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

Maestría en Ciencias en Ciencias de la Computación

Programación cerebral y su robustez contra ataques adversarios mediante la detección del Chorlo Nevado

Tesis

para cubrir parcialmente los requisitos necesarios para obtener el grado de Maestro en Ciencias

Presenta:

Roberto Pineda Núñez

Ensenada, Baja California, México

2022

Tesis defendida por

Roberto Pineda Núñez

y aprobada por el siguiente Comité

Dr. Gustavo Olague Caballero

Director de tesis

Dr. Pedro Gilberto López Mariscal

M. en C. José Luis Briseño Cervantes

Dr. Luis Eduardo Calderón Aguilera

M.C. Jonathan Vargas Vega

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 Roberto Pineda Núñez como requisito parcial para la obtención del grado de Maestro en Ciencias en Ciencias de la Computación.

Programación cerebral y su robustez contra ataques adversarios mediante la detección del Chorlo Nevado

Resumen aprobado por:

Dr. Gustavo Olague Caballero
Director de tesis

Incluso cuando las redes neuronales convolucionales han demostrado ser poderosas en la tarea de detección de objetos sobresalientes, tienen una vulnerabilidad que no debe ser ignorada: son susceptibles a ataques adversarios, los cuales las hacen muy poco confiables. La confiabilidad es un aspecto importante a considerar cuando se habla de detección; sin ésta, cualquier atacante puede volver el algoritmo inutilizable. La programación cerebral (brain programming) es una metodología evolutiva para problemas visuales que es altamente resiliente, y es capaz de soportar incluso las perturbaciones más intensas. En este trabajo, se lleva a cabo por primera vez un estudio que compara la resiliencia a los ataques adversarios y a perturbaciones con ruido usando una base de datos del mundo real de un ave playera llamada Chorlo Nevado *(Charadrius nivosus)* en una tarea de atención visual. Las imágenes en esta base de datos fueron tomadas en el campo y representan un reto en la detección debido a la naturaleza del ambiente y las características físicas del ave. Al atacar tres diferentes redes neuronales convolucionales usando ejemplos adversarios de esta base de datos, se comprueba que no se comparan con el algoritmo del brain programming cuando se trata de resiliencia, ya que sufren grandes pérdidas en su rendimiento. Por otro lado, el brain programming se mantiene robusto y su rendimiento no se ve afectado. También, al usar imágenes del Chorlo Nevado en la experimentación, el trabajo se enfoca en la importancia de la resiliencia en problemas del mundo real donde la conservación de la naturaleza está presente. La programación cerebral es el primer algoritmo evolutivo altamente resiliente usado para tareas de detección de objetos sobresalientes.

Palabras clave: Ataques Adversarios, Detección, Robustez, Conservación, Programación Genética

Abstract of the thesis presented by Roberto Pineda Núñez as a partial requirement to obtain the Master of Science degree in Computer Science.

Brain programming and its robustness against adversarial attacks by the detection of the Snowy Plover

Abstract approved by:

Dr. Gustavo Olague Caballero
Thesis Director

Even when deep convolutional neural networks have proven to be powerful at saliency detection, they have a vulnerability that should not be ignored: they are susceptible to adversarial attacks, which makes them highly unreliable. Reliability is an important aspect to consider when it comes to salient object detection; without it, any attacker can render the algorithm useless. Brain programming is an evolutionary methodology for visual problems that is highly resilient, and is able to withstand even the most intense perturbations. In this work, we perform for the first time a study that compares the resilience against adversarial attacks and noise perturbations using a real-world database of a shorebird called the Snowy Plover *(Charadrius nivosus)* in a visual attention task. The images in this database were taken on the field and even pose a detection challenge due to the nature of the environment and the physical characteristics of the bird. By attacking three different deep convolutional neural networks using adversarial examples from this database, we prove that they are no match for the brain programming algorithm when it comes to resilience, suffering great losses in their performance. On the other hand, brain programming stands its ground and sees its performance unaffected. Also, by using images of the Snowy Plover, we refer to the importance of resilience in real-world issues where conservation is present. Brain programming is the first highly resilient evolutionary algorithm used for saliency detection tasks.

Keywords: Adversarial Attacks, Salient Object Detection, Robustness, Wildlife Conservation, Genetic Programming

Dedicatoria

A Paola

Agradecimientos

Al Centro de Investigación Científica y de Educación Superior de Ensenada (CICESE).

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por el apoyo económico. No. de becario: 993993

Al Dr. Gustavo Olague.

A mis compañeros del laboratorio de EvoVisión.

Tabla de contenido

Página

Tabla de contenido (continuación)

Lista de figuras

Figura Página

Lista de figuras (continuación)

Figura Página

Lista de tablas

Capítulo 1. Introducción

El sistema visual humano y su funcionamiento es una de las características más interesantes en el mundo de la visión por computadora. Desde hace mucho tiempo, los científicos han intentado hacer que una computadora vea, es decir, ser capaz de reconocer los objetos presentes en una escena en un punto determinado. Sin embargo, ver y detectar son dos conceptos diferentes, siendo este último lo que hace la mayoría de los sistemas inteligentes hoy en día. La detección se refiere a encontrar un área donde se encuentra un objeto en una escena y relacionarla con alguna segmentación hecha por un ser humano para probar qué tan certero es el sistema (Sermanet *et al.*, 2014). Las redes neuronales, que utilizan algoritmos altamente poderosos para clasificar escenas o detectar objetos en imágenes, hacen detección pero mediante un aprendizaje de datos, donde en la etapa de entrenamiento de imágenes intentan aprender todos los patrones posibles en la imagen, y se espera que al momento de la validación las escenas sean similares a las de entrenamiento. Esto hace que las redes estén demasiado ligadas a los datos, y debido a esto poseen grandes vulnerabilidades como muy baja resistencia a ataques adversarios (Szegedy *et al.*, 2014). Por otro lado, la programación cerebral es una metodología inspirada en el sistema visual humano, donde se simulan las diferentes capas de la corteza visual del cerebro para llevar a cabo tareas de detección y clasificación (Olague *et al.*, 2014), siendo este proceso altamente robusto a cualquier tipo de perturbaciones en una escena y venciendo a las redes neuronales al ser mucho más confiable. En este trabajo se explora el efecto que diversos tipos de ataques adversarios (el acto de generar perturbaciones en las imágenes de entrada y alimentarlas a un algoritmo de detección) tienen en tres diferentes redes neuronales y en el algoritmo de la programación cerebral, y comparando los resultados en cuatro diferentes bases de datos.

La detección de objetos sobresalientes es un problema bastante útil a investigar en el mundo real, y en este trabajo una de las principales áreas a tratar es lograr detección de un ave playera llamada Chorlo Nevado, una especie de ave amenazada por el creciente disturbio humano en su hábitat. Al declarar a esta ave como objeto de estudio y construirle una base de datos de imágenes, logramos enfocarnos en un problema del mundo real que los algoritmos de detección pueden ayudar a resolver, a diferencia de usar una base de datos estándar donde los objetos no tienen relevancia

y son usados meramente para propósitos académicos. A continuación se describe al ave y los problemas que enfrenta.

1.1. Sobre el Chorlo Nevado

El Chorlo Nevado *(Charadrius nivosus)* es una ave playera que habita en las costas del Pacífico de México, Estados Unidos y Canadá, incluyendo en las costas del Golfo de México. Con un tamaño máximo de 17cm, es una de las aves playeras más pequeñas que se pueden observar en las costas (figura 1). La población reproductora que reside en la costa del pacífico es un segmento poblacional distinto que se distribuye desde Washington, EUA, hasta Baja California Sur, México. Su dependencia en las costas los han hecho vulnerables a los efectos negativos de degradación o pérdida de hábitat y a la creciente perturbación humana. El decremento en su población, los efectos negativos de especies de plantas invasoras y depredadores, y conflictos con desarrollos urbanos han contribuido a elevar el estado de conservación de la especie y a protegerla legalmente con estatus de amenazada a escala federal tanto en México (Semarnat, 2010) como en la costa del pacífico de EUA (USFWS, 1993). Con un tamaño poblacional estimado de 31,000 individuos, es una de las especies de ave playera menos abundantes en Norteamérica y en el hemisferio occidental. El chorlo nevado es especialmente sensible a la perturbación al momento de anidar. Ya que los nidos se ubican como una pequeña depresión en la arena, estos son comúnmente destruidos por gente que los pisa o que se lleva los huevos, por vehículos o caballos que los aplastan, o por otro tipo de perturbación como campamentos o perros sin correa. En algunas playas, se ha incrementado el éxito de reproducción al bloquear el acceso a las playas o al colocar jaulas de malla que protegen el nido contra depredadores grandes. Un algoritmo de detección certero y robusto puede ayudar a solucionar los problemas que enfrenta.

1.2. Preguntas de investigación

En base al problema planteado, se definen las siguientes preguntas de investigación:

1. ¿Puede un algoritmo evolutivo como la programación cerebral resolver un problema de detección al usar una base de datos del mundo real?

Figura 1. El Chorlo Nevado (Charadrius nivosus), un ave playera amenazada la cual constituye uno de los objetos de estudio de este trabajo. Ya que esta especie enfrenta peligros en su hábitat natural, es importante usar algoritmos de detección para monitorear su población.

2. ¿Qué tan robusta es la programación cerebral contra ataques adversarios, y cómo se compara con una red neuronal?

1.3. Objetivo general

Implementar la metodología de la programación cerebral para lograr la detección del Chorlo Nevado mediante el uso de una base de datos de imágenes y probar su resistencia a ataques adversarios, aplicando así la programación cerebral a un problema del mundo real.

1.4. Objetivos específicos

1. Generar una base de datos de imágenes donde se aprecie al Chorlo Nevado en distintas condiciones de posición, iluminación y acercamiento.
2. Aplicar la programación cerebral para lograr la detección del Chorlo Nevado.
3. Probar que la programación cerebral es inmune a los ataques adversarios mediante la comparación con diversas redes neuronales.
4. Probar el rendimiento de la programación cerebral mediante pruebas en campo.

Capítulo 2. Atención Visual

La atención visual se refiere al proceso que relaciona las diferentes características de una escena con el objetivo de seleccionar los aspectos más significativos para la tarea que se quiere realizar. El objetivo de apuntar nuestra visión hacia un objeto de interés a través de una cámara ha contribuido a la creación de modelos computacionales para la atención visual. En la comunidad neurocientífica, la atención visual se define como un proceso natural del cerebro cuya funcionalidad es percibir las características visuales salientes o que más resaltan en una escena. Ésta es una tarea cognitiva estrictamente necesaria en el ser humano, ya que le es imposible enfocar la vista hacia dos objetos diferentes al mismo tiempo. Sin embargo, de acuerdo con Koch y S. (1985), la atención visual es una habilidad que le permite a una criatura, viva o artificial, el dirigir su visión rápidamente hacia los objetos de interés en el ambiente. Los objetos de interés se refieren a las ubicaciones o regiones en la imagen que contienen información importante en un momento determinado. Es por esto que la atención visual es considerada uno de los mecanismos más importantes en el sistema visual, cuya funcionalidad en el cerebro es el proveer el proceso selectivo que filtra la información visual proveniente de la retina a través de la corteza visual.

En la naturaleza, el procesamiento de información es realizado por el cerebro y su funcionalidad se describe mediante el concepto de dos subsistema visuales. Una de las descripciones más aceptadas para el fenómeno de la percepción visual es la hipótesis de los dos canales; el *canal ventral* y el *canal dorsal*. Esta hipótesis especifica que ambos subsistemas reciben la misma información visual como entrada, pero la diferencia radica en las transformaciones que esta información sufre en los canales.

En primer lugar, el canal dorsal está relacionado con la tarea obtener la localización espacial de un objeto en una escena mediante el sistema de procesamiento visual, y por esta razón se le conoce como la ruta del *dónde*, o *cómo*. El canal dorsal comienza en la retina donde la información es proyectada hacia la capa V1, la cual es parte de la corteza visual primaria y se localiza en el lado posterior del cerebro. Después, el canal continua hacia las capas V2 y V3, seguido de el área temporal mediana (MT) y el área temporal mediana superior (MST), las cuales son parte de la corteza visual extra-estriada, y finalmente el canal se encamina hacia la corteza parietal posterior y las áreas adyacentes. En general, es aceptado que el proceso de atención visual es lle-

vado a cabo por el canal dorsal (véase figura 2), y el paradigma más popular para este proceso es la teoría de integración de características presentado por Treisman y Gelade (1980). Es importante notar que el primer modelo computacional para la atención visual fue propuesto por Koch y S. (1985). En este modelo, la imagen es descompuesta en múltiples dimensiones con el propósito de obtener *mapas de conspicuidad*, los cuales son fusionados en uno solo llamado *mapa de sobresaliencia*.

Por otro lado, al canal ventral se le conoce como la ruta del *qué*, ya que está mayormente asociado al reconocimiento de objetos y representación de formas (Oram y Perrett, 1994). De la misma manera que el canal dorsal, éste comienza en la retina y es proyectado hacia la capa V1. Después continua hacia las regiones visuales V2 y V4 que son parte de la corteza visual extra-estriada. Finalmente, el canal ventral termina en las área TEO y TE de la corteza temporal inferior. Desde una perspectiva computacional, el canal ventral es considerado un sistema de procesamiento de información jerárquico y biológicamente inspirado y que se especializa en reconocimiento de objetos (Hubel y Wiesel, 1953).

Resumiendo estos dos canales, se puede entender que el sistema visual es organizado en dos amplias estructuras que controlan la visión espacial y el reconocimiento de objetos. La teoría del *qué* y el *dónde* por Milner y Goodale (2006) le da énfasis a la idea de que el sistema visual está definido de acuerdo a los requerimientos de la tarea que cada canal observa.

2.1. Corteza visual artificial

La corteza visual artificial (en inglés, *Artificial Visual Cortex*, o AVC) es un modelo inspirado en la corteza visual del cerebro y en la manera en que la información visual es procesada. Olague *et al.* (2014) propusieron un enfoque de este modelo que utiliza una funcionalidad común producida en las etapas tempranas de los canales dorsal y ventrales. De esta manera, el AVC ofrece la funcionalidad del canal dorsal para distinguir regiones en la imagen, las cuales son usadas para describir un objeto y después continuar con un modelo del canal ventral. Para esta metodología es necesario entender a una imagen como la primer entrada a un sistema visual, y ésta se define como la gráfica de una función. Esta función es la base para entender la transformación de

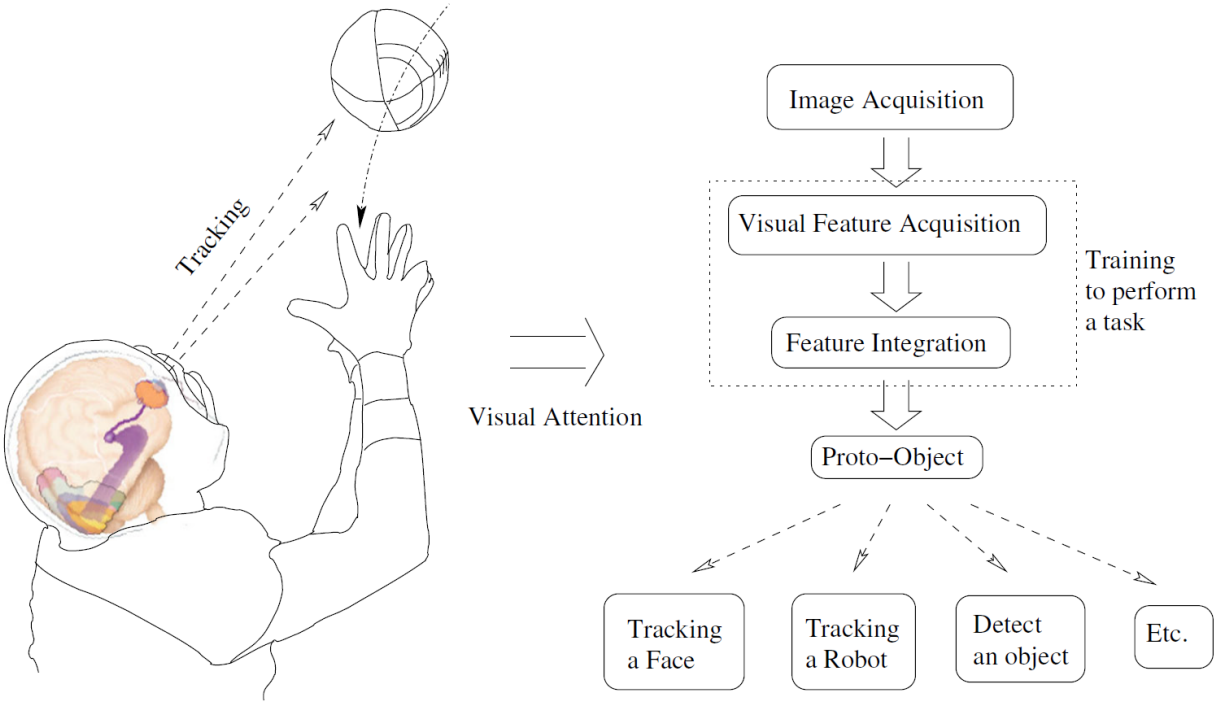


Figura 2. La tarea de la atención visual es llevada a cabo por el cerebro en el canal dorsal. Tras la integración de características, se genera un proto-objeto que representa la región más sobresaliente en una escena.

las propiedades físicas y geométricas de la escena.

Definición 1. Imagen como la gráfica de una función. *Siendo f una función $f : U \subset \mathbb{R}^2 \to \mathbb{R}$. La gráfica o imagen I de f es un subconjunto de $\mathbb{R}^3$ que consiste en los puntos $(x, y, f(x, y))$, en donde el par ordenado (x, y) es un punto en U y $f(x, y)$ es el valor en ese punto. Simbólicamente, la imagen $I = \{(x, y, f(x, y)) \in \mathbb{R}^3 | (x, y) \in U\}$.*

Vista de esta manera, la imagen es la entrada de un sistema computacional que imita la funcionalidad y estructura jerárquica de un sistema visual natural. En otras palabras, cada capa de la corteza visual puede ser modelada por un conjunto de funciones matemáticas. Para esto, se usa el paradigma de la programación genética, que es usado para implementar el modelo del AVC. Se usan un conjunto de *operadores visuales evolucionales* (EVOs), funciones que son optimizadas y siguen la estructura jerárquica del AVC. El resultado final es el programa de reconocimiento óptimo que satisface la tarea de reconocimiento de objetos. Por ende, el objetivo de la programación genética es encontrar el mejor conjunto de EVOs usando un número de funciones sobre la imagen y que funcionen como base sobre todo el proceso jerárquico.

2.2. Canal dorsal artificial

El canal dorsal artificial (del inglés, *Artificial Dorsal Stream*, o ADS) está basado en la teoría de integración de características propuesta por Treisman y Gelade (1980), que sugiere que la atención debe ser procesada de manera serial. El primer paso del ADS es representado por la etapa de adquisición de la imagen. En esta etapa, el sistema considera las imágenes digitales a color en el espacio RGB y sus transformaciones a CMYK y HSV, como valores de entrada del algoritmo. Por lo tanto, la imagen a color es definida como un conjunto de componentes de cada espacio de color: $I_{color} = \{I_r, I_g, I_b, I_c, I_m, I_y, I_k, I_h, I_s, I_v\}$, donde cada elemento corresponde a los espacios de color RGB (Red, Green, Blue), CMYK (Cyan, Magenta, Yellow, Black), y HSV (Hue, Saturation, Value). Después, tres operadores visuales son aplicados por separado para enfatizar características de la imagen correspondientes a las dimensiones de color, orientación y forma. Las operaciones del ADS son evolucionadas con programación genética para obtener un conjunto óptimo de operadores visuales *EVO*. El EVO es una función especializada que fue evolucionada a partir de un conjunto de operadores de imagen, y cada EVO tiene características adecuadas que se usan para crear un conjunto de mapas visuales *VM* a lo largo de las dimensiones de color, orientación y forma. Un cuarto mapa visual, el correspondiente a la intensidad, no es evolucionado y es calculado con el promedio de las bandas RGB. Entonces, por cada mapa visual se crea una pirámide de imágenes para lograr una invariante de posición y escala. La pirámide es reducida hasta que se obtenga un *mapa de conspicuidad* para cada una de las dimensiones anteriormente mencionadas. Después, estos mapas son integrados, mediante una función llamada integración de características evolucionada (EFI), en un solo mapa llamado *mapa de sobresaliencia*. De esta manera, la evolución artificial se encarga de optimizar las tres funciones que extraen de la imagen de entrada la información sobre orientación, color y forma, resultando en un EVO para cada una de estas dimensiones: EVO_C para color, EVO_O para orientación y EVO_S para forma.

2.3. Mapa visual de color

El construir un mapa visual de color VM_C permite crear una nueva imagen que contiene información sobresaliente con respecto a la dimensión de color de la imagen de entrada. En el sistema natural, el color está codificado mediante un conjunto de

Tabla 1. Funciones y terminales para el ADS. La entrada para las funciones pueden ser cualquiera de las terminales, también como la salida de cualquiera de las funciones o una composición de ellas. I es la imagen que entra al ADS.

Funciones para EVO_O	Terminales para EVO_O
$A+B$, $A-B$, $A\times B$, A/B, $\lvert A\rvert$, $\lvert A+B\rvert$, $\lvert A-B\rvert$, $\log_2(A)$, $A/2$, A^2, $\sqrt{A}$, $k\times A$, A/k, $A^{1/k}$, A^k, $(1/k)+A$, $A-(1/k)$, $G_{\sigma=1}(A)$, $G_{\sigma=2}(A)$, $D_x(A)$, $D_y(A)$, $round(A)$, $\lfloor A\rfloor$, $\lceil A\rceil$, $inf(A,B)$, $sup(A,B)$, $thr(A)$, $AttenuateBorders(A)$, $ConvGabor(A)$	I_r, I_g, I_b, I_c, I_m, I_y, I_k, I_h, I_s, I_v, $D_x(I_{color})$, $D_{xx}(I_{color})$, $D_y(I_{color})$, $D_{yy}(I_{color})$, $D_{xy}(I_{color})$, $AttenuateBorders(I_{color})$, $ConvGabor(I_{color})$,
Funciones para EVO_C	Terminales para EVO_C
$A+B$, $A-B$, $A\times B$, A/B, $\log_2(A)$, $exp(A)$, $\lvert A\rvert$, A^2, $\sqrt{A}$, $(A)^c$, $thr(A)$, $round(A)$, $\lfloor A\rfloor$, $\lceil A\rceil$, $k\times A$, A/k, $A^{1/k}$, A^k, $(1/k)+A$, $A-(1/k)$	I_r, I_g, I_b, I_c, I_m, I_y, I_k, I_h, I_s, I_v,DKL_r, DKL_Φ, DKL_Θ, $Op_{r-g}(I)$,$Op_{b-y}(I)$
Funciones para EVO_S	Terminales para EVO_S
$A+B$, $A-B$, $A\times B$, A/B, $\lvert A\rvert$, $k\times A$, A/k, $A^{1/k}$, A^k, $(1/k)+A$, $A-(1/k)$, $round(A)$, $\lfloor A\rfloor$, $\lceil A\rceil$, $A\oplus SE_d$, $A\oplus SE_s$, $A\oplus SE_{dm}$, $A\ominus SE_d$, $A\ominus SE_s$, $A\ominus SE_{dm}$, $Sk(A)$, $Perim(A)$, $A\circledast SE_d$, $A\circledast SE_s$, $A\circledast SE_{dm}$, $T_{hat}(A)$, $B_{hat}(A)$, $A\odot SE_s$, $A\odot SE_s$, $thr(A)$	$I_r, I_g, I_b, I_c, I_m, I_y, I_k, I_h, I_s, I_v$
Funciones para EFI	Terminales para EFI
$A+B$, $A-B$, $A\times B$, A/B, $\lvert A\rvert$, $\lvert A+B\rvert$, $\lvert A-B\rvert$, $k\times A$, A/k, $A^{1/k}$, A^k, $(1/k)+A$, $A-(1/k)$,,$Hist(A)$, $round(A)$, $\lfloor A\rfloor$, $\lceil A\rceil$, $thr(A)$, $(A)^2$, $\sqrt{A}$, $exp(A)$, $G_{\sigma=1}(A)$, $G_{\sigma=2}(A)$, $D_x(A)$, $D_y(A)$	CM_d, $D_x(CM_d)$, $D_{xx}(CM_d)$, $D_y(CM_d)$, $D_{yy}(CM_d)$, $D_{xy}(CM_d)$

células fotorreceptoras llamadas conos, ubicados en la retina. En el ADS, la imagen de entrada es transformada por la función $EVO_C : I_{color} \rightarrow VM_C$, para sobresaltar las características del color. El proceso evolutivo usa el conjunto de funciones y terminales mostrados en la Tabla 1 para generar operadores para la dimensión de color.

2.4. Mapa visual de orientación

La extracción de la información de orientación en el sistema visual humano ocurre mediante una compleja acción celular en la corteza visual V1 y V2. Estas células se encargan de estimar la sensibilidad de los estímulos de orientación mediante la descomposición de la imagen en un conjunto de pequeños segmentos lineales en diferentes orientaciones y escalas. En el ADS, el mapa visual de orientación VM_O es producido al aplicar $EVO_O : I_{color} \rightarrow VM_O$. Este operador es evolucionado con programación genética basada en árboles para optimizar la extracción de información referente a esquinas y orillas dentro de la imagen de entrada I o en sus componentes de color I_{color}. De esta manera, los valores del mapa visual VM_O representan la prominencia de características de la dimensión de orientación. Igual que en el mapa visual de color, se aplican las funciones y terminales de la Tabla 1.

2.5. Mapa visual de forma

La función usada para obtener el mapa visual de la forma se define como: $EVO_S : I_{color} \rightarrow VM_S$. Éste se evoluciona con programación genética para extraer y acentuar características interesantes basadas en la apariencia y estructura de los objetos en la escena mediante morfología matemática. Esta dimensión fue considerada por primera vez por Olague *et al.* (2014).

2.6. Mapa visual de intensidad

La intensidad es una magnitud que indica la cantidad de luz que choca contra un receptor fotosensible. Fisiológicamente, los seres humanos tienen células ganglionares para registrar esta característica. En general, cada célula ganglionar tiene un campo receptor que responde a la intensidad de luz que recibe. En el modelo del ADS, se obtiene la intensidad mediante la simple fórmula $VM_{Int} = \frac{I_r+I_g+I_b}{3}$, donde I_r, I_g y I_b son las bandas rojo, verde y azul de la imagen en el espacio de color RGB. El resultado de esta operación es un mapa visual VM_{Int} que es directamente calculado con los valores de los pixeles obtenidos por la cámara, y que representa la intensidad. Una representación del proceso para la creación de los mapas visuales se muestra en la figura 3.

2.7. Mapas de conspicuidad

Una vez que se obtienes los mapas visuales, el siguiente paso es calcular los mapas de conspicuidad (CM). Estos son obtenidos mediante una función *center-surround*, la cual se inspira de la estructura natural que mide las diferencias de activación entre el centro (C) y los alrededores (S) de las células ganglionares en la retina. El objetivo de este proceso es generar un mapa de conspicuidad *CM* por dimensión de acuerdo al modelo propuesto por Walther y Koch (2006). En el sistema artificial, el proceso se compone de dos partes, donde la información se construye para emular su contraparte natural. Primero, el calculo de los *CM* se modela como la diferencia entre las escalas finas y gruesas, las cuales son calculadas mediante una pirámide de nueve niveles $P_d^{\sigma=0} = \{P_d^{\sigma=1}, P_d^{\sigma=2}, P_d^{\sigma=3}, \ldots, P_d^{\sigma=8}\}$. Cada pirámide es calculada desde su VM_d correspondiente usando un filtro Gaussiano de suavizado, lo cual resulta en una imagen cuyo

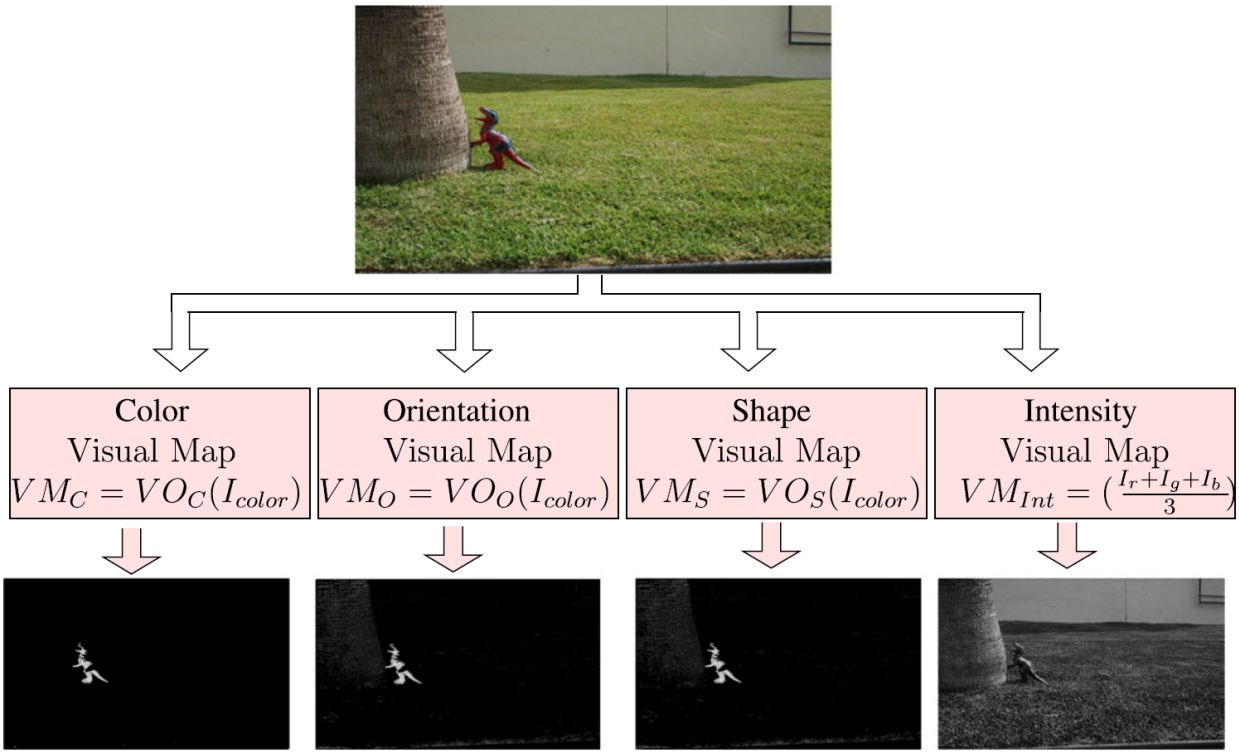


Figura 3. Proceso para calcular los mapas visuales. Los operadores visuales *VO* se usan para calcular las características de cada una de las cuatro dimensiones.

tamaño es la mitad que el mapa visual de entrada, y el proceso es repetido recursivamente ocho veces para completar la pirámide de nueve niveles. En la segunda parte del proceso, la pirámide P_d^σ es usada como dato de entrada para generar seis nuevos mapas center-surround que resultan de la diferencia entre algunos de los niveles de las pirámides calculados de la siguiente manera:

$$Q_d^j = P_d^{\sigma=\lfloor\frac{j+9}{2}\rfloor+1} - P_d^{\sigma=\lfloor\frac{j+2}{2}\rfloor+1} \quad (1)$$

donde $j = \{1, 2, ..., 6\}$. Nótese que los niveles de P_d^σ tienen diferente tamaño y son escalados hacia abajo al tamaño del nivel más alto para calcular sus diferencias. Después, cada uno de estos seis mapas es normalizado y combinado en un mapa único mediante una operación de suma, el cual es normalizado y escalado hasta el tamaño original de los VM_d usando una interpolación polinomial para definir el CM_d resultante. La figura 4 muestra una representación de este proceso.

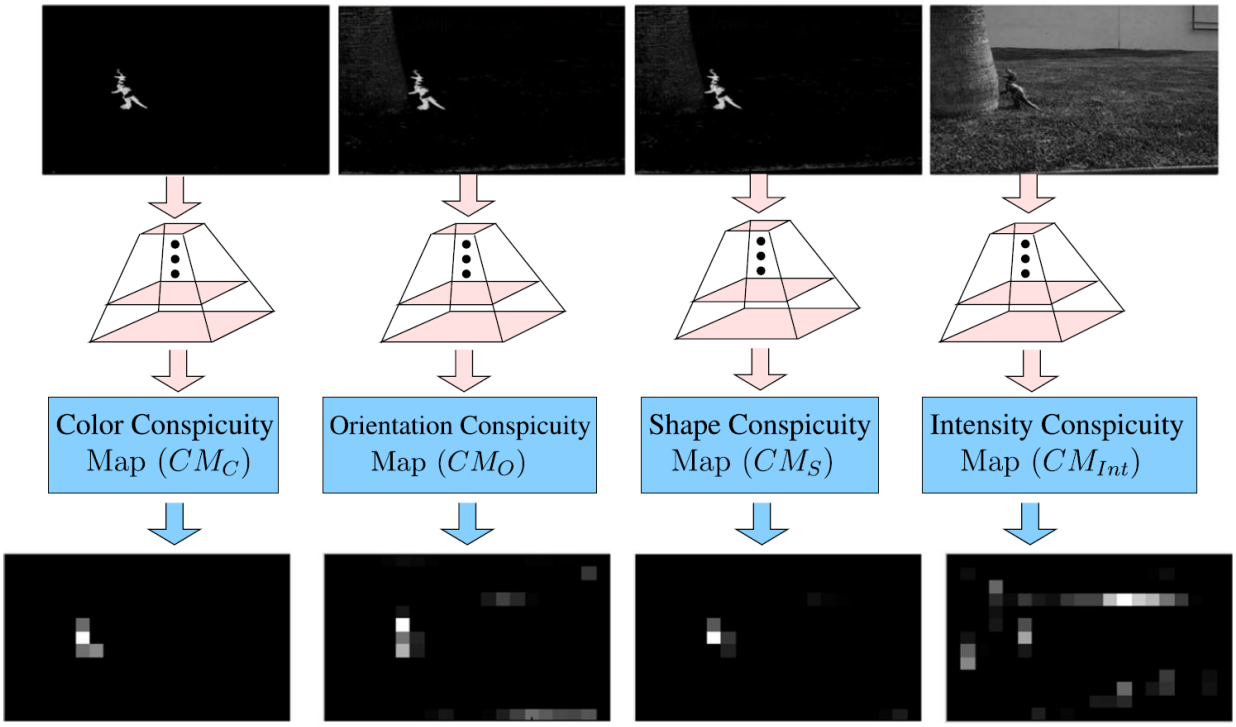


Figura 4. Proceso para calcular los mapas de conspicuidad.

2.8. Integración de características

El siguiente paso del ADS consiste en combinar los mapas de conspicuidad *CMs* del paso anterior para crear un solo mapa conocido como mapa de sobresaliencia (*SM*). La operación de integración de características está definida mediante el mapeo: $FI : CM_d \rightarrow SM$, donde CM_d son los mapas de conspicuidad de las cuatro dimensiones $d = \{C, O, S, I\}$. Actualmente, no existe una descripción detallada de cómo el sistema natural lleva a cabo este proceso; se desconoce cómo el cerebro integra los *CMs*, o en cuál región del cerebro se generan los *SM*. En la combinación de los *CMs*, los valores calculados en la localización de los pixeles representan la prominencia de cada punto en la escena. Después, un algoritmo de esparcimiento se aplica iniciando desde el pixel con el valor absoluto más alto para así definir la región más prominente de la imagen. A esta región prominente, o de alta sobresaliencia, también se le conoce como *proto-objeto* (Rensink, 2000). Es importante mencionar que la imagen de entrada puede contener más de un proto-objeto. Finalmente, el mapa es convertido en una imagen binaria, definiendo así el proto-objeto *P* compuesto de *n* puntos.

El propósito principal de este trabajo es resolver el problema de la detección mediante la generación de los mapas de sobresaliencia, donde la región más prominente en la imagen, denominada como proto-objeto, es el objeto de interés. Sin embargo, en el problema de clasificación (el ligar una imagen a una clase predeterminada), el modelo del AVC se adentra en la generación de mapas mentales y en la definición de un vector descriptor que sirve como entrada a un algoritmo de clasificación. Aunque este trabajo no está enfocado en la clasificación de imágenes, este proceso relacionado a la capa ventral se detalla a continuación.

2.9. Clasificación y cálculo de los mapas mentales

En el sistema natural, el área V4 de la corteza visual se distingue por la respuesta a estímulos complejos sobre orientación, frecuencia espacial, así como a formas como espirales y patrones complejos (David *et al.*, 2006); (Pasupathy y Connor, 1999). Inspirado en esto, en el modelo del AVC se construye un mapa que discrimina la información no deseada de los mapas de conspicuidad CM y se enfoca solamente en el objeto de la imagen a clasificar, sobresaltando las características de ese objeto. A este mapa se le llama *mapa mental*. Tras el cálculo de los mapas de conspicuidad, un conjunto de operadores visuales VO_{MM} es aplicado para describir el contenido de la imagen y producir un mapa mental MM_d por dimensión. Se debe notar que los operadores visuales propuestos son homogéneos e independientemente aplicados a cada dimensión. Esta operación se define como:

$$MM_d = \sum_{i=1}^{k} (VO_{MM_i}(CM_d)) \tag{2}$$

donde d es el índice de la dimensión, y k representa la cardinalidad del conjunto VO_{MM}. Cada suma es aplicada para integrar la salida de todas las operaciones VO_{MM_k} para producir un mapa mental MM_d por dimensión. Después, los cuatro mapas mentales son concatenados en un solo arreglo, y los valores n más altos son seleccionados para definir el vector $\vec{v}$ que describe la imagen (véase figura 5). La entrada para estos operadores es el mapa de conspicuidad correspondiente a cada dimensión.

En el sistema visual natural, la respuesta del área V4 está conectada con la región

inferior-temporal del cerebro (IT), la cual es activada selectivamente hacia el objeto observado, mostrando comportamiento invariante a transformaciones como escala, posición y orientación. Esto es, el área IT exhibe la habilidad para llevar a cabo la tarea de reconocer objetos a partir de los estímulos visuales que recibe (Gross *et al.*, 1972); (Kobatake y Tanaka, 1994); (Hung *et al.*, 2005); (Tanaka, 1996); (Desimone *et al.*, 1984). En el modelo del AVC, la analogía a este proceso es un clasificador implementado con una máquina de soporte vectorial (SVM). Entonces, un SVM es entrenado para aprender un mapeo $f(x)$ que asocia descriptores x_i a etiquetas y_i.

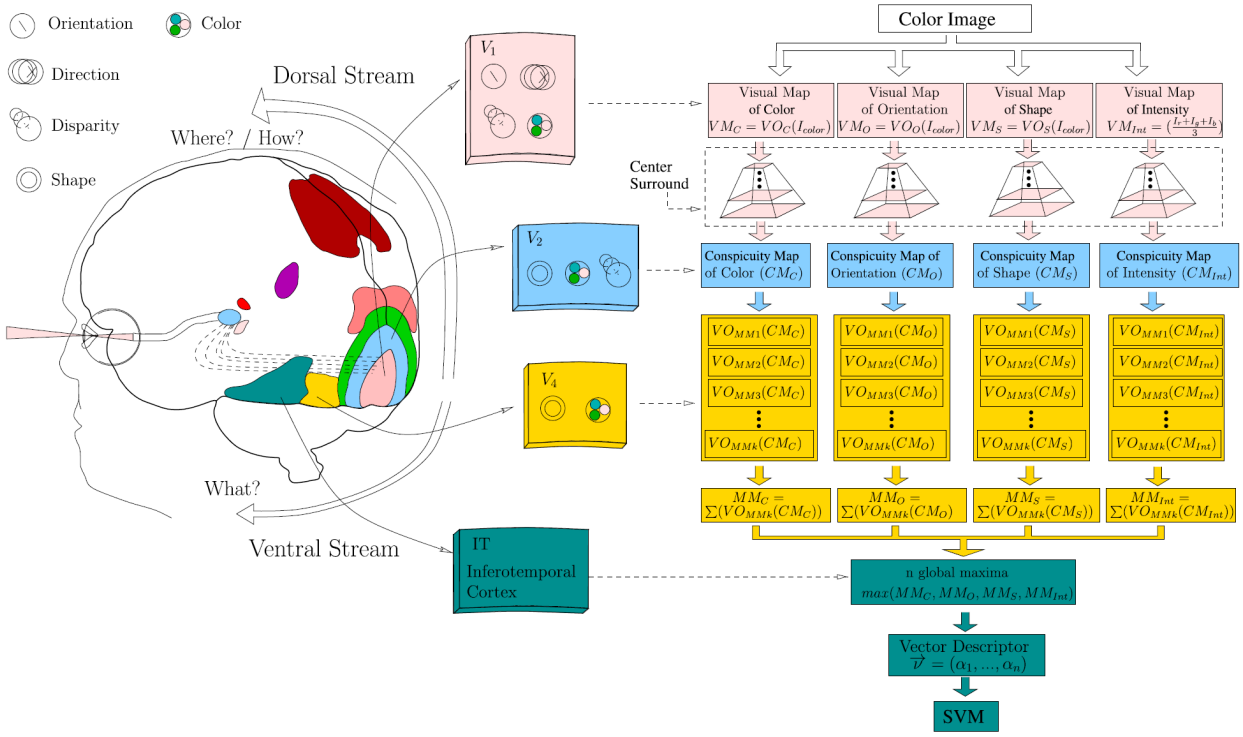


Figura 5. El modelo de la corteza visual artificial. La imagen de entrada se descompone en cuatro dimensiones (color, orientación, forma e intensidad) y pasa por múltiples etapas para producir un vector descriptor para su clasificación.

Capítulo 3. Programación cerebral

La programación cerebral, (en inglés, *brain programming*, o BP), es un nuevo estilo de algoritmo evolutivo donde múltiples programas son evolucionados al estar ligados dentro de un sistema jerárquico complejo que imita un cerebro artificial (Olague *et al.*, 2014). La programación cerebral describe una estrategia de programación genética donde el ADS puede ser optimizado para imitar la funcionalidad de áreas especializadas del cerebro mediante un conjunto de operadores. Su funcionalidad es enfocada en funciones para extraer y combinar la información relevante que resuelve el problema de la atención visual. Se trata de una estrategia bioinspirada donde la diferencia entre la programación genética es que en lugar de que las posibles soluciones o individuos se representen como un arreglo de árboles, estos son representados como una estructura compleja de varios árboles y otros procesos predefinidos. El propósito de esta diferencia es la de emular la funcionalidad de un órgano como el cerebro y su complejidad, y se inspira en la idea de que el cerebro está sujeto al proceso de la evolución (Barton, 1998).

En el trabajo de Dozal *et al.* (2014), la representación genética del ADS, también llamada genotipo, consiste en cuatro árboles ligados a un canal dorsal artificial. En la programación genética, los programas usualmente están codificados como árboles de sintaxis hechos de nodos internos (hojas), que son definidos por un conjunto de elementos primitivos también llamados conjunto de funciones y conjunto de terminales. Estos conjuntos representan el espacio de búsqueda del problema. El genotipo busca un conjunto de operaciones clave que, en combinación con toda la estructura define el diseño deseado mediante la imitación de operaciones en diferentes áreas del cerebro.

La idea del la programación cerebral es el evolucionar diferentes operadores visuales *VO* en operadores visuales evolucionados *EVO*, así como la operación que se encarga de la etapa de integración de características *EFI*, resultando en un mapa de sobresaliencia optimizado (OSM). Cada árbol tiene su propio conjunto de funciones y terminales cuidadosamente seleccionadas de acuerdo a la funcionalidad del cerebro que se quiere emular. Estas funciones y terminales se pueden ver en la Tabla 1. Esta tabla incluye funciones aritméticas entre dos imágenes A y B, funciones cuadradas y trascendentales, funciones de raíz cuadrada, complemento de imagen, oposiciones de color (Red-Green y Blue-Yellow), función de umbral dinámico, funciones aritméticas

entre una imagen A y una constante k. También incluye operaciones trascendentales con una constante k y las coordenadas esféricas del espacio de color DKL. La tabla también incorpora funciones de redondeo, mitad, piso y techo sobre una imagen A, operadores de dilatación y erosión con los elementos de estructurado de diamante, disco y cuadrado. También se incluye filtrado morfológico top-hat y bottom-hat sobre la imagen A, operadores morfológicos abierto y cerrado sobre A, valor absoluto aplicado a A y los operadores de adición y substracción. Finalmente, se agregan las funciones ínfimo y supremo entre imágenes A y B, la convolución de la imagen A, filtro Gaussiano con $\sigma = 1$ o 2, y la derivada de la imagen A sobre la dirección x o y. La entrada para estas funciones puede ser cualquiera de sus terminales correspondientes así como una composición de ellas.

El primer árbol emula la orientación, o la funcionalidad de las células sensibles a orientación en el área V1. Así, el operador de orientación VO_o se evoluciona a EVO_O mediante un conjunto de elementos específicamente seleccionados para resaltar bordes, esquinas y otras características relacionadas a la orientación usando el conjunto de funciones y terminales correspondientes.

El segundo árbol codifica la dimensión de color o la operación de las células fotorreceptoras y las células sensibles al color presentes en las capas V1 y V4 de la corteza visual cerebral. El operador visual VO se evoluciona a EVO_C para reproducir el proceso de la percepción del color. Este proceso evolutivo utiliza el conjunto de funciones y terminales ya descritas. Dentro de estas funciones, la función complemento $(A)^c$ provee una imagen negativa que es el complemento de una intensidad o imagen RGB. En otras palabras, el valor de cada pixel se sustrae del valor máximo de todos los pixeles y la diferencia es usada como el valor del pixel en la imagen de salida. Así, en la imagen resultante, las áreas oscuras se vuelven más claras, y las claras se vuelven más oscuras.

El tercer árbol codifica la dimensión de la forma. El operador de forma VO_S se evoluciona a EVO_S para resaltar las características relacionadas a la apariencia y estructura de los objetos en la imagen. Esto se logra con el conjunto de funciones y terminales relacionados a la forma.

El cuarto árbol codifica la manera en que las características se combinan para crear

el mapa de sobresaliencia optimizado (OSM). Esto se logra mediante la creación de un operador que resalta las características de los objetos de interés. Este método utiliza las funciones y terminales de la sección *EFI* de la Tabla 1.

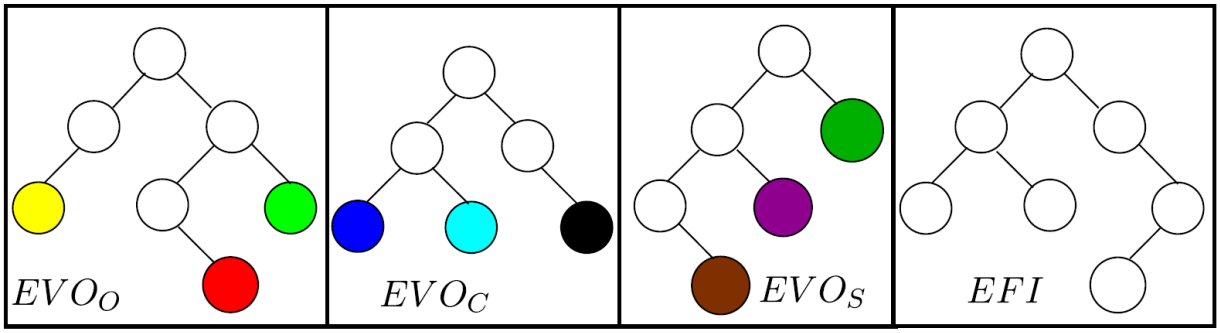


Figura 6. Representación genética del ADS. Cada operador se representa como un árbol sintáctico compuesto de nodos internos (hojas) y definidos por funciones y terminales.

La representación genética o genotipos de cada uno de los *EVO* se muestra en la figura 6. Estas cuatro funciones son evolucionadas con el programa de la programación cerebral demostrado en el diagrama de flujo de la figura 7. El BP recibe como entrada el número de generaciones a evaluar, el tamaño de la población y el número de árboles a ser evolucionados por individuo. Todos los individuos de la población inicial son un conjunto de cuatro árboles indicados por el parámetro *num_trees*. En la experimentación de Dozal *et al.* (2014), el número de generaciones (*num_gen*) y el número de individuos por población (*size_pop*) es 30. El método de inicialización que se le aplica a la población se le conoce como el método *ramped half-and-half* propuesto por Koza (1992). Este método toma la mitad de la población inicial y produce árboles desbalanceados, permitiendo ramas de diferentes longitudes. En la otra mitad, crea árboles balanceados, con todas las ramas de la misma longitud. La longitud de un árbol no debe exceder el máximo especificado para evitar que los árboles crezcan sin control durante la evolución.

La profundidad de un árbol está definida como la longitud que parte desde la raíz hasta un nodo del extremo. Ésta limita el tamaño de cualquier individuo en la población y es dinámicamente controlada por dos parámetros de profundidad que establecen el máximo permitido. El primer parámetro es *dynamic max depth* que es la profundidad máxima de un árbol que no debe ser sobrepasada por ningún individuo a menos de

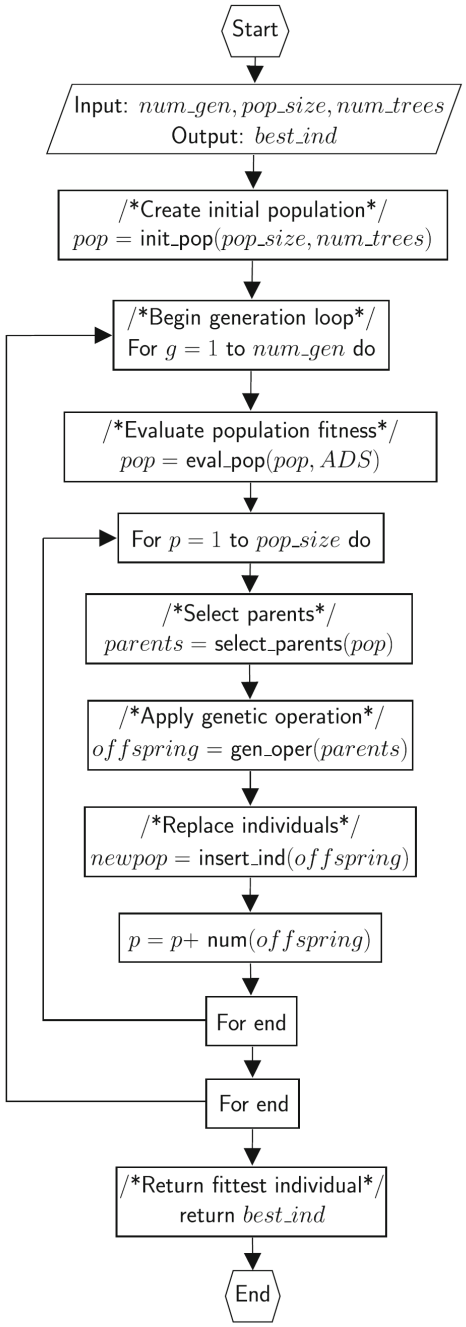


Figura 7. Diagrama de flujo del la programación cerebral.

que su fitness (su puntaje al evaluarlo mediante una función objetivo) sea mejor que la mejor solución encontrada hasta ese momento. Si esto sucede, la *dynamic max depth* es incrementada a la profundidad del árbol del nuevo individuo con el mejor fitness. Al mismo tiempo, esta propiedad es reducida si el nuevo individuo tiene una profundidad menor. El segundo parámetro es *real max depth*, que tiene un límite fijo y que ningún individuo tiene permitido sobrepasar. Finalmente, el criterio para finalizar se define como el máximo número de generaciones. Los valores de los parámetros

usados durante este proceso de evolución se muestran en la tabla 2.

Tabla 2. Parámetros del algoritmo BP.

Parámetros	Descripción
Generación	30
Tamaño de la población	30 individuos
Inicialización	Ramped half & half
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 del árbol	Dinámico
Profundidad max. dinámica	7 niveles
Profundidad max. real	9 niveles
Selección	Roulette-wheel
Elitismo	Mantener el mejor individuo

3.1. Cadenas de Markov

Harel *et al.* (2006) propusieron un modelo de sobresaliencia visual bottom-up llamado Sobresaliencia Visual Basada en Grafos (en inglés *Graph-based Visual Saliency*, o GBVS). Éste se compone de tres pasos: Primero, extrae vectores de características en ubicaciones sobre el plano de la imagen. Segundo, forma mapas de activación usando los vectores de características. Y tercero, normaliza los mapas de activación y los combina en uno solo. Para calcular los mapas de activación, se implementa un proceso basado en cadenas de Markov. A este proceso se le considera orgánico porque, biológicamente, los nodos individuales (neuronas), existen en una red organizada y conectada (la corteza visual) y se comunican una con otra (activación sináptica) en una manera en que resulta un comportamiento emergente, incluyendo decisiones rápidas sobre qué áreas en una escena requieren procesamiento adicional. En este trabajo, el algoritmo de la programación cerebral utilizado en la experimentación es una adaptación del algoritmo GBVS usando la estructura simbólica del la programación cerebral que se explicó anteriormente.

Dentro del proceso de la programación cerebral, el proceso de generación de mapas de activación viene después de la evolución de los operadores *EVO* para lograr la extracción de características. Supongamos que se tiene un mapa de características $M : [n]^2 \rightarrow \mathbb{R}$, el objetivo es calcular un mapa de activación para cada dimensión

$A : [n]^2 \rightarrow \mathbb{R}$, tal que, intuitivamente, las ubicaciones $(i, j) \in [n]^2$, donde la imagen I, o como un proxy, $M(i, j)$, sea de alguna manera inusual en su vecindario corresponderán a valores más altos de activación A. Definiendo la disimilitud de $M(i, j)$ y $M(p, q)$ como:

$$d((i, j)||(p, q)) \triangleq \log \frac{M(i, j)}{M(p, q)} \tag{3}$$

La definición natural de disimilitud se define como la distancia entre uno y la proporción de dos cantidades, medidas en una escala logarítmica. Ahora, consideremos un grafo dirigido completamente conectado denotado como G_A. Para cada nodo M etiquetado con sus índices $(i, j) \in [n]^2$ y conectado a otros nodos $n-1$, la orilla dirigida del nodo (i, j) al nodo (p, q) tendrá un peso de:

$$w_1((i, j), (p, q)) \triangleq d((i, j)||(p, q)) \cdot F(i-p, j-q)\,, \tag{4}$$

donde

$$F(a, b) \triangleq exp\frac{-(a^2 + b^2)}{2\sigma^2}\,, \tag{5}$$

y σ es un parámetro libre del algoritmo. Así, el peso de la orilla del nodo (i, j) al nodo (p, q) es proporcional a la disimilitud y su cercanía en el dominio de M. Es posible definir una cadena de Markov sobre G_A al normalizar los pesos de las orillas de cada nodo a 1, y dibujando una equivalencia entre los estados de los nodos y las probabilidades de la transición de pesos en las orillas.

El normalizar los mapas de activación es un amplia área de estudio y crucial para un algoritmo de sobresaliencia. En este último paso, el objetivo es concentrar la masa en los mapas de activación. Si no se concentra la masa en los mapas de activación antes de la combinación, el mapa resultante puede ser demasiado uniforme y por lo tanto nada informativo. Esto puede parecer trivial, pero es la base de todo algoritmo de sobresaliencia: concentrar la activación en pocas ubicaciones clave. El algoritmo GBVS propone otro enfoque Markoviano. Los autores construyen un grafo G_N con n^2 nodos etiquetados con índices de $[n]^2$. Para cada nodo (i, j) y (p, q) conectados, introducen una orilla desde (i, j) a (p, q) con el peso:

$$w_2((i,j),(p,q)) \triangleq A(p,q) \cdot F(i-p, j-q) \quad (6)$$

Aquí, los pesos de las orillas de cada nodo son normalizados, y tratando el resultado como una cadena de Markov da la oportunidad de calcular la distribución de equilibrio sobre los nodos. Así, la masa fluirá preferentemente a aquellos nodos con alta activación. Éste es un algoritmo de concentración de masa y uno que es paralelizable.

3.2. Operaciones genéticas

Después de que la población inicial fue creada y evaluada con una función de fitness, el siguiente paso del algoritmo BP es la selección y reproducción de los individuos de la población para así crear la nueva generación. Cada nuevo individuo es creado a partir de los padres seleccionados mediante la aplicación de operadores genéticos. Como en los algoritmos genéticos, el BP ejecuta procesos de cruzamiento y mutación. Estos procesos pueden ser a nivel cromosoma o a nivel gen. Ambos métodos de cruzamiento necesitan un par de padres para llevar a cabo la operación. Una representación de estas operaciones se puede observar en la figuras 8 y 9.

En el *cruzamiento a nivel cromosoma*, el algoritmo selecciona aleatoriamente un punto de cruzamiento desde una lista de árboles. El proceso crea un nuevo hijo mediante la unión de la sección izquierda del primer padre con la sección derecha del segundo padre.

En el *cruzamiento a nivel gen* se seleccionan aleatoriamente dos *VOs* desde una lista de árboles, y por cada función de ambos árboles (los genes), el sistema escoge aleatoriamente un punto de cruzamiento. Entonces, los subárboles bajo el punto de cruce son intercambiados para generar dos nuevos operadores visuales. Por lo tanto, esta operación crea dos nuevos cromosomas hijos.

En la *mutación a nivel cromosoma*, el algoritmo selecciona aleatoriamente un punto de mutación dentro del cromosoma de un padre y reemplaza completamente el operador seleccionado con un operador generado aleatoriamente.

En la *mutación a nivel gen*, se selecciona un nodo aleatorio dentro de un solo ár-

bol, llamado punto de mutación. Este nodo seleccionado es borrado y reemplazado por un nuevo subárbol creado aleatoriamente. Si el nodo seleccionado es la raíz del árbol, todo el árbol será substituido por un árbol completamente nuevo y aleatorio. El resultado de esta operación es un hijo.

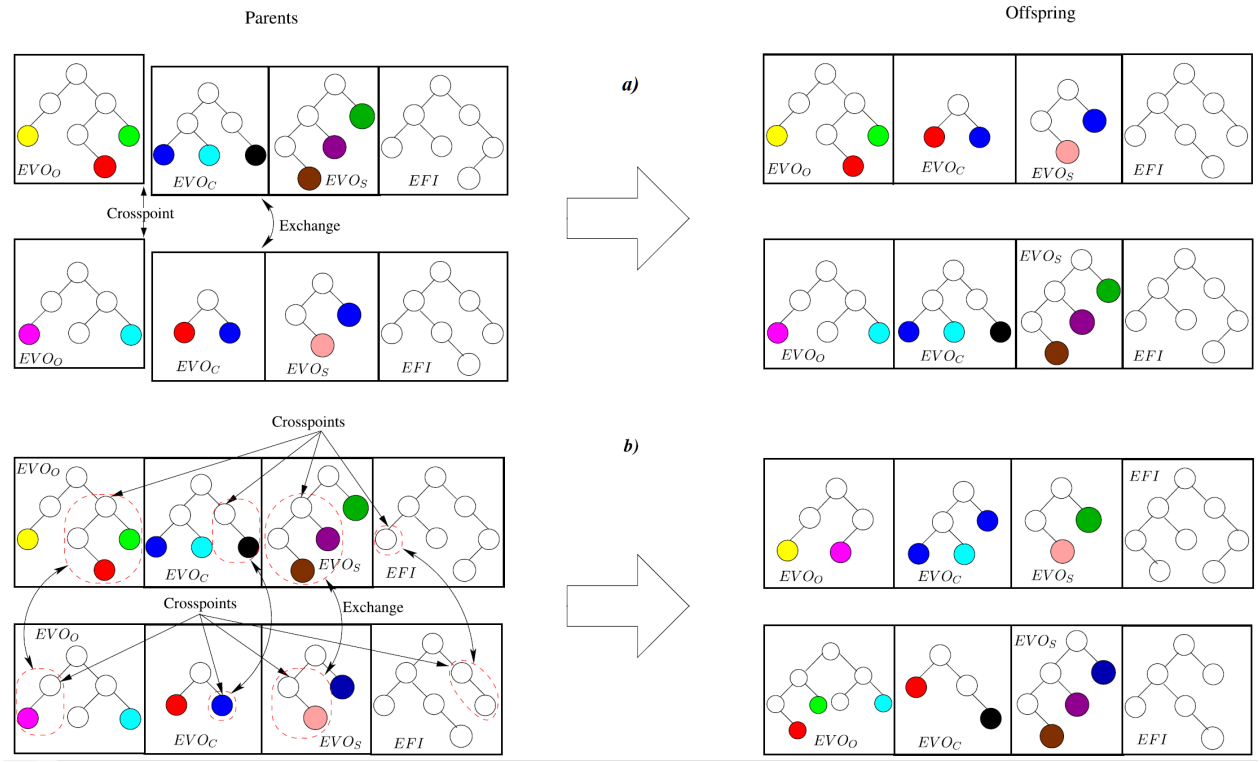


Figura 8. Esquema que ilustra las operaciones de cruzamiento sobre un conjunto de árboles. a) Cruzamiento a nivel cromosoma. b) Cruzamiento a nivel gen.

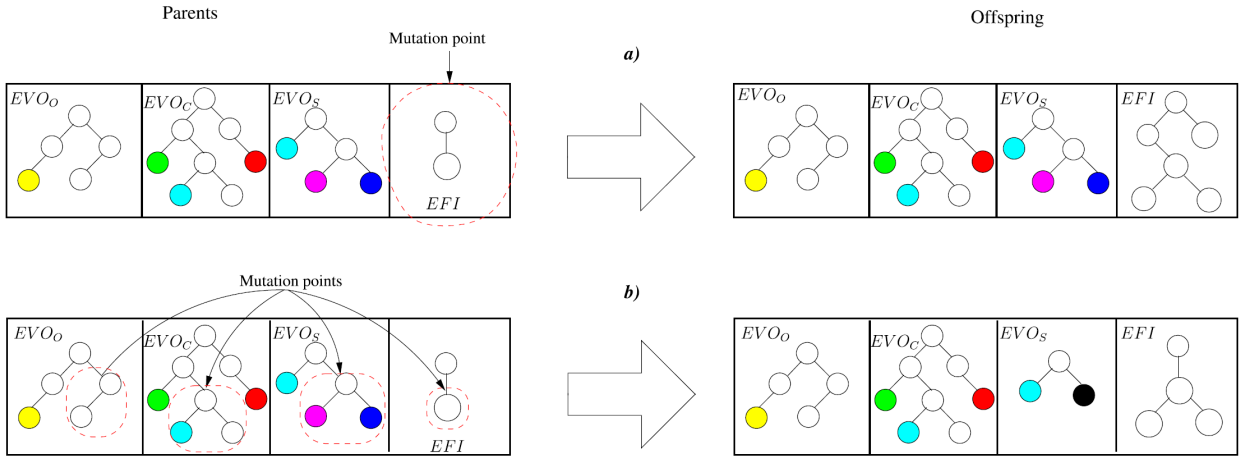


Figura 9. Esquema que ilustra las operaciones de mutación sobre un conjunto de árboles. a) Mutación a nivel cromosoma. b) Mutación a nivel gen.

Los padres son obtenidos mediante un método implementado en la estrategia *roulette-wheel*, la cual consiste en asignar a cada individuo una probabilidad de selección proporcional a su valor de fitness. Así, los ADS con el mejor fitness son más propensos a ser seleccionados, y las operaciones genéticas de cruzamiento y mutación son aplicadas secuencialmente hasta que se crea una nueva población.

El paradigma de la programación cerebral refleja la naturaleza en que es un proceso que nunca termina como cualquier otro algoritmo evolutivo. Sin embargo, por razones de practicidad, una corrida del paradigma del BP termina cuando el criterio de finalizado se satisface. De acuerdo a los parámetros establecidos en la tabla 2, la corrida termina cuando un número máximo de 30 generaciones han sido calculadas. El propósito es encontrar el individuo con el fitness más alto al terminar estas 30 generaciones. Esta función fitness y cómo calcularla se detalla en la siguiente sección.

3.3. Función objetivo

Para evaluar una población de individuos después de haber sido creados, es de vital importancia formular una función objetivo o función fitness que esté relacionada con el objetivo que el ADS intenta lograr. También llamada la medida-F, esta función se se usa para evaluar el rendimiento del ADS para cada individuo generado. Como fue descrito en la etapa de integración de características del ADS, la imagen es convertida a una imagen binaria, también llamada proto-objeto. En el problema de detección, esta imagen debe ser emparejada con su *ground-truth*, otra imagen binaria donde el objeto de interés ya fue previamente segmentado por una persona. Esto permite medir la certeza de un algoritmo al querer realizar la tarea de detección. Las regiones de traslape entre las dos imágenes pueden ser evaluadas como positivas o negativas. Existen cuatro resultados a considerar:

1. *True Positive* (TP) o verdadero positivo; la región del proto-objeto cae dentro del área segmentada manualmente, por lo que la detección en esta zona es correcta.
2. *False Positive* (FP) o falso positivo; la región del proto-objeto cae en un área donde no hay segmentación manual, por lo que la detección en esta zona es incorrecta.

3. *True Negative* (TN) o verdadero negativo; se toma como el área fuera de la segmentación manual y fuera del área del proto objeto.

4. *False Negative* (FN) o falso negativo; se toma como el área dentro de la segmentación manual pero donde el proto objeto no alcanza a tocar.

Para obtener el valor de fitness se necesitan dos medidas más, obtenidas a partir de los cuatro resultados previamente mencionados. Estas son Precisión y Recuerdo (del inglés, *Precision & Recall*):

Precisión denota la proporción que el modelo clasificó como casos verdaderos positivos sobre la cantidad de casos verdaderos positivos y falso positivo. Entonces, la precisión se define como:

$$\text{Precisión } = \frac{TP}{TP + FP} \tag{7}$$

Recuerdo denota la proporción de cuantos casos el modelo clasificó como verdaderos positivos, sobre el número total de casos. Entonces el recuerdo se define como:

$$\text{Recuerdo } = \frac{TP}{TP + FN} \tag{8}$$

La función correspondiente a la medida-F fue primeramente introducida en la Fourth Message Understanding Conference en 1992 para evaluar tareas en tecnologías de extracción de información (Chinchor, 1992). La definición completa para la medida-F se da a continuación:

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \tag{9}$$

donde $(0 \leq \beta \leq +\infty)$. β es un parámetro que controla un balance entre P y R. Cuando $\beta = 1$, F es equivalente al promedio armónico de P y R. Si $\beta > 1$, F se vuelve más orientada al recuerdo y si $\beta < 1$, se vuelve más orientada a precisión (Sasaki, 2007).

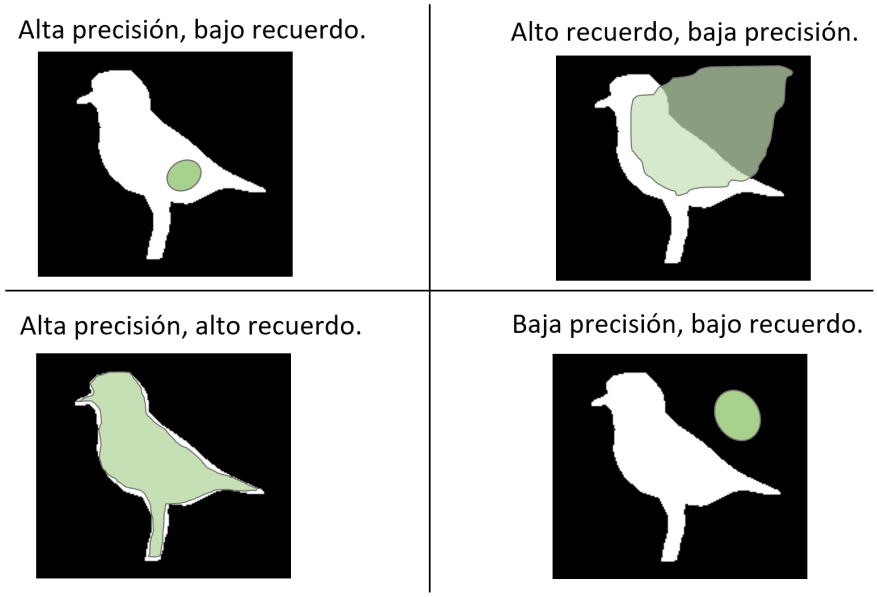


Figura 10. Representación de Precisión y Recuerdo. El área blanca es el ground-truth, o la zona que el modelo de detección debe realmente detectar, mientras que el área verde indica el acercamiento que el modelo obtiene al evaluar.

Aunque no definió la fórmula de la medida-F, el origen de la definición de la medida-F surge a partir de la ecuación para la efectividad E propuesta por Van Rijsbergen (1979):

$$E = 1 - \frac{1}{\alpha \frac{1}{P} + 1(1-\alpha)\frac{1}{R}} \tag{10}$$

donde $\alpha = \frac{1}{\beta^2+1}$, $0 \leq \beta \leq \infty$. En el contexto de la recuperación de información, la precisión y el recuerdo se definen en términos de un conjunto de documentos recuperados F y un conjunto de documentos relevantes R.

$$p = \frac{|F \cap R|}{|F|}, \text{ y } r = \frac{|F \cap R|}{|R|} \tag{11}$$

donde p es precisión $\{0 \leq p \leq 1\}$ y r es recuerdo $\{0 \leq r \leq 1\}$.

En este trabajo, la medida-F es usada como un método para evaluar los resultados arrojados por el modelo de la programación cerebral y diversas redes neuronales. Dada una base de datos de imágenes, esta medida evaluará el nivel de predicción comparando el proto-objeto generado por estos modelos en cada imagen y su seg-

mentación manual (véase figura 10). En los resultados, se utiliza la ecuación 9, donde $\beta^2 = 0.3$, un valor que le da más importancia a la precisión que al recuerdo (Achanta *et al.*, 2009), y F_β es un valor entre 0 y 1. La razón de asignar $\beta^2 = 0.3$ es debido a que la precisión es más importante que el recuerdo en la detección de objetos sobresalientes, ya que es fácil obtener un *recuerdo* = 1 al asignar a toda la imagen como una región que no es parte del fondo. La segmentación y el cómo obtener los valores para el cálculo de la medida-F son explicados en la siguiente sección.

3.4. Proto-objeto y segmentación manual

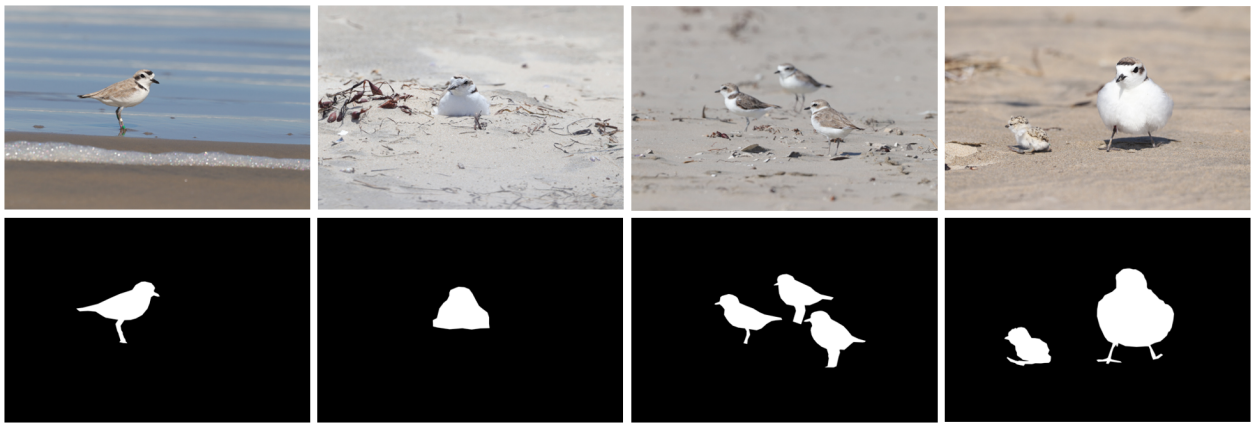


Figura 11. Ejemplos de la base de datos SNPL y su ground-truth. Cada imagen en la base de datos es convertida a una imagen binaria y el objeto de interés es manualmente segmentado.

Como se explicó en la etapa de integración de características del ADS, el resultado final de esta etapa es una imagen binaria conocida como proto-objeto. El proto-objeto representa la región más prominente de una imagen, o en otras palabras, la región donde nuestro modelo determinó que se encuentra el objeto de interés. Nótese que una imagen puede contener uno o más proto-objetos. Debido a la naturaleza de los modelos de detección de objetos sobresalientes, la generación de los proto-objetos nunca es 100 % acertada. Es decir, para que la detección automática de un objeto de interés sea perfecta, el proto-objeto generado deberá ser igual a la segmentación manual de la imagen, también conocida como *ground-truth*.

El ground-truth es una imagen binaria que corresponde a la segmentación manual del objeto de interés. Para evaluar correctamente el rendimiento de un modelo de detección sobre una base de datos de imágenes, esta base de datos debe ser con-

formada por dos conjuntos de imágenes; las imágenes originales y su equivalente en imágenes binarias. Para cada una de las imágenes de la base de datos, se realiza una segmentación manual, donde cada imagen tiene dos posibles valores de pixeles. El objeto de interés corresponde a los pixeles con valor 1, y la región no importante corresponde a pixeles con valor 0. Un ejemplo de imágenes ground-truth se puede apreciar en la figura 11.

Para estimar el rendimiento del modelo de atención visual usando la medida-F, se debe comparar el proto-objeto generado con el ground-truth. De esta manera se define una relación entre el proto-objeto y los valores de precisión y recuerdo. La región que se traslapa entre el proto-objeto y la manualmente segmentada define el área donde se encuentran los valores verdaderos positivos. Las áreas restantes afuera de la región traslapada se refieren a los falsos negativos y los falsos positivos. Véase figura 12.

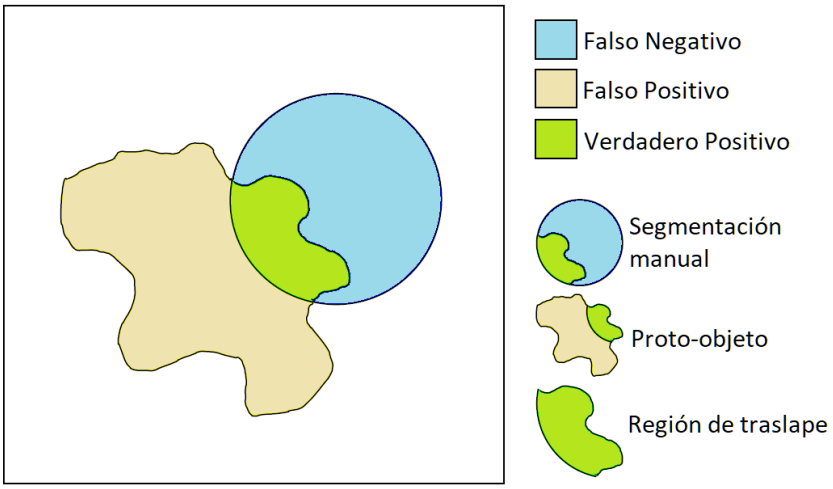


Figura 12. Ilustración que muestra las diferentes regiones en la relación entre proto-objeto y segmentación manual. Estos son usados para calcular la precisión y el recuerdo.

Para calcular la precisión y el recuerdo a partir del traslape entre el proto-objeto y la segmentación manual, las siguientes formulas son aplicadas:

Dado M, que representa el conjunto de pixeles de la región manualmente segmentada, y siendo A el conjunto de pixeles del proto-objeto, el conjunto de pixeles O ubicados en la región de traslape se describe como la intersección de ambos conjuntos $O = A \cap M$. Aquí se sustituyen los valores correspondientes en las ecuaciones de precisión y recuerdo.

$$P = \frac{|O|}{|A|} \text{ y } R = \frac{|O|}{|M|} \tag{12}$$

Así, podemos calcular la medida-F de una sola imagen en la base de datos mediante la ecuación 9.

Capítulo 4. Ataques adversarios

En el mundo del reconocimiento automático de objetos, durante muchos años el estado del arte se ha centrado en modelos basados en redes neuronales. Estos son modelos de aprendizaje bastante poderosos, capaces de obtener resultados excepcionales en problemas de reconocimiento de objetos (Krizhevsky *et al.*, 2012a). Las redes neuronales alcanzan alto rendimiento en sus métricas debido a la naturaleza de sus arquitecturas, las cuales expresan cálculos arbitrarios que consisten en un modesto número de pasos no lineales y paralelos. Szegedy *et al.* (2014) descubrió que, como los cálculos resultantes son automáticamente descubiertos por retropropagación por medio de aprendizaje supervisado, estos pueden tener propiedades contra-intuitivas. Uno de estos problemas tiene que ver con la inestabilidad de estas redes neuronales con respecto a pequeñas perturbaciones en sus imágenes de entrada. Para poder entender mejor el funcionamiento de los ataques adversarios, es necesario explicar el modelo básico de una red neuronal y su funcionamiento.

4.1. Redes neuronales

Desde su introducción por LeCun *et al.* (1989), donde logra el reconocimiento automático de dígitos en códigos postales escritos a mano, las redes neuronales convolucionales han demostrado un excelente rendimiento en este tipo de tareas, incluyendo reconocimiento de caras, imágenes naturales, detección de peatones y muchos más problemas. Con el paso de los años, estas redes han mejorado y demostrado superioridad en tareas de clasificación de imágenes (Krizhevsky *et al.*, 2012b). Varios factores han influenciado esta mejoría: el uso de enormes bases de datos de imágenes (llegando hasta las decenas de miles), y la implementación con GPUs poderosos, logrando tiempos de entrenamiento prácticos. La redes neuronales se pueden implementar para resolver dos tipos de problemas visuales: clasificación y detección (Sermanet *et al.*, 2014). En la tarea de clasificación, a cada imagen se le asigna una etiqueta que corresponde al objeto principal en la imagen, y el objetivo de la red es identificar a qué clase pertenece. En la detección, una imagen puede contener ninguno o múltiples objetos de interés, y el objetivo es, por lo general, detectar el área donde se encuentra el objeto en su totalidad y compararlo con su ground-truth, a diferencia de la clasificación donde solo se necesita una sección del objeto para clasificarlo, como la cabeza de un

perro o las llantas de un auto. Desde que surgieron la redes neuronales, éstas se han enfocado en el problema de clasificación, sin embargo, últimamente han surgido nuevas redes que han atacado el más complicado problema de la detección (Wang *et al.*, 2021). Aún cuando los problemas son diferentes, el funcionamiento de estas redes al intentar resolverlos es fundamentalmente el mismo.

La redes neuronales están modeladas a partir del comportamiento de las neuronas en el cerebro. En el modelo computacional básico, las neuronas son representadas por nodos, los cuales contienen los datos de entrada que se alimentan a la red. Estos nodos se organizan en capas, o *layers*, que estructuran la información contenida en los nodos y la procesan, y tienen el objetivo de encontrar patrones en los datos de entrada. La capa de entrada (*input layer*), que recibe los datos de entrada, contiene un nodo por cada componente presente en los datos de entrada, y estos componentes son comúnmente los valores de los pixeles de la imagen a clasificar o detectar. La capa de entrada se comunica con una o más capas escondidas (*hidden layers*), en donde todo el procesamiento sucede a partir de un par de variables llamadas *pesos* y *polarizaciones*. El peso y la polarización (del inglés, *weights & biases*) son parámetros que aplican transformaciones a los datos de entrada para así mejorar la predicción dentro del proceso de la red. Cuando un valor entra a un nodo (comúnmente en una de las capas escondidas), éste es multiplicado por un peso y sumado por un valor de polarización. Esta operación es aplicada en cada nodo de la capa actual y los resultados son sumados mediante una operación llamada *suma de pesos*. En este paso, se aplica la función *sigmoid* usando estos valores de la siguiente manera:

$$out = \sigma(wx + b) \tag{13}$$

donde σ representa la función *sigmoid*, w el peso, x los datos de entrada y b la polarización. Sustituyendo, la formula para la salida es:

$$out = \frac{1}{1 + e^{-\sum wx - b}} \tag{14}$$

Donde el resultado *out* es un valor entre 0 y 1. La función *sigmoid* es interesante ya que pequeños cambios en el valor del peso o la polarización genera pequeños cambios

en la salida, y este cambio intencional de pesos es uno de los pasos principales en el entrenamiento de la red neuronal. El resultado de este paso es transferido a la siguiente capa de la red, en un proceso llamado *feedforward*. Dependiendo de cuantas capas escondidas existan en la red neuronal, este paso es repetido en la siguiente capa para seguir calculando la predicción y el resultado es transferido mediante *feedforward* por todas las capas hasta llegar a la capa de salida (*output layer*).

Una habilidad de las redes neuronales es su capacidad de aprenderse los datos y continuamente mejorar su predicción. Conforme los pesos y la polarización que definen la conexión entre los nodos (o neuronas) se vuelven más precisos, eventualmente la salida de la red neuronal se aproximará al valor real $\hat{y}$. Para calcular qué tan cercana es la predicción al valor real, se necesita calcular el *error*, y eso se logra con una *función costo*. Esta función no es nada más que un cálculo de la diferencia entre la salida esperada y la salida predicha por la red. La función más común para esta tarea es la *mean squared error*, la cual se da como:

$$C(w, b) = \frac{1}{2n} \sum_{i=1}^{n} (\hat{y} - y)^2 \tag{15}$$

donde n es el numero total de datos de entrada en el entrenamiento, y es la salida de la red y $\hat{y}$ es el valor real a predecir. La función costo, o error, se vuelve más pequeña mientras la salida y se acerca al valor real $\hat{y}$. El objetivo de una red neuronal es encontrar un conjunto de pesos y polarizaciones adecuadas para minimizar lo más posible esta función costo. Para lograr esto, se utiliza otro algoritmo llamado *descenso de gradiente*. El gradiente de una función provee la dirección de ascenso más empinada, en otras palabras, ayuda a encontrar el máximo local de una función. Ya que el propósito de una red neuronal es minimizar la función costo, se necesita encontrar el mínimo local. Si el gradiente está dado por $\nabla(W)$, entonces el descenso de gradiente se da como $-\nabla(W)$. Así, el descenso de gradiente calcula la pendiente de la función costo y modifica los pesos y la polarización repetidamente de acuerdo a la pendiente para encontrar los valores correctos que minimicen lo más posible esta función costo. Por último, para lograr calcular el gradiente de esta función costo, se aplica otro concepto llamado *retropropagación* (Rumelhart *et al.*, 1986). Aquí, tras haber calculado el error en la capa final, los resultados son propagados hacia las capas anteriores

para actualizar los pesos y la polarización y así minimizar mejor el error. La figura 13 muestra una representación de todo este proceso.

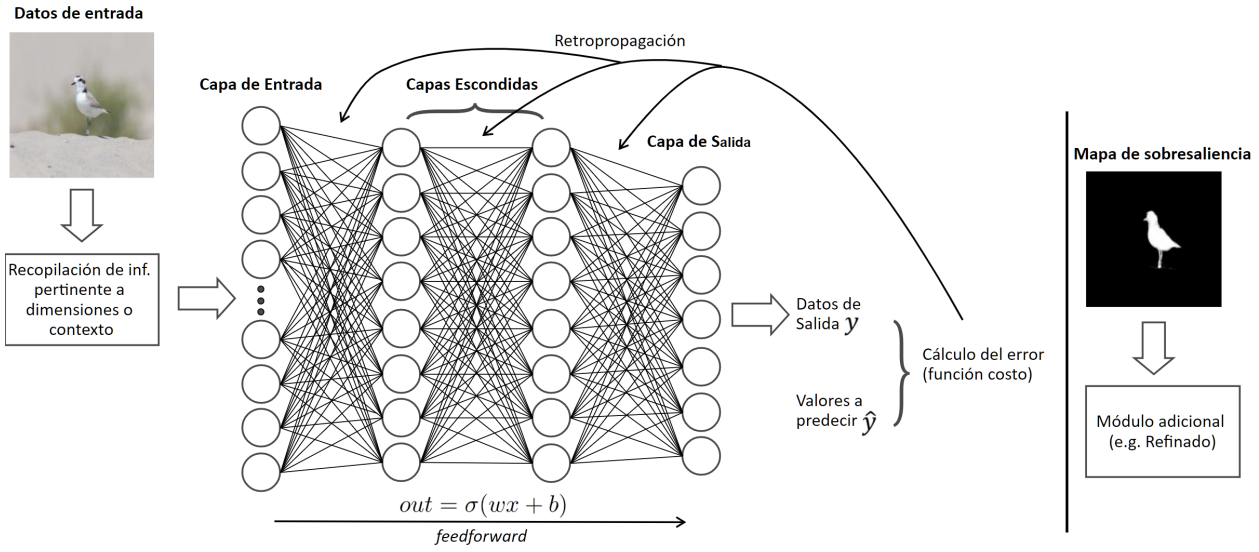


Figura 13. Representación del proceso en una red neuronal, tomando en cuenta los procesos previos y posteriores, los cuales dependen de la arquitectura de cada modelo de detección.

Este proceso es repetido múltiples veces dependiendo de la red que se utilice. A lo largo de los años han surgido una gran cantidad de redes para tareas de clasificación y detección, con diferente número de capas como VGG-19 con 19 capas (Simonyan y Zisserman, 2015), ResNet50 con 50 capas, (He *et al.*, 2016), entre otras. Generalmente se cree que entre más número de capas, mejor la predicción. Sin embargo, debido a la arquitectura que se acaba de explicar, éstas poseen una grave vulnerabilidad que se explica en la sección 4.2. En este trabajo, se probó el rendimiento de tres redes neuronales del estado del arte, las cuales se describen a continuación.

4.1.1. DHSNet

DHSNet, o la red de sobresaliencia jerárquica profunda para la detección de objetos sobresalientes (en inglés, *Deep Hierarchical Saliency Network For Salient Object Detection*) (Liu y Han, 2016) es una red de sobresaliencia jerárquica basada en redes neuronales para detectar objetos sobresalientes en imágenes. Este algoritmo primero realiza una predicción al aprender automáticamente varias características de sobresaliencia globales como contraste, áreas con alta probabilidad de objetos sobresalientes

y su combinación óptima. Después, se implementa una red neuronal convolucional recurrente (HRCNN) para refinar los mapas de sobresaliencia generados y así recuperar detalles de la imagen al integrar información de contexto local, y así evitar usar métodos para sobre-segmentar. DHSNet alega gran superioridad sobre otros algoritmos de detección de objetos sobresalientes, especialmente en bases de datos complejas.

4.1.2. PiCANet

PiCANet, o la atención contextual basada en pixeles para la detección de sobresaliencia (en inglés, *Learning Pixel-wise Contextual Attention for Saliency Detection*) (Liu *et al.*, 2018) es una red que utiliza información contextual para lograr la detección. Para cada pixel en la imagen, genera información contextual, es decir, regiones en la imagen que representan la escena, donde los pesos de la atención indican qué tan relevante es cada ubicación contextual con respecto al pixel. Las características de las regiones contextuales son pesadas y agregadas para obtener una ubicación que sea mucho más informativa y así ignorar aquellas ubicaciones irrelevantes. De esta manera se logra la detección. Para integrar el contexto en diferentes enfoques, PiCANet se implementa de manera global y local para integrar contextos globales y locales. Por último, se implementan estos PiCANets dentro de una arquitectura U-Net, la cual se utiliza para la segmentación.

4.1.3. BASNet

BASNet, o la red de segmentación atenta a contornos (en inglés, *Boundary-Aware Segmentation Network*) (Qin *et al.*, 2019) es una red para segmentar imágenes con alta precisión. Consiste en una red que predice y refina secuencialmente mapas de probabilidad. Consiste en una red parecida a la U-Net que transfiere la imagen de entrada a un mapa sobresaliencia, y un módulo de refinado refina el mapa al aprender los residuos entre el mapa de sobresaliencia con características globales y el ground-truth. Este modelo de refinado se usa una sola vez en la escala original de los mapas de segmentación. Varios tipos de pérdida o errores son usados para supervisar el proceso de entrenamiento en una jerarquía de tres niveles: pixel, parche y mapa. Esta red se enfoca en la predicción de orillas en la escena, lo que puede arrojar buenos resultados

dependiendo de la base de datos.

4.2. Ejemplos adversarios

De una red neuronal del estado del arte se espera que discrimine bien en su tarea de reconocimiento de objetos. Se espera que esa red sea robusta a pequeñas perturbaciones en sus imágenes de entrada, ya que una pequeña perturbación en sus pixeles no debería alterar su categoría porque el contenido de la imagen sigue siendo virtualmente la misma. Sin embargo, Szegedy *et al.* (2014) notó que al aplicar una perturbación imperceptible y no aleatoria a una imagen de prueba, es posible cambiar la predicción de una red neuronal arbitrariamente. Se menciona no aleatoria ya que estas perturbaciones se generan al momento de optimizar la entrada para maximizar el error de predicción, caso contrario al objetivo de una red neuronal la cual es minimizar el error. A estas imágenes perturbadas se les conoce como *ejemplos adversarios*.

En muchos casos, una gran variedad de modelos de redes neuronales con diferentes arquitecturas y entrenados en diferentes subconjuntos de las imágenes de entrenamiento, fallan al detectar o clasificar el mismo ejemplo adversario. Esto sugiere que los ejemplos adversarios exponen puntos ciegos fundamentales en estos algoritmos de entrenamiento (Goodfellow *et al.*, 2015). El problema de la detección errónea en los ejemplos adversarios se puede representar de la siguiente manera: Supongamos que existe un sistema de aprendizaje de máquina M y una imagen de entrada no perturbada I. Se asume que el ejemplo I es detectado correctamente por el sistema de aprendizaje de máquina; $M(I) = y_{true}$. Es posible construir un ejemplo adversario A cuya diferencia de I es visiblemente imperceptible pero que sea detectado incorrectamente; $M(A) \neq y_{true}$. Estos ejemplos adversarios son erróneamente detectados más comúnmente que ejemplos que han sido alterados con ruido, incluso si la magnitud del ruido es mucho mayor que la magnitud de la perturbación al generar el ejemplo adversario.

Existe otra propiedad de los ejemplos adversarios llamada *transferibilidad*. Szegedy *et al.* (2014) demostró que un ejemplo adversario que fue diseñado para ser detectado erróneamente por un modelo M_1 es comúnmente detectado erróneamente por otro modelo M_2. Esta propiedad demuestra que es posible generar ejemplos adversarios y

llevar a cabo ataques a otros modelos de detección sin tener acceso a su arquitectura. Cabe mencionar que a la acción de usar ejemplos adversarios para atacar o perturbar el rendimiento de detección de un modelo de aprendizaje de máquina también se le conoce como *ataque adversario*.

Sin embargo, es incorrecto asumir que un ataque adversario solo funcionaría al alimentar ejemplos adversarios directamente a un detector de aprendizaje de máquina. Kurakin *et al.* (2017) demostraron que es posible atacar sistemas de aprendizaje de máquina que operan en el mundo físico, donde el atacante no puede depender de la habilidad de hacer modificaciones finas pixel por pixel. Ejemplos de estos sistemas son: robots que perciben el mundo a través de cámaras y otros sensores, sistemas de vigilancia con video y aplicaciones móviles para clasificación de imagen o sonido.

Por último, existen dos tipos de ataques. En los ataques de caja blanca (white-box), el atacante o adversario tiene acceso a la arquitectura del modelo. Este incluye los parámetros con los que trabaja y el atacante es capaz de modificarlos para generar ejemplos adversarios capaces de alterar el rendimiento del modelo a su gusto. El otro tipo de ataque adversario es el de caja negra (black-box), en el cual no se tiene acceso a la arquitectura del modelo; solamente a la salida que este arroja. El atacante depende de la generación de ejemplos adversarios desde otro modelo diferente y de la esperanza de que exista transferibilidad al modelo que quiere atacar (Zhang *et al.*, 2020).

En este trabajo existe otro propósito para experimentar con ataques adversarios usando la base de datos de un ave. Dado que el Chorlo Nevado está clasificado como una especie amenazada, es esencial el trabajar para el beneficio de su protección. Los ataques adversarios exponen las vulnerabilidades de muchas redes neuronales que son usadas para propósitos de conservación de la vida animal. Estas redes neuronales han sido embebidas en sistemas de monitoreo como cámaras trampa para detección y conteo de vida salvaje (Norouzzadeh *et al.*, 2018), drones para escaneo de hábitat (Doull *et al.*, 2021) e incluso dispositivos para análisis de sonido como cantos de aves (Kahl *et al.*, 2021). Muchas de estas aplicaciones involucran el defenderse contra individuos que realizan actividades ilegales que ponen en peligro a la vida salvaje como la cacería, que se ha convertido un problema creciente en la actualidad y un tema para lo cual los conservacionistas necesitan desesperadamente una solución. Otras

actividades que pueden convertirse en amenaza para el bienestar de las especies es el incremento constante en la población humana y el disturbio que ejercen en el hábitat natural de un animal, como las playas donde el Chorlo Nevado y otras especies de aves playeras habitan. Al exponer las vulnerabilidades de estas redes neuronales, la comunidad científica se puede enfocar en mejorar la robustez de sus algoritmos para prevenir ataques por individuos maliciosos.

En este trabajo se implementan varios métodos para generar ejemplos adversarios. Entre estos, se implementan unos de caja blanca y otros de caja negra. Estos métodos se describen a continuación.

4.2.1. Método rápido de gradiente

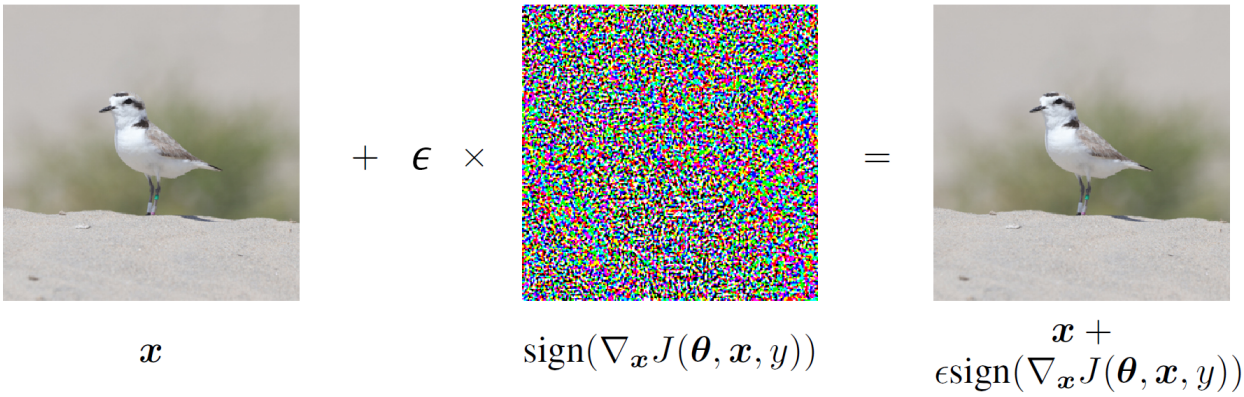


Figura 14. Ejemplo de la generación de un ejemplo adversario con FGSM usando una imagen de la base de datos SNPL. Al hacer detección con la imagen original, una red neuronal determinada obtiene un valor F de 90 %. Tras generar el ejemplo adversario, su valor baja a 50 %. Entre más alto el valor de épsilon ϵ, más notoria la perturbación.

Goodfellow *et al.* (2015) sugirieron una manera rápida de generar ejemplos adversarios aprovechando la linealidad de estos. La gran mayoría de las redes neuronales están intencionalmente diseñadas para comportarse en maneras muy lineales y así sean muy fáciles de optimizar. Este comportamiento sugiere que perturbaciones simples de un modelo lineal también deberían dañar a las redes neuronales. Este método de caja blanca, llamado en inglés *Fast Gradient Sign Method* (FGSM) está diseñado para atacar redes neuronales aprovechando la manera en que éstas aprenden; gradientes. En lugar de minimizar la pérdida mediante el ajuste de los pesos usando

retro-propagación, el ataque ajusta los datos de entrada para maximizar la pérdida basándose en los mismos gradientes. En otras palabras, este método usa el gradiente de la pérdida en relación con los datos de entrada, y después ajusta los datos de entrada para maximizar la pérdida. La ecuación para el FGSM se define como:

$$\eta = \epsilon \text{ sign } (\nabla_x J(\theta, x, y)) \tag{16}$$

donde θ son los parámetros del modelo, x los datos de entrada al modelo, y los objetivos asociados a x (para tareas de aprendizaje de máquina con objetivos), y $J(\theta, x, y)$ la pérdida o *loss* usada para entrenar a la red neuronal. El ataque retropropaga el gradiente de vuelta a los datos de entrada para calcular $\nabla_x J(\theta, x, y)$. Después, ajusta los datos de entrada por un valor ϵ $\{0 \leq \epsilon \leq 1\}$ en la dirección que maximice la pérdida, obteniendo una perturbación óptima η. Un ejemplo de este proceso se muestra en la figura 14.

El valor ϵ determina la intensidad de la perturbación en el ejemplo adversario. Para un valor $\epsilon = 0$, la perturbación es inexistente y η es equivalente a los datos de entrada originales. Para $\epsilon = 1$, la perturbación es total y la información de los datos de entrada originales se pierde. Debido a esto, y a que una de las propiedades principales de los ejemplos adversarios es que sus perturbaciones sean imperceptibles al ojo humano, se debe elegir un ϵ adecuado para engañar a la red neuronal pero que la perturbación no sea notoria en el ejemplo adversario.

En este trabajo, dado que los datos de entrada son imágenes en el espectro de colores RGB con una resolución radiométrica de 8 bits, se usan los valores de los pixeles como datos de entrada para obtener los ejemplos adversarios. Por esto mismo, se establece a ϵ como un valor entre un conjunto de valores establecidos $\epsilon = \{2, 4, 8, 16, 32, 64\}$. Al dividir cada uno de estos valores entre el valor máximo de un pixel en una imagen de 8 bits (255), se obtiene un valor entre 0 y 1. Comúnmente, la perturbación es imperceptible en los valores de $\epsilon = \{2, 4\}$. A partir del $\epsilon = 8$, la intensidad de la perturbación es suficientemente grande para ser notoria.

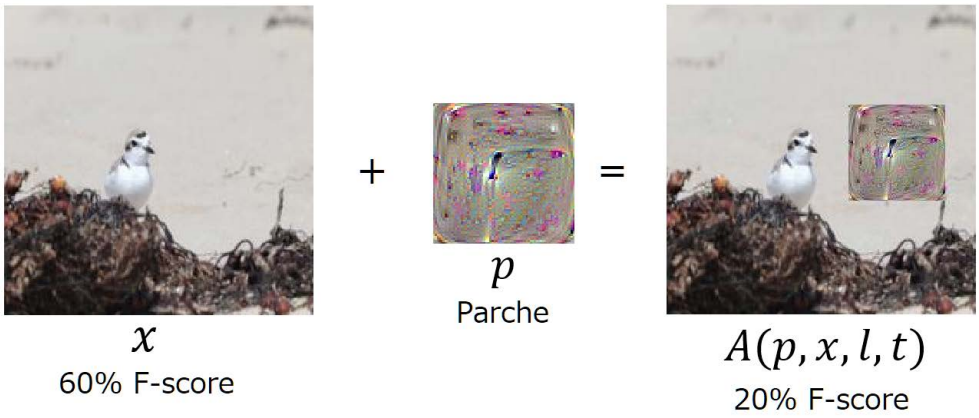


Figura 15. Ejemplo de un parche *p* colocado en una imagen *x* de la base de datos SNPL. Al alimentar la imagen sin el parche a una red determinada, ésta logra la detección con un valor de F de 60 %. Al colocar el parche en una localización *l* de la imagen original y aplicarle una transformación *t*, su detección disminuye a un valor de F de 20 %.

4.2.2. Parche adversario

A diferencia del FGSM, donde los ejemplos adversarios se enfocan en pequeñas e imperceptibles perturbaciones a los datos de entrada, el método del parche adversario propuesto por Brown *et al.* (2017) explora los efectos de un ejemplo adversario que no está restringido a cambios imperceptibles en los datos. En este método se genera un parche independiente a la imagen que es altamente saliente para una red neuronal. Es decir, al generar y colocar este parche sobre una imagen de entrada, la red neuronal es incapaz de determinar su naturaleza y lo clasifica de la misma manera que un objeto sobresaliente. Este parche puede ser puesto en cualquier localización en la imagen, y en tareas de clasificación causa que el modelo la clasifique como una clase en especifico. Además este ataque es independiente de la escena, lo que permite al atacante crear un ataque del mundo real sin tener conocimiento previo de las condiciones de luz, ángulo de la cámara o el tipo de clasificador que está siendo atacado.

Este tipo de ataque adversario tiene gran relevancia ya que el atacante no necesita saber en qué imágenes o base de datos se están generando los ejemplos adversarios para generar un parche. Tras su generación, el parche puede ser distribuido por el internet e impreso para que otros atacantes lo utilicen. Adicionalmente, ya que el parche es una perturbación muy grande, la efectividad de las redes neuronales que llegan a defenderse contra pequeñas perturbaciones son severamente afectadas por grandes perturbaciones como el parche.

En el procedimiento para generar un ejemplo adversario, se reemplaza por completo una parte de la imagen con el parche. Se enmascara el parche para permitirle tomar cualquier forma y se entrena sobre una variedad de imágenes, aplicando transformaciones aleatorias como rotación, traslación, y escala en cada imagen, siempre optimizando el parche usando descenso de gradiente. En particular, para una imagen dada $x \in \mathbb{R}^{w \times h \times c}$, un parche p, una localización de parche l, y transformaciones de parche t, se define un operador de aplicación de parche $A(p, x, l, t)$ que primero aplica la transformación t al parche p, y luego aplica el parche transformado p a la imagen x en la localización l. Un ejemplo de la aplicación de un parche se puede observar en la figura 15.

Para obtener el parche entrenado $\hat{p}$, Brown *et al.* (2017) usan una variante de la framework Expectation over Transformation (EOT) por Athalye *et al.* (2018). En particular, el parche es entrenado para optimizar la función objetivo:

$$\hat{p} = \mathbb{E}_{x \sim X, t \sim T, l \sim L}[\, log Pr(\hat{y} | A(p, x, l, t)] \tag{17}$$

donde X es el conjunto de imágenes de entrenamiento, T es una distribución sobre transformaciones del parche, y L es una distribución sobre localizaciones en la imagen. Este método se aplica sobre imágenes, lo cual causa que el parche funcione independientemente de lo que se encuentre en el fondo de la imagen. Además, la sobresaliencia del parche producido es mucho mayor que cualquier otro objeto en la escena, por lo que al atacar modelos de detección de objetos sobresalientes o de clasificación, la atención de estos modelos tienden a siempre centrarse en el parche.

En este trabajo se demuestra como tres diferentes redes neuronales del estado del arte para la detección de objetos sobresalientes son afectadas por los parches adversarios, y se compara su rendimiento con el modelo de la programación cerebral. Además, al crear los parches adversarios en una de las redes neuronales e implementarlos en nuestras bases de datos de prueba, se comprueba que existe transferibilidad. Es decir, los ejemplos adversarios creados a partir de una red afectan el rendimiento de las demás redes de nuestros experimentos.

4.2.3. Ataque de un-pixel

El ataque de un-pixel, (en inglés, *One-pixel Attack*) es un tipo de ataque adversario que, al igual que el parche adversario, se desvía de la manera tradicional de generar ejemplos adversarios mediante la alteración imperceptible de los pixeles en la imagen de entrada. El método descrito por Su *et al.* (2019) implementa un ataque donde un solo pixel de la imagen de entrada puede ser modificado. Éste es un método donde no se necesita conocer la arquitectura de la red a atacar (caja negra), y se basa en computación evolutiva.

La generación de ejemplos o imágenes adversarias puede ser formalizado como un problema de optimización con ciertas limitantes. Un conjunto de datos de entrada (la imagen) puede ser representada por un vector en el que cada elemento escalar corresponde a un pixel. Siendo f el clasificador que recibe los datos de entrada y x la imagen original a clasificar, el vector $e(x)$ es una perturbación adversaria que depende de x, la clase objetivo adv y una limitante L que limita la perturbación adversaria $e(x)$ hasta una modificación máxima. L siempre es medido por la longitud del vector $e(x)$. El objetivo de los ejemplos adversarios es encontrar la solución óptima $e(x)^*$ para la siguiente cuestión:

$$\max f_{adv}(x + e(x)) \text{ sujeto a } ||e(x)|| \leq L \tag{18}$$

La ecuación para generar el ejemplo adversario con un-pixel es un tanto diferente:

$$\max f_{adv}(x + e(x)) \text{ sujeto a } ||e(x)||_0 \leq d \tag{19}$$

donde $d = 1$.

El tamaño de la imagen de entrada requerido para poder engañar a las redes neuronales con suficiente confianza usando un-pixel es demasiado pequeño en comparación con las imágenes usadas en otros métodos de generación de ejemplos adversarios. En la experimentación de Su *et al.* (2019), encontraron que la modificación de un solo pixel en una imagen de 32x32 pixeles es suficiente para engañar a tres diferentes

redes neuronales utilizadas en su experimentación. Sin embargo, el tamaño de estas imágenes posee una desventaja; no son lo suficientemente grandes para simular un caso del mundo real. Es decir, si el propósito de los ataques adversarios es engañar a las redes neuronales, los sistemas de la vida real que trabajan con estas redes por lo general no utilizan imágenes tan pequeñas, ya que las escenas del mundo real llegan a contener una vasta cantidad de objetos y trabajan con imágenes de mayor tamaño para poder observar los detalles. Debido a esta desventaja en el ataque un-pixel, en este trabajo se necesitó utilizar un método similar pero que demostrara efectividad a la hora de llevar a cabo los ataques usando ejemplos adversarios de mayor tamaño. La solución es un ataque multipixel.

4.2.4. Ataque multipixel

Figura 16. Ejemplos adversarios usando el ataque multipixel con $d = 10,000$ para cada imagen. En comparación a un-pixel, la perturbación es mucho más notoria debido a la gran cantidad de pixeles modificados.

El ataque *multipixel* es una variante del ataque un-pixel en donde más de un pixel es modificado en la imagen de entrada para generar el ejemplo adversario. En el ataque un-pixel, el objetivo es encontrar la solución óptima que resuelva la ecuación 19. En este caso, $d = 1$. En el caso del ataque multipixel, el número de pixeles puede ser extendido mediante el incremento del valor de d, y múltiples pixeles son requeridos para generar ejemplos adversarios efectivos en imágenes de entrada de mayor tamaño. También se debe notar que debido a la mayor cantidad de pixeles en este tipo de ataque, se incrementa el riesgo de que la perturbación sea más notoria. Un ejemplo

del ataque multipixel se puede observar en la figura 16.

En este trabajo se experimenta en tres tipos de ataques adversarios que se acaban de mencionar: FGSM, Parche Adversario y Multipixel. El conjunto de validación de las cuatro bases de datos usadas en este trabajo es convertido en ejemplos adversarios; los ejemplos adversarios con FGSM y Parche Adversario se generan a partir de la etapa de entrenamiento de una red neuronal del estado del arte, y esta misma red es atacada con los ejemplos adversarios generados (un proceso antes mencionado como ataque de caja blanca), y también se realiza el ataque a otras dos redes neuronales y al algoritmo de la programación cerebral. Para el caso de Multipixel, estos ejemplos adversarios fueron generados sin involucrar el proceso de la red neuronal y se atacó a las tres redes y a la programación cerebral usando este ataque de caja negra.

4.2.5. Ruido

En la sección anterior, se menciona que varios de los ejemplos adversarios dependen de la arquitectura de las redes neuronales para generarlos. Sin embargo, existe otra manera de generar ejemplos adversarios sin depender del proceso de entrenamiento de una red neuronal. Esto involucra la inserción de ruido; señales que perturban la información de una imagen y que generalmente provienen de una señal aleatoria y que pueden llegar a destruir la mayoría de la información de una imagen. Con el avance en la tecnología, se incrementa la demanda por obtener fotografías de mayor precisión y calidad. Sin embargo, las imágenes obtenidas por las cámaras modernas son inevitablemente degradadas por ruido, lo que deteriora su calidad visual. Estos ruidos pueden surgir de múltiples maneras al momento de generar y procesar una fotografía. Entre estos se encuentran sensores dañados, lentes desalineados, mala distancia focal, compresión de imagen, entre otros. Con la presencia de ruido, tareas de procesamiento de imagen como procesamiento de video, análisis de imagen y rastreo son gravemente afectadas.

Debido a la frecuencia de aparición de estos ruidos, también surge la necesidad de eliminar el ruido pero sin perder las características de la imagen (orillas, esquinas, y otras estructuras de alta precisión en la imagen). Es por eso que han surgido trabajos con varios métodos propuestos para eliminar el ruido en imágenes (Fan *et al.*,

2019). Sin embargo, dado que se ha demostrado que las redes neuronales no poseen la suficiente robustez para clasificar imágenes perturbadas mediante ejemplos adversarios generados en base a su propia arquitectura, se tiene la necesidad de calcular su robustez usando un método de perturbación aleatoria, tales como alimentarlos con imágenes con ruido Gaussiano, ruido Sal y Pimienta o ruido Speckle. Entre la gran variedad de ruidos existentes, en este trabajo se realizan experimentos con estos tres.

4.2.5.1. Ruido gaussiano

Figura 17. Ejemplo de ruido Gaussiano aplicado a imágenes de la base de datos SNPL. La imágenes originales se muestran del lado izquierdo, y las imágenes con ruido Gaussiano a la derecha. La intensidad del ruido puede ser modificada dependiendo de la desviación estándar en la función de densidad probabilística.

El ruido Gaussiano es uno de los tipos de ruido que impacta negativamente la percepción de una imagen. Este tipo de ruido es un canal no deseado que es capturado durante la adquisición de ondas de luz en una cámara. También llamado granulado, el ruido Gaussiano representa un ruido estadístico que tiene una función de densidad probabilística (PDF), similar a la de la distribución normal (Barbu, 2013). La función de densidad probabilística P de una variable aleatoria Gaussiana Z se da como:

$$PG(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(Z-\mu)^2}{2\sigma^2}} \tag{20}$$

donde z representa el nivel de gris, μ el valor promedio de la distribución y σ la desviación estándar. La intensidad del la perturbación Gaussiana en una imagen depende de la desviación estándar. Un ejemplo de imágenes con ruido Gaussiano se puede

observar en la figura 17. El ruido Gaussiano puede surgir en el momento de la adquisición de la imagen, como ruido en el sensor causado por malas condiciones de luz o alta temperatura, además de efectos de transmisión como ruido de circuito electrónico. Filtros espaciales pueden ser usados para reducir el efecto del ruido Gaussiano en el procesamiento digital de imágenes, aunque, al momento de suavizar una imagen, un efecto no deseado puede resultar en el suavizado de orillas y demás detalles de la imagen, ya que estos componentes pertenecen a altas frecuencias en la imagen, las cuales son filtradas durante un proceso de suavizado.

4.2.5.2. Ruido sal y pimienta

Figura 18. Ejemplo de ruido Sal y Pimienta aplicado a imágenes de la base de datos SNPL. La perturbación se manifiesta como granulado blanco y negro.

El ruido Sal y Pimienta es causado debido a perturbaciones agudas y repentinas en los valores grises de la imagen. Su apariencia son valores blancos y negros aleatoriamente distribuidos sobre la imagen. La función de densidad probabilística del ruido sal y pimienta se da como:

$$P(z) = \begin{cases} Pa, \text{ cuando } z = a \\ Pb, \text{ cuando } z = b \\ 0, \text{ de otro modo} \end{cases} \tag{21}$$

Si $b > a$, el nivel de gris b aparece como un punto blanco en la imagen (sal). De otro modo, a aparece como un punto negro (pimienta). Si Pa o Pb es cero, la función

de densidad probabilística es llamada unipolar. Ya que la perturbación del ruido sal y pimienta es generalmente grande en comparación con la potencia de la señal, se asume que a y b son digitalizados como valores saturados, y es por eso que surgen como valores blanco y negro. Un ejemplo de imágenes con ruido Sal y Pimienta se puede observar en la figura 18.

4.2.5.3. Ruido speckle

Figura 19. Ejemplo de ruido Speckle aplicado a imágenes de la base de datos SNPL.

El ruido Speckle es un tipo de ruido que surge de varios métodos de generación de imagen. Un ejemplo es el radar de apertura sintética (SAR), un tipo de recolección de datos en sensado remoto donde un sensor produce su propia energía y graba la cantidad de energía reflejada después de interactuar con la tierra. Los datos en SAR requieren un tipo de interpretación en donde la señal es responsiva a las características superficiales de la tierra como la estructura y humedad (Meyer, 2019).

Estos métodos dependen de un proceso llamado *coherent imaging*. La desventaja de este proceso es que las imágenes sufren de un ruido granulado que es típicamente llamado Speckle, o ruido multiplicativo. La razón para la manifestación de este granulado es que, en aplicaciones que utilizan estos métodos, las superficies que reflejan las ondas coherentes son rugosas, cuando se consideran a una resolución comparable a la longitud de onda de la señal. Entonces, en su punto de detección, los rayos de luz correspondientes a algunos pixeles se acumularán constructivamente, mientras que otros se acumularán destructivamente. Este fenómeno se modela como un ruido multiplicativo. Un ejemplo de imágenes con ruido Speckle se observa en la figura 19.

Capítulo 5. Experimentación

En esta sección se describe el proceso de experimentación de este trabajo. Se describen tres bases de datos utilizadas y el proceso realizado para la creación, compilación y utilización de una cuarta base de datos. Estas cuatro bases de datos son utilizadas en tres algoritmos de detección de objetos sobresalientes usando redes neuronales convolucionales: DHSNet (Liu y Han, 2016), PiCANet (Liu *et al.*, 2018), BASNet (Qin *et al.*, 2019), y el algoritmo evolucionado con programación cerebral GBVS-BP (Dozal *et al.*, 2014). Para probar el rendimiento de estos algoritmos en la detección, se utiliza la función fitness o medida-F.

5.1. Bases de datos

En este trabajo se experimenta con cuatro diferentes bases de datos. Cada una de esas bases de datos alberga diferente cantidad de imágenes, diferente tamaño y muestran diferentes objetos. También, cada base de datos posee su conjunto espejo de imágenes binarias manualmente segmentadas (ground-truth), también llamadas máscaras. Para llevar a cabo el entrenamiento y las pruebas con los diversos algoritmos de detección, cada base de datos se dividió en dos subconjuntos en una proporción 70/30, 70 % de las imágenes se usaron para la etapa de entrenamiento, y el 30 % se usaron para la etapa de validación. Además, cada una de las imágenes tiene un tamaño original, pero para propósitos de experimentación, el tamaño de las imágenes fue reducido. Cada una de estas bases de datos se describe a continuación:

5.1.1. Base de datos FT

La base de datos FT (Achanta *et al.*, 2009) es un conjunto de 1000 imágenes que muestran objetos vistos en la vida diaria como flora, fauna (aves, insectos, mamíferos, etc), medios de transporte como autos, aviones, botes, señalamientos de transito y otros letreros, personas, edificios, objetos de uso cotidiano, entre otros (véase figura 20). La mayoría de las imágenes de esta base de datos muestran a los objetos en un tamaño grande respecto a la imagen, por lo que abarcan una gran región sobresaliente. Esto puede implicar que los algoritmos de detección de objetos sobresalientes

como las redes neuronales tengan una gran facilidad para detectar estos objetos, arrojando resultados bastante favorables en la experimentación. Se debe mencionar que en casos del mundo real, los objetos contenidos dentro de imágenes capturadas no siempre presentarían este gran tamaño en la escena, por lo que esta base de datos puede no ser adecuada para ayudar a resolver problemas del mundo real.

- Total de imágenes: 1000
- Conjunto de entrenamiento: 700
- Conjunto de validación: 300
- Tamaño: 224x224 pixeles.
- Segmentación manual (ground-truth) para todas las imágenes, ya incluidas.

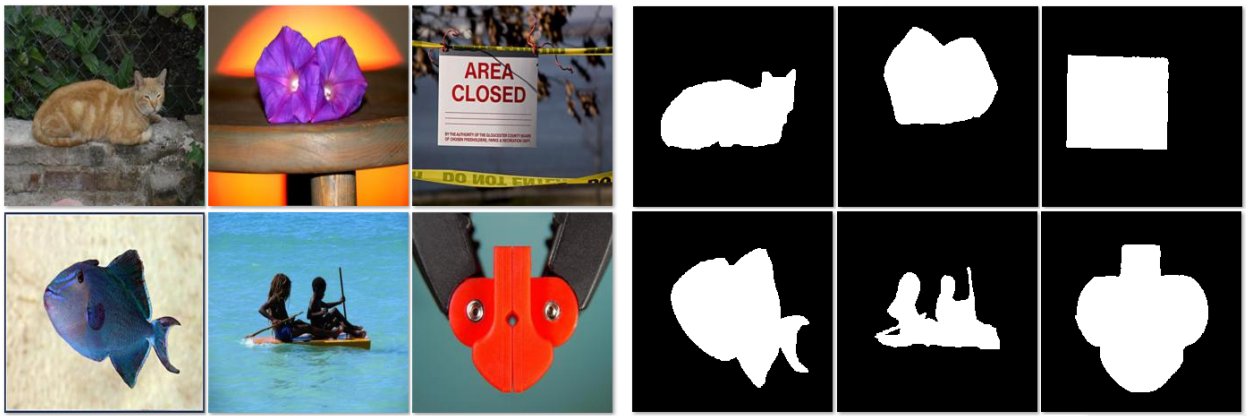


Figura 20. Extracto de imágenes de la base de datos FT con su segmentación manual o ground-truth.

5.1.2. Base de datos ImgSal

La base de datos ImgSal (Li *et al.*, 2013) es un conjunto de 235 imágenes que muestran diversos objetos como flora, fauna, personas y escenas panorámicas que incluyen uno o varios de estos objetos (véase figura 21). La base de datos se divide en seis diferentes categorías: 50 imágenes con regiones sobresalientes grandes, 80 imágenes con regiones sobresalientes intermedias, 60 imágenes con regiones sobresalientes pequeñas, 15 imágenes con fondos desordenados, 15 imágenes con distractores y 15

imágenes con regiones sobresalientes grandes y pequeñas. En la experimentación, se descubrió que el conjunto de imágenes binarias que corresponden a la segmentación manual de esta base de datos no fue correctamente segmentada por la persona responsable de llevar a cabo esta tarea, por lo que esto afecta los resultados al momento de calcular los valores de precisión y recuerdo.

- Total de imágenes: 235
- Conjunto de entrenamiento: 165
- Conjunto de validación: 70
- Tamaño: 224x224 pixeles.
- Segmentación manual (ground-truth) para todas las imágenes, ya incluidas.

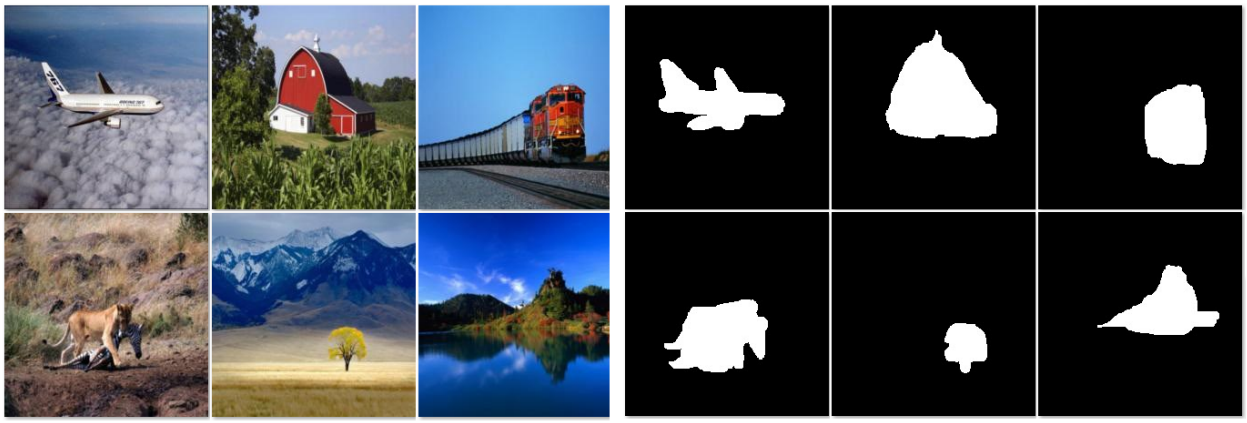


Figura 21. Extracto de imágenes de la base de datos ImgSal con su segmentación manual o ground-truth.

5.1.3. Base de datos PASCAL

La base de datos PASCAL (Li *et al.*, 2014) es un conjunto de 850 imágenes que muestran objetos como flora, fauna, medios de transporte, personas, letreros, entre otros (véase figura 22). Los objetos en la escena poseen un tamaño variable, y múltiples objetos sobresalientes pueden aparecer en una imagen. Esta base de datos mantiene un balance entre el tamaño de los objetos sobresalientes y su conjunto de ground-truths está bien segmentado.

- Total de imágenes: 850
- Conjunto de entrenamiento: 595
- Conjunto de validación: 255
- Tamaño: 224x224 pixeles.
- Segmentación manual (ground-truth) para todas las imágenes, ya incluidas.

Figura 22. Extracto de imágenes de la base de datos PASCAL con su segmentación manual o ground-truth.

5.1.4. Base de datos SNPL

La base de datos SNPL (acrónimo de Snowy Plover, inglés para Chorlo Nevado) es un conjunto de 250 imágenes del ave playera Chorlo Nevado (*Charadrius nivosus*), uno de los objetos de estudio principales de este trabajo (véase figura 23). Cada una de las 250 fotografías fueron capturadas en las playas o planicies lodosas de Ensenada y San Quintín con cámaras fotográficas comunes y con diferentes condiciones de clima e iluminación. Las imágenes muestran a uno o más chorlos, donde el tamaño de estos en la escena varía dependiendo de la distancia a la que la fotografía fue tomada, o la distancia focal del lente. Otra característica de estas imágenes es que el fondo contiene una variedad de distractores como plantas, agua, y objetos como conchas o basuras pequeñas. También, varias incluyen obstrucciones que cubren parcialmente al objeto de interés. Por último, ya que el chorlo nevado habita comúnmente en las playas, éste posee un plumaje de una tonalidad similar a la arena, por lo que una gran

proporción de las imágenes carecen de alto contraste entre el fondo y el objeto de interés. Gracias a estas propiedades, la base de datos SNPL puede ser utilizada para estudiar y resolver un problema del mundo real. En este trabajo, la base de datos SNPL fue propuesta para comparar el rendimiento de los algoritmos de detección usando un problema del mundo real, a comparación de las bases de datos mencionadas anteriormente, las cuales son diseñadas exclusivamente para propósitos académicos o para evaluar algún algoritmo novedoso.

- Total de imágenes: 250
- Conjunto de entrenamiento: 175
- Conjunto de validación: 75
- Tamaño: 224x224 pixeles.
- Segmentación manual (ground-truth) para todas las imágenes. Segmentación realizada por el autor de esta tesis.

Figura 23. Extracto de imágenes de la base de datos SNPL con su segmentación manual o ground-truth.

5.2. Generación de ejemplos adversarios

Con el objetivo de probar la robustez o resistencia a los ataques adversarios en la programación cerebral, y compararlos con el rendimiento de las redes neuronales, se generaron ejemplos adversarios para el conjunto de validación de cada base de

datos usada en este trabajo. Estos se generaron mediante el lenguaje de programación Python, versión 3.8.3, y usando la framework de aprendizaje de máquina PyTorch, versión 1.6.0. Aquí se describe el proceso para generar cada uno de ellos.

5.2.1. Ejemplos adversarios con FGSM

Para el conjunto de validación de cada base de datos, se crearon seis conjuntos de ejemplos adversarios usando el método FGSM, usando la etapa de entrenamiento de la red neuronal PiCANet (Liu *et al.*, 2018). Cada uno de los seis conjuntos fue generado con un valor de épsilon ϵ específico, por lo que las imágenes de cada conjunto tienen la perturbación con la intensidad diferente. Los valores de épsilon utilizados son: 2, 4, 8, 16, 32 y 64, siendo 2 la perturbación más baja y 64 la más alta. La perturbación generada en cada valor de épsilon se aprecia en la figura 24.

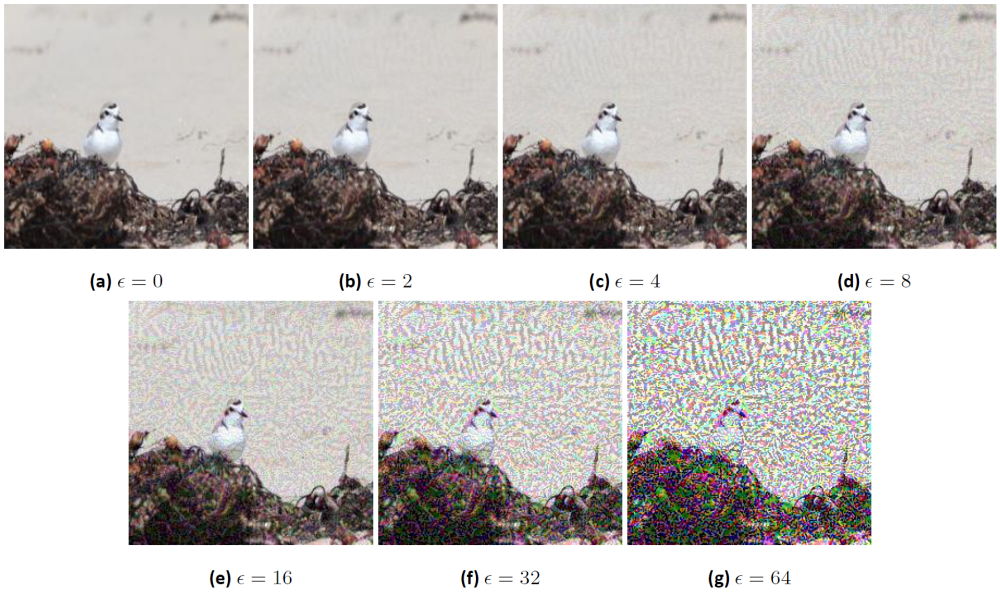
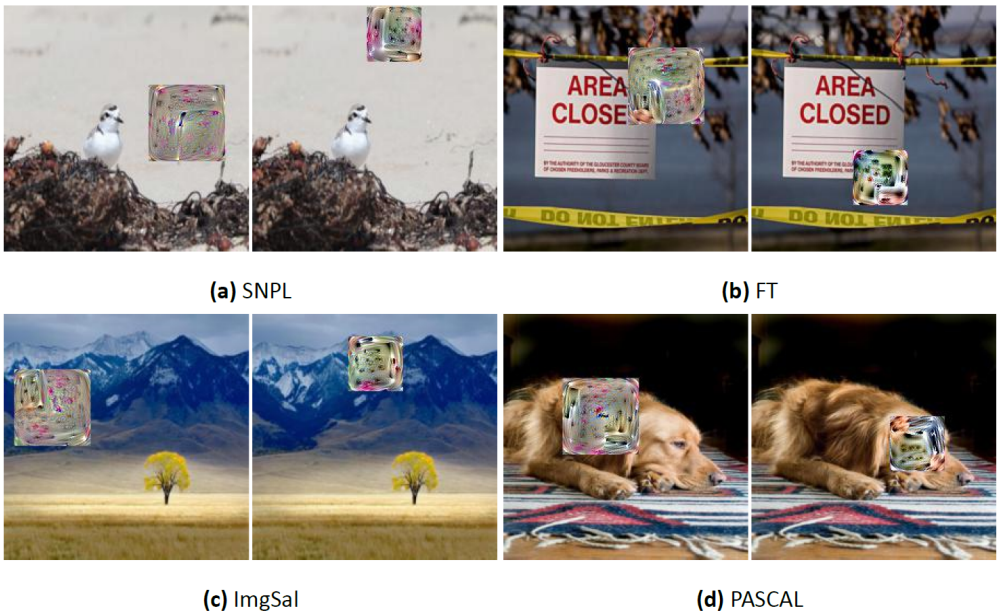


Figura 24. Ejemplos de FGSM usando la base de datos SNPL. Entre más grande el valor de épsilon, más notoria la perturbación. El valor $\epsilon = 0$ equivale a la imagen sin perturbación.

5.2.2. Ejemplos adversarios con parche adversario

Para el conjunto de validación de cada base de datos, se crearon dos conjuntos de ejemplos adversarios usando el método del Parche Adversario descrito por Brown *et al.* (2017), y la red neuronal utilizada para la generación es la ResNet-50 (He *et al.*, 2016), también usada por los mismos autores. Entre los dos conjuntos, uno de ellos contiene

las imágenes donde el parche abarca un tamaño de 70x70 pixeles, o cubriendo el 31 % de la imagen original. El otro conjunto contiene las imágenes donde el parche abarca un tamaño de 50x50 pixeles, cubriendo el 22 % de la imagen original. La razón para haber generado dos conjuntos con parches de tamaños diferentes fue para evaluar si el rendimiento de los algoritmos de detección depende del tamaño del parche, el cuánto varían los resultados, o si al final depende de la pura perturbación independientemente del tamaño. El parche que corresponde a cada imagen fue colocado en una región aleatoria, donde no se discrimina su ubicación aún cuando el parche termine cubriendo parcial o totalmente el objeto de interés. Ejemplos de parches de ambos tamaños se aprecian en la figura 25.



(a) SNPL **(b)** FT

(c) ImgSal **(d)** PASCAL

Figura 25. Ejemplos de Parche Adversario en las cuatro bases de datos, incluyendo el conjunto de parches de menor tamaño. Debido a la ubicación aleatoria del parche, este puede llegar a cubrir parte del objeto de interés.

5.2.3. Ejemplos adversarios con multipixel

Para el conjunto de validación de cada base de datos, se creó un conjunto de ejemplos adversarios usando el método Multipixel. Para este tipo de perturbación, se estableció un valor de $d = 10,000$, recordando que d define la cantidad de pixeles modificados en la imagen. Esto resulta en ejemplos adversarios con una cantidad decente de pixeles perturbados, donde la perturbación no es inconspicua, pero es suficiente para permitir cualquier objeto sobresaliente en la escena siga siendo notorio (véase figura

26). Para generar estos ejemplos adversarios no se requirió conocer los parámetros internos de ninguna red neuronal.

(a) SNPL **(b)** FT **(c)** ImgSal **(d)** PASCAL

Figura 26. Ejemplos de Multipixel en las cuatro bases de datos.

5.2.4. Ejemplos adversarios con ruido gaussiano

Para el conjunto de validación de cada base de datos, se creó un conjunto de ejemplos adversarios usando ruido Gaussiano. Para cada uno de los ejemplos generados, se definió una desviación estándar de $\sigma = 30$, lo cual implementa un granulado de una magnitud suficiente para poder distinguir sin problema los objetos en la escena (figura 27). La generación de estos ejemplos no requirió conocer los parámetros internos de ninguna red neuronal.

(a) SNPL **(b)** FT **(c)** ImgSal **(d)** PASCAL

Figura 27. Ejemplos de ruido Gaussiano en las cuatro bases de datos.

5.2.5. Ejemplos adversarios con ruido sal y pimienta

Para el conjunto de validación de cada base de datos, se creó un conjunto de ejemplos adversarios usando ruido Sal y Pimienta. En este trabajo, el ruido sal y pimienta no es agregado a imágenes en blanco y negro, sino a imágenes a color, por lo que el ruido no se presenta como puntos blancos y negros, sino como puntos de color. Esto

es debido a que el ruido sal y pimienta toma los valores de los pixeles máximos y mínimos de cada banda de color y los distribuye por la imagen (figura 28). La generación del ruido sal y pimienta no requirió conocer los parámetros internos de ninguna red neuronal.

(a) SNPL **(b)** FT **(c)** ImgSal **(d)** PASCAL

Figura 28. Ejemplos de ruido Sal y Pimienta en las cuatro bases de datos.

5.2.6. Ejemplos adversarios con ruido speckle

Para el conjunto de validación de cada base de datos, se creó un conjunto de ejemplos adversarios usando ruido Speckle. Ya que el ruido Speckle se manifiesta solamente debido a condiciones físicas al momento de la generación de la imagen, fue necesario implementar un método para simularlo. Para llevar esto a cabo, se tomaron valores aleatorios de una distribución gaussiana a partir de las dimensiones de cada imagen, y se multiplicaron por una varianza de 0.3. El resultado fue un conjunto de imágenes con una perturbación que simula el ruido Speckle (figura 29). Al igual que los otros ruidos, el ruido Speckle no requirió conocer los parámetros de ninguna red neuronal para implementarlo.

(a) SNPL **(b)** FT **(c)** ImgSal **(d)** PASCAL

Figura 29. Ejemplos de ruido Speckle en las cuatro bases de datos.

Capítulo 6. Resultados

En esta sección se describen los resultados obtenidos en la experimentación. Las pruebas con cada base de datos se divide en dos tablas: una para el ataque adversario FGSM, y otra para el Parche Adversario, Multipixel y los tres tipos de ruido. Los valores en negrita corresponden al puntaje más bajo que un algoritmo obtuvo entre todos los tipos de ataques adversarios.

El proceso para obtener la medida F en cada experimento es el siguiente: en cada corrida, a cada imagen del conjunto de validación de la base da datos se le calcula su precisión y recuerdo basado en un número n de umbrales. Después, se calcula la medida F de cada uno de estos resultados en base al mapa de sobresaliencia generado y empalmándola con su imagen ground-truth, como se mencionó en la sección 3.3. Tras esto, se toma el valor máximo, por lo que se obtiene el mejor valor F para cada imagen. Después, se calcula la media de los valores F de todas las imágenes, para así obtener el valor F final de acuerdo a la ecuación 9.

6.1. FT con FGSM

Tabla 3. Resultados del ataque adversario FGSM usando la base de datos FT. El valor ϵ controla la intensidad de la perturbación en la imagen.

FT								
Algoritmo	Medida	Original	$\epsilon = 2$	$\epsilon = 4$	$\epsilon = 8$	$\epsilon = 16$	$\epsilon = 32$	$\epsilon = 64$
GBVS-BP	Avg.	73.84	73.79	73.63	73.69	73.44	72.97	71.64
	σ	12.48	12.53	12.52	12.53	12.56	12.70	13.68
BASNet	Avg.	92.67	90.92	89.74	88.92	87.30	82.60	**58.65**
	σ	12.51	15.38	16.94	16.78	18.02	23.28	37.96
PiCANet	Avg.	81.45	74.26	67.84	61.01	57.04	51.21	**40.52**
	σ	15.27	18.74	20.83	22.58	22.91	22.76	22.19
DHSNet	Avg.	88.63	87.92	87.46	86.68	85.62	83.85	78.92
	σ	12.43	12.86	13.30	13.74	14.48	15.95	19.90

La tabla 3 muestra los resultados al aplicar un ataque adversario FGSM a las tres redes neuronales y al algoritmo de la programación cerebral usando la base de datos FT. En las pruebas con las imágenes originales (sin ataque adversario), las redes neuronales superan al algoritmo de la programación cerebral GBVS-BP. Sin embargo, a medida que se aplica el ataque y se incrementa el valor de épsilon, el rendimiento de las redes comienza a bajar. La red PiCANet es la más afectada, comenzando con un

puntaje de 81.45 % y bajando constantemente hasta 40.52 % en $\epsilon = 64$, perdiendo asi alrededor del 50 % de su rendimiento. La red BASNet, aunque tiene el mejor puntaje en la prueba con las imágenes originales con 92.67 %, también pierde su rendimiento y baja hasta 58.66 %. La red DHSNet, aunque no baja su puntaje de manera tan drástica, también se ve afectada. Por otro lado, el algoritmo de la programación cerebral GBVS-BP se ve prácticamente inafectado durante todos los valores de épsilon, lo que indica que posee una gran robustez y es bastante resistente a este tipo de ataques, siendo capaz de detectar objetos sobresalientes en la escena aún con la perturbación más intensa, a diferencia de las redes neuronales. Cabe mencionar que la base de datos FT se compone de imágenes cuyos objetos de interés cubren una gran parte de la imagen, por lo que estos algoritmos tienden a tener una gran facilidad para lograr la detección y esto se ve reflejado en los altos puntajes de esta tabla.

6.2. FT con parche adversario, multipixel y ruido

Tabla 4. Resultados de los ataques adversarios Multipixel, Parche Adversario y tres tipos de ruido usando la base de datos FT. La columna Parche(S) representa los resultados al aplicar un parche más pequeño a la imagen.

FT								
Algoritmo	Medida	Orig.	Multipixel	Parche	Parche(S)	Gauss	S&P	Speckle
GBVS-BP	Avg.	73.84	74.31	**67.96**	69.63	73.84	73.90	73.96
	σ	12.48	12.23	14.02	14.06	12.28	12.60	12.35
BASNet	Avg.	92.67	90.79	67.01	73.00	92.27	92.41	90.88
	σ	12.51	15.54	20.47	23.16	12.72	12.69	14.75
PiCANet	Avg.	81.45	66.86	61.04	69.78	78.39	78.95	75.89
	σ	15.27	22.14	19.79	17.06	17.42	19.40	20.97
DHSNet	Avg.	88.63	85.67	**74.48**	79.04	87.84	86.98	86.89
	σ	12.43	14.50	17.67	16.38	13.31	14.12	14.73

La tabla 4 muestra los resultados al aplicar los ataques del Parche Adversario, Multipixel y ruido a las tres redes neuronales y al algoritmo de la programación cerebral usando la base de datos FT. Se puede notar que el parche adversario causa una disminución de puntaje en todos los algoritmos de detección, incluyendo la programación cerebral. Sin embargo, esta disminución es más severa en las redes BASNet y PiCANet, que habían mostrado un puntaje alto con las imágenes no perturbadas. PiCANet es especialmente afectada por el ataque Multipixel, bajando hasta 66.86 %. La red DHSNet, que se había mostrado inafectada en el ataque FGSM, finalmente pierde puntaje con el parche adversario, bajando hasta 74.48 %. Por otro lado, el algoritmo de

la programación cerebral GBVS-BP muestra por primera vez disminución en su capacidad de detección bajando hasta 67.96 %, lo que sugiere que el parche adversario es un ataque bastante poderoso ya que los parches insertados en las imágenes llegan, en muchos casos, a ser clasificados como objetos sobresalientes. Por último, el parche adversario de menor tamaño, aunque logra disminuir el puntaje de la mayoría de todos los algoritmos, no logra comparase con el puntaje del parche de mayor tamaño, algo que era de esperarse pero que puede no ser el caso en los siguientes experimentos.

En el caso del ruido, todos los algoritmos de detección, a excepción de PiCANet, se ven prácticamente inafectados. PiCANet pierde capacidad de detección en los tres tipos de ruido, aunque la pérdida de puntaje no es tan drástica, lo que sugiere que sigue siendo resistente a este tipo de ataques. Es importante notar que una de las posibles razones a la alta robustez a los ataques de ruido se debe a que esta base de datos, como ya se había mencionado, contiene objetos de interés de gran tamaño y alto contraste con el fondo, lo que los hace fáciles de detectar. Esta teoría se comprueba en los resultados de las siguientes bases de datos, donde los ataques con ruido llegan a afectar el puntaje de las redes neuronales, mientras que el GBVS-BP logra mantener su alta robustez.

6.3. ImgSal con FGSM

Tabla 5. Resultados del ataque adversario FGSM usando la base de datos ImgSal. El valor ϵ controla la intensidad de la perturbación en la imagen.

ImgSal								
Algoritmo	Medida	Original	$\epsilon = 2$	$\epsilon = 4$	$\epsilon = 8$	$\epsilon = 16$	$\epsilon = 32$	$\epsilon = 64$
GBVS-BP	Avg.	62.66	62.56	62.52	61.96	60.30	58.87	54.20
	σ	23.96	23.83	24.09	24.11	24.43	24.72	24.73
BASNet	Avg.	63.75	59.84	55.96	53.78	54.76	44.34	**20.69**
	σ	31.73	33.32	34.43	35.62	34.19	35.76	30.10
PiCANet	Avg.	71.60	63.73	56.14	50.71	49.52	44.85	**25.78**
	σ	22.16	28.51	29.29	28.36	28.58	28.75	22.96
DHSNet	Avg.	53.63	53.25	52.51	50.86	49.22	45.96	**39.89**
	σ	24.85	24.62	24.73	24.97	24.81	24.98	24.34

La tabla 5 muestra los resultados al aplicar un ataque adversario FGSM a las tres redes neuronales y al algoritmo de la programación cerebral usando la base de datos ImgSal. Es importante recalcar que la base de datos ImgSal no posee ground-truths correctamente segmentados, por lo que los resultados en la tabla se deben tomar como

un aproximado al verdadero puntaje que los algoritmos hubieran arrojado si la segmentación hubiera sido correcta. En este caso las tres redes neuronales son severamente afectadas por el ataque FGSM, incluso perdiendo gran capacidad de detección a partir del valor $\epsilon = 8$. Para el valor $\epsilon = 64$, su rendimiento ya ha bajado considerablemente, especialmente en las redes PiCANet y BASNet, que bajan hasta los 25 %. El GBVS-BP se mantiene en un puntaje aceptable aún tras la intensa perturbación en este valor de épsilon, bajando solo ocho puntos porcentuales. Esto recalca de nuevo la robustez de la programación cerebral contra este tipo de ataque adversario.

6.4. ImgSal con parche adversario, multipixel y ruido

Tabla 6. Resultados de los ataques adversarios Multipixel, Parche Adversario y tres tipos de ruido usando la base de datos ImgSal. La columna Parche(S) representa los resultados al aplicar un parche más pequeño a la imagen.

ImgSal								
Algoritmo	Medida	Orig.	Multipixel	Parche	Parche(S)	Gauss	S&P	Speckle
GBVS-BP	Avg.	62.66	60.37	46.25	**46.12**	62.27	62.18	61.28
	σ	23.96	23.54	25.13	24.61	23.91	24.21	23.73
BASNet	Avg.	63.75	54.19	27.92	31.49	60.01	57.27	55.06
	σ	31.73	35.81	26.28	27.64	34.68	34.57	34.18
PiCANet	Avg.	71.6	50.33	35.6	38.49	65.12	67.23	64.29
	σ	22.16	30.62	23.57	22.48	29.82	30.56	29.41
DHSNet	Avg.	53.63	49.12	41.94	41.60	51.19	48.83	49.04
	σ	24.85	25.66	26.80	25.84	24.83	25.37	25.77

La tabla 6 muestra los resultados al aplicar los ataques del Parche Adversario, Multipixel y ruido a las tres redes neuronales y al algoritmo de la programación cerebral usando la base de datos ImgSal. En este caso, a diferencia de las base de datos FT, el ataque Multipixel logra afectar de manera significativa el rendimiento de las tres redes neuronales. Esto se puede deber a que los objetos en las imágenes de la base de datos ImgSal no son tan grandes como en FT, y por lo tanto la detección implica un mayor desafío. El algoritmo de la programación cerebral GBVS-BP no se muestra afectado por este ataque. En el caso de parche adversario, este vuelve a afectar gravemente la capacidad de detección de cada uno de los algoritmos, siendo BASNet y PiCANet lo que pierden más del 50 % de su rendimiento. Además, el parche pequeño causa un comportamiento curioso en el algoritmo GBVS-BP y el DHSNet; el puntaje obtenido es menor que en el parche de tamaño normal. Esto indica que un parche más grande no necesariamente afectará más el rendimiento de un algoritmo.

En el caso del ruido, el algoritmo GBVS-BP no se ve afectado en ninguno de los tipos, conservando su puntaje muy cercano al original. En cambio, las tres redes neuronales empiezan a sufrir un decremento en sus rendimientos, especialmente en los ruidos Sal y Pimienta y Speckle. A diferencia del uso de la base de datos FT, donde ningún algoritmo fue afectado por el ruido, el uso de ImgSal sí llega a afectarlos debido a la naturaleza de sus imágenes, las cuales establecen un desafío aún mayor al momento de la detección.

6.5. PASCAL con FGSM

Tabla 7. Resultados del ataque adversario FGSM usando la base de datos PASCAL. El valor ϵ controla la intensidad de la perturbación en la imagen.

PASCAL								
Algoritmo	Medida	Original	$\epsilon = 2$	$\epsilon = 4$	$\epsilon = 8$	$\epsilon = 16$	$\epsilon = 32$	$\epsilon = 64$
GBVS-BP	Avg.	62.94	62.79	62.34	62.53	62.38	61.53	60.53
	σ	20.89	20.65	20.80	21.13	21.60	21.28	21.54
BASNet	Avg.	80.20	77.24	74.64	71.13	65.96	50.82	**14.96**
	σ	21.15	23.16	23.53	26.03	27.74	33.78	29.49
PiCANet	Avg.	75.81	68.16	59.38	51.12	52.00	53.61	**50.08**
	σ	19.80	20.74	20.17	21.48	21.54	20.64	20.33
DHSNet	Avg.	68.14	66.57	65.41	63.44	61.65	60.03	**57.07**
	σ	21.03	21.36	21.07	20.96	21.17	21.37	21.38

La tabla 7 muestra los resultados al aplicar un ataque adversario FGSM a las tres redes neuronales y al algoritmo de la programación cerebral usando la base de datos PASCAL. A primera impresión se observa que el algoritmo GBVS-BP no es afectado por este ataque, manteniendo su puntaje muy cercano al original y manteniéndose siempre arriba del 60 %. Esto demuestra una vez más la gran robustez de la programación cerebral. Por otro lado, ya se había demostrado en las bases de datos anteriores que las redes neuronales no soportan este tipo de ataque, y este caso no es la excepción. Aún cuando BASNet posee el mayor puntaje en las imágenes originales con 80.20 %, su puntaje baja hasta 14.96 % en el épsilon más alto. El algoritmo GBVS-BP puede no haber tenido el puntaje más alto al usar las imágenes originales, pero al aplicar el ataque adversario con la perturbación más severa, este termina superando a las tres redes neuronales.

Tabla 8. Resultados de los ataques adversarios Multipixel, Parche Adversario y tres tipos de ruido usando la base de datos PASCAL. La columna Parche(S) representa los resultados al aplicar un parche más pequeño a la imagen.

PASCAL								
Algoritmo	Medida	Orig.	Multipixel	Parche	Parche(S)	Gauss	S&P	Speckle
GBVS-BP	Avg.	62.94	61.74	58.05	**57.50**	62.55	61.51	61.55
	σ	20.89	18.28	18.84	17.96	18.12	18.41	18.58
BASNet	Avg.	80.20	57.26	53.12	54.67	67.66	74.66	66.49
	σ	21.15	32.54	26.26	29.63	28.86	25.69	29.32
PiCANet	Avg.	75.81	63.31	67.94	69.34	70.16	72.06	68.65
	σ	19.80	19.69	17.24	17.53	17.29	17.25	19.19
DHSNet	Avg.	68.14	64.54	62.03	64.41	65.69	66.32	65.19
	σ	21.03	20.92	21.67	20.83	20.46	20.97	21.42

6.6. PASCAL con parche adversario, multipixel y ruido

La tabla 8 muestra los resultados al aplicar los ataques del Parche Adversario, Multipixel y ruido a las tres redes neuronales y al algoritmo de la programación cerebral usando la base de datos PASCAL. En el caso del ataque Multipixel, las redes neuronales son afectadas de manera considerable, especialmente BASNet, que baja de 80.20 % hasta 57.26 %. Para el parche adversario, el puntaje de las redes neuronales baja todavía más, a excepción de PiCANet, el cual misteriosamente recobra un 4.5 % a comparación con su puntaje en Multipixel. Esto indica que el rendimiento del algoritmo depende sobremanera de la base de datos que se utilice, y en el parche adversario, el algoritmo PiCANet toleró el parche un poco mejor. Sin embargo, la robustez es más evidente en el algoritmo de la programación cerebral GBVS-BP, el cual logra soportar estos dos tipos de ataques de manera aceptable, iniciando en 62.95 % en las imágenes originales, hasta 57.5 % en el parche adversario pequeño. De nuevo, atacar el GBVS-BP con un parche de tamaño normal no significó una menor puntuación a diferencia de haberlo atacado con el parche pequeño.

En el caso de los ataques con ruido, el algoritmo GBVS-BP se ve prácticamente inafectado, pero no se puede decir lo mismo de las redes neuronales. Aun cuando parece que el ataque con ruido no logra afectar tan drásticamente el rendimiento de las redes neuronales, no cabe duda de que hay decremento en la capacidad de detección. BASNet y PiCANet bajan un promedio de 10 a 15 puntos porcentuales, mientras que DHSNet, que ya se había comprobado ser robusto a los ataques con ruido, esta vez baja solamente un 3 %.

6.7. SNPL con FGSM

Tabla 9. Resultados del ataque adversario FGSM usando la base de datos SNPL. El valor ϵ controla la intensidad de la perturbación en la imagen.

SNPL								
Algoritmo	Medida	Original	$\epsilon = 2$	$\epsilon = 4$	$\epsilon = 8$	$\epsilon = 16$	$\epsilon = 32$	$\epsilon = 64$
GBVS-BP	Avg.	52.60	53.31	52.13	50.31	49.09	45.01	42.24
	σ	19.72	19.23	19.64	20.73	21.21	20.27	20.28
BASNet	Avg.	60.27	57.36	53.88	50.66	35.28	13.63	**0.99**
	σ	35.12	35.40	34.43	31.85	31.80	24.72	2.05
PiCANet	Avg.	86.04	72.91	63.7	57.64	44.85	9.26	**8.90**
	σ	14.64	23.56	25.12	25.32	22.41	8.16	5.65
DHSNet	Avg.	43.98	43.23	43.04	42.12	38.63	31.67	**17.84**
	σ	19.88	19.46	19.37	19.88	19.57	19.48	15.77

La tabla 9 muestra los resultados al aplicar un ataque adversario FGSM a las tres redes neuronales y al algoritmo de la programación cerebral usando la base de datos SNPL. La primera observación en este experimento es la disminución en los puntajes de la mayoría de los algoritmos en comparación con las bases de datos anteriores. Con la excepción de PiCANet en la prueba con las imágenes originales, las redes neuronales y la programación cerebral pierden la mayoría de su rendimiento al intentar lograr la detección con la base de datos SNPL. Esto se debe a las diferentes características de acercamiento y obstrucción en este conjunto de imágenes. Aún cuando las redes neuronales lograban puntajes bastante altos con las otras bases de datos, donde los objetos de interés poseen una sobresaliencia bastante alta, un desafío mayor empieza a quebrarlas. Esto se hace mucho más evidente al momento de atacarlas con ejemplos adversarios. Para el caso de FGSM, todas las redes neuronales muestran una disminución gradual comenzando en $\epsilon = 2$ y terminando con un puntaje prácticamente inexistente al atacar con un valor de $\epsilon = 64$. En el caso de la programación cerebral, al probar el algoritmo GBVS-BP con las imágenes originales, éste inicia con un puntaje de 52.6 %, por debajo de la red neuronal BASNet y PiCANet. Sin embargo, al atacar con $\epsilon = 16$, ya logró superar a todas las redes neuronales, acabando con un puntaje muy por encima de éstas. Al final, al momento de atacar con $\epsilon = 64$ la red BASNet muestra un decremento total del 98.3 %, PiCANet un 89.6 % y DHSNet un 59.5 %, mientras que GBVS-BP solo baja un 19.6 %.

Tabla 10. Resultados de los ataques adversarios Multipixel, Parche Adversario y tres tipos de ruido usando la base de datos SNPL. La columna Parche(S) representa los resultados al aplicar un parche más pequeño a la imagen.

SNPL								
Algoritmo	Medida	Orig.	Multipixel	Parche	Parche(S)	Gauss	S&P	Speckle
GBVS-BP	Avg.	52.60	47.70	41.26	**39.32**	48.40	45.74	44.98
	σ	19.72	21.12	21.65	19.89	20.98	21.91	21.26
BASNet	Avg.	60.27	18.09	20.47	20.53	31.27	30.63	10.01
	σ	35.12	28.11	21.30	26.71	34.50	34.73	19.84
PiCANet	Avg.	86.04	13.29	72.61	82.66	32.79	26.93	7.37
	σ	14.64	12.09	25.58	21.02	22.22	23.42	6.63
DHSNet	Avg.	43.98	36.13	34.76	30.01	38.92	38.19	33.60
	σ	19.88	21.02	17.76	15.59	19.42	20.86	20.01

6.8. SNPL con parche adversario, multipixel y ruido

La tabla 10 muestra los resultados al aplicar los ataques del Parche Adversario, Multipixel y ruido a las tres redes neuronales y al algoritmo de la programación cerebral usando la base de datos SNPL. En el caso del ataque Multipixel, se puede notar la inmensa disminución de puntaje en las redes neuronales, a tal grado que la red PiCANet baja de 86.04 % a 13.29 % solamente con este ataque, y la red BASNet sufre las mismas consecuencias. De nuevo, debido a la naturaleza de las imágenes de la base de datos SNPL, estas redes no logran mantener la robustez necesaria para seguir logrando la detección. El algoritmo GBVS-BP solo muestra un 9.3 % de disminución en su puntaje, demostrando su alta robustez. Para el caso del parche adversario, PiCANet logra ser sorpresivamente robusto bajando de 86.04 % a 72.06 % en el parche de tamaño normal, y a 82.66 % en el parche de menor tamaño, logrando un alto nivel de detección aún cuando el parche está diseñado para ser otro tipo de objeto sobresaliente en la escena. Por otro lado, la programación cerebral logra mantenerse cerca del 40 % en los ataques con el parche de ambos tamaños, superando a las redes BASNet y DHSNet.

Los ataques con ruido siguen demostrando que las redes neuronales no son inmunes a estos. En todos los tipos de ataques con ruido, el algoritmo de la programación cerebral GBVS-BP supera a las tres redes neuronales, siendo solamente el ruido Speckle quien logra disminuir su puntaje hasta 44.98 %, pero aún manteniéndose por arriba de las redes. BASNet y PiCANet sufren grandes disminuciones en su rendimiento, siendo el ruido Speckle el que destruye su puntaje casi en su totalidad (BASNet baja un

total de 80 % y PiCANet un 91 %). DHSNet logra demostrar cierta robustez, pero no lo suficiente para superar a la programación cerebral.

6.9. Desviaciónes estándar

Tabla 11. Desviación estándar de cada modelo de detección con respecto a los experimentos con ataques adversarios para cada base de datos.

Desviación Estándar σ				
Algoritmo	FT	ImgSal	PASCAL	SNPL
GBVS-BP	1.93	5.96	1.73	4.47
BASNet	11.09	13.74	17.14	19.39
PiCANet	12.04	13.82	8.90	29.76
DHSNet	4.37	4.56	3.02	7.15

La tabla 11 muestra las desviaciones estándar de cada uno de los modelos de detección para cada base de datos. Aquí se demuestra claramente que la dispersión de los resultados obtenidos por el algoritmo GBVS-BP son bajos a comparación de las redes neuronales. Esto se debe a que la alta robustez de la programación cerebral mantiene los resultados de los experimentos con ataques adversarios en un rango cercano al de las imágenes originales. En el caso de las redes neuronales los resultados se desvían bastante, especialmente en el caso de PiCANet para la base de datos SNPL. Aun cuando logró un alto puntaje en las imágenes originales, ésta se derrumbó en los ataques. DHSNet también demuestra una baja dispersión en los datos, sin embargo, como se vio en las tablas anteriores, es bastante susceptible a ataques como FGSM, por lo que tampoco puede competir con el GBVS-BP.

Capítulo 7. Conclusión

En este trabajo se estudió el problema de detección con un énfasis en la medición de la robustez de diversos algoritmos con respecto a ataques adversarios. Se definió la metodología de la programación cerebral, la cual logra la detección de objetos sobresalientes al involucrar un proceso semejante al de la corteza visual artificial en el ser humano. Tras los experimentos, se probó que el algoritmo de la programación cerebral es altamente robusto, soportando incluso las imágenes con la perturbación más alta. Diversos tipos de ejemplos adversarios fueron implementados, tales como el FGSM, Parche Adversario, Ataque Multipixel, y tres tipos de ruido como Gaussiano, Sal y Pimienta y Speckle. Al atacar a las redes neuronales con cada uno de estos ejemplos adversarios, se confirmó que éstas no son nada confiables. Incluso la más mínima perturbación en sus imágenes de entrada baja su rendimiento en gran medida, sin mencionar que al alimentarlas con una perturbación intensa, éstas quedan prácticamente inservibles. La ventaja de la programación cerebral es que es altamente robusta y confiable al momento de hacer detección, y aunque una red neuronal alcanza rendimientos más altos, se puede derrumbar con prácticamente cualquier perturbación en sus datos de entrada. Estos experimento se llevaron a cabo en tres bases de datos: FT, ImgSal, PASCAL y SNPL. Las tres primeras contienen objetos altamente detectables y no naturales ya que estos abarcan una gran parte de la escena y tienen alto contraste. Pero la base de datos SNPL es diferente, ya que contiene objetos con características que más se asemejan al mundo real. Al usar esta base de datos, logramos acércanos más a un problema de detección con un propósito práctico y que ayuda a resolver de manera directa una problemática del mundo real, en este caso la conservación de naturaleza.

Literatura citada

Achanta, R., Hemami, S., Estrada, F., y Susstrunk, S. (2009). Frequency-tuned salient region detection. En: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. pp. 1597–1604.

Athalye, A., Engstrom, L., Ilyas, A., y Kwok, K. (2018). Synthesizing robust adversarial examples. En: *Proceedings of the 35th International Conference on Machine Learning*, 10–15 Jul. PMLR, Vol. 80 de *Proceedings of Machine Learning Research*, pp. 284–293.

Barbu, T. (2013). Variational Image Denoising Approach with Diffusion Porous Media Flow. *Abstract and Applied Analysis*, **2013**(SI17): 1–8.

Barton, R. (1998). Visual specialization and brain evolution in primates. *Proc Biol Sci*, **265**(1409): 1933–1937.

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

Chinchor, N. (1992). Muc-4 evaluation metrics. En: *Fourth Message Understanding Conference*. pp. 22–29.

David, S., Hayden, B., y Gallant, J. (2006). Spectral receptive field properties explain shape selectivity in area v4. *Journal of Neurophysiology*, **96**(6): 3492–3505.

Desimone, R., Albright, T., Gross, C., y Bruce, C. (1984). Stimulus selective properties of inferior temporal neurons in the macaque. *Journal of Neuroscience*, **4**(8): 2051–2062.

Doull, K., Chalmers, C., Fergus, P., Longmore, S., Piel, A., y Wich, S. (2021). An evaluation of the factors affecting 'poacher' detection with drones and the efficacy of machine-learning for detection. *Sensors*, **21**(12).

Dozal, L., Olague, G., Clemente, E., y Hernandez, D. (2014). Brain programming for the evolution of an artificial dorsal stream. *Cognitive Computation*, **6**(3): 528–557.

Fan, L., Zhang, F., Fan, H., y Zhang, C. (2019). Brief review of image denoising techniques. *Visual Computing for Industry, Biomedicine and Art*, **2**(7).

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

Gross, C., Rocha-Miranda, C., y Bender, D. (1972). Visual properties of neurons in inferotemporal cortex of the macaque. *Journal of Neurophysiology*, **35**(1): 96–111.

Harel, J., Koch, C., y Perona, P. (2006). Graph-based visual saliency. En: *Proceedings of the 19th International Conference on Neural Information Processing Systems*. p. 545–552.

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

Hubel, D. y Wiesel, T. (1953). Receptive fields of single neurones in the cat's striate cortex. *J. Physiol*, **148**(3): 574–591.

Hung, C., Kreiman, G., Poggio, T., y DiCarlo, J. (2005). Fast readout of object identity from macaque inferior temporal cortex. *Science*, **310**(5749): 863–866.

Kahl, S., Wood, C. M., Eibl, M., y Klinck, H. (2021). Birdnet: A deep learning solution for avian diversity monitoring. *Ecological Informatics*, **61**: 101236.

Kobatake, E. y Tanaka, K. (1994). Neuronal selectivities to complex object features in the ventral visual pathway of the macaque cerebral cortex. *Journal of Neurophysiology*, **71**(3): 856–867.

Koch, C. y S., U. (1985). Shifts in selective visual attention: towards the underlying neural circuitry. *Humam Neurobiology*, **4**: 219–227.

Koza, J. (1992). *Genetic programming: on the programming of computers by means of natural selection.*. Cambridge: A Bradford Book.

Krizhevsky, A., Sutskever, I., y Hinton, G. E. (2012a). Imagenet classification with deep convolutional neural networks. En: F. Pereira, C. J. C. Burges, L. Bottou, y K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*. Curran Associates, Inc., Vol. 25.

Krizhevsky, A., Sutskever, I., y Hinton, G. E. (2012b). Imagenet classification with deep convolutional neural networks. En: *Advances in Neural Information Processing Systems*. Vol. 25, pp. 1097–1105.

Kurakin, A., Goodfellow, I. J., y Bengio, S. (2017). Adversarial examples in the physical world. En: *5th International Conference on Learning Representations, ICLR*.

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., Levine, M. D., An, X., Xu, X., y He, H. (2013). Visual saliency based on scale-space analysis in the frequency domain. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **35**(4): 996–1010.

Li, Y., Hou, X., Koch, C., Rehg, J. M., y Yuille, A. L. (2014). The secrets of salient object segmentation. En: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. pp. 280–287.

Liu, N. y Han, J. (2016). Dhsnet: Deep hierarchical saliency network for salient object detection. En: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 678–686.

Liu, N., Han, J., y Yang, M.-H. (2018). Picanet: Learning pixel-wise contextual attention for saliency detection. En: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 3089–3098.

Meyer, F. (2019). *SAR Handbook: Comprehensive Methodologies for Forest Monitoring and Biomass Estimation*, capítulo Chapter 2. Spaceborne Synthetic Aperture Radar – Principles, Data Access, and Basic Processing Techniques. NASA.

Milner, A. y Goodale, M. (2006). *The Visual Brain in Action (2da ed.)*. Oxford University Press.

Norouzzadeh, M., Nguyen, A., Kosmala, M., Swanson, A., Palmer, M., Packer, C., y Clune, J. (2018). Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning. *Proceedings of the National Academy of Sciences of the United States of America*, **115**(25): 5716–5725.

Olague, G., Clemente, E., Dozal, L., y Hernandez, D. (2014). Evolving an artificial visual cortex for object recognition with brain programming. *Schuetze O. et al. (eds) EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation III. Studies in Computational Intelligence, volume 500. Springer, Heidelberg.*, pp. 97–119.

Oram, M. y Perrett, D. (1994). Modeling visual recognition from neurobiological constraints. *Neural Networks*, **7**(6): 945–972.

Pasupathy, A. y Connor, C. (1999). Responses to contour features in macaque area v4. *Journal of Neurophysiology*, **82**(5): 2490–2502.

Qin, X., Zhang, Z., Huang, C., Gao, C., Dehghan, M., y Jagersand, M. (2019). Basnet: Boundary-aware salient object detection. En: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 7471–7481.

Rensink, R. (2000). Seeing, sensing and scrutinizing. *Vision Research*, **40**(10–12): 1469–1487.

Rumelhart, D., Hinton, G., y Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, **323**: 533 – 536.

Sasaki, Y. (2007). The truth of the f-measure. Reporte técnico, School of Computer Science, University of Manchester.

Semarnat (2010). Norma Oficial Mexicana NOM-059-SEMARNAT-2010, Protección ambiental-Especies nativas de México de flora y fauna silvestres-Categoría de riesgo y especificaciones para su inclusión, exclusión o cambio-Lista de especies de riesgo. Diario Oficial de la Federación. 30 de diciembre de 2010, Segunda Sección, México.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., y LeCun, Y. (2014). Overfeat: Integrated recognition, localization and detection using convolutional networks. En: *2nd International Conference on Learning Representations, ICLR 2014*.

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

Su, J., Vargas, D. V., y Sakurai, K. (2019). One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, **23**(5): 828–841.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., y Fergus, R. (2014). Intriguing properties of neural networks. En: *2nd International Conference on Learning Representations, ICLR*.

Tanaka, K. (1996). Inferotemporal cortex and object vision. *Annual Review of Neuroscience*, **19**: 109–139.

Treisman, A. y Gelade, G. (1980). A feature-integration theory of attention. *Cognitive Psychology*, **12**(1): 97–136.

USFWS (1993). Threatened status for the Pacific coast population of the Western Snowy Plover. *Federal Register*, **58**: 12864–12874.

Van Rijsbergen, C, J. (1979). Chapter 7: Evaluation. En: *Information Retrieval*. London: Butterworths.

Walther, D. y Koch, C. (2006). Modeling attention to salient proto-objects. *Neural Networks*, **19**(9): 1395–1407.

Wang, W., Lai, Q., Fu, H., Shen, J., Ling, H., y Yang, R. (2021). Salient object detection in the deep learning era: An in-depth survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Zhang, Y., Song, Y., Liang, J., Bai, K., y Yang, Q. (2020). Two sides of the same coin: White-box and black-box attacks for transfer learning. En: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*. pp. 2989–2997.