UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

DISEÑO AUTOMÁTICO DE REDES NEURONALES CONVOLUCIONALES INSPIRADAS EN ARQUITECTURAS BIOLÓGICAS RETINOTÓPICAS UTILIZANDO ALGORITMOS GENÉTICOS MULTI-CROMOSÓMICOS

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

CHRISTOFER ALEXIS CID LARA

PROFESOR GUÍA:
CLAUDIO PÉREZ FLORES

MIEMBROS DE LA COMISIÓN:
CÉSAR AZURDIA MEZA
FELIPE BRAVO MÁRQUEZ

Este trabajo fue parcialmente financiado por el proyecto FONDECYT 1231675 de ANID, además del financiamiento basal ANID, AMTC AFB220002 e IMPACT FB210024, y por el Departamento de Ingeniería Eléctrica, Universidad de Chile

SANTIAGO DE CHILE
2024

RESUMEN DE LA TESIS PARA OPTAR
AL GRADO DE MAGÍSTER EN CIENCIAS
DE LA INGENIERÍA, MENCIÓN ELÉCTRICA Y
MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO
POR: CHRISTOFER ALEXIS CID LARA
FECHA: 2024
PROF. GUÍA: CLAUDIO PÉREZ F.

DISEÑO AUTOMÁTICO DE REDES NEURONALES CONVOLUCIONALES INSPIRADAS EN ARQUITECTURAS BIOLÓGICAS RETINOTÓPICAS UTILIZANDO ALGORITMOS GENÉTICOS MULTI-CROMOSÓMICOS

Este trabajo se centra en el diseño automático de modelos de redes neuronales convolucionales (CNN) inspirados en arquitecturas biológicas retinotópicas, utilizando algoritmos genéticos multi-cromosómicos (MCGA). La investigación parte de la hipótesis de que la neuroevolución podría generar arquitecturas óptimas biológicamente inspiradas y que MCGA serían ideal para representar arquitecturas con diversas estructuras. Además, se propone que la incorporación de elementos bio-inspirados como multitarea, atención y parches neuronales mejoraría el rendimiento de las CNNs. Los resultados confirman las hipótesis iniciales, cumpliendo el objetivo general de diseñar CNNs automáticamente mediante MCGA para reconocimiento de patrones. En CIFAR-10 se obtiene un 4.61 % de error de clasificación sin utilizar mejoras bio-inspiradas y 4.25 % al incorporarlas, obteniendo mejor desempeño que modelos de estado del arte en espacios de búsqueda similares como CNN-GA e igual desempeño y menor tiempo de búsqueda que HGAPSO. Además, MCGA cumple con encontrar mejores modelos que Random Search (RS) en el mismo espacio de búsqueda. El estudio establece un precedente para futuras investigaciones en la creación de modelos de redes neuronales inspirados biológicamente, y sugiere diversas áreas para continuar explorando y mejorando estos enfoques.

Cualquier tecnología suficientemente avanzada
es indistinguible de la magia.

Arthur C. Clarke

Agradecimientos

Agradezco a todas las personas que me han apoyado durante mi carrera profesional en el ámbito de estudios y personal, partiendo por mis hermanos Benjamín y Jaime y mis padres, Maribel y Jaime, quienes me han ayudado a ser quién soy. A Javiera que me ha apoyado desde que nos conocemos, y me ha acompañado en los momentos más estresantes de estos años.

Quiero agradecer particularmente a quienes me acompañaron durante el 2023, donde tuve que tomar decisiones importantes. A Camilo por siempre escuchar mis pensamientos, a Andrés por esas discusiones abstractas durante nuestros entrenamientos, a Giovanni por las ideas y conversaciones de AI, a Vicente por las partidas de AoE2 y los chistes, a Almendra y Blanca por sus siempre cariñosos saludos y conversaciones. Agradezco también a todos los nares, a mis colegas del laboratorio de redes fotónicas y del laboratorio 5G, y en general, a todos mis amigos.

También quiero reconocer a mi profesor guía Claudio Pérez, quien me recibió como alumno tesista y me ha acompañado durante todos estos años con incontables reuniones y revisiones, lo que me ha permitido aprender cada vez más en lo que me gusta.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Motivación

Desde hace milenios, las personas han intentado replicar acciones humanas o animales mediante objetos inertes; en esencia, dar vida a objetos simulando la biología. Los ejemplos más tempranos corresponden a los escritos sobre autómatas en China en el siglo 3 antes de la era común, en la dinastía Han, donde se construyó una orquesta completa de autómatas para el emperador de la época [1].

Contemporáneamente en Europa, en la antigua Grecia, Arquitas de Tarento construyó un modelo de ave, intentando replicar el vuelo de esta mediante un mecanismo de vapor [2, 3]. Diferentes fuentes y escritos indican la existencia de diversos mecanismos automáticos basados en agua, vapor y diferencias de pesos, principalmente en China hasta aproximadamente el siglo 13 [1].

Por otra parte, en la edad de oro del Islam, surgió al-Jazarī, matemático e ingeniero del siglo 13, a quien muchos consideran el padre de la cibernética y robótica debido a la gran cantidad de autómatas y diseños de cibernética basados en energía hidráulica que documentó [4]. Se han preservado muchos de sus escritos y libros, y sus elaborados e ingeniosos diseños han sido replicados incluso en la actualidad [5]. Una de las principales contribuciones que se le atribuyen es la creación de uno de los primeros autómatas programables: una orquesta de 4 músicos a la que se le podían modificar los ritmos mediante clavijas [4, 6].

Uno de los aspectos más importantes de al-Jazarī fue la filosofía pragmática que lo llevó a crear los mecanismos que documentó, siguiendo la idea de mejorar la vida humana mediante máquinas que cumplieran con las reglas de "balance" [4].

Parte de esta filosofía ha perdurado en el tiempo y ha sido ampliamente analizada por figuras modernas de la ciencia ficción como Isaac Asimov o Arthur Clarke, llevándola a niveles donde las máquinas, inteligencias artificiales y conciencias artificiales son indistinguibles de los humanos y las problemáticas que esto podría generar [7, 8], tal como se vislumbra actualmente con la humanización de los *Large Language Models* [9].

La filosofía de ayudar a las personas mediante máquinas que puedan reemplazar tareas complejas persiste en la actualidad, siendo la Inteligencia Artificial (IA) una de las principales

herramientas utilizadas.

Volviendo a la idea de replicación de acciones humanas, la IA nace directamente de la idea de simular la inteligencia humana y las funciones cognitivas necesarias para resolver problemas y aprender. Considerando que muchas de las funciones cognitivas de los humanos provienen del hecho de ser mamíferos, en consecuencia, se puede establecer que, en realidad, la inteligencia artificial también podría estar intentando replicar a los mamíferos o animales.

Existen diversos paradigmas filosóficos y teóricos desde los que se pueden clasificar las IA. Uno de ellos los separa en tres categorías: *Artificial Narrow Intelligence* (ANI), *Artificial General Intelligence* (AGI) y *Artificial Super Intelligence* (ASI), donde la primera corresponde a IA que pueden solucionar problemas específicos, como identificar a una persona en una foto o detectar palabras en un texto; la segunda, a la inteligencia o resolución de tareas se ejecuta a nivel humano, tal como la interpretación de tonos de voz, emociones en texto o conducir un auto; y las últimas, definidas por habilidades sobrehumanas, que por consenso general aún no existen o están en las etapas iniciales de formulación.

Desde el punto de vista más técnico, se distinguen principalmente los conceptos de *Artificial Intelligence* (AI), *Machine Learning* (ML), *Bio-inspired Algorithms* (BioA), *Artificial Neural Networks* (ANN) y *Deep Learning* (DL), explicado gráficamente por el diagrama de la figura 1.1.

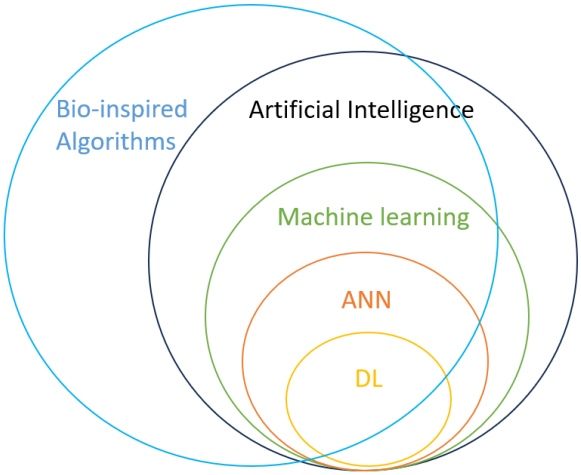


Figura 1.1: Diagrama de Venn de los conceptos de AI.

AI es un tema de investigación que busca crear máquinas capaces de realizar tareas que requieren inteligencia humana, como el razonamiento, el aprendizaje y la percepción, entre otros. Dentro de este campo se encuentra el ML, un subconjunto de temas que se enfoca en el desarrollo de algoritmos para que las máquinas aprendan de los datos. Lo relevante aquí es que este aprendizaje no está programado directamente, sino que puede ser entrenado. La base de esto normalmente se logra analizando patrones a partir de grandes conjuntos de datos, lo que permite a las máquinas distinguir, predecir o tomar decisiones basadas en la información aprendida.

Dentro del ML, una de las áreas más relevantes en la actualidad son las ANN, redes que

imitan la estructura y funcionamiento del cerebro humano, dedicadas a procesar información mediante capas interconectadas que simulan las redes neuronales biológicas. Por definición, las ANN son un tipo de BioA, ya que son un método de computación que se inspira en procesos biológicos. Sin embargo, los BioA son una categoría que no solo se encuentra dentro del paradigma de AI, sino que también pueden aplicarse a otras tareas.

Cuando los procesos son de alta complejidad, se hace necesaria una mayor complejidad de interconexiones dentro de las ANN. Este incremento en complejidad es lo que define al DL, donde se utilizan múltiples capas de redes neuronales para generar profundidad, permitiendo que las redes aprendan representaciones de los datos en múltiples niveles de abstracción, de manera similar al cerebro humano.

El DL, gracias a los avances en computación de la actualidad, se utiliza ampliamente para resolver satisfactoriamente problemas de diversa índole, como el reconocimiento biométrico y de patrones, la detección y diagnóstico de enfermedades, la traducción automática, el control de vehículos autónomos, la síntesis de imágenes y las predicciones meteorológicas, entre otros.

Una de las principales herramientas utilizadas en la actualidad, por su versatilidad para resolver problemas relacionados con imágenes, son las *Convolutional Neural Networks* o redes neuronales convolucionales (CNN). Las CNNs están basadas en las estructuras cerebrales y el procesamiento visual de mamíferos [10, 11]. Las redes pioneras de este tipo son el Neocognitron (1979) [10] y LeNet (1998) [11], siendo esta última conocida como una de las primeras CNNs y fue utilizada para la tarea de reconocer dígitos en imágenes.

A lo largo de los años, han surgido incontables diseños de CNNs, conocidos como arquitecturas, para resolver diferentes tipos de problemas. Dentro de las más reconocidas en el área de las CNN se encuentran AlexNet [12], GoogleNet o InceptionNet [13], ResNet [14], DenseNet [15], EfficientNet [16], entre otras.

La capacidad de diseñar arquitecturas de CNNs que sean eficaces y eficientes para las distintas tareas es un desafío constante en el área. Superar este desafío requiere de muchas horas humanas, iteraciones y conocimiento experto. Por esta razón, en el último lustro ha aumentado considerablemente el interés por automatizar este procedimiento [17]. Este trabajo se enfoca en resolver esta problemática, relacionándose principalmente con la intersección de BioA y DL.

1.2. Problema

Abordar el problema de construir modelos óptimos y eficaces para tareas dadas, tales como el reconocimiento de patrones, el reconocimiento de imágenes, la clasificación de texto y el procesamiento de señales, requiere de conocimiento experto técnico y además consume cantidades significativas de tiempo humano en probar hiperparámetros y parámetros de la red que se desea construir. Esto a su vez puede implicar que se estén realizando procedimientos subóptimos.

En la actualidad, el diseño de arquitecturas óptimas/eficaces es una limitante para el despliegue de DL en áreas que podrían beneficiarse de esta tecnología, principalmente por la

necesidad de conocimiento experto para diseño.

La idea de automatizar este proceso es parte de un tema más amplio que busca automatizar completamente un proceso de ML, tarea denominada *Auto Machine Learning*, que incluye no solo la optimización de arquitecturas, sino también el preprocesamiento de los datos, la evaluación del modelo e incluso la interpretación de resultados.

Existen diversas propuestas para abordar este metaproblema de manera automática o semiautomática, en el que se busca que a partir de una tarea y un conjunto de datos dado, se encuentre el mejor modelo para resolver la tarea, sujeto a restricciones de costos (energía, tiempo, limitaciones de hardware, etc.).

Acotando únicamente al problema de automatizar el diseño de arquitecturas (de CNNs), éste se conoce como *Neural Architecture Search* (NAS) y se utilizan diferentes tipos de soluciones. Las más ampliamente utilizadas son los Métodos de Gradiente (GO), aprendizaje reforzado o *Reinforcement Learning* (RL), Optimización Bayesiana (BO) y algoritmos evolutivos (EA) [18, 19]. Estos últimos se relacionan de una manera biomimética a la evolución, realizando el mismo procedimiento de transitar, a través de generaciones, hacia arquitecturas neuronales óptimas. El concepto de que el procesamiento visual en mamíferos evolucionó a través de millones de años, para realizar tareas de reconocimiento de patrones de forma eficiente y efectiva, se aplica de una manera programática a evolucionar ANNs o CNNs, procedimiento conocido como **Neuroevolución**. Cabe destacar que el diseño automático de arquitecturas es un problema de optimización empírico, ya que no se asegura optimalidad global, principalemente debido a la incapacidad de computar y determinar exactamente la estructura del espacio de búsqueda, por lo que es un método heurístico.

Para abordar el problema de NAS mediante neuroevolución, existen diferentes técnicas, entre ellas, *grid search*, *random search*, evolución diferencial, *Particle Swarm Optimization*, algoritmos genéticos, algoritmos híbridos, entre otros.

Este trabajo se enfoca en la utilización de algoritmos genéticos para resolver el problema de NAS acotado a arquitecturas basadas en redes neuronales biológicas, donde además se utilizarán mecanismos bio-inspirados para mejorar el rendimiento del mejor modelo encontrado.

1.3. Hipótesis

- Se puede utilizar neuroevolución para crear arquitecturas biológicamente inspiradas con desempeño competitivo, en términos de error de clasificación, tiempo de búsqueda y cantidad de parámetros de la red.
- Una codificación multicromosómica es una opción funcional para representar arquitecturas con estructuras diversas, y utilizar un algoritmo genético con esta codificación permite realizar búsquedas en un espacio de búsqueda donde no es factible computar el desempeño de todas las arquitecturas posibles.
- La incorporación de elementos biológicamente inspirados, como bloques densos, entre-

namiento multitarea, modulos de atención y parches neuronales, a redes neuronales convolucionales permite mejorar su desempeño.

1.4. Objetivo general

Diseñar automáticamente modelos de redes neuronales convolucionales inspirados en arquitecturas biológicas retinotópicas mediante algoritmos genéticos multicromosómicos (MCGA) para el reconocimiento de patrones.

1.5. Objetivos específicos

- Diseñar arquitecturas de redes especializadas para mejorar la capacidad de reconocimiento de patrones mediante la aplicación de algoritmos genéticos (AG).
- Desarrollar modelos de CNNs inspiradas en arquitecturas biológicas retinotópicas para el reconocimiento de patrones en la base datos CIFAR-10.
- Diseñar mediante MCGA arquitecturas con desempeño competitivo, medido en términos de error de clasificación, tiempo de búsqueda y cantidad de parámetros de la red, respecto a espacios de búsqueda similares.
- Mejorar el desempeño global de modelos de CNNs mediante la incorporación de bloques densos, módulos de atención, entrenamiento multitarea y la simulación de parches neuronales biológicos, para el reconocimiento de patrones.
- Probar y comparar los modelos desarrollados mediante MCGA con respecto a una línea base de RS y con los del estado del arte en NAS.

1.6. Contribuciones

- Se propone un tipo de arquitectura de CNNs mediante analogías a los sistemas visuales mamíferos, compuesto de bloques densos, atención, aprendizaje multitarea y parches neuronales, para la tarea de búsqueda de arquitecturas en reconocimiento de imágenes.
- Se desarrolla un método novedoso multicromosómico para optimizar modelos de CNNs mediante algoritmos genéticos.
- En la base de datos CIFAR-10, las múltiples mejoras modulares bio-inspiradas para las arquitecturas propuestas demuestran ser útiles e incrementales para el desempeño de las CNN en la tarea de clasificación de imágenes, siendo competitivo el desempeño, tamaño y tiempo de búsqueda con respecto a propuestas del estado del arte que utilizan espacios de búsquedas similares.

1.7. Estructura de la tesis

La presente tesis se estructura de la siguiente forma. El Capítulo 1 introduce el trabajo, mediante la explicación de la motivación, el problema, las hipótesis planteadas y los objetivos

para su demostración, además se indican las contribuciones de la tesis. En el Capítulo 2 se expone el marco teórico, con los conceptos fundamentales de las redes neuronales biológicas, redes neuronales artificiales y redes neuronales convolucionales, también se presentan las ideas importantes de la neuroevolución y los algoritmos genéticos. En el Capítulo 3 se muestra el estado del arte relacionado al procesamiento visual, los métodos de búsqueda automática de arquitecturas de redes neuronales convolucionales, las mejoras bio-inspiradas de redes y las bases de datos utilizadas. En el Capítulo 4 se describe la metodología utilizada en el trabajo, que incluye, el diseño de la arquitectura base, el algoritmo genético multi-cromosómico, las diferentes mejoras bio-inspiradas y los experimentos. En el Capítulo 5 se presentan los resultados de todos los experimentos realizados, basado en la metodología explicada en el Capítulo 4, donde se incluyen gráficos de evolución, entrenamiento y tablas, además se realiza una comparación con el estado del arte y una discusión de los resultados obtenidos. Para finalizar, el Capitulo 6 exhibe las conclusiones del trabajo, relacionado a las hipótesis y objetivos planteados al inicio, además se incluyen propuestas de mejoras metodológicas y trabajo futuro.

Capítulo 2

Marco teórico

En este capítulo se introducen los conceptos básicos y necesarios para la comprensión de la tesis realizada. En particular, se entregan nociones referentes a las redes neuronales biológicas, el procesamiento visual humano y mamífero, retinotopía, además de los fundamentos de las redes neuronales artificiales, incluyendo, perceptrón, perceptrón multicapa, redes neuronales convolucionales, entrenamiento de redes neuronales artificiales, diseño y arquitecturas de redes neuronales convolucionales (LeNet-5, ResNet, DenseNet, etc.). Finalmente, se introducen los conceptos del diseño automático de redes neuronales, *Neural Architecture Search*, neuroevolución, algoritmos genéticos y mecanismos de redes neuronales bio-inspirados.

2.1. Redes neuronales biológicas y procesamiento visual

Las redes neuronales biológicas, particularmente aquellas involucradas en el procesamiento de la información visual, constituyen un sistema complejo y altamente especializado que ha evolucionado durante al menos 700 millones de años [20], desde la aparición de las primeras neuronas. Estas redes, ubicadas principalmente en los cerebros de humanos y otros mamíferos, están diseñadas para interpretar y responder eficientemente a estímulos visuales.

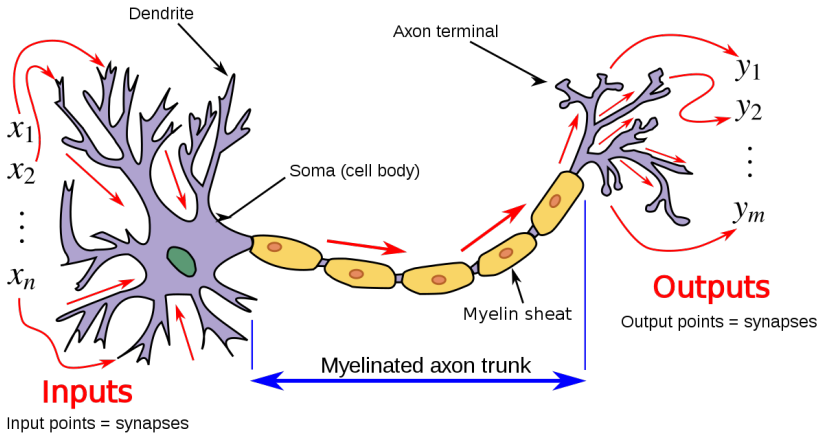


Figura 2.1: Diagrama de una neurona. La información se representa como entradas x_i que reciben las dendritas y salidas y_i por axones terminales. Obtenido de [21].

Las neuronas, células especializadas en la transmisión de información mediante señales electroquímicas, son los componentes principales de estas redes neuronales. El procesamiento de la información en el cerebro es jerárquico y paralelo a la vez, y se han identificado varias áreas específicas en el cerebro dedicadas a distintos aspectos del procesamiento visual [22, 23, 24, 25].

De acuerdo con el modelo más aceptado del procesamiento visual humano [26, 27, 28], la información procedente de la retina se transmite hacia el nervio óptico y el quiasma óptico, llegando finalmente a ambos núcleos geniculados laterales (LGN) en el tálamo, donde en humanos hay seis capas de neuronas (materia gris).

La información es recibida inicialmente por la corteza visual primaria (V1) y luego se divide en dos corrientes de procesamiento, la dorsal y la ventral, conocidas colectivamente como la vía de información visual o *Visual Information Pathway*.

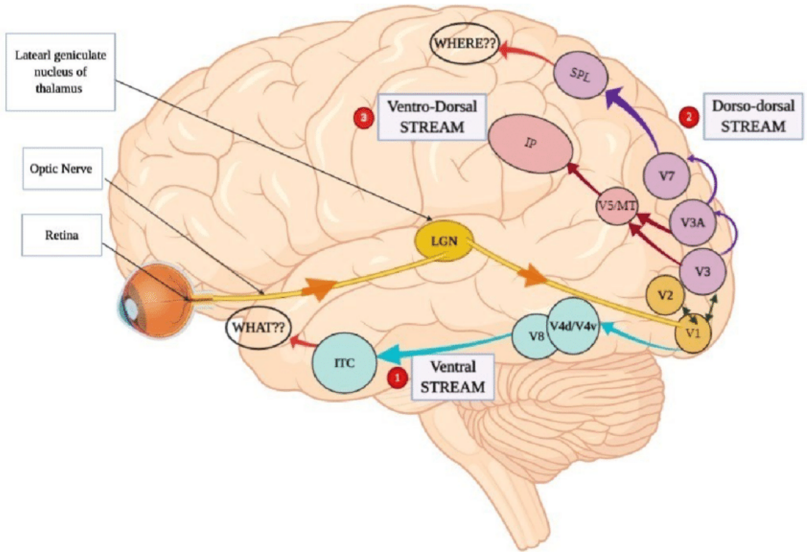


Figura 2.2: Diagrama de la vía de información visual. Se observa la vía de procesamiento dorsal (dorso-dorsal y ventro-dorsal) y la vía ventral. Obtenido de [29].

Se entiende que ambos caminos procesan información diferente y realizan tareas distintas en paralelo. La vía ventral, que comienza en V1, pasa por V2, continúa a través de V4 hasta la corteza temporal inferior (IT), es conocida como la "vía del qué", asociada con el reconocimiento de formas y la representación de objetos [26, 28, 29].

Por otro lado, la vía dorsal, que también comienza en V1 y pasa por V2, se dirige a través de las áreas dorsomedial (DM/V6), temporal medial (MT/V5) y V7 hasta la corteza parietal posterior, es conocida como la "vía del cómo/dónde", relacionada con el movimiento, la ubicación espacial de objetos y el control de brazos y ojos [26, 28, 29].

2.2. Conformación retinotópica de mamíferos

Uno de los aspectos más importantes del procesamiento visual es la retinotopía, que se refiere al mapeo de la entrada visual desde la retina a las neuronas, particularmente en la corteza visual del cerebro. Los primeros descubrimientos del mapeo de la retina al cerebro fueron identificadas independientemente por Tatsuji Inouye en Japón y Gordon Holmes en Inglaterra, donde observaron la correlación entre posición de heridas de guerra y pérdida de campo visual [25].

Los mapas retinotópicos están dinámicamente involucrados en la organización funcional del procesamiento visual, contribuyendo a la percepción de profundidad, color, movimiento y forma [30, 24]. Básicamente, la retinotopía conserva la distribución espacial de la información de la retina sobre la representación que se hace en la corteza cerebral, lo que significa que áreas cercanas en el campo visual, también se procesan de manera cercana en la corteza, lo que permite al cerebro interpretar información espacial y geométrica de forma óptima. Este fenómeno se puede observar en el mapeo angular y radial indicado por la figura 2.3.

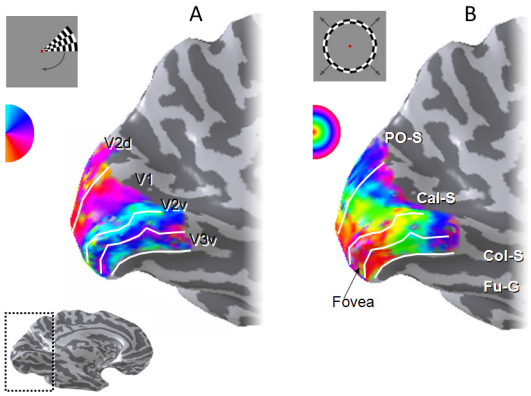


Figura 2.3: Diagrama de retinotopía humana y secciones de procesamiento visual. A. Mapeo angular a secciones ventrales. B. Mapeo radial a secciones ventrales. Obtenido de [23].

La retinotopía se encuentra presente en diferentes especies, particularmente mamíferos, y es crucial para entender cómo se procesa la información visual en el cerebro. Diferentes estudios realizados con imágenes cerebrales, utilizando resonancia magnética funcional, han permitido crear mapas retinotópicos cada vez más detallados, mejorando las propuestas de procesamiento de información visual sobre color, formas, movimiento y profundidad [22, 23, 24, 25].

2.3. Perceptrón

Los principios subyacentes al procesamiento biológico de la información visual han proporcionado históricamente valiosas perspectivas para el desarrollo de redes neuronales artificiales (ANNs). Uno de los primeros modelos en la representación de neuronas artificiales es el perceptrón.

El perceptrón se basa en un modelo simplificado de la neurona biológica, sin considerar aspectos temporales [31]. Su propósito es diferenciar entre dos clases a partir de una serie de entradas y funciona como un clasificador lineal. En términos de analogía con la neurona biológica, las entradas x representan las señales que llegan a las dendritas, cada una con una ponderación distinta; el soma actúa como un integrador o sumador de estas señales, y el axón funciona como la función de activación que determina si se genera o no una señal de salida. Esta analogía se puede observar en la figura 2.1.

Matemáticamente, el perceptrón se define como una función umbral de la forma:

$$\mathbf{y} = f(\mathbf{x}) = \theta(\mathbf{w} \cdot \mathbf{x} + b), \tag{2.1}$$

donde x corresponde a las entradas, w a los pesos de cada entrada, b a la variable de sesgo, θ a la función de activación e y a las salidas de la neurona.

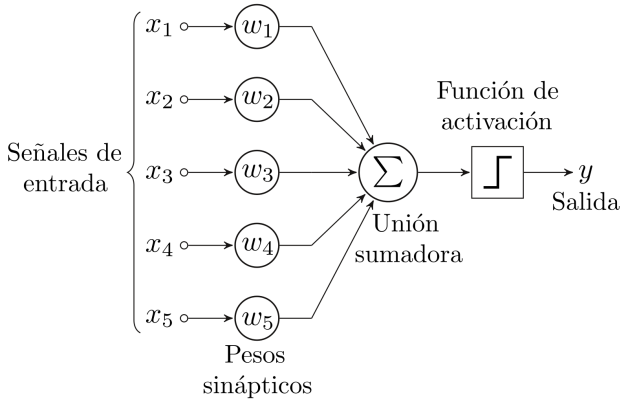


Figura 2.4: Representación de un perceptrón. Las señales de entradas x_i se ponderan por los pesos w_i y pasan por la función de activación, generando la salida y. Obtenido de [32].

A pesar de la simplicidad del perceptrón y su limitación a clasificaciones lineales, cuando se configuran en un formato de multicapa, conocido como perceptrón multicapa o *multilayer perceptron* (MLP), el modelo puede separar clases de manera no lineal. Es más, un MLP es un aproximador universal de cualquier función no lineal [33].

2.4. Redes neuronales artificiales

Las primeras redes neuronales artificiales se inspiraron en el perceptrón. El MLP surgió como solución a la limitación de clasificación lineal del perceptrón, al ser capaz de abordar problemas de clasificación no lineal. La organización de neuronas en múltiples capas da lugar a una red neuronal con profundidad, la cual está directamente relacionada con la capacidad de abstracción de la red [12]. En términos generales, cuantas más neuronas contiene una red y mayor es su profundidad, mayor es su capacidad de aprendizaje y, por ende, mejora su habilidad para clasificar.

La estructura típica de las neuronas artificiales se representa en la figura 2.5. De esta forma, la estructura incluye una capa de entrada, una o varias capas ocultas y una capa de salida.

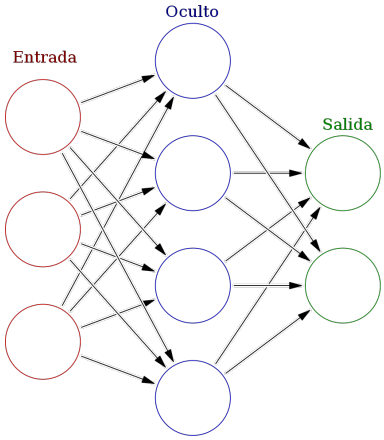


Figura 2.5: Representación de capas en una red neuronal. Clasificación en capa de entradas, ocultas y de salida. Obtenido de [34].

Aunque el MLP es un aproximador universal, esto no implica que la determinación de los pesos óptimos sea un proceso sencillo o factible. Durante el entrenamiento, estos modelos pueden quedar atrapados en óptimos locales. Además, la presencia de múltiples capas ocultas, especialmente en redes con gran profundidad, puede resultar en una pérdida de interpretabilidad, conduciendo a que estos modelos operen como *cajas negras* o también a sobre-ajustes, perdiendo la capacidad de generalizar el aprendizaje.

2.5. Redes neuronales convolucionales

Las redes neuronales convolucionales son un tipo de red de DL, comúnmente utilizada en tareas relacionadas a las imágenes. En la actualidad representan una importante herramienta para el ámbito de visión computacional, incluyendo reconocimiento y segmentación de imágenes, entre otros.

Las redes pioneras CNNs propuestas, fueron desarrolladas en los 80s con el modelo de Fukushima, el "neocognitron"[10], una red multicapa jerárquica que estaba diseñada para simular el reconocimiento de patrones visuales realizados por humanos. Este modelo fue mejorado por LeCun años después, generando una aplicación real con la red LeNet-5 [11], cuya tarea era la de reconocimiento de dígitos y demostró el potencial de las CNNs para aplicaciones reales.

El origen de las CNNs proviene de la idea de replicar la habilidad del cerebro de procesar e interpretar eficientemente información visual. La corteza visual de los mamíferos, en particular la humana, evolucionó para llevar a cabo tareas visuales complejas de manera eficiente, eficaz y con poca información [28, 30, 35]. Considerando que las CNNs fueron diseñadas a partir de mecanismos biológicos, es importante destacar que se ha demostrado que mapean características de una manera similar a las redes neuronales biológicas [36], por esto, han servido para predecir procesamiento de regiones cerebrales [36].

La serie de operaciones más importantes de una CNN son la convolución discreta y *pooling*.

Desde un punto de vista matemático la convolución discreta 2D se define como:

$$M(i,j) = (I * F)(i,j) = \sum_m \sum_n I(m,n) \cdot F(i-m, j-n). \tag{2.2}$$

Así, m y n iteran sobre un filtro F conocido como *kernel*, y (i,j) son las coordenadas de la imagen y $*$ representa la operación de convolución. Esta operación resulta en un *feature map* M que extrae ciertas características de la imagen. Conceptualmente, la convolución 2D equivale a superponer una máscara en la posición (i, j) de la imagen, y sumar los productos entre los elementos del kernel y la imagen superpuestos, ilustrado en la figura 2.6.

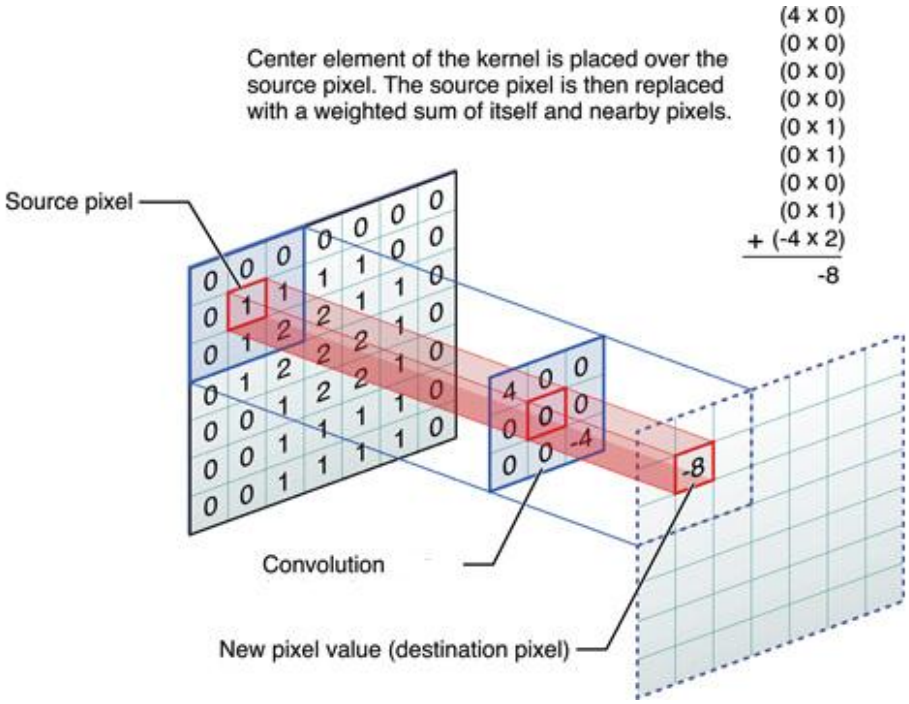


Figura 2.6: Operación de convolución. Un pixel se mapea a través de un filtro o *kernel* para obtener un pixel de salida. La CNN aprenden los filtros. Obtenido de [37].

Al igual que en los MLP, la imagen de salida M también pasa por una función de activación, normalmente no lineal, como $\text{ReLU}(x) = \max(0, x)$.

El submuestreo o *pooling* sirve para reducir las dimensiones espaciales desde un set de imágenes/*feature maps* al siguiente, esto se realiza para comprimir información, normalmente utilizando métodos como *average pooling* o *max pooling*.

$$P_{\max}(I)(i,j) = \max_{m,n \in W_{ij}} I(m,n). \tag{2.3}$$

Por ejemplo, en *max pooling*, aquí W_{ij} es una ventana de la imagen de entrada, centrada en la posición (i,j).

Al utilizar estas operaciones principales, de manera profunda e iterativamente, las CNNs pueden aprender a extraer características cada vez más complejas. La gran ventaja de los modelos convolucionales con respecto a los MLP es la complejidad de parámetros de entre-

namiento, ya que las CNN solo deben aprender los valores de los filtros de los feature maps, al no estar densamente conectados, como las MLP.

Otra importante distinción de las CNNs, es la habilidad de aprender jerarquías espaciales de las características de las imágenes de forma automática y adaptativa. Esto gracias a que la convolución es una operación que permite preservar información espacial, es decir, cómo relacionan espacialmente los pixeles, lo que es aprendido en función del tamaño de los filtros.

2.6. Entrenamiento de redes neuronales

El entrenamiento de redes neuronales es un proceso iterativo y complejo, que se basa en la utilización de múltiples ejemplos para aprender, lo cual se realiza mediante la actualización de los pesos de la red. Las partes fundamentales del entrenamiento se pueden resumir en inicialización, propagación directa, cálculo de la función de pérdida y propagación inversa.

La inicialización corresponde al valor inicial de los pesos del modelo con el que se iniciará el proceso de entrenamiento. De manera tradicional esta inicialización ha sido aleatoria, pero en la actualidad las estrategias más usadas son la inicialización He [38] o Glorot [39]. La inicialización es un factor determinante para el entrenamiento de las primeras iteraciones, particularmente para evitar problemas relacionados a la divergencia o extinción de gradientes.

Tras inicialización, el proceso de entrenamiento comienza con la propagación directa, donde se entrega los datos de entrenamiento a la red, de manera que los datos pasan por lo procesos de la red, aplicando los pesos, funciones de activación, etc. Este procedimiento finaliza con la predicción generada por la red.

Posteriormente, normalmente se utiliza el *accuracy* como métrica para determinar si el modelo mejora con el entrenamiento. La función de pérdida está directamente relacionada con la cuantificación del error entre lo que predice la red y la respuesta esperada. La función de pérdida más utilizada en clasificación multiclase corresponde a la entropía cruzada,

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log(\hat{y}_{i,c}), \tag{2.4}$$

donde C es el número de clases, $y_{i,c}$ indica si la etiqueta c está clasificada correctamente de la observación i. $\hat{y}_{i,c}$ es la probabilidad de que la observación i corresponde a la clase c.

Luego, la propagación inversa calcula el gradiente de la función de pérdida con respecto a cada peso en la red, de manera de obtener de qué manera cada peso debe ser ajustado para seguir reduciendo la función de pérdida que se busca minimizar, esto se realiza directamente por regla de la cadena, para cada peso,

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial h}\cdot\frac{\partial h}{\partial W}. \tag{2.5}$$

La propagación inversa permite calcular cómo debe ser modificados los pesos, pero el cambio de los pesos es realmente modificado por los algoritmos de optimización. Uno de los

algoritmos más utilizados para realizar este procedimiento es Gradiente Descendiente, particularmente las variaciones Gradiente Descendiente Estocástico (SGD), ADAM y RMSprop. La optimización se realiza en base a la información calculada para modificar los pesos de la propagación inversa,

$$W_{\text{new}} = W_{\text{old}} - \eta \cdot \frac{\partial L}{\partial W}, \tag{2.6}$$

donde η se conoce como la tasa de aprendizaje, un hiperparámetro muy importante para entrenar correctamente una red.

Existen diversas formas de escoger la tasa de aprendizaje, entre ellas se distinguen principalmente, si son fijas durante entrenamiento o si se realiza un cambio dinámico a través del entrenamiento. El problema del balance de la tasa de aprendizaje, se debe a que una tasa muy alta puede producir divergencias, mientras que una tasa pequeña puede producir que el entrenamiento se estanque en óptimos locales o el entrenamiento sea muy lento [40, 41].

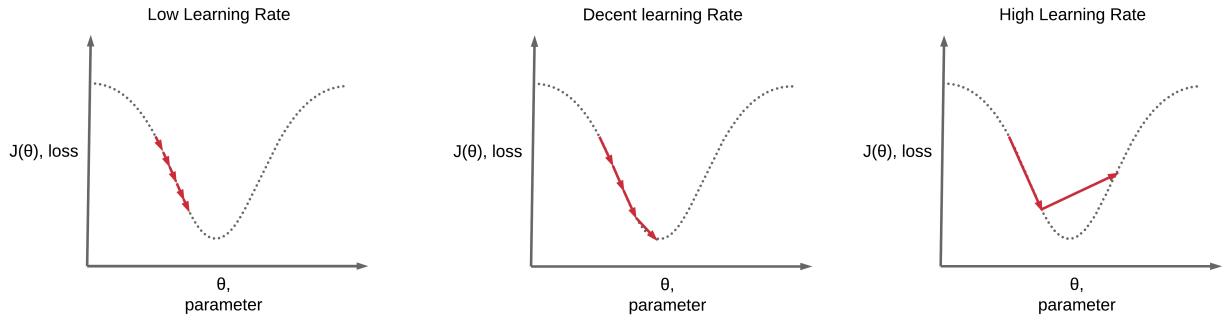


Figura 2.7: Problema de la tasa de aprendizaje. Una tasa de aprendizaje muy pequeña provoca convergencia lenta y atasco en óptimos locales, mientras que una tasa muy alta puede generar divergencia. Obtenido de [42].

Un mecanismo importante realizado en el entrenamiento de las redes es la regularización, que tiene como objetivo que una red no genere sobreajuste u *overfit* sobre los datos de entrenamiento. El sobreajuste corresponde a que el aprendizaje sobre los datos de entrenamiento se lleva hasta un punto en el que afecta negativamente el desempeño sobre datos que no ha visto, esto se refiere a que un modelo no es capaz de generalizar el aprendizaje, y más bien, solo sirve para resolver la tarea únicamente en los datos de entrenamiento, un ejemplo de dos dimensiones de características y dos clases se observa en la figura 2.8. Para atacar este problema las técnicas más utilizadas son el *dropout* que elimina neuronas aleatoriamente durante el entrenamiento, para que la red no dependa únicamente de ciertas neuronas, por otra parte, se usa la regularización L1 o L2 que penaliza cuando ciertos pesos tienen valores muy grandes, también se utiliza *early stopping*, que consiste en dejar de entrenar el modelo cuando el desempeño en los datos de validación deja de mejorar.

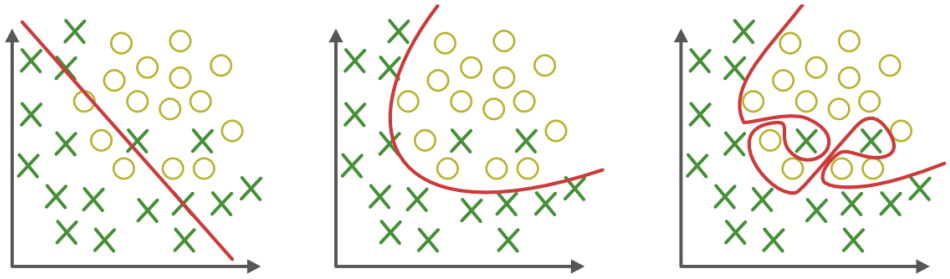


Figura 2.8: Diagrama de ajuste en un caso de dos clases con características de dos dimensiones. A la izquierda, *Underfit* o subajuste corresponde a un seccionamiento muy simple de las clases. A la derecha, *Overfit* o sobreajuste corresponde a un separación irreal de las clases. Al centro, *Appropriate-fit* o ajuste apropiado distingue correctamente la frontera de las clases. Modificado de [43].

Otros procedimientos que se relacionan al entrenamiento de la red son el número de épocas entrenamiento (cuántas veces la red ve los datos) y el *batch size* (cuántos grupos de datos se entregan a la vez a la red), hiperparámetros también importantes para optimizar correctamente el entrenamiento.

2.7. Diseño de redes neuronales convolucionales

El diseño de las arquitecturas de CNNs es un aspecto crucial que determina su desempeño, en términos de eficacia y eficiencia. El diseño considera configuraciones y selección de múltiples elementos estructurales, tales como el número de capas, tipo de capas y conexiones (convolución, *fully connected*, *densely connected*), tipos de operadores (*max pooling*, *average pooling*, así como también parámetros específicos de cada capa, es decir, tamaño de los *kernel*, número de filtros, número de neuronas, etc.

La búsqueda de arquitecturas óptimas para tareas específicas es uno de los problemas más básicos para la resolución de problemas mediante DL, pues sin una arquitectura clara no se puede procesar la información de manera correcta. A pesar de ser un aspecto fundamental, es un elemento de extensa y desafiante investigación [18, 44, 45], donde incluso los expertos no llegan a resultados óptimos [17, 46].

La importancia encontrar arquitecturas que sean mejores que las previas, ha sido desarrollado durante décadas [18] y se demuestra por el avance de técnicas pasando por el perceptrón, neocognitron, LeNet, ResNet, DenseNet y en la actualidad por los *vision transformers* [47].

A continuación se presentan las arquitecturas convolucionales más relevantes desarrolladas a través de los años:

2.7.1. LeNet-5

LeNet-5, desarrollado por Yann LeCun en 1998 [11], es considerado uno de los modelos pioneros de CNN. Fue diseñada para el reconocimiento de dígitos, aplicado particularmente

para procesar números escritos a mano en sistemas automáticos de lectura de cheques de bancos. Este modelo creó las bases para muchos de los conceptos fundamentales presentes en las arquitecturas modernas de DL.

La arquitectura de LeNet-5 introdujo innovaciones clave, incluyendo dos capas convolucionales, dos capas de submuestreo (*pooling*) y tres capas densas o *fully connected* (FC). Las capas convolucionales contienen filtros entrenables y este entrenamiento fue hecho con propagación inversa. La idea de utilizar submuestreo fue para reducir la resolución espacial y disminuir la complejidad computacional mediante la reducción del número de parámetros entrenables, una tarea muy importante para la limitada capacidad computacional de la época. Las capas FC se utilizaron para realizar la clasificación final.

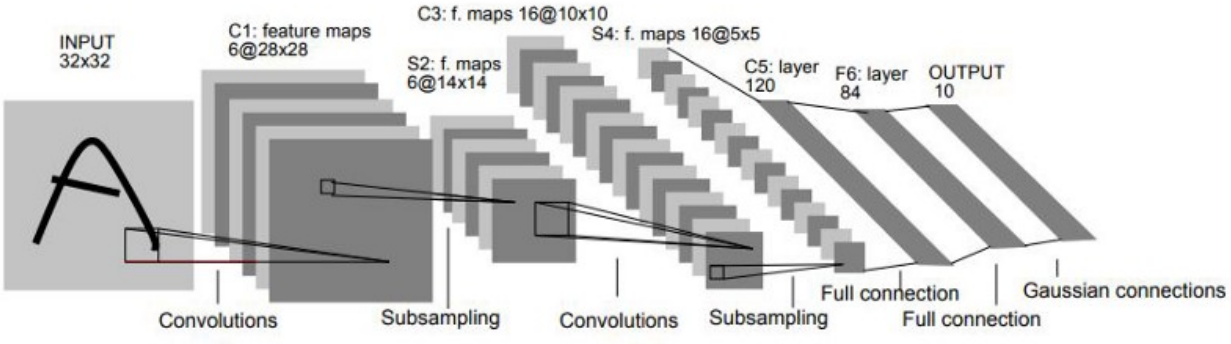


Figura 2.9: Diagrama de la arquitectura LeNet-5. Se observan dos capas convolucionales, dos de submuestreo, dos capas densas y una de conexiones gaussianas. Obtenido de [11].

El éxito de LeNet-5 versus los métodos contemporáneos en el reconocimiento de dígitos demostró el potencial de las CNN en aplicaciones reales. Esto sirvió como una motivación para un área de investigación que creció lentamente hasta la década de 2010s donde la investigación de CNN creció fuertemente [48].

2.7.2. AlexNet

AlexNet [12], desarrollada por Alex Krizhevsky, Ilya Sutskever y Geoffrey Hinton en 2012, marcó un hito para la tarea de reconocimiento visual. Esta arquitectura se consagró como ganadora en la competencia *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) en 2012, lo cual fue una demostración clara del potencial de las CNN en la clasificación de imágenes a gran escala.

AlexNet se destacó por varias características innovadoras que permitieron su gran desempeño. Consistía en un total de ocho capas entrenables, 5 convolucionales y 3 FC. Las capas convolucionales tenían 96, 256, 384, 384 y 256 filtros, con kernels cuadrados de tamaño 11, 5, 3, 3, y 3 respectivamente, por otra parte, las capas FC tenían 2048, 2048 y 1000 neuronas cada una.

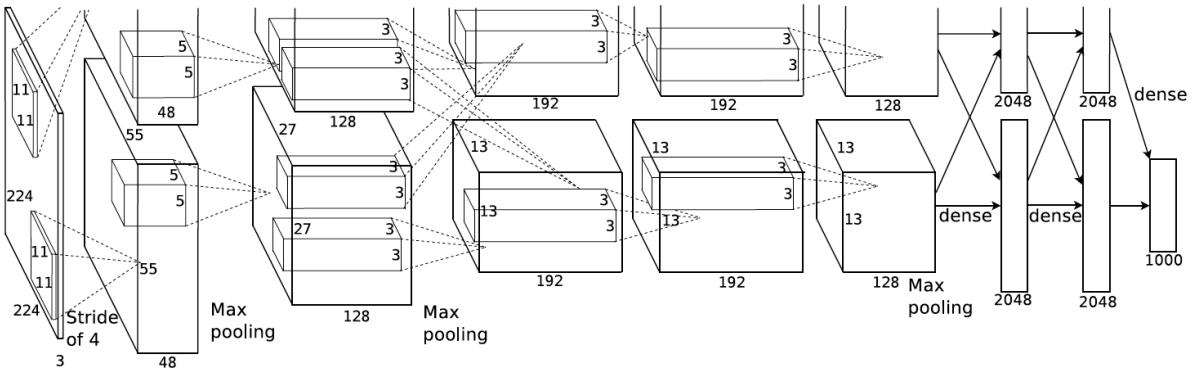


Figura 2.10: Diagrama de arquitectura AlexNet. La imagen de entrada pasa por cinco capas convolucionales y tres capas densas. Obtenido de [12].

Una de las contribuciones más importantes de AlexNet fue la introducción de la función de activación ReLU (*Rectified Linear Unit*), mucho más eficiente que las funciones de activación tradicionales como el tanh o el sigmoid.

AlexNet también incorporó técnicas de regularización y mejoras de aprendizaje como el *dropout* y *data augmentation* para contrarrestar el posible sobreajuste al ser una red más profunda de lo normal hasta la época. Desde punto de vista técnico también pudieron entrenar sobre 2 GPUs, algo novedoso para la época [48].

2.7.3. ZFNET

Sobre AlexNet se desarrolló ZFNet [49], por Matthew Zeiler y Rob Fergus en 2013, destacó en ILSVRC de 2013. Lo importante de esta arquitectura no fue solo su desempeño, sino su contribución a la comprensión de cómo las CNN procesan visualmente los datos.

La innovación de ZFNET viene de la técnica de visualización que denominaron *deconvolutional network*, que permitió mapear las activaciones de cada capa hacia la entrada. Esto sirvió para que los desarrolladores pudieran visualizar qué características se activaban con ciertas neuronas y de qué forma las capas convolucionales estaban capturando información.

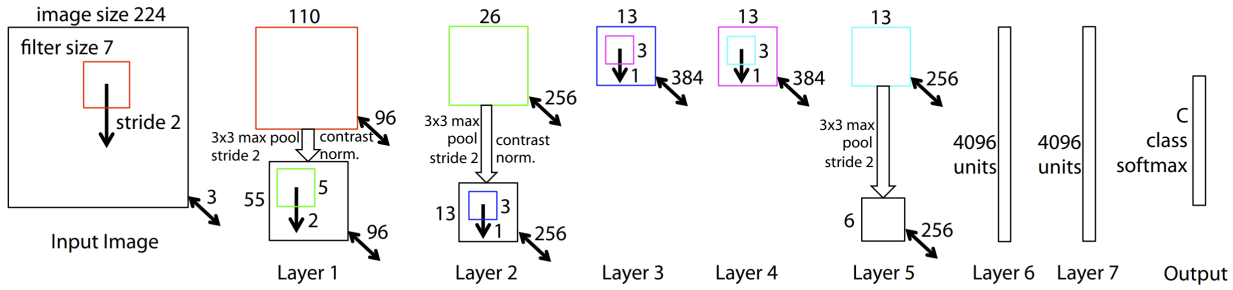


Figura 2.11: Diagrama de arquitectura ZFNet. La imagen de entrada pasa por cinco capas convolucionales y tres densas. Obtenido de [49].

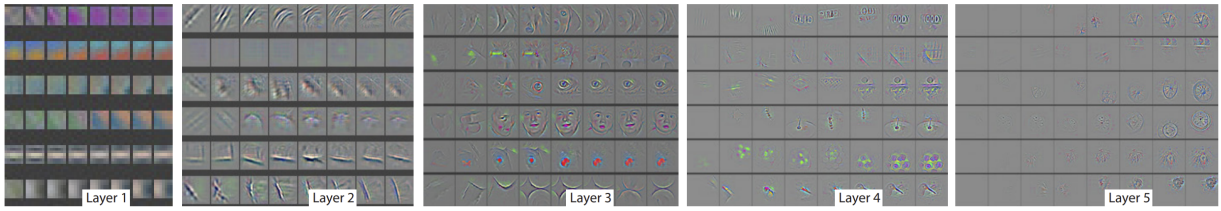


Figura 2.12: Visualización de activación de los filtros de las cinco capas convolucionales de ZFNet. Obtenido de [49].

De manera estructural, las modificaciones de ZFNet fueron menores, solo ajustando el tamaño de los *kernel* y el *stride* en la primera capa. Las demás características importantes se mantuvieron, tales como el uso de ReLu, *dropout* y la aumentación de datos.

2.7.4. GoogLeNet - Inception v1

GoogLeNet [50], también conocida como Inception v1, fue creada por investigadores de Google en 2014 y ganó el ILSVRC de ese año junto a VGGNET. Esta arquitectura introdujo una idea novedosa, en la construcción de modelos, con los módulos *inception*. Estos módulos permitieron a la red adaptarse mejor las diferentes escalas de las características de la imagen, gracias a la utilización paralela de filtros de diferentes tamaños. Los modulos *Inception* paralelizan multiples convolucionles para luego concatenar los *feature maps* en profundidad.

GoogLeNet era mucho más profunda que las anteriores (22 capas en total), pero contraintuitivamente eficiente en términos de computación y uso de parámetros. Esto se principalmente a la reducción del tamaño de los filtros y al uso de operaciones de *pooling*.

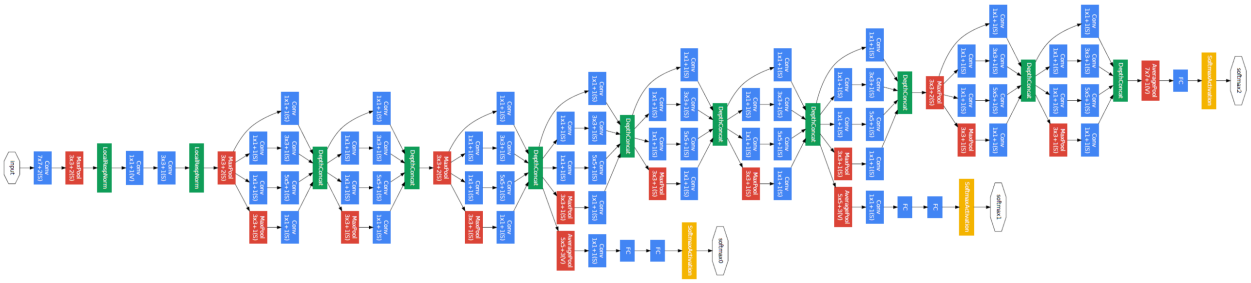


Figura 2.13: Diagrama de arquitectura GoogLeNet. Los bloques azules representan capas convolucionales y densas, los bloques rojos *pooling* y los verdes operaciones en filtros. Obtenido de [50].

Un aspecto importante de esta arquitectura fue la creación de una nueva forma de atacar el problema del gradiente desvaneciente o *vanishing gradient problem* en DL. Las capas auxiliares, insertadas en puntos intermedios de la red, permiten propagar gradientes de mejor manera durante el entrenamiento.

2.7.5. VGGNet

Karen Simonyan y Andrew Zisserman desarrollaron VGGNet entre 2013 y 2014, igualando a GoogLeNet en ILSVRC . El reconocimiento de la red es por su simplicidad y profundidad,

la cual consistía en capas repetitivas de convolución seguidas de una capa de submuestreo. Un aspecto clave de VGGNet era el uso de *kernels* pequeños (3x3 y 1x1), manteniendo controlado el número de parámetros.

VGGNet presentó dos variantes VGG-16 y VGG-19, con 16 y 19 capas, respectivamente. Por ejemplo, en VGG-16, algunos parámetros importantes son, 16 capas convolucionales, 5 capas de *Max-pooling*, y 3 capas FC, donde en algunos puntos las convoluciones alcanzan los 512 filtros. También fue aumentado el número de neuronas de las capas FC, respecto a AlexNet.

2.7.6. ResNet

ResNet [14], desarrollada por Kaiming He *et al* en Microsoft Research en 2015, generó una innovación importante y simple. Ganadora del ILSVRC 2015, ResNet introdujo el concepto de "conexiones residuales", permitiendo conectar señales de capas alejadas, aliviando el problema del gradiente desvaneciente.

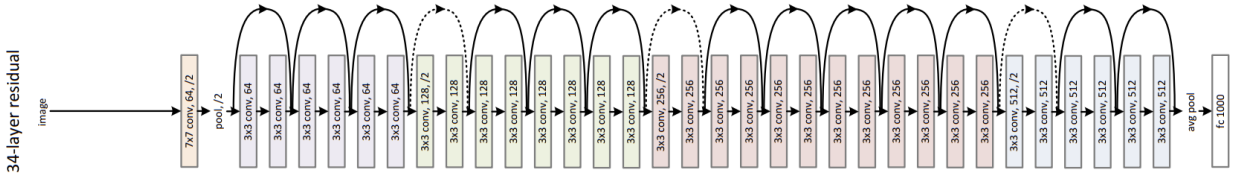


Figura 2.14: Diagrama de arquitectura ResNet-34. Las variaciones de color representan cambios de dimensionalidad, las conexiones superiores son *skip-connection*. Obtenido de [14].

ResNet presentó múltiples variaciones, desde 18 hasta 152 capas. Las conexiones residuales consideraban conectar salidas de ciertas capas y sumarlas a otras. La incorporación residual permitió entrenar redes mucho más profundas sin generar mayores problemas. Con este trabajo se demostró, una vez más, que la profundidad de la red es un factor muy importante para incrementar el desempeño.

2.7.7. DenseNet

DenseNet [15], desarrollada por Gao Huang *et al.* en 2017, introdujo una nueva forma de conectar capas. A diferencia de ResNet, que propone conexiones residuales, DenseNet concatena las salidas de manera densa, es decir, cada capa recibe como entrada todas las salidas de las capas anteriores, este procedimiento es realizado en bloques. Al incorporar esta densidad, se observa un aumento en la cantidad de filtros previo a cada capa, este factor los autores lo denominaron *growth rate*.

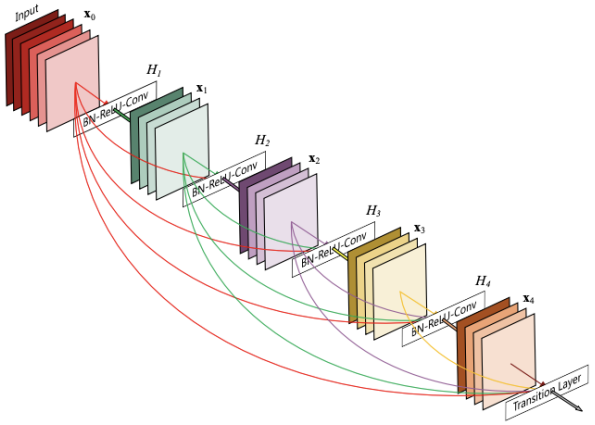


Figura 2.15: Diagrama de bloque denso tipo DenseNet. Representación de un bloque de cuatro repeticiones con *growth rate* tamaño cuatro. Obtenido de [15].

La arquitectura tenía varias ventajas, la principal siendo la eficiencia en términos de parámetros de entrenamiento. A pesar de incorporar más conexiones, contradictoriamente se genera una disminución de los parámetros totales de entrenamiento, además, creando un efecto de regularización. Los bloques densos de DenseNet intrínsecamente disminuyen el problema del gradiente desvaneciente, pues el flujo de información permite que capas muy profundas accedan a información de las primeras capas, dentro de cada bloque.

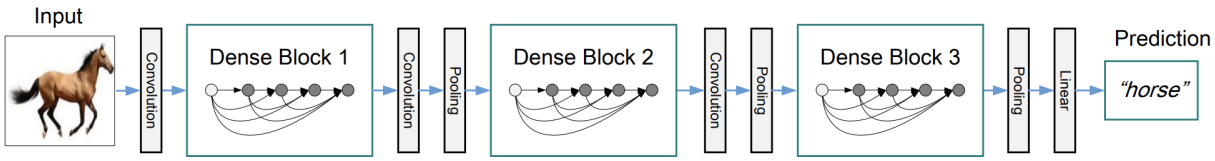


Figura 2.16: Diagrama de DenseNet. Ejemplo con tres bloques densos. Las operaciones entre bloques se denominan capas de transición que reducen dimensionalidad. Obtenido de [15].

2.7.8. PyramidNet

PyramidNet [51], presentada en 2017, basada en conexiones residuales, introdujo una nueva forma de aumentar dimensionalmente las capas en las CNN. A diferencia de las arquitecturas tradicionales donde el incremento en la dimensión de los canales se realiza abruptamente en las transiciones entre bloques, PyramidNet aumenta de manera uniforme y gradual la dimensión de los canales a lo largo de la profundidad de la red. La idea de esta técnica es permitir un flujo más suave y efectivo de la información y de los gradientes a través de las capas, facilitando entrenamiento.

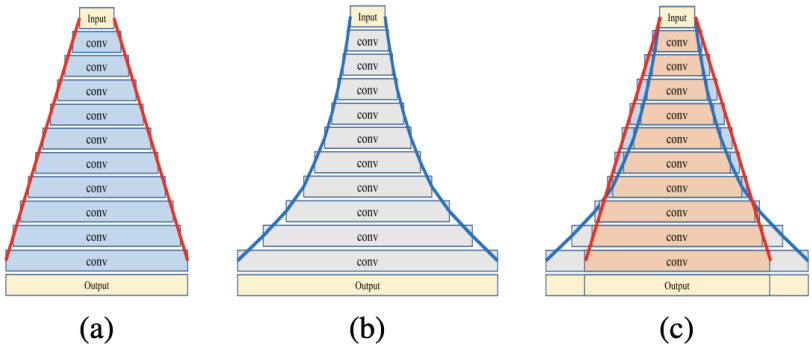


Figura 2.17: Diagramas de arquitecturas tipo PyramidNet. (a) versión aditiva, (b) versión multiplicativa y (c) comparación de versión aditiva y multiplicativa. Obtenido de [51].

Esta arquitectura demostró ser especialmente eficaz en tareas de clasificación de imágenes, logrando gran rendimiento en ImageNet, CIFAR-10 Y CIFAR-100, superando lo publicado por las redes residuales [14].

2.7.9. EfficientNet

Un cambio importante en el paradigma de diseño de redes fue presentado por Google Research mediante EfficientNet [16], en 2019. Hasta ese momento, lo común era mejorar las redes incrementando la profundidad, el ancho o la resolución. Este trabajo introdujo una metodología sistemática y escalable para ajustar estas tres dimensiones de manera balanceada, para obtener mejores modelos.

La modificación de la escala de las redes propuesta, es un tipo de escalado compuesto, donde la escala aumenta de manera uniforme en profundidad, ancho y resolución. Este enfoque permitió crear redes mucho más eficientes y con desempeño similar o superior a lo propuesto anteriormente.

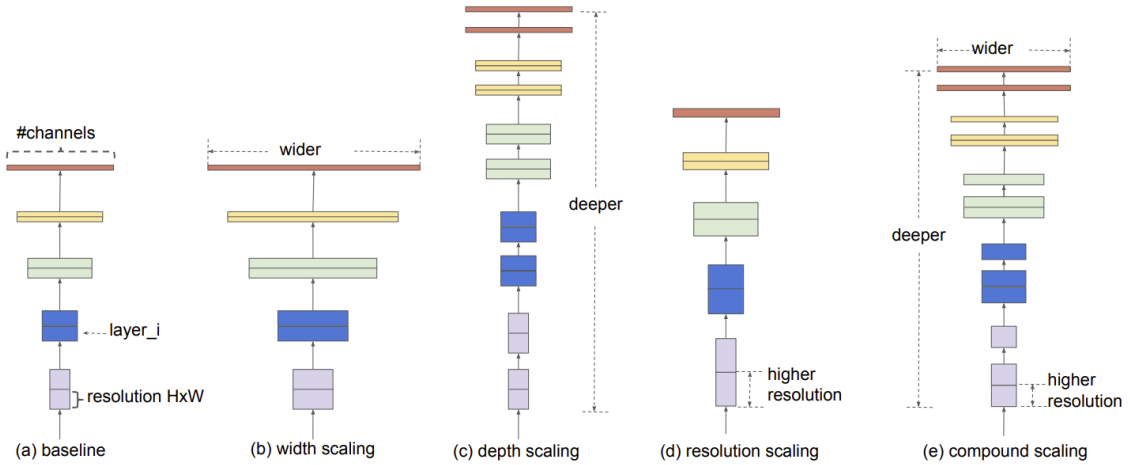


Figura 2.18: Diagramas de tipos de escalamiento en redes neuronales convolucionales definidos por [16]. (a) definición de parámetros, resolución, capas y canales, (b) escalamiento de ancho, (c) escalamiento en profundidad, (d) escalamiento en resolución y (e) escalamiento compuesto. Obtenido de [16].

Una de las ventajas más importantes de este tipo de arquitecturas es la capacidad de los modelos propuestos en comparación con la cantidad de parámetros, lo que lo hace una atractiva opción cuando la capacidad computacional es limitada.

2.8. Diseño automático de redes neuronales y neuroevolución

El diseño automático de redes neuronales o *Neural Architecture Search* (NAS) es una subtarea de *Automatic Machine Learning* (autoML), el cual se centra en automatizar el diseño de arquitecturas de ANNs, principalmente enfocado en las CNN. La idea es identificar un diseño de red óptimo para una tarea específica, trabajo que normalmente es realizado por un experto, mediante el paradigma de ensayo y error, por lo que requiere cantidades significativas de tiempo y experiencia. NAS se utiliza activamente en el campo del reconocimiento de imágenes [17, 18], en el que se centra este trabajo.

Los principios fundamentales de NAS son el espacio de búsqueda, la evaluación del rendimiento y la estrategia de búsqueda. El primero está relacionado con la composición de la red, como tipos de capas, profundidad, hiperparámetros, etc. La evaluación del rendimiento -en una tarea determinada- se basa en qué métrica se utiliza para distinguir qué red es mejor que otra. Existen varias estrategias de búsqueda: *Grid Search*, *Random Search*, *Bayesian Optimization*, *Gradient Based Optimization*, *Reinforcment Learning*, algoritmos evolutivos, métodos híbridos y otros [17, 18].

La tarea NAS es matemáticamente un problema de optimización. El espacio de búsqueda $\mathcal{A}$ se define como el conjunto de todas las arquitecturas posibles que se pueden construir dado un conjunto de N parámetros de búsqueda predefinidos $\theta = \{\theta_1, ..., \theta_N\}$. El objetivo de NAS es encontrar los valores óptimos de θ que maximicen el rendimiento de la red neuronal. Una arquitectura $a \in \mathcal{A}$ está representada por sus parámetros θ que definen su estructura. Cabe

destacar que la búsqueda muchas veces no se limita únicamente a parámetros estructurales [17, 18, 52]

El rendimiento de cada arquitectura $a(\theta)$ se evalúa utilizando una función objetivo,

$$f(a(\theta), D_{\text{train}}, D_{\text{valid}}),$$

donde D_{train} es el conjunto de datos de entrenamiento y D_{valid} es el conjunto de datos de validación. Esta evaluación podría ser *accuracy*, *precision*, *recall*, tiempo de entrenamiento, tamaño del modelo o una métrica importante para la tarea sobre el conjunto de datos utilizado [17, 18].

Por lo tanto, el problema de optimización es encontrar la arquitectura óptima $a^*(\theta)$ que maximiza (o minimiza) el objetivo

$$a^*(\theta) = \arg\max_{\theta \in \mathcal{A}} f(a(\theta), D_{\text{train}}, D_{\text{valid}}). \tag{2.7}$$

En términos prácticos, la métrica de evaluación más utilizada suele ser *accuracy* y el objetivo es maximizarla [17, 18]. La forma de resolver el problema mediante generaciones que van mejorando el rendimiento, se denomina neuroevolución [45], pues este procedimiento imita a la evolución biológica, creando redes especializadas para cumplir tareas específicas.

2.8.1. Estrategias de NAS

Grid Search o búsqueda de grilla es un tipo de algoritmo de fuerza bruta que discretiza $\mathcal{A}$ en una cuadrícula. Este método es la forma tradicional de encontrar buenos parámetros [53]. Cada punto corresponde entonces a una combinación de parámetros $\theta = (\theta_1, \theta_2, ..., \theta_n)$, donde θ_i tiene un conjunto finito de valores. Este algoritmo evalúa la función objetivo $f(a(\theta))$ en todas las combinaciones posibles y elige la arquitectura donde f es máxima. Implementaciones de esta formulación se pueden encontrar en librerías de machine learning ampliamente utilizadas como Keras.

Random Search (RS) o búsqueda aleatoria selecciona puntos aleatoriamente de $\mathcal{A}$. En cada iteración, θ se selecciona de una distribución de probabilidad sobre $\mathcal{A}$ (normalmente uniforme) y se evalúa $f(a(\theta))$. El proceso termina con un número definido de iteraciones o cuando se cumple un criterio de término. Este método fue uno de los primeros involucrados en NAS de grandes espacios [54].

Bayesian Optimization (BO) o optimización bayesiana construye un modelo probabilístico M de la función objetivo f y lo utiliza para seleccionar el θ más prometedor para evaluar a continuación. En general se utiliza un Proceso Gaussiano (GP) para modelar f y una función de adquisición g para decidir dónde muestrear a continuación: $\theta_{\text{next}} = \arg\max_{\theta} g(\theta|M)$. El modelo M se actualiza con cada evaluación, mejorando la estimación de f para la próxima iteración. Al igual que en los algoritmos genéticos (GA), este método no requiere una función objetivo explícita, sin embargo, carece de la escalabilidad de los GAs, ya que el tiempo escala cúbicamente con el número de observaciones [52].

Los métodos tipo *Gradient-based Optimization* (GO) u optimización de gradiente utili-

zan el gradiente de f con respecto a θ para encontrar los parámetros óptimos. La regla es típicamente de la forma $\theta_{\text{new}} = \theta_{\text{old}} - \eta \nabla_\theta f(a(\theta))$, donde η es la tasa de aprendizaje de la optimización. Este procedimiento matemáticamente es idéntico al utilizado en el entrenamiento de redes. Los GOs están entre los métodos más utilizados en NAS [17, 18]. Una dificultad importante a considerar en los GOs es que implica la diferenciabilidad en la función objetivo, por lo que a menudo se utiliza una relajación en el espacio de búsqueda [55]. Además, este método se usa comúnmente con una restricción sobre el espacio de búsqueda utilizando una super-red o super-grafo para encontrar la arquitectura óptima [17, 18, 55, 52], lo que limita la búsqueda. También, el diseño de la super-red o grafo requiere conocimiento experto.

En *Reinforcement Learning* (RL), un agente aprende a proponer posibles arquitecturas que maximizan la recompensa. El problema de optimización puede enmarcarse como el aprendizaje de una política o *policy* $\pi(\theta|s)$ que maximiza la recompensa esperada: $E_{\theta\sim\pi}[R(\theta)]$, donde R es la función de recompensa. Ejemplos de esto son [56], que utilizaron una red neuronal recurrente (RNN) como controlador, y el algoritmo de gradiente de políticas para entrenar este controlador. En NASNET [57], un controlador de RNN genera y entrena iterativamente diversas arquitecturas de redes secundarias, utilizando sus *accuracy* de validación para refinar la capacidad de producir arquitecturas cada vez más efectivas con el tiempo. En METAQNN [58], el agente de aprendizaje se entrena para elegir secuencialmente capas de CNNs, utilizando $Q - learning$ con una estrategia de exploración ϵ-greedy. Sin embargo, los algoritmos de RL son computacionalmente costosos, por ejemplo [56] requirió 22,400 días-GPU, mientras que NASNET y METAQNN necesitaron 2000 y 100 respectivamente.

Los Algoritmos Evolutivos (EA) seleccionan un subconjunto de arquitecturas

$$P = \{a_1(\theta), a_2(\theta), ..., a_m(\theta)\},$$

que pueden evolucionar con el tiempo, tratando de aumentar la función objetivo. Como en la búsqueda aleatoria, el proceso termina con un número definido de generaciones o cuando se cumple un criterio de parada. Hay varios tipos de EA, algunos de ellos son Algoritmos Genéticos [46, 59, 60, 61], *Differential Evolution* (DE) [62, 63], *Particle Swarm Optimization* (PSO) [64, 65] o algoritmos híbridos [66].

2.9. Algoritmos genéticos

Los Algoritmos Genéticos (AG) son un tipo de algoritmos evolutivos que se utilizan con frecuencia para resolver problemas de optimización y búsqueda. Su origen proviene de la intención de biomimetizar los principios de la evolución, específicamente el proceso de selección natural, la relación con la genética y el paradigma de supervivencia del mejor [67].

Originalmente, los AG fueron creados para estudiar el fenómeno de adaptación que ocurre en sistemas naturales y biomimetizar la robustez de estos sistemas biológicos [68]. Los mecanismos de la selección natural se traducen en tres aspectos bastante simples: entrecruzamiento, selección y mutación.

El GA comienza creando una población de soluciones, que normalmente se representan como cromosomas únicos con diferentes genes y suelen codificarse como cadenas de caracte-

res, pero también pueden ser de otros tipos [67]. Cada posible solución candidata se evalúa de alguna forma y se traduce en una puntuación de *fitness*, que determina qué tan bueno es el individuo en comparación con otros. El mecanismo de selección imita que no todos los individuos de una población pueden generar descendencia y está directamente relacionado con el *fitness* del individuo, los mecanismos de selección más comúnmente utilizados son la selección de ruleta, selección por torneo o selección por ranking. Los mecanismos de selección están estrechamente relacionados con el equilibrio de exploración/explotación [67, 69] dentro del espacio de búsqueda. Los individuos seleccionados experimentan un intercambio de genes llamado entrecruzamiento y luego una alteración de genes conocida como mutación para producir nuevos candidatos, que constituirán la próxima generación de población/soluciones.

Los AG se utilizan en diferentes escenarios debido a su versatilidad, particularmente donde el espacio de búsqueda es complejo, muy grande, poco comprendido o incluso cuando la función objetivo no está disponible [67, 70]; este último caso es exactamente la formulación del problema de NAS. Su capacidad para explorar vastos espacios de búsqueda, el potencial de escapar de mínimos locales y encontrar soluciones globales los convierte en una herramienta flexible para usar en problemas de optimización [67]. Las limitaciones de los AG están relacionadas con el extenso uso computacional, donde el cuello de botella suele ser el cálculo de *fitness* [17, 18], lo que en el caso de las redes neuronales, normalmente significa entrenar un modelo desde cero. También, los AG pueden ser sensibles a la configuración de parámetros como la tasa de mutación, la tasa de entrecruzamiento, el tamaño de la población o el tipo de operadores de reproducción utilizados [17, 18].

Los AG se prefieren en la tarea de NAS por diversas razones, entre las principales [17, 18, 45, 71]:

- Exploración de espacios de búsqueda muy grandes. Son capaces de buscar óptimos en espacios de alta dimensionalidad, donde otros métodos pueden estancarse.
- Capacidad de utilizar parámetros discretos en CNN. A diferencia de métodos GO, los AG pueden manejar de manera directa la optimización discreta.
- Flexibilidad y adaptabilidad. Tienen la capacidad de ser utilizados para búsquedas de arquitecturas significativamente distintas.
- Menor sensibilidad de parámetros. A pesar de ser sensible a cambios en parámetros internos, esta sensibilidad es menor a métodos de BO o GO.
- Incorporación de conocimiento previo. La incorporación de modelos conocidos como buenos a la población inicial permite rápidamente al algoritmo adaptarse a buscar mejores soluciones.

2.10. Mecanismos de CNN Bio-inspirados

Como se mencionó en la sección de ANNs, muchos de los diseños y mecanismos de redes neuronales provienen de una contraparte biológica. Estos siguen siendo parte de inspiración para las mejoras permanentes en la investigación de redes neuronales.

Este trabajo utiliza los mecanismos de atención y el aprendizaje multitarea para para mejorar el desempeño de las redes. También, se busca biomimetizar los parches neuronales utilizados para el reconocimiento de rostros, cuerpos y objetos [36, 72, 73].

2.10.1. Mecanismos de atención

Los mecanismos de atención en CNNs provienen de la inspiración de procesos cognitivos presentes en mamíferos y particularmente humanos. La capacidad humana de suprimir selectivamente información, para resaltar otra es lo que se denomina atención. Otra forma analizar este aspecto, se relaciona con la asignación de recursos/energía para el procesamiento de información, pues el cerebro tiene una capacidad limitada físicamente. La atención permite concentrarse selectivamente en un subconjunto de estímulos, ignorando otros, lo que mejora la eficiencia para procesar la información sensorial.

Esta capacidad se busca biomimetizar en las CNNs con el objetivo de mejorar su capacidad para enfocarse selectivamente en partes específicas de los datos que son más relevantes para una tarea dada. Estos mecanismos buscan ajustar dinámicamente la asignación de importancia o recursos a las diferentes regiones de los datos. Por ejemplo, cuando una persona intenta identificar un ave en un paisaje, se enfoca principalmente en las características del ave: plumas, color, forma, etc. La idea es realizar este mismo procedimiento mediante el aprendizaje en CNNs.

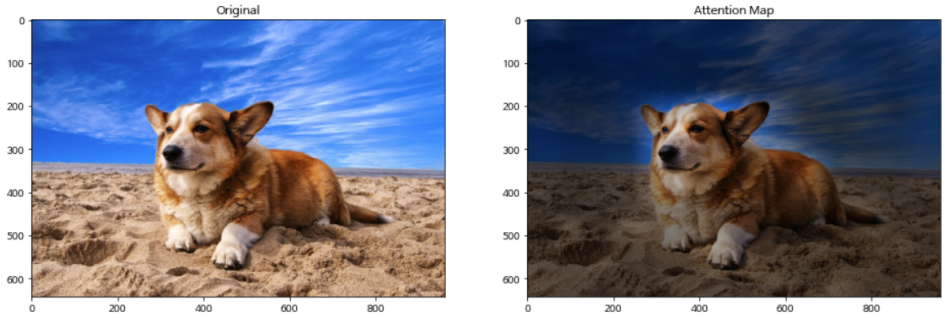


Figura 2.19: Mapa de un mecanismo de atención. La imagen izquierda muestra la imagen original, la imagen derecha mapea el brillo en función del valor de atención. Obtenido de [74].

En las CNNs la atención se puede incorporar de varias formas, entre ellas: espacialmente, en la dimensión de canales o en las transformaciones de información. Esta integración en los últimos años ha llevado a notables mejoras en el rendimiento en una variedad de aplicaciones, particularmente en aquellas que involucran reconocimiento y clasificación de imágenes.

2.10.2. Aprendizaje multitarea

Naturalmente los humanos y otros animales aprenden y realizan múltiples tareas de manera simultánea o secuencial, estas se desarrollan adaptando y utilizando diversas habilidades en función del contexto. El aprendizaje de una habilidad puede transferirse a otra, por ejemplo, jugar tenis de mesa requiere de seguir la trayectoria de un objeto muy pequeño a gran velocidad, esto incrementa la coordinación mano-ojo, y esta mejora puede transferirse a otro

dominio, por ejemplo, reaccionar más rápidamente al manejar un vehículo.

A menudo, estas habilidades no tienen una obvia relación entre ellas, de igual forma transfieren el aprendizaje entre dominios de una manera indirecta. El aprendizaje multitarea en el contexto de ANN tiene un enfoque similar, la idea es entrenar un modelo con múltiples tareas a la vez, donde las tareas pueden o no estar relacionadas. La idea es optimizar tiempo y modelos, de manera de aprovechar las sinergias y características comunes entre las tareas, con el objetivo de mejorar el aprendizaje global del modelo.

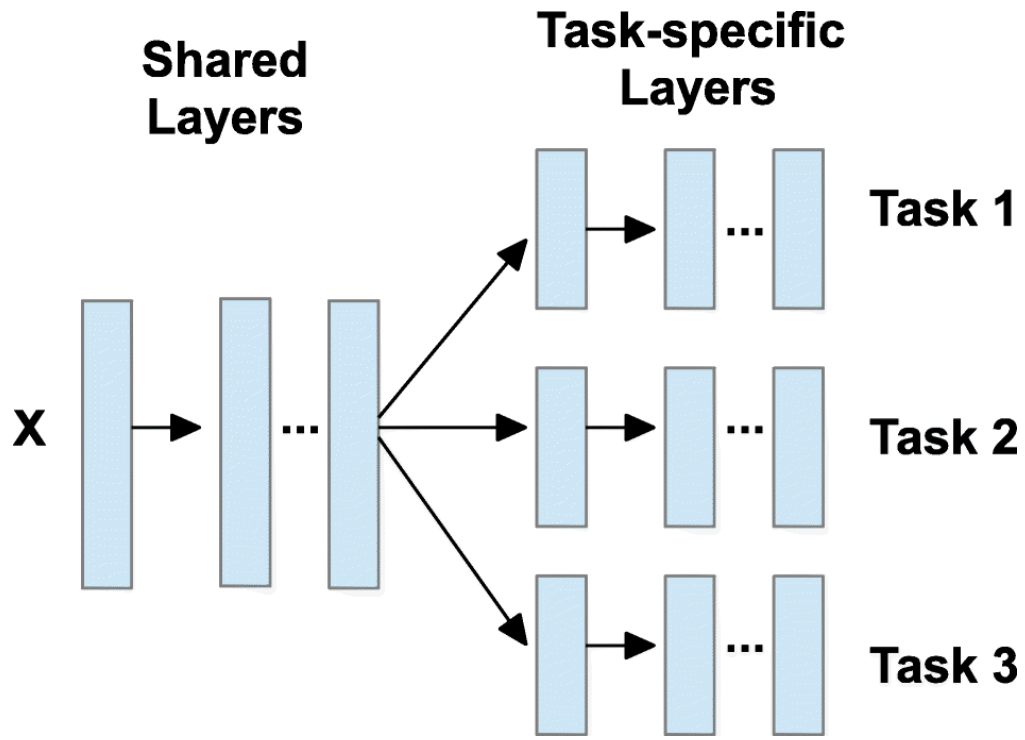


Figura 2.20: Diagrama de aprendizaje multitarea. La sección izquierda representa las capas compartidas, a la derecha un ejemplo de capas específicas para tres tareas. Obtenido de [75].

En el aprendizaje multitarea, un único modelo comparte capas para las diferentes tareas, manteniendo ciertas capas específicas para cada tarea, las capas específicas normalmente se denominan *heads*. Al compartir capas el modelo puede aprender representaciones más robustas y versátiles, las que pueden mejorar el rendimiento global y también ayudar a la generalización.

2.10.3. Modelos en cascada y *Ensemble Networks*

Los modelos en cascada se utilizan para resolver tareas parcialmente, con el objetivo de tener un modelo especializado para resolver en diferentes niveles de complejidad. Esta idea viene indirectamente inspirada de principios biomiméticos, por la manera en que se combinan procesos secuenciales, para hacer que los resultados sean más efectivos en la cadena final. Un ejemplo claro de esto son los diferentes parches neuronales cerebrales que identifican rostros, en función del ángulo [73].

Esto tiene también relación con las *Ensemble Networks* (EN), modelos creados a partir de múltiples modelos. Las EN pueden ser de diferentes tipos, combinando redes entrenadas con datos diferentes o similares. La idea de las EN es generar mejores resultados, basándose en la utilización de los mejores recursos de cada una de las redes independientes, lo que en muchas ocasiones cumple con el propósito de mejorar el rendimiento global.

Se podría decir que los modelos en cascada son un a caso particular de EN secuenciales, ya que se utilizan múltiples características en los diferentes modelos para mejorar el rendimiento global. Las EN aprovechan la diversidad y la especialización para mejorar el rendimiento general, por esto cada etapa permite un enfoque más profundo y detallado en cada parte del problema.

Capítulo 3

Estado del arte

El presente capítulo presenta el estado del arte relacionada a la tesis. En primer lugar, se presentan resultados de investigaciones modernas del procesamiento visual, retinotopía primate, atención humana y funcionamiento de los parches neuronales en la identificación de objetos. Luego, se presentan métodos del estado del arte relacionados a la búsqueda automática de redes neuronales, neuroevolución, módulos de atención y aprendizaje multitarea. Finalmente, se indican las bases de datos utilizadas regularmente en el problema abordado y la pertinencia del problema tratado en la tesis.

3.1. Procesamiento visual, retinotopía, atención y parches neuronales

Las dos conocidas vías de procesamiento visual humano, dorsal y ventral, se especializan, como ya se mencionó, en diferentes tareas: percepción, movimiento y coordinación de respuestas se encuentran en la vía dorsal [28], mientras que en la vía ventral se encuentra el reconocimiento de rostros, lugares, formas, palabras, entre otros [27, 28, 30, 36, 72, 73, 76, 77].

Una de las secciones más importantes en la clasificación de objetos en el cerebro humano es el giro temporal inferior o *Inferotemporal Cortex* (IT), que es parte de la última sección del procesamiento visual ventral y es donde se definen las categorías a las que pertenecen los objetos [36, 76, 78]. Un aspecto importante es que se ha demostrado que la extracción de características de IT es similar a las de las CNNs [36].

La información codificada que llega a IT es parte de la relación existente entre la retinotopía y la atención [22]. En el sistema visual, las diferentes secciones dentro de una escena no son procesadas con la misma precisión, un fenómeno que se observa incluso en la primera capa del procesamiento visual: la densidad de células receptoras en la retina no es uniforme, siendo más densa en la fóvea [24].

La atención permite hacer procesamiento selectivo del campo visual, lo que representa la asignación dinámica de los limitados recursos cerebrales, y de capacidad en el caso artificial, de manera eficiente y efectiva [22]. En el sistema visual humano, la atención permite seleccionar secciones de la información visual, procedimiento realizado primordialmente por la "red de atención ventral"(VAN) [79], que asigna recursos a diferentes partes de los mapas retinotópicos según sea necesario [24].

Se ha identificado que IT no solo clasifica objetos, sino que también organiza su percepción en un *espacio de objetos* que puede ser mapeado a un modelo de CNN. Este mapeo revela redes anatómicamente distintas dentro del IT, cada una correspondiente a diferentes dimensiones de este espacio-de-objetos [36].

Estudios realizados por Margaret S. Livingstone *et al.* y Rishi Rajalingham *et al.* refuerzan la idea de procesamiento mediante parches neuronales, mostrando cómo estas redes especializadas en una selección de objetos están interconectadas [77, 78].

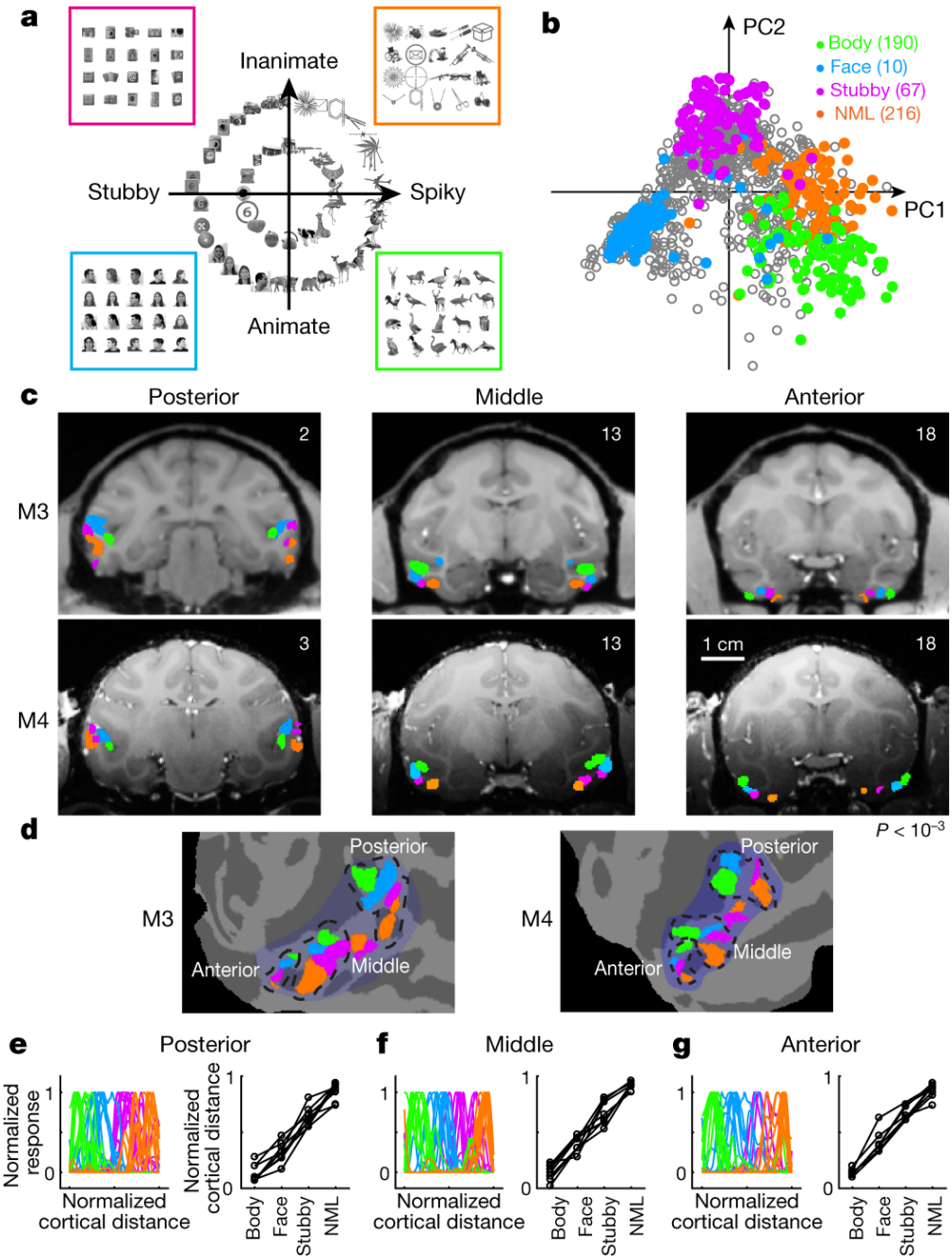


Figura 3.1: Diagramas de identificación de parches neuronales. El diagrama (a) muestra los parches neuronales predichos (*ground-truth*), (b) indica un diagrama de componentes principales para evaluaciones de diferentes clases, (c) y (d) indican los parches neuronales encontrados para cada clase en el cerebro primate, (e) indica la respuesta espacial normalizada de cada clase. Obtenido de [36].

Arcaro *et al.* destaca la especialización de las diferentes áreas de la corteza visual para el procesamiento de diferentes tipos de información visual. Este estudio detalla cómo los parches neuronales específicos en la corteza contribuyen a la tarea de reconocimiento visual [73].

3.2. Métodos de búsqueda automáticos de arquitecturas neuronales

Random Search es una de las primeras aproximaciones a realizar NAS, sorprendentemente RS es un buen *baseline* para resolver el problema [17, 80, 81]. Es más, fue utilizado en las primeras grandes búsquedas realizadas [54], obteniendo 3.91 % de error, demostrando la importancia del espacio de búsqueda [54], sin embargo, esta búsqueda requirió de 300 gpu-days. En general, RS se utiliza como *baseline* [17, 55, 82], pero es importante explicitar la manera de utilizar RS, pues es un baseline que continúa siendo muy competitivo con métodos de búsqueda actuales [82]. En espacios de búsqueda *benchmark* como DARTS [55] se ha llegado hasta 3.29 % de error con esta metodología.

A pesar de su uso en la optimización de hiperparámetros de ANNs, *Bayesian Optimisation* (BO) se utiliza con menos frecuencia para NAS, principalmente por limitaciones de *toolboxes* y porque BO se enfoca en búsquedas de baja dimensión [17]. Kandasamy *et al.* en NASBOT [83] aplicaron BO a un espacio de búsqueda que incluye parámetros no escalares, utilizando procesos gaussianos y obteniendo un 8.69 % de error en CIFAR-10, brindando una nueva herramienta para NAS. Estos métodos se han mejorado, por ejemplo, con predictores neuronales que estiman el desempeño de arquitecturas no vistas, alcanzando un 2.64 % de error [84] en el espacio de búsqueda de DARTS [55], procedimiento que se muestra en la figura 3.2.

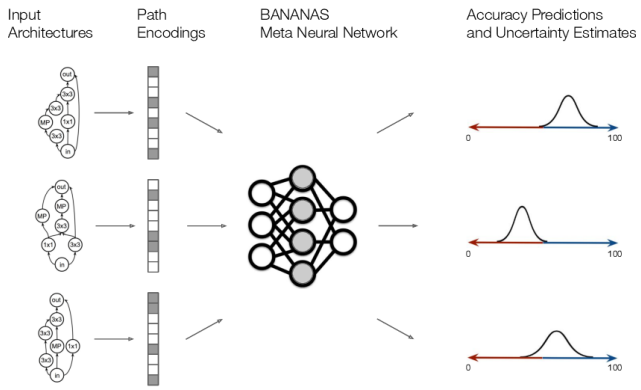


Figura 3.2: Diagrama del procedimiento de BANANAS mediante BO. Las arquitecturas candidatas se codifican para utilizar una ANN que predice su desempeño. Obtenido de [84].

DARTS [55] es un conocido método de NAS basado en optimización de gradientes (GO), con un desempeño de 2.76 % en CIFAR-10 y 26.7 % en ImageNet. Se basa en la búsqueda de la red óptima dentro de una *super-red* o *one-shot model*, una arquitectura que contiene todas las posibles combinaciones de redes del espacio de búsqueda en forma de sub-redes. Sin embargo, la construcción de esta super-red es un desafío complejo que requiere conocimiento experto [17, 18]. La visualización de super-red y sub-redes se encuentra en la figura 3.3.

Para lograr utilizar GO, es necesario que el espacio de búsqueda sea diferenciable [55, 17], lo que no es válido para búsqueda parámetros discretos u operaciones; es por esto que DARTS utiliza relajación continua, en vez de fijar operaciones o_i* en las capas, se utiliza la combinación convexa de operadores de la capa $\{o_i\}_1^n$ y se computan los ponderadores α_i de la

combinación convexa $\Sigma o_i \alpha_i$, colapsando a la operación discreta, escogiendo el $i*$ que es máximo en $o_i \alpha_i$.

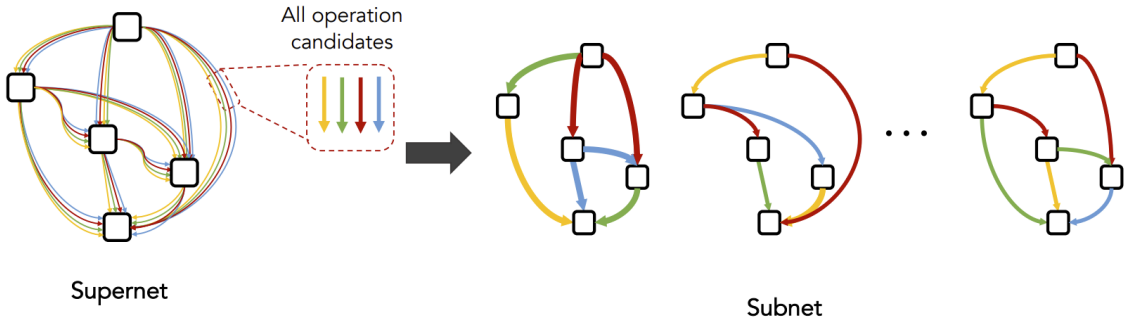


Figura 3.3: Super-red y sub-redes de DARTS. Los colores diferentes, indican las posibles operaciones entre nodos. Obtenido de [17].

Además, DARTS tiene una importante suposición: utilizar el *one-shot model* para evaluar la sub-red (es decir, entrenar únicamente la super-red y transferir pesos a la sub-red), permite generar un ranking de las redes como si fueran entrenadas en su totalidad. Esta aproximación aún se encuentra en controversia, pues existe evidencia que esta suposición no se cumple en muchos casos, y que además depende fuertemente de las arquitecturas, datos y técnicas de entrenamiento [17]. A pesar de estas limitantes, DARTS sigue siendo utilizado como referente en NAS, con investigadores atacando sus debilidades, mejorando en diferentes aristas con múltiples variaciones, tales como PC-DARTS [85], sharpDARTS [86], iDarts [87], y otros [17].

Por otra parte, el aprendizaje reforzado se ha sido utilizado desde inicios del NAS moderno, particularmente, a partir del trabajo de Zoph *et al.* [56] que utilizaron 800 GPUS por 2 días para obtener resultados competitivos con arquitecturas diseñadas manualmente en CIFAR-10 en 2017, con un 3.65 % de error. Los métodos de RL utilizan un agente que realiza búsquedas en un espacio de acción que es idéntico al espacio de búsqueda definido. La recompensa del agente, se basa en la estimación del desempeño de la arquitectura que se prueba, a partir de esto, se define una política y cómo optimizar la búsqueda, lo que puede ser realizado por una RNN [56] (el controlador). La idea es entrenar cada arquitectura de prueba, evaluarla y actualizar los parámetros de la RNN para maximizar el desempeño de la arquitectura que se busca, y para ello se utiliza REINFORCE [56], optimización proximal de políticas [88] (2.65 % en CIFAR-10) o Q-learning [58]. Un procedimiento más eficiente fue propuesto en 2018 por Pham *et al.* en ENAS [89], que utiliza transferencia de pesos entre redes, disminuyendo el tiempo de búsqueda, llegando a 2.89 % en CIFAR-10, en el mismo espacio de búsqueda de [56].

Otra forma de búsqueda utilizada son los algoritmos evolutivos que se explican en la siguiente sección.

3.3. Neuroevolución

La neuroevolución proviene de trabajos de hace más de tres décadas. En 1989 Miller *et al.* [60] utilizó algoritmos genéticos para diseñar y optimizar arquitecturas de redes y usó *backpropagation* para optimizar los pesos.

Este paradigma tuvo su resurgimiento en el último lustro como una alternativa a los modelos de RL [17] para el problema de NAS, y actualmente es una de las formas más utilizada para encontrar arquitecturas óptimas por su flexibilidad, simplicidad y capacidad [90] (aunque la parte de optimización en entrenamiento ahora se realiza mediante métodos GO como SGD).

Como se explicó previamente, las operaciones de los algoritmos genéticos son generación de población inicial, selección, *cross-over* o entrecruzamiento, generación de hijos y mutación. Es en estas operaciones que se diferencian los diferentes algoritmos que se han propuesto para NAS. Para la población inicial se puede utilizar arquitecturas simples [91], tomar arquitecturas de manera aleatoria del espacio de búsqueda [46, 90, 92] o utilizar diseños de buen desempeño conocidas como punto inicial [93].

El sistema de selección regula principalmente el problema de la explotación frente a la exploración [69]. Los sistemas de selección en NAS más utilizados son torneo [46, 90, 92, 94], ruleta [61, 95, 96], descartar el peor [90, 97, 98], descartar el más antiguo [90] o elitismo [46, 66, 99]. Torneo selecciona de la población total $P = \{a_1, a_2, ..., a_n\}$ un subconjunto aleatorio de tamaño $k < n$, $O = \{a_1, a_2, ..., a_k\} \subset P$, y luego escoge al mejor individuo de esa población; ruleta puede ser utilizada en base a *fitness* o ranking, en esta selección a los individuos se le calcula la probabilidad de ser seleccionado como $\frac{f}{\Sigma f_i}$ donde f corresponde al fitness del individuo y f_i el *fitness* del individuo i de la población. Descartar el más antiguo es una estrategia para no estancar en óptimos locales con individuos que permanecen mucho tiempo en la población, mientras que el elitismo transfiere los k mejores individuos a la siguiente generación, o en caso contrario descartar los peores k individuos también se utiliza. Es importante notar que en algunos casos, los métodos de selección se puede utilizar combinados, por ejemplo, [46] utiliza torneo y elitismo.

El operador entrecruzamiento se ejecuta posterior a la selección, normalmente el procedimiento se realiza en base a 2 padres pertenecientes a la población seleccionada [18], en algunos casos el entrecruzamiento puede no existir, al igual que la mutación [45]. La forma de realizar entrecruzamiento depende directamente de la forma de codificar los parámetros en el individuo. Las operaciones de entrecruzamiento normalmente utilizadas [67] son entrecruzamiento de pivote o *pivot cross-over* (también llamado entrecruzamiento de un punto o *one point cross-over*), entrecruzamiento de k-puntos o *k-point cross-over*, entrecruzamiento lineal o *linear cross-over* y entrecruzamiento uniforme o *uniform cross-over*. *One point cross-over* es un caso particular de *k-point cross-over*, que escoge k puntos dentro de la secuencia de genes, y toma los genes del padre 1 hasta el punto 1, luego del padre 2 hasta el punto 2, luego del padre 1 hasta el punto 3, etc, para ir completando los genes del hijo. *Linear cross-over* combina linealmente los genes de los padres con un factor $\alpha < 1$. *Uniform cross-over* selecciona intercaladamente los genes del padre 1 y padre 2.

La mutación es una forma de modificar los hijos generados para obtener genes que no existen en los padres, al igual que el entrecruzamiento, depende fuertemente del tipo de codificación. Los tipos de mutación típicamente usados son desplazamiento o *swapping*, inversión y mezcla o *scramble* [67]. El desplazamiento corresponde a tomar dos genes e intercambiar sus posiciones, la inversión normalmente corresponde a las variables binarias, donde se cambia del estado 1 al 0 o viceversa de manera aleatoria (en variables continuas o enteras, corresponde a escoger un valor aleatorio del rango de la variable), *scramble* cambia todos los genes de posición de manera aleatoria.

El ciclo de generación de hijos, mutación, cálculo de *fitness*, selección y reproducción, continúa hasta que se cumple algún criterio, el cual normalmente es el número de generaciones [45].

Los AG no son los únicos algoritmos utilizados en neuroevolución, es más, AG se reconoce como subcategoría de los Algortimos Evolutivos (EA), donde además se pueden encontrar *Genetic Programming*, y Evolutive Strategies (ES). Además de los AE, también *Swarm Intelligence* se presenta como una familia de soluciones para problemas de optimización de este tipo, en esta categoría se encuentran *Particle Swarm Optimization* (PSO) y *Ant Colony Optimization* (ACO). Otros métodos neuroevolutivos son *Differential Evolution* (DE), *Hill-Climbing Algorithm*, *Firefly Algorithm* (FA), *Artificial Inmune System* (AIS), Algoritmo Memético, y otros. La taxonomía de los algoritmos de neuroevolución se puede visualizar en la figura 3.4.

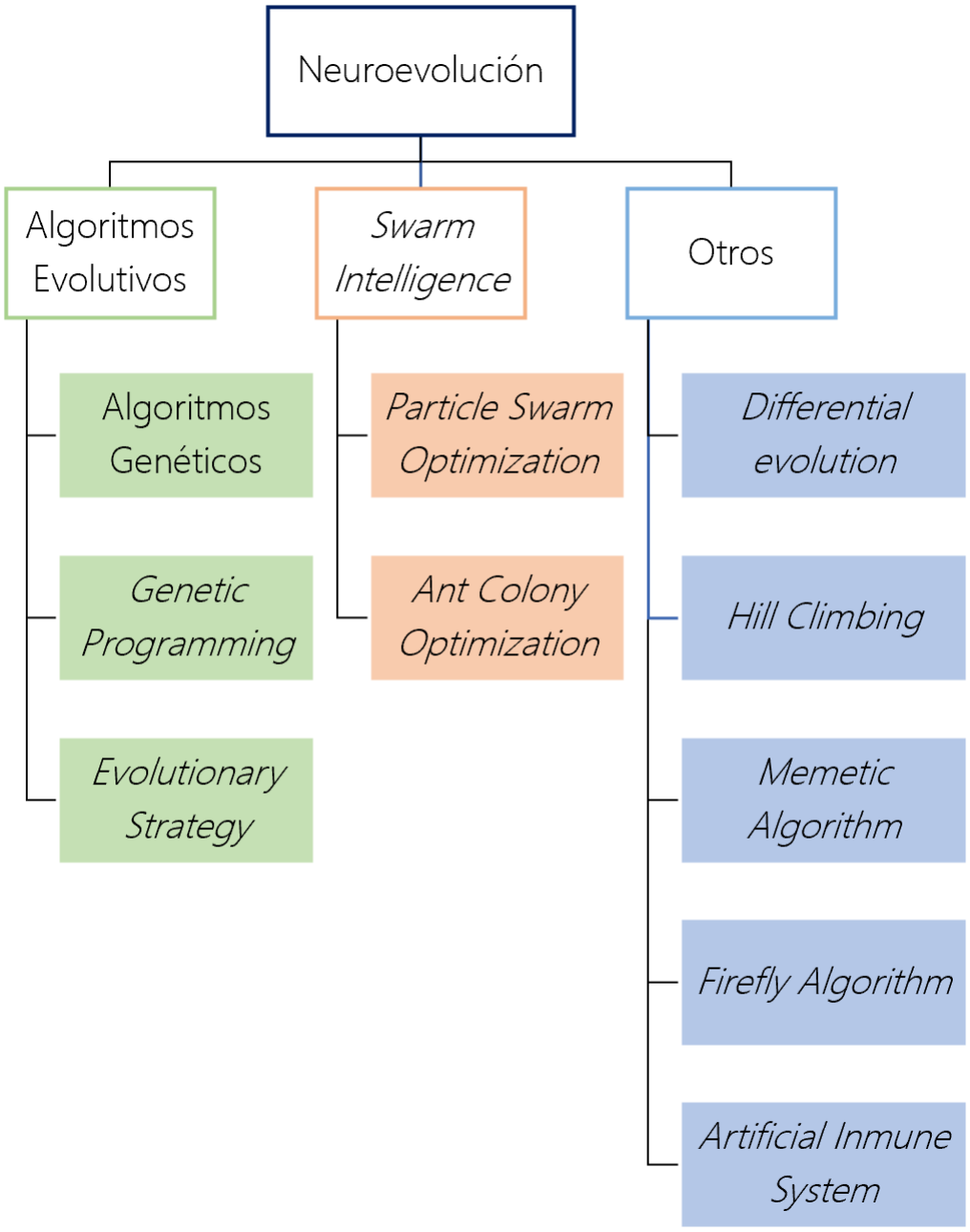


Figura 3.4: Diagrama taxonómico de neuroevolución [45].

GP es muy similar a los AG, presenta los mismos operadores de *cross-over* y mutación. La diferencia radica que con los GP, se enfocan principalmente en la estructura de la redes, además de generar un resultado más interpretable por su naturaleza [100]; [101] utiliza un árbol para generar la arquitectura de una red, desde las operaciones fundamentales de las CNN (*Pooling*, convolución y agregación) para encontrar redes en datasets de detección de peatones, detección de rostros y detección de autos. [102] utiliza una representación nodal para representar el fenotipo de las arquitecturas en un genotipo basado en GP Cartesiano (CGP), utilizando módulos funcionales como nodos, este método obtuvo dos tipos de redes

diferentes, CNN con un 6.75 % de error en CIFAR10 y RNN con un 5.98 % de error.

ES ha sido utilizado en menor medida, particularmente para la tarea *retinal vessel segmentation* en el contexto de NAS. El procedimiento ES genérico se utiliza en [103]; aquí se representa cada individuo como una variable de decisión X y con una desviación estándar σ. Ambos contienen n componentes:

$$[X, \sigma] = [x_1, x_2, \ldots, x_i, \ldots, x_n, \sigma_1, \sigma_2, \ldots, \sigma_i, \ldots, \sigma_n],$$

y la relación entre ambas variables está dado por

$$\sigma_i' = \sigma_i \cdot \exp\left(r' \cdot N(0,1) + r \cdot N_i(0,1)\right),$$
$$x_i' = x_i + \sigma_i \cdot N_i(0,1),$$

donde x_i, σ_i es la i-ésima componente de los padres, y (x_i', σ_i') es la i-ésima componente de los hijos ($N(0,1)$ es una distribución normal de media 0 y desviación estándar 1). r' es el coeficiente global y r es el coeficiente local, los que permiten generar nuevos individuos mediante antiguos con mutación Gaussiana. El procedimiento evolutivo es simple, en cada generación se modifica la variable de decisión X de cada individuo mediante mutación o recombinación y se mantienen los mejores individuos para la siguiente generación de manera iterativa. Para la forma de selección existe con dos paradigmas, (μ, λ) y $(\mu + \lambda)$. Ambos producen λ hijos desde μ padres, pero en el primer caso se selecciona a los mejores individuos únicamente desde los λ hijos, en el segundo caso se selecciona del total de población, por lo que es un método con elitismo permanente. Esta metodología se utilizó para evolucionar una variación de U-net y tuvo resultados competitivos en los dataset de segmentación DRIVE, CHASEDB1 y STARE.

Dentro de SI, se encuentra PSO un algoritmo que se basa en una población que busca en conjunto la mejor solución. En este caso se codifica los parámetros de búsqueda como un vector espacial que representa una partícula (o individuo), los cuales se mueven con una cierta velocidad dentro del espacio de búsqueda. Los parámetros de inercia y aceleración son los más importantes dentro de este algoritmo [100]. La idea general es que las partículas siguen al líder, que es la partícula que tiene el mejor *fitness* en la generación. EPSOCNN [92] utiliza PSO para evolucionar un bloque denso como *proxy* para la tarea de clasificación, buscando únicamente dos parámetros, luego al usar múltiples bloques secuencialmente se logra 3.74 % de error en CIFAR10. psoCNN [65] utiliza un tipo de codificación compleja para poder evolucionar arquitecturas de tamaño variable, logrando 0.32 % de error en MNIST y 14.28 % de error en MNIST-RD+BI.

ACO es un método basado en el movimiento de colonias de hormigas en la búsqueda de alimento, el algoritmo conceptual de origen biológico es el siguiente:

1. Una hormiga se mueve aleatoriamente.
2. Si la hormiga encuentra comida, retorna a la colonia dejando un rastro de feromonas.
3. Las feromonas atraen a hormigas cercanas, y seguirán el camino de la otra hormiga. Al volver, si lo hacen más rápido que siguiendo otro camino, la ruta se habrá fortalecido por más feromonas.

4. Este procedimiento se repite con retroalimentación positiva, haciendo más atractiva la ruta mas corta. En las rutas más largas se evaporarán las feromonas.

Este procedimiento se aplicó en NAS en DeepSwarm [104], donde se representan los operadores de las redes como nodos e ir a esos nodos son los caminos recorridos por las hormigas. El método se basa en que los caminos que van mejorando la red serán más escogidos por las hormigas que los peores al ir recorriendo el sistema. Este método logró en 11.31 % de error en CIFAR10 y 0.34 % de error en MNIST.

DE es un mecanismo muy similar a los algoritmos genéticos, pero explota información de 3 individuos para generar mejores candidatos, este método ha sido utilizado para evolucionar redes *Long short-term Memory* (LSTM), CNN y RNNs [100]. *Hill-climbing*, un método de búsqueda local, fue utilizado por Elsken *et al.* [98] en combinación con morfismos de redes y transferencias de pesos para optimizar CNN, obtuvieron 5.2 % y 4.4 % de error con un modelo único y un ensamble en CIFAR10. *Memetic Algorithm* es un caso más extendido de AG, donde además de la búsqueda global se incorpora una búsqueda local o *meme*, esto se realiza para refinar soluciones mediante la exploración de un vecindario; Lorenzo y Nalepa [105] usan este método para segmentación y además clasificación, obteniendo un 27.73 % de error en CIFAR10. FA es una variación de PSO que biomimetiza el comportamiento de luminosidad de las luciérnagas, las luciérnagas más brillantes (mejores soluciones) atraen a la menos brillantes (peores soluciones), una adaptación a NAS fue realizada en [106] obteniendo 3.3 % y 22.55 % de error en CIFAR10 y CIFAR100. AIS en ImmuNeCS [107] utiliza el enfoque de encontrar múltiples arquitecturas (comité) en vez de una, llegando a 4.38 % de error en CIFAR10.

3.4. Módulos de atención

El sistema visual humano utiliza mecanismos de atención para analizar y entender escenas complejas de manera eficiente y efectiva [47]. Esta idea transferida al dominio de AI se transforma en utilizar operaciones que sean análogas a la utilizadas en el cerebro, para dar más importancia a los aspectos relevantes de, por ejemplo, una imagen.

La atención se puede modelar de una manera muy simple [47]:

$$\text{Atención} = f(g(x), x),$$

donde $g(x)$ corresponde al proceso discriminante de focalizar regiones. Así $f(g(x), x)$ significa procesar la entrada x basándose en la atención $g(x)$.

La atención en AI se divide en principalmente tres categorías, espacial, temporal y de canal (también hay una cuarta relacionada a los datos, denominada branch) [47], dentro de la tarea de este trabajo, son relevantes las categorías canal-espaciales, ya que no se utilizan datos temporales. Dentro de esta categoría se explican a continuación algunos de los múltiples mecanismos: *Bottleneck Attention Module* (BAM), *Convolutiona Block Attention Module* (CBAM), *Triplet Attention* (TA), *Simple Attention Module* (SimAM) y *Self-Attention*.

BAM [108] es un mecanismo de atención diseñado para poder ser insertado modularmente a arquitecturas de CNNs. Se enfoca en mejorar las representaciones de las CNNs, compu-

tando la atención desde la parte espacial y de canal. Utiliza una convolución dilatada para incrementar el tamaño del *receptive field* de la sección espacial, acompañado de una estructura *bottleneck* para reducir el costo computacional. El mecanismo puede enfatizar o suprimir información en ambos canales, sin generar un costo computacional extra importante, sin embargo, falla en capturar información contextual de largo alcance [47].

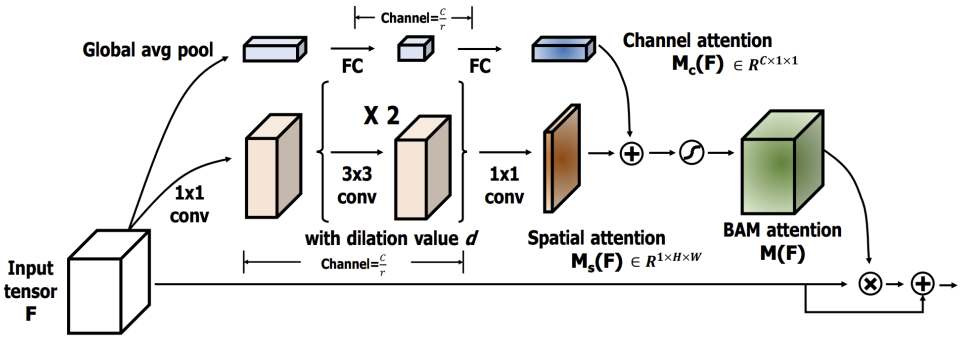


Figura 3.5: Diagrama de funcionamiento de BAM. Los *feature maps* pasan paralelamente por una operación de atención espacial y atención de canal. Obtenido de [108].

Al mismo tiempo que BAM se diseñó CBAM [109], un mecanismo similar a BAM pero más avanzado. CBAM desacopla la atención de canal y espacial para realizar un computo simplificado, utilizando además *global pooling* para obtener información espacial global. Al combinar los dos tipos de atención, CBAM utiliza el mecanismo para enfocar qué y dónde. Al igual que BAM, por su diseño simple, CBAM puede ser integrado a las CNNs generando un costo computacional marginal extra. Una limitante de CBAM es la utilización de convolución para producir la atención espacial, lo que puede limitar el campo receptivo [47].

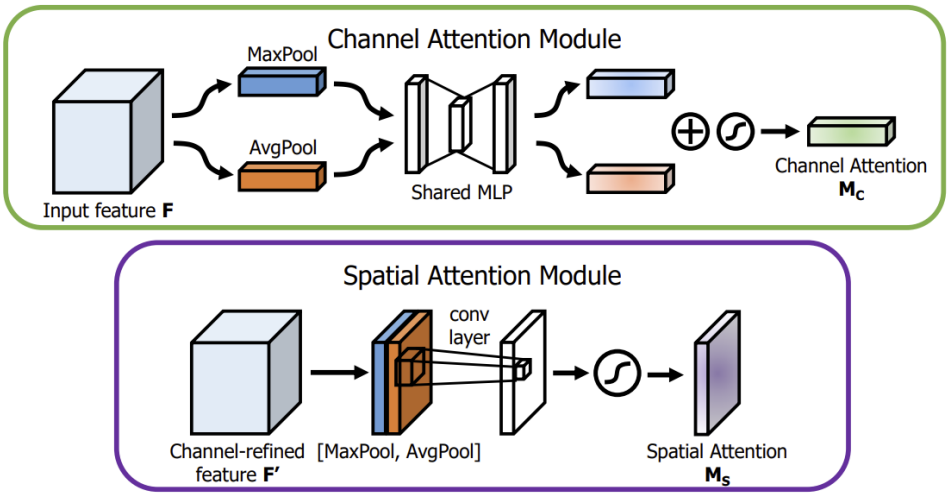


Figura 3.6: Diagrama de modulo espacial y temporal de CBAM. CBAM utiliza secuencialemente atención de canal y luego espacial. Obtenido de [109].

En CBAM y BAM los tipos de atención se calculan de manera independiente, ignorando las posibles relaciones que existen entre dominios [110]. En base a la información cruzada entre dominios fue diseñado *Triplet Attention*, un mecanismo que captura interacciones canalespaciales de manera efectiva y liviana. El nombre del mecanismo proviene de generar atención entre las tres dimensiones de un *feature map*, ancho, largo y canal en paralelo. Al igual que CBAM y BAM, TA puede ser incorporado de manera modular a las CNNs.

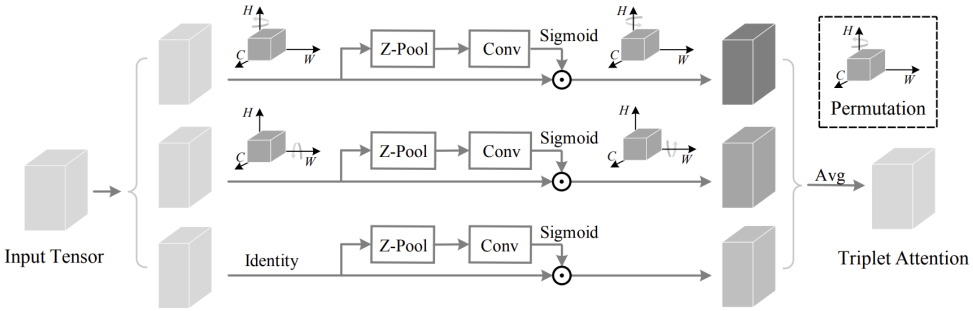


Figura 3.7: Diagrama de TA. Los *feature maps* se rotan en la dimensión H y W para obtener las atenciones cruzadas en tres direcciones. Obtenido de [110].

SimAM [111] tiene una aproximación biomimética desde la supresión de información espacial en primates [112]. En base a la supresión de informacion neuronal, diseñan un modulo que estima directamente las interacciones canal-espaciales 3D, sin expandir información 1D o 2D. El diseño de SimAM, a diferencia de los otros módulos, cuenta con únicamente un parámetro, que tiene un rango de funcionamiento tan amplio que resulta irrelevante su valor [111], haciéndolo en la práctica un módulo de atención sin parámetros.

Self-attention es un mecanismo de atención basado en *Transformers*, éste se fundamente en utilizar la secuencia de entrada para interactuar consigo misma, es decir, cada elemento de la secuencia interactúa con otros elementos, para capturar las relaciones entre estas. A pesar de ser un mecanismo que podría denominarse exclusivamente espacial [47], al combinarse con capas convolucionales se incorporan relaciones canal-espaciales de manera implícita [113]. Los mecanismos de *Vision Transformer* (ViT) han demostrado ser extremadamente efectivos en clasificación de imágenes de gran escala, sin embargo, su cantidad de parámetros hacen que sean redes costosas computacionalmente en comparación a CNNs puras [47]. La estructura clásica de *self-attention* en reconocimiento de imágenes se muestra en la figura 3.8.

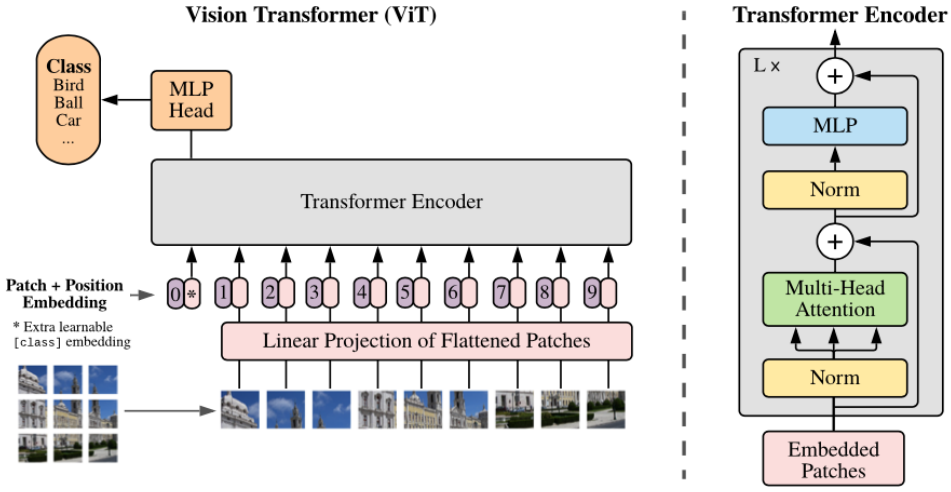


Figura 3.8: Diagrama de *Vision Transformer* (ViT). La imagen de entrada se divide en parches que se proyectan linealmente, luego esta proyección pasa por un mecanismo de auto-atención hacia una MLP de salida. Obtenido de [114].

3.5. Aprendizaje multitarea

El aprendizaje multitarea o *Multitask learning* (MTL) es un proceso para que una ANN aprenda varias tareas simultáneamente. En el contexto de CNNs ha sido utilizado, por ejemplo, para aprender a segmentar y clasificar imágenes al mismo tiempo [75]. Este paradigma no solo se limita a tareas que son similares, por ejemplo, GATO [115] es un modelo generalista que puede jugar Atari, etiquetar imágenes, chatear, utilizar un brazo robótico, y más [115].

El MTL biomimetiza la transferencia de aprendizaje entre dominios [116], para ello puede recibir una o varias fuentes de información, tener una o varias salidas. La versión estándar de aprendizaje multitarea, es un cambio simple a la función objetivo, ya que esta pasa a ser de un único objetivo a multiobjetivo. Dadas n tareas T_{i1}^{n}, cada una con un función de pérdida o *loss*

$$L_i(y_i, \hat{y}_i = f_i(h; \theta_i)),$$

donde y_i es el *ground-truth label* y $\hat{y}_i$ es el *label* que entrega para la tarea T_i, el MTL realiza una combinación (normalmente lineal) de las funciones de perdida para cada tarea individual, para generar una función de perdida global o *total loss*:

$$L_{total} = \sum_{i=1}^{n} w_i L_i(y_i, f_i(h; \theta_i)),$$

donde cada w_i son los pesos que determinan la importancia de la tarea i. De esta manera se optimizan los parámetros conjuntamente en una optimización multiobjetivo [116]. Es importante destacar que muchas veces las funciones de pérdidas tienen diferentes escalas, por lo que también se utilizan escalamientos al considerar la combinación lineal de las diferentes función de perdidas [19, 116].

En el contexto de NAS, *Neural Architecture Transfer* (NAT) [19] utilizó el MTL de manera exitosa, al integrar información de 10 set de datos diferentes para entrenar y evolucionar un modelo global de aprendizaje. NAT también se basa en dos importantes aspectos, utiliza AG para evolucionar un modelo tipo super-net pre-entranado y además realiza predicciones de desempeño de las redes para reducir el tiempo de búsqueda. Este trabajo obtiene resultados de estado del arte en datasets como ImageNet, CIFAR10 y CIFAR100 con 80.5 %, 1.6 % y 11.7 % de error respectivamente.

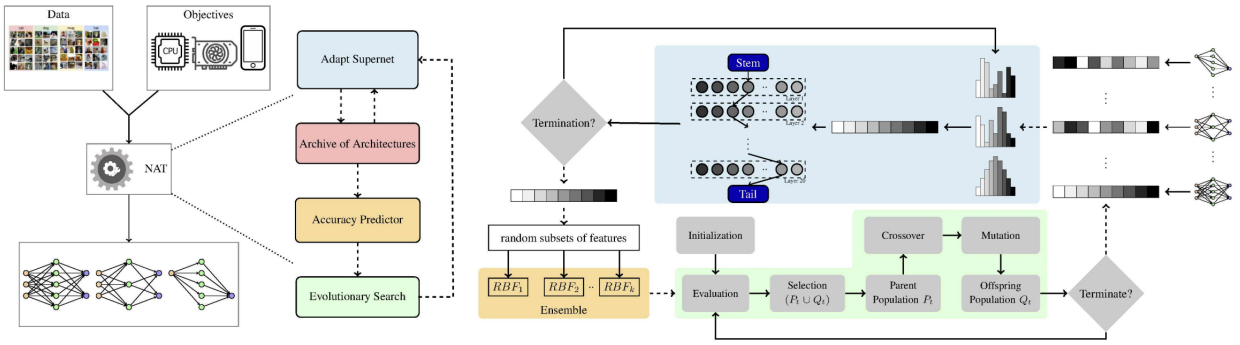


Figura 3.9: Diagrama de funcionamiento de NAT. Una super-red se adapta a partir de entrenamientos en varios conjuntos de datos, archivando las arquitecturas muestreadas. A partir de un predictor de desempeño se utiliza una búsqueda evolutiva que vuelve a adaptar la super-red. Obtenido de [19].

3.6. Bases de datos *benchmark*

En NAS aplicado a reconocimiento de imágenes las bases de datos más utilizadas son MNIST (con sus variaciones), CIFAR-10, CIFAR-100 e ImageNet [18, 17]. Otras bases de datos utilizados en menor medida, incluyen Fashion-MNIST, SVHN, Rectangle, Flowers102, Food-101, etc.

MNIST corresponde a un conjunto de imágenes de 28x28 con dígitos de 0 al 9 en escala de grises, sin fondo, sus variaciones incluyen dígitos con imágenes de fondo (MNIST-BI), dígitos con ruido de fondo (MNIST-RB), dígitos rotados (MNIST-RD) y dígitos rotados con imágenes de fondo (MNIST-RD + BI).

Figura 3.10: Ejemplo de MNIST y variaciones. De arriba hacia abajo: MNIST, MNIST-BI, MNIST-RB, MNIST-RD, MNIST-RD + BI y Fashion MNIST. Obtenido de [46].

CIFAR-10 es un conjunto de 60000 imágenes de 32x32 en RGB con 10 clases diferentes, 6 de animales y 4 de vehículos. CIFAR-100 también contiene 60000 imágenes de 32x32 en RGB, pero con 100 clases y 20 subclases.

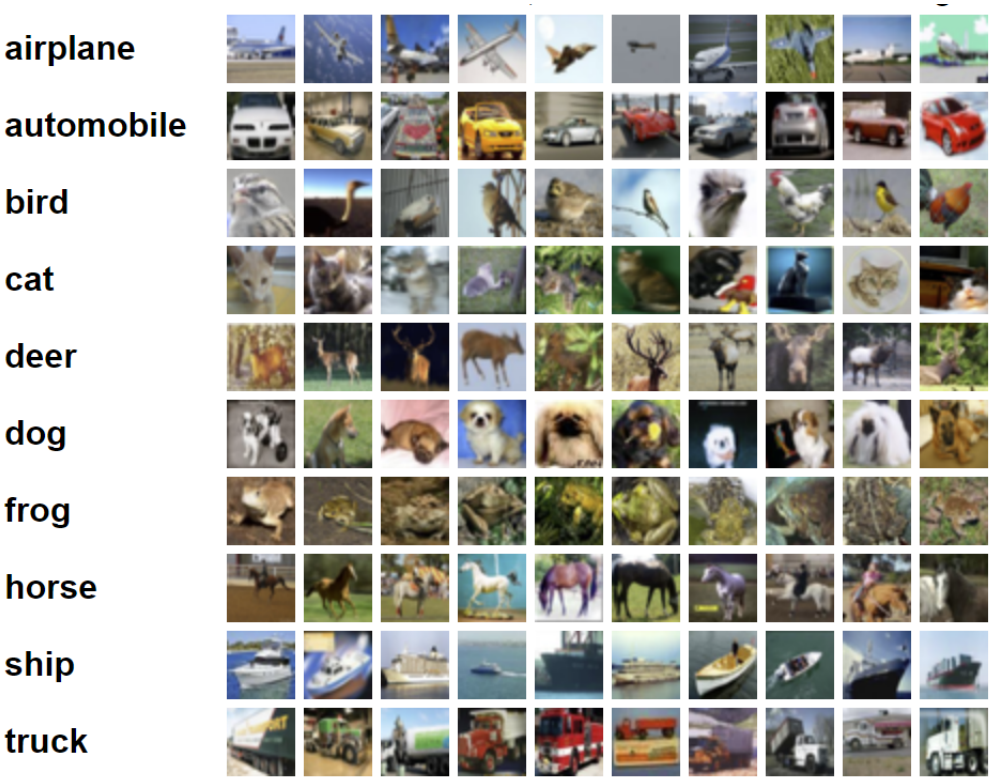


Figura 3.11: Ejemplos de imágenes de CIFAR-10 con sus diez clases. Obtenido de [117].

ImageNet es un conjunto de 1.45 millones de imágenes de 224x224 en RGB que contiene 1000 clases.

Figura 3.12: Ejemplos de imagenes de ImageNet. Obtenido de [118].

En la tabla 3.1 se comparan los distintos conjuntos de datos.

Tabla 3.1: Comparación de diferentes bases de datos.

Set	Tamaño de imagen	Tamaño *train*	Tamaño *test*	# de clases
MNIST	28x28x1	60000	10000	10
MNIST-RD	28x28x1	12500	50000	10
MNIST-RB	28x28x1	12500	50000	10
MNIST-BI	28x28x1	12500	50000	10
MNIST-RD+BI	28x28x1	12500	50000	10
CIFAR-10	32x32x3	50000	10000	10
CIFAR-100	32x32x3	50000	10000	100
ImageNET	224x224x3	1200000	150000	1000

3.7. Alcance del estado del arte y tesis propuesta

En el estado del arte analizado se presentan diferentes métodos para la realización de diseño automático de redes neuronales convolucionales con variadas metodologías, sin embargo, en los estudios analizados no se encuentran diseños bio-inspirados directamente, tampoco se encuentra la utilización de algoritmos genéticos multi-cromosómicos para la codificación de arquitecturas. La tesis propuesta busca ser un método holístico en la incorporación de diferentes ideas provenientes de la biología, incluyendo los algoritmos genéticos multi-cromosómicos, diseños basados en redes neuronales biológicas e incorporación de mecanismos de redes neuronales bio-inspirados como el aprendizaje multitarea, la atención y los parches neuronales.

Capítulo 4

Metodología

En este capítulo se presenta el conjunto de métodos utilizados en la tesis para comprobar la hipótesis inicial y cumplir con los objetivos. El capítulo comienza con la definición del conjunto de arquitecturas en el que se desarrollan las búsquedas, posteriormente la codificación elegida y los operadores del algoritmo genético diseñado. Luego, se presentan las mejoras basadas en biología a las arquitecturas encontradas mediante búsqueda automática. Finalmente, se explica el diseño de las pruebas y experimentos realizados.

4.1. Redes neuronales convolucionales inspiradas en arquitecturas biológicas retinotópicas

A partir de los diferentes flujos de información y lugares de procesamiento, se propone realizar una analogía del sistema biológico en dos escalas. En una escala macro, si solo se consideran las conexiones feed-forward, se observa que existe un flujo de información que se va integrando a través de las distintas secciones ventrales hasta llegar a IT. Este comportamiento lo podemos modelar como una red que crea una clasificación parcial, superficial o gruesa, mientras que las otras redes, equivalentes a los parches neuronales en IT, se especializan en clasificar de manera fina. Este tipo de comportamiento de especialización de redes neuronales, también se observa en, por ejemplo, la clasificación de objetos que realiza el giro temporal inferior [36, 72, 73, 78, 119].

Los diferentes núcleos de integración o procesamiento de información (V2, V3, V4, etc.) se pueden simular como un bloque de capas convolucionales densas. Mientras que los sectores intermedios iniciales y finales de flujo de información serán considerados como capas de a pares. Esto se realiza considerando que los sectores de integración tienen conexiones más complejas que los sectores iniciales y finales. Por otra parte, a partir de redes con arquitecturas similares, se simularán los parches neuronales, como clasificadores más finos de clases específicas.

Para la construcción de los modelos basados en redes neuronales biológicas para el reconocimiento de imágenes, se definen dos bloques de construcción fundamentales: capas convolucionales y bloques densos. Se construyen los modelos de la siguiente manera: las primeras dos capas convolucionales son seguidas por *max pooling, dropout y batch normalization*. A continuación, se incorporan secuencialmente los bloques densos, luego las capas convolucionales restantes se organizan en pares si es posible, siguiendo cada par el patrón del primer par.

Todas las capas convolucionales dentro de los bloques densos están construidas con batch normalization. En todos los operadores convolucionales se utiliza *padding* (con ceros) para evitar problemas de dimensionalidad.

Después de realizar todas las operaciones relacionadas con la convolución, se incorporan dos capas *fully connected* con *dropout* entre ellas.

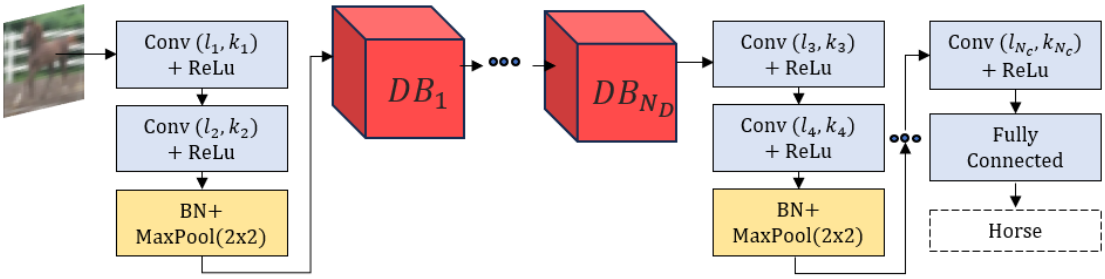


Figura 4.1: Diagrama de estructura base de arquitecturas. Los bloques azules representan capas convolucionales, los amarillos son reducción de dimensionalidad y los rojos bloques densos.

4.2. Algoritmo genético y codificación

Para construir los espacios de búsqueda de los diversos algoritmos, varios estudios proponen múltiples tipos de codificaciones de redes. Es importante elegir una buena codificación para definir un espacio de búsqueda robusto que permita a los algoritmos encontrar buenas soluciones, lo que es cierto incluso para la búsqueda aleatoria [46, 54, 57]. Una de las distinciones más relevantes es si esta codificación es de longitud fija o variable. En este trabajo se elige usar codificación de longitud variable, porque la codificación de longitud fija podría restringir la búsqueda donde las soluciones óptimas podrían ser redes más pequeñas o más grandes, un aspecto que es más relevante para los diseños de los parches neuronales.

Se propone un método de codificación basado en multi-cromosomas de longitud variable, que define parámetros independientemente para los bloques densos y las capas convolucionales, de modo que los cromosomas solo interactúen con aquellos de la misma clase, de la misma manera que ocurre en los sistemas biológicos. La codificación multi-cromosoma ha demostrado que el espacio de búsqueda puede subdividirse y converger más rápidamente en otros dominios [120].

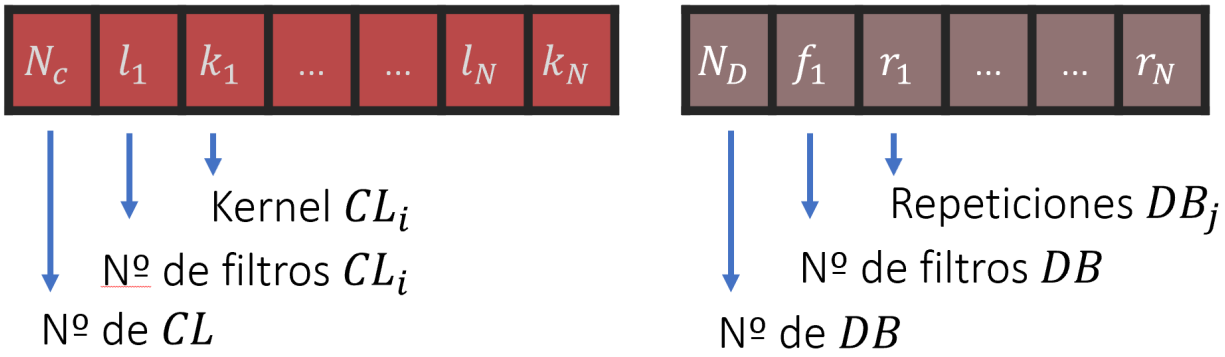


Figura 4.2: Diagrama de codificación de dos cromosomas del método propuesto. El cromosoma convolucional codifica el número de capas convolucionales y el número de filtros y tamaño del *kernel* de cada una. El cromosoma denso codifica el número de bloques densos, el número de filtros global y el número de repeticiones para cada bloque denso.

Se construyen las arquitecturas a partir de capas convolucionales en pares como en [61], donde los parámetros de búsqueda son el tamaño del kernel y el número de filtros por convolución. Se utiliza la misma representación que una lista de [61] donde se codifican N_c capas convolucionales como una lista de $2N_c + 1$ valores. El primer valor corresponde a N_c y los $2N_c$ valores son el par de l_i y k_i, números de filtros y tamaño del kernel de cada capa convolucional, de esta manera el cromosoma convolucional (CC) es

$$CC = [N_c, l_1, k_1, l_2, k_2, ..., l_{N_c}, k_{N_c}]. \tag{4.1}$$

Se define un bloque denso basado en DenseNet [15]: una capa convolucional representa una operación de *Batch normalization*+ ReLu + Convolución 2D. La construcción de los bloques se basa en dos capas convolucionales con tamaño de kernel de 1x1 y 3x3 (*Bottleneck Layer*) definidos por el número de filtros (denominado *growth rate*) y número de repeticiones de estas dos capas convolucionales. Cada capa convolucional recibe como entrada la concatenación de la entrada del bloque con cada salida previa de las capas convolucionales. Al final de las concatenaciones se realiza un reducción de dimensionalidad en 3 dimensiones, reduciendo a la mitad la cantidad de filtros, el ancho y el alto, proceso que se define como una capa *Transition Layer*. La figura 4.3 muestra un bloque denso completo, BL se refiere a *Bottleneck Layer*, TL indica *Transition Layer*, el operador $\oplus$ indica concatenación.

En el algoritmo de búsqueda se busca el número de bloques densos, el número de filtros (solo un valor para todos los bloques) y el número de repeticiones para cada bloque. Se fija un valor de búsqueda para números de filtros, como se muestra en [92], y no las repeticiones para cada bloque, porque reduciría demasiado el espacio de búsqueda y en arquitecturas hechas a mano los resultados indican que las repeticiones variables funcionan mejor [15]. Se define el cromosoma denso (DC) como un vector donde el primer índice es el número de Bloques Densos (N_D), el segundo es el número de filtros de cada bloque (F_d) y del índice 3 a $N_d + 2$ representa el número de repeticiones en el bloque j,

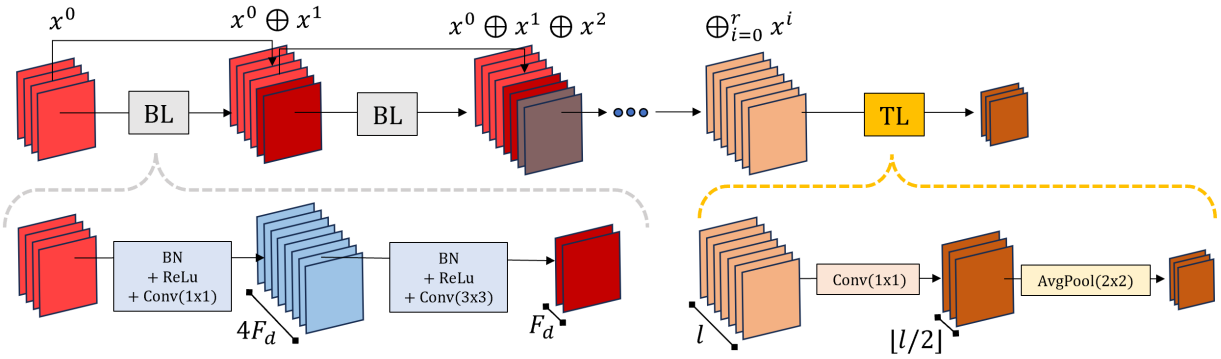


Figura 4.3: Diagrama estructural de un bloque denso. La sección izquierda (en gris) muestra el diseño de conexiones densas y la derecha (en amarillo) la reducción de dimensionalidad. La formulación de bloque denso es obtenida de [15].

$$DC = [N_D, F_d, r_1, r_2, ..., r_{N_d}]. \tag{4.2}$$

Con esta codificación multicromosómica se definen 3 diferentes operadores de mutación basados en [61]. La mutación puntual modifica solo un valor de un cromosoma excluyendo N_c y N_d, cambiándolo aleatoriamente por otro valor dentro de los rangos especificados. La mutación de tamaño cambia N_c o N_d. Si reduce N_c, se eliminan los últimos genes; en cambio, si aumenta N_c, se asignan dos genes adicionales al final del cromosoma con valores aleatorios para l_{N_c} y k_{N_c}. Se aplica la misma idea si N_d aumenta o disminuye, eliminando los últimos genes que contienen el número de repeticiones o añadiendo un valor aleatorio al último nuevo gen. La mutación *Swap* intercambia genes dentro del mismo cromosoma. Si el intercambio se realiza en el cromosoma convolucional, intercambia los pares l_i, k_i con l_j, k_j para un i, j asignado aleatoriamente donde $i \neq j$. Si el intercambio se realiza en el cromosoma denso, se aplica el mismo principio intercambiando aleatoriamente dos genes que representan el número de repeticiones. Para todos los operadores de mutación, existe una probabilidad del 50/50 de mutación del cromosoma convolucional o denso. Para todos los genes nuevos generados o cambiados se utiliza una distribución uniforme aleatoria. La forma en que se determina la selección de índices en todas las mutaciones también es aleatoria uniforme.

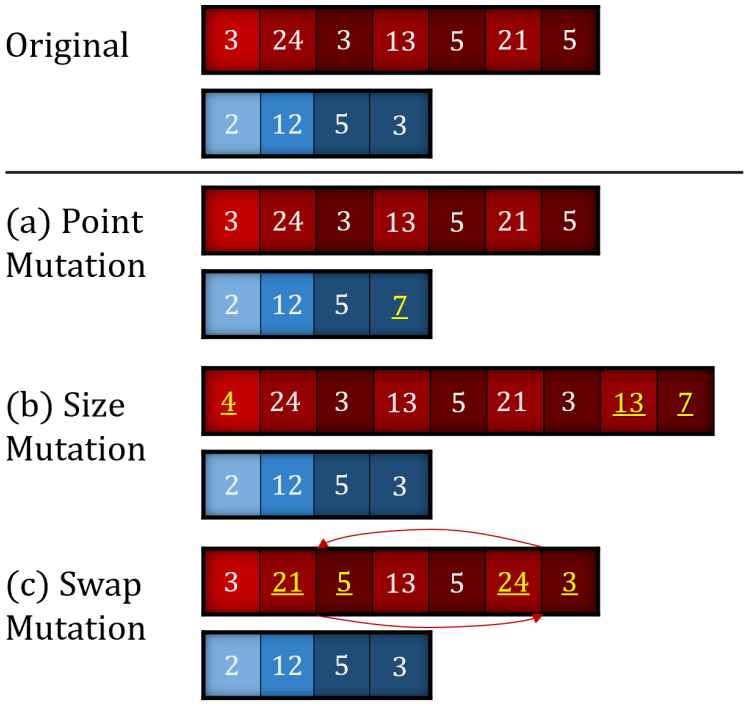


Figura 4.4: Ejemplo de operadores de mutación. A partir de un individuo original, se muestra las operaciones de (a) mutación puntual, (b) mutación de tamaño y (c) mutación *swap*.

Las operaciones de entrecruzamiento se realizan cromosoma por cromosoma con dos posibles operadores. *Pivot crossover* elige una posición aleatoria p entre 2 y $min(N_1, N_2)$, donde N_1 y N_2 son la longitud del cromosoma, y luego se generan descendientes eligiendo los genes de un progenitor desde el índice 2 hasta p y los genes del otro desde p hasta el final de los genes. Este cruce podría generar 2 posibles descendientes, pero se incrementa por la selección aleatoria de uno de los dos cromosomas densos, dando 4 posibles descendientes. *Linear crossover* genera un valor aleatorio entre 0 y 1 y hace una combinación lineal de cada gen excluyendo los encabezados que indican el número de capas y bloques.

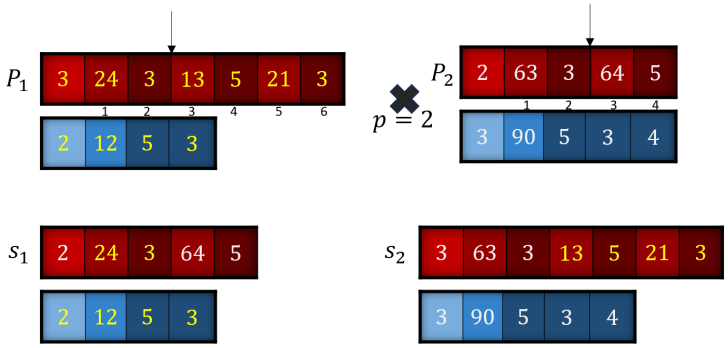


Figura 4.5: Ejemplo de *Pivot cross-over* con $p = 2$.

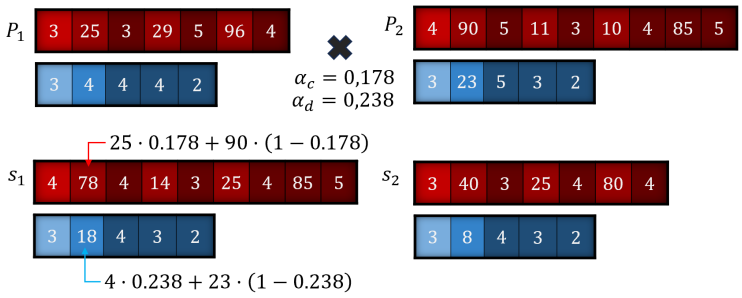


Figura 4.6: Ejemplo de *Linear cross-over* con $\alpha_c = 0.178$ y $\alpha_d = 0.238$.

Para el valor de *fitness*, cada modelo se entrena desde cero y su *accuracy* sobre el conjunto de validación en la última época se establece como el valor de *fitness*.

En la evolución, el entrenamiento de los modelos se realiza utilizando el optimizador ADAM con sus valores predeterminados en *Tensorflow*, se usa *categorial cross entropy* como función de pérdida. Se utiliza *padding* Same, activación ReLu e inicializador de pesos *Normal He* en todas las capas convolucionales, incluidas las que están dentro de los bloques densos.

Para el proceso de selección se utiliza torneo, ya que es un sistema de selección que genera un compromiso entre exploración y explotación, en comparación con la selección de ruleta, que se inclina hacia la explotación (podría converger prematuramente si la población carece de diversidad), mientras que la selección por clasificación está influenciada por la exploración (la diversidad podría ser permanentemente alta) [69, 121].

Para la siguiente generación de la población, se realizan varias selecciones con torneos para elegir a los padres del entrecruzamiento. El proceso de entrecruzamiento siempre crea 2 hijos y se realiza hasta que se alcanza el tamaño de la población. También, se utiliza elitismo para transferir a los mejores n_{elite} individuos a la siguiente generación, eligiendo aleatoriamente n_{elite} descendientes y reemplazándolos. El criterio para que el algoritmo genético termine está determinado por el número de generaciones máximas predefinidas.

Debido a la limitada capacidad de cómputo, entrenar al máximo rendimiento las CNNs mientras están dentro del GA es poco práctico y se prefiere usar un número limitado de épocas [17]. Aunque no se entrenen las redes a su máximo potencial, se está utilizando una buena estimación de clasificación entre redes cuando no se usan épocas extremadamente bajas de entrenamiento [46].

El sistema completo de codificación y búsqueda se denomina *Multi-chromosomic Genetic Algorithm* (MCGA).

4.3. Mejoras a redes basadas en biología

Para mejorar aún más el rendimiento de las arquitecturas de redes propuestas, se utilizan diferentes métodos inspirados en la biología que se incorporan modularmente a los modelos. No se incluyen estos módulos dentro del algoritmo de NAS porque podrían aumentar el tamaño de los modelos, lo que posiblemente llevaría a tiempos de entrenamiento más largos,

haciendo así más lenta la búsqueda.

4.3.1. Módulos de atención

La organización retinotópica en el cerebro es clave para mapear y enfocarse en partes específicas de una imagen para lograr el reconocimiento de objetos o la comprensión de formas [24] y, por ello, está estrechamente relacionada con la atención, especialmente en humanos [22, 30].

De lo múltiples mecanismos de atención modulares, se evalúa el desempeño al incorporar Simple Attention Module (SimAm), por su base biológicamente inspirada, y porque no requiere optimización de parámetros internos.

4.3.2. Multitarea (aprendizaje lateral)

Para el equivalente al aprendizaje lateral biológico, se realiza un aprendizaje multitarea para mejorar aún más el rendimiento y la regularización, como se muestra en varios trabajos [116] [122] [19]. Se decide usar el mismo conjunto de datos con etiquetado múltiple en lugar de varios conjuntos de datos para no aumentar la memoria utilizado, de manera que no se incremente significativamente el tiempo de entrenamiento del o los modelos finales.

4.3.3. Parches neuronales y jerarquía

Como se muestra en varios estudios realizados en primates [36, 78], la corteza inferotemporal tiene varios mapas del espacio de objetos que se utilizan en la tarea de reconocimiento de objetos [119]. El cerebro de los primates y humanos utiliza circuitos o parches neuronales diferentes, pero estrechamente relacionados, para lograr tareas macro y micro [119]. Por ejemplo, la neuroimagen en estudios humanos ha revelado diferentes patrones de respuesta a diferentes orientaciones faciales, mostrando que diferentes regiones se encargan del procesamiento de rostros dependiendo de la orientación [72, 73, 77].

Estos hallazgos han inspirado soluciones para la tarea de reconocimiento facial [123] a través de la creación de varias redes especializadas que representan parches.

Se utiliza la misma idea aplicada al reconocimiento de imágenes. Para simular la idea del parche neuronal, se busca usar varios modelos para las subtareas que son ramificaciones del primero. Se realizan evoluciones por separado de las subtareas, de manera de obtener modelos especializados en la tarea fina.

4.4. Diseño de pruebas y experimentos

Los bloques densos y las CNNs se utilizan ampliamente en diferentes tareas, como en el procesamiento del lenguaje natural: análisis de sentimientos [124], análisis de imágenes médicas [125], clasificación musical [126], segmentación de imágenes [127] y otras. Se decide usar el método y las arquitecturas propuestas para la clasificación de imágenes, ya que los métodos evolutivos se utilizan comúnmente en NAS aplicado a imágenes [46, 61, 66, 94].

Se realizan principalmente tres tipos de experimentos. Comparación de los tipos de arquitecturas propuestas, con otros tipos de arquitecturas que tienen características y configuraciones de hiperparámetros similares. También se realizan comparaciones con el espacio de

búsqueda utilizado, sus tamaños y propiedades.

En segundo lugar, comparar el MCGA propuesto con una línea base de *Random Search*,y contra diferentes métodos propuestos (evolutivos, de gradiente, *RL*, etc.) del estado del arte.

Finalmente, se experimenta con diferentes métodos para mejorar aún más el mejor modelo encontrado con el MCGA, incluyendo mecanismos de atención, aprendizaje multitarea y parches neuronales.

4.4.1. Base de datos

Para todos los experimentos, se utiliza el conjunto de datos CIFAR10: 60000 imágenes a color de 32x32 (3 canales) en 10 clases, con 6000 imágenes por clase. Hay 50000 imágenes de entrenamiento y 10000 imágenes de prueba. Este conjunto de datos ha sido ampliamente utilizado como punto de referencia para *NAS* en la tarea de clasificación de imágenes [18, 17]. Se prefiere utilizar esta base de datos a MNIST, pues este conjunto se considera en desuso, porque incluso modelos muy simples logran <1 % de error [17]. ImageNet se descarta por complejidad computacional.

La segunda etiqueta utilizada en multitarea en CIFAR10 corresponde a las superclases vehículos y animales, con 20000 y 30000 ejemplos de entrenamiento respectivamente.

Se comparan las arquitecturas finales encontradas con otros estudios de contemporáneos con diseño de arquitectura automatizados o hecho a mano, que además se evalúan en el mismo conjunto de datos.

4.4.2. Espacio de búsqueda, GA/RS hiperparámetros

Se define el espacio de búsqueda de este trabajo basándose en hiperparámetros dimensionales o estructurales de la red: para el cromosoma convolucional se optimiza el número de capas convolucionales, el número de filtros de salida y el tamaño del kernel, por otro lado, para el cromosoma denso los hiperparámetros de búsqueda son el número de bloques densos, el número de filtros densos (*growth rate*) y el número de repeticiones dentro de cada bloque.

Los hiperparámetros fijos más importantes son: función de activación *ReLu* para todas las operaciones convolucionales, *zero-padding* para evitar problemas de dimensionalidad, inicialización *He Normal* para todos los pesos y una capa *FC* final fija de 64 neuronas con un *dropout* del 40 %. Para los bloques densos, la compresión se fija en 0.5 y la reducción de dimensionalidad se realiza con un *mean pooling* de tamaño 2x2.

Para el MCGA propuesto se utiliza una inicialización aleatoria de individuos dentro de las configuraciones de hiperparámetros definidas y un tamaño de población de 20 durante 20 generaciones. Se define el *fitness* de los individuos como su *accuracy* de validación directamente después de entrenarlos con el optimizador *ADAM* en configuraciones predeterminadas con 16 épocas sin técnicas de aumento de datos, el *batch size* se fija en 256. El número de épocas se define por un equilibrio entre el tiempo de cálculo y la precisión, pero valores cercanos a 18 son indicadores de rendimiento relativamente buenos como se muestra en [46]. La selección de individuos que pueden generar descendencia utiliza un torneo K=5. También se utiliza el

Tabla 4.1: Parámetros de búsqueda MCGA.

Parámetro	**Rango/valor**
Número de capas CNN	[2,4]
Número de filtros CNN	[4, 64]
Tamaño kernel CNN	[3, 9]
Número de bloques DNN	[2, 3]
Número de filtros DNN	[6, 36]
Número de repeticiones DNN	[4, 16]
Tamaño población	20
Número de generaciones	20
Tasa de mutación	0.3
Tasa de cross-over	0.8
Épocas entrenamiento	18

elitismo con de tamaño 1, es decir, el individuo con mejor rendimiento se mantiene para la siguiente generación. Los valores de probabilidad de entrecruzamiento y mutación son 0.9 y 0.3 respectivamente.

Para la búsqueda aleatoria, los genes de cada individuo para todas las generaciones se definen mediante un muestreo uniforme de los rangos de hiperparámetros. Se conserva siempre el mejor individuo en cada generación.

4.4.3. Estudios de ablación

Con el mejor modelo encontrado a través del MCGA, se realizan varias mejoras para aumentar el desempeño con métodos inspirados en la biología. Primero, se utilizan como referencia modelos puramente basados en capas convolucionales y en bloques densos.

Se ejecutan pruebas con y sin técnicas de aumentación de datos para la regularización. El *Data Augmentation* (DA) se realiza a través de la composición de diversas técnicas. Para ello también se realiza una búsqueda tipo *grid-serach*, de manera de optimizar los parámetros de aumentación.

Se utiliza SimAM el parámetro por defecto recomendados, es decir, $\lambda = 10^{-7}$ [111]. La integración del módulo de atención se realiza antes de la operación de concatenación en todos los bloques densos.

Para la mejora mediante multitarea, la tarea primaria (T_1) se define como la clasificación de las 10 clases de CIFAR10. Mientras que la tarea secundaria (T_2), se enfoca en clasificar de manera gruesa las categorías de CIFAR10, en animales y vehículos. De esta forma, la función total de pérdida, se define como una composición lineal de las pérdidas de ambas tareas y se buscan parámetros óptimos de (w_1, w_2):

$$L_{\text{total}} = w_1 L_1(y_1, \hat{y}_1) + w_2 L_2(y_2, \hat{y}_2).$$

Para obtener diferentes modelos especializados para las dos tareas específicas subsiguientes de clasificar animales y vehículos, dos búsquedas con MCGA se realizan con parámetros idénticos, de manera de encontrar un modelo especializado en la clasificación de animales y otro en clasificación de vehículos. El mejor modelo obtenido del primer *MCGA* se utiliza como clasificador de las clases gruesas.

Capítulo 5

Resultados y discusión

A continuación, se muestran los resultados de los diferentes experimentos. Parar todos los experimentos, a menos que se explicite lo contrario, se entrenan con *batch size* 256, utilizando una función de perdida *Focal Loss* con parámetro $\gamma = 3.0$, con un entrenamiento mixto de 300 épocas con optimizador ADAM con tasa de aprendizaje 0.001 y 100 épocas con SGD con learning rate 0.01 que a la época 60 pasa a 0.001. Los experimentos de incorporación de atención, multitarea y parches neuronales se realizan con aumentación de datos, con los parámetros encontrados en la tabla 5.7. En los experimentos de comparación con métodos del estado del arte se utilizan las métricas de error de clasificación, número de parámetros y tiempo ejecución en dias-gpu en base a las propuestas de comparación de [17, 18].

5.1. Espacio de búsqueda

Se comparó el tamaño de espacio de búsqueda y las restricciones con trabajos del estado del arte, es importante destacar que calcular el tamaño de espacio de búsqueda es complejo, por la cantidad de combinatorias y variabilidad de arquitecturas, por lo que se presentan estimados (el único reportado corresponde a SOBA). La comparabilidad se define como espacios cuyas arquitecturas puedan generar densidad y no utilicen celdas. La tabla 5.1 indica los tamaños y características de los espacios de búsqueda analizados.

Tabla 5.1: Comparación espacios de búsqueda.

Modelo	Tamaño del espacio de búsqueda	Número de parámetros/operaciones de búsqueda	Usa superred	Espacio de búsqueda comparable	Usa surrogate model
NASNet-A	$> 10^{10^{13}}$	> 10	✗	✗	✓
AmoebaNet	$> 10^{10^{13}}$	> 10	✗	✗	✗
Genetic Programming CNN	$> 10^{150}$	> 10	✗	✗	✓
2LGA	$> 10^{100}$	12	✗	✗	✗
SOBA	$7.1 \cdot 10^{87}$	9	✗	✗	✗
Large-scale Evolution	$> 10^{43}$	> 10	✗	✗	✓
Hierarchical-EAS	$> 10^{43}$	> 10	✗	✗	✓
Johnson	$> 10^{25}$	3	✗	✗	✗
CNN-GA	$> 10^{21}$	> 5	✗	✓	✗
DARTS	$\sim 10^{19}$	> 8	✓	✗	✓
HGAPSO	$\sim 1.25 \cdot 10^{8}$	3	✗	✓	✗
EPSOCNN	1560	2	✗	✓	✗
MCGA	$4.89 \cdot 10^{15}$	6	✗	NA	✗

5.2. Ablación tipo de modelos

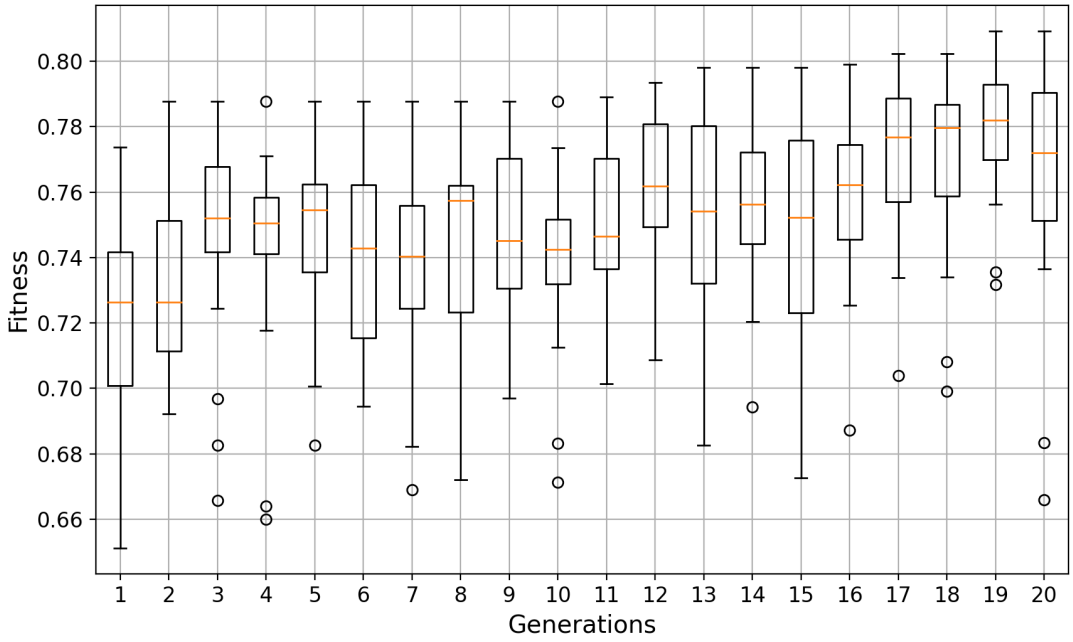
Para verificar que el individuo multicromosoma provee una arquitectura con mejor desempeño que cada cromosoma de manera individual, se realiza un estudio de ablación verificando el desempeño del modelo solo convolucional, modelo solo denso y modelo mixto, para ello se utilizan los parámetros encontrados por el MCGA. No se utiliza aumentación de datos. Los resultados de desempeño se indican en la tabla 5.2

Tabla 5.2: Resultados de ablación tipos de modelos

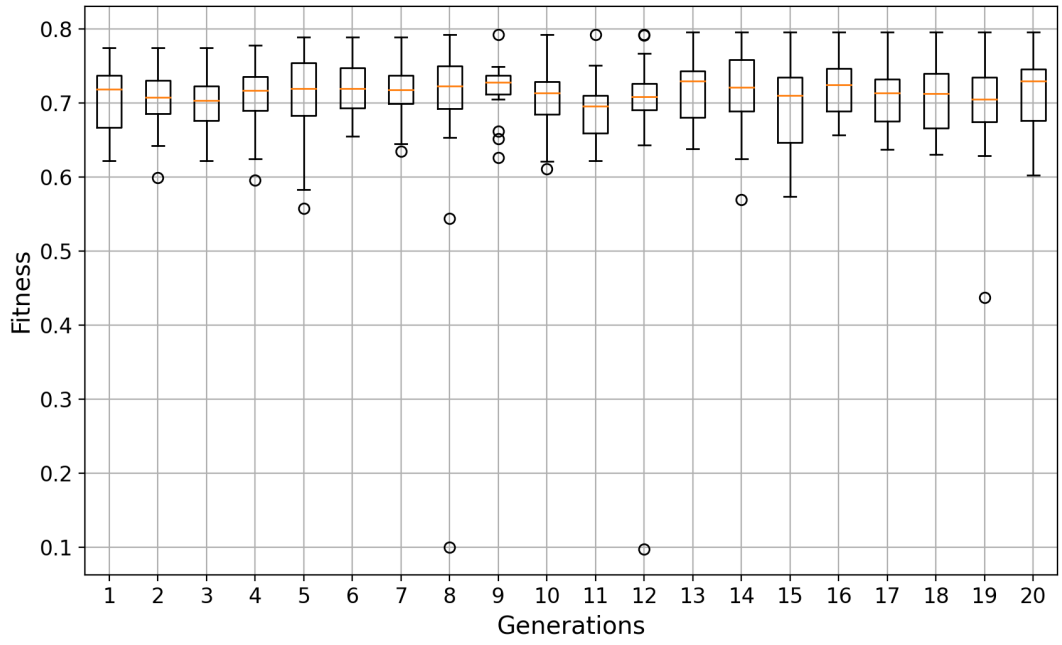
Modelo	**Test Accuracy**
Solo CNN	76.38 %
Solo DNN	88.45 %
Mixto	90.05 %

5.3. MCGA y *Random Search*

Como se indicó en la sección previa, RS resulta ser un algoritmo muy competitivo en NAS [17, 18, 82], por esto, se comparó el algoritmo genético diseñado en el espacio de búsqueda propio con respecto a RS. La figura 5.1 indica la distribución de *fitness* para cada generación, las figura 5.2 y 5.3 indican el *fitness* del mejor individuo y promedio para cada generación respectivamente. Las figuras 5.4 y 5.5 muestran la evolución de los parámetros de búsqueda para MCGA y RS respectivamente.



(a) MCGA.



(b) RS.

Figura 5.1: MCGA vs RS: Evolución distribución de fitness (*accuracy*) por generación.

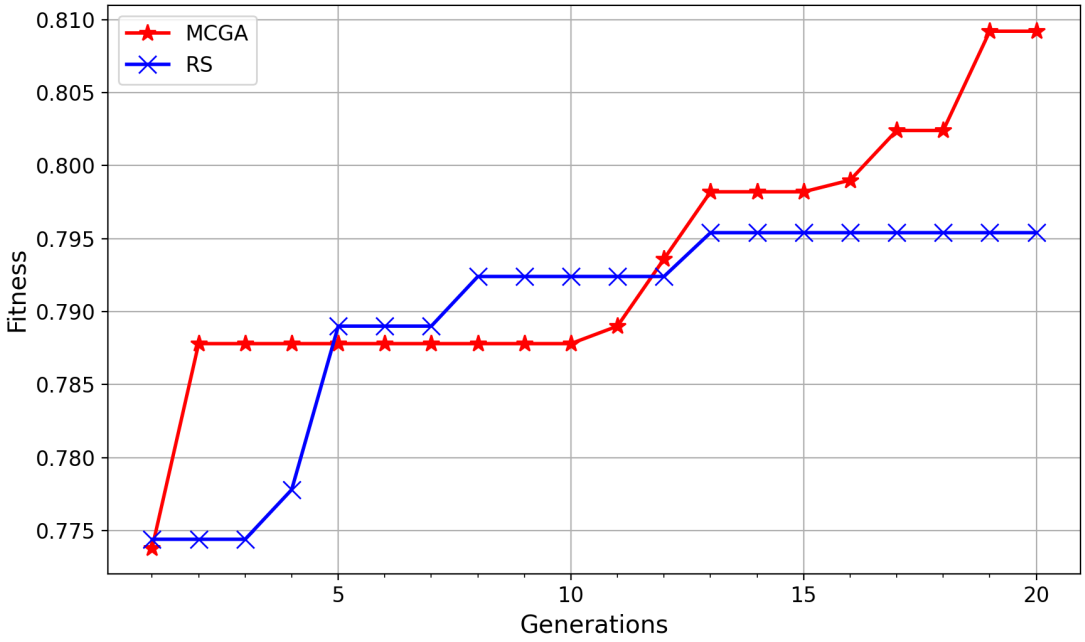


Figura 5.2: MCGA vs RS: Evolución individuo elite por generación.

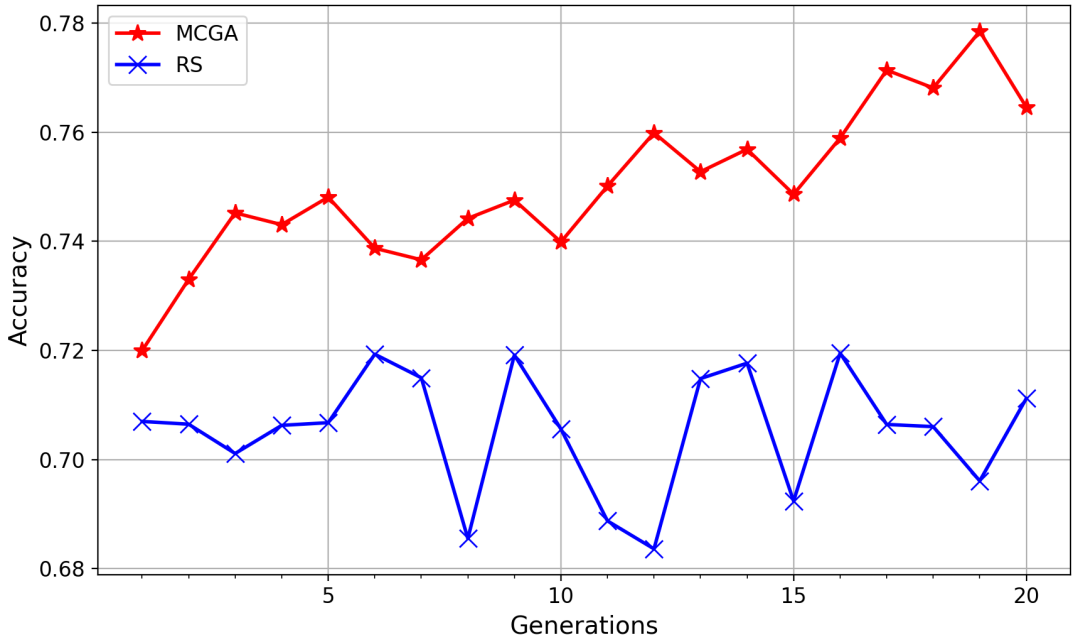


Figura 5.3: MCGA vs RS: Evolución *fitness* promedio por generación.

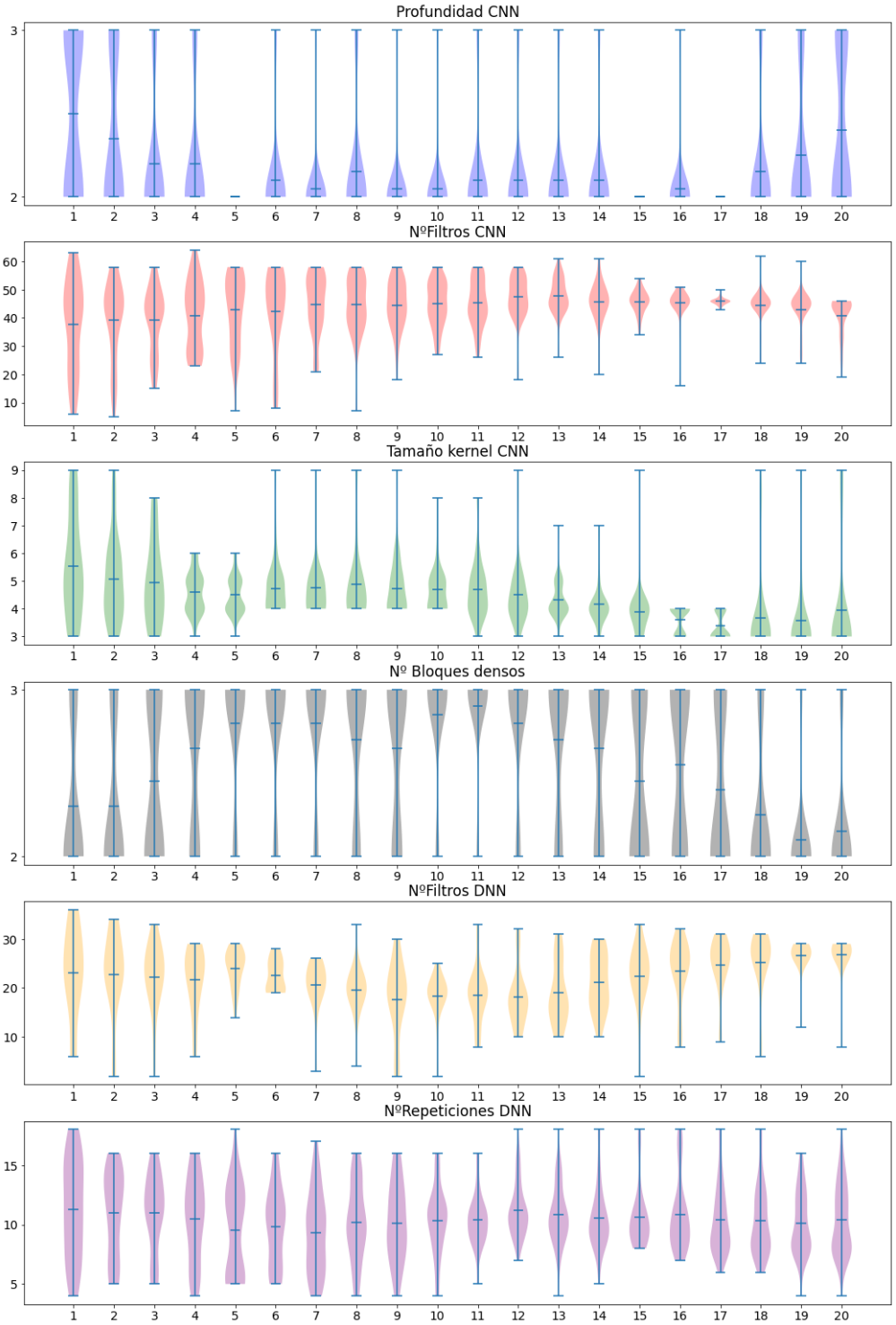


Figura 5.4: MCGA vs RS: Evolución de distribución de parámetros de búsqueda por generación, con MCGA.

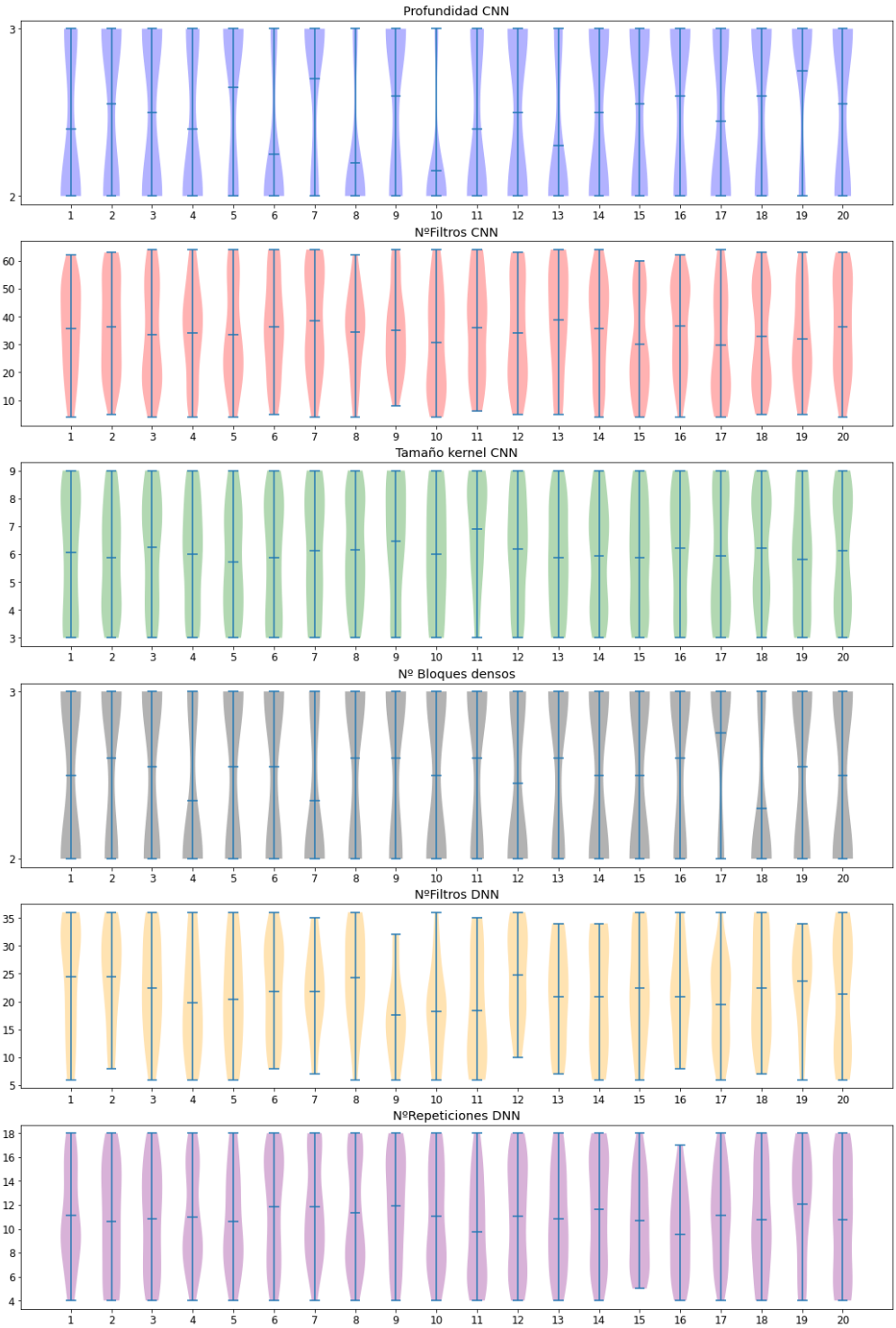
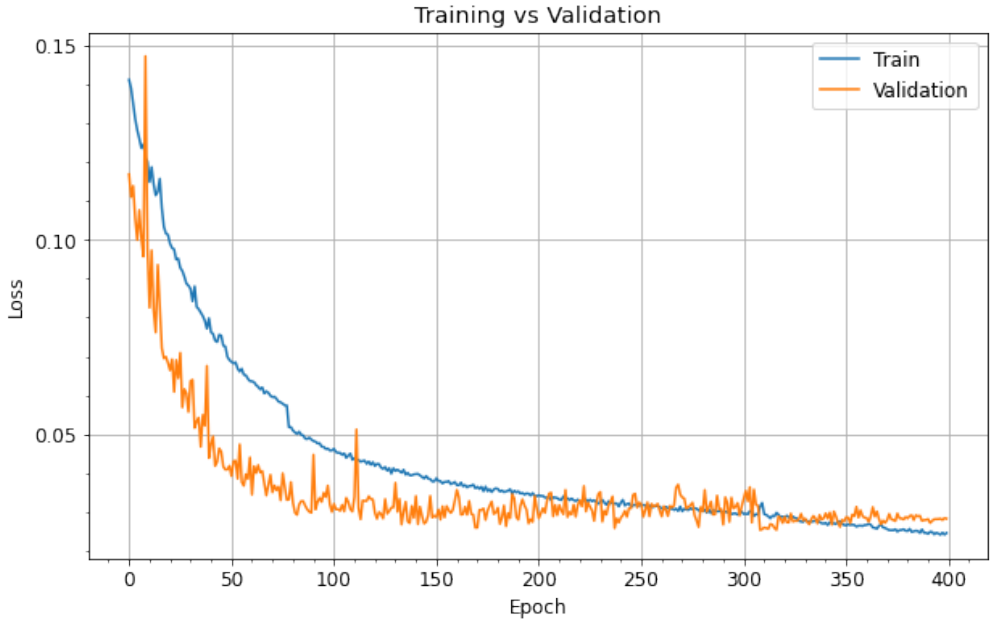


Figura 5.5: MCGA vs RS: Evolución de distribución de parámetros de búsqueda por generación, con RS.

5.3.1. Mejor modelo evolucionado

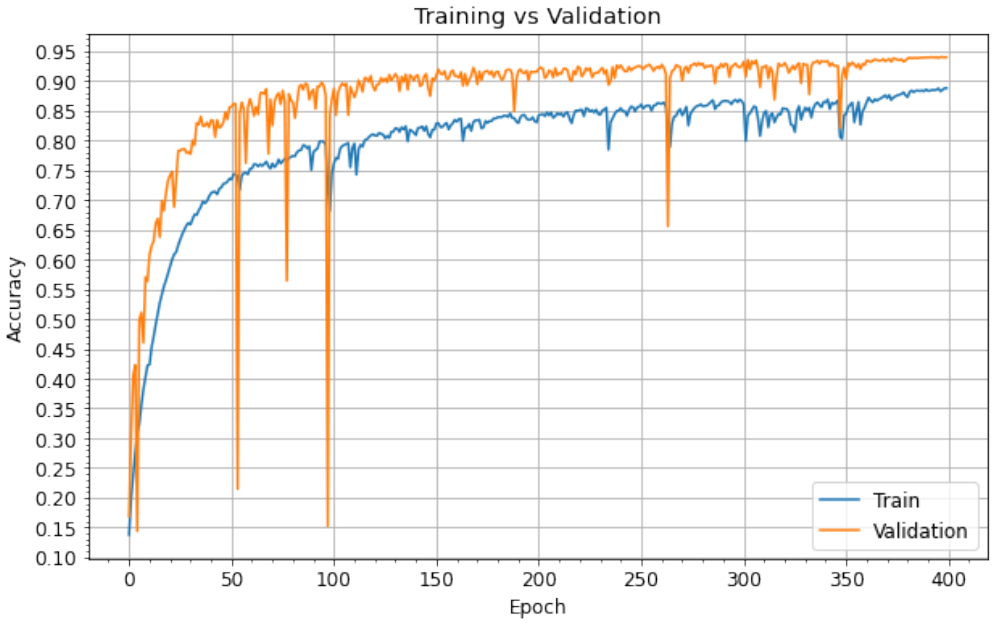
A los modelos finales encontrados se les realiza un entrenamiento estándar con optimizador ADAM con learning rate de 0.001, batch size 256, por 300 épocas. Además, un *fine tuning* con optimizador SGD por 100 épocas, con learning rate de 0.01 que decae a 0.001 en la época 60. Las figuras 5.6 y 5.7 indican las curvas de entrenamiento y validación, para el mejor modelo encontrado con MCGA y RS respectivamente. Se utilizan los parámetros de aumentación de datos indicados en la sección 5.4, tabla 5.7.

(a) Curva de entrenamiento *accuracy* en función de época.

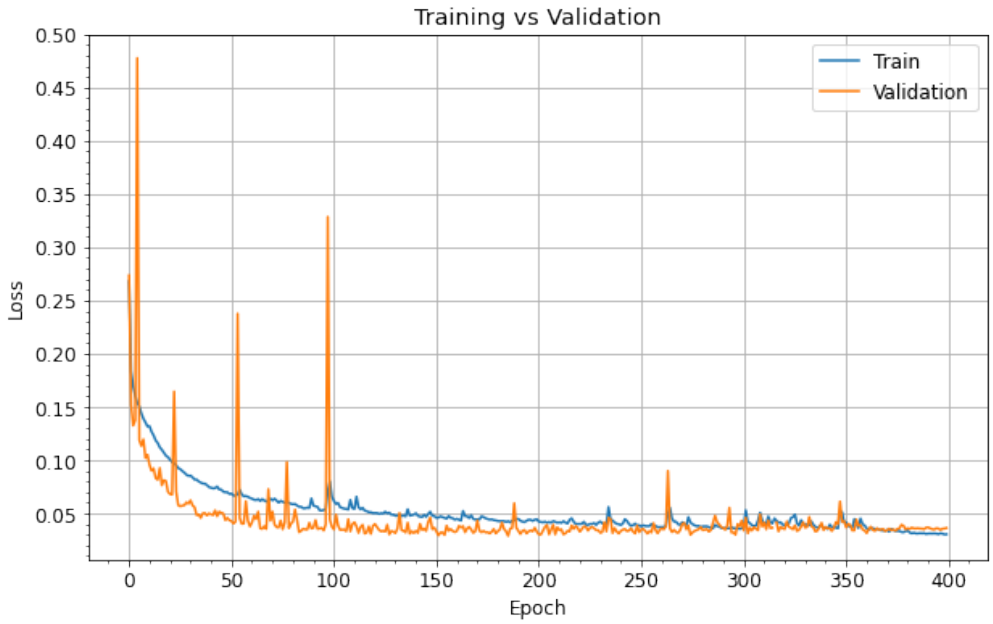


(b) Curva de entrenamiento *loss* en función de época.

Figura 5.6: Entrenamiento mejor modelo MCGA.



(a) Curva de entrenamiento *accuracy* en función de época.



(b) Curva de entrenamiento *loss* en función de época.

Figura 5.7: Entrenamiento mejor modelo RS.

Tabla 5.3: Resultados de mejor modelo MCGA vs RS.

Tipo de búsqueda	**Test Accuracy**	**Parámetros**
MCGA	94.99 %	2.8M
RS	93.99 %	2.0M

5.4. Aumentación de datos

Para mejorar el rendimiento del mejor modelo encontrado mediante MCGA, se utiliza aumentación de datos sobre CIFAR-10, utilizando la librería *Albumentations*. Además de variaciones típicas [17] como RandomCrop y CourseDropout, se incorporan otras variaciones, indicadas por la tabla 5.4.

Tabla 5.4: Operaciones de aumentación incorporadas.

Modificación	Descripción
Horizontal Flip	Espeja horizontalmente.
Rotación	Rota en los grados definidos.
Shift brillo/contraste	Realiza un *shift* en el brillo y contraste.
Shift HSV	Realiza un *shift* en el espacio de colores HSV.

Los valores de rotación*, shift brillo/contraste, shift HSV y CourseDropout fueron encontrados con un GridSearch sobre un modelo aleatorio del espacio de búsqueda, entrenado con 25 épocas (*debido a la gran cantidad de combinaciones, no se pudo computar toda las combinaciones que tenían valor de rotación 30). El espacio de búsqueda utilizado es la combinatoria de los parámetros, los conjuntos se indican en la tabla 5.5.

Tabla 5.5: Búsqueda de parámetros DA.

Modificación	Valores posibles
Rotación	{5, 15, 30*}
Shift brillo/contraste	{0.1, 0.3, 0.5, 0.9}
Shift HSV	{20, 50, 90}
Course Dropout Max Length	{4, 8, 16}
Course Dropout Fill Value	{0, 127, 255}
Course Dropout Max Holes	{2}

Para obtener los parámetros finales, primero, se seleccionaron los 10 mejores DA, mencionados en la tabla 5.6. Posteriormente, se llevó a cabo una búsqueda manual detallada sobre estos para identificar el DA óptimo para estos parámetros. Para otros parámetros, como Random Crop y Normalización, se emplearon los parámetros comúnmente usados, referenciados en [46, 17]. El DA seleccionado, es indicado en la tabla 5.7. Y la comparación de entrenamientos con aumentación y sin aumentación se indican en la tabla 5.8.

Tabla 5.6: Mejores diez variaciones de DA.

Rotacion	Bri/Cont	HSV	*dropHoles*	*dropMaxL*	*fillval*	*Accuracy*
5	0.5	20	2	4	127	0.8597
5	0.1	50	2	8	0	0.8594
30	0.1	20	2	8	127	0.8583
5	0.1	20	2	8	0	0.8560
5	0.1	50	2	4	0	0.8557
5	0.5	50	2	16	0	0.8545
15	0.3	20	2	16	127	0.8538
15	0.1	50	2	4	127	0.8536
15	0.5	20	2	4	127	0.8536
15	0.1	90	2	8	127	0.8534

Tabla 5.7: Parámetros seleccionados DA.

Modificación	Valores posibles
Rotación	[0, 30º]
Shift brillo/contraste	[0, 0.2]
Shift HSV	[0, 20]
Course Dropout Max Length	16
Course Dropout Fill Value	0
Course Dropout Max holes	2
Random Crop Resize	[0.9,1.2]
Normalización (mean, std)	(0.4914, 0.4822, 0.4465), (0.247, 0.243, 0.261)

Tabla 5.8: Resultados de mejor modelo MCGA entrenado sin DA y con DA.

Tipo de entrenamiento	**Test Accuracy**
Sin DA	93.03 %
Con DA	94.99 %

5.5. Multitarea

Para el experimento de incorporación de multitarea, se define la tarea primaria (T_1), como la búsqueda de las 10 clases diferentes dentro de CIFAR10, mientras que la tarea secundaria (T_2) será la clasificación de 2 clases más gruesas dentro de CIFAR10, animales y vehículos. Estas clases, por la distribución del conjunto de datos, tendrán 30000 y 20000 ejemplos de entrenamiento, y 6000 y 4000 ejemplos de prueba, respectivamente. La función de pérdida, por lo tanto, se define como:

$$L_{total} = w_1 L_1(y_1, \hat{y}_1) + w_2 L_2(y_2, \hat{y}_2).$$

Para evaluar el comportamiento del entrenamiento multitarea, se prueban las combina-

ciones de parámetros $(w_1, w_2) = (0.25, 0.75), (0.5, 0.5), (0.75, 0.25)$, y los resultados son los indicados en la tabla 5.9

Tabla 5.9: Resultados variación de parámetros (w_1, w_2).

(w_1, w_2)	**Test Accuracy** T_1	**Test Accuracy** T_2
(0.25, 0.75)	94.78 %	99.46 %
(0.5, 0.5)	95.05 %	99.46 %
(0.75, 0.25)	95.39 %	99.43 %

5.6. Atención

La tabla 5.10 indica los resultados de la incorporación de SimAM.

Tabla 5.10: Resultados incorporación de módulo de atención.

Modelo	**Test Accuracy**
Base	94.99 %
+SimAM	95.27 %

5.7. Parches neuronales

Para los parches neuronales se utilizaron las mismas 2 categorías que en *multitask*, por lo que la búsqueda se realizó para encontrar redes especializadas en los subconjuntos de animales y vehículos por separado. Para ambos AG se utilizaron los mismos parámetros que los mostrados en la sección resultados de MCGA. Figuras 5.8, 5.9, 5.10, 5.11 y 5.12 muestran las curvas de evolución, mientras que la tabla 5.11 indica el desempeño de los mejores modelos encontrados.

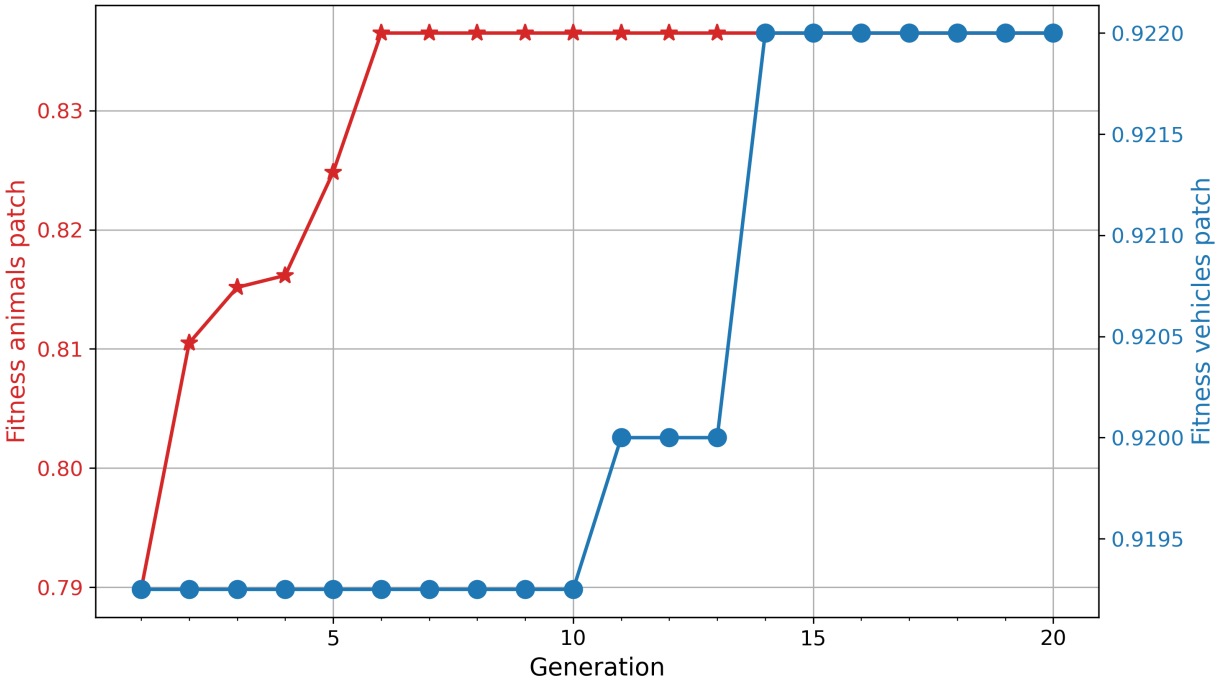


Figura 5.8: Parches neuronales: evolución individuo elite por generación. En rojo parche de animales, en azul parche de vehículos.

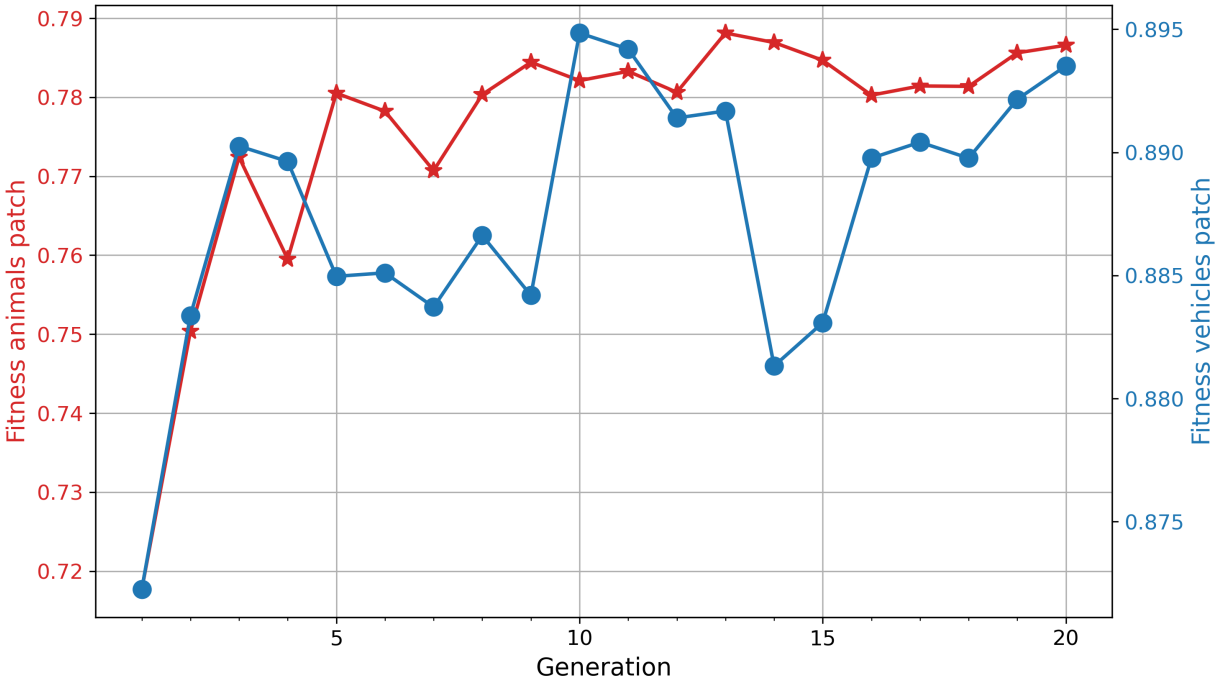
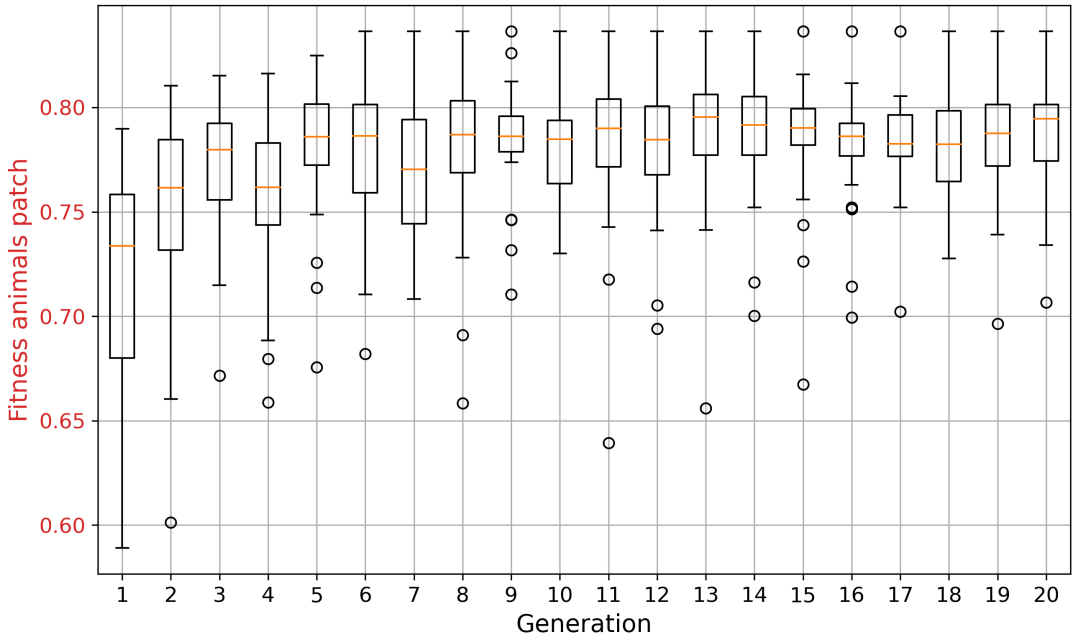
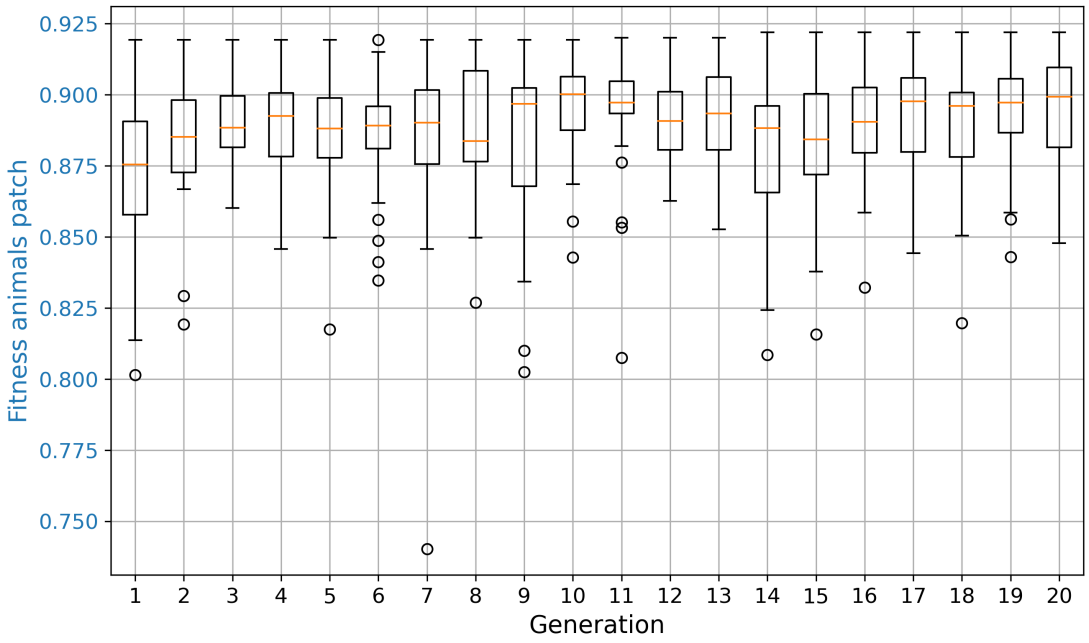


Figura 5.9: Parches neuronales: evolución *fitness* promedio por generación. En rojo parche de animales, en azul parche de vehículos.



(a) Animales.



(b) Vehículos.

Figura 5.10: Parches neuronales: evolución distribución de *fitness* (*accuracy*) por generación.

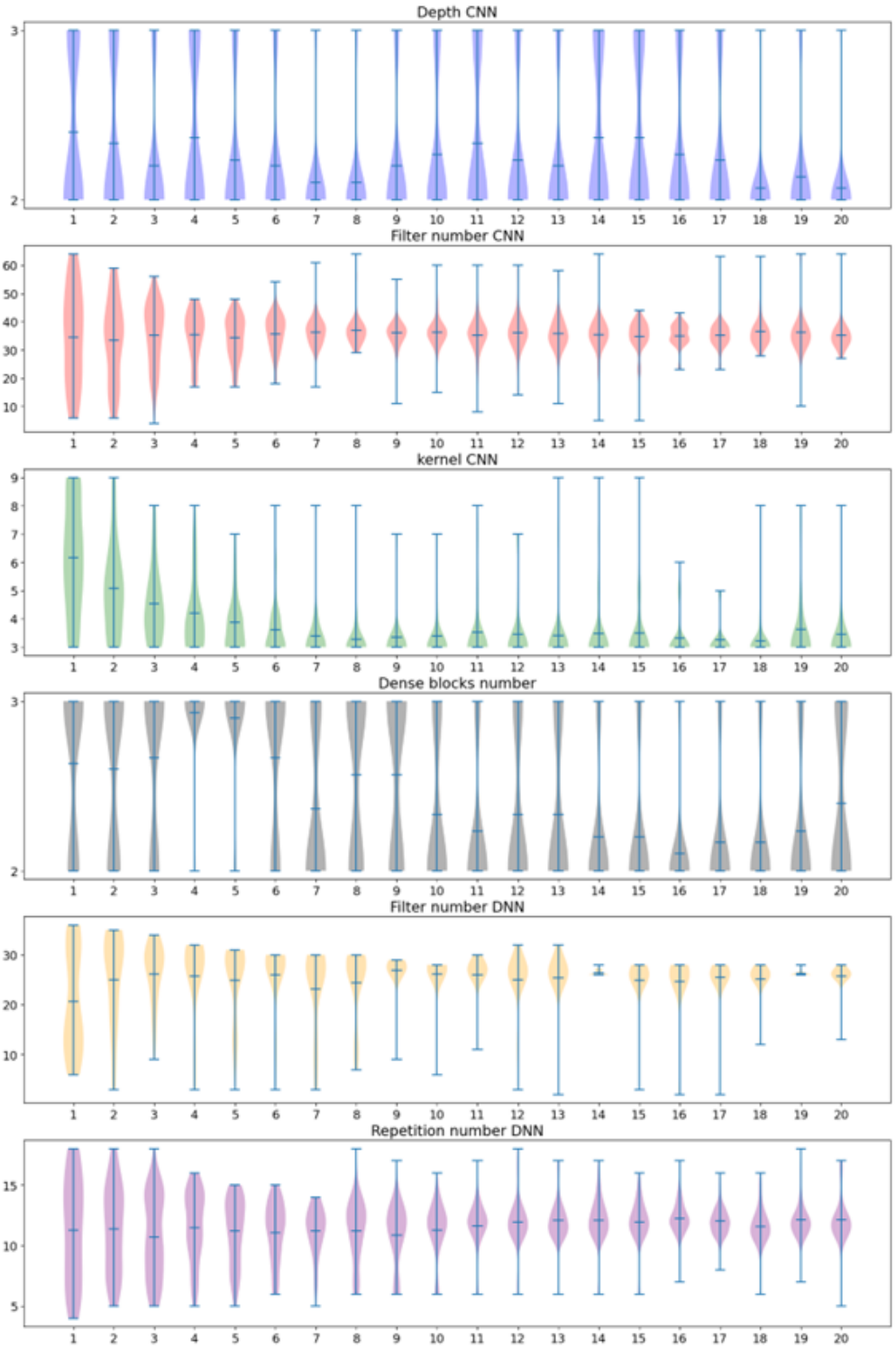


Figura 5.11: Parches neuronales: evolución de distribución de parámetros de búsqueda por generación de animales.

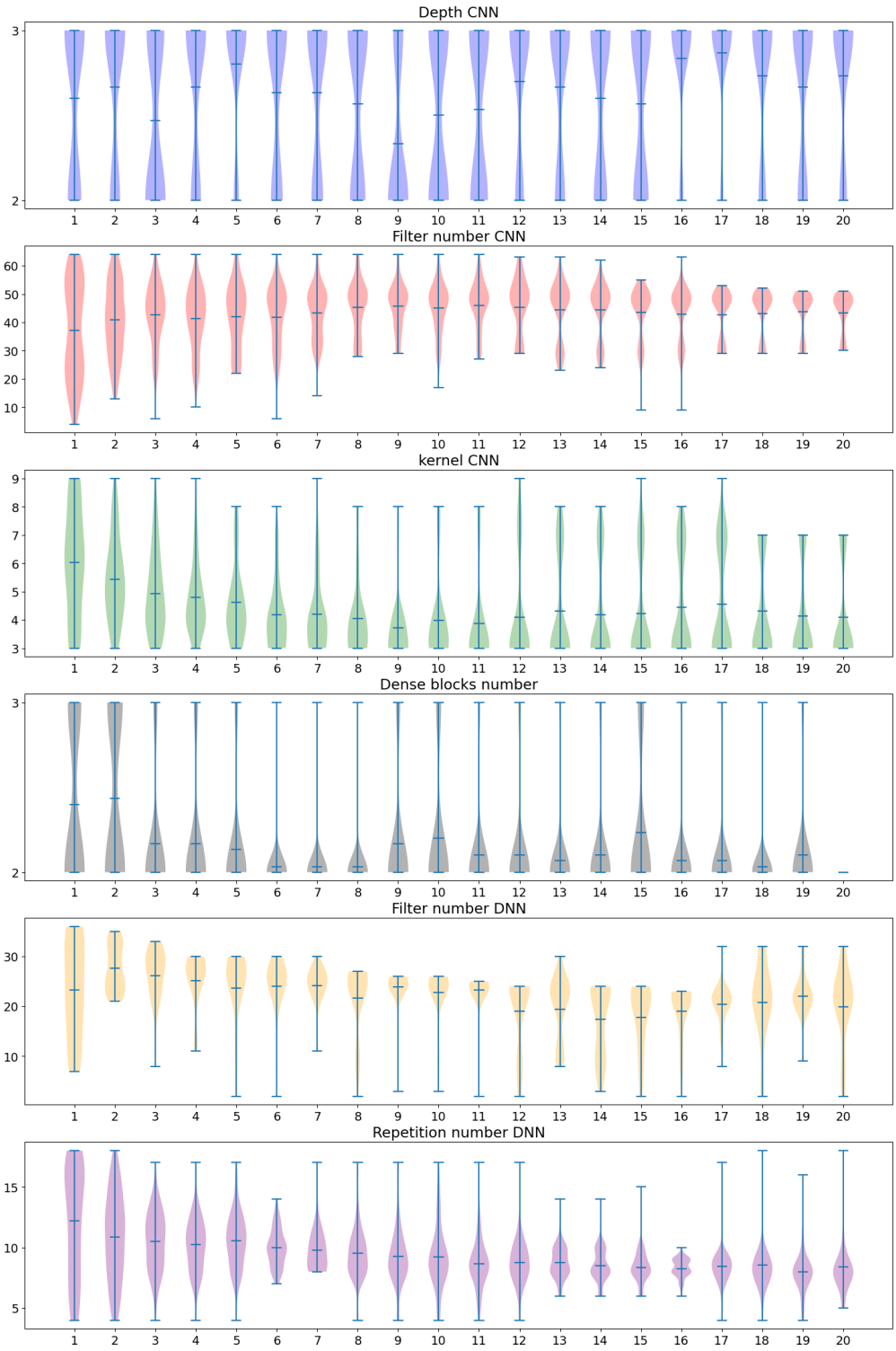


Figura 5.12: Parches neuronales: evolución de distribución de parámetros de búsqueda por generación de vehículos.

Tabla 5.11: Parches neuronales: desempeño modelo encontrado.

Parche	**Test Accuracy**	**Parámetros**
Animales	95.32 %	2.0M
Vehículos	97.43 %	2.2M

5.8. Ablación global

Dado los diferentes módulos y mejoras de los modelos, así como el ensamble de modelos con los parches neuronales, la tabla 5.12 muestra la ablación global.

Tabla 5.12: Resultados de estudio de ablación

Modelo	**Test Accuracy**
Solo CNN	76.38 %
Solo DNN	88.45 %
Baseline mixto	90.05 %
Mejor modelo MCGA	94.99 %
+SimAM	95.27 %
+MTL	95.39 %
+ MTL + Patches(+MTL)	95.60 %
+SimAM + MTL + Patches(+SimAM+MTL)	95.75 %

5.9. Comparación estado del arte

En esta sección se muestran los principales competidores dentro del área de NAS, comparando el desempeño (en porcentaje de error sobre la clasificación de clases en CIFAR-10), el número de parámetros en millones y el tiempo de búsqueda en días-GPU.

Tabla 5.13: Comparación de modelo con estado del arte.

Tipo	Nombre	Error (%)	Parámetros (M)	días-GPU
Manual	ResNet-110[14]	6.41	1.7	-
	FractalNet[128]	5.22	38.6M	-
	DenseNet-BC[15] (k = 12)	4.51	0.8M	-
	EfficientNet-B0 (pretrained)[16]	1.9	4M	-
RL	NASNet-A[57]	2.65	3.3	2000
RS	RandomNAS[82]	2.85	4.3	2.7
GO	DARTS[55]	2.76	3.3M	44
BO	BANANAS[84]	2.64	3.6	12
EA	Genetic Programming CNN [101]	5.98	1.7	14.9
	Large-scale Evolution[91]	5.4	5.4	2600
	CNN-GA[94]	4.78	-	-
	SOBA (PSO)[129]	4.78	-	1.42
	HGAPSO[66]	4.25	-	7.5
	AE-CNN[130]	4.3	2	27
	2LGA[46]	3.95	-	-
	Hierarchical-EAS[54]	3.75	15.7	300
	EPSOCNN[92]	3.58	6.74	4
	AmoebaNet-A[90]	3.34	3.2	3150
	MCGA	**4.61**	**2.8**	**2.5**
	MCGA + patches	**4.25**	**7.0**	**6**

5.10. Discusión

A partir de la estimación y cálculos de los diferentes espacios de búsqueda, se aprecia que las magnitudes de diferencia son extremadamente grandes. El espacio de búsqueda más pequeño que destaca por su baja cardinalidad es el de EPSOCNN con tan solo 1560 posibles arquitecturas, esto se debe a que los parámetros de búsqueda en este trabajo son solo dos y la cardinalidad de ambos son pequeños (26 y 20), a pesar de esto, si se quisiera usar fuerza bruta, con un promedio de 1 hora de entrenamiento por arquitectura, se requerirían 65 días-GPU para encontrar el óptimo.

El presente trabajo tiene un espacio de búsqueda con $4.89 \cdot 10^{15}$ posibles arquitecturas, si asumiéramos una hora de entrenamiento por arquitectura, esto resulta en 8.3 veces la edad del universo, lo que muestra la infactibilidad de probar todas las combinaciones posibles. No obstante, este espacio de búsqueda se encuentra limitado casi exclusivamente por la aplicación de reducciones de dimensionalidad preestablecidas, dado que el conjunto de datos soporta un máximo de 5 operaciones de submuestreo de tamaño 2x2.

Analizando en escala logarítmica, los tamaños de espacios de búsqueda más cercanos a este trabajo son HGAPSO, que se encuentra 7 décadas por debajo, DARTS, uno de los espacios más utilizados, con una estimación de 4 décadas superior, Johnson 5 décadas arriba y CNN-GA 6 décadas por encima. De estos espacios de búsqueda, DARTS es el único que utiliza una superred, y el más similar al de este trabajo es CNN-GA que, a pesar de no utilizar bloques densos, se pueden generar por la incorporación de *skip-connectcions*.

Large-scale Evolution y Hierarchical-EAS comparten magnitudes de cardinalidad similar del orden superior a 10^{43}, ya que ambos construyen modelos a partir de las operaciones básicas sin restricciones importantes para cantidad de operaciones utilizadas. SOBA (el único trabajo con datos de cardinalidad reportados) tiene una cantidad de arquitecturas posibles que es superior a la cantidad de átomos del universo (10^{80}), mientras que NASNet-A y AmoebaNet comparten es espacio de búsqueda con un número inconmensurable, 10^{13} décadas superior a cualquier otro. Considerando los tamaños de estos espacios de búsqueda, se convierte realmente irrelevante su tamaño, resultado más importante qué tipos de arquitecturas se pueden generar.

Pasando al contraste de métodos, es importante destacar que en términos de arquitecturas óptimas, se refiere a la mejora empírica de disminución del error de clasificación, por lo que la optimalidad es de manera local y no global dentro del espacio de búsqueda, de esta manera, se comparan todas las metodologías como heurísticas.

Pasando a la comparación de métodos de búsqueda dentro de la codificación de MCGA, observando las diferencias entre las curvas de evolución de ambos métodos, podemos ver que de manera esperable la curva de distribución de *fitness* con RS es más o menos uniforme a través de las generaciones, en la curva de evolución de MCGA se observa una compresión de la distribución, con algunos *outliers*, lo que significa que va explotando las mejores soluciones a través de las generaciones. A partir de las curvas promedio, se puede verificar un compartimiento similar, en MGCA se observa una tendencia positiva, mientras que en RS es más o menos uniforme. A partir de la densidad de parámetros, en MCGA se observa la tendencia

de compresión de parámetros, particularmente de los números de filtros DNN y CNN, por lo que parecen ser los parámetros más relevantes para el desempeño, caso opuesto en RS se observa una distribución relativamente uniforme.

De las curvas de mejor individuo, podemos observar por qué RS sigue siendo un método competitivo, ya que de igual forma se van encontrando mejores individuos a través de las generaciones, sin embargo, MCGA logra encontrar mejores individuos con respecto a RS.

En la ablación de los modelos, el modelo puramente convolucional equivalente tiene un desempeño muy bajo, donde además comienza con un sobreajuste sobre a partir de unas tempranas 40 épocas, esto se debe principalmente al que el modelo no es muy profundo, lo que implica que no puede aprender buenas representaciones de los datos. En el modelo estrictamente denso se observa un desempeño comparable al modelo mixto, con un error de 1.6 % más sobre los datos de prueba. De esta manera, el modelo mixto resulta ser la mejor opción.

A partir de la búsqueda de parámetros de la aumentación de datos, para *course dropout* se obtiene un valor de tamaño máximo similar al comúnmente utilizado en la literatura [17], para el valor de relleno se ve una clara preferencia a utilizar el valor medio (127) o nulo (0), se decide usar el valor nulo por ser estándar. Finalmente, el ajuste final se realiza utilizando los valores, pero modificando la rotación para tener un rango más amplio, ya que se obtuvo mejor desempeño empírico durante los demás experimentos. La aumentación de datos resulta en una disminución sustancial del error pasando de 6.97 % a 5.01 %, lo que significa una mejora del 22.1 %.

Sobre el aprendizaje multitarea, en todos los resultados se observa una mejora al desempeño sobre la tarea primaria, esto se debe a que la incorporación de la tarea secundaría obliga al modelo a crear abstracciones, que en este caso, le son también útiles a la tarea primaria. También, a partir del análisis de sensibilidad del parámetro γ, a pesar que las funciones de perdida se ven con escalas de aproximadamente un orden de magnitud de diferencia, el promedio de ambas es el que entrega resultados, esto muestra que contraintuitivamente para este tipo de modelos, una bajo aporte dentro de la función global de perdida por parte de la tarea secundaria (que es más simple y tiene mejor desempeño), entrega un mejor desempeño para la tarea primaria, que es más compleja.

La incorporación de módulos de atención mejora ligeramente el desempeño del modelo, decreciendo el error en un 5.5 % con respecto al modelo sin el módulo. En escenarios como éste, donde se tiene tasas de error relativamente bajas, estas mejoras modulares, que además no agregan más parámetros de entrenamiento, se puede observar que son especialmente útiles al reducir el error aún más.

Respecto los parches neuronales, de las búsquedas de ambos modelos secundarios, se observan comportamientos similares a la búsqueda del modelo principal, donde se tienen convergencia más marcada en los parámetros del número de filtros para DNN y CNN. Entre ambas evoluciones, se observa que el modelo de animales converge a ser más compacto con solamente 2.0M de parámetros, a diferencia del modelo de vehículos que tiene 2.2M.

De esta manera, la cantidad de parámetros entrenables es 27 % inferior para el modelo de clasificación de animales y 19 % para el modelo de clasificación de vehículos, con respecto al número de parámetros del modelo principal. De la cantidad de capas operacionales (convolucionales, *Batch Normalización*, agregación, etc), los modelos tienen 54 % y 20 % capas menos, por lo que ambos modelos necesarios en los parches son más pequeños y menos profundos. Esto confirma que el sistema completo se comporta biomiméticamente al utilizar menores recursos para las clasificaciones más específicas.

A partir de los resultados de la ablación global, se observa el comportamiento esperado, análogo al cerebro, que el desempeño mejora con las incorporaciones de más mecanismos para ayudar a la tarea de clasificación de imágenes. La progresión de desempeño que se observa indica que la combinación de las diferentes técnicas puede llevar a mejoras sustanciales en el rendimiento de los modelos. La eficacia de los módulos de atención enfatiza la importancia de mecanismos que permitan focalizar y diferenciar de mejor forma las características relevantes dentro de los datos. La integración del aprendizaje multitarea sugiere que aprender tareas secundarias sirve para regularizar la red, permitiendo una capacidad superior y que el modelo aprenda varios aspectos relevantes que sin la tarea secundaria no existen. De las diferentes incorporaciones, se observa que el aprendizaje multitarea disminuye en 7.8 % el error con respecto al modelo encontrado mediante MCGA, mientras que SimAM disminuye en un 5.5 % el error, lo que muestra que mejorar las representaciones de los *feature maps* es más importante que atender de mejor forma la información, sin embargo, ambas incorporaciones resultan en mejorar el modelo base. Al ensamblar el modelo principal, con los parches que clasifican animales y vehículos de manera independiente (incoporando MTL), se obtiene una disminución de 4.5 % con respecto al modelo que usa solo multitarea y 12.2 % con respecto al modelo base, demostrando la utilidad de utilizar modelos especializados. Para la agregación de todas las mejoras, se tiene una disminución de un 15.2 % de error con respecto al modelo base, mostrando que cada módulo aporta a mejorar el desempeño global. De todas las mejoras, la incorporación de MTL resulta ser la que más aumenta el desempeño de manera relativa, al remover 0.40 % de error neto sobre el conjunto de datos, mientras que los parches(+MTL), reducen 0.21 % neto (comparado con el modelo base+MTL) y el modelo con todas las mejoras reduce en 0.15 % neto (con respecto a parches+MTL).

Para hacer una comparación con otros modelos del estado del arte, los criterios de evaluación se centran en las tasas de error de clasificación en CIFAR10, la complejidad del modelo en términos del número de parámetros (millones), el costo computacional medido en días-GPU y finalmente la comparabilidad de espacios de búsqueda.

Los métodos manuales, como referencia del diseño de CNNs, variadas tasas de error y números de parámetros se observan. EfficientNet-B0, siendo el diseño más nuevo, es producto de un diseño manual meticuloso, con un error de sólo 1.9 % con 4M de parámetros, sin embargo, esta arquitectura, tiene un entrenamiento especializado, que le permite tener tasas de error muy bajas. DenseNet-BC (k = 12) es un tipo de modelo más comparable a la propuesta de este trabajo, consta con una tasa de error del 4.51 % con solo 0.8M de parámetros, sugiriendo eficiencia en la utilización de parámetros con un buen desempeño. Mientras tanto, FractalNet y ResNet representan enfoques que requieren más parámetros con 38.6M y 1.7M, respectivamente y con errores más altos.

El modelo basado en RL NASNet-A logra una baja tasa de error del 2.65 % con un número moderado de parámetros (3.3M), aunque con un costo computacional extremadamente alto de 2000 días-GPU. Por otra parte, RandomNAS, presenta una tasa de error ligeramente superior del 2.85 % pero requiere costo computacional mínimo de 2.7 días-GPU, sin embargo, este método utiliza un espacio de búsqueda estándar, donde además ya se conocen estrategias de codificaciones óptimas.

Los métodos basados en gradientes y optimización bayesiana (BO), también tiene nivel de error competitivos con 3.78 % y 2.64 %. BANANAS destaca por su uso eficiente de parámetros y un gasto computacional no muy alto, al igual que otros métodos también utiliza un espacio de búsqueda estándar.

Dentro de la categoría de Algoritmos Evolutivos (EA), donde se sitúa el MCGA propuesto, muestra una variedad con diferentes métricas. Notablemente, AmoebaNet-A y Hierarchical-EAS logran tasas de error muy bajas del 3.34 % y 3.75 %, respectivamente, sin embargo, estos vienen con un costo computacional sustancial, mostrado en sus días-GPU.

El método propuesto MCGA demuestra una equilibrio de precisión y eficiencia, alcanzando una tasa de error de 4.25 %, con 7M de parámetros para la estrategia de parches neuronales, encontrados en 6 días-GPU. Esto posiciona al método como una forma viable con un *trade-off* razonable, además, cumpliendo con ser una novedosa propuesta basada en biología. En comparación a los métodos con espacios de búsqueda similares, MCGA supera en rendimiento y tiempo a CNN-GA. Se tiene un desempeño idéntico a HGPSO, pero con un tamaño de espacio mayor y más parámetros de búsqueda. En cuanto a EPSOCNN se tiene un desempeño sustancialmente menor, con 0.67 % de mayor tasa de error, pero es muy importante destacar que este método solo tiene un espacio de búsqueda un tamaño muy reducido (solo 1560 arquitecturas posibles) que busca directamente en arquitecturas tipo densenet, las cuales son conocidas por tener buen desempeño.

Capítulo 6

Conclusiones

En el presente trabajó se logró diseñar modelos de redes neuronales convolucionales inspirados en arquitecturas biológicas retinotópicas utilizando algoritmos genéticos, por lo que el objetivo general del estudio se logró, los diferentes experimentos realizados con MCGA respaldan la hipótesis que se puede utilizar neuroevolución y sistemas modulares para crear arquitecturas biológicamente inspiradas.

La implementación del algoritmo genético multricromosómico demuestra ser una estrategia efectiva para representar arquitecturas con estructuras diversas, en este caso, compuestas de dos elementos, validando una de las hipótesis al inicio del trabajo, y que va en concordancia que buenas representaciones del espacio de búsqueda, permiten encontrar buenas soluciones. La incorporación de los distintas estrategias demuestra que el diseño de modelos de manera holística, incorporando pequeñas modificaciones, resulta crear mejoras significativas en el desempeño de CNN en el contexto de reconocimiento de imágenes.

La aumentación de datos permitió mejorar sustancialmente todos los entrenamientos, esto se muestra ya que en el modelo base se disminuye en un 22.1 % el error, siendo el elemento más importante para mejorar el desempeño. Los diferentes elementos bio-inspirados utilizados resultan efectivamente en mejoras incrementales al modelo base encontrado con MCGA, cumpliendo con el objetivo de incrementar el desempeño con elementos modulares. La atención ayuda a disminuir en un 5.5 % el error, el aprendizaje multitarea en un 7.8 % y los parches neuronales en un 12.2 %. La incorporación de todos las mejoras resulta en la disminución de un 15.2 % de error, comprobando que la agregación de todos los elementos resulta en una mejora mayor que de manera indivudal.

Las pruebas efectuadas al comparar el MCGA con respecto a RS, en primer lugar, muestran que la codificación del espacio de búsqueda es buena, ya que incluso RS puede navegar el espacio encontrando mejores modelos a través de las generaciones. Además, se observa que el MCGA es un algoritmo superior, ya que encuentra mejores modelos para el mismo número de generaciones, y también navega el espacio con mayor nivel de explotación.

El mejor modelo obtenido con una sola búsqueda mediante MCGA obtuvo un error de 4.61 % en CIFAR-10, con 2.5M de parámetros y en sólo 2.8 días de búsqueda. Al incorporar los parches, el aprendizaje multitarea y el módulo de atención se disminuye el error a ser 4.25 % a costa de aumentar la cantidad de parámetros a 7M y la búsqueda a 6 días. Dentro

de todos los métodos evolutivos analizados, al observar el error sobre CIFAR-10, MCGA se encuentra en quinto lugar (de los diez), mientras que en la comparación con métodos que usan espacios de búsqueda similares, MCGA se encuentra en segundo lugar (de los cuatro), solo superado por EPSOCNN, que utiliza un espacio de búsqueda muy pequeño.

En concordancia con los objetivos, se generaron de manera automática mediante el MCGA redes neuronales convoluciones que demuestran ser competitivas en desempeño, tamaño y días de búsqueda con respecto al estado del arte que utiliza espacio de búsquedas similares. Los tipos de arquitecturas factibles se diseñaron con analogías biológicas retinotópicas para el reconocimiento de patrones, lo que fue validado de manera exitosa en el dataset CIFAR10.

6.1. Trabajo futuro

Respecto a las posibles mejores, existen muchas líneas posibles de mejoras, pero acotándose a lo propuesto por la presente tesis, se podrían meta-optimizar los hiperparámetros del algoritmo genético multicromosómico, esto se puede realizar con experimentos estadísticos con diferentes probabilidades de mutación, tamaño de la población u otros, también se podrían encontrar o generar mejores operadores de *cross over* o selección. Otra forma de aprovechar el MCGA sería utilizar más cromosomas para representar el individuo, por ejemplo, un módulo de atención con parámetros, como CBAM, podría ser representado en un tercer cromosoma.

El principal problema de este metaproblema es lo costoso de calcular o estimar el *fitness* de cada modelo, para ello se podrían utilizar incorporar técnicas actuales como *transfer learning* mediante morfismos de redes o *knowledge destillation* para transferir conocimiento de modelos entrenados de generaciones previas o de modelos preentrenados diseñados manualmente. Otra forma de mejorar el *trade-off* de tiempo y desempeño sería combinar la codificación de MCGA con el algoritmo genético de dos niveles para poder obtener mejores arquitecturas con un tiempo de búsqueda ligeramente superior. Otras incorporaciones potenciales de otros elementos bio-inspirados, como capas de campos receptivos grandes, podrían ser de utilidad para mejorar aún más el desempeño.

Un elemento importante para la justa comparación de las metodologías es el tiempo de ejecución, debido a que el estado del arte utiliza una medida hardware-dependiente como los días-GPU, se podría definir una métrica más justa, por ejemplo, la normalización de valores de tiempo en base a la capacidad computacional del hardware.

Otro aspecto para validar aún más el MCGA sería utilizarlo en más bases de datos o bases de datos más grandes como ImageNet, y en esta misma idea realizar mayor investigación en el área de los parches neuronales simulados, para poder obtener resultados con más parches o en conjuntos de datos que ya contienen jerarquía como CIFAR100 con sus etiquetas finas y gruesas. Además, MCGA podría ser utilizado con más parámetros de búsqueda o incorporando ViTs a la arquitectura final o de búsqueda.

Bibliografía

[1] "Automaton | Definition, History, & Facts | Britannica." https://www.britannica.com/technology/automaton. [Accessed 11-10-2023].

[2] C. Huffman, *Archytas of Tarentum: Pythagorean, Philosopher and Mathematician King*. Cambridge University Press, 2005.

[3] C. Huffman, "Archytas," in *The Stanford Encyclopedia of Philosophy* (E. N. Zalta, ed.), Metaphysics Research Lab, Stanford University, Winter 2020 ed., 2020.

[4] P. Hill, *The Book of Knowledge of Ingenious Mechanical Devices*. Springer Science & Business Media, 2012.

[5] M. Georgievska, "The Elephant Clock: One of the greatest inventions of the outstanding mechanical engineer Al-Jazari | The Vintage News — thevintagenews.com." https://www.thevintagenews.com/2017/05/06/the-elephant-clock-one-of-the-greatest-inventions-of-the-outstanding-mechanical-engineer-al-jazari/. [Accessed 02-01-2024].

[6] M. Dirik, "Al-jazari: The ingenious inventor of cybernetics and robotics," *Journal of Soft Computing and Artificial Intelligence*, vol. 1, no. 1, p. 47–58, 2020.

[7] I. Asimov, *Sobre la ciencia ficción*. Edhasa, 1986.

[8] A. C. Clarke, *2001: a space odyssey*. Penguin, 2016.

[9] A. Tamkin, M. Brundage, J. Clark, and D. Ganguli, "Understanding the capabilities, limitations, and societal impact of large language models," *arXiv preprint arXiv:2102.02503*, 2021.

[10] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, p. 193–202, Apr. 1980.

[11] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, IEEE, June 2016.

[15] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected

convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2017.

[16] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, pp. 6105–6114, PMLR, 2019.

[17] C. White, M. Safari, R. Sukthanker, B. Ru, T. Elsken, A. Zela, D. Dey, and F. Hutter, "Neural architecture search: Insights from 1000 papers," *arXiv preprint arXiv:2301.08727*, 2023.

[18] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 1997–2017, 2019.

[19] Z. Lu, G. Sreekumar, E. Goodman, W. Banzhaf, K. Deb, and V. N. Boddeti, "Neural architecture transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 9, pp. 2971–2989, 2021.

[20] S. R. Najle, X. Grau-Bové, A. Elek, C. Navarrete, D. Cianferoni, C. Chiva, D. Cañas-Armenteros, A. Mallabiabarrena, K. Kamm, E. Sabidó, H. Gruber-Vodicka, B. Schierwater, L. Serrano, and A. Sebé-Pedrós, "Stepwise emergence of the neuronal gene expression program in early animal evolution," *Cell*, vol. 186, pp. 4676–4693.e29, Oct. 2023.

[21] "File:Neuron3.svg - Wikimedia Commons — commons.wikimedia.org." https://commons.wikimedia.org/wiki/File:Neuron3.svg. [Accessed 08-12-23].

[22] B. Wang, T. Yan, S. Ohno, S. Kanazawa, and J. Wu, "Retinotopy and attention to the face and house images in the human visual cortex," *Experimental Brain Research*, vol. 234, pp. 1623–1635, Feb. 2016.

[23] A. A. Brewer and B. Barton, "Visual field map organization in human visual cortex," in *Visual Cortex* (S. Molotchnikoff and J. Rouat, eds.), ch. 2, Rijeka: IntechOpen, 2012.

[24] R. B. Tootell, N. Hadjikhani, E. Hall, S. Marrett, W. Vanduffel, J. Vaughan, and A. M. Dale, "The retinotopy of visual spatial attention," *Neuron*, vol. 21, pp. 1409–1422, Dec. 1998.

[25] R. S. Fishman, "Gordon holmes, the cortical retina, and the wounds of war: The seventh charles b. snyder lecture," *Documenta Ophthalmologica*, vol. 93, pp. 9–28, Mar. 1997.

[26] M. A. Goodale and A. Milner, "Separate visual pathways for perception and action," *Trends in Neurosciences*, vol. 15, no. 1, pp. 20–25, 1992.

[27] S.-H. Choi, G. Jeong, Y.-B. Kim, and Z.-H. Cho, "Proposal for human visual pathway in the extrastriate cortex by fiber tracking method using diffusion-weighted mri," *NeuroImage*, vol. 220, p. 117145, 2020.

[28] M. N. Hebart and G. Hesselmann, "What visual information is processed in the human dorsal stream?," *Journal of Neuroscience*, vol. 32, no. 24, pp. 8107–8109, 2012.

[29] "Diagram showing parallel pathways of visual processing." https://www.researchgate.net/figure/Diagram-showing-parallel-pathways-of-Visual-processing-1-Ventral-stream-processes-what_fig3_350709970. [Accessed 09-12-2023].

[30] Y. Sasaki, N. Hadjikhani, B. Fischl, A. K. Liu, S. Marret, A. M. Dale, and R. B. H. Tootell, "Local and global attention are mapped retinotopically in human occipital

cortex," *Proceedings of the National Academy of Sciences*, vol. 98, pp. 2077–2082, Feb. 2001.

[31] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, p. 115–133, Dec. 1943.

[32] "File:Perceptrón 5 unidades.svg - Wikimedia Commons — commons.wikimedia.org." https://commons.wikimedia.org/wiki/File:Perceptr%C3%B3n_5_unidades.svg. [Accessed 02-01-2024].

[33] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[34] "Colored neural network — commons.wikimedia.org." https://commons.wikimedia.org/wiki/File:Colored_neural_network_es.svg. [Accessed 04-12-2023].

[35] B. Duchaine, L. Cosmides, and J. Tooby, "Evolutionary psychology and the brain," *Current Opinion in Neurobiology*, vol. 11, p. 225–230, Apr. 2001.

[36] P. Bao, L. She, M. McGill, and D. Y. Tsao, "A map of object space in primate inferotemporal cortex," *Nature*, vol. 583, pp. 103–108, Jul 2020.

[37] "What are Convolutional Neural Networks? — futurelearn.com." https://www.futurelearn.com/info/courses/intelligent-systems/0/steps/245920. [Accessed 05-12-2023].

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

[39] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, JMLR Workshop and Conference Proceedings, 2010.

[40] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, "Snapshot ensembles: Train 1, get m for free," *arXiv preprint arXiv:1704.00109*, 2017.

[41] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," in *Artificial intelligence and machine learning for multi-domain operations applications*, vol. 11006, pp. 369–386, SPIE, 2019.

[42] R. Ng, "Deep Learning Wizard - Deep Learning Wizard — deeplearningwizard.com." https://www.deeplearningwizard.com/. [Accessed 05-12-2023].

[43] R. Alberto, "¿Qué es Underfitting y Overfitting? — rubialesalberto.medium.com." https://rubialesalberto.medium.com/qu%C3%A9-es-underfitting-y-overfitting-c73d51ffd3f9. [Accessed 05-01-2024].

[44] A. Baldominos, Y. Saez, and P. Isasi, "On the automated, evolutionary design of neural networks: past, present, and future," *Neural computing and applications*, vol. 32, pp. 519–545, 2020.

[45] E. Galván and P. Mooney, "Neuroevolution in deep neural networks: Current trends and future challenges," *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 6, pp. 476–493, 2021.

[46] D. A. Montecino, C. A. Perez, and K. W. Bowyer, "Two-level genetic algorithm for evolving convolutional neural networks for pattern recognition," *IEEE Access*, vol. 9,

pp. 126856–126872, 2021.

[47] M.-H. Guo, T.-X. Xu, J.-J. Liu, Z.-N. Liu, P.-T. Jiang, T.-J. Mu, S.-H. Zhang, R. R. Martin, M.-M. Cheng, and S.-M. Hu, "Attention mechanisms in computer vision: A survey," *Computational Visual Media*, pp. 1–38, 2022.

[48] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE transactions on neural networks and learning systems*, 2021.

[49] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pp. 818–833, Springer, 2014.

[50] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[51] D. Han, J. Kim, and J. Kim, "Deep pyramidal residual networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5927–5935, 2017.

[52] X. He, K. Zhao, and X. Chu, "Automl: A survey of the state-of-the-art," *Knowledge-Based Systems*, vol. 212, p. 106622, 2021.

[53] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: a big comparison for nas," *arXiv preprint arXiv:1912.06059*, 2019.

[54] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," *arXiv preprint arXiv:1711.00436*, 2017.

[55] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[56] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[57] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8697–8710, 2018.

[58] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," *arXiv preprint arXiv:1611.02167*, 2016.

[59] H. Kitano, "Designing neural networks using genetic algorithms with graph generation system," *Complex System*, vol. 4, no. 4, pp. 461–476, 1990.

[60] G. F. Miller, P. M. Todd, and S. U. Hegde, "Designing neural networks using genetic algorithms.," in *ICGA*, vol. 89, pp. 379–384, 1989.

[61] F. Johnson, A. Valderrama, C. Valle, B. Crawford, R. Soto, and R. Ñanculef, "Automating configuration of convolutional neural network hyperparameters using genetic algorithm," *IEEE Access*, vol. 8, pp. 156139–156152, 2020.

[62] N. Awad, N. Mallik, and F. Hutter, "Differential evolution for neural architecture search," *arXiv preprint arXiv:2012.06400*, 2020.

[63] A. Ghosh, N. D. Jana, S. Mallik, and Z. Zhao, "Designing optimal convolutional neural network architecture using differential evolution algorithm," *Patterns*, vol. 3, no. 9,

2022.

[64] B. Wang, Y. Sun, B. Xue, and M. Zhang, "Evolving deep convolutional neural networks by variable-length particle swarm optimization for image classification," in *2018 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, IEEE, 2018.

[65] F. E. F. Junior and G. G. Yen, "Particle swarm optimization of deep neural networks architectures for image classification," *Swarm and Evolutionary Computation*, vol. 49, pp. 62–74, 2019.

[66] B. Wang, Y. Sun, B. Xue, and M. Zhang, "A hybrid ga-pso method for evolving architecture and short connections of deep convolutional neural networks," in *PRICAI 2019: Trends in Artificial Intelligence* (A. C. Nayak and A. Sharma, eds.), (Cham), pp. 650–663, Springer International Publishing, 2019.

[67] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia Tools and Applications*, vol. 80, p. 8091–8126, Oct. 2020.

[68] J. H. Holland, "Genetic algorithms," *Scientific American*, vol. 267, no. 1, pp. 66–73, 1992.

[69] A. Hussain and Y. S. Muhammad, "Trade-off between exploration and exploitation with genetic algorithm using a novel selection operator," *Complex & Intelligent Systems*, vol. 6, pp. 1–14, Apr. 2019.

[70] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.

[71] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[72] C. Foster, M. Zhao, T. Bolkart, M. J. Black, A. Bartels, and I. Bülthoff, "The neural coding of face and body orientation in occipitotemporal cortex," *NeuroImage*, vol. 246, p. 118783, 2022.

[73] M. J. Arcaro, T. Mautz, V. K. Berezovskii, and M. S. Livingstone, "Anatomical correlates of face patches in macaque inferotemporal cortex," *Proceedings of the National Academy of Sciences*, vol. 117, no. 51, pp. 32667–32678, 2020.

[74] "GitHub - jeonsworld/ViT-pytorch: Pytorch reimplementation of the Vision Transformer (An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale) — github.com." https://github.com/jeonsworld/ViT-pytorch/. [Accessed 09-12-2023].

[75] K.-H. Thung and C.-Y. Wee, "A brief review on multi-task learning," *Multimedia Tools and Applications*, vol. 77, pp. 29705–29725, Aug. 2018.

[76] L. Bonetti, E. Brattico, S. Bruzzone, G. Donati, G. Deco, D. Pantazis, P. Vuust, and M. Kringelbach, "Temporal pattern recognition in the human brain: a dual simultaneous processing," *bioRxiv*, pp. 2021–10, 2021.

[77] M. S. Livingstone, J. L. Vincent, M. J. Arcaro, K. Srihasam, P. F. Schade, and T. Savage, "Development of the macaque face-patch system," *Nature communications*, vol. 8, no. 1, p. 14897, 2017.

[78] R. Rajalingham and J. J. DiCarlo, "Reversible inactivation of different millimeter-scale regions of primate IT results in different patterns of core object recognition deficits," *Neuron*, vol. 102, pp. 493–505.e5, Apr. 2019.

[79] S. Vossel, J. J. Geng, and G. R. Fink, "Dorsal and ventral attention systems: Distinct neural circuits but collaborative roles," *The Neuroscientist*, vol. 20, pp. 150–159, July 2013.

[80] A. Yang, P. M. Esperança, and F. M. Carlucci, "Nas evaluation is frustratingly hard," *arXiv preprint arXiv:1912.12522*, 2019.

[81] L.-C. Chen, M. Collins, Y. Zhu, G. Papandreou, B. Zoph, F. Schroff, H. Adam, and J. Shlens, "Searching for efficient multi-scale architectures for dense image prediction," *Advances in neural information processing systems*, vol. 31, 2018.

[82] L. Li and A. Talwalkar, "Random search and reproducibility for neural architecture search," in *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference* (R. P. Adams and V. Gogate, eds.), vol. 115 of *Proceedings of Machine Learning Research*, pp. 367–377, PMLR, 22–25 Jul 2020.

[83] K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, and E. P. Xing, "Neural architecture search with bayesian optimisation and optimal transport," *Advances in neural information processing systems*, vol. 31, 2018.

[84] C. White, W. Neiswanger, and Y. Savani, "Bananas: Bayesian optimization with neural architectures for neural architecture search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 10293–10301, 2021.

[85] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, and H. Xiong, "Pc-darts: Partial channel connections for memory-efficient architecture search," *arXiv preprint arXiv:1907.05737*, 2019.

[86] A. Hundt, V. Jain, and G. D. Hager, "sharpdarts: Faster and more accurate differentiable architecture search," *arXiv preprint arXiv:1903.09900*, 2019.

[87] H. Wang, R. Yang, D. Huang, and Y. Wang, "idarts: Improving darts by node normalization and decorrelation discretization," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[88] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710, 2018.

[89] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *International conference on machine learning*, pp. 4095–4104, PMLR, 2018.

[90] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the aaai conference on artificial intelligence*, vol. 33, pp. 4780–4789, 2019.

[91] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 2902–2911, PMLR, 06–11 Aug 2017.

[92] B. Wang, B. Xue, and M. Zhang, "Particle swarm optimisation for evolving deep neural networks for image classification by evolving and stacking transferable blocks," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, 2020.

[93] S. Fujino, N. Mori, and K. Matsumoto, "Deep convolutional networks for human sketches by means of the evolutionary deep learning," in *2017 Joint 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS)*, IEEE, June 2017.

[94] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically designing cnn architectures using the genetic algorithm for image classification," *IEEE transactions on cybernetics*, vol. 50, no. 9, pp. 3840–3854, 2020.

[95] L. Xie and A. Yuille, "Genetic cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1379–1388, 2017.

[96] S. Gibb, H. M. La, and S. Louis, "A genetic algorithm for convolutional network structure optimization for concrete crack detection," in *2018 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, July 2018.

[97] H. Zhu, Z. An, C. Yang, K. Xu, E. Zhao, and Y. Xu, "Eena: Efficient evolution of neural architecture," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pp. 1891–1899, IEEE Computer Society, 2019.

[98] T. Elsken, J.-H. Metzen, and F. Hutter, "Simple and efficient architecture search for convolutional neural networks," *arXiv preprint arXiv:1711.04528*, 2017.

[99] M. Suganuma, M. Ozay, and T. Okatani, "Exploiting the potential of standard convolutional autoencoders for image restoration by evolutionary search," in *International Conference on Machine Learning*, pp. 4771–4780, PMLR, 2018.

[100] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, "A survey on evolutionary neural architecture search," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, pp. 550–570, Feb. 2023.

[101] B. Evans, H. Al-Sahaf, B. Xue, and M. Zhang, "Evolutionary deep learning: A genetic programming approach to image classification," in *2018 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, July 2018.

[102] M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in *Proceedings of the genetic and evolutionary computation conference*, pp. 497–504, 2017.

[103] Z. Fan, J. Wei, G. Zhu, J. Mo, and W. Li, "Evolutionary neural architecture search for retinal vessel segmentation," *arXiv preprint arXiv:2001.06678*, 2020.

[104] E. Byla and W. Pang, "Deepswarm: Optimising convolutional neural networks using swarm intelligence," in *Advances in Computational Intelligence Systems: Contributions Presented at the 19th UK Workshop on Computational Intelligence, September 4-6, 2019, Portsmouth, UK 19*, pp. 119–130, Springer, 2020.

[105] P. R. Lorenzo and J. Nalepa, "Memetic evolution of deep neural networks," in *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, ACM, July 2018.

[106] A. I. Sharaf and E.-S. F. Radwan, *An Automated Approach for Developing a Convolutional Neural Network Using a Modified Firefly Algorithm for Image Classification*, pp. 99–118. Springer Singapore, Nov. 2019.

[107] L. Frachon, W. Pang, and G. M. Coghill, "Immunecs: Neural committee search by an

artificial immune system," *arXiv preprint arXiv:1911.07729*, 2019.

[108] J. Park, S. Woo, J.-Y. Lee, and I. S. Kweon, "Bam: Bottleneck attention module," *arXiv preprint arXiv:1807.06514*, 2018.

[109] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[110] D. Misra, T. Nalamada, A. U. Arasanipalai, and Q. Hou, "Rotate to attend: Convolutional triplet attention module," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 3139–3148, 2021.

[111] L. Yang, R.-Y. Zhang, L. Li, and X. Xie, "Simam: A simple, parameter-free attention module for convolutional neural networks," in *Proceedings of the 38th International Conference on Machine Learning* (M. Meila and T. Zhang, eds.), vol. 139 of *Proceedings of Machine Learning Research*, pp. 11863–11874, PMLR, 18–24 Jul 2021.

[112] B. S. Webb, N. T. Dhruv, S. G. Solomon, C. Tailby, and P. Lennie, "Early and late mechanisms of surround suppression in striate cortex of macaque," *The Journal of Neuroscience*, vol. 25, pp. 11666–11675, Dec. 2005.

[113] Z. Peng, W. Huang, S. Gu, L. Xie, Y. Wang, J. Jiao, and Q. Ye, "Conformer: Local features coupling global representations for visual recognition," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 367–376, 2021.

[114] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[115] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, T. Eccles, J. Bruce, A. Razavi, A. Edwards, N. Heess, Y. Chen, R. Hadsell, O. Vinyals, M. Bordbar, and N. de Freitas, "A generalist agent," *arXiv preprint arXiv:2205.06175*, 2022.

[116] S. Liu, E. Johns, and A. J. Davison, "End-to-end multi-task learning with attention," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1871–1880, June 2019.

[117] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[118] "t-SNE visualization of CNN codes — cs.stanford.edu." https://cs.stanford.edu/people/karpathy/cnnembed/. [Accessed 17-12-2023].

[119] B. Guo and M. Meng, "The encoding of category-specific versus nonspecific information in human inferior temporal cortex," *NeuroImage*, vol. 116, pp. 240–247, Aug. 2015.

[120] A. Riazi, "Genetic algorithm and a double-chromosome implementation to the traveling salesman problem," *SN Applied Sciences*, vol. 1, p. 1397, Oct 2019.

[121] H. M. Pandey, "Performance evaluation of selection methods of genetic algorithm and network security concerns," *Procedia Computer Science*, vol. 78, pp. 13–18, 2016. 1st International Conference on Information Security and Privacy 2015.

[122] J. Zhao, Y. Peng, and X. He, "Attribute hierarchy based multi-task learning for fine-

grained image classification," *Neurocomputing*, vol. 395, pp. 150–159, 2020.

[123] J. P. Perez and C. A. Perez, "Face patches designed through neuroevolution for face recognition with large pose variation," *IEEE Access*, vol. 11, pp. 72861–72873, 2023.

[124] Y. Y. L. Z. Y. Q. Luqi Yan, Jin Han, "Sentiment analysis of short texts based on parallel densenet," *Computers, Materials & Continua*, vol. 69, no. 1, pp. 51–65, 2021.

[125] T. Zhou, X. Ye, H. Lu, X. Zheng, S. Qiu, and Y. Liu, "Dense convolutional network and its application in medical image analysis," *BioMed Research International*, vol. 2022, pp. 1–22, Apr. 2022.

[126] W. Bian, J. Wang, B. Zhuang, J. Yang, S. Wang, and J. Xiao, "Audio-based music classification with densenet and data augmentation," in *PRICAI 2019: Trends in Artificial Intelligence: 16th Pacific Rim International Conference on Artificial Intelligence, Cuvu, Yanuca Island, Fiji, August 26-30, 2019, Proceedings, Part III 16*, pp. 56–65, Springer, 2019.

[127] S. Jégou, M. Drozdzal, D. Vázquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," *CoRR*, vol. abs/1611.09326, 2016.

[128] G. Larsson, M. Maire, and G. Shakhnarovich, "Fractalnet: Ultra-deep neural networks without residuals," *arXiv preprint arXiv:1605.07648*, 2016.

[129] B. Fielding and L. Zhang, "Evolving image classification architectures with enhanced particle swarm optimisation," *IEEE Access*, vol. 6, pp. 68560–68575, 2018.

[130] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Completely automated cnn architecture design based on blocks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, pp. 1242–1254, Apr. 2020.

Anexos

Acrónimos

CNN Redes neuronales convolucionales / *Convolutional Neural Networks*

MCGA Algoritmo genético multi-cromosómico / *Multi-chromosomic Genetic Algorithm*

RS *Random Search*

LLM *Large Language Models*

IA Inteligencia Artificial

ANI *Artificial Narrow Intelligence*

AGI *Artificial General Intelligence*

ASI *Artificial Super Intelligence*

AI *Artificial Intelligence*

ML *Machine Learning*

BioA *Bio-inspired Algorithms*

ANN *Artificial Neural Networks*

DL *Deep Learning*

NAS *Neural Architecture Search*

GO Métodos de optimización por gradiente / *Gradient-based Optimization*

RL Aprendizaje reforzado / *Reinforcement Learning*

BO Optimización Bayesiana / *Bayesian Optimization*

EA Algoritmos Evolutivos / *Evolutionary Algorithms*

AG / GA Algoritmo(s) Genético(s) / *Genetic Algorithm*

LGN Núcleos Geniculados Laterales

V1 Corteza Visual Primaria

VIP Vía de información visual / *Visual Information Pathway*

IT Corteza temporal inferior / *Inferotemporal Cortex*

MLP Perceptrón multicapa / *Multilayer perceptron*

SGD Gradiente Descendiente Estocástico / *Stochastic Gradient Descent*

FC *Fully Connected*

ILSVRC *ImageNet Large Scale Visual Recognition Challenge*

ReLU *Rectified Linear Unit*

autoML *Automatic Machine Learning*

DE *Differential Evolution*

PSO *Particle Swarm Optimization*

EN *Ensemble Networks*

VAN Red de atención ventral / *Ventral Attention Network*

RNN Redes neuronales residuales / *Residual Neural Networks*

EA *Evolutive Strategies*

ACO *Ant Colony Optimization*

FA *Firefly Algorithm*

AIS *Artificial Inmune System*

BAM *Bottleneck Attention Module*

CBAM *Convolutional Block Attention Module*

TA *Triplet Attention*

SimAM *Simple Attention Module*

ViT *Vision Transformer*

MTL Aprendizaje Multitarea / *Multitask Learning*

NAT *Neural Architecture Transfer*

CC Cromosoma Convolucional

BL *Bottleneck Layer*

TL *Transition Layer*

DC Cromosoma Denso

DA Aumentación de datos / *Data Augmentation*