Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California

Doctorado en Ciencias en Ciencias de la Computación

Categorización robusta de imágenes de arte usando Brain Programming

Tesis

para cubrir parcialmente los requisitos necesarios para obtener el grado de

Doctor en Ciencias

Presenta:

Mariana Alejandra Chan Ley

Ensenada, Baja California, México

2022

Tesis defendida por

Mariana Alejandra Chan Ley

y aprobada por el siguiente Comité

Dr. Gustavo Olague Caballero

Director de tesis

Dr. César Cruz Hernández

Dr. Pedro Gilberto López Mariscal

Dr. Vitaly Kober

Dr. Eddie Helbert Clemente Torres

Dr. Pedro Gilberto López Mariscal

Coordinador del Posgrado en Ciencias de la Computación

Dr. Pedro Negrete Regagnon

Director de Estudios de Posgrado

Resumen de la tesis que presenta Mariana Alejandra Chan Ley como requisito parcial para la obtención del grado de Doctor en Ciencias en Ciencias de la Computación.

Categorización robusta de imágenes de arte usando Brain Programming

Resumen aprobado por:

Dr. Gustavo Olague Caballero
Director de tesis

En este documento se describe un enfoque para clasificar automáticamente imágenes digitales de obras de arte de acuerdo al medio en el que fueron elaboradas. Normalmente, la tarea de clasificación de obras de arte a menudo se confía a personas expertas en el área, sin embargo, con los crecientes volúmenes de bases de datos digitalizadas, la tarea se vuelve abrumadora. En este sentido, la clasificación automática es útil para organizar grandes colecciones digitales. En este trabajo seguimos un paradigma llamado Brain Programming (BP), una metodología inspirada en el funcionamiento interno del sistema visual en el cerebro. El BP evoluciona modelos computacionales de la corteza visual artificial (AVC) para resolver problemas difíciles de clasificación de imágenes naturales. Incluimos una comparación con los métodos de clasificación en el estado del arte, así como un análisis de la robustez del algoritmo ante uno de los problemas más recurrentes en algoritmos de clasificación, las perturbaciones adversarias. En donde encontramos que el Brain Programming es capaz de obtener un desempeño comparable a las metodologías en el estado del arte, pero sin la susceptibilidad a los ataques adversarios.

Palabras clave: Brain Programming, Programación Genética, Evolución Piramidal, Corteza Visual Artificial, Clasificación de imágenes

Abstract of the thesis presented by Mariana Alejandra Chan Ley as a partial requirement to obtain the Doctor of Science degree in Computer Science.

Robust art image categorization using Brain Programming

Abstract approved by:

Dr. Gustavo Olague Caballero
Thesis Director

This thesis work describes an approach for automatic art media categorization. Nowadays, classifying artworks is often entrusted to people who are experts in the field; however, with the increasing volumes of digitized databases, the task becomes overwhelming. In this sense, automatic classification is necessary for organizing extensive digital collections. In this work, we follow a paradigm called Brain Programming (BP), a methodology inspired by the inner workings of the visual system in the brain. BP evolves computational models of the artificial visual cortex (AVC) to solve the challenging problem of natural image classification. We compare with the state-of-the-art classification methods and analyze the algorithm's robustness against one of the most recurrent problems in classification algorithms, adversarial perturbations. We find that Brain Programming can obtain comparable performance to state-of-the-art methodologies without susceptibility to adversary attacks.

Keywords: Brain Programming, Genetic programming, Pyramidal Evolution, Artificial virtual cortex, image classification, art media categorization

Dedicatoria

A mis amores Juan José y Anita

A la memoria de mi madre Ana Rubí

Agradecimientos

Al Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California por ser la sede de todo el conocimiento adquirido en estos años.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por brindarme el apoyo económico para realizar mis estudios de maestría/doctorado. No. de becario: 618301

A mi asesor de tesis, Dr. Gustavo Olague. Muchas gracias por su paciencia; por darme las palabras y consejos que atesoraré por el resto de mi vida. Pero, sobre todo, por su gran amistad.

A mis padres, Ana Rubí y Rosendo por el cuidado y apoyo que siempre me otorgaron. Sus palabras de aliento, su comprensión y amor. Gracias a ustedes llegué hasta el final.

A Juan José, mi compañerito de vida, por todo el amor, cariño, aventuras, alegrías y tristezas. Gracias por ser el apoyo incondicional en mi vida.

A mis compañeros del Laboratorio Axel, Aarón, Rocío y Gerardo por su gran amistad y por todas las pláticas amenas en el laboratorio.

A los miembros de mi comité de tesis, a quienes les agradezco por todo su tiempo, observaciones y sugerencias para mejorar mi trabajo de investigación.

Tabla de contenido

Página

Tabla de contenido (continuación)

Lista de figuras

Figura Página

Lista de figuras (continuación)

Lista de tablas

Lista de tablas (continuación)

Capítulo 1. Introducción

En contraste con la típica clasificación de objetos en imágenes, que muchos consideran resuelto, el problema de clasificación de imágenes de arte presenta un nuevo conjunto de desafíos. En este sentido, la clasificación no está necesariamente relacionada con los objetos que contiene, si no con características que las obras comparten en común.

Otra diferencia fundamental es que las clases no siempre son discretas o mutuamente excluyentes, como en el aprendizaje automático supervisado. Por ejemplo, la transición de estilo a lo largo del tiempo suele ser suave, y las etiquetas de estilo son conceptos posteriores impuestos por los historiadores del arte, a veces siglos después. Las pinturas pueden tener elementos que pertenecen a múltiples estilos y no siempre es posible identificarlos con una clase de estilo única (Elgammal *et al.*, 2018b).

Existen diferentes maneras de clasificar las obras de arte, algunas de las formas más utilizadas son con respecto al estilo empleado, que se relaciona con la escuela a la cual pertenece el artista que creó la obra. También se puede clasificar con respecto al género, o de acuerdo al médium o medio empleado para crearla(Chan-Ley y Olague, 2020). Algunos autores se enfocan en hacer la clasificación con respecto al artista que creó la obra (Li *et al.*, 2011), (Elgammal *et al.*, 2018a),(Johnson *et al.*, 2008) , estos trabajos son comúnmente utilizados para identificar obras de artistas famosos y poder definir si una pieza recientemente encontrada, pertenece o no a un artista en particular.

Otro desafío presente, es la falta de imágenes en una escala de magnitud similar a la de ImageNet (Russakovsky *et al.*, 2015) (millones de imágenes). Esta limitación se debe a la cuestión de los derechos de autor, que es parte integral del dominio del arte. Además, recopilar anotaciones en este dominio es difícil, ya que requiere anotadores expertos y los anotadores típicos no están calificados (Elgammal *et al.*, 2018b). Si bien, ha habido un creciente interés en esta área, todavía faltan bases de datos y procedimientos de evaluación comunes, similares a los que se encuentran en la recuperación y anotación de imágenes fotográficas, como: Pascal VOC(Everingham *et al.*, 2015) o Imagenet.

El estudio de la clasificación de imágenes de arte abre la puerta al trabajo interdisciplinario, ya que antecede a la comprensión de la imagen artística. Para la comunidad de visión por computadora, este problema ofrece desafíos donde se pueden desarrollar nuevas técnicas; y para la comunidad de historia del arte se pueden desarrollar nuevas herramientas automáticas de análisis de arte.

En general, el problema de aprendizaje parte de un universo (usualmente infinito) Ω que representa el concepto que queremos aprender en forma de datos (ejemplos, imágenes) $\boldsymbol{x} \in \Omega$. Una suposición

básica es que los datos son uniformes y representan ejemplos específicos bien definidos del problema de aprendizaje abordado.

En programación genética, el problema se formula mediante combinaciones por pares de la forma $(\boldsymbol{y}, \boldsymbol{x})$ donde cada dato de muestra ($\boldsymbol{x}$, dominio) se empareja con un salida ($\boldsymbol{y}$, codominio) a través de una función en un lenguaje matemático que generalmente incluye operadores algebraicos, variables, escalares, etc. Si tal relación analítica existe de manera directa, por ejemplo, cuando alimentamos el programa con datos $y_i = f_{target}(x_i)$, entonces el objetivo es escribir ecuaciones candidatas $\hat{f}$(vocabulary) hasta que alguna llegue al objetivo $\hat{f} = f_{target}$ haciendo predicciones del valor $\hat{y}$ de la variable objetivo para algunos nuevo valor $\hat{x}$ de la variable de entrada.

En caso de que el algoritmo no logre encontrar una buena aproximación de la función objetivo con los datos proporcionados (conjunto de entrenamiento), es decir que no tenga buenas propiedades de generalización, mantenemos la mejor solución descubierta hasta ahora, lo que significa que queremos descubrir funciones que se comporten bien con datos nunca antes vistos (prueba y validación).

Además, esto se considera un problema intrínsecamente difícil, ya que tenemos que generalizar a partir de un conjunto de datos finito. Además, los datos observados se corrompen con ruido, por lo que para un $\hat{x}$ dado, existe incertidumbre en cuanto al valor apropiado para $\hat{y}$.

Los algoritmos genéticos y en particular la programación genética son metodologías que pueden abordar problemas de aprendizaje automático (Shapiro, 2001; Pétrowski y Ben-Hamida, 2017). Sin embargo, en los últimos años la comunidad científica ha usado de manera prácticamente exclusiva un paradigma basado en redes neuronales (He *et al.*, 2016) y lo han considerado omnisciente para el aprendizaje automático.

La comunidad de programación genética está buscando formas novedosas que puedan igualar los resultados de las redes neuronales actuales mediante el desarrollo de su paradigma principal. Las redes neuronales convolucionales profundas (DCNN [1]) presentan ciertas vulnerabilidades que las hacen suceptibles de equivocarse en su predicción. En los últimos años han surgido investigaciones dedicadas específicamente a hacer que estos modelos fallen, llamados ataques adversarios. Éstos consisten en una perturbación a una imagen, pero que es prácticamente imperceptible al ojo humano y aun así producen un bajo desempeño en las DCNN. En lugar de aplicar las tendencias líderes en el aprendizaje automático, para este trabajo seguimos un paradigma llamado Brain Programming (BP), una metodología inspirada en el funcionamiento interno del sistema visual en el cerebro (Olague y Chan-Ley, 2020). El Brain Pro-

[1] por sus siglas en inglés: Deep Convolutional Neural Networks

gramming ha probado su enorme capacidad en problemas de clasificación, donde ha logrado encontrar resultados comparables a las técnicas del estado del arte.

El BP evoluciona modelos computacionales de la corteza visual artificial (AVC[2]) para resolver problemas difíciles de clasificación de imágenes naturales. Este modelo propuesto en (Olague *et al.*, 2014a) muestra un excelente desempeño para resolver el caso de ausencia/presencia para problema de reconocimiento de objetos. En este trabajo lo emplearemos con el problema de categorización de imágenes de arte.

También incluimos en este trabajo un análisis sobre la robustez del Brain Programming, en donde sometimos a las mejores soluciones encontradas a los ataques adversarios.

1.1. Objetivos

Se definen los siguientes objetivos para este trabajo de investigación

Objetivo general

- Diseñar un sistema robusto de clasificación de imágenes de obras de arte digitalizadas de acuerdo al medio en el que fueron elaboradas

Objetivos específicos

- Realizar un estudio de las técnicas de clasificación actuales de las imágenes de obras de arte y la forma en que se ha abordado el problema en el área de ciencias de la computación.
- Realizar un estudio comparativo del sistema clasificador propuesto y los modelos del estado del arte.
- Evaluar la robustez del sistema propuesto mediante su resistencia a ataques adversarios. Comparar la robustez del sistema con la de otros métodos del estado del arte.

1.2. Planteamiento del problema

Para realizar el análisis, definiremos el problema de la programación genética desde el punto de vista del modelado de datos, similar a la explicación anterior, con algunos tecnicismos añadidos que son

[2]por sus siglas en inglés Artificial Visual Cortex

necesarios para comprender mejor la idea. En general, un problema de minimización requiere encontrar una solución $\boldsymbol{P}_{min} \in S$ tal que $f(\boldsymbol{P}_{min})$ sea un mínimo global en S. Más específicamente, se requiere encontrar un $\boldsymbol{P}_{min} \in S$ tal que

$$\forall \boldsymbol{P} \in S : f(\boldsymbol{P}_{min}) \leq f(\boldsymbol{P})$$

A diferencia de los enfoques convencionales con respecto al simple hallazgo de los parámetros de mejor ajuste en la programación genética, nos gustaría encontrar una función que realice satisfactoriamente la tarea de ajustar los datos. Este proceso incluye varias etapas, ya que no existe un mapeo directo entre el dominio y el codominio o al menos no está bien definido. De esta forma, la solución al problema de clasificación de imágenes requiere definir el siguiente problema:

$$\boldsymbol{y} = \text{mín}\,(f(\boldsymbol{x}, \boldsymbol{F}, \boldsymbol{T}, \boldsymbol{\alpha})) \tag{1}$$

donde el conjunto de datos viene dado por $(\boldsymbol{y}, \boldsymbol{x})$, $\boldsymbol{F}$ representa el conjunto de funciones, $\boldsymbol{T}$ define el conjunto de terminales , y $\boldsymbol{\alpha}$ son los parámetros que controlan el funcionamiento del algoritmo. Por lo tanto, es necesario definir dos puntos antes de intentar resolver el problema: 1) el método de extracción de características y 2) un criterio adecuado $\boldsymbol{Q}$ para la minimización.

El AVC es el algoritmo encargado de la extracción de características de las imágenes de entrada. BP es el algoritmo encargado de ajustar $(\boldsymbol{F}, \boldsymbol{T}, \boldsymbol{\alpha})$ para cada operador visual embebido en el AVC. En el presente trabajo, el criterio para la optimización es un clasificador, implementado con una máquina de soporte vectorial (SVM). Por lo tanto, un SVM es entrenado para realizar un mapeo $f(\mathbf{x})$ que asocie los descriptores $\mathbf{x}_i$ a etiquetas y_i. Formulamos nuestro problema en términos de una tarea de clasificación binaria, cuyo principal objetivo es encontrar una superficie de decisión que separe mejor los elementos de las clases. En este trabajo, usamos una SVM no lineal que trabaja con el hiperplano discriminante definido por:

$$f(x) = \sum_{i=1}^{l} \alpha_i y_i K(x_i, x) + b, \tag{2}$$

donde los datos de entrenamiento dados son $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, l$, $y_i \in \{-1, 1\}$, $\mathbf{x}_i \in \mathbf{R}^p$ y $K(\mathbf{x}_i, \mathbf{x})$ es la función kernel. El signo de la salida indica la pertenencia a la clase $\mathbf{x}$. Por lo tanto, la búsqueda

del mejor hiperplano se realiza mediante un proceso de optimización que utiliza el margen entre la clase y la no clase como criterio de búsqueda.

En resumen, podemos decir que el problema de minimización funciona con la quintupla de aprendizaje $((\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{F}, \boldsymbol{T}, \boldsymbol{a}, \boldsymbol{Q})$.

1.3. Aportes de este trabajo de tesis

Este trabajo de investigación representa la intersección en el estado del arte del problema de clasificación de imágenes de obras de arte y los ataques adversarios. También incluimos una primera aproximación a una nueva propuesta que organiza la información de la base de datos y a la población empleada para reducir el tiempo necesario para evolucionar un ciclo completo.

En este trabajo, ampliamos los primeros resultados informados en Olague y Chan-Ley (2020) y Chan-Ley y Olague (2020) en donde presentamos el Brain Programming como propuesta para resolver el problema de categorización de imágenes de arte de acuerdo al medio. Asi como los trabajos presentados en el Simposio Internacional sobre Computación Visual (ISVC'20) y el 11º Taller GECCO sobre Computación Evolutiva para el Diseño Automatizado de Algoritmos (ECADA'21), en los que se presentó un estudio sobre la robustez del Brain Programming a los ataques adversarios, a través de la desafiante tarea de clasificación de imágenes de arte y una primera propuesta para resolver la tarea de reconocimiento facial Olague *et al.* (2020); Ibarra-Vazquez *et al.* (2021).

Queremos determinar si los ataques adversarios también afectan a otros métodos de clasificación de imágenes más allá del aprendizaje automático. Es por esto que realizamos un estudio sobre la robustez del Brain Programming en la tarea de clasificación de imágenes de arte. Probamos contra cuatro redes neuronales convolucionales profundas de última generación (AlexNet, VGG, ResNet, ResNet101) y un enfoque de visión por computadora tradicional (Bag of Visual Words) usando amenazas directas e indirectas con tres ataques adversarios (Fast Gradient Sign Método: FGSM, ataque de múltiples píxeles y parche adversario). Además, empleamos pruebas estadísticas para validar la robustez del algoritmo.

Por todo lo anterior, destacamos las siguientes contribuciones de este trabajo:

- La primera contribución es la aplicación de una metodología de Cómputo evolutivo llamada Brain Programming para el difícil problema de clasificación de imágenes de arte, en específico abordamos la clasificación de acuerdo al medio utilizado. El BP es capaz de tener un resultado comparable e

incluso superar en algunas categorías a los métodos del estado del arte.

- La segunda contribución es una demostración de la inmunidad de nuestra metodología ante los ataques adversarios. Demostramos que el Brain Programming posee inmunidad a este tipo de ataque. En vez de resolver la vulnerabilidad en los sistemas de aprendizaje automático, una alternativa es utilizar metodologías que compitan con redes neuronales convolucionales profundas y que no posean la vulnerabilidad a estos ataques.

- La tercera contribución es una primera propuesta de un nuevo modelo evolutivo. En vez de utilizar fuerza bruta y simplemente dejar que el ciclo evolutivo haga toda la tarea, proponemos una nueva forma de organizar los datos y proveer al algoritmo la información que necesita por partes. Organizamos los datos utilizados en una estructura piramidal que nos proporcione una via natural para incrementar la presión selectiva a nuestra población, con lo que logramos obtener un resultado comparable a un ciclo evolutivo normal, pero utilizando menos recursos computacionales.

1.4. Organización de la tesis

Este trabajo de tesis se encuentra organizado de la siguiente manera. En la sección 2 se explica ampliamente el funcionamiento del Brain Programming, las variantes que han surgido a lo largo de los años, y una revisión de los principales trabajos de investigación que han surgido al respecto.

En la sección 3 se presenta un estudio sobre los trabajos relevantes que abordan el problema de clasificación de imágenes de arte. Cubre la complejidad del problema y cómo los investigadores lo abordan utilizando diferentes herramientas, desde funciones artesanales hasta modelos DCNN y métodos evolutivos.

En la sección 4 podemos encontrar la propuesta de una metodología para optimizar el tiempo de ejecución del Brain programming, mediante la introducción de una estructura piramidal que organiza los datos y las soluciones candidatas.

La sección 5 describe los experimentos llevados a cabo, y proporciona una discusión sobre los resultados obtenidos. En esta sección se incluye un análisis de la robustez de las soluciones encontradas por el programa. Tambien se incluyó un apartado sobre la descripción de los ataques adversarios empleados.

Finalmente el capítulo 6 presenta las conclusiones de este trabajo de tesis así como las propuestas para continuar esta línea de investigación.

Capítulo 2. Ocho años de Brain Programming

El Brain Programming es un método de aprendizaje máquina basado en programación genética (Koza, 1992); es inspirado en el funcionamiento del cerebro, en particular de la corteza visual, para abordar problemas de visión artificial, tales como clasificación, detección o seguimiento de objetos.

Los trabajos pioneros que introdujeron esta metodología fueron los presentados por Olague *et al.* (2014a) y Dozal *et al.* (2014), en donde propusieron una nueva estrategia basada en programación genética y la idea de que el cerebro y el sentido de la vista evolucionaron hasta el punto en el que se encuentran actualmente y aún están sujetos al proceso de evolución.

En la actualidad la mayoría de los enfoques utilizados para abordar problemas de visión se basan en redes neuronales convolucionales, sin embargo, esta técnica cuenta con vulnerabilidades y problemas intrínsecos, que aun despues de varios años de investigación continúan mermando su desempeño. El Brain Programming surge como una alternativa confiable, robusta y capaz de encontrar soluciones con rendimientos comparables a los que se encuentran en el estado del arte, y en muchos casos superarlos con creces. Aunado también a otras ventajas de éste método que serán descritas en las secciones posteriores.

En este capítulo se presenta a detalle el funcionamiento del Brain Programming, pero antes de ahondar en ello se hace una explicación del sistema visual humano y las teorías que ayudaron a sentar las bases de este paradigma.

2.1. Cómputo evolutivo

Existen muchas metodologías computacionales inspiradas por el proceso evolutivo explicado por Charles Darwin, estas metodologías son conocidas como computación evolutiva, y la idea común entre todas ellas puede ser resumida con el término "Darwinismo artificial", que consiste de los mismos principios y componentes básicos del Darwinismo (Olague, 2016).

Cualquier algoritmo evolutivo esta compuesto básicamente por dos principios evolutivos: la selección natural y la herencia genética. De esta forma cualquier algoritmo evoultivo consiste de: una población de individuos, una función objetivo, un paso de selección de los individuos más aptos, y un proceso de recombinación que se repiten de manera cíclica hasta obtener una solución satisfactoria o se alcanza cierto número de generaciones. Un ejemplo básico puede observarse en el Algoritmo 1.

Crea una *Población inicial* P_0 aleatoria y establece $i \leftarrow 0$.

Evalúa el desempeño de cada individuo en P_i usando la función objetivo.

mientras *no encuentre una solución aceptable o se cumpla algun otro criterio de paro (ejemplo: se alcance algun número máximo de generaciones)*

hacer

Selecciona los padres del conjunto P_i basándose en su aptitud.

Recombina los padres seleccionados para producir una población P_{i+1} y establece $i \leftarrow i + 1$

Evalúa el desempeño de la nueva población usando la función objetivo.

fin

devolver El mejor individuo como resultado de la evolución.

Algoritmo 1: Algoritmo evolutivo básico

En la computación evolutiva existen diferentes técnicas o algoritmos evolutivos (AE), que comparten muchas características, pero varían detalles como la representación de los soluciones, las técnicas que se utilizan para encontrar nuevas soluciones y algunos parámetros del proceso evolutivo. Sin embargo, todos funcionan bajo el mismo principio; es decir, son heurísticas estocásticas inspiradas en la teoría de la evolución biológica y cuyo funcionamiento se basa en los mismos mecanismos (Eiben *et al.*, 2003):

- El genotipo, la codificación que representa una posible solución, a cada solución se le conoce como individuo. Al conjunto de individuos se le llama población.
- La función objetivo, es una función que evalúa el rendimiento de un individuo al resolver el problema en cuestión. El resultado de aplicar un individuo para resolver un problema se le conoce como fenotipo. Esta función asigna un valor de aptitud a cada solución, que representa la calidad de adaptación de un individuo a un ambiente, en este caso, el ambiente representa el problema.
- Selección, es una estrategia para elegir un conjunto de individuos que se utilizan como base para generar nuevas soluciones.
- Mecanismos de variación, son técnicas para generar nuevas soluciones (hijos), a partir de individuos previamente seleccionados. Existen dos mecanismos principales: la cruza o recombinación, que consiste en utilizar la información de dos o más individuos (padres) para generar una nueva solución; y la mutación, la cual toma un individuo y utiliza un proceso estocástico para generar una nueva.

Existen diferentes tipos de algoritmos evolutivos. El Brain Programming en particular se inspira

principalmente de la programación genética (GP[1]) propuesta por Koza (1992).

La diferencia principal entre la GP y otras técnicas evolutivas radica en la representación de los individuos. En la GP las soluciones se definen como árboles sintácticos. A diferencia de los algoritmos genéticos, que se utilizan generalmente para la optimización de parámetros, la GP se enfoca en la síntesis de programas o funciones computacionales. Por ende, la GP se denota como una técnica para generar programas de manera automática

Es por ello que se establece que el principal objetivo del BP es el de optimizar modelos complejos ajustando las operaciones que se realizan dentro de ellos. Para tener una idea de lo expuesto se puede observar la Figura 1.

El Brain Programming está compuesto de varias etapas resumidas en dos ideas clave. En primer lugar, cuenta con una estructura jerárquica inspirada en la corteza visual humana; ésta utiliza el concepto de composición de funciones para extraer características de las imágenes. La segunda parte es un proceso evolutivo basado en programación genética, cuyo propósito principal es descubrir funciones que optimicen los modelos complejos ajustando las operaciones que se usarán en la estructura descrita en el punto anterior (Chan-Ley y Olague, 2020).

El Brain Programming resulta una herramienta muy versátil, dependiendo de la tarea que vamos a realizar, podemos cambiar el modelo, entre la Ruta Dorsal Artificial (ADS[2]) para problemas de detección de objetos o de Atención Visual, o la Corteza Visual Artificial (AVC [3]) para problemas de clasificación. Los detalles de éstas dos partes serán explicados a detalle en la sección 2.2.

Todas la soluciones generadas por el programa, comparten esa misma estructura jerárquica conformada por el modelo que se esté utilizando, la diferencia entre una solución y otra, son las funciones elegidas por el ciclo evolutivo. De esta forma, se sigue la analogía en donde la evolución modifica la funcionalidad de las áreas del cerebro adaptándola a diferentes tareas, pero sin alterar el orden en que el cerebro realiza estas acciones.

En Hernández *et al.* (2016) se utilizó el sistema visual humano como inspiración para resolver tareas de detección y clasificación de objetos. Se desarrollaron modelos computacionales de una corteza visual artificial (AVC) para resolver el problema de clasificación de objetos en imágenes naturales.

La Figura 1 presenta un esquema para visualizar la idea general del sistema propuesto para la clasi-

[1]por sus siglas en inglés: genetic programming
[2]por sus siglas en inglés: Artificial Dorsal Stream
[3]por sus siglas en inglés: Artificial Virtual Cortex

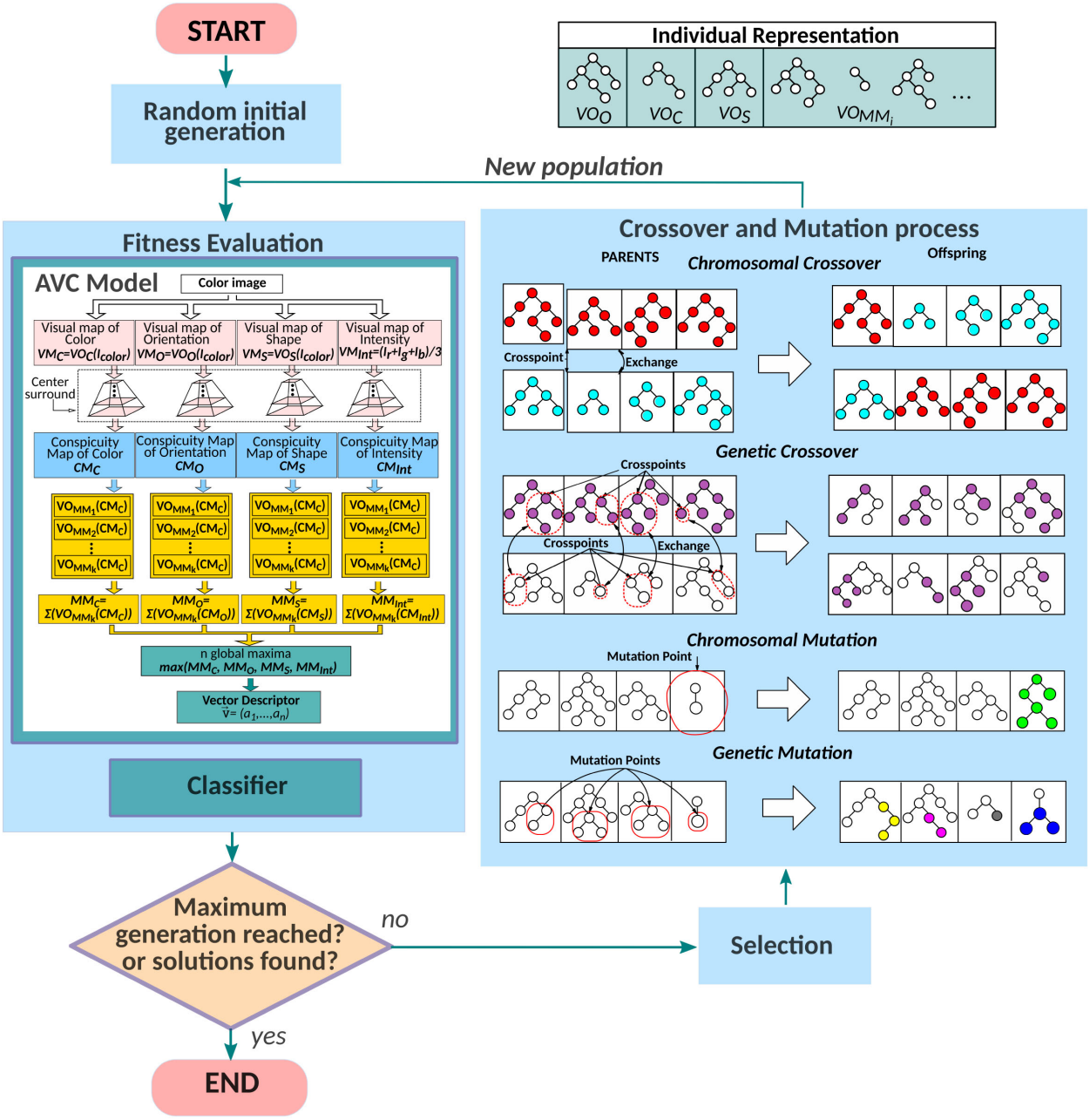

Figura 1. Diagrama de flujo del Brain Programming, Chan-Ley y Olague (2020)

ficación de imágenes. La idea es encontrar un vector descriptor del objeto para clasificación y al mismo tiempo encontrar la ubicación del objeto dentro de la imagen. En ese trabajo se presenta el paradigma de la programación cerebral desde una perspectiva multiobjetivo.

Un solo operador de las soluciones generadas podría no ser de interés, ya que solo tienen significado mientras trabajan con otros operadores dentro de la estructura jerárquica.

La función de fitness utilizada es un equilibrio entre las rutas dorsal y ventral artificiales. Cada solución se evalúa a través de dos medidas de desempeño. El primero mide el rendimiento de la clasificación con

una métrica llamada tasa de error (eer [4]), que define la probabilidad de que un algoritmo decida si dos instancias corresponden a la misma clase. El segundo objetivo se basa en la medida F, que se utiliza como una forma de calcular la correspondencia entre una verdad fundamental de la ubicación del objeto en la imagen y la región seleccionada por el algoritmo como la posición posible. La metodología fue probada en las bases de datos GRAZ-01 y GRAZ-02. Las soluciones coinciden y en algunos casos superan a otras técnicas en el estado del arte para clasificar las bases de datos antes mencionadas.

La corteza visual artificial es un modelo bioinspirado para la clasificación de objetos que es computacionalmente costoso, principalmente debido en parte al alto contenido de información que es necesario procesar mientras se trabaja con grandes bases de datos, con imágenes de alta resolución, o en el peor de los casos, cuando necesitan analizar video. En Hernández *et al.* (2017), el modelo original se mejoró a través de la arquitectura de dispositivo unificado de cómputo (CUDA) mediante la explotación de las capacidades computacionales de las unidades de procesamiento de gráficos (GPU). En este trabajo, la corteza visual artificial se codificó en CUDA para lograr una funcionalidad en tiempo real. Como resultado, el sistema que propusieeron pudo procesar imágenes hasta 90 veces más rápido que las imágenes originales. Además, cuando el tamaño de la imagen crece, el AVC-CUDA propuesto es más rápido en comparación con otros enfoques como CNN-CUDA.

2.2. Estructura jerárquica neuroinspirada

2.2.1. ¿Por qué inspirarse en el cerebro?

Una gran cantidad de información es percibida a cada momento por el cerebro humano, mucha más de la que puede ser procesada eficientemente. Sin embargo, la detección y el reconocimiento de objetos son tareas que usualmente se realizan con un mínimo de esfuerzo. En contraste, en visión por computadora la detección y el reconocimiento de objetos son unos de los problemas más difíciles y recurrentes (Forsyth y Ponce, 2003).

Las aplicaciones de la visión por computadora van desde la video vigilancia, monitoreo de tráfico, sistemas de asistencia para conductores, e inspección industrial hasta la interacción entre computadora y humano, recuperación de imágenes en bibliotecas digitales y análisis de imágenes médicas. En robótica, la detección de obstáculos, manipulación de objetos, creación de mapas semánticos, y la detección de puntos de referencia para la navegación, son tareas imprescindibles.

[4] del ingles: equal error rate

Como se mencionó anteriormente, las técnicas actuales de detección, reconocimiento o clasificación de objetos, se basan principalmente en redes nueronales. Éstas no imitan directamente la organización de la corteza visual, pero a menudo se cita la biología como fuente de inspiración. Si bien estos modelos no son estrictamente hablando, modelos del sistema visual, su impresionante éxito en múltiples tareas de reconocimiento visual ofrecen evidencia de apoyo para los modelos jerárquicos del sistema visual.

Se han desarrollado varios sistemas sofisticados para tareas especializadas como la detección de caras (Viola y Jones, 2004) o peatones (Papageorgiou *et al.*, 1998), pero el sistema visual humano, es capaz de reconocer miles de objetos desde diferentes puntos de vista, bajo condiciones de iluminación variables y con oclusiones parciales sin necesidad de cambios en nuestro modelo.

Uno de los mecanismos que hace a los humanos tan efectivos para actuar en la vida diaria es la habilidad de la atención visual. La información más relevante es extraída y dirigida a áreas más profundas del cerebro donde toman lugar procesos complejos tales como el reconocimiento de objetos. Restringir estos procesos únicamente a un subconjunto limitado de los datos sensoriales aumenta su eficiencia. Esa gran capacidad de ordenar la información que posee el cerebro, parece estar lejos todavía para los sistemas artificiales. Aún se puede mejorar el rendimiento de los sistemas computacionales con la búsqueda de inspiración en los sistemas biológicos y simular sus mecanismos de procesamiento.

A continuación se explicará con más detalle el funcionamiento del sistema visual humano y las teorías que sirvieron como inspiración para crear el modelo del Brain Programming.

2.2.1.1. El sistema visual humano

El sistema visual humano puede ser dividido en dos partes principales que describen su funcionamiento: la corteza visual y los ojos. En el presente trabajo nos centraremos en la corteza visual, dado que es la parte del cerebro que recibe la información de la imagen y la procesa para cumplir funciones como detección, reconocimiento de objetos, seguimiento, entre otras. Dentro de ésta, las áreas principales de procesamiento visual son las llamadas ruta ventral (VS [5]) y la ruta dorsal (DS [6]) (ref.pdte).

La ruta ventral (o ruta del qué) es definida como el área donde se realizan los procesos de reconocimiento de objetos y/o percepción de estímulos visuales. La ruta dorsal (o ruta del dónde/cómo), es el área relacionada con la localización en el espacio y la dirección de la atención hacia las zonas de interés en la escena (Itti y Koch, 2001). Estas dos áreas trabajan conjuntamente procesando la imagen precibida

[5] por sus siglas en inglés: Ventral Stream
[6] por sus siglas en inglés: Dorsal Stream

por los ojos y realizando las funciones visuales que se utilizarán posteriormente en otras áreas como el sistema motriz, la memoria, o el sistema de aprendizaje.

Cabe señalar que la imagen reproducida por las células de la retina no tiene la misma nitidez en todas sus partes debido a la distribución de los conos y bastones. La nitidez se refiere a la resolución de la imagen (número de puntos por unidad de medida). En la retina, cada punto de la imagen es representado por un conjunto de conos -rojo, verde o azul-, o un bastón. La distribución de las células conos y bastones es no uniforme, y la zona del ojo que posee una mayor densidad de conos es la fóvea. Ésta es la zona de la retina donde se obtiene la parte más nítida de la imagen.

El campo visual del ojo tiene un ángulo de 200 grados aproximadamente, en tanto que la fóvea percibe la luz en un ángulo de 15 grados (Chun y Wolfe, 2001). Sin embargo, de la información total que el ojo manda al cerebro, aproximadamente la mitad proviene de la fóvea y el resto de la retina. En las imágenes que captura el ojo, una parte pequeña tiene gran detalle. Algunas actividades que requieren la capacidad de percibir detalles son la lectura, el reconocimiento de rostros, la identificación y seguimiento de objetos, el aprendizaje visual, entre otras. Para este tipo de actividades es preciso que el ojo se mueva para posicionar el área que desea ubicar dentro del campo de la fóvea.

Cuando se analiza una imagen o se recibe algún estímulo, como en movimientos o cambios súbitos en la imagen, la corteza visual (particularmente la ruta dorsal) tiene la capacidad de influir en el movimiento del ojo para que éste permita ubicar el área de la fóvea en la zona de la imagen que necesita procesar. Este conjunto de acciones se conoce como *Mecanismo de atención visual*.

Por medio de éste mecanismo, se realizan acciones como la selección de información relevante de una escena observada. Casi cualquier escena o ambiente visual es complejo y el cerebro no puede procesar la totalidad de la información observada (Chun y Wolfe, 2001).

Para abordar este problema, el cerebro tiene mecanismos de atención que cumplen con dos importantes funciones. La primera es la selección de información relevante a la tarea que se esta realizando e ignorando la que es irrelevante o redundante. La segunda función tiene el objetivo de modular o realzar la información seleccionada de acuerdo al estado y los objetivos del perceptor (Chun y Wolfe, 2001).

2.2.1.2. Modelo de atención visual

Las primeras descripciones del mecanismo de atención visual aparecieron en la década de los 60´s, sin embargo hasta la década de los 90's se comenzaron a implementar en los sistemas de visión artificial.

En un principio, los trabajos para hacer reconocimiento u otras tareas visuales, realizaban un análisis previo de toda la escena utilizando métodos como segmentación, clasificación y métodos estadísticos. A partir de entonces han surgido diferentes modelos para el mecanismo de atención visual.

Dentro de las principales teorías que describen la atención visual está la Teoría de integración de características (Treisman y Gelade, 1980). Esta teoría propone que las características de una escena (color, intensidad, orientación, etc.) se perciben de forma paralela y automática. Por lo tanto, la atención visual tiene el propósito de unir correctamente éstas características para construir la representación de un objeto. El proceso completo está dividido en 3 fases: percepción, atención e identificación o reconocimiento.

Un modelo propuesto para explicar la forma en que el ser humano puede encontrar estímulos visuales deseados en una escena es el modelo de búsqueda guiada propuesto por Wolfe (1994). La primera versión se publicó en 1989 y se basa en la propuesta de Treisman y Gelade. El modelo guía la atención utilizando un valor llamado *activación*, de tal manera que mientras mayor sea el valor de activación de una zona en la escena, mayor disposición tendrá el modelo para dirigir la atención a dicha zona.

Se consideran 2 tipos de activación: de abajo hacia arriba (impulsada por estímulos) y de arriba hacia abajo. El primer caso consiste en una medida que define qué tan distinto es un elemento de su vecindario o localidad. Esto significa que se da un valor al contraste de un elemento con sus vecinos para cada una de las dimensiones que lo componen. En este sentido, una dimensión se refiere a una propiedad como orientación, forma, color, intensidad, disparidad o movimiento. El propósito principal de esta activación es guiar el foco de atención para centrarse sobre el elemento más constrastante en la escena sin necesidad de un conocimiento previo para tal acción. Cabe señalar que el contraste se refiere a las diferencias en cada una de las dimensiones que componen a la escena, y no solo a las diferencias en brillo o intensidad en la imagen.

La activación de arriba hacia abajo (o impulsada por usuario) es una medida para guiar el foco de atención hacia objetos que el usuario busca, que conoce previamente y desea encontrar en la escena, aunque no sean atrayentes o conspicuos. El mecanismo para lograr esto consiste en seleccionar las dimensiones o canales que contribuyen a encontrar el objetivo buscado, filtrando la información obtenida previamente por estímulo.

Una vez obtenida la medida de activación, se representa en un mapa de activación, que se puede visualizar como un conjunto de valles y montañas en una área que corresponde al tamaño de la imagen. De ésta forma, la atención se guía hacia el punto más alto de éste mapa.

Otro marco de referencia en el tema de atención visual es el trabajo de Christof Koch y Shimon Ullman (Koch y Ullman, 1987), donde se propone un modelo de atención visual basado en prominencia (Saliency-Based Visual Attention). En 1985, los autores concluyen que los primates poseen un sistema especializado de procesamiento por foco que se mueve a lo largo de la escena que observan. Proponen que dicho sistema esta formado por:

- Una fase de representación y mapeo selectivo en el que, se calcula en paralelo un conjunto de características elementales a través del campo visual. Éstas características se representan en un conjunto de mapas topográficos corticales donde cada uno corresponde a una característica como color, orientación, forma, disparidad, dirección de movimiento, etc. Las ubicaciones en el espacio visual que difieren de su entorno con respecto a un rasgo elemental como la orientación, el color o el movimiento se señalan en el mapa correspondiente. Estos mapas se combinan en el mapa de saliencia, codificando la conspicuidad relativa de la escena visual.

- Un mecanismo de selección realizado por una red neuronal WTA [7] que opera sobre este mapa señalando las localidades más destacadas o conspicuas, dirigiendo las propiedades correspondientes a una representación central (zona de foco) no topográfica de la zona seleccionada. Un modelo WTA se caracteriza porque cada neurona compite con las demás para su activación, y solo la neurona con el valor de activación más alto queda encendida, el resto se apaga.

- Una fase en la que las propiedades de la localidad previamente seleccionada son guiadas a la representación central. La red WTA cambia automáticamente a la próxima localidad más destacada, y este cambio es de acuerdo a las preferencias de proximidad o similaridad. El sistema visual procesa una escena en una forma secuencial y automática mediante la inspección selectiva de la información presente en localidades destacadas.

El mecanismo descrito anteriormente es aplicable al desplazamiento del foco atencional y a rutinas como seguimiento de contornos, conteo de objetos o señalamiento de un lugar específico.

Los algoritmos del modelo anterior fueron desarrollados por Itti *et al.* (1998). Una versión extendida, considera la representación de arriba hacia abajo; en la cual se usa la información de otras áreas (como el conocimiento) y un sistema de pesos que detemina cuáles características influyen más en el cálculo de la atención.

[7] del inglés Winner Take All

En 1959, Hubel y Wiesel (1959) establecieron los fundamentos para el estudio del sistema cortical de los gatos. En donde describieron las propiedades de las neuronas de la Corteza visual y el concepto de campo receptivo [8]. Ellos fueron los pioneros que propusieron el modelo de procesamiento jerárquico de información en la corteza visual, esta estructura se representa por una red formada por una capa de entrada, seguida por una serie de estructuras modulares también conocidas como células simples "S" y células complejas "C". Su idea intenta imitar las características y funcionalidad de las células simples, así como también las células complejas.

Con estos avances en las neurociencias, Fukushima (1980) propuso al predecesor de las Redes Neuronales Convolucionales (CNN [9]) llamado Neocognitron que es una red jerárquica que tiene la capacidad de asignar objetos similares a la misma categoría de objetos, independientemente de su posición o algunas variaciones morfológicas. En la práctica, este modelo es capaz de reconocer letras y números sin importar la posición o rotación.

Anteriormente, la metodología implementada para resaltar las regiones de interés que arroja el paradigma de atención visual codifican una serie de funciones que son acorde a los estudios neurológicos realizados, donde las funciones se mantienen fijas para cualquier imagen de entrada. Estas funciones vienen de una compleja codificación de operaciones que se aplican a diferentes fases de la ruta dorsal artificial. La metodología usada para este trabajo, el Brain Programming, describe una manera en la cual la ruta dorsal artificial puede optimizarse para imitar la funcionalidad de áreas especializadas del cerebro a través de un conjunto de operadores visuales

2.3. Ciclo evolutivo

Como se mencionó al incio de este capítulo, en el proceso evolutivo de la programación cerebral, se utiliza el enfoque clásico de la programación genética, donde se realiza la analogía con la evolución natural de las especies. Es decir, se genera una población inicial de individuos, donde los más aptos son seleccionados para sobrevivir. Después, los individuos seleccionados se reproducen para crear una nueva generación que reemplaza a sus padres que posteriormente es evaluada. Este ciclo se repite hasta alcanzar un número de generaciones u otro tipo de condición de finalización definida, por ejemplo, la mejor solución definida por un individuo obtiene cierto nivel de aptitud, o se alcanza cierto número de generaciones.

[8] del inglés: receptive field
[9] por sus Siglas en Inglés: Convolutional Neural Network

A continuación se dará una explicación detallada de lo que ocurre en cada una de las etapas del ciclo evolutivo, y se resaltarán las diferencias que tiene con respecto a la programación genética convencional.

2.3.1. Inicialización

Se genera aleatoriamente una población inicial con el método half-and-half, que consiste en que la mitad de la población serán individuos balanceados, y la otra mitad serán desbalanceados. La profundidad máxima elegida para los árboles se fija en 9 niveles. El tamaño del cromosoma completo (número de árboles) difiere dependiendo del objetivo que busquemos, si hablamos directamente del foco de atención para realizar la atención visual, la cantidad de árboles será fija en 4 . En cambio, si se desea emplear el modelo de la corteza visual completa, el número de árboles es inicializado aleatoriamente con una longitud mínima de 4 y máxima de 12 genes. Esto es debido a las necesidades específicas de cada modelo.

En esta etapa también se define la probabilidad de aplicación de los operadores genéticos. Debido a las características del BP es necesario redefinir algunos operadores genéticos que cubran las necesidades de la estructura jerárquica multiárbol. Éstos operadores son explicados a detalle en la sección 2.3.4. Un resumen de los parámetros comúnmente utilizados en el BP se describen en la tabla 7

Tabla 1. Inicialización de parámetros para el Brain Programming.

Parámeteros	Descripción
Generaciones	30
Población Inicial	30
Profundidad de árbol	Selección dinámica
Número de árboles	Dinámico (entre 4 y 12)
Máxima profundidad dinámica	7 niveles
Máxima profundidad real	9 niveles
Método de Selección	Torneo con presión lexicográfica
Supervivencia	Elitismo
Probabilidad de los operadores genéticos	
Cruzamiento a nivel cromosoma	0.4
Cruzamiento a nivel gen	0.4
Mutación a nivel cromosoma	0.1
Mutación a nivel gen	0.1

2.3.1.1. El genotipo de los individuos

La representación genética de los individuos o genotipo, consiste en un conjunto de árboles sintácticos que codifican el funcionamiento de diferentes áreas del cerebro. Cada árbol tiene su propio conjunto de funciones y terminales que fueron cuidadosamente elegidos de acuerdo a la función que estamos tratando de emular.

Cada árbol corresponde a un operador visual (*OV*) de la corteza cerebral. Estos árboles corresponden a los operadores visuales de orientación, color, forma y otros correspondientes al modelo utilizado y a la tarea a resolver. Estos tres operadores (color, orientación y forma) han sido constantes en el modelo del Brain Programming y se utilizan en todas las tareas que se han abordado. Dependiendo de la tarea a resolver, se evolucionará un un los mapas mentales. La idea es evolucionar a cada uno de estos *OV*.

El primer árbol imita la operación de las células sensibles a la orientación presentes en la zona V1. De manera que, el operador de orientación

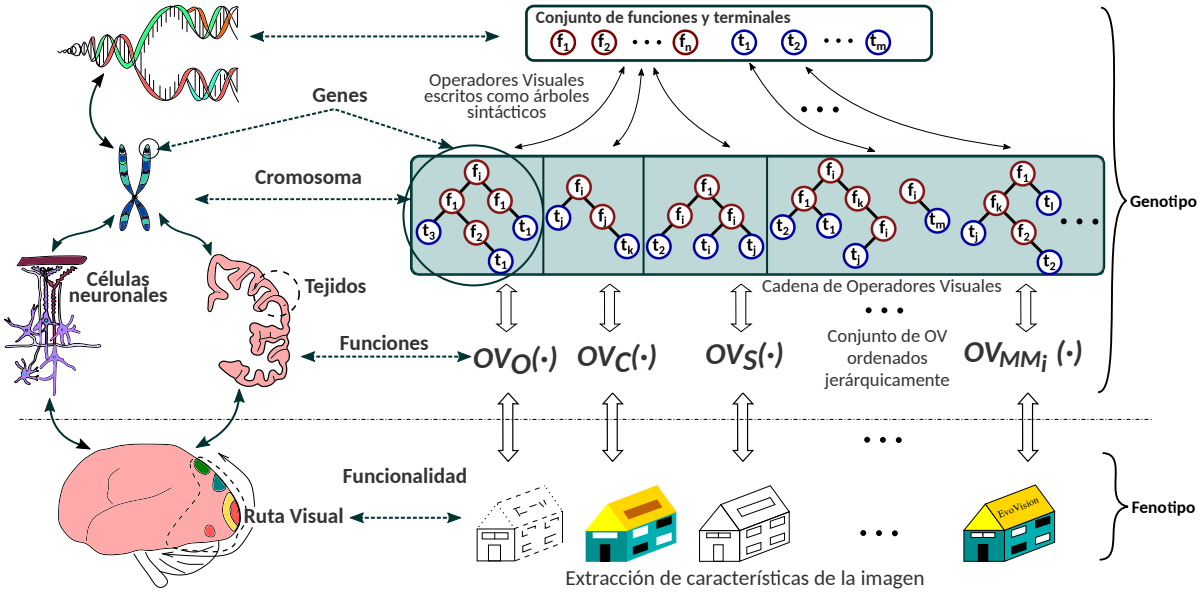

Figura 2. Analogía de la cortesa visual y sus componentes con la representación del individuo (genotipo)

La selección de estos conjuntos de funciones se hizo de forma empírica, son resultado de diversos trabajos realizados en el laboratorio de EvoVisión (Trujillo y Olague, 2008; Perez y Olague, 2009; Dozal *et al.*, 2014; Clemente *et al.*, 2015a; Hernández *et al.*, 2014).

La construcción de un individuo del modelo del AVC usando la representación descrita requiere de la optimización de varios operadores, por esto el espacio de todas las posibles soluciones para el modelo

Tabla 2. Funciones y terminales para el operador visual de orientación (VO_O).

	Expresión	Descripción
Funciones	$A+B$, $A-B$, $A\times B$, A/B	Funciones aritméticas entre dos terminales A y B
	$\|A\|$, $\|A+B\|$, $\|A-B\|$	Valor absoluto aplicado a la terminal A y a las operaciones de suma y resta
	$log(A)$	Logaritmo aplicado a la terminal A
	$(A)^2$	Función cuadrática aplicada a la terminal A
	$\sqrt{A}$	Obtiene la raíz cuadrada de la terminal A
	$k+A$, $k-A$, $k\times A$, A/k	Funciones aritméticas entre una terminal A y una constante k
	$round(A)$, $\lfloor A \rfloor$, $\lceil A \rceil$	Funciones de redondeo, piso y techo aplicadas a una terminal A
	$inf(A,B)$, $sup(A,B)$	Funciones ínfimo y supremo aplicadas a las terminales A y B
	$G_{\sigma=1}(A)$, $G_{\sigma=2}(A)$	Convolución de la terminal A con un filtro Gaussiano de $\sigma=1$ o 2
	$D_x(A)$, $D_y(A)$	Obtiene la derivada de la terminal A a lo largo de la dirección x y y
	$thr(A)$	Calcula un umbral dinámico de la terminal A
Terminales	I_r,I_g,I_b,I_c,I_m,I_y,I_k,I_h,I_s,I_v	Todos los elementos de I_{color} obtenidos a partir de la imagen de entrada I
	$D_x(I_x)$,$D_y(I_x)$,$D_{xx}(I_x)$, $D_{yy}(I_x)$,$D_{xy}(I_x)$	Derivadas de 1er y 2do orden aplicadas a cada banda de color I_x, donde $I_x \in I_{color}$
	$G_{\sigma=1}(I_x)$	Filtro Gaussiano de $\sigma=1$ aplicado a cada banda de color I_x, donde $I_x \in I_{color}$

Tabla 3. Funciones y terminales para el operador visual de color (OV_C).

	Expresión	Descripción
Funciones	$A+B$, $A-B$, $A\times B$, A/B	Funciones aritméticas entre dos terminales A y B
	$k+A$, $k-A$, $k\times A$, A/k	Funciones aritméticas entre una terminal A y una constante k
	$round(A)$, $\lfloor A \rfloor$, $\lceil A \rceil$	Funciones de redondeo, piso y techo aplicadas a una terminal A
	$log(A)$, $exp(A)$	Funciones trascendentales aplicada a la terminal A
	$(A)^2$	Función cuadrática aplicada a la terminal A
	$\sqrt{A}$	Obtiene la raíz cuadrada de la terminal A
	$(A)^c$	Determina el complemento de la terminal A
	$thr(A)$	Calcula un umbral dinámico de la terminal A
Terminales	$Op_{r-g}(I)$, $Op_{b-y}(I)$	Oponencias de color rojo - verde y azul - amarillo de la imagen de entrada I
	I_r,I_g,I_b	Bandas del modelo RGB (roja, azul y verde) de la imagen de entrada I
	I_c,I_m,I_y,I_k	Bandas del modelo CYMK (cian, magenta, amarilla e intensidad) de la imagen de entrada I
	I_h,I_s,I_v	Bandas del modelo HSV (matiz, saturación y brillo) de la imagen de entrada I

presentado, resulta de la combinación del espacio factible de cada uno de los operadores. Así, el espacio de búsqueda para el algoritmo se define como el número de posibles soluciones que se pueden lograr al combinar todos los posibles operadores visuales. Entonces, el tamaño del espacio de búsqueda se puede obtener analizando el conjunto de funciones y terminales descritos en las Tablas 2,3,4 y 5. En consecuencia, dada una estructura de árbol i, nT terminales y nF_U funciones unarias y nF_B funciones

Tabla 4. Funciones y terminales para el operador visual de forma (OV_F).

	Expresión	Descripción
Funciones	$A+B$, $A-B$, $A\times B$, A/B	Funciones aritméticas entre dos terminales A y B
	$k+A$, $k-A$, $k\times A$, A/k	Funciones aritméticas entre una terminal A y una constante k
	$round(A)$, $\lfloor A \rfloor$, $\lceil A \rceil$	Funciones de redondeo, piso y techo aplicadas a una terminal A
	$A \ominus SE_d$, $A \ominus SE_s$, $A \ominus SE_{dm}$	Función de erosión con elementos estructurales (SE), de disco, cuadro y diamante aplicadas a la terminal A
	$A \oplus SE_d$, $A \oplus SE_s$, $A \oplus SE_{dm}$	Función de dilatación con elementos estructurales (SE), de disco (d),cuadro y diamante aplicadas a la terminal A
	$Sk(A)$	Función *skeleton* aplicada a la terminal A
	$Perim(A)$	Función *perimeter* aplicada a la terminal A
	$A \circledast SE_d$, $A \circledast SE_s$, $A \circledast SE_{dm}$	Transformación Hit-or-miss con elementos estructurales (SE), disco, cuadro y diamante aplicada a la terminal A
	$T_{hat}(A)$, $B_{hat}(A)$	Filtro morfológico top-hat y bottom-hat aplicado a la terminal A
	$A \circledcirc SE_s$, $A \circledcirc SE_s$	Funciones morfológicas de apertura y cerradura aplicadas a la terminal A
	$thr(A)$	Calcula un umbral dinámico de la terminal A
Terminales	I_r,I_g,I_b	Bandas del modelo RGB (roja, azul y verde) de la imagen de entrada I
	I_c,I_m,I_y,I_k	Bandas del modelo CYMK (cian, magenta, amarilla e intensidad) de la imagen de entrada I
	I_h,I_s,I_v	Bandas del modelo HSV (matiz, saturación y brillo) de la imagen de entrada I

Tabla 5. Funciones y terminales para los operadores visuales de mapas mentales (OV_{MM}).

	Expresión	Descripción
Funciones	$A+B$, $A-B$, $A\times B$, A/B	Funciones aritméticas entre dos terminales A y B
	$\lvert A+B \rvert$, $\lvert A-B \rvert$	Valor absoluto aplicado a los operadores de adición y substracción
	$log(A)$	Función logaritmo aplicada a la terminal A
	$(A)^2$	Función cuadrática aplicada a la terminal A
	$\sqrt{A}$	Raíz cuadrática de A
	$G_{\sigma=1}(A)$, $G_{\sigma=2}(A)$	Convolución de la terminal A con un filtro Gaussiano de $\sigma = 1$ o 2
	$D_x(A)$, $D_y(A)$	Obtiene la derivada de la terminal A a lo largo de la dirección x y y
Terminales	MC_d, $D_x(MC_d)$,$D_y(MC_d)$, $D_{xx}(MC_d)$, $D_{yy}(MC_d)$, $D_{xy}(MC_d)$	Mapa conspicuo y sus derivadas de 1er y 2do orden, sobre cada dimensión

binarias; el número de posibles operadores visuales nOV_i se puede calcular a través de la relación:

$nVO_i = nT^{nln} \times nF_U^{npn_1} \times nF_B^{npn_2}$, donde, nln es el número de nodos hoja, npn_1 es el número de nodos padres con al menos un hijo, y npn_2 es el número de nodos padre con al menos dos nodos hijos. Por lo tanto, el tamaño del espacio de búsqueda S_s se puede calcular multiplicando la cantidad de posibles combinaciones de operadores visuales para tres dimensiones de características y los mapas

mentales.

2.3.2. La función objetivo

El siguiente paso consiste en la evaluación de los individuos que fueron generados en el paso anterior, esto se lleva acabo mediante la función objetivo. De la misma forma que en los algoritmos genéticos, se hace uso de la función objetivo, que mide la calidad de las soluciones para resolver el problema que se esta estudiando.

El Brain Programming ha sido aplicado para resolver diferentes problemáticas, es por ello que la función objetivo no siempre es la misma, en esta sección se dará un repaso acerca de las diferentes métricas de evaluación que han sido empleadas

A continuación se presentará la descripción de algunos conceptos que son generales y permiten comprender mejor estas métricas de evaluación. Para el problema de reconocimiento de objetos comúnmente se usan: la exactitud, las gráficas de precisión y cobertura, la precisión media, y las gráficas y curvas ROC. También se complementa la explicación con el estudio de la medida-F, la cual ha sido utilizada en problemas de detección y atención visual.

En primer lugar se considera el problema de clasificación binaria, éste es el caso que se ha abordado con mayor frecuencia por el Brain Programming.

Formalmente, cada imagen de un conjunto de prueba se mapea a una predicción, usando una etiqueta positiva o negativa que define la clase. Después, el modelo de clasificación, también conocido como clasificador que ya fue preentrenado, produce una predicción de cada uno de los elementos del conjunto sobre su pertenencia o no a una clase.

Tabla 6. Matriz de confusión

		Valor Real	
		Positivo	Negativo
Predicción	Positiva(P)	Verdaderos Positivos (VP)	Falsos Positivos (FP)
	Negativa(N)	Falsos Negativos (FN)	Verdaderos Negativos (VN)

Dado un clasificador, y un conjunto de prueba, se puede construir una matriz de tamaño 2x2 para representar la información del conjunto, para formar lo que se conoce como matriz de confusión, que se muestra en la Tabla 6. Como se puede observar, existen cuatro posibles salidas:

Verdadero positivo (VP): Este es el caso en donde la predicción del clasificador es Positiva(pertenece a la clase), y el elemento *sí* pertenece a la clase.

Verdadero Negativo (VN): El clasificador predice Negativo (no pertenece a la clase) y el elemento *no* pertnenece a la clase.

Falso positivo (FP): El clasificador predice Positivo, pero el elemento *no* pertenece a la clase.

Falso negativo (FN): El clasificador predice Negativo, pero el elemento *sí* pertenece a la clase.

De la matriz de confusión se derivan las ecuaciones que definen las diversas métricas para evaluar el rendimiento de los métodos de clasificación y reconocimiento de objetos. Los números de la diagonal mayor representan las clasificaciones correctas, y los numeros fuera de la diagonal representan los errores o confusiones de las predicciones entre las clases.

2.3.3. Selección

Después de que todos los individuos de la población han sido evaluados mediante la función objetivo, el siguiente paso consiste en producir la nueva generación. El primer paso es realizar un proceso de selección de los individuos que se convertirán en padres para propagar sus genes a la siguiente generación. Los individuos se obtienen usando un enfoque basado en la técnica *lexicographic parsimony pressure* (Luke y Panait, 2002); en donde se usa un operador de selección, conocido como *torneo* [10], este método fue modificado para considerar el tamaño del individuo. Para seleccionar un padre, dos individuos son seleccionados aleatoriamente, y se compara su aptitud. El individuo con mayor aptitud, tiene más probabilidad de ser seleccionado; si las aptitudes son iguales, entonces se compara el tamaño de los individuos, y se selecciona el de menor longitud. Si los dos individuos tienen el mismo tamaño se elige uno aleatoriamente. Para adaptar este enfoque al genotipo utilizado en el BP, el tamaño del árbol más grande en el individuo es tomado como el tamaño del individuo completo; es decir, si tienen la misma aptitud entonces seleccionamos al individuo más pequeño.

2.3.4. Mecanismos de variación y recombinación genética

Este paso comúnmente se realiza mediante los operadores de cruzamiento y mutación, que extraen información genética de padres que fueron seleccionados como se explica en la sección 2.3.3. El método

[10] del inglés: tournament

se inspira en sus homólogos naturales, el objetivo es la recombinación genética y el intercambio de información entre cromosomas para pasar a la siguiente generación. Durante la creación de la nueva población, se eligen los operadores genéticos que serán utilizados para la reproducción basándose en una probabilidad que es asignada siguiendo el esquema propuesto por Koza. En este caso, las operaciones son aplicadas independientemente y su probabilidad total suma uno.

Los individuos generados por el BP requieren de operadores de recombinación diferentes puesto que tienen una estructura diferente a la de la de programación genética convencional. Estas operaciones genéticas permiten la recombinación del material genético, mientras promueven la inovación genética de los individuos en todos los niveles, y mantienen la diversidad de la población. Se utilizan las operaciones de cruzamiento y mutación que se describen a continuación.

Cruzamiento a nivel cromosoma: En el cruzamiento a nivel cromosoma, los puntos de cruzamiento son seleccionados a nivel del genotipo y son iguales en ambos padres, las partes correspondientes son intercambiadas para producir dos nuevos descendientes; ver figura 3.

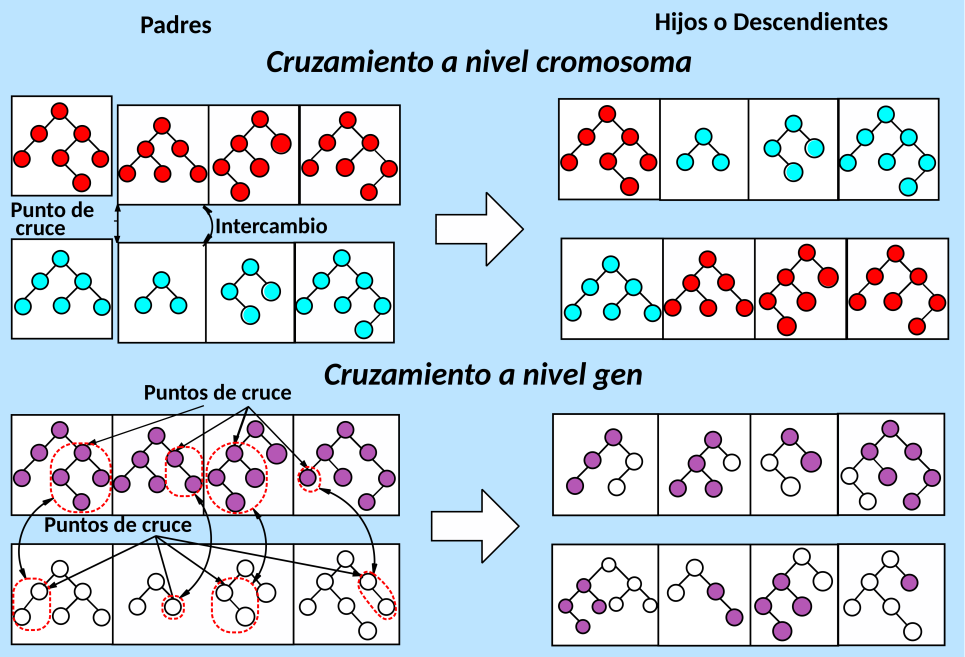

Figura 3. Operador de cruzamiento

Cruzamiento a nivel gen: Para realizar el cruzamiento se requieren dos padres. Se elige un punto de cruce aleatoriamente en cada padre y en sus árboles respectivos; después, los sub-árboles seleccionados se intercambian entre los padres para generar dos nuevos operadores. Este proceso genera dos nuevos cromosomas hijos. Ver figura 3.

Mutación a nivel cromosoma: El procedimiento es análogo de la mutación cromosómica natural y consiste en la sustitución de un operador completo elegido entre los operadores que construyen el cromosoma por un operador generado de manera aleatoria.

Mutación a nivel gen: Se le conoce como mutación genética al fenómeno natural donde se alteran las cadenas del ADN que forman al gen. Este procedimiento es emulado de manera computacional, solo se altera un segmento de la información de cada operador. En otras palabras, a cada operador del cromosoma se le elige un nodo de manera aleatoria y el sub-árbol resultante debajo de este punto de mutación, se sustituye por uno nuevo generado aleatoriamente. Esto se lleva a cabo en cada uno de los árboles que componen al genotipo, y la estrategia se repite para producir un nuevo individuo y eventualmente un nueva población. Ver figura 4.

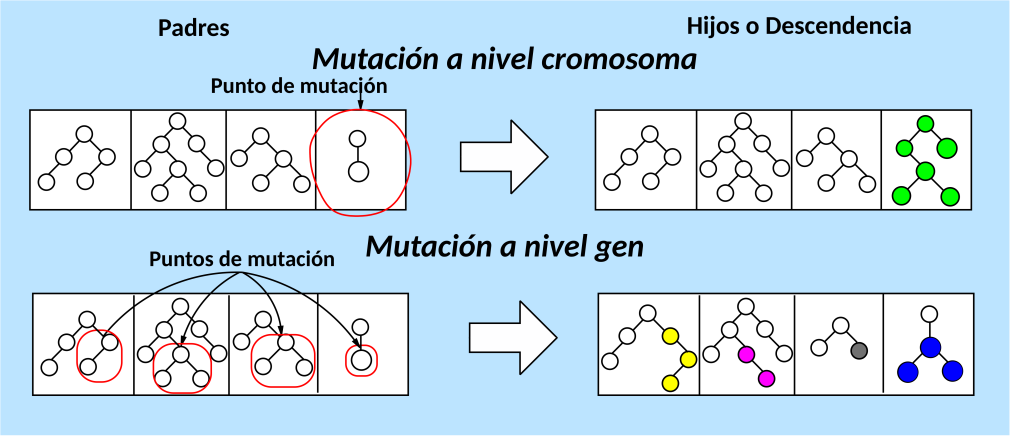

Figura 4. Operadores genéticos de mutación

Es importante mencionar que al aplicar cualquiera de los operadores genéticos, se mantiene la congruencia entre las dimensiones del individuo. Por ejemplo para el cruzamiento a nivel cromosoma, el punto de cruce es el mismo en ambos padres, ésto hace que los individuos sigan teniendo el mismo tipo de operaciones y mantengan la misma estructura.

Los árboles que se modifican en la mutación a nivel cromosoma estan formados por las ecuaciones correspondientes a la parte del individuo que se esta modificando; es decir, si el punto de mutación es en la dimensión de forma, el nuevo árbol esta compuesto por ecuaciones correspondientes a la dimensión de forma.

2.3.5. Condición de paro

El criterio utilizado para concluir un ciclo evolutivo es cuando se alcanza un máximo numero de iteraciones, frecuentemente definido en el Brain Programming en treinta.

2.4. Trabajos relacionados

Dada la capacidad de esta técnica y las diferentes aplicaciones que puede tener, se han desarrollado investigaciones que resuelven diferentes tareas del área de visión por computadora. En la literatura, el Brain programming se ha aplicado tomando ambos enfoques descritos en la sección 2.2. Se han desarrollado trabajos en donde se optimizan programas para realizar la atención visual basados en la ruta dorsal artificial y otros para reconocimiento de objetos, es decir, se han abordado y simulado los dos flujos principales de información que realiza la corteza visual

Los primeros trabajos donde se introdujo el término Brain Programming fueron los realizados por Olague *et al.* (2014a) y Dozal *et al.* (2014) sin embargo, estas investigaciones son fruto de otros trabajos previos en donde se definieron las funciones que mejor extraían características de cada una de las dimensiones empleadas en esta metodología.

Una característica fundamental del Brain Programming es que las soluciones encontradas pueden ser interpretadas. Uno de los trabajos donde esto queda en evidencia es en el estudio realizado por Dozal *et al.* (2014). En este trabajo lo ponen a prueba con tres bases de datos que cuentan cada una de un objeto diferente: una lata roja, una señal de advertencia en forma de triángulo y señales de tránsito. También descubrieron algunas propiedades interesantes en los operadores visuales, notaron que EVO están altamente especializados en la extracción y combinación de las principales características del objeto de interés. Por ejemplo, para el problema de la atención de la lata roja, los EFI se centraron en la aplicación de la orientación y las dimensiones del color, siendo el color la característica más destacada para el problema de la lata roja. Mientras tanto, el EVO_O utiliza la banda de color magenta y la función Dx. Esto es comprensible ya que el magenta está más cerca en el espacio de color en comparación con el color rojo"de la lata, mientras que Dx mejora la apariencia de su posición vertical.

Clemente *et al.* (2015b) utiliza el modelo de la Corteza Visual Artificial (AVC), agregando una nueva metodología para la descripción de la imagen, que permite la detección y descripción de un objeto en la escena. Utiliza un enfoque multi-objetivo para crear un descriptor usando la información de la región de la imagen donde se encuentra el objeto, encontrando implícitamente su ubicación. En este estudio

implementan esta metodología para clasificar el conjunto de personas de la base de datos de imágenes GRAZ-02

En 2016, Hernández *et al.* (2016) extiende el problema multi-objetivo, ahora en imágenes naturales utilizando las bases de datos de GRAZ-01 y GRAZ-02, que continenen objetos en escenarios reales, es decir, el objeto puede estar en cualquier parte de la imagen y no necesariamente en primer plano, ni ocupando la mayor parte de la pantalla. Las soluciones encontradas por esta investigación tienen puntos de contactos y en repetidas ocasiones superan las expectativas en comparación con otras técnicas de vanguardia para clasificar las bases de datos antes mencionadas. Ver Figura 5

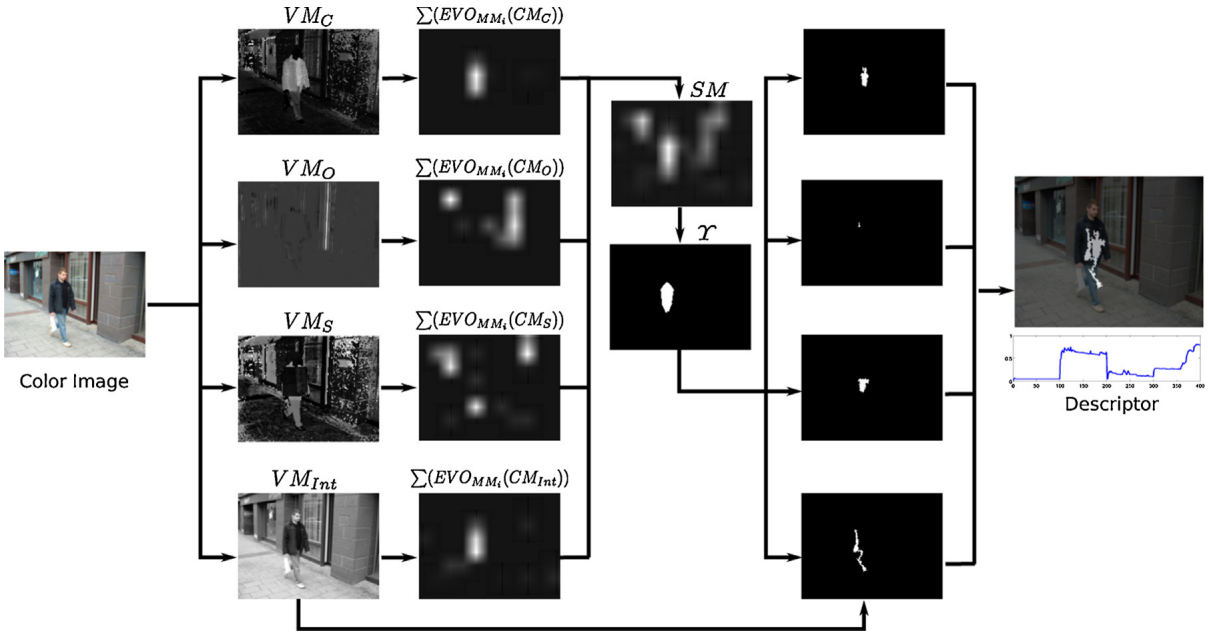

Figura 5. Reconocimiento de objetos en una escena natural, Hernández *et al.* (2016)

Como se mencionó al inicio de este capítulo, este enfoque considera que el proceso de extracción y descripción de la información visual puede ser emulado por la composición de funciones mediante un conjunto de operaciones matemáticas, pero en Olague *et al.* (2017) descubrieron que el Brain Programming encontraba buenas soluciones en las primeras iteraciones del algoritmo, y realizaron un estudio para medir la frecuencia y calidad de soluciones encontradas mediante una búsqueda aleatoria. Estudio que posteriormente extendieron en Olague *et al.* (2014a) donde descubrieron que el número total de operaciones computacionales utilizadas para clasificar una imagen era relativamente bajo, en comparación con estraterigas similares que están basadas en la aplicación de parches de imágenes.

En el artículo Olague *et al.* (2018), proponen una metodología para abordar el problema de segui-

miento de cabezas [11], mediante la optimización de un modelo para la ruta dorsal. Este problema ha sido estudiado desde hace mucho tiempo y posee muchas apliaciones prácticas que van desde la vigilancia y seguridad, control de tráfico, hasta interacciones humano-computadora y reconocimiento de actividades. Específicamente en este trabajo, los autores abordan el problema de seguir la cabeza de un individuo en una secuencia de video. La solución encontrada es capaz de manejar eficazmente dificultades como la oclusión, la distracción y la iluminación. Una idea presentada en este trabajo en la capacidad de controlar automáticamente la cámara para continuar con el rastreo de la cabeza de la persona en cuestión. En este trabajo demuestran que el método propuesto supera a varios métodos del estado de la técnica en el desafiante problema del seguimiento de cabezas. Para más ver la Figura 6

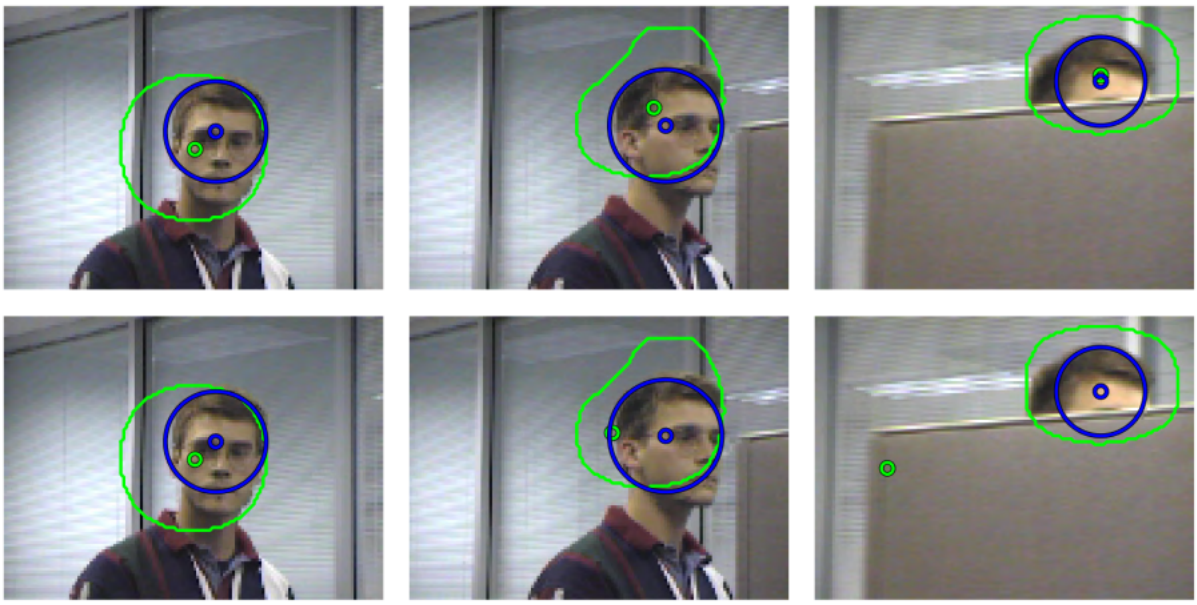

Figura 6. Seguimiento automático de una cabeza, Olague *et al.* (2018)

El BP al ser un modelo basado en programación genética, que esta abordando problemáticas más complejas, también a veces sufre los problemas relacionados con los algoritmos evolutivos, como quedarse en un mínimo local. En Olague y Chan-Ley (2020) los autores proponen una metodología para encontrar mejores soluciones a la que dan por nombre "Hands-on" en la que, después de realizar algunas ejecuciones del programa, es posible tomar las mejores soluciones de cada ejecución y colocarlas en un nuevo conjunto. Este nuevo conjunto de soluciones es utilizado como generación inicial en un nuevo ciclo evolutivo, de esta forma, las soluciones se enriquecen beneficiándose del conocimiento adquirido de otras ejecuciones del programa. Con esta metodología lograron incrementar el score de clasificación obtenido hasta en 6 puntos porcentuales.

[11] Del inglés: Head tracking

Para este punto, el BP ya había demostrado su enorme capacidad de resolución de problemas de clasificación de objetos. En Chan-Ley y Olague (2020), los autores abordaron un problema más complejo, la categorización de imágenes de obras de arte, desde el punto de vista de los materiales y técnicas utilizados por los artistas.

El estudio fue realizado utilizando las bases de datos WikiArt y Kaggle; las categorías que comprenden la base de datos son: dibujos, grabados, pinturas, iconografías y esculturas. Es necesario mencionar que este problema es especialmente difícil debido que los ejemplos dentro de una misma clase varían notablemente entre ellos como se puede observar en la figura 17. Por ejemplo en la categoría dibujos, puede existir cualquier cosa, ya sea el dibujo de una mano, una cara, un animal o paisaje.

Una vez mas, se demuestra la capacidad innegable de este paradigma debido a que se alcanzan resultados meritorios. Estos confirman que la programación cerebral coincide o supera al aprendizaje profundo en tres de las cinco clases estudiadas(más del 90 %) mientras que está cerca (menos del 5 %) en las dos restantes. A diferencia de las redes neuronales, los modelos generados por el BP son significativamente más simples.

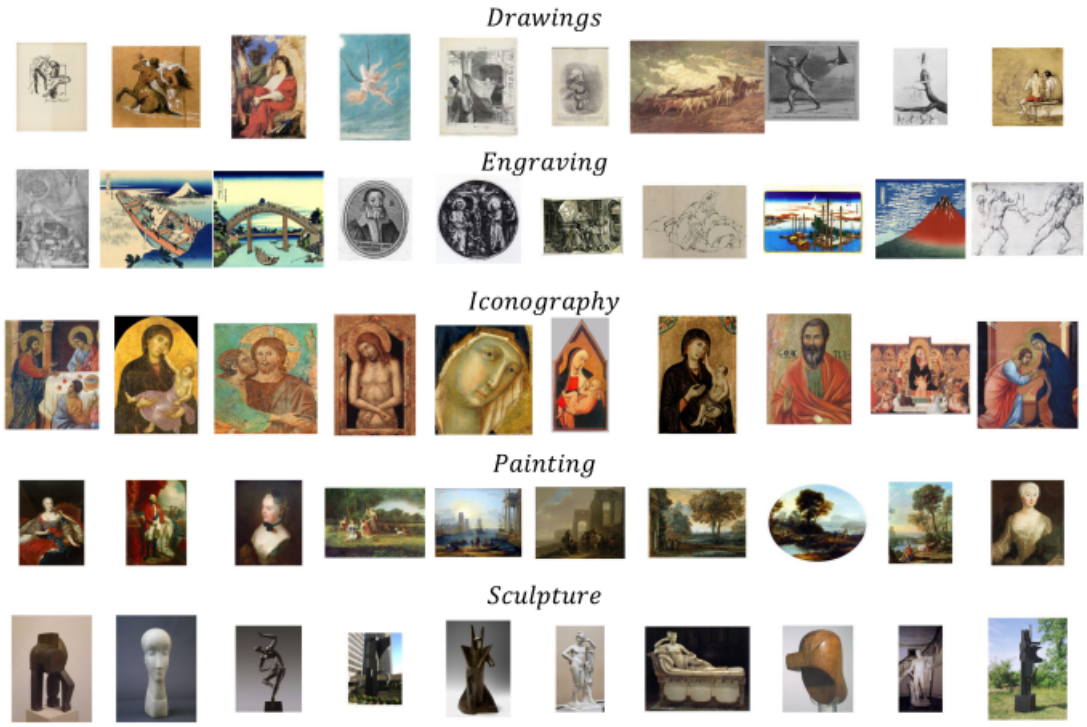

Figura 7. Base de datos de prueba utilizada en Chan-Ley y Olague (2020)

Las funciones que realiza el cerebro no se limitan a una sola tarea, como se ha venido mmencionando, la ruta dorsal tambien influye en algunas tareas de movimiento, como lo son posicionar el ojo al foco de

atención, o reaccionar ante un estimulo percibido. Entonces las funciones cerebrales estan integradas, es por ello que se puede hacer alguna integración tambien en la analogia computacional. Por su parte, Guerra(2015) presena un modelo donde añade la idea de potencialidades al modelo que evoluciona.

Por su parte Menendez(2021) añadió la dimensión de la textura al modelo computacional, sin embargo no consiguió superar el desempeño del modelo del Brain Programming como está, pero valdria la pena hacer una revisión del conjunto de funciones y terminales que seleccionó para su modelo y reacondicionarlo para ver si se pueden obtener mejores resultados.

Para resolver el problema de la atención de Abajo hacia arriba, en la que el programa localiza en la imagen el objeto que sea más sobresaliente en la escena. Muchas veces para este tipo de tareeas, las bases de datos son diseñadas de forma que La base de datos empleada para este trabajo, consiste en imágenes de diferentes objetos, y el ground truth, es decir el objeto que se considera sobresaliente, se proporciona en la forma de una máscara binaria en donde las imágenes tienen el valor de 1 en todo el objeto sobresaliente y 0 en lo que se considera Background. Este trabajo abrió una nueva linea de investigación para el Brain PProgramming ya que obtuvo un

2.5. Áreas de oportunidad

Como se mencionó anteriormente, casi todos los problemas que se han abordado son tratando de resolver el problema de clasificación binaria. Lo que representa una desventaja fundamental cuando se esta trabajando con respecto a los retos actuales de clasificación, que cuentan con bases de datos de más de 1000 clases.

2.6. Conclusión

En este capítulo se hizo un estudio completo del Brain Programming, desde los primeros trabajos en el 2014 hasta los últimos generados en el 2021. Después de haber hecho este estudio exhaustivo, podemos concluir que la metodología presentada es bastante versátil y competitiva con respecto a las tecnologías del estado del arte. Valdría la pena profundizar más en el estudio para poder explotar las ventajas que puede aportar en las diversas áreas de visión por computadora

Capítulo 3. El problema de clasificación de obras de arte

El creciente desarrollo de las tecnologías ha significado que casi todos los aspectos de nuestras vidas estén digitalizados. El caso del arte no es una excepción, casi todas las obras de arte actuales, cuentan con su imagen digitalizada, ya sea por una simple fotografía o mediante un proceso minucioso en un museo. Con los crecientes volúmenes de bases de datos de arte en Internet, surge la abrumadora tarea de organización de las obras de arte digitalizadas

La categorización de objetos o el reconocimiento de objetos genéricos busca modelar y reconocer objetos basándose en su forma prototípica burda, pero cuando hablamos de obras de arte, la tarea se vuelve mucho más compleja (Dickinson *et al.*, 2009). Al examinar una obra de arte, un experto puede realizar su clasificación con respecto a diferentes características. Algunas de las más empleadas son con respecto al estilo, género o medio con el que fueron elaboradas.

El *estilo* se refiere a los elementos visuales, técnicas y métodos distintivos que componen la obra. Suele corresponder a un movimiento artístico o a una escuela (grupo) a la que pertenece el autor. Reconocer el estilo artístico no está necesariamente correlacionado con lo que aparece en la imagen, es decir, no se refiere a la correspondencia de los objetos específicos existentes en la obra de arte. El estilo se relaciona principalmente con la forma y se puede asociar con características en diferentes niveles.

El *género* divide las obras de arte de acuerdo con los temas y objetos representados. La jerarquía clásica de géneros se desarrolló en la cultura europea en el siglo XVII. Dividiendo los géneros en alto (pintura de historia y retrato) y bajo (pintura de género, paisajismo y bodegón). Esta jerarquía se basó en la noción del hombre como medida de todas las cosas. El paisaje y el bodegón, también llamado naturaleza muerta, fueron los más bajos porque no involucraban temas humanos. La historia obtuvo una clasificación más alta porque trata de los eventos más nobles de la humanidad. El sistema de géneros no es tan relevante para un contemporáneo.

Con respecto a estos dos aspectos (estilo y género), las personas que estudian el problema de clasificación de obras de arte, utilizan el estilo para referirse a ambos términos. Se han propuesto varios enfoques que intentan resolver el problema de la clasificación de obras de arte.

El reconocimiento de imágenes de arte se diferencia de la tarea del reconocimiento de objetos, ya que dos pinturas pueden describir la misma escena utilizando técnicas artísticas muy diferentes. Los resultados muestran que todavía hay un margen de mejora considerable.

Saleh y colaboradores. (Saleh *et al.*, 2016) se centran en clasificar de acuerdo a la influencia artística. La base de datos que proponen contiene un total de 1710 imágenes de obras de arte de 66 artistas elegidos de la base de datos Artchive:Mark Harden [1].

Los artistas y curadores utilizan diferentes conceptos para describir obras de arte. Algunas de ellas son características básicas como el color, la textura y la forma, pero también elementos más abstractos como el movimiento, la armonía, el equilibrio, la proporción y los patrones. Incluso algunos atributos físicos, como pinceladas u objetos en la escena, se utilizan para categorizar imágenes artísticas. Aprender y juzgar conceptos visuales tan complejos es una capacidad impresionante de percepción humana (Saleh y Elgammal, 2015).

3.1. Antecedentes

Se ha abordado el problema de clasificación de imágenes de arte desde tres perspectivas: 1) extracción manual de características, 2) redes neuronales convolucionales profundas y 3) metodologías basadas en programación genética.

3.1.1. Extracción manual de características

La extracción manual de características fue el primer y único método empleado durante mucho tiempo, la manera en la que se hacía era desarrollando fórmulas que puedan extraer características particulares de las imágenes y obtener una representación que permita clasificar una imagen de manera efectiva.

Una de las primeras obras que emplean la extracción manual de características fue Keren (2002). En ese trabajo, los autores propusieron un esquema de coeficientes de la transformada de coseno discreto (DCT [2]) extraer características que permitan identificar al autor de la obra. Para este propósito, construyeron una base de datos personalizada de aproximadamente 300 imágenes en escala de grises de cinco pintores (Rembrandt, Van-Gogh, Picasso, Magritte y Dali).

Li y Wang (Li y Wang, 2004) por su parte, propusieron usar un modelo de Markov bidimensional para analizar las pinceladas hechas por el autor y proporcionar información para distinguir a los artistas de pinturas chinas antiguas. Su base de datos consta de 276 imágenes en escala de grises de cinco artistas

[1] http://www.artchive.com/cdrom.htm
[2] por sus siglas en inglés: Discrete cosene transform

chinos con una resolución de 3000×2000 pixeles, pero escalados a 512 en la sección más corta de la imagen, manteniendo la relación de aspecto.

Los autores en Arora y Elgammal (2012) presentan un estudio comparativo de diferentes metodologías de clasificación basadas en características extraídas manualmente. Compararon características de nivel semántico con SVM, SIFT de color y SIFT con BoV, y la asignación latente de Dirichlet con un modelo de BoV generativo para la clasificación de género. En su estudio, se utilizó una base de datos de siete categorías de pinturas (Abstracto, Barroco, Renacimiento, Popart, Expresionismo, Impresionismo y Cubismo) del conjunto de datos Artchive utilizando 70 imágenes de cada clase.

Rosado (Rosado, 2019) empleó un BoV utilizando un método dense-SIFT para la extracción de características y Análisis Semántico Probabilístico Latente (PLSA) para realizar un análisis de 434 imágenes digitalizadas de pinturas, dibujos, libros y grabados de Antoni Tàpies. En general, observamos que el uso de características diseñadas a mano hace posible obtener resultados alentadores, pero no perfectos. Con el tiempo, la complejidad de estas características comenzó a ser más difícil de diseñar. Además del proceso de diseño de características, el desarrollo del algoritmo de aprendizaje fue un área de investigación completamente independiente necesaria para coincidir con la extracción de características.

3.1.2. Redes neuronales convolucionales

Las DCNN han presentado un gran avance en muchas áreas del procesamiento de imágenes, y trabajos recientes en el problema de clasificación de imágenes de arte. Los autores de Karayev *et al.* (2014), introdujeron el uso de un modelo DCNN entrenado para el reconocimiento de objetos para reconocer el estilo. Estas características logran un alto rendimiento en la identificación del estilo y superan a la mayoría de las funciones de extraídas manualmente. Bar et al. (Bar *et al.*, 2014) propusieron una representación binaria compacta que combina los descriptores PiCoDes y las funciones de activación convolucional profunda de un modelo DCNN para identificar estilos artísticos en pinturas que muestran resultados excepcionales para clasificar imágenes de obras de arte de WikiArt usando 27 clases.

Noord et al.(van Noord *et al.*, 2015) emplearon una adaptación de AlexNet para clasificar los estilos de obras de arte de las imágenes del Museo Rijks. Podrían visualizar las regiones con un mapa de calor de la obra de arte que afecta la predicción del estilo. Cetinic y Grgic Cetinic y Grgic (2016) utilizaron las funciones extraídas de VGG para clasificar las imágenes de la base de datos de WikiArt en siete clases de género, como retrato, paisaje, ciudad, naturaleza muerta, desnudo, flor y animal. Superan las funciones con características extraídas manualmente como SIFT, descriptor esencial, HOG, matriz de

concurrencia de nivel de gris (GLCM) e histogramas de color HSV con su método de clasificación. Seguín et al. Seguin *et al.* (2016) propone extraer de VGG componentes similares compartidos por varias obras de arte denominadas enlace visual. Estos vínculos intentan encontrar similitud a partir de pinturas de los mismos creadores o de las mismas escuelas. El experimento utilizó imágenes de la base de datos de la Web Gallery of Art que informan que su método logra un mejor rendimiento que las funciones de características extraídas manualmente como SIFT.

Sun et al.(Sun *et al.*, 2017) emplearon AlexNet y VGG para construir una estructura con dos rutas que extrae características de objetos y texturas. La DCNN realiza el cálculo del objeto y la ruta de la textura utiliza las matrices de Gram. Los autores utilizaron en sus experimentos las bases de datos WikiPaintings, Flickr Style y AVA Style. Elgammal et al. Elgammal *et al.* (2018a) por su parte, propusieron un análisis de trazos en dibujos lineales usando una base de datos de 300 dibujos digitalizados con más de 80 mil trazos. Emplean funciones extraídas manualmente, funciones de aprendizaje profundo y la combinación de ambas para discriminar entre artistas a nivel de trazo con alta precisión. Además, su trabajo sirve para descubrir falsificaciones realizadas por artistas. Por su parte, Cetinic *et al.* (2018) un extenso experimento con CNN utilizando cinco modelos Caffe (CaffeNet, red Hybrid-CNN, red MemNet, red Sentiment y red Flickr) para cinco tareas diferentes de clasificación relacionadas con el arte (artista, género, estilo, período y asociación con un contexto artístico nacional específico) en tres grandes conjuntos de datos (WikiArt, Web Gallery of Art y TICC Printmaking Dataset). En Yang y Min (2020), los autores emplearon modelos DCNN previos al entrenamiento (AlexNet, VGG, GoogLeNet, ResNet, DenseNet) para reconocer los el medio de las obras de arte. Recolectaron alrededor de 1000 imágenes por clase (pincel de pintura al óleo, pastel, lápiz y acuarela) a través de varios motores de búsqueda y sitios web para clasificarlas. Obtuvieron resultados comparables con los de personas que recibieron entrenamiento previo.

3.1.3. Programación genética

Kowaliw y sus colaboradores propusieron la programación genética (GP) para descubrir una colección útil de características para la descripción de los estilos de artistas individuales (Kowaliw *et al.*, 2010). Presentan una base de datos de historietas y novelas gráficas. Es una colección de dibujos de paneles en escala de grises (imágenes de 200 x 200 píxeles o menos) organizados de acuerdo al artista que los creó y enfatizando el estilo estético. Hay 150 imágenes de entrenamiento y al menos 80 imágenes de validación para cada clase, con 240 imágenes de validación en el grupo de control. Obtuvieron resultados de alta precisión (por encima del 90 %) con respecto a los artistas en la base de datos, así como seleccionados al azar de los resultados del motor de búsqueda. Algunos inconvenientes de esta propuesta es que no

realiza una comparación significativa con bases de datos estándar con más imágenes y contenido de información significativamente más complejo.

Finalmente, una metodología similar a GP llamada Brain Programming obtuvo resultados competitivos en comparación con un modelo DCNN para la tarea de clasificación de imágenes de arte de acuerdo al medio Chan-Ley y Olague (2020). Esta técnica tiene como objetivo emular el comportamiento del cerebro basado en procesos de aprendizaje de la neurociencia con un nuevo aprendizaje simbólico a través de la programación genética. En los experimentos, Chan-Ley y Olague utilizan dos bases de datos de obras de arte de alta resolución de renombre (Kaggle y WikiArt) para clasificar cinco clases de medios artísticos (dibujos, grabado, pintura, iconografía y escultura). La técnica propuesta logra resultados comparables a AlexNet en un problema de clasificación binaria.

En resumen, podemos observar de los trabajos revisados que las soluciones al problema de clasificación han evolucionado en sistemas más complejos, utilizando cada vez más imágenes, con métodos basados en aprendizaje profundo, pero al ser un área tan vasta aún no existen estándares para la clasificación automática.

Capítulo 4. Evolución piramidal

En la computación evolutiva, la definición de espacio de búsqueda se basa en la representación de las soluciones y un conjunto de transformaciones sobre ellas para resolver un problema . En particular, para problemas de big data, la descripción del espacio de búsqueda requiere no solo la representación de características de las soluciones, sino una estrategia para administrar la base de datos. En este capítulo se plantea una propuesta en la que, usamos el paradigma de la programación cerebral con un nuevo enfoque que nos permitirá orientar la búsqueda mediante una modificación al ciclo evolutivo.

El esquema tradicional del Brain programming sigue el algoritmo evolutivo clásico, donde las estructuras que representan a las soluciones (cerebros artificiales), evolucionan mientras se evalúan contra una base de datos hasta descubrir la mejor solución o alcanzar un número determinado de iteraciones. Un esquema de este tipo consume una gran cantidad de recursos y en algunos de los problemas estudiados con el Brain Programming se necesitaron hasta dos meses para encontrar una solución (Olague y Chan-Ley, 2020). En esta sección se presenta una nueva estrategia en donde la base de datos y las estructuras computacionales generadas, son administradas a través de una representación piramidal que reduce el tiempo de ejecución sin sacrificar la precisión habitual del Brain Programming, mejorando así la exploración y explotación del espacio de búsqueda.

4.1. Inspiración

Como se mencionó en el Capítulo 2, el Brain Programming consta de dos partes principales, una estructura jerárquica bioinspirada y un ciclo evolutivo. Desde el momento de su creación hasta ahora, los cambios y actualizaciones que ha tenido el Brain programming han sido enfocados en el modelo jerárquico del cerebro; ya sea agregando o reduciendo dimensiones, actualizando la lista de funciones de acuerdo a la aplicación estudiada. En esta sección se presenta una propuesta en la que los cambios serán hechos en el ciclo evolutivo.

La idea consiste en ordenar los datos de entrenamiento y los individuos en una estructura piramidal, en el que aumentamos la presión selectiva conforme los individuos suben a los niveles más altos de la pirámide, haciendo que las mejores soluciones sean descubiertas de una manera más rápida. Éste nuevo modelo podemos observarlo en la Figura 8, pero para poner en contexto, en esta sección hablaremos de la inspiración que nos condujo a esta propuesta.

Olague *et al.* (2017) descubrieron que el Brain Programming es capaz de encontrar soluciones buenas con el tiraje aleatorio inicial del ciclo evolutivo, sin la necesidad de evolucionar. Algunas de esos individuos incluso obtenían un porcentaje de acierto del 100 % cuando se evalúan con conjuntos pequeños de imágenes. Sin embargo, algunas de esas soluciones funcionaban únicamente para el conjunto que se les estaba presentando, ya que al evaluarlas con otras imágenes o con conjuntos más grandes, el desempeño disminuía. Pero lo importante de este trabajo, es que algunas de esas soluciones si mantenían su desempeño. Es por eso que en el primer nivel de la pirámide tenemos conjuntos pequeños de imágenes. Para corroborar si encontramos una solución óptima, en los niveles superiores, los individuos son evaluados con conjuntos más grandes de imagenes.

Otro trabajo que sirvió como referencia es el descrito en la sección 5.5, en donde iniciamos un ciclo evolutivo con individuos que ya habían resultado ser buenos. Esto nos llevó a una idea que consiste en retroalimentar los individuos que han resultado ser buenos, a los niveles inferiores de la pirámide, para que aporten su experiencia en la forma de material genético. Esta idea se explica con mayor detalle en la sección 4.3.4

4.2. Planteamiento

La propuesta es un algoritmo que aplica la forma triangular de una pirámide para crear una estructura computacional dividida en cajas. Dentro de cada caja podemos encontrar dos tipos de datos, un conjunto de programas (AVCs) y un subconjunto de imágenes que definen el problema de clasificación.

La idea principal es que los individuos sean evaluados en cada nivel de la pirámide usando un subconjunto de imágenes, los mejores individuos son seleccionados para subir al siguiente nivel, en donde serán evaluados con un conjunto más grande de imágenes, haciendo que el problema sea más complejo. Las mejores soluciones son aquellas que conservan un buen desempeño aunque le aumentemos el número de imágenes

La estructura piramidal sirve para crear presión selectiva ya que los mejores individuos ascienden de nivel, en donde el número de imágenes en cada caja aumenta mientras que el número total de cajas disminuye, así como el número total de posibles soluciones. La pirámide promueve la interacción entre los elementos de la péntupla de aprendizaje. A continuación, revisaremos los pasos principales del algoritmo propuesto.

La estructura de este modelo consta de una pirámide de n niveles, en este trabajo definimos $l = 5$

niveles, en cada nivel colocamos un conjunto de cajas, como se puede ver en la figura 8. Para el primer nivel (nivel de entrada), hay cinco cajas; para el segundo hay cuatro cajas, en el tercer nivel, tenemos tres cajas; en el cuarto nivel hay dos, y para la capa de salida de la pirámide solo hay una caja. De esta forma podemos definir el vector $B = [b_1, ..., b_l]$ que nos representa el número de cajas por cada nivel de la pirámide.

En cada caja, asignamos dos tipos de información, imágenes de la base de datos y un conjunto de individuos (AVCs). En cuanto a las imágenes, en cada nivel se asigna un porcentaje de la base de datos en cada caja. Si bien el porcentaje es el mismo para cada contenedor en cada nivel, cada subconjunto es diferente de una caja a otra. Definimos el vector $D = [d_1, ..., d_l]$ que indica el porcentaje de la base de datos que habrá en cada nivel de la pirámide. La idea consiste en que en el nivel más alto de la pirámide se evalúe la base de datos completa. En el penúltimo nivel la mitad de la base de datos y así sucesivamente reduciendo el porcentaje de la base de datos hasta llegar al nivel más bajo de la pirámide (que corresponde al nivel de entrada).

Siguiendo esta propuesta, y considerando que el modelo piramidal que usamos tiene 5 niveles, los porcentajes quedan de la siguiente forma: $D = [6.2, 12.5, 25, 50, 100]$. En otras palabras, para el primer nivel, encontramos 6.2 % de imágenes, en el segundo nivel tenemos el 12.5 %, en el tercero podemos encontrar conjuntos del 25 % del número total de imágenes, y así sucesivamente. Es importante mencionar que cada caja contiene imágenes diferentes ya que tomamos muestras de las fotos del conjunto de datos de entrenamiento original sin reemplazo.

Además del conjunto de imágenes, asignamos a cada caja una colección de individuos (programas) para su evaluación con el correspondiente subconjunto de imágenes. En el esquema tradicional del Brain Programming, se ha demostrado que con 30 individuos por generación es suficiente para resolver el problema de clasificación (Olague, 2016; Clemente *et al.*, 2015a; Olague *et al.*, 2014b).

Como tenemos cinco cajas en el primer nivel, el número de individuos por caja para el primer nivel es seis, de esta forma obtenemos los 30 individuos de inicialización. Conforme se avanza en el modelo, el número de individuos por caja se reduce en uno por cada nivel hasta obtener 4 en el tercer nivel. Es decir, definimos el vector $N = [n_1, ..., n_l]$, en nuestro caso particular $N = [6, 5, 4, 4, 4]$. Decidimos no reducir más los individuos en los niveles superiores para tener la posibilidad de encontrar diferentes soluciones.

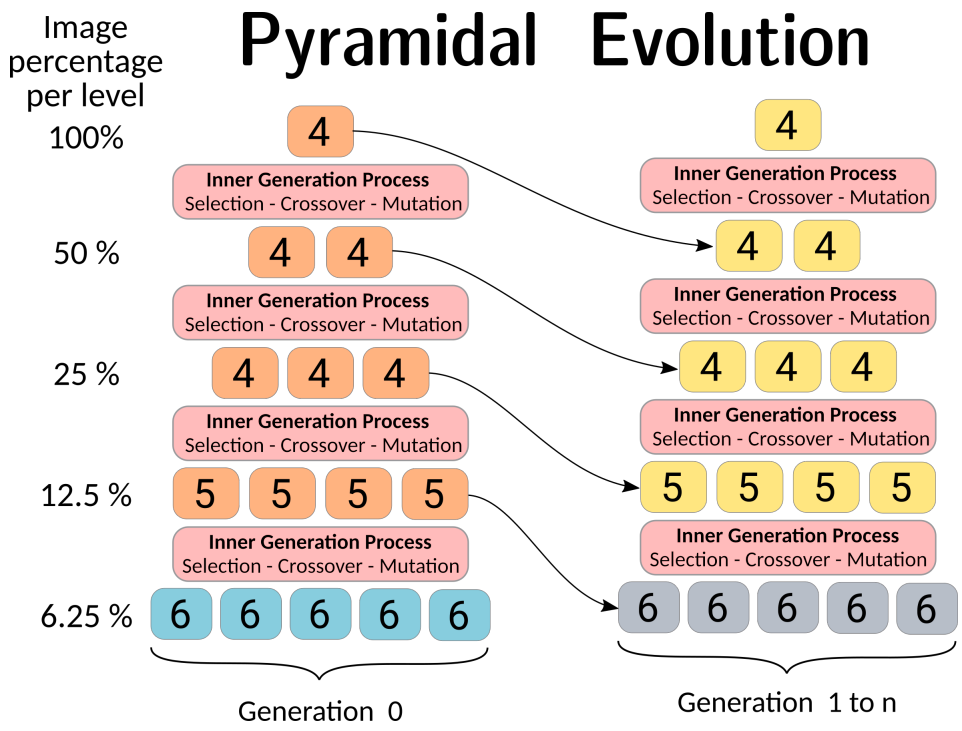

Figura 8. Modelo de evolución piramidal propuesto, donde el número de cajas se representa con B = [5,4,3,2,1], el porcentaje de la base de datos D= [6.2,12.5,25,50,100] y el número de individuos N = [6,5,4,4,4]. El número que vemos dentro de cada caja, representa el número de individuos asignados a ella.

En resumen, podemos parametrizar una pirámide usando un conjunto de tres vectores: $B = [_1, ..., b_l]$, $N = [n_1, ..., n_l]$ y $D = [d_1, ..., d_l]$. Donde el subíndice en cada vector se refiere al nivel l. B se refiere a la cantidad de cajas en cada nivel, N es la cantidad de individuos asignados a cada caja en cada nivel y D contiene la proporción de la base de datos evaluada en cada caja en cada nivel.

4.3. Procedimiento

4.3.1. Inicialización

El proceso comienza con la división de la base de datos, en cada uno de los diferentes niveles de la pirámide. Mientras ascendemos de nivel, aumentamos el número de imágenes utilizadas para entrenamiento / validación en cada nivel. Para el nivel de entrada, del 6.2 % de fotos asignadas a cada cuadro, dividimos el subconjunto en 70 % para entrenamiento y el restante para validación. Al avanzar a la segunda capa, el algoritmo usa 12.5 % para entrenamiento y validación, luego 25 % en el tercer nivel, 50 % para el cuarto nivel y finalmente, el algoritmo utiliza todas las imágenes en el último nivel. Mantenemos la proporción del 70 % para entrenamiento con el 30 % restante como la validación establecida en todos los niveles de la pirámide. Necesitamos distribuir las imágenes de manera uniforme

en todos los cuadros, por lo que, para cada nivel, las imágenes se mezclan y se asignan a diferentes contenedores.

Debido a la configuración piramidal, no es necesario repetir las imágenes dentro de cada conjunto, ya que la suma de porcentajes dentro de la misma capa no supera el 100 % de la base de datos. De esta forma, tenemos la certeza de que cada conjunto tiene diferentes imágenes por nivel, ayudando a producir diversidad en nuestra población. Es importante tener en cuenta que después de la inicialización, mantenemos los conjuntos de datos seleccionados para todo el ciclo evolutivo. El conjunto de imágenes del nivel dos incluyen completamente a las imágenes del nivel uno.

Por otro lado, otra parte importante del proceso de inicialización es la declaración de los parámetros que utiliza el algoritmo en el proceso evolutivo; la Tabla 7 proporciona la lista de parámetros utilizados.

Tabla 7. Inicialización de parámetros para el algoritmo de evolución piramidal.

Parámetros	Descripción
B	[5,4,3,2,1]
N	[6,5,4,4,4]
D	[6.2,12.5,25,50,100]
Generaciones	30
Población Inicial	30
Población por Generación	74
Cruzamiento a nivel cromosoma	0.4
Cruzamiento a nivel gen	0.4
Mutación a nivel cromosoma	0.1
Mutación a nivel gen	0.1
Profundidad de árboles	Selección de profundidad dinámica
Máxima profundidad dinámica	7 niveles
Máxima profundidad real	9 niveles
Selección	Torneo con presión lexicográfica
Supervivencia	Elitismo por caja

4.3.2. Representación y asignación de los individuos

Una vez dividida la base de datos, generamos aleatoriamente una población inicial de 30 individuos, ésta será dividida y asignada a cada una de las cinco posibles cajas del primer nivel (seis individuos por caja). En cuanto a la representación de los individuos, utilizamos la representación convencional para el algoritmo AVC.

Los individuos se evalúan de la forma convencional del Brain Programming. Los mejores individuos

son seleccionados y actualizados a través de operadores genéticos como se explica en la siguiente sección.

4.3.3. Selección, supervivencia y recombinación genética entre niveles

Una vez que se evalúan los individuos del primer nivel $l = 1$, se seleccionan los mejores individuos y se copian directamente en las cajas del siguiente nivel b_2 (uno para cada caja del nivel b_2). Es decir, consideramos que la supervivencia de los individuos se realiza mediante elitismo por cada caja. Sin embargo, únicamente se copian cuatro individuos, ya que en el nivel $l = 2$ se tienen 4 cajas (B(2) cajas).

El conjunto de imágenes del segundo nivel contiene completamente a las imágenes del primer nivel b_1; de forma que los mejores individuos se evalúan en el mismo problema, pero aumentando la complejidad del problema. Entonces, el individuo con mejor desempeño del primer nivel será evaluado con conjuntos de datos más grandes en el segundo nivel y así sucesivamente para los niveles restantes.

Para completar el número de individuos para cada caja, emulamos el proceso realizado en la evolución convencional al cambiar de generación, es decir, realizamos el ciclo de: selección, cruce y mutación. A este proceso le llamamos generación interna.

En cuanto a la selección, ordenamos a todos los individuos de todas las cajas en el nivel uno, de modo que compitan entre sí independientemente de la caja de procedencia. Para cada individuo, calculamos una probabilidad de ser seleccionado que sea proporcional a su aptitud y después realizamos el proceso de recombinación genética. Al realizar este paso, todavía no estamos haciendo el cambio generacional, simplemente los individuos pasan al siguiente nivel de la pirámide, donde los deben lidiar con una mayor presión selectiva. Este proceso se repite hasta llegar al nivel más alto de la pirámide en donde los individuos son entrenados y evaluados utilizando la base de datos completa.

Para este nuevo modelo piramidal, consideramos una generación completa una vez que el algoritmo completa la evaluación y recombinación de todos los individuos en toda la pirámide.

En el segundo nivel $l = 2$, asignamos a cinco individuos para cada caja, pero solo hay cuatro cajas en b_2. Por lo tanto, el número total de individuos en el nivel dos será 20 individuos, $(n_2 \times b_2)$, que competirán en este nivel, evaluando soluciones con diferentes conjuntos de tamaño $d_2 = 12.5\,\%$ de la base de datos. Nuevamente, se lleva a cabo el proceso de generación interna, en el cual clasificamos a los individuos según su aptitud, realizamos el mismo procedimiento de selección, supervivencia, cruzamiento y mutación para obtener la población de la siguiente capa como se explica en Olague *et al.* (2019).

Para el tercer nivel $l = 3$, el algoritmo asigna $n_3 = 4$ individuos para cada caja. En este nivel, hay tres cajas b_3; por lo tanto, tenemos 12 individuos en la tercera capa $n_3 \times b_3$. Estos individuos serán entrenados con subconjuntos compuestos por el 25 % de la base de datos.

En el nivel $l = 4$ solo hay dos cajas b_4 con cuatro individuos cada uno que intentarán clasificar la mitad de la base de datos cada uno. De estas dos cajas, solo un individuo pasará a la última capa, donde se evaluará con la base de datos completa, dicho individuo creará un nuevo conjunto junto con otros tres descendientes obtenidos de la recombinación genética.

Cuando un individuo logra resolver un problema usando por decir, 100 imágenes, es de esperarse que también pueda resolver un problema con 50 muestras; más aún, un problema con 25 imágenes. Esto quiere decir, que los individuos que mejor se desempeñaron en los niveles más altos de la pirámide, en teoría deberían ser capaces de resolver los problemas planteados en los niveles inferiores. Son individuos que ya aprendieron a resolver una buena parte del problema, es por eso que, para sacar ventaja de este procedimiento, diseñamos un sistema de retroalimentación en el que el material genético de estos individuos.

En resumen, este fue el proceso que describe la mitad del primer ciclo de la pirámide. Para crear un ciclo evolutivo completo, proponemos retroalimentar los mejores resultados en cada nivel a la capa anterior en una nueva pirámide. Por lo tanto, tenemos dos pirámides para completar todo el ciclo evolutivo, ver figura 8. Estos pasos para copiar individuos de los niveles inferiores de la primera pirámide a los niveles superiores de la segunda pirámide tienen como objetivo retroalimentar a los individuos competentes a la segunda pirámide como una forma de mejorar el nuevo nivel de población a pesar de haber sido evaluados con menos imágenes. La siguiente sección explica estos pasos.

4.3.4. Retroalimentación hacia arriba y hacia abajo

El modelo piramidal nos ofrece la ventaja de reducir el número de evaluaciones con conjuntos de datos más pequeños. También proporciona un modelo en el que, los individuos de cada nivel que han demostrado ser buenos para clasificar una proporción más sustancial de la base de datos, pasan a niveles inferiores en la segunda pirámide. Tenga en cuenta que los mejores individuos de la primera pirámide fueron copiados en los niveles más altos de la pirámide. En la segunda pirámide, los niveles inferiores se benefician del conocimiento aprendido en el primer ciclo, en este punto se retroalimentan los mejores individuos sobre la nueva estructura; estas soluciones aportan información al proceso de recombinación al cruzarse con otras soluciones. Por lo tanto, el algoritmo contiene procedimientos de retroalimentación

hacia arriba y hacia abajo.

Se ejecuta un proceso de retroalimentación para construir la segunda pirámide a partir de la capa $1-1$, seleccionando la mejor solución de la última capa en la primera pirámide. Luego, la capa $l-2$ de la segunda recibe las mejores soluciones seleccionadas de ambos cuadros en la capa $l-1$ de la primera pirámide. A continuación, la segunda capa recibe las tres mejores soluciones de las tres cajas del nivel l_3. Finalmente, el algoritmo selecciona las cuatro mejores soluciones de l_2 en la primera pirámide y las copia en l_1 en la segunda pirámide.

Para completar la segunda pirámide, creamos la descendencia restante en la primera capa con las soluciones de la primera capa en la primera pirámide mediante las operaciones de recombinación. En general, cada cuadro en la segunda capa y debajo contiene dos individuos copiados directamente que demostraron ser buenos, y dos o tres individuos generados con selección, cruce y mutación. Por lo tanto, el algoritmo llena estas capas a través del mecanismo de retroalimentación, similar a la explicación de la primera pirámide.

El modelo piramidal se puede construir con un número variable de niveles, cajas por nivel o individuos por nivel. Sin embargo, para que esta explicación quede más clara se describirá con los valores que fue utilizada para nuestros experimentos.

Tenemos $B = [5, 4, 3, 2, 1]$ cajas en cada nivel, $N = [6, 5, 4, 4, 4]$ individuos en cada nivel, $D = [6.25, 12.5, 25, 50, 100]$ por ciento de la base de datos en cada nivel, l representa el nivel que se esta estudiando.

Entonces, los individuos dentro de cada uno de los grupos de la pirámide quedan dispuestos como se muestra en la figura 9. El número dentro de cada caja representa el número de individuos que hay en ese grupo.

En el nivel 1 de la generación cero de la pirámide (marcado en la imagen en color azul), podemos encontrar en cada caja a 6 individuos creados de manera aleatoria. El segundo conjunto de datos son las cajas marcadas en color naranja, en el interior de cada una de ellas, podemos encontrar un individuo de los mejores del nivel anterior, y 3 o 4 creados mediante recombinación genética de los individuos del nivel anterior.

Una vez que el primer ciclo concluyó, la distribución de los individuos de la siguiente generación hasta la última, es ligeramente diferente. Tomamos los individuos de los 3 niveles más altos de la pirámide de la generación anterior (en este caso de la generación 0), con ellos acompletamos para formar una población

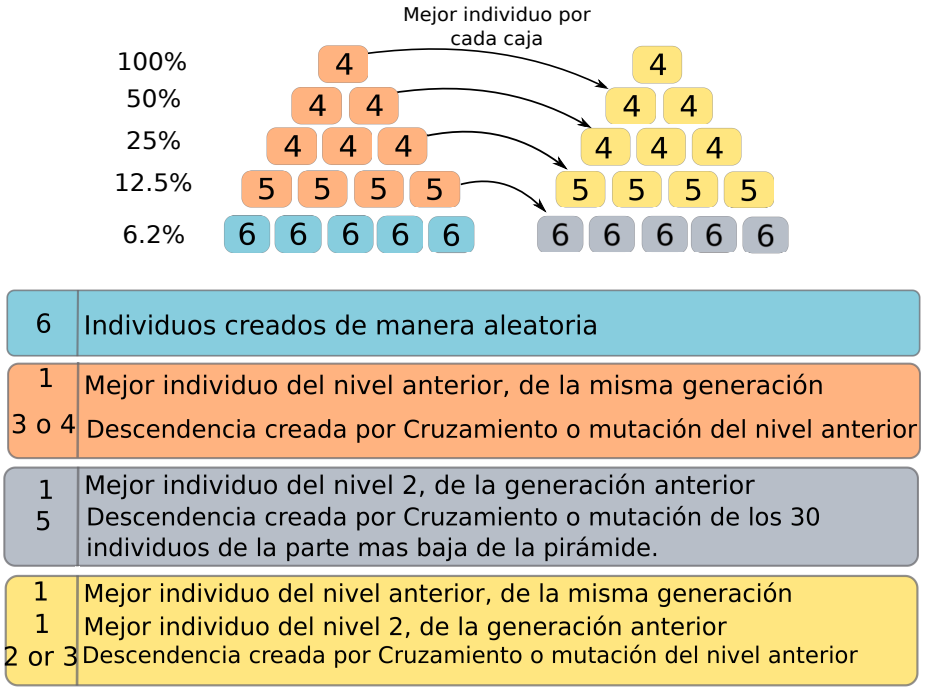

Figura 9. Distribución de los individuos en cada nivel de la pirámide

de 30 con los mejores individuos del nivel 2. Con este conjunto realizaremos la recombinación genética. Entonces, cada caja del conjunto de datos representados en la imagen en color gris, está formado por los mejores cuatro individuos del nivel 2 de la generación cero y el resto de los individuos es resultado de la recombinación genética.

Finalmente, las cajas en color amarillo, contiene un individuo de los mejores del nivel anterior, un individuo de los mejores de la pirámide anterior, y el resto de los individuos son resultado de recombinación genética.

Las generaciones restantes del ciclo evolutivo serán como la segunda pirámide de la imagen.

4.4. Discusión

Los resultados de la evolución piramidal se encuentran en la Sección 5.7, para que puedan compararse con un ciclo evolutivo convencional, sin embargo podemos recalcar brevemente algunos aspectos del modelo.

El modelo piramidal es más rápido que un ciclo evolutivo convencional, esto lo podemos corroborar con el número de operaciones que realiza. Supongamos que tenemos un conjunto de 1000 imágenes de entrenamiento de nuestro modelo. En un ciclo evolutivo convencional, como todos los individuos deben eva-

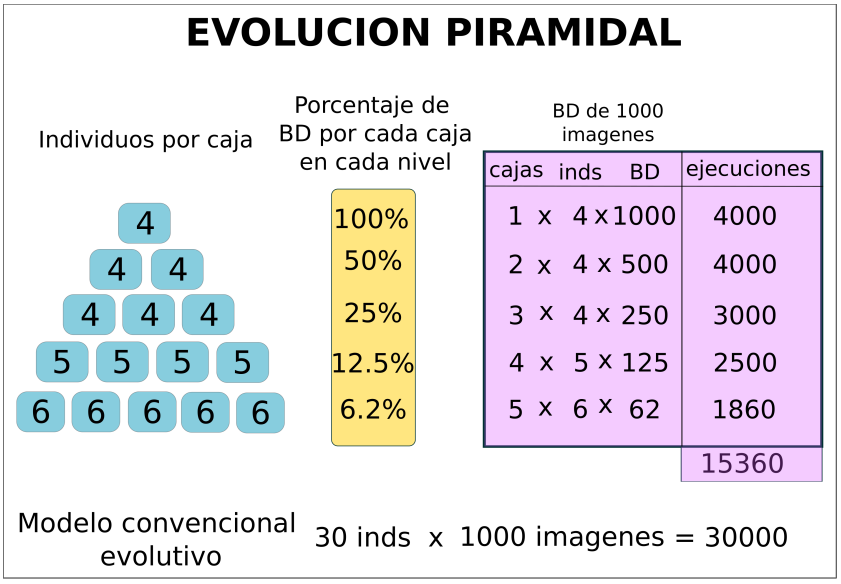

Figura 10. Simulación sobre el número de evaluaciones del programa

.

luarse en las mil imágenes, se estarían realizando $número de individuos \times número de imgenes \times número de generaciones$. Lo que en total serían $30 \times 30 \times 1000$ evaluaciones del programa.

Con el modelo piramidal (véase figura 10), a pesar de que, en total estamos evaluando a 74 individuos por cada generación (la suma de todos los individuos de una pirámide), el porcentaje de imágenes que atiende cada uno es mucho menor, y como podemos observar en la figura, el número de ejecuciones se reduce prácticamente a la mitad.

Capítulo 5. Experimentos y resultados

En este capítulo se presentan los experimentos llevados a cabo para respaldar nuestra propuesta. Podemos encontrar también la descripción del Hands-on, una estrategia que fue implementada con éxito para mejorar el desempeño en la búsqueda de una solución óptima para el problema. En este capítulo también se incluye un análisis de la robustez de las soluciones encontradas mediante el Brain Programming.

En este trabajo se abordó el problema de clasificación de imágenes de arte de acuerdo al medio empleado. Los principales sistemas usados para clasificación se basan en aprendizaje profundo, es por eso que en este estudio implementamos varios sistemas basados en redes neuronales convolucionales para comparar los resultados obtenidos. La descripción de estos trabajos la podemos encontrar en la sección 5.6.

Una de las mayores desventajas de los métodos basados en aprendizaje profundo es la susceptibilidad que tienen ante los ataques adversarios. Estos consisten en presentar una entrada al modelo que, de forma intencional, ha sido ligeramente modificada, de manera que es prácticamente imperceptible al ojo humano, pero que es capaz de hacer que el modelo genere una salida incorrecta.

Las redes neuronales son muy frecuentemente utilizadas en la vida diaria en sistemas de reconocimiento o clasificadores. En este sentido, los ataques adversarios representan un problema de seguridad, ya que incluso tienden a ser transferibles entre los diferentes modelos clasificadores, es por eso que decidimos evaluar el desempeño del Brain Programming ante este tipo de perturbaciones.

En la sección 5.8.3 podemos encontrar la descripción de los ataques adversarios que se usaron para este estudio, así como los resultados que obtuvimos.

5.1. Descripción de los experimentos

La metodología usada para realizar los experimentos sigue el protocolo de presencia/ausencia de la clase que se está estudiando. Utilizamos una base de datos de imágenes de arte del sitio web de kaggle [1]. Separamos las imágenes en dos conjuntos, uno para entrenamiento y uno para validación. Posteriormente agregamos un tercer conjunto de imágenes que fueron utilizados para prueba, pero esta vez las imágenes

[1] https://www.kaggle.com/thedownhill/art-images-drawings-painting-sculpture-engraving

son de una base de datos de Wikiart [2], la descripción de las bases de datos usadas se hará en la sección 5.3.

La implementación fue realizada en MATLAB en una estación de trabajo Dell Precision con el sistema operativo Linux OpenSuse Leap 15.0. Usamos un tamaño de población de 30 individuos. De la misma manera, los programas se ejecutan durante solo 30 generaciones. Se utilizan parámetros atípicos de programación genética, que son especificados en la tabla 7, esto se debe al análisis del problema visual y la estructura del modelo jerárquico.

A continuación se describe a grandes rasgos los experimentos realizados:

- Se ejecutaron 15 ciclos evolutivos completos del Brain Programming para cada una de las clases estudiadas.
- Se implementó el Hands-On, una técnica para mejorar el desempeño del ciclo evolutivo que será explicada en la sección 5.5 y se ejecutó durante 10 corridas del programa.
- Se hizo el mismo experimento en redes neuronales convolucionales, generamos una red neuronal básica y la entrenamos desde cero. Entrenamos las redes Alexnet, ResNet18, Resnet101, VGG mediante transferencia de aprendizaje usando las imágenes de nuestra base de datos.
- Se implementó un conjunto de ataques adversarios para verificar si le afectaban al desempeño del Brain Programming.
- Se realizó el análisis de las soluciones obtenidas

5.2. Métricas de evaluación

Para medir el desempeño de nuestro clasificador empleamos la medida de precisión de clasificación, que es simplemente la tasa de clasificaciones correctas dada por la siguiente fórmula:

$$Precision = \frac{1}{N}\sum_{n=1}^{N} d(y'_n, y_n) \quad , \tag{3}$$

donde N es el total de imágenes de prueba, y'_n es la etiqueta predicha para la imagen n, y_n es la etiqueta original para la imagen n , y $d(x, y) = 1$ si $x = y$ y 0 en caso contrario.

[2] https://www.wikiart.org

Como medida de robustez, utilizamos la relación de precisión entre ejemplos adversarios e imágenes sin ataque, esta medida fue propuesta por Kurakin *et al.* (2017). Esta métrica significa que, si la relación alcanza uno, la precisión obtenida con los ejemplos adversarios y las imágenes limpias, es la misma. Lo que significa que el ataque no logró confundir a la red neuronal. Sin embargo, si este valor tiende a cero, significa que el ataque logró engañar al clasificador. Si esta relación excede 1, implica que el ataque adversario está ayudando a corregir imágenes que habían sido mal clasificadas. La siguiente ecuación calcula la razón:

$$Razon = \frac{acc_{adv}}{acc_{limpias}} \quad , \tag{4}$$

donde acc_{adv} es la precisión de clasificación del ejemplo adversario, y $acc_{limpias}$ es la precisión de clasificación en las imágenes sin ataque.

5.3. Base de datos de imágenes

5.3.1. Kaggle

La base de datos que utilizamos fue descargada del sitio de kaggle [3], consta de aproximadamente 9000 imágenes de arte, incluidas las clases Dibujos, Grabados, Pinturas, Iconografía y Esculturas. Esta base de datos contiene imágenes que fueron recopiladas de diferentes fuentes para clasificar diferentes estilos de arte, incluye imágenes de alta resolución de diferentes tamaños. De acuerdo a la información proporcionada por el sitio web, las cinco clases fueron recopiladas de imágenes de Google, imágenes de Yandex y el museo ruso virtual. Para todas las clases, separamos las imágenes en conjuntos de entrenamiento y validación. Agregamos como no-clase la clase de fondo de Caltech-101. En las Figuras 11 a 16 se proporcionan ejemplos de las categorías principales con fines de visualización.

5.3.2. Wikiart

Para corroborar los datos obtenidos en nuestra propuesta, decidimos hacer una comparación utilizando una base de datos estándar. Tomamos como referencia la base de datos WikiArt [4] que contiene aproximadamente 250.000 imágenes de obras de arte, realizadas por más de 3000 artistas.

[3] https://www.kaggle.com/thedownhill/art-images-drawings-painting-sculpture-engraving
[4] https://www.wikiart.org

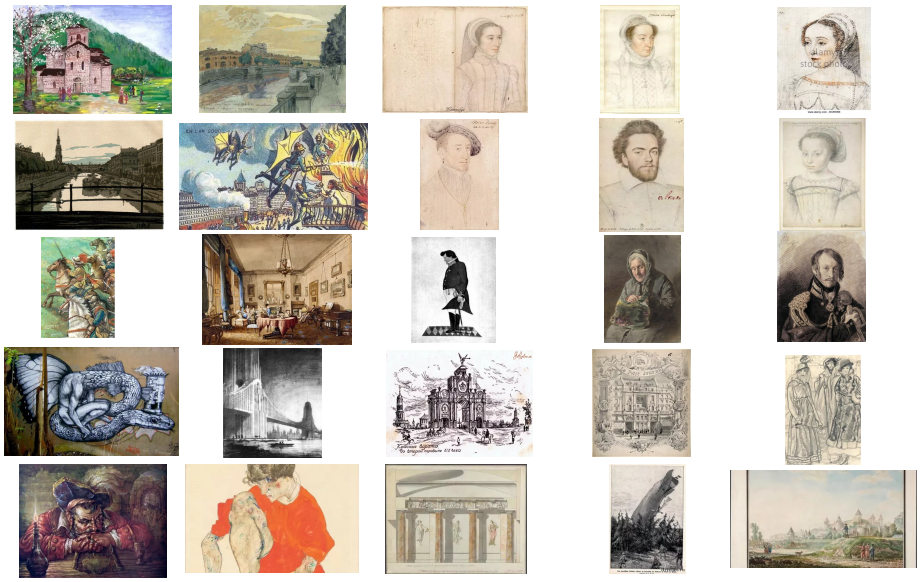

Figura 11. Imágenes de ejemplo de la categoría "Dibujos". La categoría incluye dibujos y acuarelas. Nótese la diversidad en color, tamaño, perspectiva, estilo y contenido de las imágenes. Hay 1108 imágenes en esta clase.

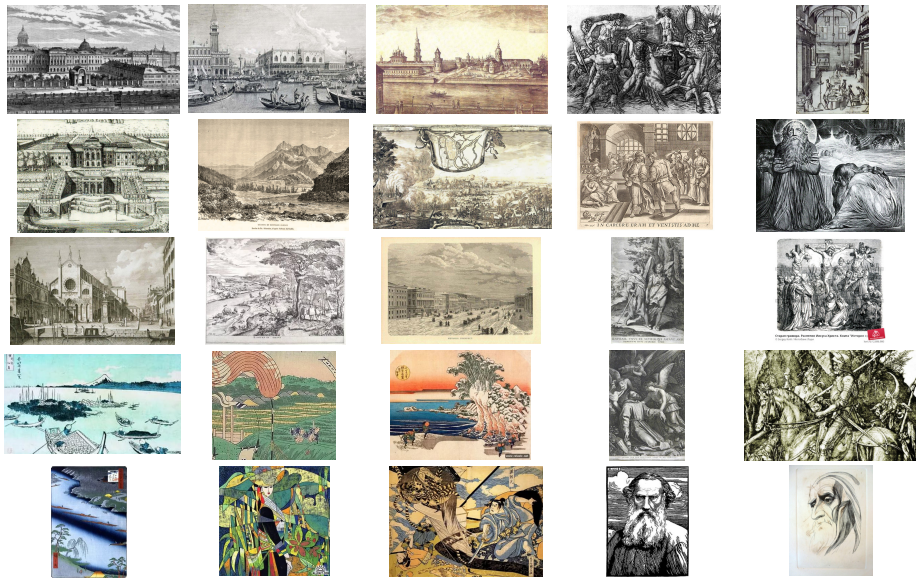

Figura 12. Imágenes de ejemplo de la clase "Grabados"de la base de datos de Kaggle. Esta clase representa un área de las bellas artes que es usualmente impresa en una superficie plana realizando incisiones en un material plano y rígido. Estas incisiones son hechas con un burin. En esta categoría hay 757 imágenes.

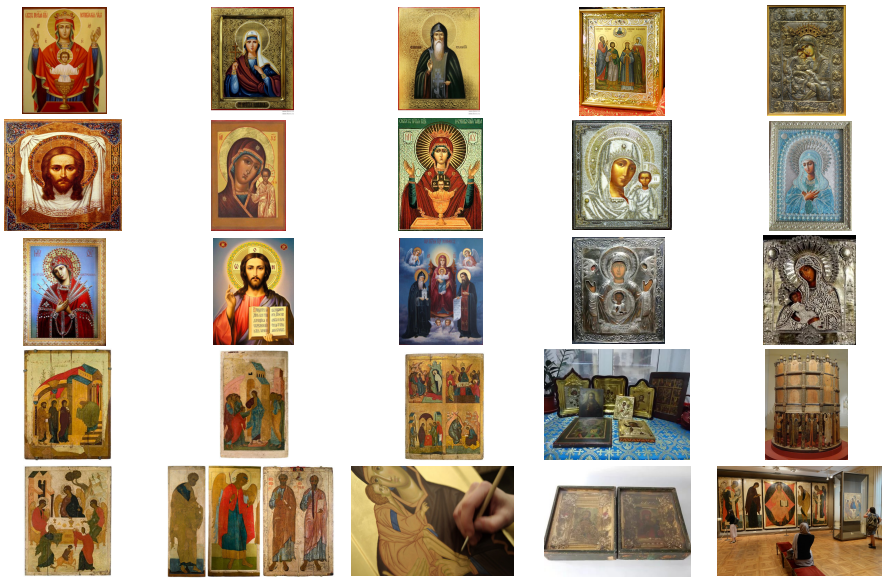

Figura 13. Imágenes de ejemplo de la clase "Iconografía"de la base de datos de Kaggle. El término se refiere a la producción de imágenes con motivos religiosos llamadas "íconos" en la tradición cristiana, ortodoxa y bizantina. En esta categoría hay 2076 imágenes.

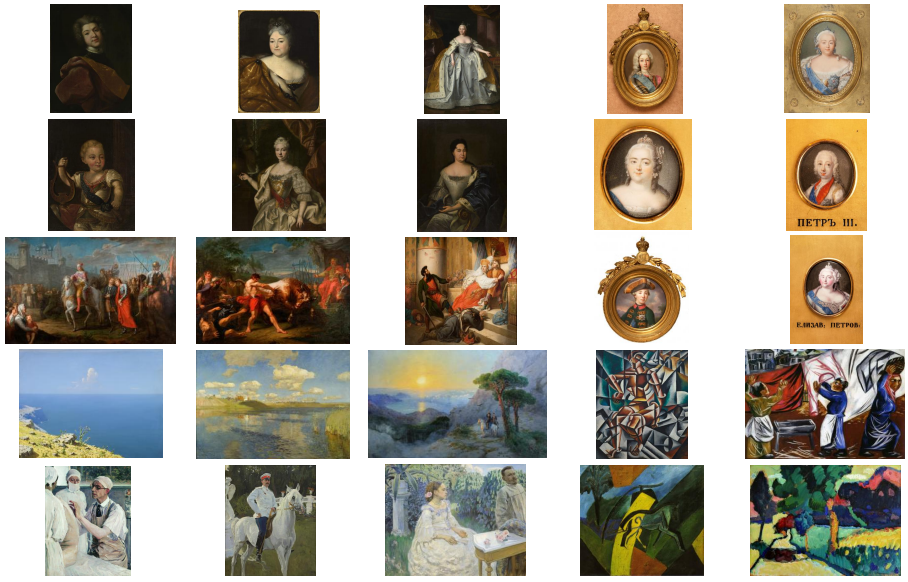

Figura 14. Imágenes de ejemplo de la clase "Pinturas"de la base de datos de Kaggle. Pintar se refiere a la práctica de aplicar pintura, pigmento, color o cualquier otro medio sobre una superficie sólida. Incluye elementos como dibujo, gestos, composición, narración o elementos abstractos. Las pinturas pueden ser naturalistas o representacionales, fotográficas, simbólicas, emotivas o políticas. En esta categoría hay 2043 imágenes.

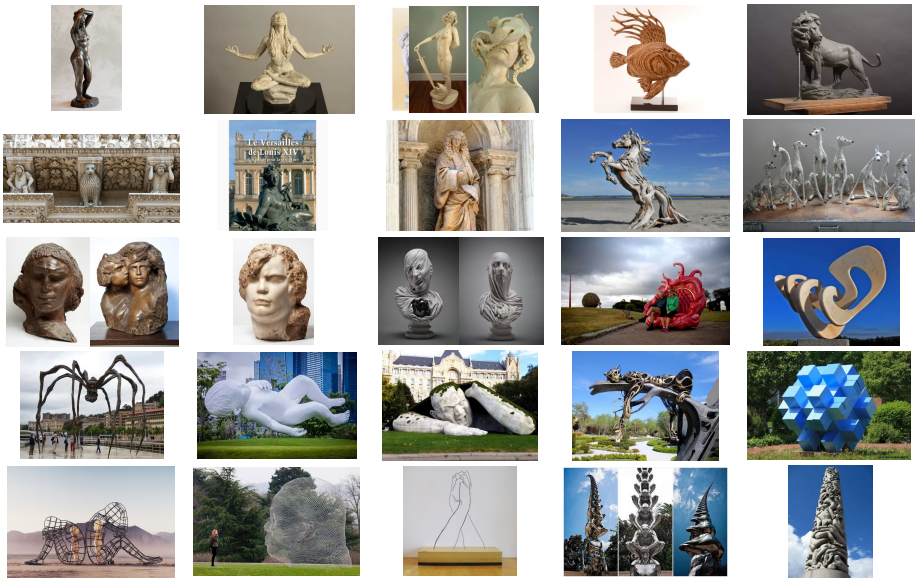

Figura 15. Imágenes de ejemplo de la clase "Esculturas"de la base de datos de Kaggle. La escultura es una rama de las artes visuales que contiene obras en tres dimensiones. Es una de las llamadas artes plásticas. Incluye grabado (que para su creación se remueve material) y modelado (que agrega material, como la plastilina). Puede ser realizado en piedra, metal, cerámica, madera y otros materiales. En esta categoría hay 2076 imágenes.

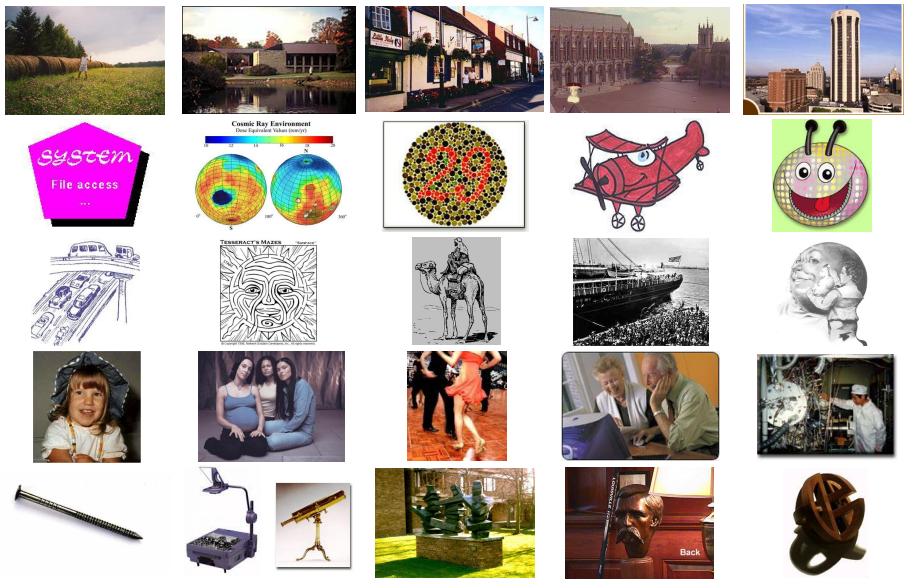

Figura 16. Imágenes de ejemplo de la clase "fondo"de la base de datos Caltech-101. Es necesario hacer notar, que por error en el diseño del experimento, esta clase contiene imágenes que podrían pertenecer a las 5 clases estudiadas en este trabajo. Incluye algunas esculturas y otros elementos similares a grabados y pinturas. There are 468 images.

El objetivo es emular un escenario del mundo real donde los modelos propuestos son entrenados con un conjunto de datos y se evalúan con puntos de referencia estándar.

WikiArt organiza las imágenes de arte en diferentes categorías, ya sea por estilo, género, medio o por el artista, y dentro de cada una podemos encontrar múltiples subcategorías.

Al realizar nuestro estudio con las imágenes de WikiArt, pudimos notar que en esta base de datos no existían exactamente las mismas clases que ya habíamos estudiado, por lo que tuvimos que hacer una selección manual de las imágenes correspondientes a cada una de las categorías. La Figura 17 muestra algunas imágenes seleccionadas de WikiArt.

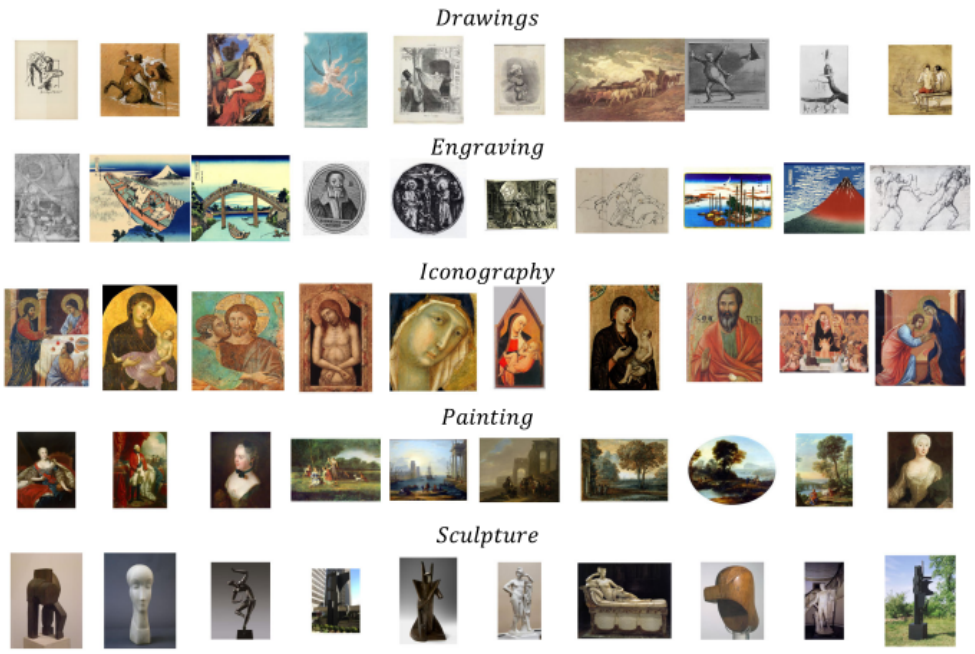

Figura 17. Estas imágenes fueron descargadas del sitio WikiArt para evaluar los mejores programas previamente entrenados con la base de datos de Kaggle.

5.4. Resultados preliminares

Nuestro método demora 450 horas o 18,75 días para la clase de dibujos, 544 horas o 22,66 días para la clase de grabados, 810 horas o 33,75 días para la clase de Iconografía, 597 horas o 24,87 días para la clase de Pinturas y 562 horas o 23,41 días para la clase Escultura. Utilizamos diez estaciones de trabajo para acelerar el proceso de optimización. La tabla 12 muestra un resumen de los resultados estadísticos después de ejecutar 15 veces por cada clase. Mientras que la Tabla 8 muestra el tamaño de los conjuntos de imágenes utilizados durante el entrenamiento y la validación. De acuerdo con los trabajos revisados, incluimos dos métodos de aprendizaje profundo para la comparación: CNN básica entrenada desde cero

Tabla 8. En esta tabla se muestran los resultados de aplicar el Brain Programming al problema de clasificación de imágenes de arte. Incluimos los resultados obtenidos mediante aprendizaje profundo para comparación.

Run	Dibujos	Grabados	Pinturas	Iconografía	Esculturas
1	80.15	80.03	90.27	83.52	81.10
2	84.73	**83.46**	89.71	89.11	80.65
3	78.84	83.30	86.61	87.09	84.21
4	80.65	74.95	92.94	85.60	85.37
5	83.46	78.88	88.43	87.56	84.74
6	79.00	79.70	**98.24**	**89.37**	84.83
7	83.84	76.26	95.93	87.17	84.74
8	74.32	81.99	96.46	86.54	85.64
9	75.94	76.26	90.43	87.01	**89.37**
10	82.57	82.16	90.51	83.71	84.92
11	78.88	74.63	90.74	85.83	85.28
12	**86.01**	79.54	89.47	80.15	86.19
13	79.75	73.48	90.27	84.20	84.83
14	83.33	79.86	92.82	88.72	83.01
15	75.76	81.50	92.50	83.81	84.92
Mean	80.48	79.19	91.40	86.34	84.70
Std. Dev.	3.48	3.19	3.26	2.02	1.93
Min.	74.32	74.14	87.07	83.52	81.10
Max.	86.01	83.46	98.24	89.37	89.37
From scratch CNN	76.18 ± 2.38	79.16 ± 4.88	91.78 ±2.43	91.49 ± 0.51	81.14 ± 2.51
AlexNet	89.34 ± 0.89	92.50 ± 1.98	97.34	96.46 ± 0.36	93.81 ± 0.91

y AlexNet (Krizhevsky *et al.*, 2012).

De la tabla 8 podemos observar que el mejor programa encontrado para la clase Pinturas prácticamente resuelve el problema, y su precisión es mejor que AlexNet. Solo se equivoca en 40 imágenes de todo el conjunto de datos. En cuanto a la clase dibujos, el mejor programa encontrado se aproxima al desempeño AlexNet. Para la clase Dibujos, nuestro programa también supera a AlexNet, mientras que, en escultura e iconografía, nuestros mejores programas alcanzan alrededor del 90 % y son ligeramente inferiores a AlexNet. La clase Grabados es la que tiene la puntuación más baja. Realizamos un análisis más exhaustivo a esta clase y los resultados se presentan en la sección 5.5.1.

5.5. Evolución Hands-On

En general cuando se trabaja con cualquier técnica de computación evolutiva, el uso de principios aleatorios es aplicado con bastante frecuencia. En esta sección se explica una metodología que imple-

mentamos para evitar la aplicación innecesaria de soluciones arbitrarias o no planificadas dentro del algoritmo. Esto nos permite avanzar hacia una metodología más orientada a objetivos. La primera idea propuesta en Olague y Chan-Ley (2020) es llamada evolución hands-on y consiste en usar las mejores soluciones obtenidas durante experimentos previos y formar con ellos la población inicial para un nuevo conjunto de ejecuciones de programación cerebral. La tabla 12 proporciona los resultados preliminares logrados para las cinco clases.

En particular para este experimento, utilizamos las 15 ejecuciones reportadas en la tabla 8. Por cada una de las ejecuciones, seleccionamos a los mejores dos individuos, en total conseguimos treinta individuos que sirvieron como población inicial de un nuevo conjunto de experimentos. En los resultados se puede observar que la Clase Dibujos mejoró su desempeño superando a AlexNet. Incluso para la clase de pinturas que ya tenía un rendimiento por encima del de AlexNet, se obtuvo un mejor desempeño. Además, se mejoró ligeramente el desempeño para las clases de Iconografía y Escultura. Los mejores individuos encontrados para cada una de las clases se presentan en las tablas 10 y 11

Sin embargo, todavía quedaba una clase, que no ha mostrado mejoras, por lo que decidimos examinar más de cerca el conjunto de datos. El grabado es un arte gráfico en el que el artista utiliza diferentes técnicas de impresión, que tienen en común el dibujar una imagen sobre una superficie rígida, llamada matriz, dejando una huella que después alojará tinta y será transferida por presión a otra superficie como papel o tela, lo que permite obtener varias reproducciones de las estampas

5.5.1. Grabados

En la base de datos de Kaggle, había dos tipos diferentes de grabados; la mayoría de ellos eran grabados con un solo color que definía la obra de arte. Los bordes y las sombras en esas obras se hicieron utilizando diferentes grosores de líneas o incisiones, pero la impresión fue hecha con solo un color de tinta, lo que da la ilusión de tener una imagen en escala de grises.

El otro estilo fueron los grabados japoneses, que introducen color en las imágenes, un ejemplo de cada tipo se puede observar en la figura 18. Al ser dos conjuntos de imágenes completamente diferentes el programa no logró encontrar una solución que generalice las características y resuelva ambas tareas. Entonces, decidimos dividir el problema en dos partes, separando los conjuntos de imágenes de la siguiente manera.

Dada una imagen de entrada en RGB (imagen en color), el objetivo es saber si la imagen está en

Tabla 9. Esta tabla muestra los resultados obtenidos después de evolucionar el conjunto de las mejores soluciones obtenidas mediante el Brain Programming para el problema de clasificación de imágenes de arte. Incluimos los resultados obtenidos mediante aprendizaje profundo para comparación.

Run	Dibujos	Grabados	Pinturas	Iconografía	Esculturas
1	90.41	84.61	**99.02**	91.11	89.37
2	89.15	84.78	98.24	90.48	89.73
3	**91.32**	84.12	98.24	89.85	89.37
4	90.41	**84.94**	98.32	90.87	89.37
5	90.78	84.45	98.64	91.11	89.74
6	89.15	84.29	98.32	**91.18**	89.74
7	89.18	83.96	98.32	90.95	89.83
8	89.31	83.96	98.32	90.79	89.37
9	88.78	83.63	98.32	90.72	**89.92**
Original	86.01	83.46	98.24	89.37	89.37
Mean	89.76	84.54	98.41	90.64	89.62
Std. Dev.	0.88	0.87	0.24	0.59	0.23
Best Solution	91.32	84.94	99.02	91.18	89.92
Improvement	5.31	3.28	0.78	1.73	0.25
AlexNet	89.34 $\pm$ 0.89	92.50 $\pm$ 1.98	96.46 $\pm$ 0.36	98.14 $\pm$ 0.43	93.81 $\pm$ 0.91

escala de grises, es decir, si hay sombras del mismo color como la imagen de la izquierda en la Fig. 18, o cuando en la imagen se presentan colores diferentes. Para resolver el problema usamos la teoría de indexación de imágenes. Consiste en mapear los valores de cada píxel en la imagen de entrada a los valores de un mapa de color. Los mapas de color pueden tener cualquier longitud, pero con tres columnas, una para cada componente RGB que indica las intensidades de color de rojo, verde y azul. Las imágenes que contienen grabados en escala de grises tienden a crear pocos mapas de color; mientras que las imágenes a color, generan más mapas. Como paso final, creamos una regla de decisión que actúa sobre las imágenes de entrada para aplicar el clasificador correcto. Este experimento nos da las primeras pistas sobre la posibilidad de considerar la teoría de la indexación visual como una forma de abordar los problemas de clasificación.

Considerando estas diferencias, dividimos el conjunto de datos en dos clases y entrenamos dos nuevos programas mediante el BP (obtenidos de cinco ejecuciones cada uno), lo que resultó en una precisión de clasificación del 92 % para escala de grises y 98.33 % para imágenes grabadas en color. Luego aplicamos los mejores programas descubiertos a la base de datos WikiArt obteniendo 92.3 %, véase Tabla 10 (tenga en cuenta que estas imágenes están en escala de grises). Como resultado, nuestro mejor programa entrenado coincide con el rendimiento de AlexNet en ambos conjuntos de datos. En cuanto al color, realizamos otra prueba utilizando la clase ukiyo-e (grabados japoneses) de WikiArt para probar el mejor programa con una puntuación del 89,71 % con un total de 1167 imágenes seleccionadas. Esta diferencia

se debe a los conjuntos desequilibrados, ya que Kaggle tiene solo 79 imágenes en color; por tanto, el rendimiento alcanzado es alto considerando las pocas imágenes utilizadas durante el entrenamiento.

Figura 18. La imagen muestra dos diferentes imágenes de la misma categoría grabados. Se puede ver la diferencia entre ambas imágenes y el reto que representa para el clasificador.

5.6. Métodos basados en redes neuronales

En LeCun *et al.* (1989), los autores introducen las bases de las Redes neuronales convolucionales (CNN [5]). Sin embargo, la primera vez que las CNN comienza a llamar la atención fue con el desarrollo de AlexNet (Krizhevsky *et al.*, 2012), un modelo de CNN diseñado para participar en la competencia: ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)(Russakovsky *et al.*, 2015) 2012, donde logró reducir a la mitad la tasa de error en la tarea de clasificación de imágenes.

AlexNet consta de 5 capas convolucionales, 3 de max-pooling, 2 de normalización y 3 capas completamente conectadas (la última con 1000 salidas), en total 60 millones de parámetros y 500.000 neuronas. Además, LeCun *et al.* (1989) introdujo el uso de ReLU ([6]) como función de activación, lo que produce un entrenamiento mucho más rápido que el uso de funciones sigmoideas o tanh. Para evitar el sobreajuste, también introdujeron el método dropout y generación de datos artificiales.

Otro modelo de aprendizaje profundo que contribuyó de manera importante al estado del arte fue la red VGG del Grupo de Geometría Visual de la Universidad de Oxford (Simonyan y Zisserman, 2015). La red VGG aumentó la profundidad de las CNN, creando un VGG-16 con 13 capas convolucionales y 3 capas completamente conectadas, posteriormente añadieron 3 capas convolucionales adicionales para VGG-19. Además, redujeron los filtros que utilizaron al tamaño más pequeño para capturar la noción de arriba/abajo, izquierda/derecha y centro usando un filtro de 3x3. VGG se distinguió por su desempeño

[5] por sus siglas en inglés: Convolutional Neural Network
[6] del inglés Rectified Linear Unit

Tabla 10. Mejores individuos encontrados mediante el Hands-on para las clases dibujos, grabados, iconografías y esculturas

	Dibujos vs Fondo
$*EVO_O$	'kSum(imRound(ip_GaussDy_1(supremum(kSum(supremum(ip_GaussDy_1 (supremum(ip_GaussDx_1(C)kSum(supremum(ip_GaussDx_1(C), imRound(ip_GaussDx_1(M))), 0.31))), imRound(ip_GaussDx_1(M))), 0.31), kSum(supremum(ip_GaussDx_1(C), imRound(ip_GaussDx_1(C))), 0.31))), 0.31)'
EVO_C	'kSum(kDiv(kSum(imFloor(imCeil(kDiv(kSum(kSum(kSum(kSum(imFloor (imFloor(S)), 0.20), 0.20), 0.20), 0.20), 0.31))), 0.20), 0.31), 0.20)'
EVO_S	'ip_imdivide(hitmissDmnd(hitmissDsk(hitmissDsk(hitmissDmnd (hitmissDmnd(bottomHat(ip_imdivide(hitmissDmnd(hitmissDsk(hitmissDsk (hitmissDmnd(bottomHat(dilateDsk(G))))))), R))))))), R)'
$*MM_1$	'ip_Sqrt(ip_Sqrt(ip_Sqrt(ip_GaussDx_1(ip_GaussDy_1(imagen)))))'
MM_2	'ip_GaussDy_1(ip_imsubtract(ip_GaussDy_1(ip_imsubtract(ip_GaussDy_1 (ip_GaussDy_1(imagen)), ip_Sqr(ip_GaussDy_1(ip_GaussDy_1(imagen))))), ip_GaussDy_1(ip_imsubtract(ip_GaussDy_1(ip_GaussDy_1(imagen)), ip_GaussDy_1(ip_GaussDy_1(imagen))))))'
	Grabados vs Fondo
$*EVO_O$	'ip_Gauss_1(imagen)'
EVO_C	'ip_imcomplement(ip_Exp(ip_Exp(ip_Exp(ip_Exp(ip _Exp(RedGreenOpon(image)))))))'
EVO_S	Y
$*MM_1$	'ip_Logarithm(ip_Half(ip_Half(ip_GaussDx_1(imagen))))'
MM_2	'ip_GaussDx_1(ip_GaussDx_1(imagen))'
MM_3	'ip_Logarithm(ip_imdivide(ip_GaussDy_1(ip_GaussDy_1(imagen)), ip_imadd (ip_imadd(ip_GaussDy_1(ip_GaussDy_1(imagen)), ip_imadd(ip_GaussDy_1(ip_GaussDy_1 (imagen)), ip_imadd(ip_GaussDy_1(ip_GaussDy_1(imagen)), ip_GaussDx_1(ip_GaussDy_1 (imagen))))), ip_imadd(ip_GaussDy_1(ip_GaussDy_1(imagen)), ip_GaussDx_1(ip_GaussDy_ 1(imagen))))))'
	Iconografía vs Fondo
$*EVO_O$	'ip_GaussDy_1(kSum(ip_GaussDy_1(ip_GaussDy_1(ip_GaussDy_1(ip_Gauss Dy_1(ip_Gauss_2(ip_GaussDx_1(ip_GaussDx_1(C)))))), 0.68))'
EVO_C	'R'
EVO_S	'hitmissSqr(kDiv(skeletonShp(kDiv(skeletonShp(Y), 0.74)), 0.74))'
$*MM_1$	'ip_Sqrt(ip_imabsadd(ip_imabsadd(ip_imabsadd(ip_imabsadd(ip_Sqrt(ip_ imabsadd(ip_imabsadd(ip_imabsadd(ip_imabsadd(ip_imabsadd(ip_GaussDx_1(ip_ GaussDy_1(imagen)), ip_Sqrt(ip_imabsadd(ip_GaussDx_1(ip_GaussDx_1(imagen)), ip_ GaussDx_1(ip_GaussDy_1(imagen))))), imagen), ip_Sqrt(ip_Sqrt(ip_imabsadd(ip_ imabsadd(ip_imabsadd(ip_imabsadd(ip_imabsadd(ip_GaussDx_1(ip_GaussDy_1(imagen)), ip_Sqrt(ip_imabsadd(ip_GaussDx_1(ip_GaussDx_1(imagen)), ip_GaussDx_1 (ip_GaussDy_1(imagen))))), imagen), ip_Sqrt(ip_imabsadd(imagen, ip_GaussDx_1 (ip_GaussDy_1(imagen))))), imagen), imagen)))), imagen), imagen)), imagen), ip_Sqrt (ip_imabsadd(imagen, ip_GaussDx_1(ip_GaussDy_1(imagen))))), ip_GaussDy_1 (ip_GaussDy_1(imagen))), imagen))'
MM_2	'ip_GaussDy_1(ip_GaussDy_1(imagen))'
	Esculturas vs Fondo
$*EVO_O$	'ip_GaussDx_1(kSust(ip_GaussDx_1(kSust(kSust(ip_GaussDx_1 (kSust(imFloor(ip_imabs(ip_GaussDx_1(ip_GaussDy_1(imagen)))), 0.46)), 0.46), 0.46)) , 0.46))'
EVO_C	'ip_imcomplement(imFloor(ip_Sqrt(imFloor(M))))'
EVO_S	'kSum(M, 0.00)'
$*MM_1$	'ip_imabs(ip_GaussDx_1(ip_GaussDy_1(imagen)))'

* EVO = operador visual evolucionado (por sus siglas en inglés) y MM = Mapa Mental

Tabla 11. Mejores individuos encontrados mediante el Hands-on para las clase pinturas

	Pinturas vs Fondo
$*EVO_O$	*'ip_imadd(ip_GaussDx_1(ip_GaussDx_1(H)), ip_imabsadd(ip_GaussDx_1 (ip_GaussDx_1(H)), ip_GaussDx_1(ip_GaussDx_1(H))))'*
EVO_C	*'RedGreenOpon(image)'*
EVO_S	*'hitmissSqr(ip_imadd(erodeDmnd(erodeDmnd(kDiv(K, 0.88))), ip_imadd (ip_imsubtract(erodeDsk(B), erodeDsk(B)), ip_imsubtract(Y, erodeDsk(B)))))'*
$*MM_1$	*'ip_imabsadd(ip_Sqrt(ip_imabsadd(ip_imabsadd(ip_Sqrt(ip_Logarithm (ip_GaussDx_1(ip_Logarithm(ip_GaussDx_1(ip_GaussDx_1(ip_GaussDy_1(imagen)))))))), ip_GaussDx_1(ip_GaussDy_1(imagen))), ip_Logarithm(ip_GaussDx_1(ip_GaussDy_1 (imagen))))), ip_Logarithm(ip_imabs(ip_imabs(ip_GaussDx_1(ip_Logarithm(ip_GaussDx_ 1(ip_Logarithm(ip_GaussDx_1(ip_GaussDx_1(ip_GaussDy_1(imagen)))))))))))'*

* EVO = operador visual evolucionado (por sus siglas en inglés) y MM = Mapa Mental

Tabla 12. Esta tabla muestra un resumen de los resultados obtenidos después de evolucionar el mejor conjunto de soluciones utilizando el Brain Programming para el problema de clasificación de imágenes de arte. Se incluyen resultados obtenidos mediante algunas redes neuronales profundas para comparación.

	Dibujos	Grabados	Pinturas	Iconografía	Esculturas
Evolutionary process	80.48±3.48	79.19±3.19	91.40±3.26	86.34±2.02	84.70±1.93
Best BP	86.01	83.46	98.24	89.37	89.37
ImprovedBP	**91.32**	84.51	**99.02**	91.18	89.73
Test Wikiart BP	96.08	87.28	100	92.21	85.34
From scratch CNN	76.18 ± 2.38	79.16 ± 4.88	91.78 ±2.43	91.49 ± 0.51	81.14 ± 2.51
AlexNet	89.34 ± 0.89	92.50 ± 1.98	97.34±0.42	96.46 ± 0.36	93.81 ± 0.91
AlexNet Wikiart Test	82.61	92.24	91.82	97.52	89.74

en tareas de reconocimiento y localización en competencias internacionales como ILSVRC, también en otras bases de datos de reconocimiento de imágenes.

ResNet (He *et al.*, 2016) (Deep Residual Learning for Image Recognition) también contribuyó a redefinir la capa como funciones residuales de aprendizaje en la arquitectura de CNN. Esto ayuda a mitigar el problema del cuello de botella de la fase de entrenamiento en las CNN. ResNet mostró su capacidad para entrenar su arquitectura con una profundidad de hasta 152 capas y menor complejidad que GoogLeNet. Además, ResNet ganó el ILSVRC 2015 en la tarea de clasificación logrando por primera vez la tasa de error por debajo del error de percepción humana con un error de 3.57 %.

5.7. Evolución piramidal

Para llevar a cabo la prueba del modelo piramidal propuesto, usaremos las mismas clases que se está estudiando en este trabajo. Imágenes de arte de una base de datos de Kaggle [7], más detalles sobre la base de datos podemos encontrarlos en la sección 5.3.

En las primeras ejecuciones del programa, logramos observar, que muchas veces el programa convergía demasiado rápido y la mayoría de los individuos en la población terminaban siendo iguales, por lo que decidimos implementar una medida para promover la diversidad llamda Niching.

El niching consiste en repartir el fitness de una solución que se encuentre repetida. Cuando varios individuos comparten un mismo fitness, éste se les reduce para que baje la probabilidad de qser seleccionado. Promoviendo la selección de otras soluciones, que, si bien no tienen el mejor desempeño, aportan un material genético diferente a la siguiente generación.

El resultado de evolucionar el modelo piramidal se reporta en la tabla 13. En la primera columna se presenta la clase que se esta evaluando, en la segunda podemos encontrar el número de corrida para cada clase;la tercera nos muestra la aptitud de clasificación que se obtuvo, el número representa el porcentaje de imagenes correctamente clasificadas.

Tabla 13. Resultado de ejecutar el modelo piramidal propuesto para la base de datos de kaggle

Clase	Corrida	Aptitud	Generación	
Escultura	1	84.01	13	Niche
Escultura	2	83.01	4	niche
Escultura	3	83.65	12	Niche
Escultura	4	82.65	17	niche
Escultura	5	83.38	1	Niche
Escultura	1	81.56	7	No Niche
Escultura	2	82.83	4	No Niche
Escultura	3	83.20	19	No Niche
grabados	1	73.61	5	niche
grabados	2	72.66	14	No Niche
iconografía	1	85.99	5	Niche
iconografía	2	88.90	6	NoNiche

La siguiente columna nos habla de la generación en la que el programa encontró la solución mostrada.

[7] https://www.kaggle.com/thedownhill/art-images-drawings-painting-sculpture-engraving

Este valor lo incluimos para saber si el programa sigue evolucionando, es decir, si sigue encontrando buenas soluciones o si encontró un mínimo local. Finalmente en la última columna indicamos si esa ejecución fue hecha empleando el niching o no.

Realizamos una variación al modelo cambiando los valores de configuración de la pirámide, de forma que haya más individuos por cada nivel, en este sentido, cabría la posibilidad de que encontrara la solución en menos generaciones. La configuración tomada fue: $N\acute{u}merodeCajas = [6, 4, 3, 2, 1]$, $Porcentajedeimagenes = [6.2, 12.5, 25, 50, 100]$ y $No.individuos = [10, 10, 8, 8, 8]$. Los resultados los podemos observar en la tabla 14.

Tabla 14. Configuración 2 No.Cajas = [6,4,3,2,1] Porcentaje de imagenes= [6.2,12.5,25,50,100] No.individuos = [10,10,8,8,8]

Clase	Corrida	Aptitud	Generación	
grabados	1	79.38	11	niche
grabados	2	75.12	27	niche
grabados	3	74.14	9	NoNiche
dibujos	1	76.72	12	NoNiche
dibujos	2	76.84	9	NoNiche

5.8. Resultados

5.8.1. Evaluación con la base de datos de Wikiart

Para los experimentos descritos en esta sección, seleccionamos las mejores soluciones (entrenadas con la base de datos de Kaggle) descritas en las tablas 10 y 11 y probamos cada una de ellas con la clase correspondiente de la base de datos de WikiArt.

La Tabla 15 proporciona un resumen de los resultados y una comparación con los resultados obtenidos usando AlexNet. En la Tabla 16 podemos encontrar el tamaño de cada conjunto de datos empleado para todos los experimentos realizados.

Como se mencionó en la sección 5.3.2, fue necesario seleccionar manualmente las imágenes que se utilizarían para que representen a las mismas clases que se usaron para el entrenamiento. Para la clase Dibujos fue más difícil encontrar manualmente las correspondencias, ya que la base de datos de Kaggle incluye imágenes de acuarelas y varios dibujos hechos a mano. Después de una revisión, seleccionamos la categoría "caricatura" de WikiArt, un conjunto de 204 imágenes similares a las de nuestro problema. Usando estos datos obtuvimos un rendimiento del 96,08 %.

Para la clase de grabados, elegimos 695 imágenes en escala de grises seleccionadas de WikiArt, todas realizadas por Albrecht Durer, y se encuentran clasificadas en la categoría Renacimiento. El programa descubierto al entrenar con las imágenes de la base de datos de kaggle, alcanza 87,28 %, que fue similar a la puntuación obtenida durante la validación. En cuanto a las imágenes de prueba para la clase de pintura, se utilizó el conjunto completo de 2089 imágenes de estilo rococó por su similitud con las imágenes que se habían utilizado para entrenar nuestro programa. Además, también se realizó una prueba con el conjunto de imágenes de la categoría paisajes, en particular, las obras de los autores Claude Lorrain, Jan Dirksz Both y Jan Van Goyen. Ambas pruebas obtuvieron una puntuación del 100 %.

Para la clase de iconografía, se utilizaron 251 obras de arte creadas por Ambrogio Lorenzetti, Cennino Cennini, Cimabue, Duccio, Luca di Tommè, Paolo Veneziano, Pietro Lorenzetti y Simone Martini para realizar las pruebas, y nuestro mejor programa descubierto alcanzó el 92,21 %. A pesar de que WikiArt se dedica principalmente a pinturas, la base de datos incluye algunas esculturas, de las cuales seleccionamos 117 imágenes de obras de arte realizadas por Antonio Canova, Constantin Brâncuși, Donatello, Alexander Archipenko, Umberto Boccioni, John Chamberlain y Louise Nevelson que utilizamos para probar el mejor programa, obtuvimos una puntuación de 85,34 %.

Tabla 15. Esta tabla muestra un resumen de los resultados después de evolucionar el mejor conjunto de soluciones mediante el Brain Programming. Incluimos los resultados encontrados mediante redes neuronales profundas para comparación.

	Dibujos	Grabados	Pinturas	Iconografía	Esculturas
Test Wikiart BP	96.08	87.28	100	92.21	85.34
AlexNet Wikiart Test	82.61	92.24	91.82	97.52	89.74

Tabla 16. Numero de imágenes empleadas en cada clase para entrenamiento, validación y prueba.

	Dibujos	Grabados	Pinturas	Iconografía	Esculturas	Fondo Caltech)
Train	554	378	1021	1038	868	233
Validation	554	378	1021	1038	868	233
Test Wikiart	204	695	2089	2089	117	233
Wikiart Landscapes			136			

5.8.2. Robustez de las soluciones generadas

La robustez es una característica importante cuando se desea implementar cualquier sistema de clasificación automático, la importancia se vuelve aún mayor cuando hablamos en términos de seguridad y confianza para aplicaciones del mundo real. Los experimentos llevados a cabo en esta sección, consisten en

medir la robustez en la precisión de clasificación para el problema de la categorización de imágenes de arte frente a ataques adversarios. Realizamos una comparación con un algoritmo tradicional de características extraídas manualmente (SIFT + FV) y los algoritmos que se encuentran en el estado del arte para problemas para clasificación, es decir las redes neuronales convolucionales profundas (DCNN [8]).

Las perturbaciones realizadas por los ataques adversarios afectan no solo a un único modelo, sino también a diferentes arquitecturas que se entrenen para la misma tarea. Queremos verificaar esta transferibilidad entre los modelos ya que evaluamos si el BP o el SIFT + FV se ven afectados a través de los ejemplos adversarios. Se hará la comparación con cuatro arquitecturas: AlexNet, VGG, ResNet18 y ResNet101.

Se utilizaron conjuntos de datos de prueba, validación y entrenamiento no convencionales. Los conjuntos de datos de entrenamiento y validación se construyen a partir de la base de datos Kaggle, mientras que para las pruebas utilizamos una base de datos estándar WikiArt. El objetivo es emular un escenario del mundo real donde los modelos propuestos se prueban con puntos de referencia estándar.

5.8.3. Ataques adversarios empleados

Para corroborar la robustez de las soluciones encontradas mediante el Brain Programming, elegimos tres ataques diferentes: una ataque de tipo white-box (FGSM [9]), un de tipo black-box (ataque de un píxel [10]) y un ataque dirigido (Adversarial Patch), que se explicarán en los siguientes párrafos.

5.8.3.1. FGSM

El FGSM propuesto por Goodfellow *et al.* (2015), es el método más utilizado para calcular ejemplos adversarios a partir de una imagen de entrada, esto debido a su fácil implementación (ver imágenes de ejemplo en la Figura 19). Propone aumentar la función de pérdida del clasificador resolviendo la siguiente ecuación: $\rho = \epsilon \, \text{sign}(\nabla J(\theta, \mathbf{x}, y_l))$, donde $\nabla J()$ calcula el gradiente de la función de costo alrededor del valor actual de los parámetros del modelo θ con respecto a la imagen $\mathbf{x}$ y la etiqueta de destino y_l. sign() denota la función signo, que asegura que la magnitud de la pérdida se maximice y ϵ es un valor escalar pequeño que restringe la norma L_∞ de la perturbación.

[8] por sus siglas en inglés: Deep convolutional neural networks
[9] Por sus siglas en inglés: Fast Gradient Sign Method
[10] One Pixel Attack

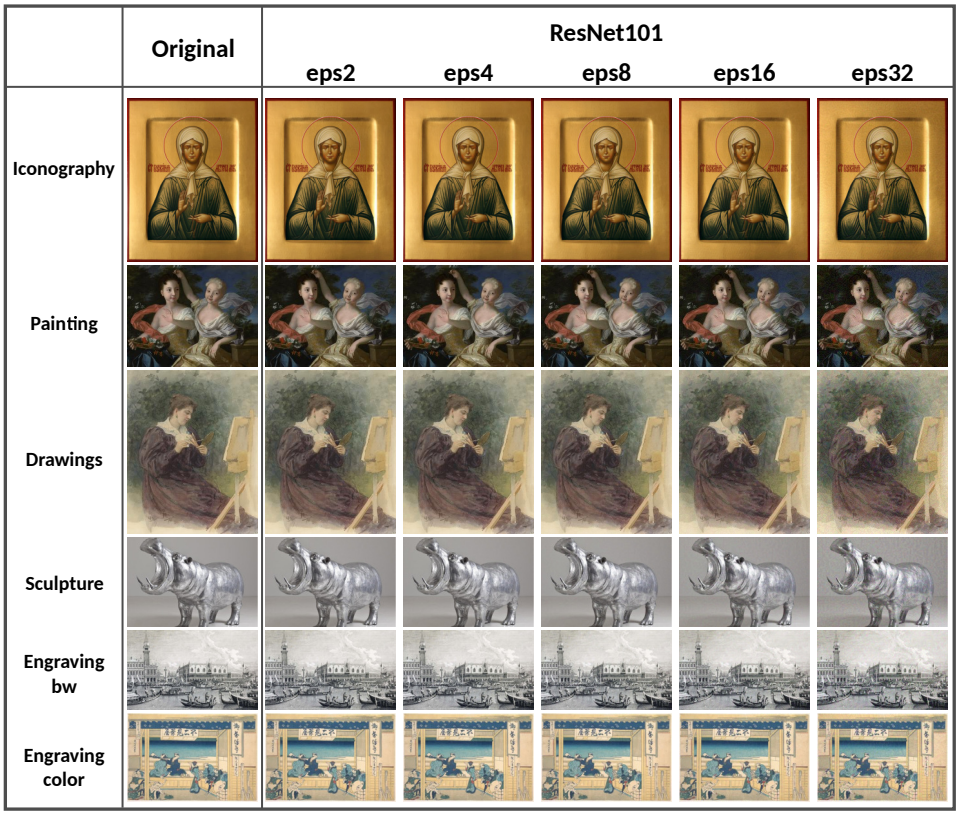

Figura 19. Imágenes de muestra de ejemplos adversarios mediante FGSM dirigido a la red ResNet101 para cada una de las clases estudiadas.

Las perturbaciones generadas con FGSM aprovechan la linealidad de los modelos DCNN en el espacio dimensional superior para hacer que el modelo clasifique erróneamente la imagen. La implicación de la linealidad de los modelos DCNN generados por FSGM es que existe transferibilidad entre modelos. Los autores de Kurakin *et al.* (2017) informaron que con el conjunto de datos de ImageNet, la tasa de error top-1 usando las perturbaciones generadas por FGSM es de alrededor de 63-69 % para $\epsilon \in [2, 32]$.

5.8.3.2. Ataque de múltiples pixeles

El ataque de un píxel consiste en variar los valores RGB de únicamente un pixel en una imagen para engañar a los modelos de DCNN. Este ataque es exclusivamente para imágenes de un tamaño reducido de 32×32 píxeles. Con estas limitaciones, logran engañar a tres modelos diferentes de CNN en el 70,97 % de las imágenes de prueba con la modificación de solo un píxel por imagen Su *et al.* (2019). Además, reportó que la confianza media de las CNN en la predicción errónea de las imágenes era del 97,47 %, es decir, el clasificador estaba convencido de la decisión que había tomado.

Tabla 17. Resultados obtenidos usando los conjuntos de entrenamiento y validación para la base de datos de Kaggle. Cada método presenta su precisión de clasificación para entrenamiento, validación y los ejemplos adversarios usando FGSM .

Iconografía																						
			AlexNet					VGG					ResNet18					ResNet101				
	train	val	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$
BP	92.84	91.42	91.42	91.42	91.42	91.42	91.42	91.42	91.42	91.42	91.42	91.42	91.42	91.42	91.42	91.42	91.42	91.42	91.42	91.42	91.42	91.42
SIFT+FV	99.92	95.91	95.91	95.91	95.91	95.59	94.26	95.83	95.83	96.07	95.52	94.57	95.75	95.75	96.07	95.52	94.34	95.99	95.99	96.07	95.67	94.73
AlexNet	99.61	98.66	96.3	96.3	83.24	52.56	38.39	98.51	98.51	98.43	98.03	97.64	98.58	98.58	98.66	98.03	97.4	98.51	98.51	98.51	98.03	97.48
VGG	100	99.21	99.29	99.29	99.06	98.82	96.85	91.9	91.9	47.05	17.7	16.76	99.21	99.21	98.98	98.74	95.83	99.21	99.21	98.98	98.35	97.32
ResNet18	100	98.9	98.66	98.66	98.66	98.9	97.95	98.66	98.66	98.66	98.03	95.83	90.24	90.24	52.01	29.03	32.1	98.66	98.66	98.43	97.17	95.75
Resnet101	100	99.37	99.21	99.21	99.21	99.06	97.72	99.29	99.29	99.06	98.9	97.01	99.37	99.37	99.21	97.95	95.28	94.34	94.34	67.98	50.04	51.3
Pinturas																						
	train	val	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$
BP	99.68	99.04	98.25	98.25	98.48	98.41	98.48	98.78	98.8	98.64	98.33	98.41	98.8	98.8	98.56	98.64	98.56	98.41	98.41	98.56	98.8	97.69
SIFT+FV	99.76	92.24	92.08	92.08	92.00	89.84	87.84	92.16	92.16	92.08	90.48	88.08	91.92	91.92	91.76	90.08	88.00	92.00	92.00	91.84	89.76	87.60
AlexNet	98.96	97.69	93.46	93.46	83.01	66.99	69.3	97.53	97.53	97.13	96.89	96.41	97.45	97.45	96.89	96.97	96.49	97.45	97.45	97.21	97.05	96.73
VGG	99.92	98.17	97.93	97.93	97.53	96.73	92.82	89.31	89.31	32.14	14.27	14.91	97.69	97.69	97.05	95.45	88.28	97.69	97.69	96.81	95.14	88.12
ResNet18	100	97.85	97.93	97.93	97.93	97.45	96.33	97.77	97.77	97.05	96.33	93.22	86.92	86.92	43.94	31.82	40.75	97.69	97.69	97.13	95.77	92.9
Resnet101	100	98.56	98.72	98.72	98.48	98.17	96.65	98.64	98.64	98.25	96.49	93.86	98.72	98.72	98.17	95.85	92.58	91.15	91.15	55.42	43.94	49.68
Dibujos																						
	train	val	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$
BP	96.56	90.59	90.59	90.59	90.59	90.59	90.59	90.59	90.59	90.59	90.59	90.59	90.59	90.59	90.59	90.59	90.59	90.59	90.59	90.59	90.59	90.59
SIFT+FV	99.87	83.84	83.84	83.84	83.97	83.46	81.30	84.22	84.22	84.48	83.59	81.93	84.22	84.22	84.22	82.95	81.68	84.10	84.10	84.35	82.95	80.79
AlexNet	96.44	91.35	85.75	85.75	66.79	44.91	35.62	90.84	90.84	91.22	90.59	88.55	91.09	91.09	91.09	90.59	89.06	90.71	90.71	91.09	91.09	90.08
VGG	99.75	95.42	95.29	95.29	94.78	93.51	87.02	74.43	74.43	28.75	15.78	14.38	94.78	94.78	93.13	88.68	77.86	94.78	94.78	93.77	90.59	83.46
ResNet18	99.87	94.44	94.27	94.27	93.64	92.37	86.9	93.38	93.38	91.22	86.77	77.48	72.9	72.9	31.04	23.41	22.77	93.64	93.64	92.37	88.17	80.28
Resnet101	99.87	95.8	95.8	95.8	95.42	93.89	89.31	95.55	95.55	93.89	90.84	83.33	95.29	95.29	93.13	88.68	80.79	76.08	76.08	47.96	41.48	38.55
Esculturas																						
	train	val	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$
BP	93.19	93.26	92.79	92.79	92.79	92.79	92.79	92.79	92.79	92.79	92.7	92.79	92.88	92.88	92.79	92.79	92.7	92.88	92.88	92.79	92.88	92.7
SIFT+FV	99.55	87.35	87.44	87.44	85.79	85.15	83.68	87.26	87.26	86.34	85.15	84.42	87.35	87.35	85.98	84.97	85.06	87.44	87.44	85.98	85.24	85.15
AlexNet	99.36	95.78	90.93	90.93	63.24	27.50	14.57	95.78	95.78	95.42	94.68	89.55	95.88	95.88	95.78	94.13	89.09	95.97	95.97	96.06	94.68	90.10
VGG	100	97.62	98.26	98.26	97.89	94.87	78.28	84.69	84.69	37.76	17.87	14.21	98.08	98.08	97.07	91.38	72.59	97.98	97.98	96.98	93.31	78.00
ResNet18	100	96.88	97.25	97.25	96.88	95.05	80.66	96.88	96.88	96.15	92.39	77.54	84.88	84.88	45.92	25.30	19.07	96.70	96.70	95.69	92.58	79.65
Resnet101	100	97.89	98.44	98.44	98.17	96.06	87.08	98.44	98.44	98.08	95.42	84.88	98.35	98.35	96.98	92.30	77.45	89.00	89.00	60.49	44.18	37.86
Grabados BW																						
	train	val	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$
BP	89.76	92.05	92.23	92.23	92.23	91.70	91.87	91.70	91.70	92.06	91.87	91.70	91.70	91.70	92.23	92.05	91.53	91.70	91.70	91.87	91.87	92.05
SIFT+FV	100	93.99	94.35	94.35	94.70	94.17	92.76	94.35	94.35	94.35	94.17	93.64	94.35	94.35	94.52	94.17	93.46	94.35	94.35	94.88	94.35	93.46
AlexNet	99.76	99.29	96.11	96.11	78.62	56.71	47.88	99.12	99.12	99.12	98.94	98.41	99.12	99.12	99.12	98.94	98.06	99.12	99.12	99.12	98.94	98.41
VGG	100	100	99.82	99.82	99.82	99.65	99.29	98.53	97.53	73.14	49.29	47.17	99.82	99.82	99.82	99.82	99.12	99.82	99.82	99.82	99.82	99.29
ResNet18	100	100	100	100	99.82	99.82	98.94	99.82	99.82	99.82	99.65	98.23	95.58	95.58	78.98	64.49	63.07	100	100	100	100	98.41
Resnet101	100	100	100	100	100	99.82	99.47	100	100	99.82	99.82	99.47	100	100	99.65	99.65	98.76	98.94	98.94	94.70	89.75	88.16
Grabados Color																						
	train	val	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$
BP	98.33	97.37	97.37	97.37	97.37	97.37	97.37	97.37	97.37	97.37	97.37	97.37	97.37	97.37	97.37	97.37	97.37	97.37	97.37	97.37	97.37	97.37
SIFT+FV	100	50.00	44.74	44.74	44.74	44.74	50.00	47.37	47.37	47.37	47.37	50.00	47.37	47.37	44.74	47.37	47.37	50.00	50.00	47.37	47.37	50.00
AlexNet	100	100	73.68	73.68	23.68	13.16	15.79	100	100	100	100	94.74	100	100	100	100	94.74	100	100	100	94.74	92.11
VGG	100	100	100	100	100	100	97.37	97.37	97.37	26.32	15.79	13.16	100	100	100	100	100	100	100	100	100	100
ResNet18	95.00	100	97.37	97.37	97.37	97.37	81.58	97.37	97.37	97.37	89.47	81.58	52.63	52.63	13.16	02.63	21.05	97.37	97.37	97.37	94.74	78.95
Resnet101	100	100	100	100	100	100	97.37	100	100	100	100	97.37	100	100	100	97.37	94.74	94.74	94.74	81.58	65.79	68.42

Las perturbaciones realizadas por este método, se basan en un ataque de tipo black-box, en el que no se requiere información sobre el clasificador que se está atacando. Utiliza un algoritmo de optimización llamado Evolución diferencial Das y Suganthan (2010) para resolver problemas complejos de optimización multimodal y generar el ataque. El programa busca una solución en un espacio vectorial $\mathbb{R}^5$ que contiene (x, y) coordenadas correspondientes al tamaño de la imagen y las tres bandas de los valores de color RGB. Dentro de una población, modifica los elementos de los individuos de cinco dimensiones aleatoriamente para crear una nueva descendencia que compita en la iteración actual. En el caso de dos píxeles, un individuo tiene un espacio vectorial $\mathbb{R}^{10}$ que contiene las coordenadas y los valores de colores de ambos píxeles, y así sucesivamente para los individuos con más píxeles. Durante la ejecución, el algoritmo utiliza la probabilidad de la etiqueta predicha para calcular el criterio de aptitud. El último individuo superviviente se utiliza para modificar los píxeles de la imagen.

Tabla 18. Resultados obtenidos al realizar la prueba con la base de datos de Wikiart. Para cada método se presenta su precisión de clasificación para entrenamiento, validación y los ejemplos adversarios usando FGSM.

Iconografía																					
		AlexNet					VGG					ResNet18					ResNet101				
	test	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$
BP	91.74	91.66	91.66	91.82	91.74	91.74	91.66	91.66	91.74	91.74	91.74	91.66	91.66	91.66	91.58	91.5	91.58	91.58	91.58	91.58	91.58
SIFT+FV	86.16	85.54	85.54	84.71	83.26	77.69	85.54	85.54	84.92	83.47	77.48	85.95	85.95	84.71	83.06	76.24	86.16	86.16	84.71	83.06	75.62
AlexNet	96.07	93.39	93.39	70.04	37.4	28.72	95.87	95.87	95.04	94.42	92.98	96.07	96.07	95.87	94.83	93.18	96.07	96.07	95.45	94.63	92.15
VGG	95.87	95.45	95.45	94.83	91.32	80.99	76.65	76.65	36.98	23.97	21.69	95.66	95.66	94.21	87.81	76.86	95.87	95.87	95.87	90.91	82.44
ResNet18	96.49	95.87	95.87	94.83	94.21	87.81	95.66	95.66	94.42	90.5	83.88	76.86	76.86	38.64	25.21	21.49	96.07	96.07	94.21	90.29	85.12
Resnet101	95.25	95.25	95.25	94.83	92.77	89.88	95.45	95.45	94.63	91.94	88.02	95.45	95.45	92.56	87.6	83.26	79.96	79.96	49.38	36.16	36.36
Pinturas																					
	test	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$
BP	100	95.65	95.65	95.65	95.65	95.65	95.65	95.65	95.65	95.65	95.65	95.65	95.65	95.65	95.65	95.65	95.65	95.65	95.65	95.65	95.65
SIFT+FV	94.83	94.83	94.83	94.70	94.57	93.63	94.92	94.92	94.88	94.57	93.63	94.88	94.88	94.79	94.44	93.28	94.88	94.88	94.70	94.32	93.20
AlexNet	94.06	90.57	90.57	64.64	41.04	41.00	94.10	94.10	93.90	94.01	94.92	94.10	94.10	94.06	94.32	95.35	94.10	94.10	94.06	94.06	95.00
VGG	93.37	93.28	93.28	92.64	87.47	60.12	61.15	61.15	13.14	10.42	10.68	92.89	92.89	91.17	80.10	47.55	92.59	92.59	90.78	81.05	44.96
ResNet18	94.23	94.19	94.19	94.40	94.40	92.64	94.01	94.01	93.63	91.30	81.91	64.86	64.86	15.25	13.01	15.07	93.80	93.80	92.72	89.84	80.19
Resnet101	95.91	95.82	95.82	95.78	94.62	90.09	95.82	95.82	95.69	90.44	79.03	95.61	95.61	94.66	88.33	73.47	75.24	75.24	30.62	19.04	19.98
Pinturas Landscapes																					
	test	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$
BP	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
SIFT+FV	75.34	75.07	75.07	72.90	70.46	62.6	75.61	75.61	73.71	70.46	62.60	75.34	75.34	73.17	68.83	60.70	75.34	75.34	72.90	68.29	59.62
AlexNet	93.77	86.99	86.99	61.25	41.46	35.77	93.50	93.50	92.68	91.06	90.24	93.50	93.50	92.68	91.60	90.51	93.77	93.77	92.68	91.33	90.51
VGG	94.58	94.58	94.58	94.31	90.79	73.71	80.76	80.76	42.01	28.73	30.89	94.31	94.31	94.31	88.08	72.63	94.04	94.04	93.22	87.53	70.19
ResNet18	95.12	94.85	94.85	93.77	92.95	90.79	94.31	94.31	92.41	89.43	81.84	72.36	72.36	42.82	33.60	38.48	94.31	94.31	91.6	88.62	79.95
Resnet101	95.39	95.12	95.12	95.12	93.77	89.43	94.58	94.58	94.58	93.50	83.74	94.31	94.31	92.95	88.89	78.86	80.76	80.76	50.96	43.90	44.72
Dibujos																					
	test	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$
BP	94.05	94.28	94.28	93.59	93.81	94.5	93.82	93.82	94.05	93.59	94.73	93.81	93.81	93.59	93.59	94.05	93.81	93.81	93.59	93.59	93.81
SIFT+FV	73.61	68.42	68.42	66.36	62.7	56.52	68.42	68.42	67.51	62.7	57.89	68.42	68.42	66.82	62.01	56.75	68.19	68.19	67.28	62.01	56.75
AlexNet	86.73	77.8	77.8	57.21	41.19	32.72	85.81	85.81	86.27	84.67	79.18	85.81	85.81	85.35	83.75	80.09	85.81	85.81	85.58	84.67	81.69
VGG	91.99	91.99	91.99	90.89	88.79	80.78	72.77	72.77	35.7	20.59	18.99	91.3	91.3	89.02	83.3	73.91	91.53	91.53	89.7	83.98	76.89
ResNet18	90.85	90.85	90.85	89.7	88.79	81.46	90.39	90.39	87.41	81.69	74.37	71.85	71.85	36.16	24.49	24.49	90.16	90.16	86.96	81.92	74.83
Resnet101	93.59	93.36	93.36	93.14	91.53	85.81	93.14	93.14	90.62	85.58	76.43	93.14	93.14	90.16	83.07	75.29	72.27	72.27	45.54	35.24	33.41
Esculturas																					
	test	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$
BP	90.54	90.83	90.83	90.83	90.83	90.83	85.96	85.96	85.96	85.96	85.96	90.83	90.83	90.83	90.83	90.83	90.83	90.83	90.83	90.83	90.83
SIFT+FV	60.47	52.80	52.80	52.80	53.10	51.62	53.39	53.39	53.10	52.51	52.51	52.80	52.80	52.51	52.51	51.33	53.39	53.39	52.51	52.51	50.74
AlexNet	91.45	87.61	87.61	65.49	44.25	36.87	91.15	91.15	90.56	89.38	87.32	91.45	91.45	91.45	89.09	89.38	91.45	91.45	91.45	90.27	88.20
VGG	94.69	94.99	94.99	94.99	92.33	84.37	79.06	79.06	45.43	32.74	34.51	95.28	95.28	94.10	88.20	82.01	94.69	94.69	93.81	91.74	86.73
ResNet18	92.63	91.74	91.74	90.86	89.38	83.19	91.74	91.74	87.91	84.96	80.24	75.81	75.81	46.61	34.81	33.92	91.15	91.15	89.38	86.14	83.19
Resnet101	92.92	93.22	93.22	92.63	90.86	87.61	92.92	92.92	92.92	89.97	86.14	93.22	93.22	91.15	88.20	83.48	80.53	80.53	56.64	50.44	56.34
Grabados BW																					
	test	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$
BP	91.55	92.64	92.64	91.97	91.72	91.63	92.30	92.30	92.05	92.05	91.63	91.97	91.97	91.80	91.97	91.80	92.13	92.13	91.97	91.80	91.63
SIFT+FV	89.79	89.79	89.79	89.71	89.87	90.96	89.79	89.79	89.79	89.37	90.96	89.87	89.87	89.62	89.54	90.88	89.96	89.96	89.46	89.54	90.96
AlexNet	98.58	94.06	94.06	75.06	57.32	54.64	98.66	98.66	98.66	98.49	97.32	98.66	98.66	98.66	98.33	97.15	98.66	98.66	98.66	98.58	97.49
VGG	99.58	99.83	99.83	99.67	99.50	99.16	91.05	91.05	62.85	45.94	49.87	99.58	99.58	98.74	98.41	97.91	99.58	99.58	99.25	99.00	98.83
ResNet18	99.83	99.92	99.92	99.83	99.67	99.16	99.75	99.75	99.41	98.83	97.49	93.22	93.22	71.55	59.41	61.09	99.83	99.83	99.67	98.91	97.82
Resnet101	99.67	99.75	99.75	99.75	99.83	99.75	99.83	99.83	99.50	99.25	98.74	99.67	99.67	99.50	99.08	98.16	95.90	95.90	90.13	85.77	83.01
Grabados Color																					
	test	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$	$\epsilon 2$	$\epsilon 4$	$\epsilon 8$	$\epsilon 16$	$\epsilon 32$
BP	89.92	89.68	89.68	89.74	89.86	89.80	89.92	89.92	89.74	89.86	89.62	89.68	89.68	89.74	89.98	89.80	89.92	89.92	89.86	89.50	90.16
SIFT+FV	66.95	66.77	66.77	66.59	66.89	68.09	66.83	66.83	66.59	67.19	68.09	66.89	66.65	66.95	68.33	66.71	66.71	66.53	66.53	66.95	66.95
AlexNet	94.72	73.55	73.55	25.49	12.30	17.22	94.78	94.78	94.90	94.48	93.64	94.72	94.72	94.66	95.14	94.24	94.54	94.54	95.02	94.66	94.00
VGG	99.40	99.46	99.46	99.46	99.28	96.52	79.90	79.90	16.02	05.46	06.06	99.52	99.52	99.22	99.10	97.18	99.40	99.40	99.10	98.50	95.98
ResNet18	96.40	95.98	95.98	96.16	95.02	89.14	95.92	95.92	95.50	93.88	89.50	49.13	49.13	06.84	05.58	10.74	95.62	95.62	95.02	92.68	86.98
Resnet101	99.88	99.76	99.76	99.76	99.52	98.92	99.70	99.70	99.70	99.22	98.44	99.82	99.82	99.76	99.40	98.56	92.86	92.86	61.91	49.19	54.53

En resumen, sea el vector $\mathbf{x} = (x_1, \ldots, x_n)$ una imagen n-dimensional la entrada del clasificador objetivo$f$. Es un clasificador entrenado que predice correctamente la clase t de la imagen. La probabilidad de $\mathbf{x}$ asociada a la clase t es $f_l(\mathbf{x})$. El objetivo es construir un vector de perturbación aditivo $e(\mathbf{x}) = (e_1, \ldots, e_n)$ de acuerdo con $\mathbf{x}$, la clase *target* y un valor que limita las dimensiones de la modificación d, un pequeño número que expresa las dimensiones que se modifican mientras que otras dimensiones de $e(\mathbf{x})$ quedan como ceros. El propósito principal es encontrar la solución óptima $e(\mathbf{x})*$ que resuelva la siguiente ecuación:

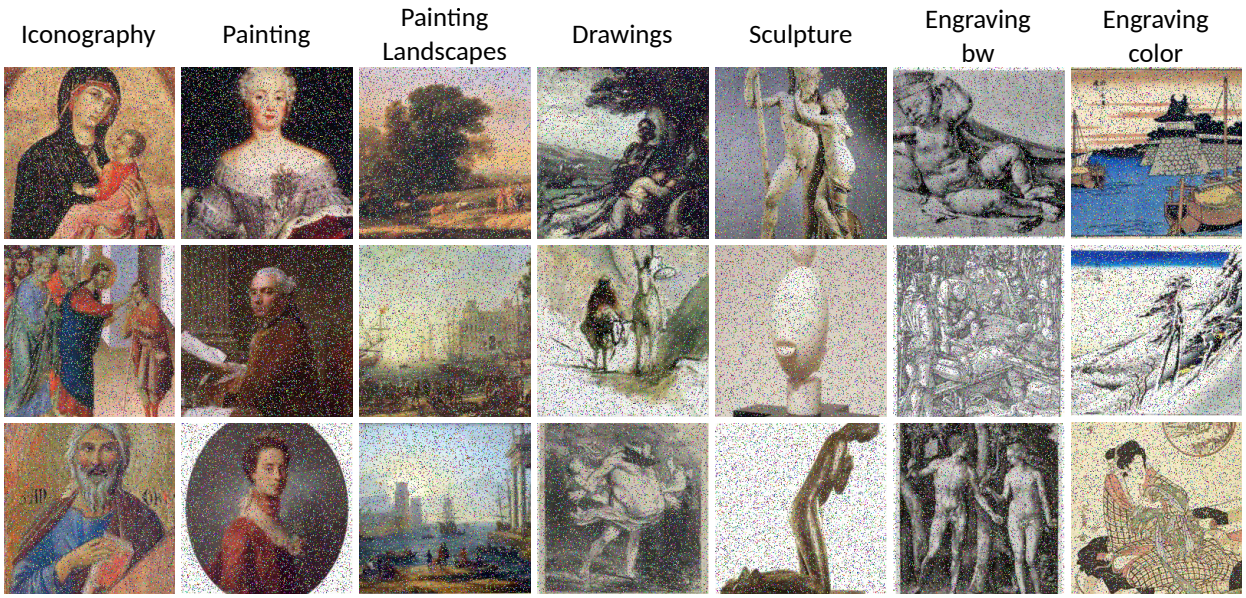

Figura 20. Imágenes de ejemplo del ataque de múltiples pixeles usando $d = 10,000$ para cada clase. Cada columna muestra tres ejemplos de imágenes por cada clase de la base de datos de Wikiart.

$$\begin{aligned} \underset{e(\mathbf{x})*}{\text{máx}} \quad & f_{target}(\mathbf{x} + e(\mathbf{x})) \\ \text{s.t.} \quad & ||e(\mathbf{x})||_0 \leq d \quad . \end{aligned} \tag{5}$$

Entonces, el ataque e un pixel tiene $d = 1$, pero puede ser extendido a múltiples pixeles incrementando d. Es importante notar que el ataque de un pixel fue hecho para atacar modelos de aprendizaje profundo que tenían como entrada las imágenes de la base de datos de CIFAR10. Así que, un cambio en un pixel, si representa una modificación importante, considerando el tamaño tan pequeño de la imágenes. Sin embargo, para nuestro estudio, un pixel resulta insignificante ya que utilizamos imágenes mucho más grandes. Es por eso que para evaluar la robustez de nuestro algoritmo, implementamos un ataque de múltiples pixeles $(d >> 1)$ para poder trabajar con las imágenes en el tamaño original.

Debe tenerse en cuenta que aumentar el número de pixeles en este ataque aumentará el riesgo de que la perturbación sea más notoria (Ver imágenes de ejemplo en la Figura 20).

5.8.3.3. Parche adversario

La estrategia general para crear un ejemplo adversario consiste encontrar una perturbación máxima $e(\mathbf{x})$ que maximice el $f_{target}(\mathbf{x}+e(\mathbf{x}))$. En contraste, el Parche Adversario es un método que reemplaza una perturbación en toda la imagen con un parche (Ver Figura 21). La versatilidad de estos parches

Tabla 19. Resultados de los experimentos para el ataque de múltiples pixeles con $d = 10,000$ en 100 imágenes aleatorias del conjunto de prueba. La precisión original se refiere a la precisión obtenida por el clasificador en las imágenes sin ataque. Taza de éxito se refiere al porcentaje de imágenes en las que el clasificador cambió de decisión después del ataque con un valor de Confianza de la probabilidad posterior sobre la nueva predicción.

Iconografía	BP	SIFT+FV	AlexNet	VGG	ResNet18	ResNet101
Precisión Original.	92.00	88.00	96.00	94.00	96.00	92.00
Taza de éxito	0.00	32.00	32.00	44.00	46.00	42.00
Confianza	NA	64.96	85.09	85.72	76.34	77.61
Tiempo (segundos)	94.22	301.21	138.51	147.72	152.37	237.73
Pinturas	BP	SIFT+FV	AlexNet	VGG	ResNet18	ResNet101
Precisión Original.	100	78.00	94.00	90.00	92.00	94.00
Taza de éxito	2.00	0.00	54.00	60.00	64.00	64.00
Confianza	51.83	NA	78.11	97.34	99.37	98.06
Tiempo (segundos)	90.16	598.12	119.78	122.59	111.14	242.58
Pinturas Landscapes	BP	SIFT+FV	AlexNet	VGG	ResNet18	ResNet101
Precisión Original.	100	78.00	88.00	88.00	92.00	92.00
Taza de éxito	2.00	40.00	54.00	60.00	64.00	66.00
Confianza	54.06	62.04	75.70	97.25	99.26	97.37
Tiempo (segundos)	98.83	585.69	141.85	163.51	143.62	205.53
Dibujos	BP	SIFT+FV	AlexNet	VGG	ResNet18	ResNet101
Precisión Original.	88.00	70.00	80.00	90.00	86.00	92.00
Taza de éxito	0.00	38.00	68.00	68.00	74.00	78.00
Confianza	NA	66.53	83.91	91.94	95.11	94.24
Tiempo (segundos)	118.85	462.92	110.18	111.48	128.07	220.69
Esculturas	BP	SIFT+FV	AlexNet	VGG	ResNet18	ResNet101
Precisión Original.	86.00	62.00	88.00	98.00	96.00	96.00
Taza de éxito	4.00	60.00	62.00	54.00	56.00	54.00
Confianza	58.14	67.61	92.65	98.60	97.45	96.93
Tiempo (segundos)	71.20	601.53	121.22	130.06	137.16	181.14
Grabados BW	BP	SIFT+FV	AlexNet	VGG	ResNet18	ResNet101
Precisión Original.	94.00	94.00	100	100	100	100
Taza de éxito	0.00	0.00	40.00	50.00	32.00	20.00
Confianza	NA	NA	77.63	68.07	71.86	61.25
Tiempo (segundos)	88.71	599.41	148.90	169.56	152.11	177.61
Grabados Color	BP	SIFT+FV	AlexNet	VGG	ResNet18	ResNet101
Precisión Original.	94.00	74.00	98.00	100	92.00	100
Taza de éxito	0.00	60.00	40.00	50.00	46.00	22.00
Confianza	NA	55.98	73.80	66.15	62.96	65.31
Tiempo (segundos)	87.01	600.82	150.70	174.51	154.52	186.13

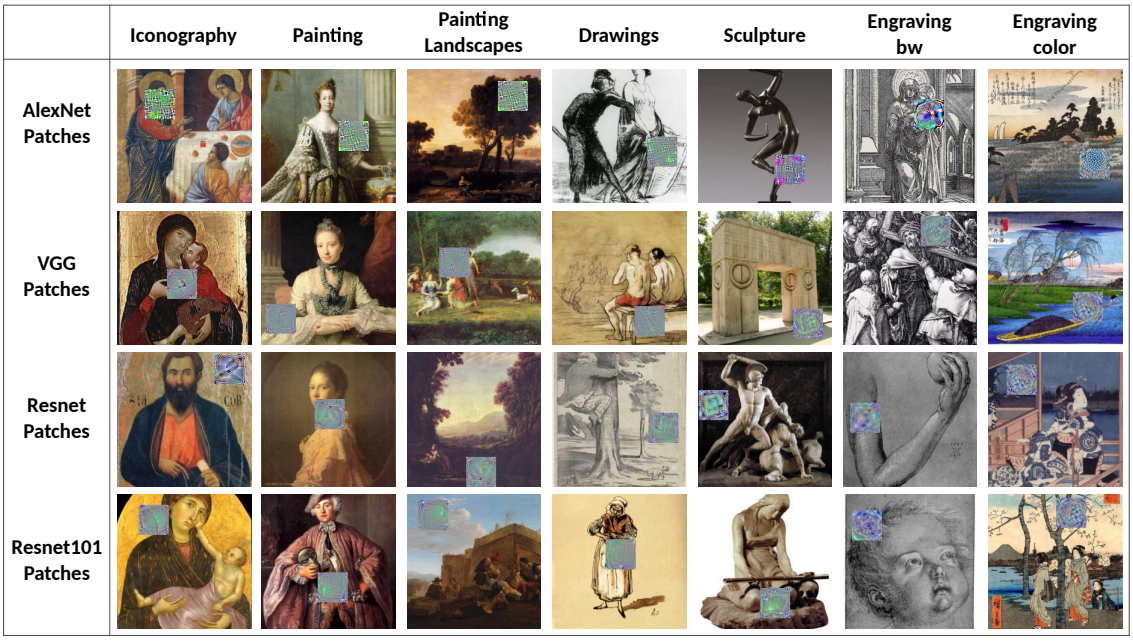

Figura 21. Imágenes de ejemplo del parche adversario. Cada columna representa las clases usadas de la base de datos de Wikiart, cada fila representa el parche correspondiente del modelo de DCNN

reside en la amplia variedad de transformaciones en las que pueden atacar cualquier imagen y dirigir la predicción del clasificador hacia otra clase deseada. Además, funcionan en entornos de trabajo reales donde se pueden imprimir, fotografiar o incluso cuando el parche es demasiado pequeño; es capaz de engañar a un clasificador haciendo que ignoren toda la escena para predecir la clase objetivo.

Para construir el parche $\hat{p}$, se utilizó una variante de Expectation over Transformation (EOT), en el que se entrena el parche para optimizar la siguiente ecuación:

$$\hat{p} = _{\hat{p}} \mathbb{E}_{x \in X, t \in T, l \in L} [\log f(y, A(p, \mathbf{x}, l, t))] \quad , \tag{6}$$

donde X es un conjunto de imágenes de entrenamiento, T es una distribución sobre las transformaciones del parche, L es una distribución sobre ubicaciones en la imagen y $(y, \mathbf{x})$ son la etiqueta y el vector de imagen respectivamente. La expectativa sobre las imágenes de entrenamiento mejora la efectividad del parche, independientemente de lo que haya en el fondo. Brown *et al.* (2017) demostraron la universalidad del parche usando varias imágenes con diferentes fondos. Una variación de este método es agregar una restricción de la forma $||p - p_{orig}||_\infty < \epsilon$ al objetivo del parche para camuflarlo. La restricción obliga al parche final a estar dentro de ϵ en la norma L_∞ de algún parche inicial p_{orig}.

Tabla 20. Resultados obtenidos usando el parche adversario. Cada columna muestra el resultado obtenido para las 100 imágenes originales por clase y el ejemplo adversario cuando se le agrega el parche.

Iconografía	Precisión Original.	Parche AlexNet	Parche VGG	Parche ResNet18	Parche ResNet101
BP	99.00	99.00	99.00	99.00	99.00
SIFT+FV	92.00	89.00	93.00	93.00	92.00
AlexNet	98.00	74.00	97.00	97.00	98.00
VGG	94.00	91.00	45.00	82.00	81.00
ResNet18	94.00	87.00	90.00	58.00	90.00
ResNet101	93.00	87.00	87.00	78.00	70.00
Pinturas	Precisión Original.	Parche AlexNet	Parche VGG	Parche ResNet18	Parche ResNet101
BP	100.00	100.00	100.00	99.00	100.00
SIFT+FV	97.00	98.00	97.00	98.00	96.00
AlexNet	96.00	54.00	94.00	94.00	94.00
VGG	92.00	71.00	48.00	73.00	61.00
ResNet18	94.00	67.00	76.00	23.00	43.00
ResNet101	97.00	72.00	72.00	69.00	56.00
Pinturas Land.	Precisión Original.	Parche AlexNet	Parche VGG	Parche ResNet18	Parche ResNet101
BP	100.00	100.00	100.00	100.00	100.00
SIFT+FV	87.00	81.00	78.00	84.00	81.00
AlexNet	94.00	24.00	85.00	86.00	77.00
VGG	95.00	41.00	19.00	48.00	23.00
ResNet18	95.00	22.00	39.00	0.00	9.00
ResNet101	96.00	43.00	41.00	35.00	22.00
Dibujos	Precisión Original.	Parche AlexNet	Parche VGG	Parche ResNet18	Parche ResNet101
BP	91.00	91.00	91.00	91.00	91.00
SIFT+FV	72.00	67.00	68.00	69.00	67.00
AlexNet	94.00	30.00	85.00	80.00	73.00
VGG	98.00	81.00	69.00	74.00	62.00
ResNet18	96.00	82.00	91.00	66.00	79.00
ResNet101	99.00	88.00	90.00	85.00	75.00
Esculturas	Precisión Original.	Parche AlexNet	Parche VGG	Parche ResNet18	Parche ResNet101
BP	85.00	85.00	85.00	85.00	85.00
SIFT+FV	95.00	92.00	94.00	94.00	95.00
AlexNet	97.00	32.00	92.00	89.00	86.00
VGG	97.00	93.00	72.00	85.00	85.00
ResNet18	95.00	92.00	86.00	66.00	89.00
ResNet101	94.00	87.00	89.00	86.00	87.00
Grabados BW	Precisión Original.	Parche AlexNet	Parche VGG	Parche ResNet18	Parche ResNet101
BP	90.00	90.00	91.00	91.00	91.00
SIFT+FV	91.00	94.00	93.00	95.00	92.00
AlexNet	100.00	99.00	100.00	100.00	100.00
VGG	100.00	99.00	83.00	96.00	97.00
ResNet18	100.00	100.00	96.00	71.00	96.00
ResNet101	100.00	100.00	100.00	100.00	100.00
Grabados Color	Precisión Original.	Parche AlexNet	Parche VGG	Parche ResNet18	Parche ResNet101
BP	93.00	92.00	93.00	92.00	92.00
SIFT+FV	94.00	94.00	96.00	95.00	95.00
AlexNet	97.00	67.00	98.00	95.00	93.00
VGG	100.00	100.00	99.00	100.00	100.00
ResNet18	98.00	99.00	100.00	98.00	99.00
ResNet101	100.00	100.00	100.00	100.00	100.00

5.8.4. Implementación

En esta subsección, describimos los detalles de implementación para todos los modelos que fueron entrenados para realizar las pruebas:

DCNN: para la implementación de los cuatro modelos (AlexNet, VGG, ResNet18 y ResNet101) utilizamos los modelos previamente entrenados de PyTorch v1.1. Estos modelos fueron reentrenados utilizando la transferencia de aprendizaje [11] para el problema de los clasificación de imágenes de arte.

SIFT + FV: se implementó en Matlab usando las librerías de VLFeat para la el SIFT, GMM y Fisher vectors. Se utilizó el SVM proporcionado por Matlab para la clasificación.

Además, describimos cada uno de los ataques adversarios aplicados:

FGSM:se implementó en PyTorch v1.1 usando los conjuntos de datos de validación y prueba para calcular el ejemplo adversario con valores estándar para la escala $\epsilon = 2, 4, 8, 16, 32$ para todos los modelos DCNN.

Ataque de múltiples pixeles: se implementó utilizando 100 imágenes aleatorias del conjunto de datos de prueba (50 de cada clase) en Matlab y Python. La versión de Python se programó usando evolución diferencial con la librería Pygmo, y la versión de Matlab usó la librería de evolución diferencial disponible en el sitio web propio de Matlab. Ambas implementaciones utilizaron la misma configuración de 50 individuos, 30 generaciones, una probabilidad de cruce de 0.9 y $d = 10,000$ píxeles.

Adversarial Patch: se implementó usando 100 imágenes del conjunto de datos de entrenamiento para cada modelo DCNN en PyTorch v1.1 con los siguientes parámetros configurados para construir el parche: tamaño del parche de 50×50 píxeles, un máximo de 100 iteraciones por imagen con un criterio de parada de 0,9 de probabilidad posterior de la clase objetivo. Como trabajamos con el problema de clasificación binaria, elegimos la clase de fondo como la predicción de destino para medir el número de imágenes de clase en las que el clasificador cambia de decisión.

5.8.5. Resultados

Los resultados obtenidos de los experimentos mencionados anteriormente se presentan y discuten a continuación.

[11] del inglés transfer learning

5.8.5.1. FGSM

En la Tabla 17, presentamos los resultados de los conjuntos de datos de entrenamiento y validación de Kaggle junto con los ejemplos adversarios calculados usando FGSM para todos los modelos DCNN. Se reporta la precisión de la clasificación en cada etapa del entrenamiento y la validación junto con la precisión de todos los modelos probados con los ejemplos adversarios.

Lo que se pretende, es medir la influencia en la predicción de la FGSM de dos maneras: 1) directa, ya que conocemos los parámetros y la perturbación del modelo, y 2) indirecta, a través de la transferibilidad del ataque. Anteriormente, otros investigadores informaron que los ejemplos adversarios podrían afectar diferentes modelos de CNN al configurarlos para la misma tarea. Se eligió este ataque para comprobar la robustez del Brain Programming, además se incluyen resultados de SIFT + FV que también podrían verse afectadas por estas sutiles perturbaciones en las obras de arte.

Se puede observar en la Tabla 17 que tan drásticamente se puede reducir el rendimiento de DCNN. El peor de los casos fue la clase de escultura, el rendimiento de VGG pasó de 97,62 % a 14,38 %, AlexNet bajó de 95,78 % a 14,57 %, ResNet18 bajó de 96,88 % a 19,07 % y ResNet101 bajó de 97,89 % hasta 37,86 %. Además, se percibe que el efecto de transferibilidad entre los modelos DCNN es más significativo en $\epsilon = 32$. La clase de dibujos presenta casi el mismo comportamiento que la clase de escultura, donde las otras redes se ven afectadas por el ejemplo adversario. Para todas las demás clases, el efecto es imperceptible, pero la precisión se ve significativamente afectada cuando el modelo coincide con el ejemplo adversario.

En algunos casos, SIFT + FV se vio afectado por FGSM. Por ejemplo, en la clase de dibujo, el rendimiento se redujo en casi un 8 %. Y para la clase pintura, la precisión se redujo aproximadamente en un 4 %. Este resultado muestra una transferibilidad parcial del ejemplo adversario a SIFT + FV porque independientemente de la aplicación de DCNN, la perturbación comprometió el rendimiento de estas dos clases. Sin embargo, BP mantiene su desempeño en casi todas las pruebas; la variación de precisión a lo largo de todo el análisis fue inferior al 2 %.

En las Figura 24, presentamos de manera gráfica los resultados de la Tabla 17. Calculamos la relación entre la precisión de los ejemplos adversarios y la precisión de las imágenes sin ataque. Observamos que la variación de BP es muy leve en comparación con los modelos SIFT + FV y DCNN. Además, se puede notar que el rendimiento de la DCNN se puede reducir en casi todas las clases hasta menos del 20 % de su precisión original cuando la perturbación coincide con el diseño de la red. En caso de que

no coincidan, la perturbación también reduce la precisión hasta un 20 % del rendimiento original de las imágenes limpias para las clases escultura, grabado en blanco y negro y grabado en color.

La etapa de prueba presenta prácticamente el mismo comportamiento, pero fue más notable la caída en el rendimiento, así como la transferibilidad de los ejemplos adversarios a otros modelos DCNN cuando el factor de escala ϵ aumenta. En la Tabla 18, notamos que hay algunas clases como Pintura, Dibujos y grabados de color que redujeron el rendimiento de todos los modelos DCNN de forma significativa. Por ejemplo, grabados de color fue el peor de los casos, en donde AlexNet cayó al 17.22 % desde el 94.72 % de precisión con imágenes sin ataque, VGG y ResNet18 disminuyeron su rendimiento a casi el 5 % de precisión desde aproximadamente el 99 % y el 96 % respectivamente. ResNet101 fue el menos afectado con un 49 % de eficiencia al realizar pruebas con sus ejemplos adversarios. Además, se percibió el efecto de transferibilidad entre los modelos para $\epsilon = 32$.

Con esta prueba también podemos notar el rendimiento limitado de SIFT + FV. En cuatro de las siete clases (Pinturas de paisajes, Dibujos, Escultura y Grabado en color), su desempeño está muy por debajo para competir con DCNN. Además, se vio afectado por ejemplos adversarios en Iconografía, Pintura de paisajes y Escultura, donde se redujo en aproximadamente un 10 % de su puntuación original. Finalmente, el Brain Programming demostró la misma robustez que el experimento anterior con el conjunto de datos de validación.

Es importante hacer notar que, a diferencia de SIFT + FV, la precisión del BP es más comparable a la precisión de los modelos DCNN. En las Figuras 25 y 26 se puede observar la relación de precisión ante los ejemplos adversarios para la fase de prueba. Observamos un comportamiento muy similar, al menos para BP, cuya tasa para todos los experimentos se mantiene casi en uno. Cuando la perturbación coincide con la red, podemos observar la caída drástica en el rendimiento de los modelos DCNN, pero también la influencia de los ejemplos adversarios en los otros modelos, así como SIFT + FV en algunos casos.

5.8.5.2. Ataque de múltiples pixeles

Como se mencionó anteriormente, el ataque de múltiples pixeles fue una adaptación realizada al Ataque de un pixel para imágenes de alta resolución. Ya que un solo pixel no perturba este tipo de imágenes. Por lo tanto, descubrimos experimentalmente que cuando hay entre 8000 y 10,000 pixeles editados, los modelos DCNN pueden ser engañados para que cambien su predicción, por lo que establecimos un segundo experimento con un ataque de 10,000 píxeles. Presentamos en la Tabla 19 el número

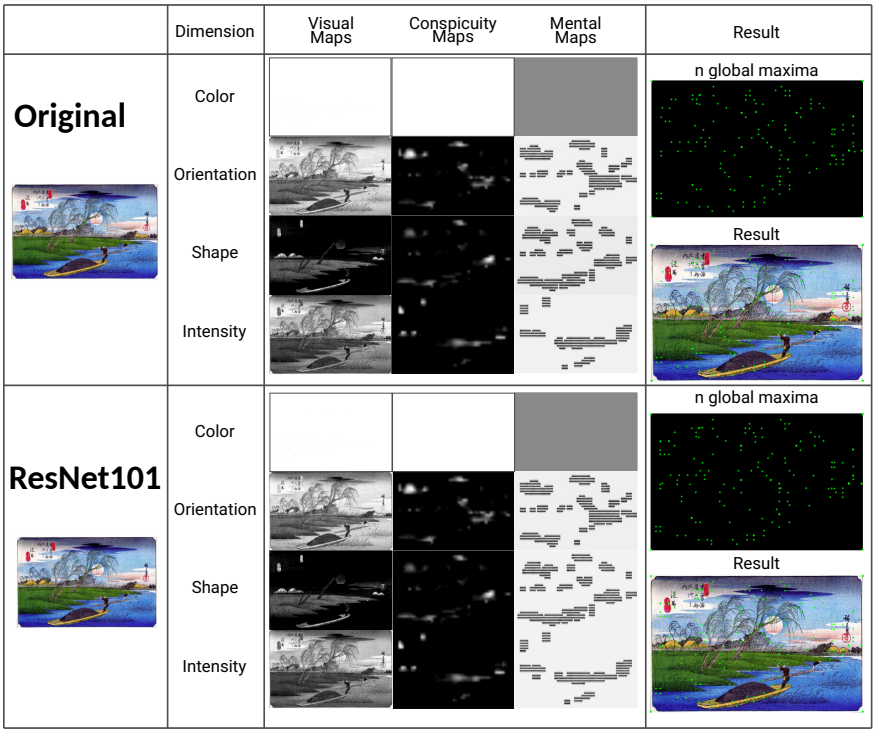

Figura 22. Mapas generados en cada una de las etapas del AVC, extraídas usando la imagen original limpia y otra usando el parche adversario. Nótese que, a pesar del ataque, los mapas generados no cambian mucho en relación con los mapas generados con la imagen sin ataque.

de imágenes que cambian su pronóstico, lo medimos con la tasa de éxito y la probabilidad posterior media de estas nuevas predicciones en la fila de confianza.

Observamos que DCNN cambia en una cantidad considerable de sus predicciones con alta confianza al modificar múltiples píxeles. SIFT + FV también fue engañado en cinco de las siete clases que lograron el mismo número de imágenes que los modelos DCNN con menor confianza. De esta forma, solo dos categorías resistieron el ataque. Por el contrario, BP fue robusto a este ataque teniendo cuatro de las siete clases sin cambios y el resto con un error máximo del 4 %. Nótese que la cantidad de píxeles modificados en este experimento contradice la motivación principal de los ataques adversarios, en el sentido de que las perturbaciones realizadas deberían ser imperceptibles para la visión humana. Aún así, el Brain Programming fue robusto ante esta perturbación. Además, informamos el tiempo medio de procesamiento en segundos (ver Tabla 19), se puede observar que demora bastante, lo que hace que este ataque no sea factible de realizar en aplicaciones en tiempo real.

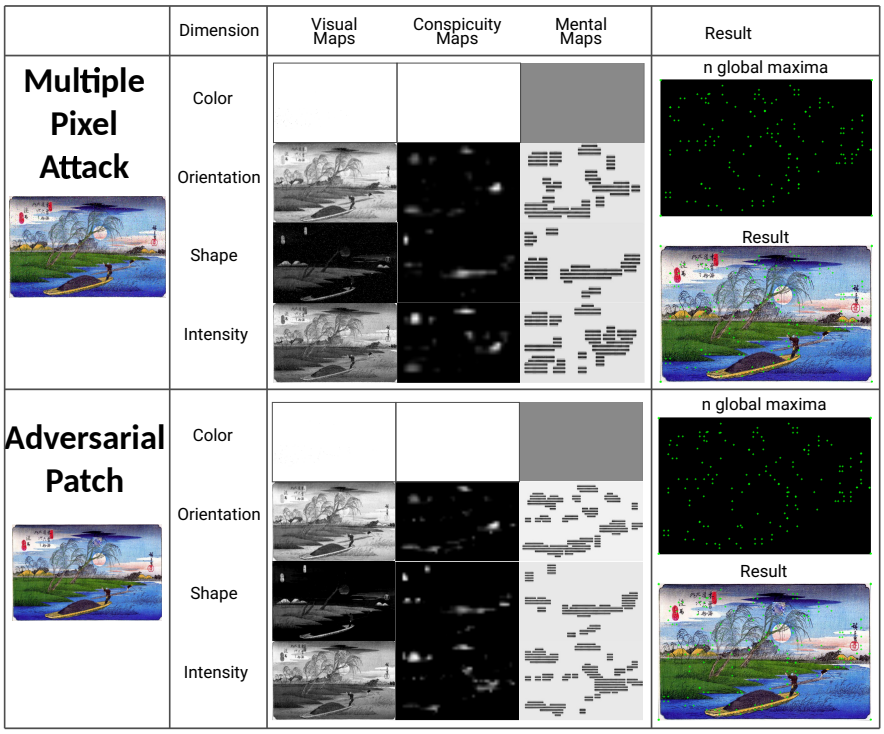

Figura 23. Mapas generados en cada una de las etapas del AVC, extraídas usando una imagen previamente atacada con el ataque de múltiples pixeles y otra usando el parche adversario. Nótese que a pesar del ataque, los mapas generados no cambian mucho en relación con los mapas generados con la imagen sin ataque.

5.8.5.3. Parche adversario

Se presentan los resultados del parche adversario en la Tabla 20. En este experimento se analiza el cambio en las predicciones del modelo agregando los parches entrenados en los diferentes modelos DCNN; se utilizaron 100 imágenes de cada clase en una ubicación y orientación aleatorias. Los resultados de la Tabla 20 muestran que estos parches afectan de manera significativa a los modelos DCNN en la mayoría de los experimentos. También, al observar los resultados, podemos ver que los parches de un modelo afectaron el desempeño de otras redes, por lo que podemos concluir que los ataques mediante parches pueden transferirse a otros modelos DCNN.

El experimento con la clase Pinturas, en particular con los paisajes mostró el peor desempeño para los modelos DCNN, en el que observamos un efecto de transferibilidad considerable entre los modelos. Observamos que VGG, ResNet18 y ResNet101 se vieron afectados por todos los parches. Los modelos DCNN redujeron su rendimiento a aproximadamente la mitad de su precisión original y, en algunos casos, el resultado fue inferior al 50 %. ResNet18 fue engañado en todas las imágenes usando su parche respectivo. En contraste, SIFT + FV y BP demostraron un desempeño robusto ante los parches adversarios, mostrando un comportamiento casi inalterable.

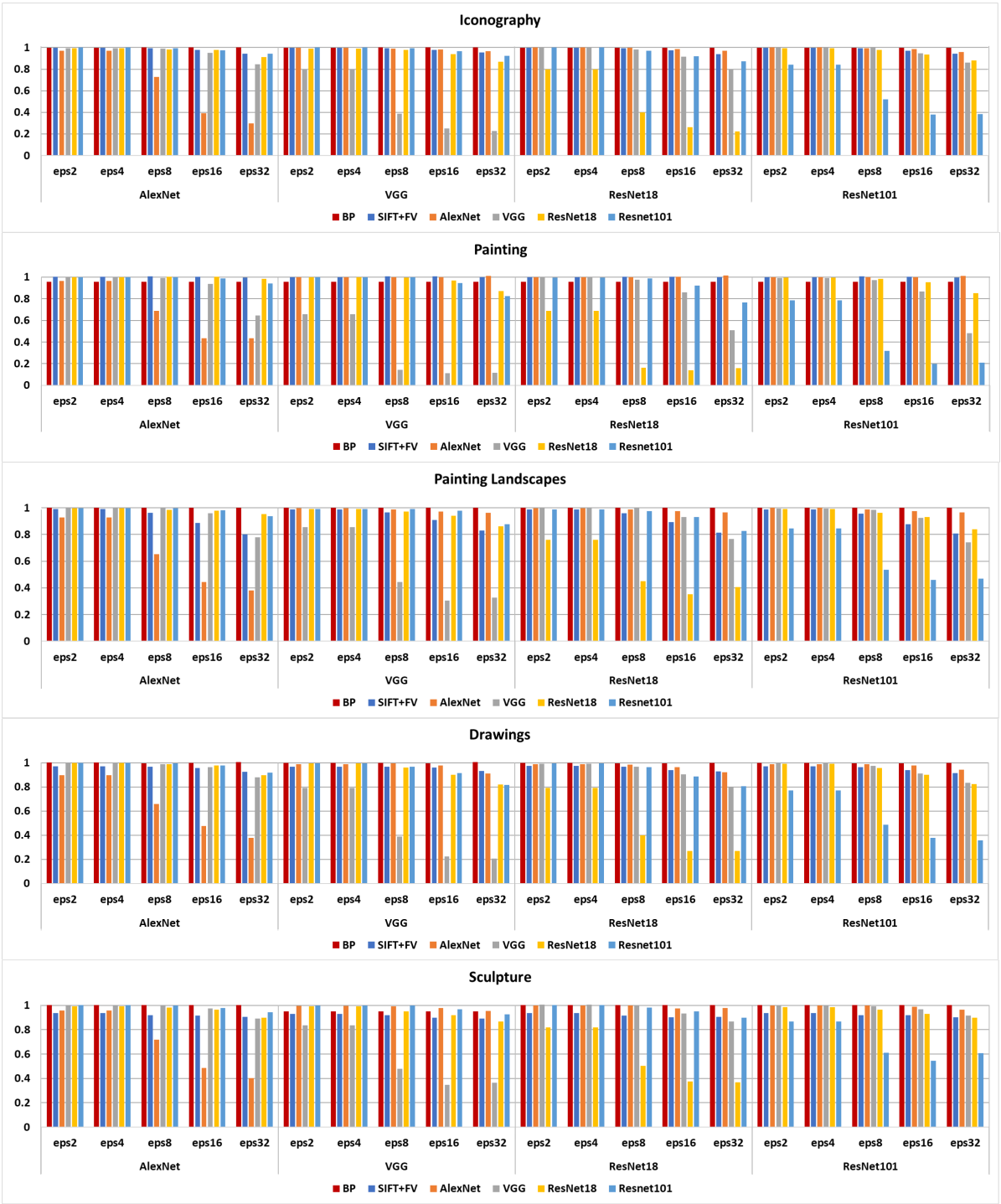

Figura 24. Gráfico comparativo de la precisión calculada entre las imágenes con ataque y las imágenes limpias para cada método, para el conjunto de validación de kaggle.

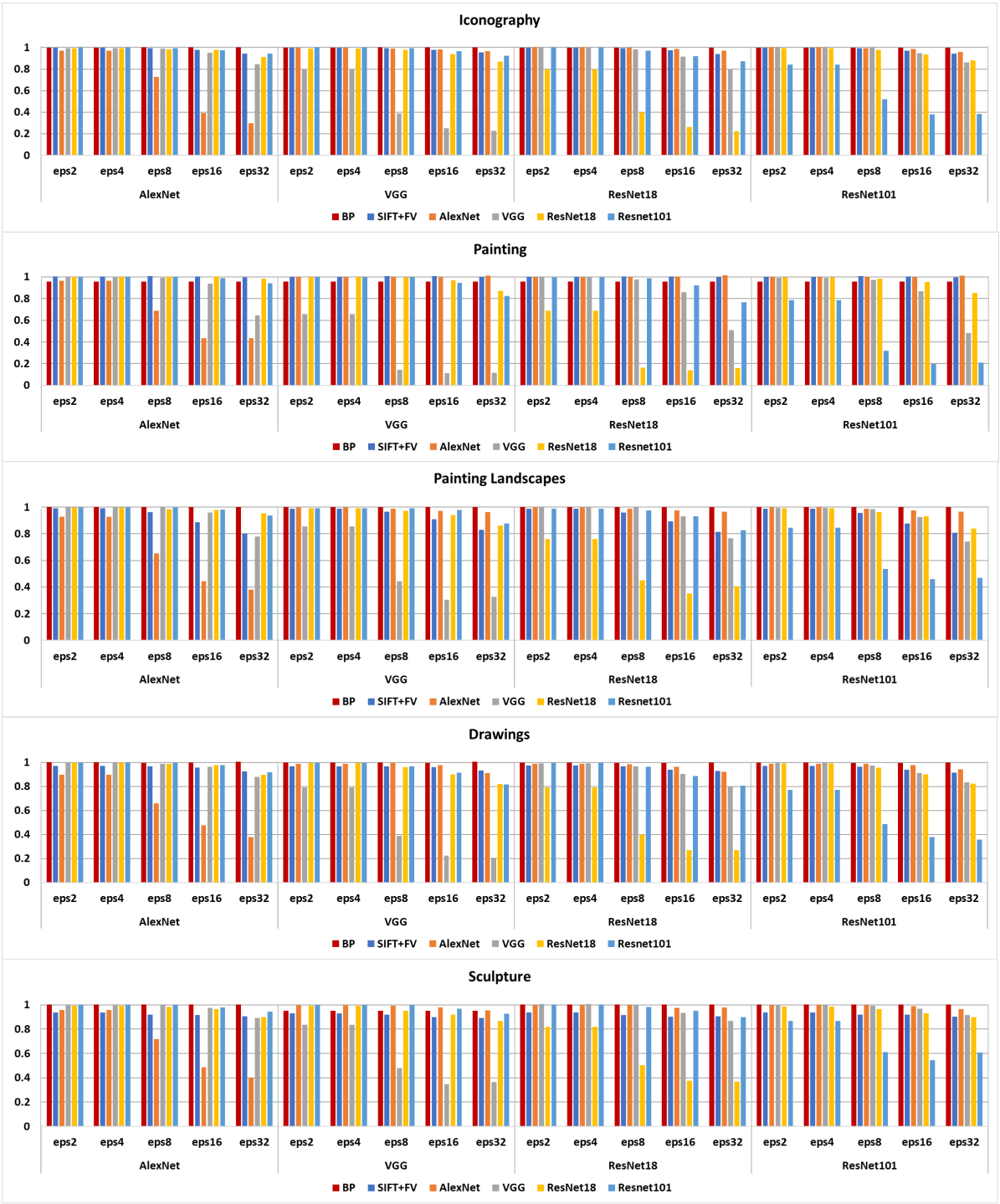

Figura 25. Gráfico comparativo de la precisión calculada entre las imágenes con ataque y las imágenes limpias para cada método. Se presentan los datos para las imágenes de prueba de las clases Iconografía, pinturas, pinturas de paisajes, dibujos y esculturas.

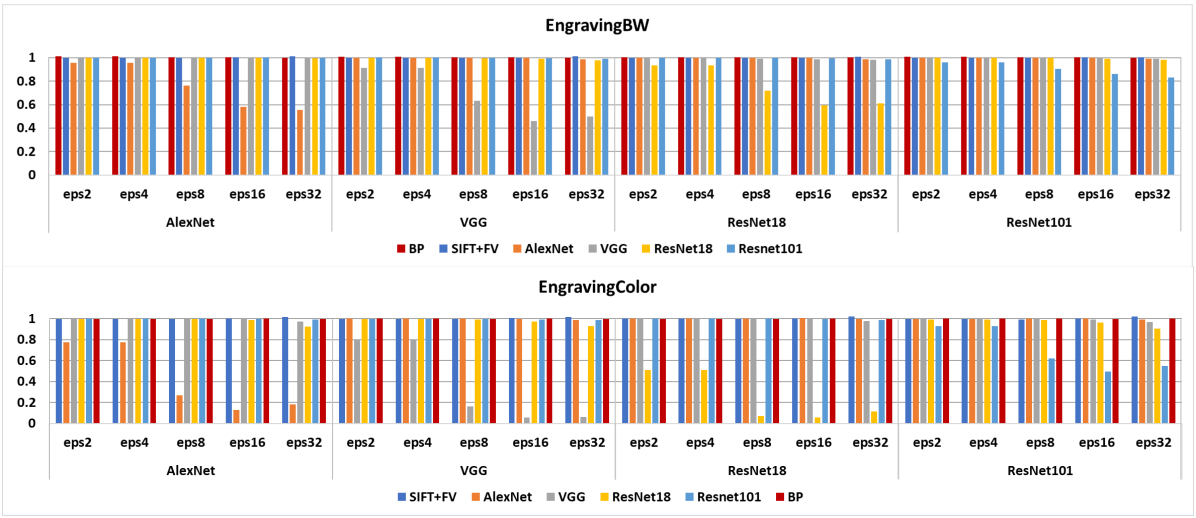

Figura 26. Gráfico comparativo de la precisión calculada entre las imágenes con ataque y las imágenes limpias para cada método. Se presentan los datos para las imágenes de prueba de las clases Grabados en Blanco y negro y Grabados a color.

Capítulo 6. Discusión y conclusiones

El Brain Programming ya habia probado ser una metodología muy eficaz para el problema de clasificación en las bases de datos diseñadas para ese propósito, sin embargo el problema de clasificación de imágenes de arte al ser un problema más cercano a los retos del mundo real, presenta una complejidad mayor que las tareas que se le habian presentado con anterioridad.

En este trabajo, ademas de abordar el problema de clasificación de imágenes de arte, se aplicó una metodología llamada Hands-On que nos permitió mejorar aún más el desempeño del Brain Programming. También se realizó un estudio acerca de la robustez de las soluciones generadas

6.1. Conclusiones

El objetivo de esta investigación fue probar una metodología llamada Brain Programming que puede desafiar al aprendizaje profundo en el problema de categorización de arte de acuerdo al medio. BP es un sistema evolutivo que diseña automáticamente modelos computacionales (simbólicos) que describen una clase específica. Esta representación simbólica es similar a los sistemas tradicionales de visión por computadora (invariancia en el punto de vista, oclusión, segmentación, agrupación, integración, descripción y similares) necesarios para recuperar formas prototípicas. Por tanto, las soluciones propuestas son susceptibles de ser estudiadas.

Se presentaron los resultados de la aplicación de este método al problema de categorización de obras de arte digitalizadas de acuerdo al medio. Los resultados confirman que BP iguala o supera a DL en tres de las cinco clases con una puntuación de precisión global superior al 90 %. Esta metodología, a diferencia del aprendizaje profundo, puede agregar conocimiento experto al proceso para resolver el problema. Por tanto, representa un punto intermedio entre los métodos convencionales de reconocimiento de arte que dedican una cantidad considerable de tiempo a diseñar sus extractores de características y el aprendizaje profundo que funciona como una caja negra en la que el investigador no puede incorporar conocimientos expertos para elegir las características.

Nuestra propuesta puede explicar el funcionamiento de cada una de sus etapas, a diferencia del aprendizaje profundo, donde no es evidente explicar las razones por las que funciona el algoritmo.

Con la creciente integración de la digitalización en la vida diaria y la confianza depositada en este tipo de tecnología, la robustez contra ataques adversarios debe ser una preocupación cuando se desa-

rrolla un sistema de reconocimiento automático. Para complementar el trabajo, presentamos un estudio comparativo para el problema de categorización de imágenes de arte sujeto a ataques adversarios en el cual comparamos el desempeño del Brain Programming con 5 modelos de 2 de los principales enfoques que se usan comúnmente para clasificar imágenes: un enfoque de características extraídas a mano (SIFT + FV), y cuatro DCNN (AlexNet, VGG, ResNet18 y ResNet101). Además, se utilizaron tres ataques para crear los ejemplos adversarios: white box sin objetivo (FGSM), black box sin objetivo (ataque de un píxel) y un ataque dirigido (Parche adversario).

En este sentido, el experimento ha demostrado que los ataques adversarios son una grave amenaza para el rendimiento de las DCNN. El uso de FGSM demostró que si el atacante conoce el modelo, puede hacer que la DCNN disminuya su rendimiento hasta menos del 20 % de su puntuación original. Además, comprobamos el efecto de transferibilidad entre modelos DCNN, que no es tan severo para la clasificación binaria, pero que puede reducir en algunos casos hasta un 20 % del rendimiento. Por otro lado, SIFT + FV también se vio afectado por algunas de los atques, pero en menor medida. Sin embargo, la desventaja de las funciones diseñadas manualmente, es la limitación de su precisión ya que no logra competir con DCNN. En contraste, el Brain Programming exhibe un desempeño (eficiencia) comparable al DCNN con ligeras diferencias en su precisión ante estas perturbaciones, lo que demuestra que no se puede transferir directamente los ataques a este modelo

El estudio realizado usando el ataque de un píxel confirma el mal diseño de este tipo de ataques debido a que es utilizado con una imagen de entrada de un tamaño de 32×32 píxeles. Concluimos que es un desafío aplicar este ataque en condiciones del mundo real, principalmente porque es necesario ampliar el número de pixeles para que el ataque tenga efecto sobre una imagen real, y de esta forma el ataque pierde la intención de ser imperceptible para la visión humana. Aún así, el BP muestra que es difícil hacerlo fallar ante estos ataques e incluso fue necesario aumentar en cinco veces el número de píxeles del ataque para producir algun efecto. El parche adversario mostró que una perturbación precalculada colocada en una ubicación aleatoria podría hacer que la predicción sea incorrecta para los modelos DCNN. Además, esta perturbación que se hace sobre un modelo en particular, también afectó a otros modelos DCNN; mientras que el Brain Programming conservó su puntaje original.

En conclusión, BP ha demostrado ser resistente a estos ataques adversarios. Además, mostró que no hay transferibilidad directa de tales perturbaciones a los modelos evolucionados. La categorización de los medios artísticos es un problema complejo en el que es difícil superar el rendimiento de DCNN, pero el brain programming obtuvo resultados comparables a los de DCCN, por lo que puede considerarse como una propuesta alternativa de un clasificador de arte de acuerdo al medio sin las vulnerabilidades de AA.

Además, puede agregar conocimiento experto al proceso para resolver el problema. Por último, se puede explicar el funcionamiento de cada una de sus etapas, a diferencia de DCNN, donde generalmente se desconoce qué está sucediendo exactamente dentro del modelo.

El Brain Programming ha demostrado ser una herramienta muy versátil y robusta, siendo su único detractor el tiempo de ejecución por lo que la propuesta de la evolución piramidal surge como una alternativa prometedora ya que logramos obtener un desempeño similar al obtenido en la evolución convencional del Brain Programming, pero con la ventaja de que realiza menos cálculos, lo que se traduce en un programa igual de bueno pero que podemos obtener más rápido.

6.2. Trabajo futuro

Partiendo de los resultados obtenidos a lo largo de este trabajo de investigación, se pueden realizar diferentes proyectos futuros:

El Brain Programming resultó ser una herramienta competitiva para resolver el problema de clasificación de imágenes de arte, sin embargo la clasificación *per sé* no siempre es lo que interesa a los historiadores del arte, si no lo que el aprendizaje automático puede decirnos sobre cómo se identifican las características y los patrones o la secuencia de los cambios. Ninguno de los artículos mencionados en la revisión de literatura aportó una comprensión del problema que pueda ser útil para el historiador del arte. La capacidad de la máquina para clasificar, implica que la máquina ha aprendido una representación interna que codifica características discriminatorias a través de su análisis visual de las pinturas. Sin embargo, es típico que la máquina utilice características visuales que no son interpretables por humanos. Los modelos generados en este trabajo por el Brain Programming tienen la posibilidad de ser explicables dentro de cada una de sus etapas, pueden utilizarse para encontrar una interpretación con sentido físico que sea de utilidad para los historiadores.

Otro punto a considerar, es que al Brain Programming se le pueden agregar conocimientoo experto, y muchas técnicas del estado del arte abordan el problema extrayendo la textura de las pinturas. Existe un trabajo donde se le añade la dimensión de la textura al Brain Programming, sin embargo no ha sido aplicado directaemente al problema de categorización de imágenes de arte, lo que podría llevar a otra linea de investigación.

Para el modelo de la evolución piramidal demostró tener resultados prometedores, pero que aún le falta afinar detalles para su correcto funcionamiente. Es necesaria una prueba estadística para poder

dar un veredicto sobre si la estrategia es mejor que la evolución convencional. Para mejorar el modelo se pueden implementar otras estrategias que nos permitan aumentar la diversidad de los individuos. Se observó durante la evolución, que los individuos tienden a converger hacia un modelo similar demasiado pronto, lo que interpretamos como un mínimo local de la solución. Integrando nuevas estrategias para salirse de ese mínimo local harían que se explore mejor el espacio de búsqueda y aumentaría el desempeño del programa.

Literatura citada

Arora, R. S. y Elgammal, A. (2012). Towards automated classification of fine-art painting style: A comparative study. En: *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. IEEE, pp. 3541–3544.

Bar, Y., Levy, N., y Wolf, L. (2014). Classification of artistic styles using binarized features derived from a deep neural network. En: *European conference on computer vision*. Springer, pp. 71–84.

Brown, T. B., Mané, D., Roy, A., Abadi, M., y Gilmer, J. (2017). Adversarial patch. *31st Conference on Neural Information Processing Systems, NIPS*, p. 6.

Cetinic, E. y Grgic, S. (2016). Genre classification of paintings. En: *International Symposium ELMAR*. pp. 201–204.

Cetinic, E., Lipic, T., y Grgic, S. (2018). Fine-tuning convolutional neural networks for fine art classification. *Expert Systems With Applications*, 114: 107–118.

Chan-Ley, M. y Olague, G. (2020). Categorization of digitized artworks by media with brain programming. *Appl. Opt.*, 59(14): 4437–4447.

Chun, M. M. y Wolfe, J. M. (2001). Chapter nine visual attention. *Blackwell handbook of sensation and perception*, p. 272.

Clemente, E., Chávez, F., Vega, F., y Olague, G. (2015a). Self-adjusting focus of attention in combination with a genetic fuzzy system for improving a laser environment control device system. *Applied Soft Computing*, 32.

Clemente, E., Olague, G., Hernández, D., Briseño, J. L., y Mercado, J. (2015b). Object detection in natural images using the brain programming paradigm with a multi-objective approach. En: *European Conference on the Applications of Evolutionary Computation*. Springer, pp. 201–213.

Das, S. y Suganthan, P. N. (2010). Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15(1): 4–31.

Dickinson, S. J., Leonardis, A., Schiele, B., y Tarr, M. J. (2009). *Object categorization: computer and human vision perspectives*. Cambridge University Press.

Dozal, L., Olague, G., Clemente, E., y Hernández, D. E. (2014). Brain Programming for the Evolution of an Artificial Dorsal Stream. *Cognitive Computation*.

Eiben, A. E., Smith, J. E., *et al.* (2003). *Introduction to evolutionary computing*, Vol. 53. Springer.

Elgammal, A., Kang, Y., y Leeuw, M. D. (2018a). Picasso, matisse, or a fake? automated analysis of drawings at the stroke level for attribution and authentication. En: *32nd AAAI Conference on Artificial Intelligence*.

Elgammal, A., Liu, B., Kim, D., Elhoseiny, M., y Mazzone, M. (2018b). The shape of art history in the eyes of the machine. En: *Thirty-Second AAAI Conference on Artificial Intelligence*.

Everingham, M., Eslami, S., Van Gool, L., Williams, C. K., Winn, J., y Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1): 98–136.

Forsyth, D. A. y Ponce, J. (2003). A modern approach. *Computer vision: a modern approach*, 17: 21–48.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*.

Goodfellow, I. J., Shlens, J., y Szegedy, C. (2015). Explaining and harnessing adversarial examples. En: *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*. p. 11.

He, K., Zhang, X., Ren, S., y Sun, J. (2016). Deep residual learning for image recognition. En: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 770–778.

Hernández, D. E., Olague, G., Clemente, E., y Dozal, L. (2014). Optimizing a conspicuous point detector for camera trajectory estimation with brain programming. *Studies in Computational Intelligence*.

Hernández, D. E., Clemente, E., Olague, G., y Briseño, J. L. (2016). Evolutionary multi-objective visual cortex for object classification in natural images. *Journal of Computational Science*.

Hernández, D. E., Olague, G., Hernández, B., y Clemente, E. (2017). Cuda-based parallelization of a bio-inspired model for fast object classification. *Neural Computing and Applications*, 30: 3007–3018.

Hubel, D. H. y Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*.

Ibarra-Vazquez, G., Olague, G., Puente, C., Chan-Ley, M., y Soubervielle-Montalvo, C. (2021). Automated design of accurate and robust image classifiers with brain programming. En: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. pp. 1–9.

Itti, L. y Koch, C. (2001). Computational modelling of visual attention. *Nature Reviews Neuroscience*.

Itti, L., Koch, C., y Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11): 1254–1259.

Johnson, C. R., Hendriks, E., Berezhnoy, I. J., Brevdo, E., Hughes, S. M., Daubechies, I., Li, J., Postma, E., y Wang, J. Z. (2008). Image processing for artist identification. *IEEE Signal Processing Magazine*, 25(4): 37–48.

Karayev, S., Trentacoste, M., Han, H., Agarwala, A., Darrell, T., Hertzmann, A., y Winnemöller, H. (2014). Recognizing image style. En: *British Machine Vision Conference*.

Keren, D. (2002). Painter identification using local features and naive bayes. En: *Object recognition supported by user interaction for service robots*. Vol. 2, pp. 474–477 vol.2.

Koch, C. y Ullman, S. (1987). Shifts in Selective Visual Attention: Towards the Underlying Neural Circuitry. En: *Matters of Intelligence*.

Kowaliw, T., McCormack, J., y Dorin, A. (2010). Evolutionary automated recognition and characterization of an individual's artistic style. En: *IEEE Congress on Evolutionary Computation*. IEEE, pp. 1–8.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press. Cambridge, MA, USA.

Krizhevsky, A., Sutskever, I., y Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. En: *Advances in Neural Information Processing Systems*. pp. 1097–1105.

Kurakin, A., Goodfellow, I. J., y Bengio, S. (2017). Adversarial machine learning at scale. *5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings*, p. 17.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., y Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4): 541–551.

Li, J. y Wang, J. Z. (2004). Studying digital imagery of ancient paintings by mixtures of stochastic models. *IEEE Transactions on Image Processing*, 13(3): 340–353.

Li, J., Yao, L., Hendriks, E., y Wang, J. Z. (2011). Rhythmic brushstrokes distinguish van gogh from his contemporaries: findings via automated brushstroke extraction. *IEEE transactions on pattern analysis and machine intelligence*, 34(6): 1159–1176.

Luke, S. y Panait, L. (2002). Lexicographic parsimony pressure. En: *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc., GECCO'02, p. 829–836.

Olague, G. (2016). *Evolutionary computer vision: the first footprints*. Springer.

Olague, G. y Chan-Ley, M. (2020). Hands-on artificial evolution through brain programming. En: *Genetic Programming Theory and Practice XVII*. Springer, pp. 227–253.

Olague, G., Clemente, E., Dozal, L., y Hernández, D. E. (2014a). Evolving an artificial visual cortex for object recognition with brain programming. *Studies in Computational Intelligence*.

Olague, G., Dozal, L., Clemente, E., y Ocampo, A. (2014b). Optimizing an artificial dorsal stream on purpose for visual attention. *Studies in Computational Intelligence*.

Olague, G., Clemente, E., Hernández, D., y Barrera, A. (2017). Brain programming and the random search in object categorization. 03. pp. 522–537.

Olague, G., Hernandez, D. E., Clemente, E., y Chan-Ley, M. (2018). Evolving Head Tracking Routines with Brain Programming. *IEEE Access*.

Olague, G., Hernández, D. E., Llamas, P., Clemente, E., y Briseño, J. L. (2019). Brain programming as a new strategy to create visual routines for object tracking. *Multimedia Tools and Applications*.

Olague, G., Ibarra-Vazquez, G., Chan-Ley, M., Puente, C., Soubervielle-Montalvo, C., y Martinez, A. (2020). A deep genetic programming based methodology for art media classification robust to adversarial perturbations. En: *Advances in Visual Computing*. Springer International Publishing, pp. 68–79.

Papageorgiou, C., Oren, M., y Poggio, T. (1998). A general framework for object detection. En: *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*. pp. 555–562.

Perez, C. B. y Olague, G. (2009). Evolutionary learning of local descriptor operators for object recognition.

Pétrowski, A. y Ben-Hamida, S. (2017). Genetic programming for machine learning. En: *Evolutionary Algorithms*, Vol. 9. John Wiley & Sons, Ltd, capítulo 6, pp. 183–216.

Rosado, P. (2019). Computer vision models to categorize art collections according to the visual content: A new approach to the abstract art of antoni tàpies. *Leonardo*, 52: 255–260.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., *et al.* (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3): 211–252.

Saleh, B. y Elgammal, A. (2015). A unified framework for painting classification. En: *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*. IEEE, pp. 1254–1261.

Saleh, B., Abe, K., Arora, R. S., y Elgammal, A. (2016). Toward automated discovery of artistic influence. *Multimedia Tools and Applications*, 75(7): 3565–3591.

Seguin, B., Striolo, C., diLenardo, I., y Kaplan, F. (2016). Visual link retrieval in a database of paintings. En: *European Conference on Computer Vision (ECCV)*. pp. 201–204.

Shapiro, J. (2001). *Genetic Algorithms in Machine Learning*, pp. 146–168. Springer Berlin Heidelberg, Berlin, Heidelberg.

Simonyan, K. y Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*, p. 14.

Su, J., Vargas, D., y Sakurai, K. (2019). One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23: 828–841.

Sun, T., Wang, Y., Yang, J., y Hu, X. (2017). Convolution neural networks with two pathways for image style recognition. *IEEE Transactions on Image Processing*, 26: 4102–4113.

Treisman, A. M. y Gelade, G. (1980). A feature-integration theory of attention. *Cognitive Psychology*.

Trujillo, L. y Olague, G. (2008). Automated design of image operators that detect interest points. En: *Evolutionary Computation*.

van Noord, N., Hendriks, E., y Postma, E. (2015). Toward discovery of the artist's style: Learning to recognize artists by their artworks. *IEEE Signal Processing Magazine*, 32: 46–54.

Viola, P. y Jones, M. J. (2004). Robust real-time face detection. *International journal of computer vision*, 57(2): 137–154.

Wolfe, J. (1994). Guided search 2.0 a revised model of visual search. *Psychonomic Bulletin & Review*, 1: 202–238.

Yang, H. y Min, K. (2020). Classification of basic artistic media based on a deep convolutional approach. *The Visual Computer*, 36(3): 559–578.