintos modelos CNN preentrenados, arquitecturas personalizadas u otros métodos de clasificación que se ajusten mejor a las características específicas del problema de optimización.

El problema de entender los problemas de optimización podría tratarse como un problema de reconocimiento de patrones.

Conclusiones

Se propone un método de clasificación que se segmenta en un algoritmo de muestreo multiescala y un método de aprendizaje automático para categorizar problemas de optimización según su modalidad. Los resultados indican que la propuesta captura eficazmente la característica de modalidad de los paisajes de prueba.

Los resultados de los primeros dos experimentos indican que, aunque la exactitud en validación mejora con más pasos Lévy, el rendimiento en pruebas disminuye a partir de ciertos umbrales. El mejor rendimiento se logró con 32 pasos Lévy en el primer experimento y 64 pasos Lévy en el segundo, lo que sugiere que un equilibrio entre el número de pasos y la generalización del modelo es crucial.

Del tercer experimento se observó que las imágenes muestreadas con MUSSA superaron a las muestreadas uniformemente en términos de la exactitud inyectándolas al mejor modelo de clasificación obtenido, lo que resalta la importancia del método de muestreo. En trabajos futuros este experimento se podría mejorar con el entrenamiento de redes neuronales con conjuntos de datos muestreados uniformemente para una comparación más robusta.

El mejor modelo identificado durante las pruebas mostró un rendimiento interesante sobre un grupo representativo de problemas de optimización (con exactitud del 85.3 %). Sin embargo, este método tiene una aplicación parcial en el mundo real, ya que solo aborda problemas de optimización bidimensionales. Así mismo, el mejor modelo se logró utilizando una caminata que sigue una ley de potencia que emplea una muestra pequeña del paisaje de optimización. Este enfoque ayudaría a evitar la sobreexplotación del paisaje de optimización con fines de clasificación. También es importante destacar que la clasificación del paisaje no se realizó de manera tradicional, como en el ELA. En su lugar, el estudio se centró en extraer propiedades específicas de los problemas de optimización, de manera similar a la metodología utilizada en FLA.

4.4. Aportes originales

- Se planteó un algoritmo de muestro multiescala titulado Algoritmo de muestreo exploratorio multiescala (Multiscale Scouting Samplig Algorithm - MUSSA). Este algoritmo y en particular esta forma de muestrear, según el conocimiento de los investigadores, no ha sido planteada antes en la literatura.

- A partir del trabajo realizado, se elaboró un artículo titulado Optimization Landscape Classification with Multi-Scale Sampling and Deep Neural Networks, el cual fue presentado en la conferencia: "The Sixth International Conference On Intelligent Computing in Data Sciences (ICDS)" . Tras un proceso de revisión por pares, el artículo fue aceptado para presentación oral en el evento y también para su inclusión en las memorias de la conferencia, el cual será un libro con ISBN 979-8-3503-5120-0.

- EL trabajo ayudó a la visualización de otras ideas de investigación materializadas en proyectos de grado de pregrado orientadas a fortalecer el objetivo 1. Particularmente, el tema de estudio fue la creación de nuevas funciones de optimización a partir de modelos de aprendizaje automático, el primer trabajo se basó en un método para la síntesis de paisajes de optimización bidimensionales basado en redes adversarias generativas [107] y el segundo fue un método de síntesis de funciones de prueba para optimizadores basado en autocodificadores variacionales y regresión simbólica [108].

- Se colaboró en la elaboración del artículo titulado Optimization test function synthesis with generative adversarial networks and adaptive neuro-fuzzy systems en la revista Information Sciences [109], realizando ELA a las funciones generadas. El artículo se relaciona con el cumplimiento del objetivo 1, ya que estudia la generación de nuevas funciones de prueba para su uso en el problema de selección de algoritmos de optimización. En este artículo se obtienen las representaciones matemáticas de algunas de las funciones generadas con la DCGAN, y por medio de estas representaciones, en el apéndice de este trabajo se realiza la clasificación de las funciones generadas (apéndice A.1) apreciando el funcionamiento del mecanismo de clasificación sobre funciones totalmente nuevas no antes vistas en la literatura. Así mismo, el mecanismo de clasificación se probó con las funciones generadas por el método basado en autocodificadores variacionales (apéndice A.2).

4.5. Trabajo futuro

Como trabajo futuro, sería valioso expandir este método a dimensiones más altas, por ejemplo, aplicando una transformación de la muestra en una imagen que preserve la mayoría de las características del problema de optimización de alta dimensionalidad. Esto también

podría ayudar a abordar el desafío subsiguiente de seleccionar un algoritmo de optimización adecuado. Además, para avanzar en los resultados obtenidos en este estudio, futuras investigaciones podrían explorar el método descrito en [43], donde se propone un enfoque de AutoML para predecir el rendimiento de un algoritmo frente a un nuevo problema de optimización.

La clasificación de los paisajes de optimización, basada en la definición de Kanemitsu [2], sigue siendo un campo sujeto a interpretación y requiere más investigación. Futuras investigaciones podrían utilizar esta definición para segmentar y clasificar problemas de optimización de manera más directa.

La implementación de MUSSA, con su enfoque de muestreo multiescala, es un primer paso hacia mejorar la representación de los paisajes de optimización. En trabajos futuros se podría explorar variantes del muestreo que incorporen memoria para evitar redundancias, lo que podría optimizar la recolección de muestras y mejorar la clasificación de características clave. Así mismo, el uso de transfer learning en la red neuronal permitió aprovechar arquitecturas previas, aunque pudo introducir sesgos.en trabajos futuros se podría adaptar la estrategia de clasificación considerando modelos CNN preentrenados alternativos o arquitecturas personalizadas para una mejor alineación con las características específicas del problema de optimización.

Bibliografía

[1] Ryan Dieter Lang and Andries Petrus Engelbrecht. An Exploratory Landscape Analysis-Based Benchmark Suite. *Algorithms 2021, Vol. 14, Page 78*, 14(3):78, 2 2021.

[2] Hideo KANEMITSU, Hideaki IMAI, and Masaaki MIYASKOSHI. Definitions and Properties of (Local) Minima and Multimodal Functions using Level Set for Continuous Optimization Problems. *IEICE Proceeding Series*, 2:94–97, 3 2014.

[3] Wikimedia Commons. Typical CNN architecture, 12 2015.

[4] Hadj Ahmed Bouarara. A survey of computational intelligence algorithms and their applications. *Handbook of Research on Soft Computing and Nature-Inspired Algorithms*, pages 133–176, 3 2017.

[5] Michel Gendreau and Jean-Yves Potvin. *Handbook of Metaheuristics*, volume 272 of *International Series in Operations Research & Management Science*. Springer International Publishing, Cham, 2019.

[6] Aaron Courville Heaton Jeff Ian Goodfellow, Yoshua Bengio. *Deep learning*, volume 19. MIT Press, 2018.

[7] Hossin M and Sulaiman M.N. A Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process*, 5(2):01–11, 3 2015.

[8] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 1960.

[9] Zhilu Zhang and Mert R. Sabuncu. Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels. *Advances in Neural Information Processing Systems*, 2018-December:8778–8788, 5 2018.

[10] Jon T. Selvik and Eirik B. Abrahamsen. On the meaning of accuracy and precision in a risk analysis context. *http://dx.doi.org/10.1177/1748006X16686897*, 231(2):91–100, 1 2017.

[11] minimize(method=Powell) SciPy v1.14.1 Manual.

[12] minimize(method=Nelder-Mead) SciPy v1.14.1 Manual.

[13] M. S. Bazaraa, John J. Jarvis, and Hanif D. Sherali. *Linear programming and network flows*. 2011.

[14] David H Wolpert and William G Macready. No Free Lunch Theorems for Optimization. Technical Report 1, 1997.

[15] Luis R. Izquierdo, José Manuel Galán Ordax, José I. Santos, and Ricardo Del Olmo Martínez. Modelado de sistemas complejos mediante simulación basada en agentes y mediante dinámica de sistemas. *Empiria. Revista de metodología de ciencias sociales*, 0(16):85, 10 2008.

[16] John R. Rice. The Algorithm Selection Problem. *Advances in Computers*, 15(C):65–118, 1 1976.

[17] Ryan Dieter Lang. *Landscape analysis-based automated algorithm selection*. PhD thesis, Stellenbosch University, 2024.

[18] Thomas Back, David B. Fogel, and Zbigniew Michalewicz. *Handbook of Evolutionary Computation*. CRC Press, 1 1997.

[19] Kate A. Smith-Miles. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys*, 41(1), 12 2008.

[20] David B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence | IEEE eBooks | IEEE Xplore*. 2006.

[21] Katherine Mary Malan. A Survey of Advances in Landscape Analysis for Optimisation. *Algorithms 2021, Vol. 14, Page 40*, 14(2):40, 1 2021.

[22] G. M. Ostrovsky, Ye M. Mikhailova, and T. A. Berezhinsky. Optimization of large-scale complex systems. *International Journal of Systems Science*, 17(8):1121–1132, 2017.

[23] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. Exploratory landscape analysis. In *Genetic and Evolutionary Computation Conference, GECCO'11*, pages 829–836, New York, New York, USA, 2011. ACM Press.

[24] Pascal Kerschke and Heike Trautmann. Automated Algorithm Selection on Continuous Black-Box Problems by Combining Exploratory Landscape Analysis and Machine Learning. *Evolutionary Computation*, 27(1):99–127, 3 2019.

[25] Mario A. Muñoz, Michael Kirley, and Saman K. Halgamuge. Exploratory landscape analysis of continuous space optimization problems using information content. *IEEE Transactions on Evolutionary Computation*, 19(1):74–87, 2 2015.

[26] Kent C.B. Steer, Andrew Wirth, and Saman K. Halgamuge. Information theoretic classification of problems for metaheuristics. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5361 LNAI, pages 319–328, 2008.

[27] V. K. Vassilev, T. C. Fogarty, and J. F. Miller. Information characteristics and the structure of landscapes. *Evolutionary computation*, 8(1):31–60, 2000.

[28] Mario Andrés Muñoz, Michael Kirley, and Kate Smith-Miles. Analyzing randomness effects on the reliability of exploratory landscape analysis. *Natural Computing*, 21(2):131–154, 6 2022.

[29] Bas van Stein, Fu Xing Long, Moritz Frenzel, Peter Krause, Markus Gitterle, and Thomas Bäck. DoE2Vec: Deep-learning Based Features for Exploratory Landscape Analysis. *GECCO 2023 Companion - Proceedings of the 2023 Genetic and Evolutionary Computation Conference Companion*, pages 515–518, 3 2023.

[30] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, and Jonathan Taylor. An Introduction to Statistical Learning. 2023.

[31] Irene Moser and Marius Gheorghita. Combining search space diagnostics and optimisation. *2012 IEEE Congress on Evolutionary Computation, CEC 2012*, 2012.

[32] Anja Jankovic and Carola Doerr. Adaptive landscape analysis. *GECCO 2019 Companion - Proceedings of the 2019 Genetic and Evolutionary Computation Conference Companion*, pages 2032–2035, 7 2019.

[33] Ye Tian, Shichen Peng, Xingyi Zhang, Tobias Rodemann, Kay Chen Tan, and Yaochu Jin. A Recommender System for Metaheuristic Algorithms for Continuous Optimization Based on Deep Recurrent Neural Networks. *IEEE Transactions on Artificial Intelligence*, 1(1):5–18, 8 2020.

[34] William Gilpin. Chaos as an interpretable benchmark for forecasting and data-driven modelling. 10 2021.

[35] Raphael Patrick Prager, Moritz Vinzent Seiler, Heike Trautmann, and Pascal Kerschke. Automated Algorithm Selection inăSingle-Objective Continuous Optimization: A Comparative Study ofăDeep Learning andăLandscape Analysis Methods. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 13398 LNCS:3–17, 2022.

[36] Wenfei Liu, Jingcheng Wei, and Qingmin Meng. Comparisions on KNN, SVM, BP and the CNN for Handwritten Digit Recognition. *Proceedings of 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications, AEECA 2020*, pages 587–590, 8 2020.

[37] Abdurrahim Yilmaz, Ali Anil Demircali, Sena Kocaman, and Huseyin Uvet. Comparison of Deep Learning and Traditional Machine Learning Techniques for Classification of Pap Smear Images. 9 2020.

[38] Pin Wang, En Fan, and Peng Wang. Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. *Pattern Recognition Letters*, 141:61–67, 1 2021.

[39] Ragav Venkatesan and Baoxin Li. *Convolutional Neural Networks in Visual Computing: A Concise Guide*. CRC Press, 10 2017.

[40] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. 2004.

[41] Vincent Hénaux, Adrien Goëffon, and Frédéric Saubion. Evolving Fitness Landscapes with Complementary Fitness Functions. *Artificial Evolution*, pages 110–120, 10 2019.

[42] Peter F. Stadler. Fitness landscapes. *Biological Evolution and Statistical Physics*, pages 183–204, 10 2002.

[43] Mario A. Muñoz, Michael Kirley, and Saman K. Halgamuge. A meta-learning prediction model of algorithm performance for continuous optimization problems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7491 LNCS(PART 1):226–235, 2012.

[44] G. H. Weiss. Aspects and applications of the random walk. *Journal of Statistical Physics*, 79(1):497–500, 4 1995.

[45] George C.. Montgomery, Douglas C.; Runger. Applied Statistics and Probability for Engineers. 2011.

[46] Momin Jamil and Xin-She Yang. A Literature Survey of Benchmark Functions For Global Optimization Problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150–194, 8 2013.

[47] Feng Zou, Debao Chen, Hui Liu, Siyu Cao, Xuying Ji, and Yan Zhang. A survey of fitness landscape analysis for optimization. *Neurocomputing*, 503:129–139, 9 2022.

[48] Jeffrey Horn and David E. Goldberg. Genetic Algorithm Difficulty and the Modality of Fitness Landscapes. 3:243–269, 1 1995.

[49] Erik Pitzer, Michael Affenzeller, and Andreas Beham. A closer look down the basins of attraction. *2010 UK Workshop on Computational Intelligence, UKCI 2010*, 2010.

[50] Christian Hanster and Pascal Kerschke. Flaccogui: Exploratory landscape analysis for everyone. In *GECCO 2017 - Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1215–1222. Association for Computing Machinery, Inc, 7 2017.

[51] Pascal Kerschke, Mike Preuss, Carlos Hernández, Oliver Schütze, Jian Qiao Sun, Christian Grimme, Günter Rudolph, Bernd Bischl, and Heike Trautmann. Cell mapping techniques for exploratory landscape analysis. *Advances in Intelligent Systems and Computing*, 288:115–131, 2014.

[52] Katherine M. Malan and Andries P. Engelbrecht. Quantifying ruggedness of continuous landscapes using entropy. In *IEEE Congress on Evolutionary Computation, CEC 2009*, pages 1440–1447, 2009.

[53] Alroomi Website - Unconstrained.

[54] Index AMPGO 0.1.0 documentation.

[55] Micheal F. Shlesinger, George M. Zaslavsky, and Uriel Frisch. Lévy Flights and Related Topics in Physics. 450, 1995.

[56] Jean Philippe Bouchaud and Antoine Georges. Anomalous diffusion in disordered media: Statistical mechanisms, models and physical applications. *Physics Reports*, 195(4-5):127–293, 11 1990.

[57] Masato S. Abe. Functional advantages of Lévy walks emerging near a critical point. *Proceedings of the National Academy of Sciences of the United States of America*, 117(39):24336–24344, 9 2020.

[58] Rosario N. Mantegna and H. Eugene Stanley. Stochastic Process with Ultraslow Convergence to a Gaussian: The Truncated Lévy Flight. *Physical Review Letters*, 73(22):2946–2949, 1994.

[59] G. M. Viswanathan, Sergey V. Buldyrev, Shlomo Havlin, M. G.E. Da Luz, E. P. Raposo, and H. Eugene Stanley. Optimizing the success of random searches. *Nature 1999 401:6756*, 401(6756):911–914, 10 1999.

[60] Ralf Metzler and Joseph Klafter. The restaurant at the end of the random walk: Recent developments in the description of anomalous transport by fractional dynamics. *Journal of Physics A*, 2004.

[61] Albert-László Barabási and Réka Albert. Emergence of Scaling in Random Networks. *Mat. Res. Soc. Symp. Proc*, 74:677, 1995.

[62] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313, 1 1965.

[63] M. J. D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155–162, 1 1964.

[64] S Kirkpatrick, ; C D Gelatt, and ; M P Vecchi. Optimization by Simulated Annealing. *New Series*, 220(4598):671–680, 1983.

[65] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):6999–7019, 12 2022.

[66] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1):106, 1 1962.

[67] Douglas M. Hawkins. The Problem of Overfitting. *Journal of Chemical Information and Computer Sciences*, 44(1):1–12, 1 2004.

[68] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 4 1980.

[69] Siddharth Sharma, Simone Sharma, and Anidhya Athaiya. Activation Functions in Neural Networks. *International Journal of Engineering Applied Sciences and Technology*, 04(12):310–316, 5 2020.

[70] Diederik P. Kingma and Jimmy Lei Ba. Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 12 2014.

[71] Houda Bichri, Adil Chergui, and Mustapha Hain. Image Classification with Transfer Learning Using a Custom Dataset: Comparative Study. *Procedia Computer Science*, 220:48–54, 1 2023.

[72] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv*, 4 2017.

[73] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 1 2018.

[74] Davide Chicco and Giuseppe Jurman. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21(1):1–13, 1 2020.

[75] C. Ferri, J. Hernández-Orallo, and R. Modroiu. An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, 30(1):27–38, 1 2009.

[76] Sylvain Arlot and Alain Celisse. A survey of cross-validation procedures for model selection. *https://doi.org/10.1214/09-SS054*, 4(none):40–79, 1 2010.

[77] Rafael C. Gonzalez and Richard E. Woods. Digital Image Processing, Global Edition. *Person Education*, pages 19–44, 2018.

[78] Karl Pearson. III. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London. (A.)*, 185:71–110, 12 1894.

[79] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Is a correction for chance necessary? *ACM International Conference Proceeding Series*, 382, 2009.

[80] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 4 2004.

[81] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Part F128815:847–855, 8 2012.

[82] Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. An Efficient Approach for Assessing Hyperparameter Importance, 1 2014.

[83] James Bergstra, James Bergstra@umontreal Ca, and Yoshua Bengio@umontreal Ca. Random Search for Hyper-Parameter Optimization Yoshua Bengio. *Journal of Machine Learning Research*, 13:281–305, 2012.

[84] Daniel Horn and Bernd Bischl. Multi-objective parameter configuration of machine learning algorithms using model-based optimization. *2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016*, 2 2017.

[85] Renan Netto, Sheiny Fabre, Tiago Augusto Fontana, Vinicius Livramento, Laercio L. Pilla, Laleh Behjat, and Jose Luis Guntzel. Algorithm Selection Framework for Legalization Using Deep Convolutional Neural Networks and Transfer Learning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(5):1481–1494, 5 2022.

[86] Itai Dagan, Roman Vainshtein, Gilad Katz, and Lior Rokach. Automated algorithm selection using meta-learning and pre-trained deep convolution neural networks. *Information Fusion*, 105:102210, 5 2024.

[87] S ; Xu, W ; Liu, C ; Wu, J Li, Siyi Xu, Wenwen Liu, Chengpei Wu, and Junli Li. CNN-HT: A Two-Stage Algorithm Selection Framework. *Entropy 2024, Vol. 26, Page 262*, 26(3):262, 3 2024.

[88] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, Dinani Amorim, and Amorim Fernández-Delgado. Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? *Journal of Machine Learning Research*, 15:3133–3181, 2014.

[89] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep Double Descent: Where Bigger Models and More Data Hurt. *8th International Conference on Learning Representations, ICLR 2020*, 12 2019.

[90] Yaxin Li, Jing Liang, Kunjie Yu, Ke Chen, Yinan Guo, Caitong Yue, and Leiyu Zhang. Adaptive local landscape feature vector for problem classification and algorithm selection. *Applied Soft Computing*, 131:109751, 12 2022.

[91] Moritz Vinzent Seiler, Pascal Kerschke, and Heike Trautmann. Deep-ELA: Deep Exploratory Landscape Analysis with Self-Supervised Pretrained Transformers for Single- and Multi-Objective Continuous Optimization Problems. 1 2024.

[92] Yaxin Li, Jing Liang, Kunjie Yu, Caitong Yue, and Yingjie Zhang. Keenness for characterizing continuous optimization problems and predicting differential evolution algorithm performance. *Complex and Intelligent Systems*, 9(5):5251–5266, 10 2023.

[93] Kangjing Li, Saber Elsayed, Ruhul Sarker, and Daryl Essam. Multiple landscape measure-based approach for dynamic optimization problems. *Swarm and Evolutionary Computation*, 88:101578, 7 2024.

[94] Gaper Petelin, Gjorgjina Cenikj, and Tome Eftimov. TinyTLA: Topological landscape analysis for optimization problem classification in a limited sample setting. *Swarm and Evolutionary Computation*, 84:101448, 2 2024.

[95] Vojtech Uher and Pavel Kromer. Impact of Different Discrete Sampling Strategies on Fitness Landscape Analysis Based on Histograms. *ACM International Conference Proceeding Series*, 9, 12 2023.

[96] Jakob Bossek. Smoof: Single-and Multi-Objective Optimization Test Functions. *The R Journal*, 9, 2017.

[97] GitHub - ciren/benchmarks: A collection of n-dimensional functions.

[98] Zhen Zhang, Kuo Yang, Jinwu Qian, and Lunwei Zhang. Real-time surface EMG pattern recognition for hand gestures based on an artificial neural network. *Sensors (Switzerland)*, 19(14), 7 2019.

[99] Yi Zhao, Tongfeng Weng, and Defeng Huang. Lévy walk in complex networks: An efficient way of mobility. *Physica A: Statistical Mechanics and its Applications*, 396:212–223, 2 2014.

[100] Xin She Yang and Suash Deb. Multiobjective cuckoo search for design optimization. *Computers & Operations Research*, 40(6):1616–1624, 6 2013.

[101] Hemanth. levy(n,m,beta), 2019.

[102] Jorge; Nocedal and Stephen J. Wrigh. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006.

[103] Saket Gupta, Narendra Kumar, Laxmi Srivastava, Hasmat Malik, Alberto Pliego Marugán, and Fausto Pedro García Márquez. A Hybrid JayaPowells Pattern Search Algorithm for Multi-Objective Optimal Power Flow Incorporating Distributed Generation. *Energies 2021, Vol. 14, Page 2831*, 14(10):2831, 5 2021.

[104] M. D. Buhmann. Michael J.D. Powells work in approximation theory and optimisation. *Journal of Approximation Theory*, 238:3–25, 2 2019.

[105] Marco A. Luersen and Rodolphe Le Riche. Globalized NelderMead method for engineering optimization. *Computers & Structures*, 82(23-26):2251–2260, 9 2004.

[106] Nitish Shirish Keskar, Jorge Nocedal, Ping Tak Peter Tang, Dheevatsa Mudigere, and Mikhail Smelyanskiy. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 9 2016.

[107] Mariana Medina Muñoz. Método para la síntesis de paisajes de optimización bidimensionales basado en redes adversarias generativas, 2024.

[108] Juan Pablo López Guevara. Método de síntesis de funciones de prueba para optimizadores basado en autocodificadores variacionales y regresión simbólica, 2024.

[109] Miguel Melgarejo, Mariana Medina, Juan Lopez, and Angie Rodriguez. Optimization test function synthesis with generative adversarial networks and adaptive neuro-fuzzy systems. *Information Sciences*, 686:121371, 1 2024.