Reconocimiento de marca y modelo de vehículos desde un enfoque de reconocimiento de conjunto abierto y descubrimiento de clases nuevas asistido por algoritmos evolutivos

Diana Itzel Vázquez Santiago

Director: Dr. Héctor Gabriel Acosta Mesa

Co-Director: Dr. Efrén Mezura Montes

Tesis sometida como requisito
parcial para obtener el grado de
Maestra en Inteligencia Artificial

Instituto de Investigaciones en Inteligencia Artificial

Universidad Veracruzana

Xalapa-Enríquez, Veracruz, México

2023

Resumen

Los Sistemas de Transporte Inteligentes (STI) han atraído mucha atención en los últimos años debido a sus aplicaciones para mejorar la seguridad, eficiencia y gestión de las redes de transporte. Un punto crucial en muchos STI es el reconocimiento de Marca y Modelo de Vehículos (RMMV) ya que, entre muchas posibles aplicaciones, puede ser empleado para la vigilancia de vehículos sospechosos, la clasificación de los tipos de vehículos que circulan por ciertas áreas, etc. Desafortunadamente el RMMV usualmente se enfrenta a la escasez de bases de datos que contenga los ejemplos suficientes de todos los vehículos de las distintas marcas y modelos que existen en circulación, lo que genera que ciertos vehículos queden fuera de su capacidad de reconocimiento.

En el campo de la Inteligencia Artificial, las Redes Neuronales Convolucionales, han logrado resultados sin precedentes en muchas aplicaciones de Visión por Computadora como clasificación o segmentación de imágenes, detección de objetos, etc. Sin embargo, estos potentes modelos computacionales aún se enfrentan a numerosos retos como la disminución de su costo computacional, el aumento de su precisión de clasificación, el diseño de arquitecturas óptimas para tareas específicas, etc. Si bien muchas propuestas se han planteado para intentar contrarrestar estas problemáticas, la mayoría se han diseñado dentro de un planteamiento de *conjunto cerrado* donde se asume que todos los datos de prueba estarán bien representados por el conjunto de entrenamiento. Desafortunadamente, en problemas de reconocimiento o clasificación de la vida real esto no suele ser el caso y datos de *clases desconocidas* pueden aparecer durante las pruebas, lo que debilita drásticamente la robustez de los modelos.

Para este tipo de problema, el reconocimiento de *conjunto abierto* [29] propone un nuevo enfoque en el que se asume que el conocimiento del mundo está incompleto y los algoritmos deben estar preparados para detectar y rechazar objetos de *clases desconocidas*. Si bien este nuevo enfoque ha logrado disminuir el riesgo de clasificar objetos desconocidos como conocidos, no se incluye la detección de las *clases nuevas* ocultas dentro de las instancias

rechazadas, lo que sería beneficioso para aumentar el conocimiento y la capacidad de clasificación del modelo, incluso después de ser entrenado.

En este trabajo se propone una estrategia de reconocimiento de *conjunto abierto* con una extensión para la detección de *clases nuevas* orientada al reconocimiento de marca y modelo de vehículos. Se empleó una técnica de Neuroevolución y la función de pérdida contrastiva para diseñar una Red Neuronal Convolucional específica al dominio que genera una distribución consistente de vectores de características pertenecientes a la misma clase dentro del espacio en términos de similitud coseno, manteniendo este comportamiento en *clases desconocidas* lo cual funciona como principal guía de un modelo probabilístico y un algoritmo de clustering para detectar objetos de *clases nuevas* y descubrir sus clases simultáneamente. Los resultados muestran que la estrategia presentada funciona eficazmente para abordar el problema de reconocimiento de marca y modelo de vehículos como un problema de reconocimiento de *conjunto abierto* y simultáneamente reconocer las *clases nuevas* ocultas dentro de los objetos rechazados, lo cual es ideal en algoritmos de clasificación/reconocimiento de objetos de la vida real. La solución presentada en este trabajo logró clasificar las instancias de *clases conocidas* del conjunto de prueba con una precisión del 100%, logró identificar las instancias de *clases desconocidas* dentro del conjunto de prueba y logro descubrir las *clases nuevas* ocultas dentro de las instancias de *clases desconocidas.* Con lo cual obtuvo una puntuación micro-F1 de 1.00.

Agradecimientos

Al Consejo Nacional de Humanidades, Ciencias y Tecnologías (CONAHCyT), institución del Gobierno de México, por el apoyo financiero otorgado a través de la "Beca Nacional" con CVU 1141251 como parte del Programa de Becas para Estudios de Posgrado.

A la Universidad Veracruzana (UV) y al Instituto de Investigación en Inteligencia Artificial (IIIA).

Al Dr. Héctor Gabriel Acosta Mesa y al Dr. Efrén Mezura Montes por su tiempo, apoyo y mentoría en todo el proceso de esta investigación.

A mi familia.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1 Panorama General

El reconocimiento de marca y modelo de vehículos (RMMV) tiene como objetivo ofrecer servicios innovadores para mejorar la eficiencia y seguridad de las redes de transporte. Algunos de estos servicios son el análisis y gestión inteligente del tráfico, cobro electrónico de peaje, notificaciones a vehículos de emergencia, aplicación automática de normas de tránsito, sistemas anticolisión, entre otros.

Originalmente el problema de RMMV se abordó con clasificación manual humana y más adelante con el reconocimiento automático de placas, sin embargo, estas soluciones no lograban la precisión y desempeño esperado para las posibles aplicaciones. Para mejorar los resultados, diversos autores han desarrollado algoritmos que, como su nombre lo indica, son capaces de detectar la empresa que fabricó el vehículo (marca) y el nombre del producto que el fabricante vende a los clientes (modelo) empleando distintos enfoques y técnicas para presentar soluciones factibles a los múltiples retos que conlleva la problemática del RMMV como son diferentes modelos de vehículos con apariencia similar [18,21], variaciones en las imágenes por condiciones climáticas o a la resolución [3,19,20], reconocimiento a través de diferentes puntos clave o regiones [15,17], entre otros. Sin embargo, la mayoría de estas soluciones están diseñadas dentro de un enfoque de *conjunto cerrado*, en el que se asume que todos los datos de consulta estarán bien representados por el conjunto de entrenamiento y, por tanto, carecen de mecanismos para detectar durante las pruebas cuándo una muestra de entrada no pertenece a ninguna de las clases predefinidas. Estas situaciones imprevistas son muy probables de suceder en la vida real y debilitan drásticamente la robustez de los modelos.

Cada día tenemos mayor acceso a datos etiquetados, lo que genera que los algoritmos hambrientos de datos, como lo son los algoritmos de clasificación que emplean aprendizaje supervisado, mejoren su precisión de clasificación al disponer de más información de entrenamiento. Sin embargo, es irreal pensar que podremos entrenar estos algoritmos para que reconozcan cualquier objeto que se les presente. En el caso concreto del RMMV, se estima que actualmente existen más de 3,300 marcas de vehículos en el mundo que han añadido y retirado modelos del mercado, modificando el diseño en cada generación y produciendo diferentes versiones de un mismo vehículo, lo que ha hecho muy complicado disponer de una base de datos que contenga suficientes ejemplos de todos los vehículos existentes en circulación para entrenar correctamente un modelo. Esta limitante es muy común en tareas de reconocimiento/clasificación del mundo real, como el RMMV, que en la mayoría de los casos resulta en vehículos mal clasificados porque los algoritmos no estaban preparados para tratar con objetos de *clases desconocidas* (novedosas).

A lo largo de los años, se han presentado diversas soluciones a esta problemática [23,24,25,26,27,28], en donde se les da a los modelos una mayor flexibilidad o se les da la posibilidad de aumentar eventualmente su potencial de clasificación, sin embargo, en estas propuestas no se aborda el reconocimiento de los objetos de clases novedosas durante las pruebas. En un escenario más realista, *clases nuevas* no vistas en el entrenamiento pueden aparecen en las pruebas, por lo que se requiere que los clasificadores no sólo clasifiquen con precisión los objetos de *clases conocidas*, sino que también traten eficazmente con objetos de clases no vistas en el entrenamiento. Este escenario fue descrito y formalizado como *reconocimiento de conjunto abierto* (RCA) por Scheirer et al. [29], quienes además, propusieron una solución llamada máquina 1-vs-Set donde se mide el riesgo de etiquetar una muestra como conocida si está lejos de los datos conocidos (*espacio abierto*) y su objetivo es minimizar este riesgo (*riesgo de espacio abierto*) rechazando objetos que se encuentren más allá del respaldo razonable de los datos conocidos.

El RCA dio lugar a muchas investigaciones que se centraron en limitar de forma más eficaz el *riesgo de espacio abierto* [30,31,32,33]. Desafortunadamente se ha desarrollado poca investigación en torno a realizar el reconocimiento de conjuntos abiertos y simultáneamente descubrir las *clases nuevas* ocultas dentro de las instancias rechazadas. Algunas de las

soluciones propuestas para esta problemática han empleado aprendizaje incremental [34], aprendizaje por transferencia [35,36] o Clustering [37,38]. Aunque obtuvieron buenos resultados, la mayoría de ellas presentan limitaciones como la determinación del número de *clases nuevas* en un evento posterior o separado del reconocimiento de las instancias novedosas; o el uso de ejemplos de *clases desconocidas* durante las etapas de validación, pre-entrenamiento o reentrenamiento como estrategia para afinar sus representaciones/parámetros, sin embargo en el RCA casi nunca se cuenta con información de *clases desconocidas* durante el entrenamiento.

En el caso concreto del RMMV, se han propuesto pocos trabajos que, aunque no han sido descritos dentro de un marco de RCA, disponen de algún tipo de mecanismo para tratar las *clases desconocidas*. Uno de estos trabajos fue propuesto por Nazemi et al. [2] a partir de un enfoque de detección de anomalías. Su sistema base es capaz de clasificar 50 modelos específicos de vehículos, al que le fue añadido una detección de anomalías basada en un umbral de confianza para identificar los vehículos que no pertenecen a ninguna de las 50 clases. Las "anomalías" se clasifican a su vez con base en sus dimensiones dentro de 2 clases: "Desconocido pesado" y "Desconocido ligero". Aunque se disminuye el riesgo de etiquetar un objeto desconocido como conocido, esta propuesta no descubre las *clases desconocidas* dentro de los objetos rechazados, solo presenta un clasificador de 52 clases que necesita ser entrenado con numerosos ejemplos de todas las clases incluyendo las 2 "*desconocidas*". Otro trabajo fue propuesto por Kezebou et al. [23] quienes emplearon un enfoque de aprendizaje Few-Shots que requiere entre 1 y 20 imágenes para la generación de *clases nuevas*. Si bien esta estrategia también presenta una alternativa interesante, la detección de "*clases nuevas*" necesita el apoyo de un conjunto de datos de "consulta" etiquetados que contenga ejemplos de las clases que durante las pruebas fungirán como "*desconocidas*" para poder determinar su clase y de no existir tales ejemplos se generará una clasificación errónea, por lo que ésta solución solo es efectiva si las *clases desconocidas* cuentan con referencias *conocidas*.

1.2 Definición del problema

Sería ideal que los algoritmos de RMMV, al ser un problema del mundo real, fueran capaces de reconocer todos los modelos de vehículos de las distintas marcas que existen en circulación, sin embargo, la mayoría han sido desarrollados dentro de un marco de *conjunto cerrado* por lo que su capacidad de reconocimiento se ve limitada por las clases que se incluyan en su entrenamiento y debido a la escasez de bases de datos que cubran la totalidad de marcas y modelos de vehículos que existen en circulación, los algoritmos tienen un conocimiento incompleto del mundo y cuando se enfrentan a situaciones de la vida real, los vehículos de clases que no fueron contempladas en el entrenamiento son incorrectamente clasificados dentro de alguna de las *clases conocidas*, lo que debilita drásticamente la robustez de los algoritmos.

Una de las estrategias más eficaces para la clasificación/reconocimiento de objetos del mundo real es el reconocimiento de *conjunto abierto* donde se asume que hay un conocimiento incompleto del mundo y se prepara al algoritmo para rechazar objetos de *clases desconocidas*, sin embargo, la mayoría de los trabajos propuestos no son de dominio específico y no se presta atención al descubrimiento de *clases nuevas* escondidas dentro de los objetos rechazados lo cual sería ideal para que los algoritmos adquieran un conocimiento más completo y especializado incluso después de ser entrenados.

1.3 Propuesta de investigación

En este trabajo se propone abordar el problema de Reconocimiento de Marca y Modelo de Vehículos como un problema de Reconocimiento de *conjunto abierto* con una extensión para el descubrimiento de *clases nuevas*. La solución se divide en cuatro etapas:

Primera: Se empleará un algoritmo de neuroevolución para obtener la estructura de una Red Neuronal Convolucional de dominio específico, incluyendo la sección de extracción de características y la sección de clasificación, que genere vectores de características espacialmente cercanos en términos de distancia coseno si pertenecen a la misma clase y

lejanos si pertenecen a clases distintas, manteniendo este comportamiento en imágenes de *clases desconocidas*. Debido al alto costo computacional del proceso de neuroevolución, únicamente se utilizará una pequeña muestra de imágenes de cada clase.

Segunda: La Red Neuronal Convolucional resultante del proceso de neuroevolución será reentrenada con el conjunto de entrenamiento completo para mejorar la calidad de los vectores de características y la precisión de clasificación. La sección de extracción de características de la Red Neuronal Convolucional será empleada para extraer los vectores de características de las imágenes de prueba que incluyen imágenes de clases con las que se diseñó y entrenó el modelo (*conocidas*) e imágenes de *clases nuevas* (*desconocidas*).

Tercera: Los vectores de características de las imágenes de prueba serán ingresadas a un modelo probabilístico y a un algoritmo de clustering, para separar las imágenes en dos conjuntos, uno que contenga objetos de clases *conocidas* y otro de *desconocidas*. Simultáneamente se realizará el descubrimiento de las *clases nuevas* dentro de las instancias *desconocidas.* Este proceso toma como principal guía la distribución de los vectores de características, generada por la Red Neuronal Convolucional.

Cuarta: Los vectores de características de las imágenes de vehículos que si pertenezcan a clases *conocidas* serán ingresados a la sección de clasificación de la Red Neuronal Convolucional para ser clasificados dentro de alguna de las clases para las cuales si fue entrenada la Red.

1.4 Justificación

La mayoría de los algoritmos o sistemas de RMMV suman sus esfuerzos en mejorar los métodos para aumentar la precisión de clasificación. Sin embargo, existen numerosos retos y áreas de oportunidad en el estado del arte que deben ser analizados para mejorar los resultados y al mismo tiempo maximizar los alcances y reducir los posibles riesgos. Al ser un problema del mundo real, uno de los riesgos que tienen la mayoría de algoritmos de RMMV es de clasificar objetos cuya clase no fue contemplada en el entrenamiento dentro de alguna de las *clases conocidas*, esto sucede porque la mayoría carece de mecanismos que les

permitan detectar durante las consultas si todos los objetos que están recibiendo pertenecen a las clases para las cuales fue entrenado. Hasta donde llega el conocimiento del autor, los escasos trabajos reportados de RMMV que contemplan esta problemática, utilizan ejemplos etiquetados de las clases "*desconocidas*" para entrenar sus modelos hacia el reconocimiento de objetos desconocidos dentro de clases genéricas o predefinidas con pocos ejemplos de soporte. Esto sigue sin ser ideal ya que por un lado el modelo que emplea clases genéricas, aunque disminuye el riesgo de etiquetar *clases desconocidas* como *conocidas*, no será capaz de descubrir las clases dentro de los objetos rechazados por lo que los objetos clasificados dentro de las clases genéricas deberán ser posteriormente clasificados manualmente por personal humano con conocimiento específico y por otra parte el modelo que emplea clases "*desconocidas*" predefinidas con pocos ejemplos de soporte sigue considerando un ambiente de *conjunto cerrado* lo cual sigue suponiendo un riesgo porque solo será capaz de clasificar imágenes dentro de alguna clase conocida o "desconocida" con poco soporte.

El RCA presenta un enfoque más realista para el reconocimiento/clasificación de objetos del mundo real ya que asume un conocimiento incompleto del mundo y requiere que los clasificadores sean capaces de clasificar con precisión objetos de *clases conocidas* y rechazar objetos de *clases desconocidas*. Este planteamiento aumenta la robustez de clasificación de los algoritmos, pero no se preocupa en descubrir las clases de los objetos rechazados lo cual sería ideal para que los clasificadores no solo reaccionen a las consultas, sino que sean capaces de seguir aprendiendo incluso después de ser entrenados. Por otra parte, los trabajos de RCA usualmente son validados con bases de datos de referencia como ImageNet, MNIST, CIFAR-100, etc., que no son de dominio específico, por lo que pueden resultar ineficaces en aplicaciones del mundo real de dominio específico.

En este proyecto se plantea desarrollar un algoritmo de RMMV desde un enfoque de RCA con una extensión para el descubrimiento de *clases nuevas*, por lo cual será capaz de detectar objetos de *clases desconocidas* disminuyendo el riesgo de clasificar objetos desconocidos como conocidos y simultáneamente descubrirá las clases de los objetos rechazados empleando como principal guía la distribución de los vectores de características generado por una RNC que será diseñada para el dominio específico del RMMV.

Este planteamiento presenta un algoritmo robusto y capaz de seguir aprendiendo después de ser entrenado. De igual manera esta metodología resalta la importancia de desarrollar algoritmos de dominio específico y puede ser aplicada para diversos dominios.

1.5 Hipótesis

Aplicar un enfoque de RCA con una extensión para el descubrimiento de clases nuevas a la problemática del RMMV generará un algoritmo más robusto al disminuir el riesgo de clasificar imágenes de clases desconocidas dentro de alguna clase conocida y, también, aumentará su capacidad de reconocimiento y aprendizaje después de ser entrenado al ser capaz de reconocer instancias que no pertenezcan a las *clases conocidas* y simultáneamente descubrir sus clases.

1.6 Objetivos

1.6.1 Objetivo general

Emplear Neuroevolución y un enfoque de Reconocimiento de *conjunto abierto* con una extensión para el descubrimiento de *clases nuevas* para obtener un algoritmo de Reconocimiento de Marca y Modelo de Vehículos que clasifique con precisión vehículos de sus *clases conocidas* y que sea capaz de detectar vehículos que no pertenezcan a sus clases predefinidas y simultáneamente descubrir sus clases.

1.6.2 Objetivos específicos

(1) Aplicar un preprocesamiento a las imágenes de 8 clases de vehículos distintos obtenidas de la base de datos VMMRdb [6].

(2) Emplear un algoritmo de Neuroevolución y un Aprendizaje Contrastivo para diseñar una Red Neuronal Convolucional de dominio específico que genere vectores de características espacialmente cercanos en términos de distancia coseno si los objetos

pertenecen a la misma clase y lejanos si pertenecen a clases distintas, conservando este comportamiento en objetos de *clases desconocidas*.

(3) Implementar un mecanismo capaz de detectar objetos de *clases desconocidas* y simultáneamente descubrir sus clases, tomando como principal guía el mapeo de los vectores de características, que fue descrito en el objetivo (2).

(4) Agregar el mecanismo descrito en el objetivo (3) entre las secciones de extracción de características y clasificación de la RNC.

(5) Ejecutar una serie de experimentos utilizando el conjunto de prueba que incluye imágenes de clases con las que se diseñó y entrenó la Red (*conocidas*) e imágenes de *clases nuevas* (*desconocidas*) para probar que el algoritmo es capaz de detectar objetos de *clases desconocidas* y simultáneamente descubrir sus clases.

(6) Clasificar las imágenes de *clases conocidas* con una precisión superior al 90%.

(7) Analizar estadísticamente los resultados obtenidos.

1.7 Organización del documento

El contenido de este documento se encuentra organizado de la siguiente manera:

- **Capítulo 2.** Se presenta el marco teórico de los principales temas de la tesis, donde se abordan los siguientes contenidos: Redes Neuronales Convolucionales, Preprocesamiento de datos, Neuroevolución, Función de pérdida contrastiva, Reconocimiento de *conjunto abierto* y Clustering.

- **Capítulo 3.** Se presenta el marco referencial donde se citan trabajos previos relacionados al Reconocimiento de Marca y Modelo de Vehículos, el *Reconocimiento de conjunto abierto,* la Neuroevolución, el Clustering y la Función de pérdida contrastiva.

- **Capítulo 4.** Se detalla la propuesta presentada para abordar el problema de Reconocimiento de Marca y Modelo de Vehículos como un problema de

Reconocimiento de *conjunto abierto* con una extensión para el descubrimiento de *clases nuevas*.

- **Capítulo 5.** Se muestran los experimentos realizados con la propuesta planteada, los resultados obtenidos y su respectiva discusión.

- **Capítulo 6.** Se presentan las conclusiones alcanzadas con este trabajo y los trabajos a futuro que podrían ser implementados para mejorar los resultados.

Capítulo 2

Marco Teórico

2.1 Redes Neuronales Convolucionales

Las Redes Neuronales Convolucionales (RNC) son un tipo de Red Neuronal Artificial (RNA), inspiradas de algunos aspectos del patrón de conexiones en la corteza visual de los animales. Este algoritmo de aprendizaje profundo ha logrado grandes resultados, principalmente en el campo de visión por computadora, demostrando ser de gran utilidad en tareas como detección de objetos, clasificación de imágenes, creación de imágenes, procesamiento del lenguaje natural, entre otros.

Las RNC están conformadas por diferentes parámetros, como el número de capas convolucionales, el tamaño de los filtros, el optimizador, la tasa de aprendizaje, etc., que variarán en función de la tarea a realizar. Son entrenadas empleando aprendizaje supervisado y retropropagación lo cual les permite corregirse a sí mismas para lograr una mejor precisión.

La topología de una RNC se divide en múltiples etapas de aprendizaje compuestas por una combinación de capas convolucionales, unidades de procesamiento no lineal y capas de submuestreo (Khan et al. 2020). Si bien no existe una regla general para el diseño de RNCs su funcionamiento general se divide en 2 actividades principales:

- **Extracción de características**: son un conjunto de operaciones de convolución y *pooling* que tienen como objetivo principal extraer las características más significativas de las imágenes.
- **Clasificación:** son un conjunto de capas totalmente conectadas que tienen la función de combinar las características extraídas para brindar la clasificación final de las imágenes.

En la Figura 2.1 se muestra la arquitectura general de una RNC. A continuación, se describen brevemente las partes que la conforman.

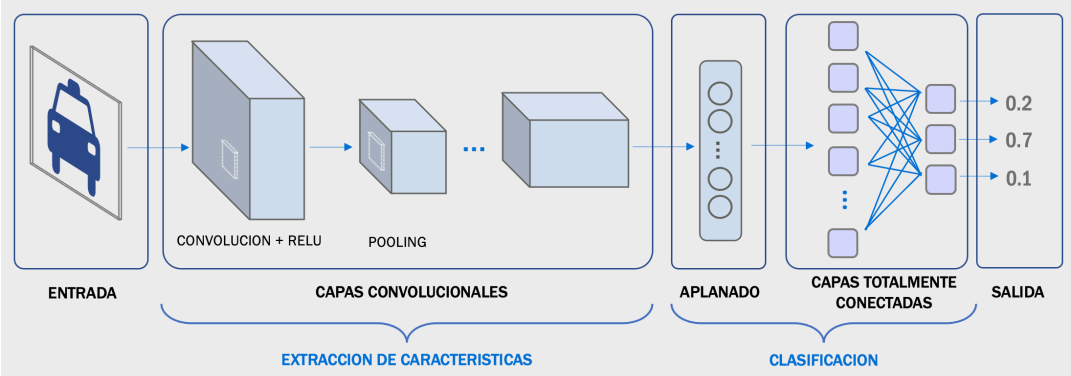

Figura 2.1 Arquitectura general de una Red Neuronal Convolucional.

2.1.1 Imagen de entrada

La entrada de una Red Neuronal Convolucional es una imagen digital, la cual está compuesta por un conjunto de elementos, cada uno con una posición y valor específico. Estos elementos son llamados pixeles los cuales denotan la unidad mínima de medida de una imagen digital. Los valores que tendrá cada píxel serán dados por el espacio de color que se esté utilizando, comúnmente para representar imágenes se emplea el espacio de color RGB (Red Green Blue), el cual descompone el color de las imágenes en un vector de intensidades, cuyos componentes representan la intensidad de rojo, verde y azul de cada píxel. Un ejemplo de esta descomposición se muestra en la Figura 2.2.

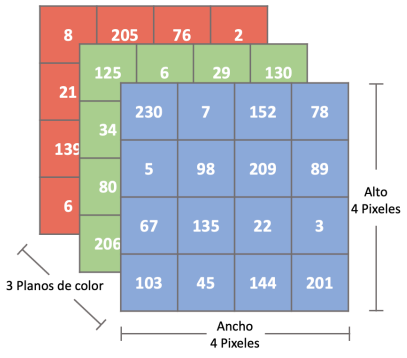

Figura 2.2. Ejemplo de la descomposición de una imagen RGB en sus 3 planos de color (Rojo, Verde y Azul).

Si bien la arquitectura de las RNCs puede estar diseñada para recibir imágenes de entrada en 3 dimensiones: alto, ancho y profundo, lo cual les permite trabajar directamente con los 3 planos o canales de color de las imágenes RGB, esto no siempre es el caso. Dependiendo el

diseño de la RNC que se está empleando se debe ajustar el espacio de color (RGB, CIELAB, Escala de Grises, etc.) y las dimensiones de las imágenes (usualmente en la etapa de preprocesamiento), para que las dimensiones de las imágenes coincidan con las dimensiones de entrada de la primera capa de la RNC. Es importante mencionar que el costo computacional está correlacionado con las dimensiones de las imágenes de entrada, si aumentan las dimensiones, mayor será el costo.

2.1.2 Capas convolucionales

Las capas convolucionales son el componente principal de las RNC al funcionar como extractores de características auto entrenados mediante la operación de convolución que ejecutan. Esta operación matemática traslada un filtro o kernel sobre una imagen, esencialmente fusionando los dos conjuntos de información generando un mapa de características de salida.

Los filtros o kernels de una capa convolucional son una cuadricula de números discretos, llamados pesos, los cuales comúnmente tienen forma cuadrada y dependiendo el espacio de color de las imágenes, pueden tener una profundidad para ajustarse al número de planos. Su función es dividir la imagen en pequeños fragmentos comúnmente conocidos como *campos receptivos*, que ayudan a extraer características específicas que jerárquicamente construyen características más complejas. Pueden ser considerados como neuronas artificiales a las cuales se les asocia un valor de sesgo el cual será determinado, en conjunto con los pesos, durante el proceso de entrenamiento utilizando el algoritmo de retropropagación.

Durante el proceso de convolución, se posiciona el filtro o kernel sobre los pixeles superiores izquierdos de la imagen, asegurando que esté contenido totalmente dentro de las dimensiones de la entrada y se aplica el producto punto entre los valores del filtro y los valores de la región cubierta de la imagen. El resultado de este producto más el sesgo asociado con el filtro, será el nuevo valor del píxel el cuál será agregado al mapa de características. En la Figura 2.3 se puede observar la operación de convolución entre una imagen de 6x6 y un filtro de 3x3. Este proceso se repite desplazando el filtro sobre toda la imagen y cada posición se procesa con valores de peso diferentes.

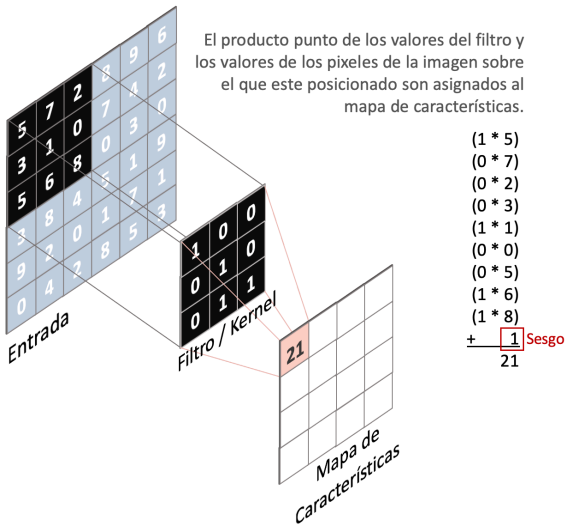

Figura 2.3. Ejemplo de la operación de convolución en una imagen de 2 dimensiones de 6x6 con un kernel de dos dimensiones de 3x3.

En las RNC, el proceso de convolución se realiza sobre imágenes digitales, en el caso de imágenes RGB, se representan como una matriz de 3 dimensiones: alto, ancho y profundidad. Como se mencionó previamente, dependiendo la distribución de la imagen, se deben igualar las dimensiones del filtro para que éste sea capaz de cubrir la profundidad de las entradas. En la Figura 2.4 se muestra un filtro de 3x3x3 que será utilizado para realizar el proceso de convolución sobre una imagen con 3 planos de color (p. ej. RGB), con lo cual se cubre la necesidad de abarcar la profundidad de la entrada.

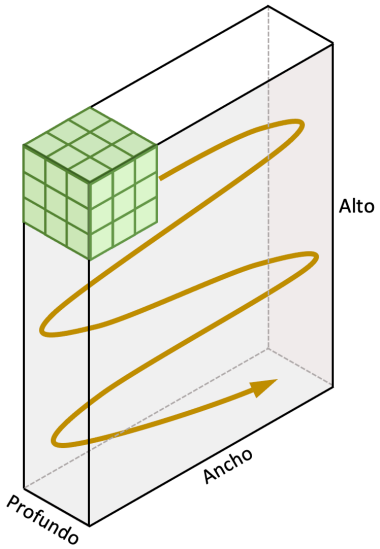

Figura 2.4. Kernel de 3x3x3 transpuesto sobre una imagen con 3 planos de color.

El proceso de convolución en este caso se realiza por cada capa de profundidad, cada dimensión de la imagen que representa un canal de color es sometida al proceso de convolución utilizando la dimensión correspondiente del filtro. En el caso de las imágenes RGB, serán 3 procesos de convolución cuyos resultados y el valor del sesgo asignado al filtro, serán sumados para obtener el valor del píxel de salida que será agregado al mapa de características. Aunque el proceso de convolución se realiza sobre una imagen de 3 dimensiones (bandas), el resultado final (mapa de características) tendrá 1 sola dimensión (banda). Un ejemplo de este proceso se puede observar en la Figura 2.5.

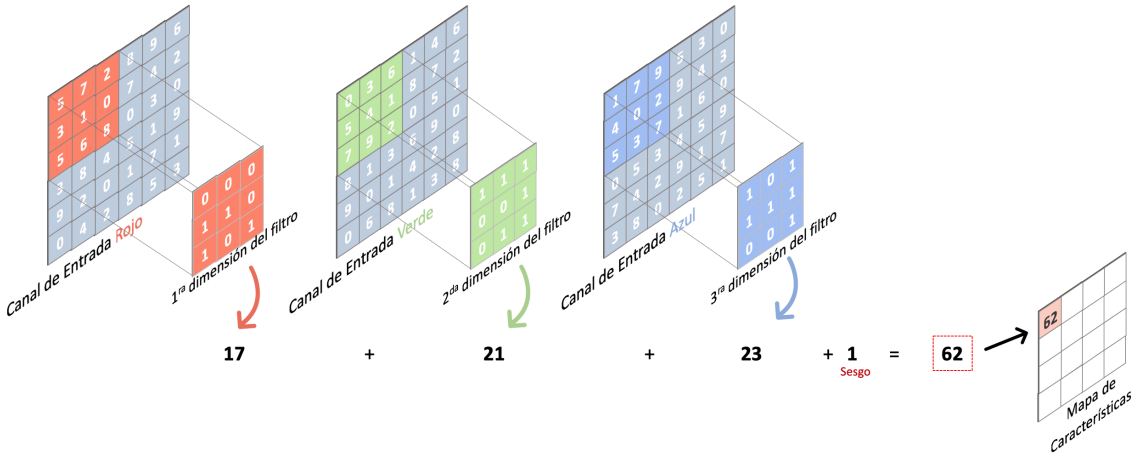

Figura 2.5. Ejemplo del proceso de convolución sobre una imagen RGB (3 canales) con un Kernel de tamaño 3x3x3.

Las RNC están conformadas por *n* capas convolucionales y cada capa convolucional está conformada por un conjunto de filtros (del mismo tamaño) que se aplican a una imagen con el proceso de convolución mencionado previamente y dan como resultado un mapa de características. Al final de este proceso, la nueva imagen (mapa de características) tendrá dimensiones distintas de la imagen original, su profundidad será equivalente al número de filtros y lo alto y ancho dependerán de las dimensiones del filtro y la imagen de entrada que son invariantes durante el proceso, por lo que pueden ser calculados con las siguientes formulas:

$$Alto_{Nueva\ Imagen} = Alto_{Imagen\ Entrada} - Alto_{Filtro} + 1$$
$$Ancho_{Nueva\ Imagen} = Ancho_{Imagen\ Entrada} - Ancho_{Filtro} + 1 \quad (2.1)$$

En la Figura 2.6 se puede apreciar el cambio de dimensionalidad después de que una imagen pasa por una capa convolucional. A la izquierda de la Figura 2.6 se muestra un ejemplo de las

dimensiones resultantes en un proceso de convolución entre una imagen de 32x32x3 y un filtro de 5x5x3 que producen una salida de 28x28x1 y a la derecha de la Figura 2.6 se muestra un ejemplo de las dimensiones resultantes en un proceso de convolución entre una imagen de 32x32x3 y cuatro filtros de 5x5x3 que producen una salida de 28x28x4.

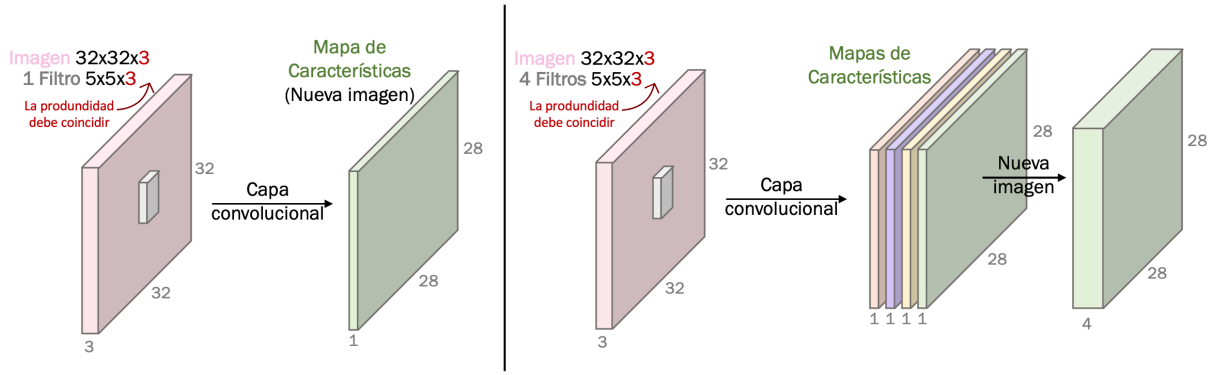

Figura 2.6. Ejemplos de las dimensiones resultantes en procesos de convolución dependiendo el número de filtros.

2.1.3 Stride y Padding

El *stride* es un valor numérico que especifica la cantidad de espacios o pixeles que se desplazará el filtro hacia la derecha y hacia abajo durante el proceso de convolución. A medida que se aumenta el valor del *stride,* se reduce el solapamiento entre los *campos receptivos* al ignorar ciertas regiones de la imagen, lo que genera que el mapa de características sea más pequeño.

Sin importar el número de *stride* el mapa de características resultante de un proceso de convolución siempre será de menor dimensionalidad. Como medida opcional para mantener las dimensiones, los bordes de la imagen pueden rellenarse con valores "dummy", generalmente utilizando el valor cero (*zero-padding*). A este proceso se le conoce como *padding*.

Las dimensiones de alto y ancho del mapa de características después de aplicar *stride* o *padding* pueden ser calculados con la Eq. 2.2, recordando que la profundidad siempre dependerá del número de filtros que tenga la capa convolucional.

$$MC_{medida} = \left\lceil \frac{I - F + 2P}{S} \right\rceil + 1 \qquad (2.2)$$

donde I es el tamaño de la imagen de entrada (cuando las dimensiones de alto y ancho son iguales), F es el tamaño del Filtro, P es el número de bordes de *padding* agregados, S es el valor del stride y los símbolos [·] denotan la función ceil.

En la Figura 2.7 se observan dos ejemplos de operaciones de convolución, ambas con una imagen de entrada de dimensiones 5x5 con 1 borde de *zero-padding* y un kernel de 3x3. En la izquierda se muestra el proceso con stride de 1 dando como resultado un mapa de características de dimensiones 5x5 y en la derecha se muestra el proceso con stride de 2 dando como resultado un mapa de características de dimensiones 3x3.

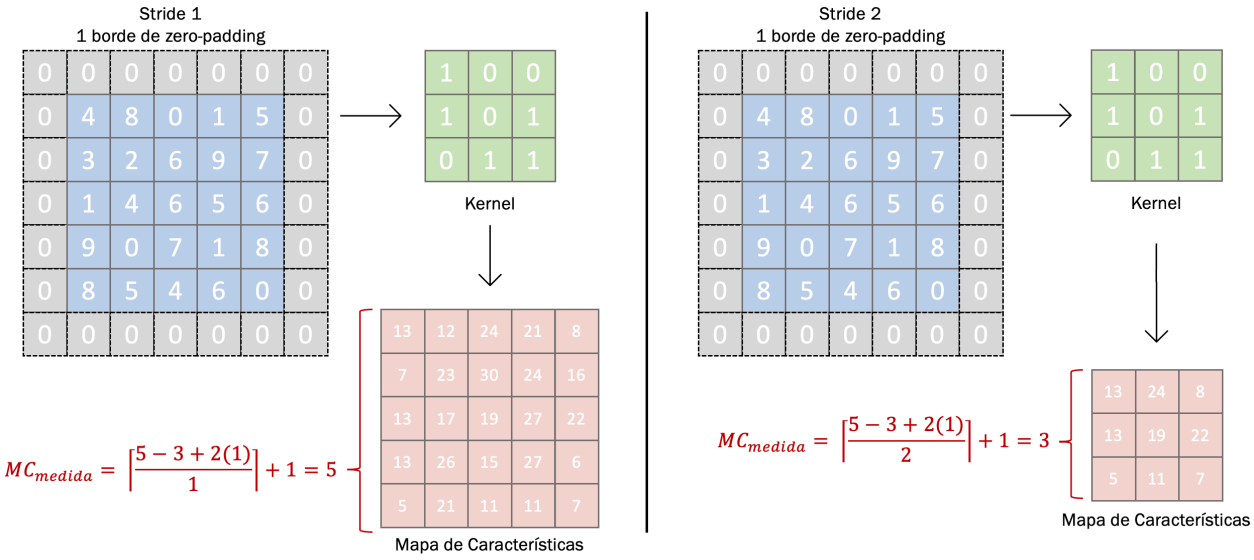

Figura 2.7. Ejemplos de procesos de convolución con stride y padding.

2.1.4 Capas de Activación

El proceso que realizan las capas convolucionales tiende a tener un comportamiento lineal, sin embargo, para que cualquier tipo de red neuronal sea potente, requiere de una función no lineal o umbral que modifique el valor resultante o imponga un límite que se debe sobrepasar para poder continuar a la siguiente capa, esta función es comúnmente denominada como función de activación. En el caso específico de las RNC la unidad lineal rectificadora (ReLU) expresada matemáticamente en la Eq. 2.3 es utilizada regularmente como función de activación. Se aplica sobre los valores de los mapas de características y emite el valor directamente si es un numero positivo mayor a cero y de lo contrario emite

cero, como se muestra en la Figura 2.8. Con este proceso se aumenta el poder de representación de las características extraídas.

$$ReLU(x) = max(0, x) \tag{2.3}$$

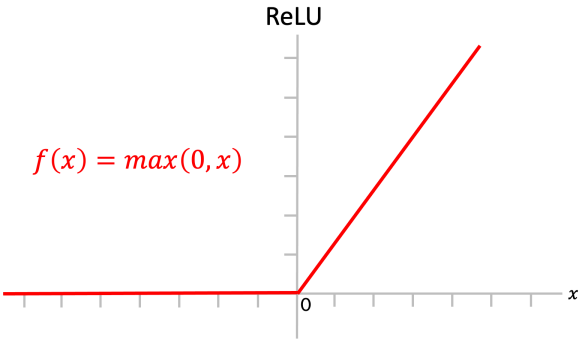

Figura 2.8. Gráfica de la función ReLU (señalado en rojo).

2.1.5 Capas de Pooling

Comúnmente, después de un proceso de convolución, se realiza un proceso de *pooling* para reducir la dimensionalidad a lo alto y ancho, manteniendo la profundidad. Con este proceso se resumen las características presentes en una región del mapa de características, lo que permite disminuir el número de parámetros por aprender en las siguientes capas, reduciendo el tiempo de entrenamiento y el sobreajuste.

Las capas de *pooling* utilizan un filtro o kernel, el cual es trasladado con cierto valor de stride sobre una entrada con *padding* opcional, similar al proceso de convolución. La principal diferencia es que el filtro empleado en este proceso no tiene pesos, su función es extraer ciertas características en función de un criterio determinado. El tipo de *pooling* más utilizado es el *max pooling*, el cuál toma el valor máximo de la región que esté cubriendo el kernel. Un ejemplo de este proceso es ilustrado en la Figura 2.9 donde se aplica el proceso de *max pooling* a una entrada de 4x4 con un filtro de 2x2 y un stride de 2 resultando en una salida con dimensión de 2x2.

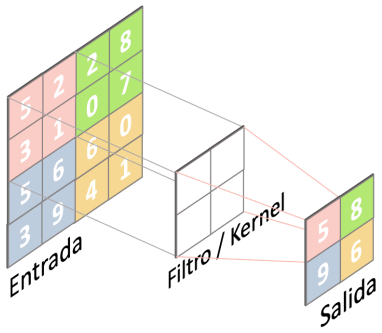

Figura 2.9. Proceso de max pooling aplicado a una imagen de entrada de 4x4 con un kernel de 2x2 y un stride de 2.

2.1.6 Capas de Normalización

Estas capas pueden ser posicionadas en distintos puntos entre las capas de las RNC, normalmente se aplican a la entrada de la Red o después de una capa de convolución o *pooling* como una estrategia para evitar el *Internal Covariate Shift*, el cual es un cambio en la distribución de los valores de activación causado por el ajuste de los pesos durante el entrenamiento de la Red. Para mitigar este problema se agregan las capas de normalización con la finalidad de regularizar los valores, evitando el sobreajuste del modelo.

Una práctica común de muchos algoritmos de aprendizaje profundo es utilizar subconjuntos de los datos en cada época del entrenamiento de la Red con la finalidad de obtener una aproximación más real al comportamiento que tendrá. Las capas de normalización trabajan sobre los pesos de los subconjuntos entrantes aplicando la Eq. 2.4 sobre cada componente.

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{2.4}$$

donde $\hat{x}_i$ es el mapa de características normalizado, x_i representa el mapa de características de la i-ésima capa, μ_B y σ_B^2 son la media y varianza de cada subconjunto y ϵ es una pequeña constante para evitar la división entre cero.

Después de este proceso el mapa de características tiene una media de cero y una varianza de uno. Para no limitar el poder de representación de la capa convolucional anterior, se aplica otra transformación sobre cada valor normalizado aplicando la Eq. 2.5.

$$y_i = \lambda \hat{x}_i + \beta \tag{2.5}$$

donde λ y β son nuevos parámetros por aprender asociados a cada entrada $\hat{x}_i$.

2.1.7 Capas totalmente conectadas

Los procesos mencionados anteriormente se encargan de descomponer la imagen en características y analizarlas de forma independiente, mientras que el ultimo componente esencial de las RNC llamado capas totalmente conectadas, controla la decisión final de clasificación.

Estas capas están conformadas por un conjunto de funciones llamadas neuronas, cada una recibe un vector de entrada al cual se le aplica una transformación lineal que multiplica las entradas ***x*** con los respectivos pesos **w** asociados a sus entradas para posteriormente sumar los resultados de las multiplicaciones y el sesgo asociado **b**. Este proceso se representa matemáticamente en la Eq. 2.6. Posteriormente se aplica una transformación no lineal, llamada función de activación, al resultado obtenido y se determina si la neurona será o no activada. Una representación de este modelo se muestra en la Figura 2.10.

$$f\left(\sum_{i=1}^{m} w_i x_i + b\right) \tag{2.6}$$

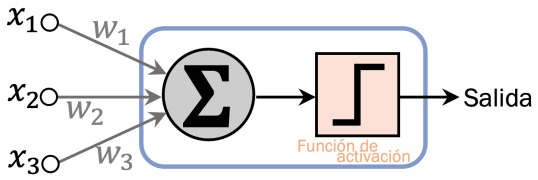

Figura 2.10. Representación del modelo de una neurona artificial con 3 entradas.

En las RNC usualmente se emplean varias capas totalmente conectadas las cuales producen una potente composición de funciones conocidas como *redes neuronales* o *perceptrón multicapa*. Las capas totalmente conectadas toman como entrada el mapa de características de la última capa de *pooling* o convolución realizada en la sección de extracción de características de la RNC, el cual es aplanado mediante un proceso que consiste en transformar una matriz multidimensional en un vector. Este vector se ingresa a la primera capa totalmente conectada y durante el entrenamiento se aplica un proceso de retropropagación para modificar los pesos de las neuronas y reducir el error obtenido en cada época mediante una función de pérdida.

Para que la clasificación sea posible, la última capa totalmente conectada debe tener la misma cantidad de neuronas como el número de clases a predecir y para obtener la distribución de probabilidad sobre el conjunto final del número total de clases se emplea una función de activación, usualmente la Sigmoide, o cuando se necesita una clasificación multiclase, comúnmente se utiliza la función de activación Softmax cuyo comportamiento es muy similar a la función Sigmoide y su fórmula se muestra en la Eq. 2.7.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.7}$$

donde z_i es el vector de entrada y K es el número de clases.

2.2 Preprocesamiento de datos

En cualquier área de investigación siempre será esencial contar con datos relevantes al tema que se esté estudiando, para poder llegar a conclusiones precisas. Cada día aumenta la cantidad de datos a los que tenemos acceso y que podemos encontrar en distintos formatos como texto, video, audio, imagen, etc. Desafortunadamente, la mayoría de los datos disponibles contienen inconsistencias que pueden impactar negativamente el proceso o análisis que se intente hacer con ellos. Para mitigar estos posibles errores existen ciertas estrategias o técnicas que pueden ser empleadas para consolidar los datos y facilitar su uso, esta preparación es conocida como preprocesamiento de datos.

Las técnicas de preprocesamiento a ser aplicadas sobre los datos dependerán del área de investigación y las técnicas que serán empleadas. En el caso específico de clasificación de imágenes con RNC, los procesos más comunes a aplicar sobre las imágenes son modificaciones en las dimensiones de las imágenes, aumento de los ejemplos disponibles, definición de la región de interés (RDI), eliminar duplicados, mejorar la calidad de las imágenes, etc.

Usualmente el primer paso en el preprocesamiento de imágenes es la limpieza de datos para eliminar duplicados de la base de datos y seleccionar las imágenes que serán funcionales al propósito de la investigación, posteriormente se plantean soluciones para mejorar las

imágenes, delimitando la RDI, aplicando filtros para resaltar características significativas o lidiar con variaciones por condiciones climáticas, normalizar los valores de las imágenes, etc.

Un punto importante a considerar en el preprocesamiento de imágenes que serán empleadas en una RNC, son las dimensiones de entrada de la red, ya que para que las imágenes puedan ser ingresadas deberán coincidir en dimensiones con la entrada del modelo, lo cual usualmente se logra escalando las imágenes o cambiando el espacio de color en el que se encuentren. Es posible que una RNC sea diseñada específicamente para trabajar con las dimensiones originales de las imágenes, sin embargo, lo más común es ajustar las dimensiones de las imágenes a las del modelo.

Finalmente, para que los algoritmos de clasificación tengan un buen rendimiento, especialmente los que utilizan redes neuronales, es crucial tener un conjunto de datos extenso y balanceado. Sin embargo, en muchas ocasiones no se cuenta con ello, lo que provoca un pobre rendimiento predictivo, específicamente para las clases que tienen la minoría de ejemplos. Para lidiar con esta problemática se proponen distintas soluciones, como adecuar las métricas de evaluación o aplicar un aumento de datos, lo cual se logra haciendo ligeras alteraciones a las imágenes del conjunto de datos con el que se cuenta. Estas alteraciones generalmente se conocen como transformaciones afines que incluyen procesos como: traslación, escalamiento, rotación, reflexión, etc. En la Figura 2.11 se puede apreciar una imagen que tiene el numero 5 escrito a mano y algunas de las posibles transformaciones afines que se le pueden aplicar con la finalidad de aumentar los datos. Para la red, las imágenes generadas con este proceso serán únicas y aprenderá de todas ellas.

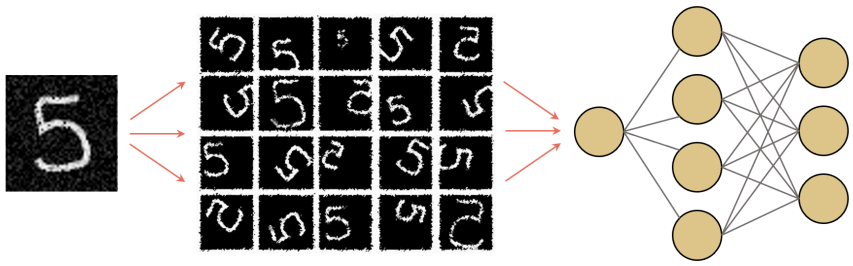

Figura 2.11. Aplicación de transformaciones afines a una imagen para aumentar los datos que serán alimentados a una red neuronal.

2.3 Neuroevolución

La tarea de diseñar la topología de cualquier tipo de RNA, es conocida por ser un proceso que toma mucho tiempo y conocimientos especializados debido a la falta de nociones bien establecidas sobre cómo diseñar estas redes. Como propuesta a este problema se han desarrollado técnicas de optimización para ajustar automáticamente las estructuras de las Redes y sus hiperparámetros. Hasta el momento, la técnica que ha conseguido los mejores resultados para solucionar esta problemática es la Neuroevolución (NE) , la cual surgió en el campo del Cómputo Evolutivo (CE) donde se utilizan algoritmos de este tipo, particularmente el Algoritmo Genético (AG) para buscar las mejores soluciones en problemas de optimización.

La evolución biológica es el cambio gradual de las características de los individuos de una población a lo largo de generaciones con la finalidad de adaptarse a su nicho biológico. Este principio fue tomado como inspiración por el CE para generar un AG como el de Neuroevolución que emula este proceso biológico. El proceso de neuroevolución actual se puede resumir de la siguiente forma: se genera una población aleatoria de redes individuales (con una codificación neuronal), se crean redes reales a partir de ellas, se evalúan las redes con una función que mide la calidad de los resultados (función de aptitud), se seleccionan las redes más aptas, se introducen ciertos cambios aleatorios para crear una descendencia a partir de ellas y se selecciona la nueva población (generación). Este proceso se repite hasta que se alcanza un determinado nivel de aptitud o de generaciones. Un modelo general de este proceso se muestra en la Figura 2.12.

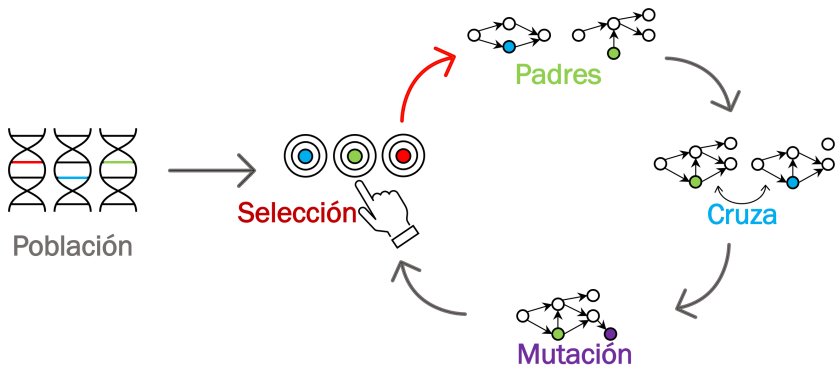

Figura 2.12. Modelo general de neuroevolución. La flecha roja indica que podría ser necesario un proceso de decodificación de genotipo a fenotipo.

Un punto crucial para el funcionamiento de este algoritmo es el diseño de codificaciones neuronales, las cuales contienen toda la información de la topología de una RNA y definen el espacio de búsqueda. Estas codificaciones son el equivalente a los genes de los individuos que contienen toda su información genética la cual es transformada durante el proceso de evolución biológica. Otro punto importante en la NE es la función de aptitud, la cual debe ser elegida cuidadosamente ya que será la encargada de determinar en qué medida los individuos cumplen con el o los criterios que el algoritmo optimiza.

Originalmente la Neuroevolución fue empleada para evolucionar los pesos de una RNA (Montana and Davis 1989) y en la actualidad es empleada mayormente para diseñar y configurar topologías de RNA. En años recientes se ha comenzado a aplicar esta técnica en el desarrollo de diseños y configuraciones de arquitecturas de RNC para tareas demandantes. El éxito en los resultados obtenidos se debe en gran medida al diseño de nuevas codificaciones neuronales como las codificaciones híbridas que permitieron encontrar nuevos tipos de soluciones al generar mejores condiciones en la representación del espacio de búsqueda.

2.3.1 Codificaciones neuronales

Como se mencionó previamente, las codificaciones de redes neuronales artificiales contienen toda la información de la topología de una RNA. Son esenciales en el proceso evolutivo ya que representan computacionalmente a las RNA y tienen un gran impacto en la complejidad del espacio de búsqueda. Las codificaciones neuronales de las RNA (incluyendo las RNC) pueden clasificarse en dos categorías principales:

- **Codificaciones Indirectas:** Son una representación a nivel de genotipo que requiere un conjunto de reglas para transformarse en un fenotipo que pueda evaluarse. Una de sus características principales es que se limita el espacio de búsqueda al dominio del problema al no codificar explícitamente toda la información de las redes. Estas codificaciones ayudan a que el modelo sea más rápido pero muy a menudo conlleva a resultados subóptimos.
- **Codificaciones Directas:** Son una representación a nivel de fenotipo, ya que todas las características de las redes se definen explícitamente en la codificación. Estas

codificaciones tienen pocas limitaciones en el espacio de búsqueda lo que puede generar que aumenten los requerimientos de tiempo y memoria computacional.

En años recientes se ha comenzado a estudiar una codificación neuronal, llamada codificación híbrida, que combina elementos de las codificaciones mencionadas anteriormente para eliminar algunas de sus limitaciones. Estas representaciones híbridas son útiles para distribuir las representaciones de las RNC en diferentes subestructuras generando una mejoría en la búsqueda.

Un ejemplo de esta codificación fue presentada por Vargas-Hákim et al. [10] dentro de un marco evolutivo llamado Deep Genetic Algorithm (DeepGA) el cual se muestra en el Algoritmo 1 (tomado y traducido directamente de [10]) y fue diseñado específicamente para evolucionar RNCs. Un ejemplo de la codificación empleada en DeepGA se muestra en la Figura 2.13 la cual está conformada por dos niveles que serán descritos a continuación y que fue basada en la codificación de Wang (Wang et al. 2019).

- **Primer nivel. Bloques:**
    1. **Bloques Convolucionales:**

       Se definen únicamente por una capa convolucional y una operación de *pooling* (opcional). Abstraen las características de una capa convolucional (número de filtros y tamaño de los filtros). Utiliza *stride* de 1 y *zero-padding* de 1. Pueden utilizar *max pooling*, *average pooling*, o ningún *pooling*. Finalmente, se aplica la función ReLU y normalización inmediatamente después de la convolución.
    2. **Bloques Fully-Connected:**

       Son capas totalmente conectadas que se colocan siempre al final de la sección de extracción de características (Bloques Convolucionales) y se describen por su número de neuronas. Se utiliza la función ReLU como función de activación y siempre se añade un último bloque fully-connected con un número de neuronas igual a la cantidad de clases, la cual utiliza la función de activación *softmax*.

- **Segundo nivel. Conexiones:**

  Es una cadena binaria que define la conectividad entre bloques convolucionales. Cada bit representa la conectividad de una capa anterior no consecutiva, iniciando a partir del tercer bloque.

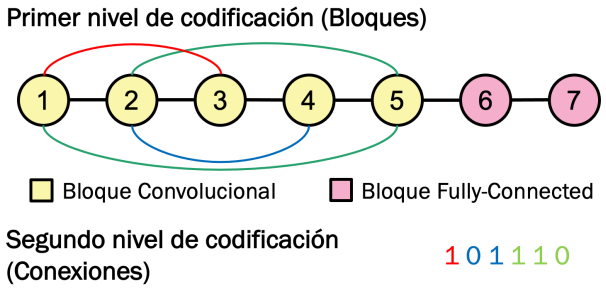

Figura 2.13. Ejemplo de la codificación hibrida implementada en DeepGA.

Algoritmo 1. Pseudocodigo de DeepGA.

Entrada: Una población P de N individuos. Un numero de generaciones T, un porcentaje de cruza $CXPB$, un porcentaje de mutación $MUPB$, un tamaño de torneo $TSIZE$.
Salida: El mejor individuo de P con base en la Aptitud.
Inicializar la población (entrenar las redes).
$t \leftarrow 1$
mientras $t \leq T$
  Seleccionar $N/2$ padres con torneo de selección probabilístico
  Decendencia $\leftarrow \{\}$
  mientras |Decendencia| < N /2
    Seleccionar dos padres $p1$ y $p2$ aleatoriamente.
    si random(0,1) ≤ $CXPB$ **entonces**
      $O1, O2 \leftarrow$ Cruza($p1, p2$) // *Cruza*
      si random(0,1) ≤ $MUPB$ **entonces**
        Mutación($O1, O2$) // *Mutación*
      Aptitud($O1, O2$) // *Evaluación*
    $P \leftarrow P \cup$ Decendencia
    Seleccionar los mejores N individuos de P como sobrevivientes.
  fin
  Seleccionar el mejor individuo de P con base a la Aptitud.
fin

2.3.2 Operadores evolutivos

Los AG varían en su estructura en función de su finalidad, pero todos comparten algunos componentes llamados operadores evolutivos que construyen una nueva generación de soluciones a partir de las anteriores de forma que se preserve el material genético de las

mejores soluciones. De esta forma se espera que las nuevas soluciones sean mejores con el paso de las generaciones. Estos operadores son los encargados de realizar la búsqueda de mejores soluciones; los más comunes son la selección de padres, cruza, mutación y selección de sobrevivientes.

La operación de selección de padres consiste en seleccionar individuos de la población para reproducirse y transmitir sus genes. El número de individuos que serán elegidos dependerá del AG y los operadores de cruza y mutación que serán empleados. Los individuos de la población tienen cierta probabilidad de ser seleccionados dependiendo de su calidad (valor de aptitud) de forma que los individuos que mejor cumplen con el o los criterios que el algoritmo este optimizando tendrán mayor probabilidad de ser seleccionados para replicar sus genes.

La operación de cruza toma dos de los individuos seleccionados como *"padres"* y combina sus genes para crear nuevos individuos (*"descendencia"*), la cantidad de individuos a ser creados generalmente es igual al número de padres o al tamaño de población. Es importante resaltar que la descendencia derivada de la cruza solo posee información heredada de los padres y no se añade información nueva, lo que garantiza la explotación del espacio de búsqueda que está cerca de los padres, con la esperanza de encontrar soluciones con mayor aptitud.

La operación de mutación, por otro lado, se aplica sobre los individuos generados en la cruza, cambiando o agregando aleatoriamente algunos genes de los individuos, con el objetivo de explorar el espacio de búsqueda tomando prestado el concepto evolutivo de que las mutaciones genéticas aleatorias pueden producir una descendencia superior. Puede verse entonces que la mutación, a diferencia de la cruza, sí añade elementos nuevos a los individuos.

Después de que se realiza la cruza y la mutación, se lleva a cabo la selección de sobrevivientes o remplazo, que consiste en renovar la población. Usualmente se seleccionan los mejores individuos (basándose en la aptitud) entre la población anterior y la descendencia generada para conformar la nueva población o, como es el caso de los Algoritmos Genéticos

Generacionales, la población es completamente remplazada por la descendencia en cada generación.

2.3.3 Métodos de evaluación

Uno de los criterios más importantes en los AG es la función de aptitud, la cual se encarga de evaluar las soluciones (individuos) que se están generando. Esta función cuantifica en qué medida los individuos cumplen con el o los criterios que el algoritmo optimiza y determinará cuáles individuos se reproducirán y los que sobrevivirán para la siguiente generación, por lo cual es de gran importancia elegir esta función cuidadosamente, ya que es la guía hacia soluciones competitivas. Un punto crucial para no impactar negativamente el rendimiento de un AG es elegir una función de aptitud que sea computacionalmente eficiente ya que la mayoría de AG deben iterar muchas veces para producir buenos resultados.

En algunos problemas de optimización, no es posible elegir una solución basada únicamente en una función objetivo, para solucionar este problema se han propuesto soluciones como Algoritmos Evolutivos Multi-Objetivo (MOEAs) que han generado resultados altamente competitivos en problemas de Optimización Multiobjetivo (MOPs). Estos problemas buscan optimizar simultáneamente dos o más funciones objetivo que generalmente están en conflicto, lo que implica que la mejora de un objetivo produce el decremento de al menos otro objetivo.

Para encontrar todas las posibles soluciones eficientes en un MOP, se emplea el concepto de Óptimo de Pareto, desarrollado por Vilfredo Pareto. Esta estrategia encuentra el conjunto de soluciones optimas, conocidas como no dominadas, dentro del espacio de soluciones. Las soluciones optimas serán aquellas que no son dominadas entre sí, y que a su vez, conforman el Frente de Pareto.

Se dice que la solución x_1 domina a la solución x_2 si y solo si se cumplen las siguientes condiciones:

1. x_1 es mejor o igual que x_2 en todas las funciones objetivo.
2. x_1 es mejor que x_2 en al menos una función objetivo.

Al realizar un análisis de dominancia entre dos soluciones es posible que no se cumplan las condiciones, cuando esto sucede, se dice que las soluciones son no-dominadas. De esta forma, si el análisis de dominancia se realiza entre todos los pares posibles dentro de un conjunto finito de soluciones se tendrá como resultado dos subconjuntos, el primero conformado por las soluciones que no son dominadas entre sí y el segundo de soluciones que son dominadas por el primero. Siendo las soluciones del primer subconjunto, las soluciones óptimas del problema que conforman el Frente de Pareto. Un ejemplo de esta técnica se observa en la Figura 2.14 donde se busca minimizar dos objetivos de un problema.

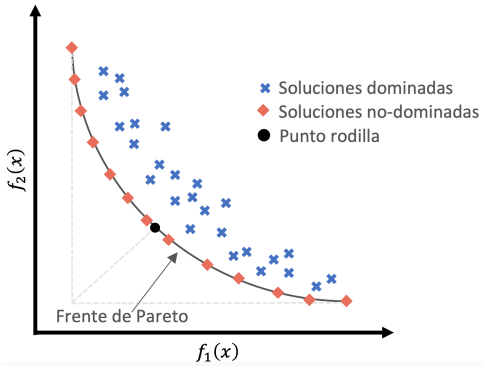

Figura 2.14. Soluciones de un problema de minimización multiobjetivo distribuidas dentro del espacio objetivo. Las soluciones se dividen en 2 subconjuntos: el Frente de Pareto (soluciones no-dominadas) y las soluciones dominadas.

2.4 Función de pérdida contrastiva

Las funciones de pérdida, también llamadas funciones de costo o funciones de error, desempeñan un rol esencial en los algoritmos de Aprendizaje Automático ya que se encargan de medir el error en las predicciones del algoritmo haciendo una cuantificación de qué tan cerca se encuentra la salida actual de la salida esperada.

Durante la etapa de entrenamiento, los algoritmos se ajustan mediante un proceso de retropropagación para intentar minimizar el error que están teniendo. El indicador de qué tan bien o mal se está autocorrigiendo el algoritmo está dado por las funciones de pérdida, cuyo valor disminuye si el ajuste se está haciendo en la dirección correcta o aumenta en caso contrario.

La elección de función de pérdida es crucial para definir los resultados de forma que sean sensibles a la aplicación que se esté estudiando. Una de las aplicaciones más comunes donde

se emplean las funciones de pérdida es en tareas de clasificación, donde Cross-entropy o Negative log-likelihood son algunas de las funciones más empleadas. Sin embargo, el estudio de estas funciones ha continuado y se han propuesto alternativas como la función de pérdida contrastiva supervisada [14] que ha logrado resultados superiores a Cross-entropy en tareas específicas como el entrenamiento de algoritmos de clasificación de imágenes.

Originalmente, la función de pérdida contrastiva fue introducida por Raia Hadsell et al. [13] en 2005 como parte de su trabajo sobre reducción de dimensionalidad. La función de mapeo que presentaron preserva la relación de vecindad entre los datos durante el mapeo y generaliza su comportamiento en nuevos datos no vistos. Dado que este comportamiento era deseable en aplicaciones donde intervienen datos de alta dimensionalidad, como imágenes que, en la mayoría de los casos, no están etiquetados, se comenzó a aplicar esta función de pérdida en aplicaciones que emplean aprendizaje autosupervisado, derivando en una nueva técnica llamada aprendizaje contrastivo.

La idea general del aprendizaje autosupervisado es que el modelo reciba datos no etiquetados ni estructurados y que los etiquete con cierto nivel de confianza. Con el aprendizaje contrastivo el modelo de aprendizaje autosupervisado es guiado utilizando un ejemplo denominado "*ancla"*, un ejemplo que pertenezca a la misma distribución que el "*ancla"*, denominado muestra "*positiva"* y otros ejemplos que pertenezca a una distribución diferente, denominados muestras "*negativas"*. Su finalidad es minimizar la distancia (en términos de distancia euclídeana, similitud coseno o alguna otra métrica) entre el "*ancla"* y la muestra *"positiva"* en el espacio y maximizar la distancia entre el "*ancla"* y las muestras *"negativas"*.

Con el paso del tiempo el aprendizaje contrastivo ha evolucionado y ahora se utiliza en entornos totalmente supervisados y semisupervisados. En estos avances, la función de pérdida contrastiva supervisada (SupCon) [14], fue desarrollada como una alternativa a la función de pérdida Cross-entropy, logrando resultados superiores en el entrenamiento de algoritmos de clasificación de imágenes. Esta técnica se desarrolla en un entorno de aprendizaje supervisado, manteniendo el principio de mapeo de ejemplos en el espacio del aprendizaje contrastivo, pero tomando ventaja de datos etiquetados.

Aunque existen diferencias marcadas entre las diversas versiones de las funciones de pérdida contrastiva, la familia de funciones de pérdida contrastiva en general considera lo siguiente: para un conjunto de N pares de muestras/etiquetas (lotes) muestreados aleatoriamente, $\{x_k, y_k\}\ k = 1,2, \dots, N$, el lote utilizado para el entrenamiento consta de 2N pares, $\{\tilde{x}_\ell, \tilde{y}_\ell\}_{\ell=1,2\dots,N}$, donde $\tilde{x}_{2k}$ y $\tilde{x}_{2k-1}$ son dos aumentos aleatorios (lotes multivista) de $x_k (k = 1,2, \dots, N)$ y $\tilde{y}_{2k-1} = \tilde{y}_{2k} = y_k$.

Dado lo anterior, la función de pérdida contrastiva autosupervisada, se calcula de la siguiente manera:

$$\mathcal{L}^{self} = \sum_{i \in I} \mathcal{L}_i^{self} = -\sum_{i \in I} \log \frac{\exp\left(z_i \cdot z_{j(i)}/\tau\right)}{\sum_{a \in A(i)} \exp\left(z_i \cdot z_a/\tau\right)} \tag{2.8}$$

donde el símbolo • denota el producto punto, τ es un parámetro escalar de temperatura, z representa las muestras, $i \in I \equiv \{1,2, \dots, N\}$ es el índice de una muestra aumentada (ancla), $j(i)$ es el índice de la otra muestra aumentada originada del mismo ejemplo llamado *positivo* y los otros $2(N-1)$ índices ($\{k \in A(i) \setminus \{j(i)\}$) son llamados *negativos* de forma que $A(i) \equiv I \setminus \{i\}$ es el conjunto de todos los índices de las muestras *negativas*.

La función de pérdida contrastiva autosupervisada ha logrado buenos resultados en casos donde no se cuenta con datos etiquetados, sin embargo, es incapaz de tratar escenarios donde sí se cuenta con etiquetas, con lo cual se sabe que más de una muestra puede pertenecer a la misma clase. Para contrarrestar esto, con la función de pérdida SupCon se propone una serie de modificaciones a la Eq. 2.8 para generalizarla a un número arbitrario de ejemplos *positivos* la cual se presenta a continuación:

$$\mathcal{L}_{in}^{sup} = \sum_{i \in I} \mathcal{L}_{in,i}^{sup} = \sum_{i \in I} -\log \left\{ \frac{1}{|P(i)|} \sum_{p \in P(i)} \frac{\exp\left(z_i \cdot z_p/\tau\right)}{\sum_{a \in A(i)} \exp\left(z_i \cdot z_a/\tau\right)} \right\} \tag{2.9}$$

donde z representa las muestras, $i \in I \equiv \{1,2, \dots, N\}$ es el índice de una muestra aumentada (ancla), $A(i) \equiv I \setminus \{i\}$ es el conjunto de todos los índices de las muestras distintos de i, $P(i) \equiv \{p \in A(i) : \tilde{y}_p = \tilde{y}_i\}$ es el conjunto de todos los índices de muestras *positivas* distintos de i, $|P(i)|$ es su cardinalidad, el símbolo • denota el producto punto y τ es un parámetro escalar de temperatura.

Agregar la función de pérdida SupCon a una RNC permite que los vectores de características se mapeen en regiones cercanas (en términos de distancia euclídea, similitud coseno o alguna otra métrica) si pertenecen a la misma clase y lejanos si pertenecen a clases distintas.

En la Figura 2.15 se presenta una comparativa entre los resultados que se tendrían al aplicar la función de pérdida contrastiva autosupervisada (izquierda, Eq. 2.8) y SupCon (derecha, Eq. 2.9) al mismo conjunto de datos. Se puede apreciar que, si se toma en cuenta la información de las etiquetas de las clases, se obtiene un espacio en el que los elementos de la misma clase se orientan en regiones más cercanas que en el caso autosupervisado.

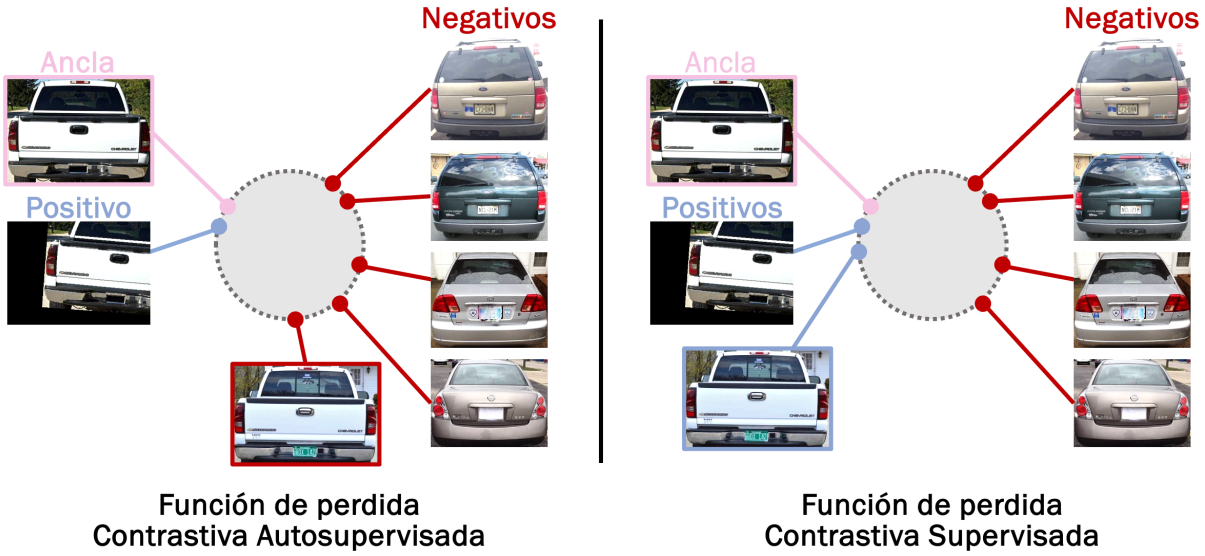

Figura 2.15. Comparativa entre los resultados que se tendrían al aplicar la función de pérdida contrastiva autosupervisada y SupCon al mismo conjunto de datos

2.5 Reconocimiento de Conjunto Abierto

Los modelos de Aprendizaje Automático (AA) han aumentado en relevancia en los últimos años, debido a sus potenciales aplicaciones en problemas del mundo real como protección contra correos de spam, reconocimiento de imágenes, sistemas de seguridad bancarios que pueden detectar fraude o incluso en redes sociales para sugerir contenido basado en nuestra actividad. Sin importar cual sea la aplicación, una de las necesidades primordiales para los modelos de AA es contar con datos relevantes al tema que se esté estudiando. Como se mencionó en la Sección 2.2, cada día tenemos acceso a más datos en distintos formatos como texto, video, audio, imagen, etc., que si bien, pueden mejorarse empleando técnicas de

preprocesamiento para mitigar inconsistencias dentro de ellos y favorecer el aprendizaje de los modelos, el tipo de aprendizaje a emplear por un modelo de AA dependerá de la tarea que se busque realizar y la estructura de los datos a los que se tiene acceso. El aprendizaje se puede clasificar dentro de 3 enfoques:

- **Aprendizaje supervisado:** Se emplea en tareas como clasificación o regresión y su característica principal es el uso de datos etiquetados para supervisar el entrenamiento del algoritmo hacia resultados más precisos.
- **Aprendizaje no supervisado:** Se emplea en tareas como clustering, asociación o reducción de dimensionalidades y su característica principal es el uso de datos no etiquetados para analizar o agrupar los datos descubriendo patrones sin necesidad de supervisión.
- **Aprendizaje por refuerzo:** Este aprendizaje no requiere de un tipo específico de datos ya que el entrenamiento se hace mediante un método de prueba y error. En este caso el agente interactúa con su entorno y dependiendo de sus acciones es recompensado por los movimientos correctos, los cuales intentará maximizar y castigado por los incorrectos, los cuales intentara minimizar.

Dentro de todo este contexto, nos enfocaremos en los algoritmos que emplean aprendizaje supervisado, específicamente los de clasificación. Estos algoritmos son entrenados para reconocer un número predeterminado de clases o categorías mediante ejemplos etiquetados que los guían para reconocer patrones que, posteriormente en su etapa de prueba, utilizarán para predecir la probabilidad de que los datos ingresados pertenezcan a una de sus categorías predeterminadas. Con estos algoritmos se puede conseguir un alto nivel de precisión de clasificación cuando son entrenados con la suficiente cantidad de datos para aprender las características esenciales que les permitirán realizar el reconocimiento. Sin embargo, la mayoría son diseñados dentro de un enfoque de *conjunto cerrado*, donde se asume que todos los datos de prueba estarán bien representados por el conjunto de entrenamiento. Con este enfoque se carece de mecanismos que detecten cuando alguno de los objetos ingresados no pertenece a alguna de las clases predefinidas. Estas situaciones imprevistas, son muy probables en escenarios de la vida real, lo que debilita drásticamente la robustez de los modelos.

Como propuesta a esta problemática se han planteado algunas estrategias como el reentrenamiento periódico de los algoritmos, la incorporación de un mecanismo de actualización incremental [24,25], el uso del aprendizaje zero-shot [26,27] o one-shot (few-shot) [23,28], etc. Si bien estas estrategias les brindan a los modelos mayor flexibilidad o la posibilidad de incrementar su potencial de clasificación, el problema fundamental de reconocer una clase novedosa al momento de ser ingresada al modelo (problema de *conjunto abierto*) no es abordada por estas estrategias.

Los primeros en describir un escenario más realista en el que *clases nuevas* no vistas en el entrenamiento aparecen en las pruebas y requiere que los clasificadores no sólo clasifiquen con precisión los objetos de *clases conocidas*, sino que también puedan lidiar efectivamente con los objetos que no fueron previstos en la etapa de entrenamiento fueron Scheirer et al. [29] quienes formalizaron este problema como *reconocimiento de conjunto abierto* (RCA) y propusieron una solución llamada máquina 1-vs-Set. Con esta propuesta se mide el riesgo de etiquetar una muestra como conocida si está lejos de los datos conocidos (*espacio abierto*) y su objetivo es minimizar este riesgo (*riesgo de espacio abierto*) rechazando objetos que se encuentren más allá del respaldo razonable de los datos conocidos.

Para lograr un entendimiento más exhaustivo del tipo de datos a los que se puede enfrentar un algoritmo de reconocimiento en situaciones de la vida real, en un trabajo posterior, los autores de [29] presentaron una categorización de clases [39] que más adelante fue ampliada por Geng et al. [40] donde se contemplan las siguientes categorías:

1. **Clases Conocidas Conocidas (CCC):** Clases con muestras de entrenamiento positivas claramente etiquetadas (que también sirven como muestras negativas para otras CCC), que incluso disponen de la información secundaria correspondiente, como información semántica, de atributos, etc.
2. **Clases Conocidas Desconocidas (CCD):** Clases con muestras negativas etiquetadas, no necesariamente agrupadas en clases significativas.
3. **Clases Desconocidas Conocidas (CDC):** Clases sin muestras disponibles en el entrenamiento, pero de las que se dispone de información lateral (por ejemplo, información semántica/de atributos) durante el entrenamiento.

4. **Clases Desconocidas Desconocidas (CDD):** Clases sin ninguna información relativa a ellas durante el entrenamiento: no sólo no vistas, sino que tampoco tienen información secundaria (por ejemplo, información semántica/de atributos, etc.) durante el entrenamiento.

La clasificación tradicional, que es el modelo dominante utilizado para los problemas de visión por computadora multiclase, sólo toma en cuenta las CCC. La inclusión de CCD da lugar a modelos con "otra clase" explícita, o a un detector entrenado con negativos sin clasificar [39]. A diferencia de la clasificación tradicional, el aprendizaje zero-shot (ZSL) y one/few-shot (FSL) toman en cuenta la información semántica compartida entre las CCC y las CDC para implementar un reconocimiento de las CDC. Sin embargo, en un escenario más realista, como el que contempla el RCA, solo se tiene acceso a información de CCC para el entrenamiento de los modelos de reconocimiento y no obstante, debe ser capaz de reconocer los objetos de CDD.

Los algoritmos de RCA tienen que considerar que su conocimiento del mundo es incompleto y formular estrategias para minimizar el *riesgo de espacio abierto* (R_O). Los autores de [29] hicieron una formulación probabilística de R_O (Eq. 2.10), como la medida relativa del *espacio abierto* (O) positivamente etiquetado comparado con la medida global del espacio positivamente etiquetado (S_O).

$$R_O(f) = \frac{\int_O f(x)\,dx}{\int_{S_O} f(x)\,dx} \tag{2.10}$$

donde f denota una función de reconocimiento medible.

A partir del trabajo presentado en [29], se han realizado numerosos estudios para minimizar el riesgo de los conjuntos abiertos y rechazar de forma más eficaz los objetos de *clases desconocidas* [30,31,32,33], que es el objetivo principal del RCA. Sin embargo, en un contexto deseable, un algoritmo de RCA debería ir más allá y descubrir las *clases desconocidas* ocultas dentro de los objetos rechazados. En este contexto, algunos autores han propuesto el uso de aprendizaje incremental [34], aprendizaje por transferencia [35,36], o Clustering [37,38]. Aunque obtuvieron buenos resultados, la mayoría de las propuestas presentan limitaciones como la determinación del número de *clases nuevas* en un evento posterior o separado del

reconocimiento de las instancias novedosas; o el uso de ejemplos de *clases desconocidas* durante las etapas de validación, pre-entrenamiento o reentrenamiento como estrategia para afinar sus representaciones/parámetros; sin embargo en el RCA casi nunca se cuenta con información de *clases desconocidas*.

2.6 Clustering

Como se mencionó en la Sección 2.5, el tipo de aprendizaje que emplean los modelos de AA dependen de la tarea que se busque realizar y la estructura de los datos a los que se tiene acceso. En las tareas donde se busca realizar agrupaciones entre datos no etiquetados, el clustering es la técnica comúnmente empleada.

El clustering se define como un modelo de aprendizaje no supervisado en el que un conjunto de objetos es agrupado en subconjuntos llamados clusters con base en alguna similitud inherente entre los datos. Esta técnica es utilizada para distintos propósitos como reconocimiento de patrones, análisis de datos, procesamiento de imágenes, procesos de clasificación, entre otros.

La estructura del clustering puede ser representada como un conjunto S de subconjuntos $S_1, S_2, \dots, S_K$ de tal forma que:

$$S_1 \cap S_2 \cap S_3 \dots \cap S_K = \phi \tag{2.11}$$

Lo cual significa que cualquier instancia en $S(S_1, S_2, \dots, S_K)$ pertenece exactamente a un único subconjunto y no puede pertenecer a ningún otro.

Distintas técnicas de clustering con diversos enfoques han sido propuestas, la más popular y simple es K-means, la cual necesita un número predefinido *k* de clusters donde a cada uno le corresponde un centroide y el proceso consiste en asignar los datos al clúster cuya media es más cercana. Este algoritmo es capaz de agrupar los datos en un proceso simple, sin embargo, existen casos donde más de un objetivo se debe cumplir para agrupar los datos y, a la vez, se requiere la determinación del número de clusters, con el cual los objetivos sean mayormente satisfechos, sea determinado automáticamente. Para cumplir este propósito, se han propuesto algoritmos de optimización multiobjetivo como MOCK [7] el cual optimiza dos objetivos: conectividad y compacidad de los grupos y determina automáticamente el número

de los k clusters. La optimización es realizada por un algoritmo evolutivo multiobjetivo llamado PESA-II (Pareto envelope-based selection algorithm II) el cual es un algoritmo de selección basado en el Frente de Pareto. Otros algoritmos genéticos multiobjetivo han sido agregados al algoritmo MOCK en sustitución de PESA-II [9] y han logrado mejorar los resultados. Algunos de los algoritmos genéticos multiobjetivo que han sido agregados a MOCK son: NSGA-II, SPEA-2 y MOEA/D. En la Sección 2.6.1, se explica más a detalle el algoritmo MOCK original y algunas de sus variaciones.

Otra limitante de algoritmos como K-means, es que no cuentan con ninguna medida de incertidumbre o probabilidad que nos indique en qué medida un punto de los datos está asociado a un clúster específico. Esta limitante ha derivado en el desarrollo de modelos probabilísticos para efectuar el clustering, como es el caso del modelo de mezclas gaussianas (MMG) el cual considera que todos los puntos de datos se derivan de una mezcla finita de distribuciones gaussianas con parámetros desconocidos. En la Sección 2.6.2 se explica más a detalle este modelo.

2.6.1 MOCK

El algoritmo original MOCK propuesto por Handl y Knowles [7] es un algoritmo de clustering multiobjetivo con determinación automática del número de clusters optimizado con un MOEA, llamado PESA-II, propuesto por Corne et al. [8]. A continuación, se presentan los aspectos específicos del algoritmo MOCK y cómo fue mejorado al sustituir PESA-II con NSGA-II (Non-dominated sorting genetic algorithm II).

2.6.1.1 Representación genética

La codificación de los individuos emplea una representación donde cada individuo g está conformado por N genes $g_1, \dots, g_N$, donde N es el tamaño del conjunto de datos a ser agrupado. Cada gen toma un valor j en el rango de 1 a N. De esta forma, el valor j asignado al i-ésimo gen representa una unión entre los datos j e i. Para la inicialización de los individuos se genera un árbol de expansión mínima con el algoritmo Prim, el cual será el primer individuo de la población. Para la generación subsecuente de la población se eliminan las (i - 1) conexiones más largas, lo cual se refleja en el genotipo asignándose a sí mismo el gen que representaba la conexión. La decodificación de esta representación requiere la identificación

de todos los subgrafos ya que todos los datos pertenecientes a cada uno son asignados a un clúster. Un ejemplo de la inicialización de individuos se muestra en la Figura 2.16, donde se observa el primer individuo generado como un árbol de expansión mínima (izquierda) y el tercer individuo generado con MOCK eliminando las 2 conexiones más largas (derecha).

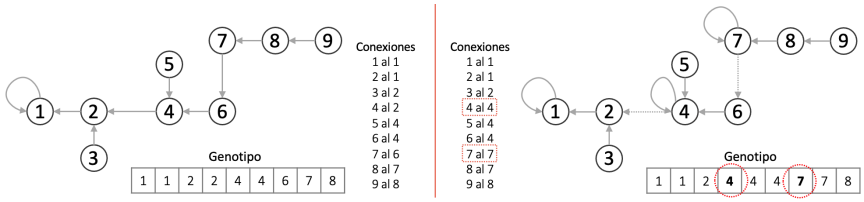

Figura 2.16. El individuo de la izquierda es el generado en la inicialización como un árbol de expansión mínima. El individuo de la derecha es el tercer individuo generado con MOCK.

2.6.1.2 Operadores de Variación

Los operadores de variación en este caso son la cruza uniforme que genera un único descendiente y el operador de mutación vecino más cercano, este operador limita el espacio de búsqueda ya que sólo podrá generar conexiones entre los vecinos más cercanos del gen que se esté mutando. Un ejemplo de cómo estos operadores de variación son aplicados en el algoritmo MOCK se muestra en la Figura 2.17, a la izquierda se muestra el ejemplo de la cruza y a la derecha la mutación.

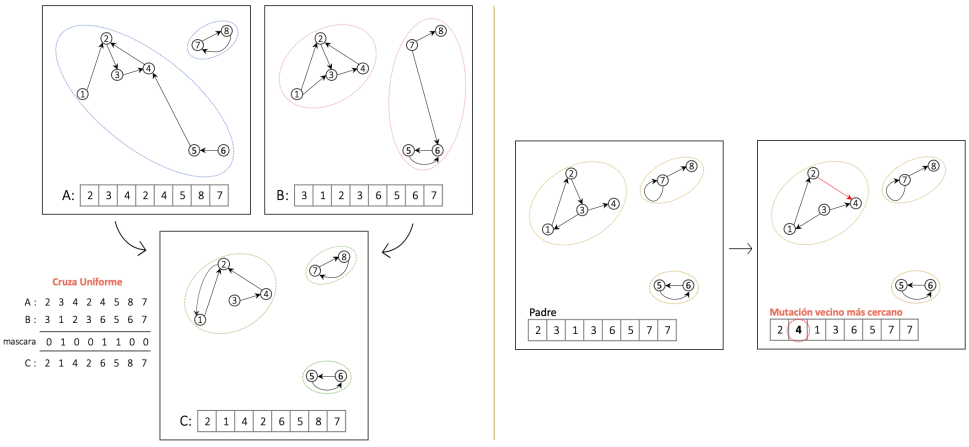

Figura 2.17. A la izquierda se muestra el proceso de cruza uniforme estándar donde dos padres generan un descendiente. A la derecha se muestra el proceso de mutación vecino más cercano donde se modifica un gen del padre.

2.6.1.3 Funciones Objetivo

En este enfoque se tienen dos funciones objetivo a ser minimizadas:

1. ***Compactación de clusters*** o desviación global la cual se calcula sumando las distancias totales entre cada dato y su correspondiente centroide en un grupo determinado y se representa matemáticamente como:

$$Dev(C) = \sum_{C_k \epsilon C} \sum_{i \epsilon C_k} \delta(i, \mu_k) \tag{2.12}$$

   donde C es el conjunto de clusters, μ_k es el centroide del cluster C_k y $\delta(.,.)$ es una medida de distancia.

2. ***Conectividad de clusters*** que evalúa el grado en que los vecinos más cercanos de un elemento se han colocado en el mismo clúster del elemento actual, se representa matemáticamente como:

$$Conn(C) = \sum_{i=1}^{N} \left(\sum_{j=1}^{L} x_{i,nn_i(j)} \right), donde\ x_{r,s} = \begin{cases} \frac{1}{j}\ si\ \nexists C_k : r, s \in C_k \\ 0\ de\ lo\ contrario \end{cases} \tag{2.13}$$

   donde C es el conjunto de clusters, N es la cantidad de datos en el dataset, L es la cantidad de vecinos más cercanos (parámetro definido por el usuario), $nn_i(j)$ es el j-ésimo vecino más cercano y $x_{r,s}$ es la función de penalización, donde solo habrá penalizaciones si algún j-ésimo vecino más cercano no se encuentra en el mismo clúster que el i-ésimo dato.

2.6.1.4 PESA-II

En la versión original de MOCK se empleó el algoritmo evolutivo multiobjetivo PESA-II, el cual emplea dos poblaciones de soluciones: IP que tiene un tamaño fijo y está encargada de explorar nuevas soluciones y EP que tiene un tamaño limitado, pero no fijo y tiene como trabajo explotar buenas soluciones. En otras palabras, EP será utilizado como un archivo de soluciones no-dominadas (Frente de Pareto) y en cada generación se realizará un análisis de dominancia para incorporar las soluciones de IP a EP que no sean dominadas por alguna existente en el archivo, tomando en cuenta que se preferirán las soluciones que no se

encuentren en “nichos” muy poblados. El funcionamiento general de PESA-II se describe en el Algoritmo 2 (tomado y traducido directamente de [9]).

Algoritmo 2. PESA-II.

```
1  Inicializar aleatoriamente la población (interna) IP
2  Evaluar cada miembro de IP
3  Inicializar la población (externa) EP como un conjunto vacío
4  Repetir
5     Incorporar los vectores no dominantes de IP a EP
6     Se elimina el contenido de IP
7     Repetir
8        Con probabilidad Pc, se seleccionan dos padres de EP, donde Pc = Probabilidad de
9        cruza
10       Se produce un solo hijo con el proceso de cruza
11       Se muta el hijo creado en el paso anterior
12       Con probabilidad 1-Pc, se selecciona un padre
13       Se muta el padre para producir un hijo
14    Hasta que la población de IP esta completa;
15 Hasta que se llega a la condición de paro;
16 Regresa como resultado los miembros de EP
```

2.6.1.5 NSGA-II

Desde la publicación de MOCK, se han presentado variaciones y mejoras como el caso de Martínez-Peñaloza et al. [9] quienes lograron mejorar los resultados obtenidos en la versión original, que empleaba PESA-II, generando 3 nuevas versiones del algoritmo MOCK empleando 3 algoritmos evolutivos multiobjetivo distintos: NSGA-II, SPEA-2 y MOEA/D. Estas modificaciones fueron comparadas entre ellas y contra la implementación original basada en PESA-II y sus resultados los llevaron a concluir que NSGA-II genera un mejor desempeño.

NSGA-II ordena la población de individuos en distintos niveles de no-dominancia, el proceso de este algoritmo es el siguiente: para el primer nivel se comparan todos los pares de soluciones posibles dentro de la población, para encontrar aquellas que no sean dominadas entre sí, para el siguiente nivel se realiza el mismo proceso, sin tomar en cuenta las soluciones pertenecientes al primer nivel. Este proceso se repite consecutivamente hasta asignar todas las soluciones a un nivel específico. Para la selección de padres se emplea

Torneo Binario y para la generación de la descendencia se aplican los operadores de variación (cruza y mutación).

Para mantener la diversidad y ayudar al algoritmo a explorar el paisaje de aptitud, NSGA-II emplea una distancia de aglomeración la cual es calculada para cada individuo tomando sus vecinos más cercanos pertenecientes al mismo frente como vértices de un cuboide (figura geométrica de 6 caras, 8 vértices y 12 aristas) dentro del cual está contenida la solución que está siendo analizada, siendo la longitud lateral media del cubo la distancia de aglomeración. En la Figura 2.18 se muestra un ejemplo de este proceso para un problema de 2 objetivos y en el Algoritmo 3 (tomado y traducido directamente de [9]) se muestra el funcionamiento general de NSGA-II.

Algoritmo 3. NSGA-II.

```
1  Inicializar la población
2  Generar la población aleatoriamente de tamaño M
3  Evaluar las funciones objetivo
4  Asignar el rango (nivel) basándose en la dominancia de Pareto - "ordenar"
5  Generar descendencia
6     Realizar la selección con Torneo Binario
7     Realizar cruza y mutación
8  para i=1 a Numero de generaciones hacer
9     para cada padre e hijo en la población hacer
10       Asignar el rango (nivel) basándose en la dominancia de Pareto - "ordenar"
11       Generar conjuntos de soluciones no-dominadas
12       Determinar la distancia de aglomeración
13       Bucle (interno) añadiendo soluciones a la siguiente generación iniciando desde el
14       "primer" frente hasta encontrar M individuos.
15    Termina
16 Seleccionar los puntos (elitista) de la parte inferior del frente (con menor rango) que
17 estén fuera de una distancia de aglomeración.
18 Crear la siguiente generación
19    Realizar la selección con Torneo Binario
20    Realizar cruza y mutación
21 Termina
```

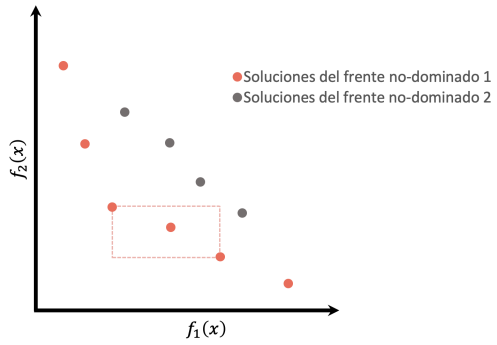

Figura 2.18. Representación de la distancia de aglomeración.

2.6.2 Modelo de Mezclas Gaussianas

El MMG es un modelo probabilístico que puede ser visto como una generalización de las distribuciones gaussianas ya que usualmente es empleado para modelar conjuntos de datos del mundo real que puedan agruparse en múltiples subconjuntos. Este método supone que cada subconjunto de datos puede modelarse mediante una distribución gaussiana y que la distribución que modela el comportamiento de los datos es una mezcla de distribuciones gaussianas finitas con parámetros desconocidos.

Matemáticamente, la función de densidad de un MMG es una suma ponderada de K componentes gaussianas descrita de la siguiente manera:

$$p(x) = \sum_{j=1}^{K} w_j \mathcal{N}(x|\mu_j, \sum_j) \tag{2.14}$$

donde w_j es la probabilidad a priori del j-ésimo componente de la mezcla que debe satisfacer $0 \leq w_j \leq 1$ y $\sum_{j=1}^{K} w_j = 1$. $\mathcal{N}(x|\mu_j, \sum_j)$ es la función de densidad gaussiana del j-ésimo componente de la mezcla, con media μ_j y matriz de covarianza $\sum_j$. La función de densidad gaussiana para un vector $x = (x_1, x_2, \dots, x_d)^T$ en dimensión d, es definida de la siguiente manera:

$$\mathcal{N}(x|\mu, \sum) = \frac{1}{(2\pi)^{d/2}|\sum|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \sum^{-1}(x-\mu)} \tag{2.15}$$

donde μ es la media de la gaussiana y $\sum$ es su matriz de covarianza.

Para simplificar el problema a 2 dimensiones podemos asumir que tenemos un conjunto de datos conformado por vectores $z = (x, y)$ que se distribuyen normalmente de forma

bivariada con componentes de media μ_1 y μ_2, componentes de varianza σ_1^2 y σ_2^2, componentes de desviación estándar σ_1 y σ_2 y una correlación de ρ. La matriz de covarianza es igual al producto de las varianzas por 1 menos la correlación al cuadrado:

$$|\sum| = \sigma_1^2\sigma_2^2(1-\rho^2) \tag{2.16}$$

y la inversa de la matriz de covarianza se calcula la siguiente forma:

$$\sum^{-1} = \frac{1}{\sigma_1^2\sigma_2^2(1-\rho^2)}\begin{pmatrix}\sigma_2^2 & -\rho\sigma_1\sigma_2\\ -\rho\sigma_1\sigma_2 & \sigma_1^2\end{pmatrix} \tag{2.17}$$

Finalmente, después de sustituir las expresiones anteriores en la Eq. 2.15, se obtiene la función de densidad para una distribución gaussiana bivariada, que se expresa en la Eq. 2.18 y con la cual se puede calcular la probabilidad de algún vector $z = (x,y)$ en una distribución gaussiana bivariada.

$$\phi(x,y) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}}e^{-\frac{1}{2(1-\rho^2)}\left[\left(\frac{x-\mu_1}{\sigma_1}\right)^2-2\rho\left(\frac{x-\mu_1}{\sigma_1}\right)\left(\frac{y-\mu_2}{\sigma_2}\right)+\left(\frac{y-\mu_2}{\sigma_2}\right)^2\right]} \tag{2.18}$$

En la Figura 2.19 se puede apreciar la función de densidad de una distribución gaussiana bivariada con $\mu_1 = 0, \mu_2 = 0, \sigma_1 = 0.25$ y $\sigma_2 = 1$.

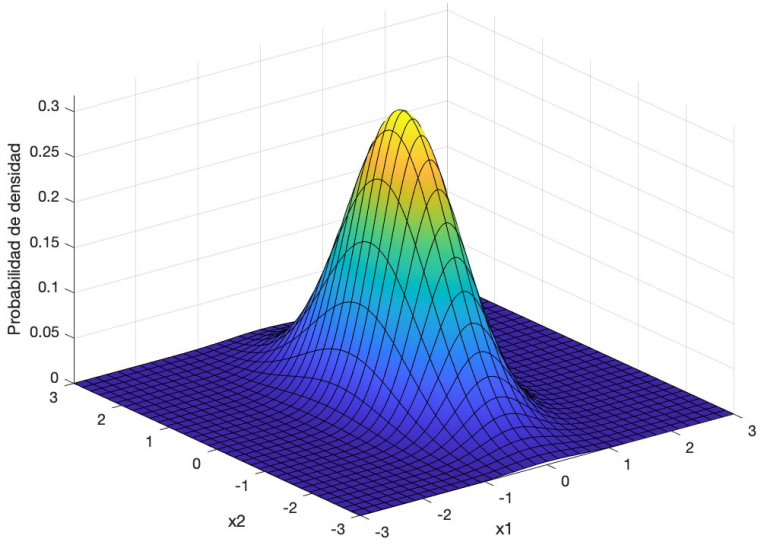

Figura 2.19. Distribución gaussiana bivariada con $\mu_1 = 0$, $\mu_2 = 0$, $\sigma_1 = 0.25$ y $\sigma_2 = 1$.

En la Figura 2.20 se puede apreciar la distribución de un MMG bivariado de dos componentes, donde el primer componente tiene valores $\mu_1 = 1, \mu_2 = 2, \sigma_1 = 2$ y $\sigma_2 = 0.5$ y el segundo componente tiene valores $\mu_1 = -3, \mu_2 = -5, \sigma_1 = 1$ y $\sigma_2 = 1$.

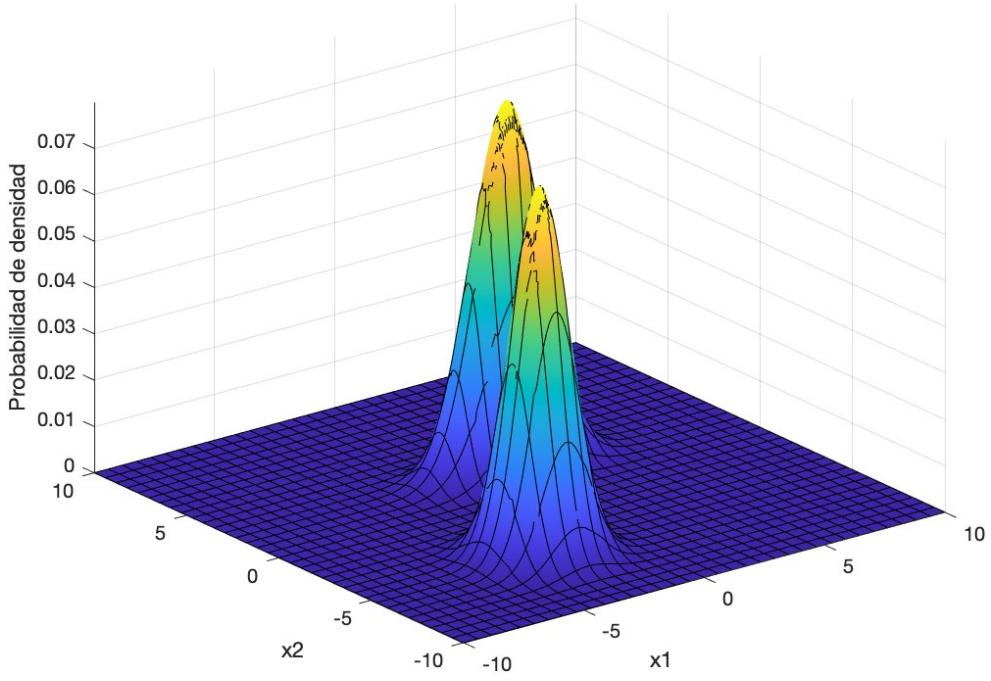

Figura 2.20. Distribución de un MMG bivariada de dos componentes.

2.7 Resumen del capítulo

En este capítulo se abordaron los fundamentos teóricos básicos de las principales técnicas y herramientas que serán empleadas en esta tesis. Se explicó en detalle el funcionamiento y las estructuras que conforman las Redes Neuronales Convolucionales, así como uno de los principales algoritmos empleados para el diseño de estas arquitecturas que es la Neuroevolución, donde también se dieron detalles de la implementación específica de NE que será empleada en este trabajo.

En este capítulo también se dio la información necesaria para entender el concepto de Reconocimiento de *conjunto abierto* y cómo se relaciona a este trabajo. De igual manera, se cubrieron temas de interés para la metodología que se desarrollara como son la Función de pérdida contrastiva y el Clustering.

Capítulo 3

Marco Referencial

En este capítulo se presentan algunos de los enfoques del estado del arte relevantes al tema de esta tesis. Se citan trabajos previos relacionados al Reconocimiento de Marca y Modelo de Vehículos, el *Reconocimiento de conjunto abierto,* el Clustering, la Neuroevolución y la Función de pérdida contrastiva.

3.1 Reconocimiento de marca y modelo de vehículos

Como cualquier algoritmo de reconocimiento, los específicos al reconocimiento de marca y modelo de vehículos tienen retos específicos, para los cuales se han presentado soluciones mediante la aplicación de diferentes enfoques, metodologías y técnicas. Algunas de las problemáticas más comúnmente abordadas en trabajos de RMMV son el reconocimiento de diferentes modelos de vehículos con apariencia similar [18,21], lidiar con variaciones en las imágenes debido a las condiciones climáticas [3,20], reconocimiento a través de diferentes puntos clave o regiones (vista delantera, vista trasera, sección de parrilla, etc.) [15,17], entre otros.

Independientemente del reto de RMMV que se intente resolver, siempre se buscará contar con una base de datos funcional para el propósito. Algunas de las bases de datos de acceso libre más utilizadas en la literatura especializada son Stanford Cars Dataset [4], the car connection picture dataset [5], VMMRdb [6], etc., que contienen imágenes de vehículos de distintas marcas, modelos y años tomadas en diferentes horarios y condiciones climatológicas. Desafortunadamente, hasta el momento ninguna de las bases de datos de acceso libre ha sido capaz de cubrir la totalidad de marcas y modelos existentes en circulación con ejemplos suficientes de cada clase para el correcto entrenamiento de algoritmos de clasificación que emplean aprendizaje supervisado (el más empleado para esta tarea).

La limitante en base de datos que enfrentan la mayoría de problemas de reconocimiento/clasificación del mundo real, como lo es el RMMV, aunado a que la mayoría de soluciones para el problema específico de RMMV han sido diseñadas dentro de un enfoque de *conjunto cerrado,* genera que las soluciones sean susceptibles a clasificar objetos de clases que no se vieron en el entrenamiento dentro de alguna de las clases predefinidas, lo que debilita drásticamente la robustez de los modelos. Los pocos trabajos existentes de RMMV que toman en cuenta esta problemática y disponen de algún tipo de mecanismo para tratar instancias de *clases desconocidas* fueron propuestos por Nazemi et al. [2] y Kezebou et al. [23].

En [2] el sistema base es capaz de clasificar 50 modelos específicos de vehículos, al que le fue añadido una detección de anomalías basada en un umbral de confianza para identificar los vehículos que no pertenecen a ninguna de las 50 clases. Las "anomalías" se clasifican a su vez dentro de 2 clases "genéricas" con base en sus dimensiones: "Desconocido pesado" y "Desconocido ligero". Esta propuesta disminuye el riesgo de etiquetar un vehículo de clase desconocida dentro de alguna clase conocida, sin embargo, no descubre las clases específicas de los objetos que no superaron el umbral, por lo que los objetos de las clases "*desconocidas*" deberán ser posteriormente clasificadas manualmente por personal humano con conocimiento específico. Por otra parte, su modelo necesita ser entrenado con numerosos ejemplos de todas las clases incluyendo las 2 "*desconocidas*" por lo que esencialmente esta propuesta solo presenta un clasificador de 52 clases.

En [23] emplearon un enfoque de aprendizaje Few-Shots que requiere entre 1 y 20 imágenes para la generación de *clases nuevas*. Si bien esta estrategia también presenta una alternativa interesante, la detección de "*clases nuevas*" necesita el apoyo de un conjunto de datos de "consulta" etiquetados que contenga ejemplos de las clases que durante las pruebas fungirán como "*desconocidas*" para poder determinar su clase y, de no existir tales ejemplos, se generará una clasificación errónea, por lo que esta solución solo es efectiva si las *clases desconocidas* cuentan con referencias *conocidas* etiquetadas.

3.2 Reconocimiento de conjunto abierto

El RCA [29] introdujo un escenario más realista para las tareas de reconocimiento/clasificación del mundo real, en el que *clases nuevas* no vistas durante el entrenamiento pueden aparecer durante las consultas. Para hacer frente a estas situaciones imprevistas, los algoritmos de RCA consideran que su conocimiento del mundo se encuentra incompleto y deben contar con alguna estrategia para minimizar el *riesgo del espacio abierto*. Los autores de [29] presentaron una solución llamada maquina 1-vs-Set donde se mide el riesgo de etiquetar una muestra como conocida si se encuentra lejos de los datos conocidos (*espacio abierto*) y su objetivo es minimizar este riesgo (*riesgo de espacio abierto*) rechazando las consultas que se encuentran más allá del soporte razonable de los datos conocidos. A partir de esta propuesta, se han realizado numerosos estudios para minimizar el *riesgo de conjunto abierto* y rechazar más eficazmente los objetos de *clases desconocidas* [30,31,32,33], que es el objetivo principal del RCA. Sin embargo, en un contexto más deseable, el RCA debería ir más allá y descubrir las *clases desconocidas* ocultas dentro de los objetos rechazados. En este contexto, algunos autores han propuesto el uso de aprendizaje incremental [34], aprendizaje por transferencia [35,36], o Clustering [37,38].

Bendale y Boult [34] ampliaron el concepto del problema de *Reconocimiento de conjunto abierto* al problema de *Reconocimiento de Mundo Abierto* (RMA) para considerar conjuntamente el RCA y el aprendizaje incremental de *clases nuevas*. Propusieron que un sistema de RMA eficaz debe realizar cuatro tareas: detectar objetos desconocidos, elegir qué objetos etiquetar para añadirlos al modelo, etiquetar los objetos y actualizar el modelo. En su trabajo presentaron el algoritmo de RMA llamado Nearest Non-Outlier (NNO), sin embargo, no todas las tareas que proponen están automatizadas en NNO, por un lado, requieren de supervisión humana para el etiquetado y por otra parte, la determinación del número de clases ocurre en un evento posterior tras el reconocimiento de las nuevas instancias. En [35] Wang et al. estudian el problema RMA con más detalle incorporando el aprendizaje por transferencia para transferir el conocimiento de las clases antiguas a las nuevas. Sin embargo, necesitan reentrenar su modelo con la presencia de muestras de *clases desconocidas*, lo que supone una limitación ya que, en un contexto de RCA, la información de

clases desconocidas casi nunca está disponible. Una propuesta similar de transferencia de conocimiento fue introducida por Han et al. [36], sin embargo, tienen la misma limitación que [35] ya que su idea es pre-entrenar su modelo con imágenes de *clases conocidas* y *desconocidas*. Otra propuesta interesante fue desarrollada en [37] por algunos autores de [36] donde también aprovechan el enfoque de transferencia de conocimiento, pero utilizando clustering. La principal limitante de este trabajo es que la determinación del número de *clases nuevas* sucede en un evento separado del descubrimiento de nuevas instancias lo que, como en [34], puede llevar a soluciones subóptimas.

Hasta donde tiene conocimiento el autor, el trabajo más relacionado a esta tesis en términos de descubrir simultáneamente los objetos de *clases nuevas* y sus clases fue presentado en [38] donde introducen un marco de RCA basado en una estrategia de decisión colectiva/por lotes (CD-OSR) modificando ligeramente el proceso jerárquico de Dirichlet (HDP). CD-OSR realiza primero un proceso de co-clustering en la fase de entrenamiento para obtener los parámetros adecuados. En la fase de prueba, modela cada clase conocida como un grupo utilizando un Modelo de Mezcla Gaussiana con un número desconocido de subclases (se pueden obtener una o más subclases que representen la misma clase) y trata todo el conjunto de prueba (colectivo/lote) del mismo modo. A continuación, todos los grupos se agrupan conjuntamente en el marco del HDP y cada grupo se etiqueta como alguna de las *clases conocidas* o como desconocida, dependiendo de si la subclase que se le asigna está asociada a una clase conocida o no. Otros trabajos de RCA como [41] también han aprovechado las distribuciones gaussianas para obtener representaciones discriminativas de los datos con el fin de detectar desconocidos y clasificar conocidos.

Otra propuesta que puede estar relacionada con este trabajo se presentó en [33], donde el problema de RCA se aborda dentro de un enfoque de aprendizaje de transferencia utilizando el Aprendizaje Contrastivo para modelar los datos. También en este trabajo se menciona la importancia de desarrollar y probar soluciones de RCA con bases de datos de dominios específicos para probar su eficiencia en el tratamiento de aplicaciones del mundo real. Desafortunadamente, este trabajo solo rechaza objetos de *clases nuevas* y no incluye el descubrimiento de sus clases.

3.3 Neuroevolución y Función de pérdida contrastiva

La Neuroevolución ha logrado resultados destacados en la tarea de optimizar las RNAs a distintos niveles y ha avanzado rápidamente hacia la optimización de topologías de RNCs [42,43,44,45,46,39,22,18]. Un punto crucial en el rendimiento de los algoritmos de NE son las codificaciones neuronales, que contienen la información topológica de una RNA y, por tanto, tienen un gran impacto en la complejidad del espacio de búsqueda.

Para poder implementar esta técnica en RNCs, la NE se enfrentó al problema de diseñar codificaciones neuronales que pudieran abstraer los parámetros de las RNC y poder tratar con estas arquitecturas tan complejas. Existen dos tipos de codificaciones neuronales que se emplean habitualmente en la NE de RNCs: directa e indirecta. Entre las propuestas que utilizan un marco de codificación indirecta, se encuentran trabajos como [42,43,44]. Y en el caso de los trabajos que utilizaron codificaciones indirectas, se encuentran propuestas como [45,46,39]. En los últimos años se ha empezado a estudiar una codificación neuronal "Híbrida" que combina elementos de las codificaciones directas e indirectas para eliminar algunas de sus limitaciones. Estas representaciones "Híbridas" han demostrado ser muy útiles para distribuir las representaciones de las RNC en diferentes subestructuras generando una mejora en la búsqueda [10,22,18]. Las ventajas y desventajas de los diferentes esquemas de codificación, así como importantes nichos de oportunidad para futuras investigaciones fueron descritos en detalle en [17].

Por otra parte, aunque los algoritmos de NE tienen una estrategia encargada de determinar que tan bien, los individuos, cumplen el criterio o criterios que se están optimizando (función de aptitud), las RNCs tienen su propia estrategia, llamada función de pérdida, para cuantificar qué tan cerca están sus predicciones de la salida esperada. Cross-entropy o Negative log-likelihood son algunas de las funciones más utilizadas, sin embargo, el estudio de estas funciones ha continuado y se han propuesto alternativas que han conseguido resultados superiores. En estos avances, la función de pérdida contrastiva supervisada (SupCon) [14] fue desarrollada siguiendo el enfoque del Aprendizaje Contrastivo, pero en un entorno supervisado, lo que le permitió mantener el principio de mapeo de ejemplos en el espacio del aprendizaje contrastivo (la distancia se minimiza en términos de distancia

euclidiana, similitud coseno, etc. entre objetos similares y se maximiza para objetos disímiles) pero aprovechando los datos etiquetados.

3.4 Clustering

Los algoritmos de clustering son la principal herramienta para el análisis exploratorio de datos debido a su capacidad para detectar patrones en los datos y agruparlos basándose en ellos. Uno de los algoritmos de clustering más antiguos es K-means el cual particiona un conjunto de datos en k grupos en el que cada dato pertenece a un único grupo cuyo valor medio sea el más cercano. Si bien este algoritmo continúa siendo influyente hoy en día, conlleva múltiples limitaciones como la determinación manual del número k de grupos, la optimización de un único objetivo y no contar con ninguna medida de incertidumbre o probabilidad que nos indique en qué medida un punto está asociado a un clúster específico.

Desde la introducción de K-means se han desarrollado múltiples algoritmos de clustering más eficientes y competentes aplicando técnicas de diversas disciplinas para presentar soluciones a retos como optimización multiobjetivo o la determinación automática del número de grupos. Uno de los algoritmos que presentó una solución a estos retos es el algoritmo multiobjetivo con determinación automática del número de clusters (MOCK) propuesto por Handl y Knowles [7] el cual emplea un algoritmo evolutivo multiobjetivo (MOEA) llamado PESA-II con el cual se lograron resultados competitivos. Sin embargo, desde su publicación se han presentado variaciones y mejoras como el caso de Martínez-Peñaloza et al. [9] quienes lograron mejorar los resultados obtenidos en la versión original generando 3 nuevas versiones del algoritmo MOCK empleando 3 algoritmos evolutivos multiobjetivo distintos: NSGA-II, SPEA-2 y MOEA/D. Estas modificaciones fueron comparadas entre ellas y contra la implementación original basada en PESA-II y sus resultados los llevaron a concluir que con NSGA-II se obtiene un mejor desempeño.

Otra estrategia interesante para efectuar clustering se desarrolló mediante el uso de modelos probabilísticos, como es el caso del modelo de mezclas gaussiana (MMG) el cual considera que todos los puntos de datos se derivan de una mezcla de distribuciones gaussianas finitas con parámetros desconocidos y, dado un número suficiente de datos de la mezcla, MMG puede modelar distribuciones de probabilidad arbitrarias.

3.5 Resumen del capítulo

En este capítulo se presentó la revisión del estado del arte de trabajos previos relacionados a la propuesta de esta tesis. Se presentó el marco referencial donde se citaron trabajos previos relacionados al Reconocimiento de Marca y Modelo de Vehículos, haciendo énfasis en aquellos trabajos que abordaron con alguna estrategia el mismo problema que se aborda en este trabajo. También se mencionaron los trabajos de Reconocimiento de *conjunto abierto* más relevantes y aquellos que de alguna forma se relacionan con el enfoque que se propone en este trabajo.

Por último, también se cubrió la revisión de los trabajos previos de las herramientas que serán empleadas y harán posible el funcionamiento general de la presente propuesta como son: la Neuroevolución, el Clustering y la Función de pérdida contrastiva.

Capítulo 4

Propuesta de Solución (Metodología)

En este capítulo se presenta la metodología propuesta para abordar el RMMV como un problema de RCA con una extensión para el descubrimiento de *clases nuevas*. El proceso general de la propuesta se muestra en la Figura 4.1 y a lo largo de este capítulo se describirá la metodología experimental seguida para obtener y analizar los resultados empíricos, la cual se encuentra distribuida en las siguientes secciones: (1) preprocesamiento y distribución de la base de datos, (2) proceso de neuroevolución con la función de pérdida contrastiva, (3) entrenamiento de la RNC generada con el proceso de NE, (4) MMG con MOCK/NSGA-II y (5) clasificación de *clases conocidas*.

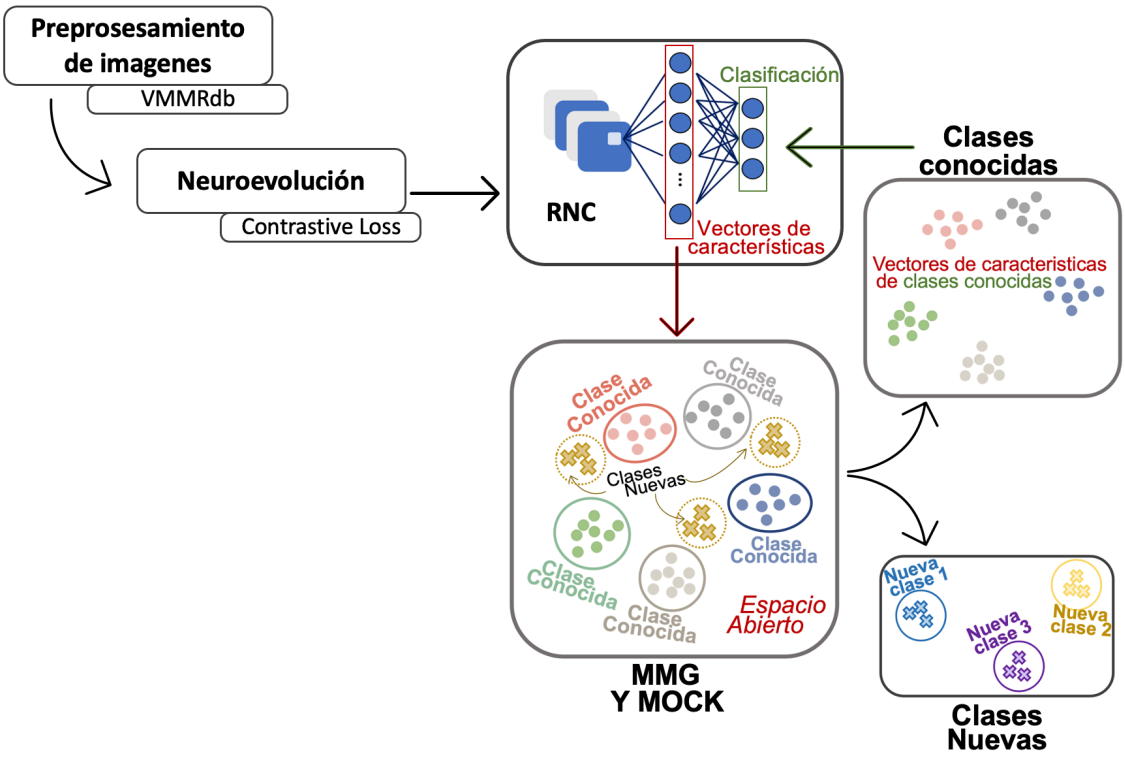

Figura 4.1. Proceso general propuesto para abordar el RMMV como un problema de RCA con una extensión para el descubrimiento de clases nuevas.

4.1 Preprocesamiento y distribución de base de datos

En este trabajo se utilizó la base de datos The Vehicle Make and Model Recognition dataset (VMMRdb) [6], por ser una de las más citadas en la literatura especializada, la cual se puede encontrar en el siguiente repositorio: https://github.com/faezetta/VMMRdb. Esta base de datos contiene 9,170 clases, compuestas por 291,752 imágenes, que abarcan modelos fabricados entre 1950 y 2016. Las imágenes fueron tomadas desde diferentes dispositivos, ángulos, condiciones climatológicas y algunas contienen fondos irrelevantes.

Para el presente trabajo se seleccionaron las cinco clases de modelos distintos (sin tomar en cuenta el año de fabricación) de la base de datos VMMRdb que tuvieran la mayor cantidad de ejemplos por clase: Chevrolet Silverado 2004, Ford Explorer 2002, Ford Mustang 2000, Honda Civic 2002 y Nissan Altima 2005. En adelante se hará referencia a estas clases como "*clases conocidas*". Adicionalmente se seleccionaron tres clases aleatorias de la misma base de datos: Acura RSX 2003, Chevrolet Avalanche 2009 y Ford Escape 2011 que en adelante se referenciaran como "*clases desconocidas*".

Las imágenes de las *clases conocidas* se filtraron manualmente para conservar las imágenes únicas y que mostraran la vista trasera de los vehículos. De las *clases desconocidas,* se eligieron 3 imágenes únicas de cada clase que mostraran la vista trasera de los vehículos. Dado que para este estudio la información de fondo no era relevante, se extrajo la región de interés (ROI), de las imágenes filtradas de las *clases conocidas* y *desconocidas*, la cual se conformaba por la porción de las imágenes que contenían únicamente al vehículo. Finalmente, las imágenes resultantes de los procesos anteriores se transformaron a escala de grises, se redimensionaron a 28x28 píxeles y se normalizaron (de acuerdo con la recomendación de Pytorch) con una media de 0.456 y una desviación estándar de 0.224.

Después del proceso de filtrado mencionado anteriormente, el número de muestras funcionales por *clase conocida* variaba entre 44-250 imágenes, de las cuales, se reservaron tres imágenes de cada *clase conocida* como conjunto de prueba para validar la metodología general propuesta. De las imágenes restantes, seis de cada *clase conocida* fueron empleadas en el proceso NE para diseñar la RNC de dominio específico de RMMV.

Posteriormente, las imágenes de cada *clase conocida* sin contar el conjunto de prueba se sometieron a un proceso de aumento de datos para equilibrar el número de ejemplos por clase, lo que dio como resultado 250 imágenes de cada *clase conocida*. De las cuales 200 imágenes de cada *clase conocida* (80%) fueron empleadas para el entrenamiento de la RNC de dominio específico y para modelar las *clases conocidas* con un Modelo de Mezclas Gaussianas (MMG). Las 50 imágenes restantes de cada *clase conocida* (20%) se utilizaron para probar la precisión de clasificación de la RNC de dominio específico y para definir un umbral de *clases conocidas* en el MMG.

Las tres imágenes únicas de las *clases desconocidas* se agregaron al conjunto de prueba para utilizarlas únicamente durante la fase de prueba y validar que la metodología propuesta puede detectarlas y descubrir sus clases. En la imagen 4.2 se muestran las imágenes que conforman el conjunto de prueba y en la Tabla 4.1 se muestra el detalle de la cantidad de muestras y como se distribuyeron en conjunto de prueba y entrenamiento, así como, en *clases conocidas* y *desconocidas*.

Figura 4.2. Imágenes que conforman el conjunto de prueba. A la izquierda se muestran las imágenes de las clases conocidas y a la derecha imágenes de las clases desconocidas.

Tabla 4.1. Distribución de las muestras de la base de datos VMMRdb empleadas en la metodología para abordar el RMMV como un problema de RCA con una extensión para el descubrimiento de clases nuevas.

Clase				Muestras en VMMRdb	Muestras Vista trasera	Muestras Conjunto Prueba	Muestras Conjunto de Entrenamiento (A)	Muestras Aumentadas a partir de A (B)	Muestras Conjunto de Entrenamiento Aumentado (A+B)
Tipo	**Marca**	**Modelo**	**Año**						
Conocidas	Chevrolet	Silverado	2004	487	44	3	41	209	250
	Ford	Explorer	2002	883	191	3	188	62	250
	Ford	Mustang	2000	504	157	3	154	96	250
	Honda	Civic	2002	535	98	3	95	155	250
	Nissan	Altima	2005	716	250	3	247	3	250
Desconocidas	Acura	RSX	2003	74	3*	3	0	0	0
	Chevrolet	Avalanche	2009	6	3*	3	0	0	0
	Ford	Escape	2011	56	3*	3	0	0	0

[1] Las celdas marcadas con * indican que existía un número mayor de muestras en la base de datos, pero para propósitos de este estudio, únicamente se eligieron 3.

4.2 Neuroevolución y la función de pérdida contrastiva

Uno de los principales objetivos de este trabajo es explotar la capacidad de las RNCs de extraer características significativas de las imágenes para diseñar un mecanismo que permita detectar instancias de *clases desconocidas,* basando en la distribución consistente de los vectores de características en el espacio. Para facilitar la interpretación de los vectores en el espacio, en este trabajo se buscaba que los vectores de características extraídos por una RNC se encuentren mapeados espacialmente cercanos, en términos de similitud coseno (Eq. 4.1), si pertenecen a la misma clase y lejanos si pertenecen a clases distintas, manteniendo este comportamiento incluso si las clases son *desconocidas*. De acuerdo con la revisión del estado del arte, añadir la función de pérdida contrastiva a una RNC genera que los vectores de características mapeados cumplan con estas características.

$$similitud\ coseno = \cos\theta = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \qquad (4.1)$$

donde A y B son vectores de dimensión n.

Por otra parte, aunque existen arquitecturas de RNCs como VGGNet, AlexNet, GoogLeNet, ResNet, etc. que han alcanzado excelentes resultados en los benchmarks más conocidos

como ImageNet, CIFAR-100, etc., en este trabajo se propone una nueva arquitectura de dominio específico al problema que se está abordando (RMMV), la cual generará el comportamiento descrito anteriormente en los vectores de características. Para el diseño de esta nueva arquitectura se empleó el algoritmo de NE llamado DeepGA [10], el cual se muestra en el Algoritmo 1, combinado con la función de pérdida SupCon [14] la cual se expresa en la Eq. 2.9, con una muestra de seis imágenes de cada *clase conocida* como se menciona en la Sección 4.1.

En el artículo donde presentan la función de pérdida SupCon [14], los autores pusieron a disposición general del público su implementación de SupCon en Pytorch, la cual se puede encontrar en el siguiente repositorio: https://t.ly/supcon. Para fines del presente trabajo, se modificó el algoritmo de DeepGA para emplear la implementación de SupCon en Pytorch como función de pérdida en las RNCs generadas en el proceso de NE (En la versión original de DeepGA se empleaba la función de pérdida negative log-likelihood). Otra modificación al algoritmo de DeepGA fue sobre su función de aptitud (Algoritmo 1, línea 15), en la implementación original de DeepGA el objetivo era maximizar la precisión de clasificación y al mismo tiempo generar RNCs computacionalmente eficientes. En el presente trabajo el objetivo que se buscaba optimizar era el comportamiento consistente en los vectores de características (cercanos si pertenecen a la misma clase y lejanos si son de clases distintas), el cual se podía medir con SupCon, por lo que se tomó el valor de SupCon en la última época de entrenamiento de cada RNC generado durante el proceso evolutivo como valor de aptitud de cada individuo. Dado que la función de pérdida (que en este caso también es la función de aptitud) disminuye a medida que las redes se acercan a la salida deseada, DeepGA se configuró para trabajar como un problema de minimización.

Como se mencionó en la Sección 2.3.1, la codificación híbrida empleada en DeepGA le permite al algoritmo considerar en su búsqueda de la mejor solución, el número de capas totalmente conectadas y su correspondiente número de neuronas. Sin embargo, al ejecutar el proceso de NE, se detectó que permitirle al algoritmo considerar en su búsqueda el número de capas totalmente conectadas generaba un aumento en la complejidad y el tiempo de ejecución, ya que con sólo dos capas totalmente conectadas se consiguieron precisiones de clasificación superiores al 90%. Por este motivo se decidió eliminar de la búsqueda el

número de capas totalmente conectadas durante el proceso evolutivo, lo cual se logró modificando el primer nivel del operador de mutación. En el primer nivel del operador de mutación, el cual se describe a detalle en la Sección 3.3 en [10], si $U_1(0,1) > 0.5$ se añade un nuevo bloque, y si $U_2(0,1) > 0.5$ el bloque añadido es una capa totalmente conectada. El operador se modificó de tal forma que si $U_2(0,1) > 0.5$ no se añade ningún bloque. Con esta modificación se pudo asegurar que durante todo el proceso evolutivo, las RNCs generadas sólo tendrían dos capas totalmente conectadas. Esto permitió que la búsqueda del algoritmo se centrara en los bloques de capas convolucionales, sobre las que recae el objetivo principal de generar vectores de características espacialmente cercanos, en términos de similitud coseno, si pertenecen a la misma clase y lejanos si son de clases distintas.

Originalmente la clase "CNN" del algoritmo de DeepGA, que se encarga de construir el modelo, a partir de las codificaciones, para el entrenamiento y las pruebas, solo generaba como salida las probabilidades de las imágenes de pertenecer a las distintas clases. Sin embargo, en este trabajo era prioritario acceder a los vectores de características generados por las RNCs para emplearlos en los procesos subsecuentes. Por este motivo se modificó la clase "CNN", añadiendo a su salida original, la salida aplanada del bloque convolucional (vectores de características).

La última modificación al algoritmo de DeepGA fue una mejora en la lectura de imágenes, se empleó la función ImageFolder de PyTorch para poder leer las imágenes de todas las clases en un único proceso en lugar de leer las imágenes de cada clase individualmente, como originalmente se diseñó.

4.3 RNC de dominio específico

El proceso de NE genera resultados altamente competitivos, sin embargo, tiene un alto costo computacional y de tiempo inherente. Por este motivo, en este trabajo solo se empleó una pequeña muestra de 6 imágenes de cada *clase conocida* para el proceso de NE y se decidió entrenar desde cero, empleando todo el conjunto de entrenamiento. La RNC que consiguió el mejor valor de aptitud con DeepGA y SupCon, será referenciada como RNC de dominio específico.

La sección de extracción de características de la RNC de dominio específico se separó de la sección de clasificación para cumplir dos propósitos:

1. Entrenar el bloque convolucional con la función de pérdida contrastiva y el bloque Fully-Connected con la función de pérdida Cross-Entropy; utilizando el conjunto de entrenamiento completo descrito en la Sección 4.1 (200 imágenes de cada *clase conocida*) y realizar una prueba de precisión de clasificación (con 50 imágenes de cada *clase conocida*) asumiendo momentáneamente un entorno de *conjunto cerrado* para validar que se podía obtener una buena precisión de clasificación de instancias de *clases conocidas,* ya que es un punto esencial para cualquier algoritmo de RCA.
2. Separar el proceso de extracción de características y el proceso de clasificación, ya que la detección de nuevos objetos de clase y el descubrimiento de sus clases deben producirse entre estos dos eventos.

4.4 Modelo de Mezclas Gaussianas (MMG) y MOCK/NSGA-II

El objetivo principal de este trabajo es abordar el RMMV como un problema de RCA con una extensión para el descubrimiento de *clases nuevas*. Para lograrlo, se dividió la estrategia en dos fases, ambas basadas en la distribución consistente de vectores de características en el espacio, generada por la RNC de dominio específico.

En la primera fase de la estrategia se realizó un modelado de las *clases conocidas* y se definió un umbral de confianza para *clases conocidas* con el MMG. El proceso se realizó extrayendo los vectores de características de las imágenes del conjunto de entrenamiento con la sección de extracción de características de la RNC de dominio específico. Los vectores de características obtenidos se comprimieron utilizando Principal Component Analysis (PCA), de este proceso se seleccionó el segundo y el tercer componente, que contribuían 27.81% y 20.97% respectivamente al porcentaje de varianza. Se decidió no utilizar el primer componente de PCA ya que al ser el común a todas las clases no se lograba la proyección esperada en el espacio. Los componentes se emplearon para realizar una regresión lineal sobre los vectores de características de las imágenes del conjunto de entrenamiento para obtener sus proyecciones en 2 dimensiones (2D). Como se mencionó en la Sección 4.1, con la misma proporción de datos con la que se entrenó y probó la RNC de dominio específico, se

utilizó el 80% de las proyecciones 2D de los vectores de características de las imágenes de entrenamiento para modelar cada *clase conocida* con un Modelo de Mezcla Gaussiana (MMG) y se empleó el 20% restante para definir un umbral de reconocimiento de *clases conocidas*.

Para modelar cada *clase conocida*, se calcularon los valores de media, desviación estándar, correlación y matriz de covarianza para cada clase del 80% de las proyecciones 2D de los vectores de características de las imágenes de entrenamiento. La función de densidad paramétrica empleada fue la función de densidad para una distribución gaussiana bivariada, expresada en la Eq. 2.18. Para la definición del umbral, se ingresó a cada modelo gaussiano el 20% restante de las proyecciones 2D de los vectores de características de las imágenes de entrenamiento, para calcular las probabilidades de cada proyección 2D de pertenecer a las *clases conocidas*. Dado que las proyecciones 2D eran de *clases conocidas*, se tomó como valor de umbral, el valor de probabilidad mínimo obtenido por las proyecciones 2D en su respectiva distribución gaussiana.

La segunda fase de la estrategia se realiza en un contexto de prueba donde la finalidad es reconocer las instancias de objetos desconocidos y simultáneamente descubrir sus clases empleando el MMG y el algoritmo de clustering MOCK/NSGA-II. Dado que la segunda fase de la estrategia representa la prueba para validar la hipótesis propuesta, se hizo una simulación completa del proceso donde las imágenes del conjunto de prueba que, como se mencionó en la Sección 4.1, incluye imágenes de *clases conocidas* y *desconocidas*, se ingresaron a la sección de extracción de características de la RNC de dominio específico para extraer sus vectores de características.

En este punto se realizaron dos procesos separados:

1. El algoritmo de clustering MOCK/NSGA-II agrupó los vectores de características de las imágenes de prueba extraídos por la RNC de dominio específico (sin modificar su dimensionalidad) y de este proceso se generaron soluciones competitivas, donde cada solución representaba una agrupación distinta de los vectores de características.
2. Se realizo la regresión lineal de los vectores de características de las imágenes de prueba extraídos por la RNC de dominio específico, utilizando los componentes extraídos de PCA mencionados anteriormente para obtener sus proyecciones 2D. Las

proyecciones se ingresaron a cada componente gaussiana del MMG para obtener las probabilidades de cada proyección de pertenecer a cada *clase conocida*. Las proyecciones que superaron el umbral establecido se agruparon en subconjuntos, donde cada conjunto representa una *clase conocida* y las proyecciones que no superaron el umbral se agruparon en un único conjunto que representa *clases desconocidas*.

De estos dos procesos se obtienen dos resultados, por una parte el algoritmo MOCK/NSGA-II genera individuos que representan distintas agrupaciones de los vectores de características y por otra parte el MMG genera una única solución que incluye un conjunto donde se encuentran los vectores de características de *clases desconocidas* y otro conjunto que está conformado por subconjuntos de vectores de características que representan *clases conocidas*. Con los dos resultados obtenidos por MOCK/NSGA-II y el MMG se realiza una serie de comparativos utilizando diferentes criterios para determinar las instancias que pertenecen a *clases conocidas*, las instancias que pertenecen a *clases desconocidas* y descubrir las *clases desconocidas*.

Dado que el presente MMG puede determinar con cierta confianza, cuales de las instancias ingresadas pertenecen a *clases conocidas* y sus respectivas clases, se realizó una comparación entre los subconjuntos de *clases conocidas* generados con el MMG y las soluciones generadas con MOCK/NSGA-II. Cada solución, que está conformada por agrupaciones distintas de los vectores de características, se compara con los subconjuntos de *clases conocidas* generados por el MMG y por cada estructura compartida (vectores agrupados en el mismo conjunto) la solución obtiene un punto. Aunque todas las soluciones de MOCK/NSGA-II son óptimas, se seleccionó como "*mejor solución*" la que obtuvo la puntuación más alta (mayor coincidencia con el MMG en las *clases conocidas*) y esté más cerca del punto rodilla (La solución en el frente de Pareto con la máxima utilidad marginal. Ilustrado en Figura 2.14).

Posteriormente, se determinaron cuáles conjuntos de la *mejor solución* contienen instancias de *clases conocidas* y se separaron de los que contienen instancias de *clases desconocidas* de forma similar a la puntuación de las soluciones. Se comparan los conjuntos de la *mejor solución* con los subconjuntos de *clases conocidas* generados con el MMG y las estructuras

compartidas automáticamente se determinan como *clases conocidas*. Por otra parte, dado que el MMG también detecta con cierta confianza los objetos de *clases desconocidas* (los objetos que no superaron el umbral), los conjuntos de la *mejor solución* que sólo contienen instancias de *clases desconocidas* del MMG se determinan automáticamente como *clases nuevas*. Si después de estos procesos, aún quedan conjuntos indeterminados como conocidos o desconocidos en la *mejor solución*, se cuenta el número de instancias *conocidas* y *desconocidas* (según la determinación del MMG) dentro de los conjuntos indeterminados y se les asigna la misma categoría que la mayoría de instancias que contengan. Además, se denotará como desconocidos si contienen el mismo número de instancias *desconocidas* y *conocidas* para intentar mitigar el *riesgo de conjunto abierto*.

Al final de esta estrategia se obtiene como resultado conjuntos determinados como *clases conocidas* y conjuntos determinados como *clases nuevas*. Cada conjunto determinado como *clase nueva* representa una clase descubierta, conformada por instancias de la misma clase, para la cual la RNC de dominio específico no fue entrenada y de la cual no tuvo información durante su entrenamiento pero que ahora es capaz de reconocer en pruebas subsecuentes. Los conjuntos de vectores de las *clases conocidas* son agrupados en un solo conjunto de vectores y son procesados en un evento separado que será detallado a continuación.

4.5 Clasificación de clases conocidas

Las RNC son la herramienta más utilizada en aplicaciones relacionadas con la clasificación de imágenes debido a su excepcional potencial de clasificación y extracción de características significativas. Por este motivo se decidió diseñar una RNC de dominio específico para el RMMV y ampliar su potencial agregando un mecanismo de detección de instancias de *clases desconocidas* y descubrimiento de *clases nuevas*. El mecanismo propuesto actúa entre la sección de extracción de características y clasificación; su finalidad es descubrir y clasificar instancias de *clases desconocidas*. Mientras que las instancias de *clases conocidas* son clasificadas con la sección de clasificación de la RNC de dominio específico, como es su trabajo habitual.

Como se mencionó al final de la Sección 4.4, el mecanismo propuesto para la detección de instancias de *clases desconocidas* y descubrimiento de *clases nuevas,* da como resultado

conjuntos determinados como *clases conocidas* y conjuntos determinados como *clases nuevas*. Donde los conjuntos de vectores de las *clases conocidas* se agrupan en un solo conjunto de vectores, el cual se ingresa a la sección de clasificación de la RNC de dominio específico para clasificar cada vector dentro de una determinada *clase conocida* como es el proceso habitual de una RNC.

4.6 Resumen del capítulo

En este capítulo se presentó la metodología detallada propuesta en esta tesis para abordar el RMMV como un problema de RCA con una extensión para el descubrimiento de *clases nuevas*. A lo largo de este capítulo se vio en detalle la selección de la base de datos y cómo se distribuyó en los distintos procesos, se especificaron los cambios realizados al algoritmo de NE para cumplir el propósito de obtener una RNC de dominio específico que generara vectores de características consistentes en el espacio, lo cual en gran medida se logró gracias al uso de la función de pérdida contrastiva. También en este capítulo se detalló el mecanismo propuesto para detectar instancias de *clases desconocidas* y el descubrimiento de sus clases, conformado por distintas técnicas de clustering como MMG y MOCK/NSGA-II.

Por último, este capítulo también aportó un esquema general de la metodología propuesta, lo que funciona como guía para un mejor entendimiento del proceso general que se propone en este trabajo.

Capítulo 5

Experimentos y resultados

En este capítulo se describen los experimentos y resultados obtenidos de la ejecución de la metodología propuesta para abordar el RMMV como un problema de RCA con una extensión para el descubrimiento de *clases nuevas*, descrita en el capítulo 4. En este capítulo también se agregan los detalles técnicos de las ejecuciones de los experimentos, incluyendo los parámetros que se utilizaron.

Por último se incluye la discusión de la aplicación de los objetivos generales y específicos y las pruebas para la validación de la hipótesis.

5.1 Detalles técnicos de la implementación

La implementación de la metodología propuesta se desarrolló y ejecutó casi en su totalidad usando Python 3.9.13 en el editor de código Visual Studio Code instalado en una MacBook Pro con procesador Intel Core i7 de cuatro núcleos a 2,2 GHz y 16 GB de memoria DDR3 a 1600 MHz. Como única excepción, la ejecución de PCA para obtener los componentes para generar las proyecciones de los vectores con regresiones lineales y la gráfica de la distribución de las *clases conocidas* del MMG se desarrollaron y ejecutaron en la plataforma de MATLAB instalada en la misma computadora.

5.1.1 Datos

En la Sección 4.1 se dio un análisis detallado de cómo se distribuyó y procesó la base de datos empleada en este trabajo (VMMRdb [6]). En esta sección se dará un resumen que incluye únicamente la cantidad de datos que se emplean en cada proceso:

- Neuroevolución: 6 imágenes de cada *clase conocida* (30 en total).

- Entrenamiento de la RNC de dominio específico: 200 imágenes de cada *clase conocida* (1,000 en total).
- Prueba de la RNC de dominio específico en entorno de *conjunto cerrado*: 50 imágenes de cada *clase conocida* (250 en total).
- Modelado de las *clases conocidas* con el MMG: 200 imágenes de cada *clase conocida* (1,000 en total).
- Definición del umbral para *clases conocidas* del MMG: 50 imágenes de cada *clase conocida* (250 en total).
- Prueba de la metodología completa para abordar el RMMV como un problema de RCA con descubrimiento de *clases nuevas*: 3 imágenes de cada *clase conocida* y 3 imágenes de cada *clase desconocida* (24 en total).

Como se hace mención en la Sección 4.1, todas las imágenes empleadas en los distintos pasos de este trabajo pasaron por diversos procesos antes de ser empleadas, se recortaron para mantener únicamente su ROI, se transformaron a escala de grises, se redimensionaron a 28x28 píxeles y se normalizaron con una media de 0.456 y una desviación estándar de 0.224. Un ejemplo de este preprocesamiento se muestra en la Figura 5.1 donde a la izquierda se muestran las imágenes originales y a la derecha las imágenes resultantes posteriores al procesamiento.

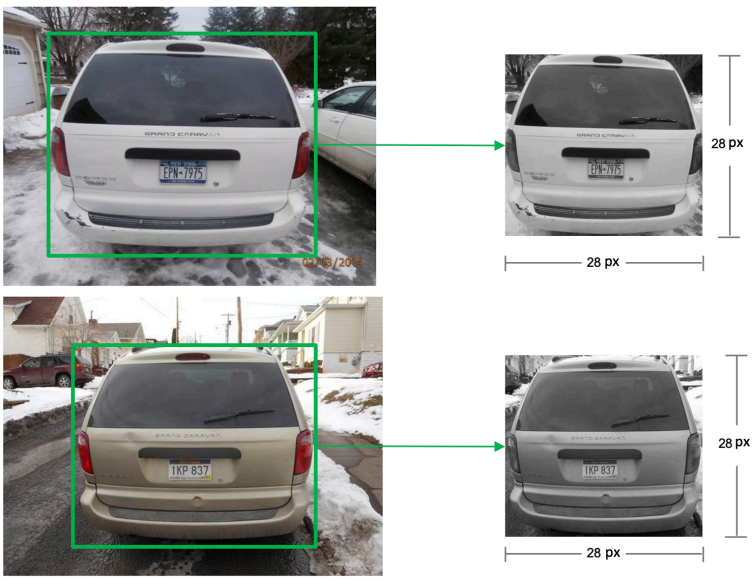

Figura 5.1. Preprocesamiento de las imágenes empleadas en el presente trabajo tomadas de la base de datos VMMRdb [6].

5.1.2 Parámetros

5.1.2.1 Parámetros DeepGA

Para el proceso de Neuroevolución con DeepGA [10] y SupCon [14] se utilizaron los parámetros descritos en la Tabla 5.1 para inicializar la población. Los parámetros empleados durante la ejecución del algoritmo de NE se calibraron manualmente ya que por restricciones de tiempo, no fue posible utilizar un programa de calibración de parámetros. Los parámetros calibrados manualmente con los que se obtuvieron los mejores resultados durante el proceso evolutivo para generar las RNCs se muestran en la Tabla 5.2. Finalmente, los diferentes valores que podía tener cada hiperparámetro durante el proceso evolutivo fueron los mismos que estableció el autor de DeepGA [10] y se presentan en la Tabla 5.3.

Tabla 5.1. Parámetros para la inicialización de la población.

Parámetros	Valores
Número Mínimo de Capas Convolucionales	4
Número Máximo de Capas Convolucionales	9
Número Mínimo de Capas Totalmente Conectadas	1
Número Máximo de Capas Totalmente Conectadas	1

Tabla 5.2. Parámetros del proceso evolutivo.

Parámetros	Valores
Tamaño de Población	20
Numero de Generaciones	100
Tamaño de Torneo	5
Porcentaje de Cruza	0.7
Porcentaje de Mutación	0.7
Épocas de entrenamiento por individuo	20
Tasa de aprendizaje (Optimizador Adam)	1×10^{-4}
Temperatura (Función de pérdida SupCon)	0.07
Tamaño de lote	15

Tabla 5.3. Valores que podría tener cada hiperparámetro durante el proceso evolutivo.

Hiperparámetros	Valores
Número de Filtros*	{2, 4, 8, 16, 32}
Tamaño de Filtros*	{2, 3, 4, 5, 6, 7, 8}
Tipo de Pooling*	{Max, Avg}
Tamaño de Pooling*	{2, 3, 4, 5}
Número de Neuronas	{4, 8, 16, 32, 64, 128}

[1] Las filas marcadas con * corresponden al bloque convolucional, mientras que la fila no marcada corresponde al bloque totalmente conectado.

5.1.2.2 Parámetros entrenamiento RNC de dominio específico

Para el entrenamiento de la RNC de dominio específico se emplearon los parámetros que se muestran en la Tabla 5.4.

Tabla 5.4. Parámetros empleados en el entrenamiento de la RNC de domino específico.

Parámetros	Valores
Épocas de entrenamiento	20
Tasa de aprendizaje (Optimizador Adam)	1×10^{-3}
Temperatura (Función de pérdida SupCon)	0.07
Tamaño de lote	15

5.1.2.3 Parámetros MOCK/NSGA-II

Para la ejecución del algoritmo de clustering MOCK/NSGA-II se emplearon los parámetros que se muestran en la Tabla 5.5.

Tabla 5.5. Parámetros empleados en el algoritmo de clustering MOCK/NSGA-II.

Parámetros	Valores
Tamaño de Población (M)	9
Vecinos Cercanos (L)	2
Número de Generaciones	10

5.2 Experimentación y Resultados

Los experimentos descritos en este capítulo tienen como finalidad obtener los resultados suficientes para la validación de la hipótesis de esta tesis, la cual se enuncia nuevamente:

> **Hipótesis**: Aplicar un enfoque de RCA con una extensión para el descubrimiento de clases nuevas a la problemática del RMMV generará un algoritmo más robusto al disminuir el riesgo de clasificar imágenes de clases desconocidas dentro de alguna clase conocida y, también, aumentará su capacidad de reconocimiento y aprendizaje después de ser entrenado al ser capaz de reconocer instancias que no pertenezcan a las *clases conocidas* y simultáneamente descubrir sus clases.

La primera parte de la metodología propuesta en esta tesis, la cual se describió en la Sección 4, tenía como finalidad la ejecución del algoritmo modificado DeepGA [10] y SupCon [14] para el proceso de NE de RNCs. Con los datos y parámetros descritos en la Secciones 4.1 y 5.1.2.1, se realizaron siete ejecuciones del proceso de NE. La Figura 5.2 muestra las curvas de convergencia de las siete ejecuciones y la Tabla 5.6 muestra un breve análisis estadístico de los valores de aptitud obtenidos en cada ejecución. Como puede observarse, todas las ejecuciones comenzaron con un valor de aptitud dentro de un rango de 3.3 y 3.5 y la ejecución que logró la mejor búsqueda fue la numero 7, la cual obtuvo un valor final de aptitud de 1.96096, tardó 7 horas en ejecutarse y se encuentra marcada en rojo en la Figura 5.2.

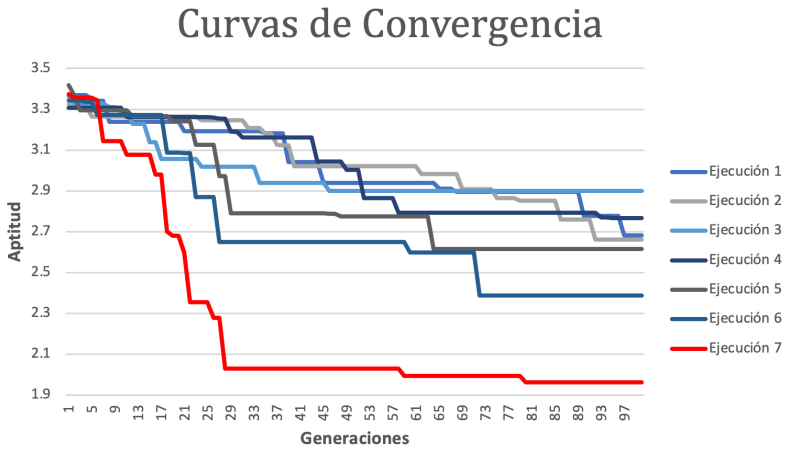

Figura 5.2. Curvas de convergencia de las siete ejecuciones del proceso de NE.

Tabla 5.6. Análisis estadístico de las siete ejecuciones del proceso de NE.

Aptitudes	Ejecución 1	2.6816294
	Ejecución 2	2.6608574
	Ejecución 3	2.9001975
	Ejecución 4	2.765866
	Ejecución 5	2.6154811
	Ejecución 6	2.3863246
	Ejecución 7	1.9609622
	Mejor	**1.96096**
	Peor	**2.90020**
	Promedio	**2.56733**
	Desviación Estándar	**0.30951**

Dado que la finalidad de utilizar DeepGA [10] y SupCon [14] era diseñar una RNC que generara vectores de características espacialmente cercanos en términos de similitud coseno si pertenecían a la misma clase y lejanos si pertenecían a clases distintas, se realizó una prueba para validar la consistencia en los vectores de características de las RNC diseñadas en el proceso de NE. La prueba consistió en generar Matrices de Distancias, utilizando similitud coseno como métrica, de los vectores de características obtenidos en la última época de entrenamiento de las RNC con mejor aptitud en cada ejecución del algoritmo de NE. Sobre los mismos vectores de características, se utilizó la técnica t-SNE [15] para reducir la dimensionalidad de los vectores a 2 dimensiones, con el fin de visualizarlos en un plano bidimensional. Los resultados de esta prueba se muestran en la Figura 5.3 donde, mediante estas dos técnicas, se puede observar que la ejecución 7, que obtuvo el mejor valor de aptitud en el proceso de NE, logró el comportamiento deseado en los vectores de características, mapeando las instancias de la misma clase en un espacio cercano y alejando las instancias de las diferentes clases dentro del espacio.

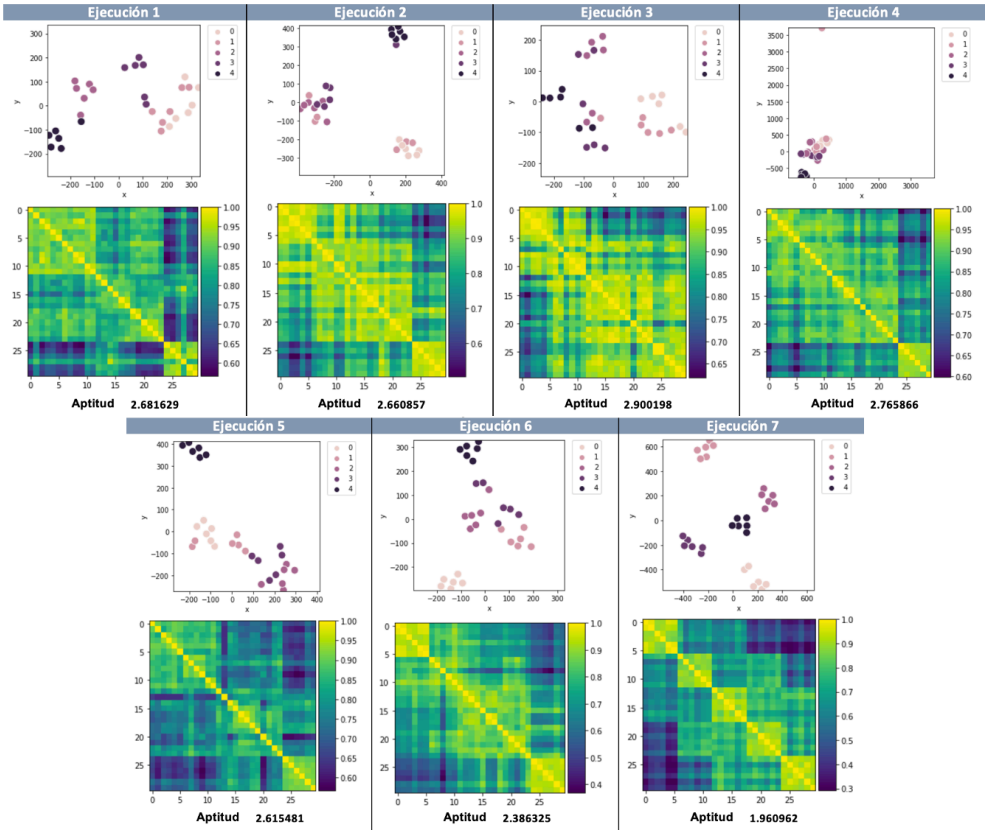

Figura 5.3. Matrices de distancia (con similitud coseno) y proyecciones en un plano bidimensional de los vectores de características de la última época de entrenamiento de las RNC con mejor aptitud en cada ejecución del algoritmo de NE.

La RNC con la mejor aptitud obtenida con el proceso de NE, que como se mencionó previamente se denominará RNC de dominio específico, se muestra en la Figura 5.4 en términos de su codificación. En el primer nivel, se puede observar que la arquitectura tiene 13 bloques convolucionales (cada uno formado por una única capa convolucional) y tiene 2 bloques Totalmente Conectados (el primero representa la capa de entrada de la Fully Connected conformada por 128 neuronas y el segundo la capa de salida con 5 neuronas para igualar el tamaño el vector de salida al número de clases). En el segundo nivel, se muestra la cadena binaria que define la conectividad entre bloques convolucionales, donde cada bit representa la conexión con una capa anterior no consecutiva, a partir del tercer bloque. En la Figura 5.4, la cadena binaria se segmentó en distintos colores para lograr un mejor entendimiento de su representación en el primer nivel. Dado que la mayoría de los siguientes experimentos se basan en los vectores de características y su distribución en el espacio, es importante mencionar que el último bloque/capa convolucional de la RNC de dominio específico genera vectores de 288 características.

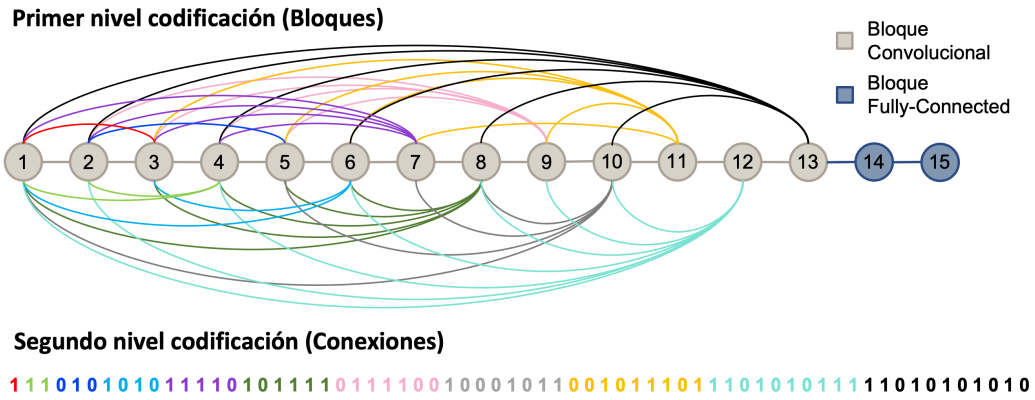

Figura 5.4. Codificación de la RNC de dominio específico. El primer nivel representa las capas convolucionales y las capas de la Fully Connected. El segundo nivel (cadena binaria) determina las conexiones entre las capas convolucionales recibidas a partir del tercer bloque.

El siguiente paso en la experimentación fue la separación de los bloques convolucionales (sección de extracción de características) de los bloques Totalmente Conectados (sección de clasificación) de la RNC de dominio específico, para entrenarlos en procesos separados con las funciones de pérdida contrastiva y Cross-Entropy, respectivamente, empleando los datos descritos en la Sección 4.1. Dado que obtener una buena precisión de clasificación de instancias de *clases conocidas* es un punto esencial para cualquier algoritmo de RCA, se realizó una prueba de clasificación asumiendo momentáneamente un entorno de *conjunto cerrado* donde el 80% del conjunto de entrenamiento fue empleado para entrenar el modelo y el 20% restante para probarlo (En la sección 4.1 se da un mayor detalle de los datos empleados). El resultado de esta prueba fue una precisión de clasificación del 97.6%. Para lograr una mejor visualización y resumen de los resultados se calculó la matriz de confusión la cual se muestra en la Figura 5.5.

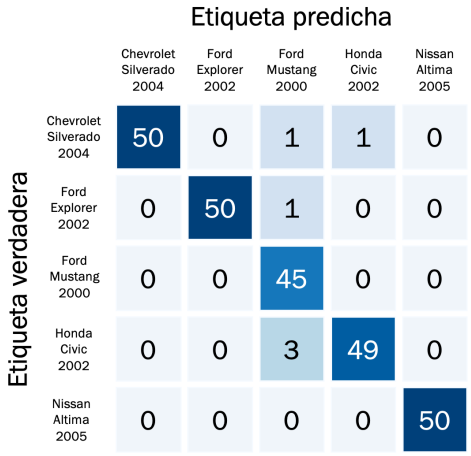

Figura 5.5. Matriz de Confusión de los resultados de la prueba de clasificación asumiendo un entorno de conjunto cerrado.

Otro punto crucial de la metodología propuesta en esta tesis es que la RNC de dominio específico sea capaz de generar vectores de características espacialmente cercanos, en términos de similitud coseno, si pertenecen a la misma clase y lejanos si pertenecen a clases distintas, manteniendo este comportamiento en objetos de *clases desconocidas,* ya que el mecanismo propuesto en esta tesis para la detección de instancias de *clases desconocidas* y el descubrimiento de *clases nuevas* depende del comportamiento consistente en los vectores de características. Con los resultados mostrados en la Figura 5.3 se validó que el comportamiento era consistente en instancias de *clases conocidas*, sin embargo para validar que el comportamiento se mantendría cuando se ingresaran instancias de *clases conocidas* y *desconocidas*, se realizó un experimento que consistió en ingresar las imágenes del conjunto de prueba que, como se describió en la Sección 4.1, incluye imágenes de *clases conocidas* y *desconocidas* al bloque convolucional de la RNC de dominio específico para extraer sus vectores de características y poder visualizar su distribución en el espacio. Para visualizar la distribución de los vectores de características de las imágenes de prueba, se generó una Matriz de Distancias con los vectores originales de 288 características, utilizando similitud coseno como métrica. Para poder visualizar los vectores de características generados por la RNC de dominio específico, en un plano bidimensional se realizó una proyección bidimensional mediante una regresión lineal utilizando los 2 componentes obtenidos con PCA descritos en la Sección 4.4. Los resultados de esta prueba se muestran en la Figura 5.6, donde se puede observar que la RNC de dominio específico es capaz de generar vectores de características cercanos en términos de similitud coseno si pertenecen a la misma clase y lejanos si pertenecen a clases distintas y consiguió mantener este comportamiento incluso en instancias de *clases desconocidas*.

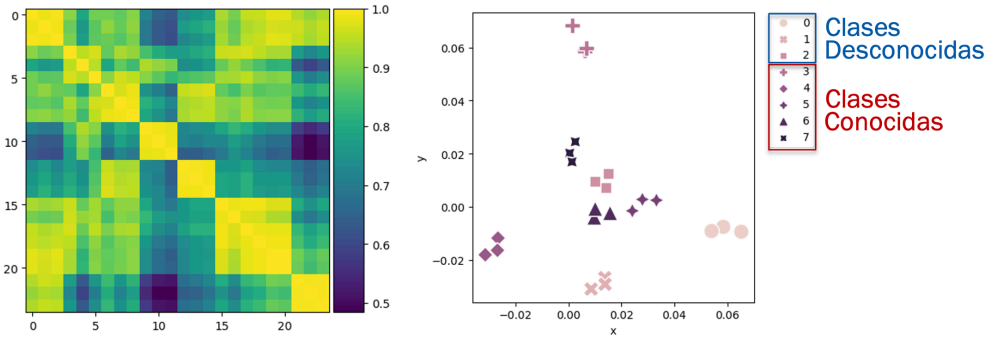

Figura 5.6. Matriz de distancia (con similitud de coseno) de los vectores originales de 288 características y proyección en plano bidimensional de los vectores de características de las imágenes del conjunto de prueba.

De acuerdo con la metodología propuesta, el siguiente paso de la experimentación fue la aplicación de la primera fase de la estrategia para implementar el mecanismo de detección de instancias de *clases desconocidas* y el descubrimiento de *clases nuevas*. Para la implementación de la primera fase, se ingresaron las imágenes del conjunto de entrenamiento (solo incluye imágenes de las 5 *clases conocidas*) al bloque convolucional de la RNC de dominio específico para extraer sus vectores de características. Los vectores de características se comprimieron utilizando PCA donde se seleccionó el segundo y el tercer componente, que contribuían 27.81% y 20.97%, respectivamente, al porcentaje de varianza para posteriormente realizar una regresión lineal sobre los vectores de características de las imágenes del conjunto de entrenamiento para obtener sus proyecciones en 2D.

El siguiente paso fue calcular los valores de media, desviación estándar, correlación y matriz de covarianza para cada clase del 80% de las proyecciones 2D del conjunto de entrenamiento, los cuales se pueden apreciar en la Tabla 5.7. Los valores calculados de cada clase se ingresaron en la función de densidad para una distribución gaussiana bivariada, expresada en la Eq. 2.18. y se graficaron, utilizando [-0.1,0.1] como intervalos para x y *y*, para visualizar la distribución del MMG, donde cada gaussiana representa la distribución de cada *clase conocida*. La grafica del MMG se muestra en la Figura 5.7.

Tabla 5.7. Medidas de dispersión y matriz de covarianza para cada clase del 80% de las proyecciones 2D del conjunto de entrenamiento.

Clases conocidas	Media		Desviación estándar		Correlación	Matriz de covarianza
	μ_1	μ_2	σ_1	σ_2	ρ	Σ
Clase 1	0.0534208	-0.00666438	0.00998274	0.00603652	-0.000038777	$\begin{bmatrix} 1.00055228e^{-04} & -3.87773350e^{-05} \\ -3.87773350e^{-05} & 3.65859732e^{-05} \end{bmatrix}$
Clase 2	0.01324696	-0.02696672	0.00456533	0.00741663	-0.000007198	$\begin{bmatrix} 2.09259000e^{-05} & -7.19759802e^{-06} \\ -7.19759802e^{-06} & 5.52272824e^{-05} \end{bmatrix}$
Clase 3	0.01474776	0.00918005	0.00277614	0.00295129	-0.000000954	$\begin{bmatrix} 7.73789938e^{-06} & -9.53777804e^{-07} \\ -9.53777804e^{-07} & 8.74507640e^{-06} \end{bmatrix}$
Clase 4	0.00691647	0.053336	0.00262568	0.00808586	-0.000002653	$\begin{bmatrix} 6.92188309e^{-06} & -2.65337234e^{-06} \\ -2.65337234e^{-06} & 6.56436787e^{-05} \end{bmatrix}$
Clase 5	-0.02324388	-0.01037511	0.00550063	0.00514579	0.000017330	$\begin{bmatrix} 3.03784544e^{-05} & 1.73296973e^{-05} \\ 1.73296973e^{-05} & 2.65854691e^{-05} \end{bmatrix}$

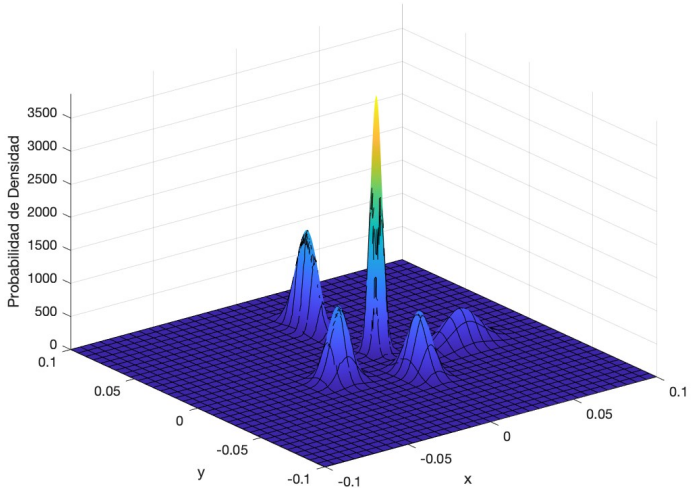

Figura 5.7. Distribución de las clases conocidas mediante el MMG.

Posteriormente se realizó el proceso para el cálculo del umbral de *clases conocidas,* el cual consistió en ingresar a cada modelo gaussiano las proyecciones 2D del 20% restante de los vectores de características de las imágenes de entrenamiento, para obtener las probabilidades de cada proyección 2D de pertenecer a las *clases conocidas*. Dado que se conocía la clase verdadera de cada proyección 2D, se compararon las probabilidades que cada proyección 2D obtuvo en el modelo gaussiano de su clase verdadera para obtener el valor mínimo de probabilidad. El resultado de este proceso fue un valor de umbral de **0.9999** al ser la probabilidad mínima obtenida por las proyecciones en sus respectivos modelos gaussianos.

El siguiente paso de la experimentación fue la aplicación de la segunda fase de la estrategia para implementar el mecanismo de detección de instancias de *clases desconocidas* y el descubrimiento de *clases nuevas*. Esta segunda fase de la estrategia representa la prueba final para la validación de la hipótesis, ya que cubre la detección de objetos de *clases desconocidas* y el descubrimiento de *clases nuevas*, por lo cual se pone a prueba el algoritmo completo y se involucran todas las partes del proceso.

Siguiendo el proceso habitual de clasificación con una RNC, se ingresan las imágenes del conjunto de prueba, que incluyen instancias de *clases conocidas* y *desconocidas*, al bloque convolucional de la RNC de dominio específico para extraer sus vectores de características. En este punto el mecanismo propuesto para la detección de instancias de *clases desconocidas* y el descubrimiento de *clases nuevas* toma acción y se ejecutan dos procesos separados:

1. Se ejecuta el algoritmo de clustering MOCK/NSGA-II con los parámetros descritos en la Tabla 5.5. para agrupar los 24 vectores de características de las imágenes de prueba, extraídos por la RNC de dominio específico, los cuales, como se mencionó previamente, tienen una dimensionalidad de 288 características. El resultado de este proceso son 9 individuos (soluciones óptimas), donde cada solución representa una agrupación distinta de los vectores de características. Las 9 soluciones se muestran en la Figura 5.8 en términos de sus valores de aptitud y en la Figura 5.9 se muestra cómo se agruparon los vectores en diferentes estructuras

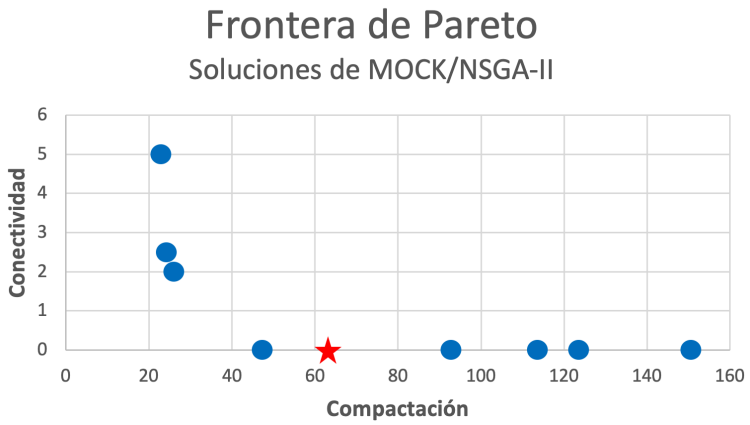

Figura 5.8. El Frente de Pareto de las soluciones generadas por el algoritmo de clustering MOCK mejorado con NSGA-II. El individuo marcado con una estrella roja es el punto de rodilla.

Solución 1 `[[0, 2, 1], [10, 9, 11], [12, 13, 14], [16], [17, 15], [18, 20], [19], [21, 22, 23], [3, 4, 5], [6, 8, 7]]`
Solución 2 `[[0, 2, 1], [10, 9, 11], [12, 13, 14], [16, 15], [17], [18, 20], [19], [21, 22, 23], [3, 4, 5], [8, 6, 7]]`
Solución 3 `[[0, 2, 1], [10, 9, 11], [12, 13, 14], [16, 15], [17], [18, 19, 20], [21, 22, 23], [3, 4, 5], [7, 8, 6]]`
Solución 4 `[[0, 2, 1], [10, 9, 11], [12], [13, 14], [16, 17, 15], [18, 19, 20], [21, 22, 23], [3, 4, 5], [8, 6, 7]]`
Solución 5 `[[0, 1, 2], [10, 9, 11], [12, 13, 14], [16, 17, 15], [18, 19, 20], [21, 22, 23], [3, 4, 5], [8, 6, 7]]`
Solución 6 `[[0, 2, 1], [10, 23, 21, 9, 11, 22], [12, 13, 14], [16, 17, 15, 19, 18, 3, 20, 4, 5], [8, 6, 7]]`
Solución 7 `[[0, 2, 1, 17, 16, 15], [10, 23, 21, 9, 11, 22, 12, 13, 14], [18, 19, 20], [3, 4, 5], [7, 8, 6]]`
Solución 8 `[[0, 2, 1, 17, 16, 19, 15, 18, 3, 20, 4, 5, 8, 6, 7, 21, 22, 23, 10, 9, 11], [12, 13, 14]]`
Solución 9 `[[0, 2, 1, 17, 16, 19, 15, 18, 3, 20, 4, 5, 8, 6, 7, 21, 22, 23, 12, 10, 13, 9, 11, 14]]`

Figura 5.9. Los 9 individuos (soluciones optimas) finales generados por el algoritmo de clustering MOCK mejorado con NSGA-II. El individuo marcado en rojo es el punto de rodilla.

2. Se realiza la regresión lineal de los vectores de características de las imágenes de prueba extraídos por la RNC de dominio específico, utilizando los componentes calculados con PCA mencionados anteriormente para obtener sus proyecciones 2D. Las proyecciones 2D se ingresan a cada modelo gaussiano del MMG para obtener las probabilidades de cada proyección de pertenecer a cada *clase conocida*, las cuales se

muestran en la Tabla 5.8, donde también se resaltan en gris aquellas probabilidades que superaron el umbral de *clases conocidas* (**0.9999**). Como se puede observar, los objetos de *clases conocidas* fueron correctamente identificados dentro de sus clases y en el caso de los objetos de *clases desconocidas*, se puede observar que con el límite marcado por el umbral, 8 de las 9 instancias fueron correctamente identificados como desconocidas. Dadas las probabilidades y el umbral de *clases conocidas,* se agrupan las proyecciones 2D que no superaron el umbral en un único conjunto que representa *clases desconocidas* y las proyecciones 2D que superaron el umbral establecido se agrupan en subconjuntos, donde cada conjunto representa una *clase conocida*. Esta agrupación se muestra en la Figura 5.10.

Tabla 5.8. Probabilidades de las instancias de prueba obtenidas con el MMG. En gris están marcadas las probabilidades que superaron el umbral (**0.9999**).

	x	y	Etiqueta Real		Prob Clase 0	Prob Clase 1	Prob Clase 2	Prob Clase 3	Prob Clase 4
0	5.84E-02	-7.59E-03	0		1.000E+00	8.152E-27	7.294E-57	1.650E-89	1.483E-73
1	5.40E-02	-9.16E-03	0		1.000E+00	4.920E-22	6.332E-48	8.417E-77	1.826E-67
2	6.52E-02	-9.41E-03	0		1.000E+00	1.412E-33	3.304E-75	1.581E-112	1.321E-88
3	1.37E-02	-2.67E-02	1		4.068E-17	1.000E+00	6.451E-33	3.341E-22	8.823E-29
4	1.38E-02	-2.92E-02	1		6.396E-19	1.000E+00	1.408E-37	1.556E-23	2.039E-31
5	8.56E-03	-3.10E-02	1		1.170E-22	1.000E+00	3.308E-43	6.810E-24	3.034E-27
6	1.43E-02	7.13E-03	2	Conocidas	1.020E-04	3.865E-06	9.999E-01	4.394E-09	2.252E-11
7	1.52E-02	1.24E-02	2		1.451E-04	9.982E-08	9.999E-01	7.854E-08	1.672E-11
8	1.02E-02	9.51E-03	2		5.701E-05	4.896E-06	9.999E-01	5.240E-07	1.290E-08
9	1.53E-03	6.82E-02	3		7.202E-36	3.243E-35	5.417E-87	1.000E+00	4.510E-58
10	6.30E-03	5.86E-02	3		3.311E-28	6.318E-30	2.282E-61	1.000E+00	1.830E-42
11	6.97E-03	5.96E-02	3		2.972E-29	1.191E-30	1.055E-63	1.000E+00	1.461E-43
12	-2.66E-02	-1.18E-02	4		2.727E-28	2.310E-17	1.384E-65	1.521E-56	1.000E+00
13	-2.68E-02	-1.64E-02	4		8.075E-32	2.380E-17	1.771E-72	2.738E-59	1.000E+00
14	-3.14E-02	-1.81E-02	4		3.329E-36	3.051E-21	3.378E-87	2.106E-71	1.000E+00
15	2.79E-02	2.76E-03	5		9.995E-01	2.116E-06	5.257E-04	6.300E-19	3.457E-21
16	3.32E-02	2.46E-03	5		1.000E+00	2.820E-09	3.713E-09	7.638E-27	2.355E-27
17	2.41E-02	-1.54E-03	5		9.779E-01	4.606E-03	1.751E-02	2.796E-15	1.285E-18
18	9.74E-03	-4.05E-03	6	Desconocidas	4.675E-05	9.986E-01	1.350E-03	2.120E-09	2.322E-08
19	1.57E-02	-2.33E-03	6		1.208E-02	4.648E-01	5.231E-01	9.067E-10	1.078E-11
20	1.01E-02	-7.46E-04	6		9.939E-04	5.405E-01	4.585E-01	7.914E-08	2.899E-07
21	4.78E-04	2.02E-02	7		5.100E-02	1.262E-04	1.164E-02	9.322E-01	5.056E-03
22	1.33E-03	1.69E-02	7		1.022E-01	1.459E-03	6.103E-01	2.326E-01	5.334E-02
23	2.57E-03	2.45E-02	7		2.779E-04	8.184E-08	1.406E-05	9.997E-01	4.287E-07

Conocidas 5 Grupos

[0,1,2,16] [3,4,5] [6,7,8] [9,10,11] [12,13,14]

Desconocidas 1 Grupo

[15,17,18,19,20,21,22,23]

Figura 5.10. Instancias agrupadas según las probabilidades del MMG y el umbral de clases conocidas.

De estos dos procesos se obtuvieron dos resultados, el primero son las 9 soluciones generadas por MOCK/NSGA-II que se muestra en la Figura 5.9 y el segundo son las agrupaciones de *clases conocidas* y *desconocidas* generadas con las probabilidades del MMG y el umbral de *clases conocidas* que se muestra en la Figura 5.10. Con estos resultados se comienzan las comparativas descritas en la Sección 4.4 para determinar las instancias que pertenecen a *clases conocidas*, las instancias que pertenecen a *clases desconocidas* y descubrir las *clases desconocidas*.

La primera comparación es entre las 9 soluciones generadas con MOCK/NSGA-II y los subconjuntos (grupos) de *clases conocidas* generados con el MMG y el umbral de *clases conocidas*. En esta comparativa se le da una puntuación a cada solución con base en las estructuras compartidas (vectores agrupados en el mismo conjunto) y se selecciona como "*mejor solución*" a aquella solución que obtenga la puntuación más alta (mayor coincidencia con el MMG en las *clases conocidas*) y esté más cerca del punto de rodilla. Este proceso se muestra en la Tabla 5.9 donde se puede apreciar que las soluciones 1, 2, 3 y 5 tienen cuatro estructuras compartidas. Sin embargo, dado que la solución 5 (marcada con *) es la más cercana al punto rodilla, se selecciona como la *mejor solución* (en este caso la *mejor solución* resultó ser el punto de rodilla marcado en rojo en las Figuras 5.8 y 5.9).

Tabla 5.9. Posiciones en el Frente de Pareto y Puntuaciones obtenidas de la comparación de las soluciones generadas por MOCK/NSGA-II y los subgrupos de *clases conocidas* generados por MMG y el umbral correspondiente.

	Posición en el Frente de Pareto	**Puntuación**
Solución 1	1	4
Solución 2	2	4
Solución 3	3	4
Solución 4	4	3
Solución 5*	**5**	**4**

Solución 6	6	2
Solución 7	7	2
Solución 8	8	1
Solución 9	9	0

[1] La solución marcada con * se seleccionó como *mejor solución* dada su puntuación y posición en el Frente de Pareto.

El siguiente paso es la determinación de conjuntos de *clases conocidas* y conjuntos de *clases nuevas* dentro de la *mejor solución*. Para la determinación de *clases conocidas* se comparan los conjuntos de la *mejor solución* y los subconjuntos de *clases conocidas* generados con el MMG y se determinan como *clases conocidas* las estructuras compartidas (vectores agrupados en el mismo conjunto). Para la determinación de *clases nuevas*, los conjuntos restantes de la *mejor solución* que sólo contienen instancias del conjunto de *clases desconocidas* del MMG se determinan automáticamente como *clases nuevas*. Este proceso se puede apreciar en la Figura 5.11.

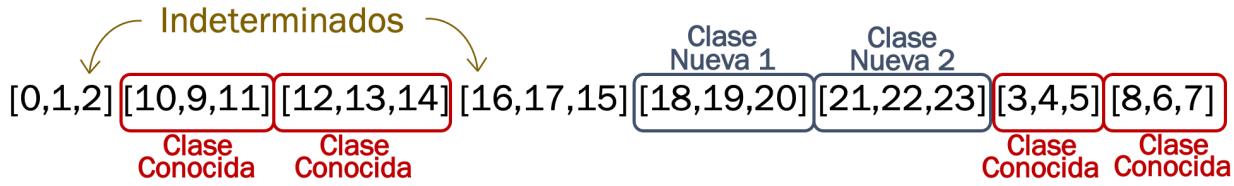

Figura 5.11. Selección de clases conocidas, clases nuevas y conjuntos indeterminados dentro de la mejor solución.

Dado que aún existían conjuntos indeterminados como conocidos o desconocidos dentro de la *mejor solución*, se hizo el conteo de instancias dentro de los conjuntos indeterminados para realizar la determinación. El conjunto indeterminado [0,1,2] se estableció como *clase conocida,* dado que las tres instancias que contenía fueron clasificadas dentro de una *clase conocida* por el MMG. El conjunto indeterminado [16,17,15] se determinó como *clase nueva,* dado que dos de las tres instancias fueron clasificadas dentro la *clase desconocida* por el MMG. El resultado final de este proceso se muestra en la Figura 5.12 donde se puede apreciar la determinación final de todos los conjuntos de la *mejor solución* en cinco grupos de *clases conocidas* y tres de *clases nuevas*.

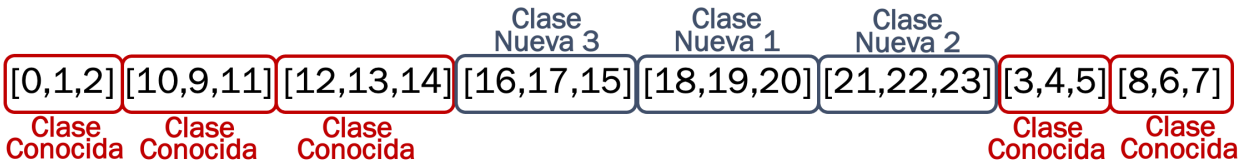

Figura 5.12. Selección final de clases conocidas y clases nuevas dentro de la mejor solución.

Observando los datos de la Tabla 5.8 donde se muestran las etiquetas reales, se pudo verificar que los vectores de las *clases nuevas* corresponden a instancias de *clases desconocidas* y que se agruparon en la misma estructura que su *clase desconocida*, confirmando que efectivamente se pueden descubrir *clases nuevas* de objetos de *clases desconocidas* con la metodología propuesta, lo que a su vez valida la hipótesis enunciada en esta tesis, ya que al aplicar un enfoque de RCA con una extensión para el descubrimiento de *clases nuevas* a la problemática del RMMV se generó un algoritmo más robusto que disminuyó el riesgo de clasificar imágenes de *clases desconocidas* dentro de alguna *clase conocida* y también aumentó su capacidad de reconocimiento y aprendizaje después de ser entrenado, ya que fue capaz de reconocer instancias que no pertenecían a las *clases conocidas* y simultáneamente descubrió sus clases.

Finalmente, para concluir el proceso general propuesto en esta tesis, los 15 vectores de características de las clases que fueron determinadas como *conocidas* se introdujeron en la sección de clasificación de la RNC de dominio específico con su dimensionalidad original (288 características), donde se obtuvo una precisión de clasificación del 100%. Dados los resultados de clasificación obtenidos, se calcularon los valores de Verdadero Positivo (VP), Falso Positivo (FP) y Falso Negativo (FN) tanto de las *clases conocidas* como de las *desconocidas* y se agregó el cálculo de la Puntuación micro-F1 (Eq. 5.1), ya que es una de las métricas más utilizadas en los algoritmos de RCA. Los resultados obtenidos se muestran en la Tabla 5.10.

$$micro\text{-}F1 = \frac{VP}{VP+\frac{1}{2}(FP+FN)} = 2\frac{PRE*REC}{PRE+REC} \tag{5.1}$$

donde VP son los Verdaderos Positivos, FP son los Falsos Positivos, FN son los Falsos Negativos, PRE es la precisión y REC es el recall.

Tabla 5.10. Cálculo de la puntuación micro-F1 para la clasificación de objetos de *clases conocidas* y *desconocidas* con la metodología propuesta.

	Etiqueta	Verdadero Positivo (VP)	Falso Positivo (FP)	Falso Negativo (FN)	micro-F1
Clases conocidas	Chevrolet Silverado 2004	3	0	0	Precisión = 1.0 Recall = 1.0 micro-F1 = 1.0
	Ford Explorer 2002	3	0	0	
	Ford Mustang 2000	3	0	0	
	Honda Civic 2002	3	0	0	
	Nissan Altima 2005	3	0	0	
Clases desconocidas	Clase Nueva 1	3	0	0	
	Clase Nueva 2	3	0	0	
	Clase Nueva 3	3	0	0	
	Total	24	0	0	

5.3 Discusión de resultados

En esta Sección se discute cómo se cubrieron los objetivos generales y específicos dentro de la experimentación realizada y cómo los resultados obtenidos en las diversas pruebas llevan a la validación de la hipótesis.

Para el desarrollo de este trabajo, se consideraron 7 objetivos específicos para cumplir con el propósito de abordar el RMMV como un problema de RCA con una extensión para el descubrimiento de *clases nuevas,* lo que llevaría a la validación de la hipótesis enunciada en esta tesis. La implementación del primer objetivo se detalló en la Sección 4.1 donde se describe el preprocesamiento al que se sometieron las imágenes de 8 clases de la base de datos VMMRdb [6], con las cuales se llevó a cabo la experimentación de la metodología propuesta.

El cumplimiento del segundo objetivo se pudo validar con los resultados mostrados en las Figuras 5.3 y 5.6, donde se puede observar que la RNC de dominio específico diseñada con el proceso NE y la función de pérdida contrastiva, logró mapear en un espacio cercano, en términos de similitud coseno, los vectores de características que pertenecían a las mismas

clases y alejar dentro del espacio las distintas clases, manteniendo este comportamiento tanto para *clases conocidas* como *desconocidas*.

Los objetivos específicos 3, 4 y 5 se encuentran estrechamente relacionados, ya que el tercer objetivo, cuya implementación se describió detalladamente en la Sección 4.4, es la parte teórica del quinto objetivo. En la Sección de experimentación y resultados se detallaron paso a paso los experimentos realizados para poner a prueba la metodología propuesta en la Sección 4.4, la cual actúa entre las secciones de extracción de características y clasificación de la RNC de dominio específico (Objetivo 4). Con los resultados obtenidos se validó que el mecanismo propuesto es capaz de detectar objetos de *clases desconocidas* y simultáneamente descubrir sus clases. Un punto por destacar es que la estrategia propuesta no está restringida por los datos de entrenamiento y se puede ajustar a medida que los datos cambian. Más concretamente, el utilizar el aprendizaje contrastivo para entrenar la sección de extracción de características de la RNC de dominio específico, generó que el mapeo de los vectores de características no sólo este guiado por las características aprendidas de las *clases conocidas*, sino que es capaz de generalizar su distribución consistente dentro del espacio para instancias de *clases desconocidas*, lo que permite el funcionamiento eficaz del mecanismo de detección de instancias de *clases desconocidas* y descubrimiento de *clases nuevas*.

Desde el principio, se decidió emplear una RNC no sólo para explotar su potente capacidad de extraer características significativas, sino que estas redes son *conocidas* por ser potentes clasificadores y dado que la RNC de dominio específico se entrenó con numerosos ejemplos bien etiquetados, se pudo confiar en su precisión para clasificar instancias de *clases conocidas*. Por tanto, el sexto objetivo se cumplió al lograr una precisión de clasificación del 100% de las imágenes de *clases conocidas* del conjunto de prueba.

El último objetivo se concretó con las diversas pruebas estadísticas llevadas a cabo a lo largo de la Sección de experimentación y resultados, donde cada paso de la experimentación realizada venía seguida de un análisis estadístico de las correspondientes pruebas.

5.4 Resumen del capítulo

En este capítulo se aborda la experimentación realizada para la aplicación de la metodología propuesta en el Capítulo 4 con los detalles técnicos de las ejecuciones. También se presentan los resultados obtenidos y su respectiva discusión.

Los resultados obtenidos permitieron establecer las pruebas suficientes para la validación de la hipótesis propuesta en esta tesis. El enfoque de RCA aplicado a una de las herramientas más potentes de clasificación de imágenes, como son las RNCs, robusteció el algoritmo al identificar correctamente las instancias de *clases desconocidas*, con lo cual a su vez, redujo el riesgo de *conjunto abierto*.

Con la experimentación realizada a lo largo de este capítulo, se probó que el mecanismo propuesto no solo es capaz de detectar las instancias de *clases desconocidas*, sino que es capaz de descubrir las clases ocultas en ellas, probando que con la metodología propuesta una RNC puede aumentar su capacidad de reconocimiento, con lo cual no solo reaccionará a las entradas que se ingresen cuando se pone a prueba sino que seguirá aprendiendo incluso después de ser entrenada.

Por otra parte, si bien esta metodología puede ser ajustada para ser aplicada a diversos dominios, el desarrollar y probar la metodología con una base de datos de un dominio específico y no con uno de los benchmarks más conocidos como ImageNet, CIFAR-100, etc., aseguró que el enfoque propuesto es eficiente para tratar con el problema de RMMV.

La mayor parte de la experimentación de este capítulo se utilizó para la publicación de un artículo en la revista Mathematical and Computational Applications (MCA), JCR, Q2, como parte del número especial New Trends in Computational Intelligence and Applications 2022. Este artículo se adjunta en el Apéndice 1.

Una parte de la experimentación de este capítulo se utilizó para el desarrollo de un artículo presentado en el 4to Workshop en New Trends in Computational Intelligence and Applications (CIAPP 2022), el cual se encuentra publicado en los workshops proceedings del MICAI 2022. Este artículo se adjunta en el Apéndice 2.

Capítulo 6

Conclusiones y Trabajo futuro

En esta tesis se presentó una estrategia para abordar el RMMV como un problema de RCA con una extensión para el descubrimiento de *clases nuevas*, tomando como guía principal la distribución consistente entre clases de los vectores de características generados por una RNC de dominio específico. Hasta dónde llega el conocimiento del autor, esta es la primera vez que se aborda el RMMV desde un enfoque de RCA y es uno de los pocos trabajos de RCA de dominio específico que se enfoca en el descubrimiento de las clases ocultas dentro de las instancias de *clases desconocidas*.

Los algoritmos de clasificación de imágenes que emplean aprendizaje supervisado son conocidos por su potencia y precisión de clasificación, sin embargo, la mayoría son desarrollados asumiendo un ambiente de *conjunto cerrado,* por lo que carecen de mecanismos que les permitan lidiar con imágenes de *clases desconocidas*. Por esta razón, en el presente trabajo se decidió añadir un mecanismo de detección de objetos de *clases desconocidas* y de descubrimiento de *clases nuevas* a uno de los clasificadores de imágenes más utilizados como son las RNCs y demostrar que es posible ampliar su potencial de clasificación y aumentar su robustez para trabajar de forma más efectiva en escenarios reales. Con los resultados mostrados en este trabajo se pudo validar que, al aplicar la metodología propuesta, la RNC diseñada para el dominio específico del RMMV no sólo reacciona a las consultas, sino que es capaz de seguir aprendiendo, incluso después de haber sido entrenada.

La solución presentada en este trabajo alcanzó una puntuación micro-F1 de 1.00 al lograr clasificar las instancias de *clases conocidas* del conjunto de prueba con una precisión del 100%, lograr identificar las instancias de *clases desconocidas* dentro del conjunto de prueba y lograr descubrir las clases ocultas dentro de las instancias de *clases desconocidas*. Si el

conjunto de prueba utilizado para probar la metodología propuesta en este trabajo se ingresara a un algoritmo de clasificación que haya sido desarrollado en un contexto de *conjunto cerrado*, todas las instancias de *clases desconocidas* (9 de las 24 imágenes del conjunto de prueba pertenecían a *clases desconocidas*) se habrían clasificado dentro de alguna clase conocida, por lo que el modelo no habría podido alcanzar una precisión de clasificación superior al 62.5%. La pobre precisión de clasificación que podría lograr el clasificador de *conjunto cerrado* en el escenario de la vida real que simula el conjunto de prueba empleado en este trabajo, se debería a su conocimiento incompleto del mundo. Esto resalta la importancia de la metodología propuesta en este trabajo que sí es capaz de lidiar con instancias de *clases desconocidas*, lo cual es muy deseable si se planea enfrentar el clasificador a un escenario de la vida real.

La efectividad de la metodología propuesta se debió en gran medida a la función de pérdida contrastiva SupCon que fue empleada para diseñar y entrenar la RNC de dominio específico, si bien el proceso de NE brindó la herramienta que facilitó la búsqueda de la arquitectura, SupCon dirigió la búsqueda para encontrar la arquitectura que generara un mapeo consistente entre clases de los vectores de características en el espacio. El mapeo consistente que se logró gracias a SupCon permitió que la estrategia propuesta no esté restringida por los datos de entrenamiento, sino que se pueda ajustar a medida que los datos cambian, ya que es capaz de generalizar la distribución consistente dentro del espacio para instancias de *clases desconocidas*, lo cual fue el punto clave para el funcionamiento efectivo del mecanismo para la detección de instancias de clases *desconocidas* y descubrimiento de *clases nuevas*.

Este trabajo también pretende resaltar la importancia de generar estrategias de RCA de dominio específico y la necesidad de aplicarlas a problemas de clasificación/reconocimiento del mundo real como es el RMMV, para obtener clasificadores no sólo más precisos sino también más robustos, al estar preparados para enfrentarse a escenarios de dominio específico del mundo real.

Aunque este trabajo se centró en el dominio de RMMV, la metodología propuesta puede ajustarse para ser aplicada a otros dominios como el reconocimiento de dígitos escritos a mano, la clasificación de radiografías de tórax, etc. De igual manera los resultados obtenidos

pueden utilizarse como punto de referencia para futuros trabajos de RCA aplicados al dominio de RMMV.

Como trabajo futuro, la metodología propuesta se puede ampliar o mejorar aplicando los siguientes enfoques:

- Crear una muestra estadísticamente más representativa de ejecuciones del algoritmo de Neuroevolución. Debido a limitaciones de tiempo, el algoritmo de NE solo se ejecutó siete veces para el desarrollo de este trabajo, sin embargo, es recomendable realizar al menos 30 ejecuciones de los AG.
- Utilizar un algoritmo de calibración de parámetros para la ejecución del algoritmo de Neuroevolución con lo cual posiblemente se logre una mejor búsqueda y se obtengan resultados con un mejor valor de aptitud.
- Aumentar el número de *clases conocidas* en el entrenamiento y diseño de la RNC de dominio específico. Con la aplicación de este enfoque posiblemente se aumente el potencial de clasificación y reconocimiento de *clases conocidas*.
- Aplicar otras estrategias de RCA, como la presentada en [38], al problema de RMMV para obtener una comparación de resultados más representativa.
- Mejorar la estrategia para la determinación de *clases conocidas* o *desconocidas* de los conjuntos de la mejor solución. Se propone como posible opción el cálculo de la entropía para calcular la incertidumbre basándose en las probabilidades obtenidas con el MMG.

Referencias

[1] Naseer S., Shah S. M. A., Aziz S., Khan M. U. and Iqtidar K. (2020). "Vehicle Make and Model Recognition using Deep Transfer Learning and Support Vector Machines". In: *IEEE 23rd International Multitopic Conference (INMIC)*, pp. 1-6. DOI: 10.1109/INMIC50486.2020.9318063.

[2] Nazemi A., Azimifar Z., Shafiee M. J. and Wong A. (2020). "Real-Time Vehicle Make and Model Recognition Using Unsupervised Feature Learning". In: *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 7, pp. 3080-3090. DOI: 10.1109/TITS.2019.2924830.

[3] Nazemi A., Shafiee M. J., Azimifar Z. (2013). "On road vehicle make and model recognition via sparse feature coding". In: *8th Iranian Conference on Machine Vision and Image Processing (MVIP)*, Zanjan, pp. 436-440. DOI: 10.1109/IranianMVIP.2013.6780025.

[4] Krause J., Stark M., Deng J. and Fei-Fei L. (2013). "3D Object Representations for Fine-Grained Categorization". In: *IEEE International Conference on Computer Vision Workshops*, pp. 554-561. DOI: 10.1109/ICCVW.2013.77.

[5] Gervais, N. (2020). *The Car Connection* [Dataset]. https://github.com/nicolas-gervais/predicting-car-price-from-scraped-data/tree/master/picture-scraper

[6] Tafazzoli F., Frigui H. and Nishiyama K. (2017). "A Large and Diverse Dataset for Improved Vehicle Make and Model Recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 874-881. DOI: 10.1109/CVPRW.2017.121.

[7] Handl J., Knowles J. (2004) "Multiobjective clustering with automatic determination of the number of clusters". Technical Report TR-COMPSYSBIO-2004-02. UMIST, Manchester, UK

[8] Corne D.W., Jerram N.R., Knowles J.D., Oates M.J. (2001). "PESA-II: Region-Based Selection in Evolutionary Multiobjective Optimization". In: *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*. Morgan Kaufmann Publishers Inc, San Francisco, pp 283–290.

[9] Martínez-Peñaloza, M. G., Mezura-Montes, E., Cruz-Ramírez, N., Acosta-Mesa, H. G., & Ríos-Figueroa, H. V. (2016). "Improved multi-objective clustering with automatic determination of the number of clusters". In: *Neural Computing and Applications*, 28(8), 2255–2275. DOI: 10.1007/s00521-016-2191-1.

[10] Gustavo-Adolfo Vargas-Hákim, Efrén Mezura-Montes, and Héctor-Gabriel Acosta-Mesa. (2021). "Hybrid encodings for neuroevolution of convolutional neural networks: a case study". In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '21)*. Association for Computing Machinery, New York, NY, USA, 1762–1770. DOI: 10.1145/3449726.3463133.

[11] Lu L. and Huang H. (2019). "A Hierarchical Scheme for Vehicle Make and Model Recognition From Frontal Images of Vehicles". In: *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 5, pp. 1774-1786. DOI: 10.1109/TITS.2018.2835471.

[12] Fang J., Zhou Y., Yu Y. and Du S. (2017). "Fine-Grained Vehicle Model Recognition Using A Coarse-to-Fine Convolutional Neural Network Architecture". In: *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 7, pp. 1782-1792. DOI: 10.1109/TITS.2016.2620495.

[13] Hadsell R., Chopra S., LeCun Y. (2006). "Dimensionality Reduction by Learning an Invariant Mapping". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, pp. 1735-1742. DOI: 10.1109/CVPR.2006.100.

[14] Khosla P., Teterwak P., Wang C., Sarna A., Tian Y., Isola P., Maschinot A., Liu C., Krishnan D. (2020). "Supervised contrastive learning". In: *Advances in Neural Information Processing Systems*, vol. 33 pp. 18661-18673. DOI: 10.48550/arXiv.2004.11362.

[15] Van der Maaten L., Hinton G.: Visualizing Data using t-SNE. In: Journal of Machine Learning Research, 9, 2579—2605 (2008).

[16] Boukerche A., Ma X. (2022). "A Novel Smart Lightweight Visual Attention Model for Fine-Grained Vehicle Recognition". In: *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 13846-13862. DOI: 10.1109/TITS.2021.3131530.

[17] Vargas-Hákim G. -A., Mezura-Montes E. and Acosta-Mesa H. -G. "A Review on Convolutional Neural Network Encodings for Neuroevolution". In: *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 1, pp. 12-27, Feb. 2022, doi: 10.1109/TEVC.2021.3088631.

[18] Hu M., Wang W., Liu L., and Liu Y. "Apenas: An asynchronous parallel evolution based multi-objective neural architecture search". In: *IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, Exeter, United Kingdom, 2020, pp. 153-159, doi: 10.1109/ISPA-BDCloud-SocialCom-SustainCom51426.2020.00045.

[19] Hassaballah M., Kenk M. A., Muhammad K., Minaee S. (2021). "Vehicle Detection and Tracking in Adverse Weather Using a Deep Learning Framework". In *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4230-4242. DOI: 10.1109/TITS.2020.3014013.

[20] Hussain K.F., Afifi M., Moussa G. (2019). "A Comprehensive Study of the Effect of Spatial Resolution and Color of Digital Images on Vehicle Classification". In *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 3, pp. 1181-1190. DOI: 10.1109/TITS.2018.2838117.

[21] Agarwal A., Shinde S., Mohite S., Jadhav S. (2022). "Vehicle Characteristic Recognition by Appearance: Computer Vision Methods for Vehicle Make, Color, and License Plate

Classification". In: *IEEE Pune Section International Conference (PuneCon)*, Pune, India, pp. 1-6. DOI: 10.1109/PuneCon55413.2022.10014731.

[22] Zhang H., Jin Y., Cheng R., and Hao K. "Efficient evolutionary search of attention convolutional networks via sampled training and node inheritance". In: *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 2, pp. 371-385, April 2021, doi: 10.1109/TEVC.2020.3040272.

[23] Kezebou L., Oludare V., Panetta K., Agaian S. "Few-Shots Learning for Fine-Grained Vehicle Model Recognition". In: *IEEE International Symposium on Technologies for Homeland Security (HST)*, pp. 1-9 (2021). DOI: 10.1109/HST53381.2021.9619823.

[24] Masana M., Liu X., Twardowski B., Menta M., Bagdanov A. D. and van de Weijer J.: "Class-Incremental Learning: Survey and Performance Evaluation on Image Classification". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5513-5533, 1 May 2023. DOI: 10.1109/TPAMI.2022.3213473.

[25] Hafeez M. A., Ul-Hasan A. and Shafait F. "Incremental Learning of Object Detector with Limited Training Data". In: *Digital Image Computing: Techniques and Applications (DICTA)*, Gold Coast, Australia, 2021, pp. 01-08. doi: 10.1109/DICTA52665.2021.9647245.

[26] Zhang F. "Learning Unsupervised Side Information for Zero-Shot Learning". In: *International Conference on Signal Processing and Machine Learning (CONF-SPML)*, Stanford, CA, USA, 2021, pp. 325-328. doi: 10.1109/CONF-SPML54095.2021.00070.

[27] Li Y., Kong D., Zhang Y., Chen R. and Chen J. "Representation Learning of Remote Sensing Knowledge Graph for Zero-Shot Remote Sensing Image Scene Classification". In: *IEEE International Geoscience and Remote Sensing Symposium IGARSS*, Brussels, Belgium, 2021, pp. 1351-1354. doi: 10.1109/IGARSS47720.2021.9553667.

[28] Zhou F., Zhang L., Wei W., Bai Z. and Zhang Y. "Meta Transfer Learning for Few-Shot Hyperspectral Image Classification". In: *IEEE International Geoscience and Remote Sensing Symposium IGARSS*, Brussels, Belgium, 2021, pp. 3681-3684. doi: 10.1109/IGARSS47720.2021.9553981.

[29] Scheirer W. J., de Rezende Rocha A., Sapkota A. and Boult T. E. "Toward Open Set Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 7, pp. 1757-1772, July 2013. doi: 10.1109/TPAMI.2012.256.

[30] Scheirer W. J., Jain L. P. and Boult T. E. "Probability Models for Open Set Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2317-2324, 1 Nov. 2014. doi: 10.1109/TPAMI.2014.2321392.

[31] Rudd E. M., Jain L. P., Scheirer W. J. and Boult T. E. "The Extreme Value Machine". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 762-768, 1 March 2018. doi: 10.1109/TPAMI.2017.2707495.

[32] Ribeiro Mendes Júnior P., Boult T. E., Wainer J. and Rocha A. “Open-Set Support Vector Machines”. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 6, pp. 3785-3798, June 2022. doi: 10.1109/TSMC.2021.3074496.

[33] Alfarisy G. A. F., Malik O. A. and Hong O. W. “Quad-Channel Contrastive Prototype Networks for Open-Set Recognition in Domain-Specific Tasks”. In: *IEEE Access*, vol. 11, pp. 48578-48592, 2023. doi: 10.1109/ACCESS.2023.3275743.

[34] Bendale A. and Boult T. “Towards Open World Recognition”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, 2015, pp. 1893-1902. doi: 10.1109/CVPR.2015.7298799.

[35] Wang Z., Salehi B., Gritsenko A., Chowdhury K., Ioannidis S. and Dy J. “Open-World Class Discovery with Kernel Networks”. In: *IEEE International Conference on Data Mining (ICDM), Sorrento*, Italy, 2020, pp. 631-640. doi: 10.1109/ICDM50108.2020.00072.

[36] Han K., Rebuffi S. -A., Ehrhardt S., Vedaldi A. and Zisserman A. “AutoNovel: Automatically Discovering and Learning Novel Visual Categories”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 10, pp. 6767-6781, 1 Oct. 2022. doi: 10.1109/TPAMI.2021.3091944.

[37] Han K., Vedaldi A. and Zisserman A. “Learning to Discover Novel Visual Categories via Deep Transfer Clustering”. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South), 2019, pp. 8400-8408. doi: 10.1109/ICCV.2019.00849.

[38] Geng C. and Chen S. “Collective Decision for Open Set Recognition”. In: *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 192-204, 1 Jan. 2022. doi: 10.1109/TKDE.2020.2978199.

[39] Sun Y., Xue B., Zhang M. and Yen G. G. “Evolving deep convolutional neural networks for image classification”. In: *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 394-407, April 2020, doi: 10.1109/TEVC.2019.2916183.

[40] Geng C., Huang S. -J. and Chen S. "Recent Advances in Open Set Recognition: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3614-3631, 1 Oct. 2021, doi: 10.1109/TPAMI.2020.2981604.

[41] Sun X., Yang Z., Zhang C., Ling K. -V. and Peng G. “Conditional Gaussian Distribution Learning for Open Set Recognition”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, 2020, pp. 13477-13486, doi: 10.1109/CVPR42600.2020.01349.

[42] Ye W., Liu R., Li Y. and Jiao L. “Quantum-inspired evolutionary algorithm for convolutional neural networks architecture search”. In: *IEEE Congress on Evolutionary Computation (CEC)*, Glasgow, UK, 2020, pp. 1-8, doi: 10.1109/CEC48606.2020.9185727.

[43] Liu J., Zhou S., Wu Y., Chen K., Ouyang W. and Xu D. “Block proposal neural architecture search”. In: *IEEE Transactions on Image Processing*, vol. 30, pp. 15-25, 2021, doi: 10.1109/TIP.2020.3028288.

[44] Zhou Y., Gen G. G., and Yi Z. "A knee-guided evolutionary algorithm for compressing deep neural networks". In: *IEEE Transactions on Cybernetics*, vol. 51, no. 3, pp. 1626-1638, March 2021, doi: 10.1109/TCYB.2019.2928174.

[45] Operiano K. R. G., Iba H. and Pora W. "Neuroevolution architecture backbone for x−ray object detection". In: *IEEE Symposium Series on Computational Intelligence (SSCI)*, Canberra, ACT, Australia, 2020, pp. 2296-2303, doi: 10.1109/SSCI47803.2020.9308453.

[46] Hassanzadeh T., Essam D. and Sarker R. "2D to 3D evolutionary deep convolutional neural networks for medical image segmentation". In: *IEEE Transactions on Medical Imaging*, vol. 40, no. 2, pp. 712-721, Feb. 2021, doi: 10.1109/TMI.2020.3035555.

Apéndices de publicaciones

Artículo publicado en MCA como parte del número especial New Trends in Computational Intelligence and Applications 2022

Mathematical and Computational Applications

MDPI

Article

Vehicle Make and Model Recognition as an Open-Set Recognition Problem and New Class Discovery

Diana-Itzel Vázquez-Santiago, Héctor-Gabriel Acosta-Mesa * and Efrén Mezura-Montes

Artificial Intelligence Research Institute, Universidad Veracruzana, Veracruz 91097, Mexico; diana.v.s@hotmail.com (D.-I.V.-S.); emezura@uv.mx (E.M.-M.)
* Correspondence: heacosta@uv.mx

Abstract: One of the main limitations of traditional neural-network-based classifiers is the assumption that all query data are well represented within their training set. Unfortunately, in real-life scenarios, this is often not the case, and unknown class data may appear during testing, which drastically weakens the robustness of the algorithms. For this type of problem, open-set recognition (OSR) proposes a new approach where it is assumed that the world knowledge of algorithms is incomplete, so they must be prepared to detect and reject objects of unknown classes. However, the goal of this approach does not include the detection of new classes hidden within the rejected instances, which would be beneficial to increase the model's knowledge and classification capability, even after training. This paper proposes an OSR strategy with an extension for new class discovery aimed at vehicle make and model recognition. We use a neuroevolution technique and the contrastive loss function to design a domain-specific CNN that generates a consistent distribution of feature vectors belonging to the same class within the embedded space in terms of cosine similarity, maintaining this behavior in unknown classes, which serves as the main guide for a probabilistic model and a clustering algorithm to simultaneously detect objects of new classes and discover their classes. The results show that the presented strategy works effectively to address the VMMR problem as an OSR problem and furthermore is able to simultaneously recognize the new classes hidden within the rejected objects. OSR is focused on demonstrating its effectiveness with benchmark databases that are not domain-specific. VMMR is focused on improving its classification accuracy; however, since it is a real-world recognition problem, it should have strategies to deal with unknown data, which has not been extensively addressed and, to the best of our knowledge, has never been considered from an OSR perspective, so this work also contributes as a benchmark for future domain-specific OSR.

Keywords: open-set recognition; new class discovery; VMMR; CNN; contrastive loss; clustering; neuroevolution

Citation: Vázquez-Santiago, D.-I.; Acosta-Mesa, H.-G.; Mezura-Montes, E. Vehicle Make and Model Recognition as an Open-Set Recognition Problem and New Class Discovery. *Math. Comput. Appl.* **2023**, *28*, 80. https://doi.org/10.3390/mca28040080

Academic Editor: Leonardo Trujillo

Received: 14 March 2023
Revised: 28 June 2023
Accepted: 29 June 2023
Published: 3 July 2023

1. Introduction

Automatic vehicle make and model recognition (VMMR) aims to offer innovative services to improve the efficiency and safety of transportation networks. These services include intelligent traffic analysis and management, electronic toll collection, emergency vehicle notifications, the automatic enforcement of traffic rules, etc. In recent years, several authors have proposed and implemented different approaches and techniques to present solutions to the various challenges of VMMR such as the similar appearance of different vehicle models [1,2], variations in the images due to weather conditions or resolution [3–5], recognition through different key points or regions [6,7], etc. However, most of these solutions are designed within a *closed-set* approach, where it is assumed that all query data are well represented by the training set, and therefore these solutions lack mechanisms to detect during testing when an input sample does not belong to any of the predefined classes. These unforeseen situations are very likely to happen in real-life scenarios and drastically weaken the robustness of the models.

Every day, we have more and more access to labeled data, which makes data-hungry algorithms such as classification algorithms that employ supervised learning improve their classification accuracy by having more training information. However, it is unrealistic to think that we will be able to train these algorithms to recognize any object that may be presented to them. In the specific case of this application domain, it is estimated that there are currently more than 3300 vehicle makes in the world, for which models have been added and removed from the market, modifying the design in each generation and producing different versions of the same vehicle, which has made it very difficult to have a database containing enough examples of all the existing vehicles in circulation to correctly train a model. This limitation is very common in real-world recognition/classification tasks such as VMMR and, in most cases, results in misclassified vehicles because the algorithms were not prepared to deal with objects of unknown (novel) classes.

To solve this problem, some strategies have been proposed, such as periodically retraining the algorithms, incorporating an incremental update mechanism [8,9], using zero-shot [10,11] or one-shot (few-shot) [12,13] learning, etc. Although these strategies provide models with greater flexibility or the possibility of eventually increasing their classification potential, they do not address the fundamental problem of recognizing a novel class during testing (*open-set* problem). Scheirer et al. were the first to describe a more realistic scenario in which new classes not seen in training appear in testing and require classifiers not only to accurately classify objects of known classes but also to effectively deal with classes not considered in the training set [14]. They formalized this problem as *open-set recognition* (OSR) and proposed a solution called 1-vs-Set machine, where the risk of labeling a sample as known if it is far from the known data (*open space*) is measured, and its objective is to minimize this risk (*open-space risk*) by rejecting queries that lie beyond the reasonable support of the known data.

OSR led to extensive research that mostly focused on more effectively limiting the *open-space risk* [15–18], and little research was developed around efficiently performing *open-set recognition* and simultaneously discovering new classes hidden in the rejected instances. Some of the proposed solutions employed incremental learning [19], transfer learning [20,21], or clustering [22,23]. Although they achieved good results, most of them present limitations such as the determination of the number of new classes in a later or separate event from the recognition of novel instances, or the use of examples of unknown classes during validation, pretraining, or retraining stages as a strategy to fine-tune their representations/parameters; however, in OSR, there is almost never information of unknown classes.

In the specific case of VMMR, few works have been proposed that, although not described within an OSR framework, have mechanisms to deal with new classes. One of these studies was conducted by Nazemi et al. [3] from an anomaly detection approach. Their base system is capable of classifying 50 specific vehicle models, to which they added an anomaly detection based on a confidence threshold to identify vehicles that do not belong to any of these 50 classes. The "anomalies" are further classified based on their dimensions within two new classes: "unknown heavy" and "unknown light". Another approach was proposed by Kezebou et al. [12], with a few-shot learning approach requiring between 1 and 20 images for the generation of new classes.

In this paper, we propose to approach VMMR as an OSR problem extended for new class discovery. Since the known classes are supported by numerous well-labeled examples, we can very effectively train an image classification algorithm that employs supervised learning like convolutional neural networks (CNNs), which are the most widely used tool for this task. While these networks cannot deal with the recognition of new classes, their ability to extract meaningful features can be exploited to design a mechanism that can detect objects of new classes based on the distribution of feature vectors in the embedded space that, when aggregated between feature extraction and classification, would adopt an OSR approach. However, feature vectors are usually of high dimensionality, their distribution is not always clear, and there is no assurance that the behavior will be maintained in

instances of unknown classes, which can complicate the representation and interpretation of the space to detect new classes. To tackle these problems, we propose to train a CNN with contrastive learning using the contrastive loss function during the training stage to reorganize the space where the feature vectors are mapped. Instead of separating the images with a hyperplane, the contrastive loss function brings similar images in near space (in terms of, e.g., Euclidean distance, cosine similarity, or some other metric) and moves dissimilar images away, generalizing this behavior on new unseen data.

Although there are CNN architectures such as VGG16, AlexNet, etc., that have achieved state-of-the-art results in the most well-known benchmarks such as ImageNet, CIFAR-100, etc., we propose a new CNN architecture designed from images of the application domain of this work (VMM) and the contrastive loss function using a neuroevolution technique to ensure consistent distribution of feature vectors within the embedded space, which serves as the main guide for a probabilistic model and a clustering algorithm that carry out the detection of objects of new classes and simultaneously discover their classes.

The remainder of this paper is organized as follows: Section 2 presents the related work. Section 3 describes the proposed methodology to approach VMMR as an OSR problem with an extension for new class discovery. This section also presents the proposed global scheme and delves deeper into each stage, detailing how the techniques of neuroevolution, contrastive loss function, the probabilistic model, and clustering are linked so as to achieve the general purpose. Section 4 details the tests performed, including the parameters and justifications for each test and the results obtained at each stage with their respective interpretation. Finally, the conclusions are drawn, and future work is discussed in Section 5.

2. Literature Review

2.1. Open-Set Recognition

OSR [14] introduced a more realistic scenario for real-world recognition/classification tasks, where new classes not seen during training appear at query time during testing. To deal with these unforeseen situations, OSR algorithms have to consider that their knowledge of the world is incomplete and formulate strategies to minimize the risk of considering an unknown instance as known. The authors of [14] formalized this risk as an *open-set risk* (R_O) in a probabilistic formulation (Equation (1)) as the relative measure of positively labeled open space O compared with the overall measure of positively labeled space S_O.

$$R_O(f) = \frac{\int_O f\,(x)dx}{\int_{S_O} f\,(x)dx'} \tag{1}$$

where f denotes a measurable recognition function.

Numerous studies have been conducted to minimize the risk of open sets and more effectively reject objects of unknown classes [15–18], which is the main goal of OSR. However, in a more desirable context, an OSR should go further and discover the unknown classes hidden inside the rejected objects. Within this context, some authors have proposed the use of incremental learning [19], transfer learning [20,21], or clustering [22]. Bendale and Boult [19] extended the *open-set recognition* problem to open-world recognition (OWR) to jointly consider the OSR and incremental learning of new classes. They proposed that an effective OWR system must perform four tasks: detecting unknown objects, choosing which objects to label for addition to the model, labeling these objects, and updating the model. In their paper, they presented the NNO algorithm. However, the tasks they proposed are not automated in the NNO, they require human supervision for labeling, and the determination of the number of classes happens in a later event after the recognition of new instances. In [20], Wang et al. studied the OWR problem in more detail by incorporating transfer learning to transfer knowledge from old classes to new ones. However, they needed to retrain their model with the presence of samples of unknown classes, which is a limitation since, in an OSR context, information from unknown classes is almost never available. A similar knowledge transfer proposal was presented by Han et al. [21]; however, they have

the same limitation since their idea was to pretrain their model with images of known and unknown classes. Another interesting proposal was developed in [22] by some authors of [21], where they also took advantage of the knowledge transfer approach but used clustering. The main limitation of this work is that they determined the number of new classes in a separate event from the discovery of new instances, which, as in [19], can lead to suboptimal solutions.

To our knowledge, the most related work to ours, in terms of simultaneously discovering the objects of new classes and these classes themselves, is [23]. They introduced a collective/batch decision-strategy-based OSR framework (CD-OSR) by slightly modifying the hierarchical Dirichlet process (HDP). CD-OSR first involves a co-clustering process in the training phase to obtain the appropriate parameters. In the testing phase, each known class is modeled as a group using a Gaussian mixture model (GMM) with an unknown number of subclasses (one or more subclasses representing the same class can be obtained), and the entire test set (collective/batch) is treated in the same way. Then, all the groups are co-clustered under the HDP framework, and each one is labeled as one of the known classes or as unknown, depending on whether the subclass assigned to it is associated with a known class or not. Other works on OSR such as [24] also took advantage of Gaussian distributions to obtain discriminative representations of the data to detect unknowns and classify knowns.

Another proposal that may be related to our work was presented in [18], where the OSR problem was addressed within a transfer learning approach using contrastive learning to model the data. They also highlighted the importance of developing and testing OSR solutions with domain-specific databases to test their efficiency in dealing with real-world applications. Unfortunately, this solution only rejects objects of new classes and does not include the discovery of their classes.

2.2. Neuroevolution and Contrastive Learning

In the field of evolutionary computation (EC), a technique called neuroevolution (NE) emerged to optimize artificial neural networks (ANNs) at different levels. Its current overall process can be summarized as follows: A random population of individual networks is generated (with a neural coding), real networks are created from them, and the networks are evaluated with a function that measures the quality of the results (the fitness function). The networks with the highest fitness are selected, certain random changes are introduced to generate offspring from them, and a new population (generation) is selected. This process is repeated until a certain level of fitness or number of generations is reached.

NE has achieved excellent results in this optimization task and has rapidly advanced toward the optimization of CNN topologies [25–32]. A crucial point in the performance of NE algorithms is neural encodings, which contain the topology information of an ANN and therefore have a great impact on the complexity of the search space. So, in order to implement this technique in CNNs, NE was faced with the problem of designing neural encodings that could abstract the parameters of CNNs in order to deal with these highly complex architectures. There are two types of neural encodings that are commonly employed: direct and indirect. Among the proposals using an indirect coding framework, we find works such as [25–27], and in the case of works that used indirect encodings, we find proposals such as [28–30]. In recent years, researchers have started to study a "hybrid" neural coding, which combines elements of the encodings mentioned above to eliminate some of their limitations. These "hybrid" representations have proved to be very useful to distribute the CNN representations in different substructures, leading to improvement in the search [31–33]. The advantages and disadvantages of different encoding schemes, as well as important niches of opportunity for future research, were described in detail in [34].

Although NE algorithms have a strategy to determine how well individuals are meeting the criterion (or criteria) being optimized (the fitness function), CNNs have their own strategy to quantify how close their predictions are to the expected output (the loss function), for which cross-entropy or negative log-likelihood are some of the most

frequently used functions. However, the study of these functions has continued, and alternatives have been proposed that have achieved superior results. In these advances, the supervised contrastive loss function (SupCon) [35] was developed following the contrastive learning approach but in a supervised environment, which allowed it to maintain the principle of mapping examples into the embedded space of contrastive learning (distance is minimized in terms of Euclidean distance, cosine similarity, etc., between similar objects and maximized for dissimilar objects) but take advantage of labeled data.

Although there are marked differences between various versions of contrastive loss functions, the family of contrastive loss functions, in general, considers the following: For A set of N randomly sampled sample/label pairs (batch), $\{x_k, y_k\}$ $k = 1 \ldots N$ is considered; the corresponding batch used for training (multiviewed batch) consists of $2N$ pairs, $\left\{\widetilde{x}_\ell, \widetilde{y}_\ell\right\}_{\ell=1\ldots 2N}$, where $\widetilde{x}_{2k}$ and $\widetilde{x}_{2k-1}$ are two random augmentations ("views") of $x_k (k = 1 \ldots N)$ and $\widetilde{y}_{2k-1} = \widetilde{y}_{2k} = y_k$. Given the above, SupCon is calculated as follows:

$$\mathcal{L}_{in}^{sup} = \sum_{i \in I} \mathcal{L}_{in,i}^{sup} = \sum_{i \in I} -\log \left\{ \frac{1}{|P(i)|} \sum_{p \in P(i)} \frac{\exp\left(z_i \cdot z_p / \tau\right)}{\sum_{a \in A(i)} \exp\left(z_i \cdot z_a / \tau\right)} \right\} \tag{2}$$

where $i \in I \equiv \{1 \ldots 2N\}$ is the index of an augmented sample (anchor), $A(i) \equiv I \setminus \{i\}$ is the set of all the indices of the samples different than i, $P(i) \equiv \left\{p \in A(i) : \widetilde{y}_p = \widetilde{y}_i\right\}$ is the set of all positive sample indices different than i, $|P(i)|$ is its cardinality, the $\bullet$ symbol denotes the inner (dot) product, and τ is a scalar temperature parameter. SupCon's formulation generalizes the SimCLR loss function [36] to an arbitrary number of positive examples to deal with scenarios in which labels are available so that it is known that more than one sample can belong to the same class.

3. Materials and Methods

This section describes the methodology proposed to approach VMMR as an OSR problem with an extension for new class discovery. Figure 1 shows the overall process of our proposal, and the following subsections describe the process in detail, covering the following objectives:

1. Employ an NE algorithm and contrastive learning to design a domain-specific CNN that generates feature vectors spatially close in terms of cosine distance if the instances belong to the same class and distant if they belong to different classes, preserving this behavior in instances of unknown classes.
2. Implement a mechanism between the feature extraction and classification sections of the CNN capable of detecting objects of unknown classes and simultaneously discovering their classes, taking the mapping of feature vectors, described in the previous objective, as the main guide.
3. Run a series of tests using the test set that includes images of classes with which the CNN was designed and trained (known) and images of new classes (unknown) to test that the algorithm is able to detect objects of unknown classes and simultaneously discover their classes.
4. Classify images of known classes with a classification accuracy above 90%.

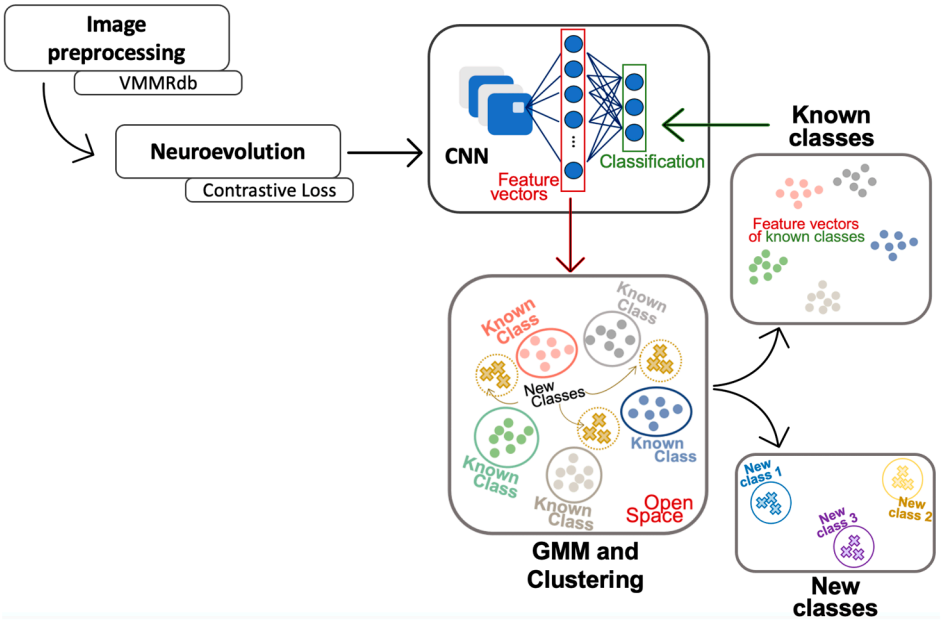

Figure 1. The proposed global process to approach VMMR as an OSR problem with an extension for new class discovery.

3.1. Dataset

The VMMRdb database [37] (available at https://github.com/faezetta/VMMRdb, accessed on 13 March 2023) was used in this work since it is one of the most cited in the specialized literature [12,38,39]. Only eight classes from the VMMRdb database were used and were manually filtered to retain only unduplicated images showing the rear view of the vehicles (i.e., samples of each class were not balanced). The filtered images were transformed to grayscale, resized to 28×28 pixels, and normalized with a mean of 0.456 and a standard deviation of 0.224.

Of the eight classes, five classes were used as "*known classes*": Chevrolet Silverado 2004, Ford Explorer 2002, Ford Mustang 2000, Honda Civic 2002, and Nissan Altima 2005. A sample of six images of each "*known class*" was used in the NE process to design the domain-specific CNN for VMMR. For the training of the resulting CNN from the NE process, the largest number of examples per class was needed, which had to be the same among different classes. However, due to the number of available samples in the database and the image filtering mentioned above, the final number of functional samples per "*known class*" varied between 75 and 250 images. Among the functional samples, three images of each class were kept for testing the complete OSR framework, and the rest were subjected to a data augmentation process to balance the number of examples per class, resulting in 250 images of each "*known class*". Furthermore, 200 images were used to train the CNN and model the "*known classes*" with a Gaussian mixture model (GMM), and the remaining 50 images were used to test the CNN classification accuracy and define the threshold of "*known classes*" in the GMM.

From the three remaining classes chosen from the database (Acura RSX 2003, Chevrolet Avalanche 2009, and Ford Escape 2011), three images of each class were chosen to only be used during the testing stage to represent "*unknown classes*" and validate that the proposed approach can detect them and discover their classes.

3.2. Neuroevolution and Contrastive Loss

One of the main objectives of this work is to exploit the ability of a CNN to extract meaningful features for designing a mechanism to detect objects of unknown classes based on the distribution of the feature vectors in the embedded space. To facilitate the interpretation of the embedded space, feature vectors extracted using the CNN are

considered to be spatially close in terms of cosine similarity if they belong to the same class and spatially distant if they belong to different classes, maintaining this behavior even if the classes are unknown. According to the state-of-the-art review, adding the contrastive loss function to the CNNs causes the feature vectors to be mapped in near space (in terms of, e.g., Euclidean distance, cosine similarity, or some other metric) if they are similar and far if they are dissimilar.

Although there are CNN architectures such as VGG16, AlexNet, etc., that have achieved state-of-the-art results in the most well-known benchmarks such as ImageNet, CIFAR-100, etc., we propose a new domain-specific architecture that would generate the previously described behavior in feature vectors, using the images mentioned in Section 3.1, an NE algorithm called DeepGA [33] (shown in Algorithm 1), and SupCon [35] expressed in Equation (2).

In [35], the authors made their PyTorch implementation of SupCon generally available (https://t.ly/supcon, accessed on 13 March 2023), and this was used in this work as a loss function in the CNNs generated in the NE process with DeepGA. (Originally, the negative log-likelihood loss was used.) The fitness function of DeepGA (Algorithm 1, line 15) was also modified to measure the desired behavior in feature vectors since optimization was the objective of our study. Thus, as the fitness function, we used the value of SupCon in the last training epoch of each generated CNN. Since the loss function decreases as the desired output is approached, DeepGA was set to work as a minimization problem, i.e., as the generated CNNs approached the desired target, the value of the loss/fitness function decreased.

The hybrid coding employed in DeepGA allows the algorithm to consider the number of fully connected layers and their corresponding number of neurons in its search for the best solution. However, during the NE process, it was detected that leaving the number of fully connected layers to DeepGA only increased the complexity and execution time since with only two fully connected layers, classification accuracies above 90% were achieved. To limit the number of fully connected layers during the evolutionary process, the first level of the mutation operator was modified. At the first level of the mutation operator, if $U_1(0,1) > 0.5$, a new block is added, and if $U_2(0,1) > 0.5$, the added block is a fully connected layer; then, the operator was modified so that if $U_2(0,1) > 0.5$, no block is added. This modification is shown in line 4 of Algorithm 2, which shows the mutation operator of DeepGA. This ensures that, during the whole evolutionary process, the generated networks only have two fully connected layers, allowing the algorithm's search to focus on the blocks of convolutional layers since they would be in charge of generating the feature vectors with the desired behavior.

To access the feature vectors generated using the CNNs, the *CNN class* of DeepGA, which builds the model for training and testing, was modified. As output, this class only generated the probabilities of the images belonging to the different classes. The modification consisted of the addition of the flattened outputs of the convolutional block (feature vectors) to the original output to be able to access them in the next process (i.e., to distinguish objects from new classes and simultaneously discover these classes).

The last modification to the DeepGA algorithm was an improvement in image reading. The PyTorch ImageFolder function was used to be able to read the images of all classes in a single process instead of reading the images of each class individually.

Algorithm 1: DeepGA pseudocode.

Input: A population P of N individuals. The number of generations T,
crossover rate $CXPB$, mutation rate $MUPB$, tournament size $TSIZE$.
Output:
Initialize population (training the networks).
$t \leftarrow 1$
while $t \leq T$ **do**
  Select $N/2$ parents with probabilistic tournament selection
  Offs $\leftarrow$ {}
  while |Offs| $< N/2$ do
    Select two random parents $p1$ and $p2$.
    if random(0,1) $\leq CXPB$ **then**
      $O1$, $O2 \leftarrow$ Crossover($p1$, $p2$) // Crossover
      if random(0,1) $\leq MUPB$ **then**
        Mutation($O1$, $O2$) **// *Mutation (modified)***
      fitness($O1$, $O2$) (Equation (1)) **// *Evaluation (modified)***
    $P \leftarrow P \cup$ Offs
    Select the best N individuals in P as survivals.
  end
end

Algorithm 2: Mutation process DeepGA.

if random(0,1) $\leq MUPB$ then
  if random(0,1) $\leq$ U1 then // Adding a new block
    if random(0,1) $\leq U2$ then
      A convolutional block is added **// *Removed***
    else
      A fully connected block is added
  else // Restarting a block
    if random(0,1) $\leq$ W1 then
      Restarting a convolutional block
    else
      Restarting a fully connected block

3.3. Neuroevolved CNN

The CNN architecture with the best fitness generated using DeepGA and SupCon was split to fulfill two purposes. First, the goal was to train the convolutional block with the contrastive loss function and the fully connected block with the cross-entropy loss function, using the full test set described in Section 3.1 (200 images of each of the five "*known classes*"), and to perform a classification accuracy test momentarily assuming a closed-set environment to validate that a good classification accuracy could be obtained since it is an essential point for OSR. Second, we sought to have the feature extraction process and the classification process separate since the detection of new class objects and the discovery of their classes must be accomplished between these events.

3.4. Gaussian Mixture Model (GMM) and Clustering

The main objective of this work is to approach VMMR as an OSR problem with an extension for the discovery of new classes. To achieve this, we divided our strategy into two phases, both relying on the consistent distribution of feature vectors in the embedded space generated using DeepGA and SupCon.

The first phase consisted of extracting the feature vectors from the images used to train the CNN and validating its classification accuracy. The feature vectors were compressed using principal component analysis (PCA) where the second and third components, which contributed 27.81 and 20.97 to the percentage of variance, respectively, were selected to perform a linear regression on the original feature vectors to obtain their projections. As

mentioned in Section 3.1, with the same proportion of data with which the CNN was trained and tested (80–20%), the 2D projections of the feature vectors were used to model each "*known class*" with a Gaussian mixture model (GMM) and define a recognition threshold of "*known classes*". In the test stage, where objects of both known and unknown classes were included, the GMM divided the objects as a group of unknown classes that did not pass the threshold and subsets of known classes whose probabilities matched the "*known class*" models. The above only served as a partial guide in the recognition of new class objects since, in the second phase of the strategy, a multiobjective clustering algorithm with automatic determination of the number of clusters (MOCKs) was employed and optimized with a multiobjective evolutionary algorithm (MOEA), called NSGA-II [40].

In the second phase, the clustering algorithm grouped the feature vectors extracted using the domain-specific CNN without any modification in their dimensionality. Since the GMM can determine the objects of known classes and their respective classes with some confidence, due to the threshold, we compared the subgroups of known classes generated using the GMM with the solutions of MOCK/NSGA-II to select the individual from the population with the highest similarity, where different criteria were used. First, the solutions that grouped the instances that the GMM determined as known and were in the same structures (subgroups) as the GMM had a higher score (one point for each shared structure). Although all the solutions of the clustering algorithm were optimal for the problem, we selected the solution that had the highest score (higher match with the GMM in the known classes) and was closest to the knee point as the "best solution".

Finally, we determined which clusters of the "best solution" contain known objects and separated them from the clusters containing unknown objects in a similar way to how the solutions were scored. Then, since the GMM also detected the objects of unknown classes (the objects that did not pass the threshold) with some confidence, those clusters that only contained objects that the GMM determined as unknown were automatically determined as new classes. After these processes, if there were still undetermined clusters as known or unknown, the number of known and unknown instances within the undetermined clusters were counted (according to the GMM determination), and the clusters were defined in the same category as that containing the majority of instances or as unknown if it contained the same number of examples to try to mitigate the *open-set risk*.

At the end of this strategy, the objects of the clusters that were determined as known were entered into the CNN's fully connected block to be classified, and the clusters that were determined to be unknown were the newly discovered classes of the objects detected as unknown.

The original version of the MOCK algorithm was proposed in 2004 by Handl et al. [41] and employed the MOEA called PESA-II. In 2016, Martinez-Peñaloza et al. [42] managed to improve the results by using the MOEA NSGA-II instead of PESA-II. In the MOCK version improved with NSGA-II, individuals are ranked and sorted according to their non-dominated level, and a crowding distance is used to perform niching. This distance is calculated for each member to be used by the selection operator to maintain a diverse front by ensuring that each member stays a crowding distance apart. Algorithm 3 shows NSGA-II's pseudocode.

Algorithm 3: NSGA-II pseudocode.

```
Initialize Population
Generate random population -size M
Evaluate Objective values
Assign Rank (level) Based on Pareto Dominance -"sort"
Generate Child Population
Binary Tournament Selection
Recombination and Mutation
for i = 1 to Number of Generations do
    for each Parent and Child in Population do
        Assign Rank (level) Based on Pareto –"sort"
        Generate sets of non-dominated fronts
        Loop (inside) by adding solutions to next generation starting
        from the "first" front until M individuals found determine
        crowding distance between points on each front
    end
    Select points (elitist) on the lower front (with lower rank) and
    are outside a crowding distance. Create next generation
    Binary Tournament Selection
    Recombination and Mutation
end
```

4. Experiments and Results

This section describes the experiments and results obtained from our proposal to approach VMMR as an OSR problem with an extension for new class discovery.

For the neuroevolution process of CNNs performed with DeepGA [33] and SupCon [35], as mentioned in Section 3.1, six images of each "*known class*" taken from the VMMRdb [37] database were used.

The parameters described in Table 1 were used to initialize the population. Due to time constraints, it was not possible to use a parameter calibration program. Then, the parameters for the evolutionary process were calibrated manually. The parameters with which the best results were obtained, and which were used to generate the CNNs are shown in Table 2. The different values that each hyperparameter could have during the evolutionary process were the same as those established by the author of DeepGA and are presented in Table 3.

Table 1. Parameters for the population initialization.

Parameter	Values
Min number of convolutional layers	4
Max number of convolutional layers	9
Min number of fully connected layers	1
Max number of fully connected layers	1

Table 2. Parameters for the evolutionary process.

Parameter	Values
Population size	20
No. of generations	100
Tournament size	5
Crossover rate	0.7
Mutation rate	0.7
No. of epochs per individual	20

Table 3. Values that each hyperparameter could have during the evolutionary process.

Hyperparameter	Values
No. of filters *	{2, 4, 8, 16, 32}
Filter size *	{2, 3, 4, 5, 6, 7, 8}
Pooling type *	{Max, Avg}
Pooling size *	{2, 3, 4, 5}
No. of neurons	{4, 8, 16, 32, 64, 128}

Rows marked with * correspond to the convolutional block, while the unmarked row corresponds to the fully connected block.

Seven executions of the NE process with DeepGA and SupCon were performed. Figure 2 shows the convergence curves of the seven executions and a short analysis of the fitness values obtained in each one. As can be seen, all the executions started with a fitness within a range of 3.3 and 3.5, and most reached premature convergence or stalled at local optima. However, one of the executions (marked in red) achieved a more accurate search space that led to a fitness value of 1.96096.

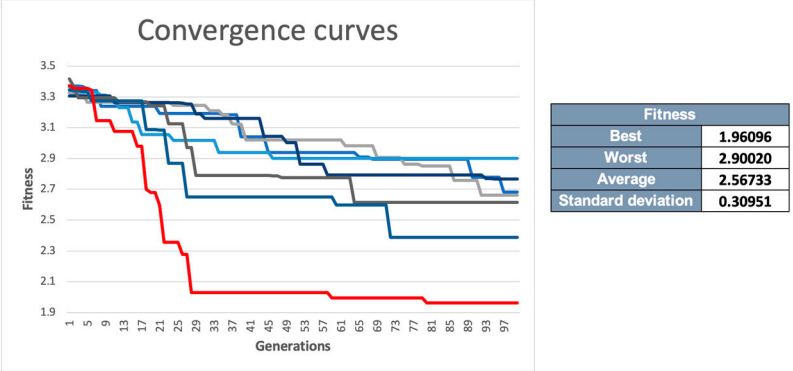

Figure 2. Convergence curves and the most relevant data of the fitness values of the seven runs of the NE process.

The CNN with the best fitness obtained in the NE process (henceforth referred to as the "domain-specific CNN") had a value of 1.96096, which was the value of SupCon in the last training epoch of the CNN (its justification is explained in Section 3.2 in more detail) and took 7 h to execute in the Visual Studio Code editor running on a MacBook Pro with a 2.2 GHz Quad-Core Intel Core i7 processor with 16 GB 1600 MHz DDR3 of memory. Figure 3 illustrates the architecture in terms of its encoding. In the first level, it can be observed that the architecture has 13 convolutional blocks (each one consisting of a single convolutional layer) and 2 fully connected blocks (each one comprising a single layer and a fully connected layer). The last convolutional block/layer generates feature vectors of 288 features. At the second level, the binary string defines the connectivity between convolutional blocks. Each bit represents the connectivity of a previous non-consecutive layer, starting from the third block. For a better understanding, we will explain three examples to understand the connections. The third convolutional block (first bit marked in red) can only have connections with previous blocks that are not its immediately previous consecutive block, so the third block cannot have a connection with the second convolutional block, but it can with the first one, which is why only one bit is assigned to it, and the bit value is 1. This means that there is a connection, which is represented by the red line on the first level. The next two bits (green) are for the fourth block, which can have a connection with the first or second block, and since the bit values are 1, both connections exist (represented by the green lines on the first level). A different case is shown in the next three bits (highlighted in yellow) assigned to the fifth block, which can have connections with the first, second, and third blocks; however, of those three bits, only the second one has a value of 1, which means that the fifth block only connects to the second block.

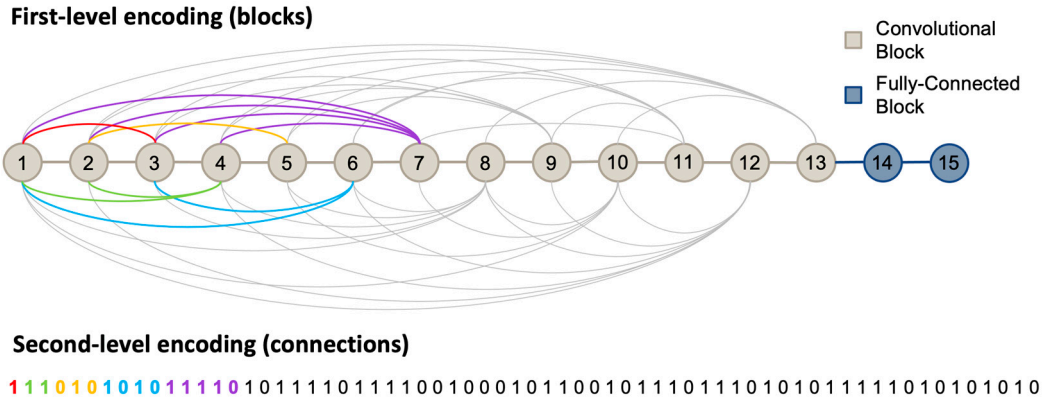

Figure 3. CNN architecture with the best fitness obtained using DeepGA and SupCon. The first level (blocks) represents simple convolutional operations instead of a set of convolutional layers. The second level (binary string) determines the skip connections received from the third block onward. Each bit represents the connectivity from previous layers, from the third layer onward.

To verify that the domain-specific CNN could generate the feature vectors extracted spatially close in terms of cosine similarity if they belong to the same class and far apart if they belong to different classes, a distance matrix using cosine similarity as the metric was generated with the feature vectors obtained in the last training epoch of the domain-specific CNN. On the same feature vectors, the t-SNE [43] technique was used to reduce the dimensionality from 288 to 2 in order to visualize them in a two-dimensional plane. The results of the distance matrix and t-SNE are shown in Figure 4. By means of these two techniques, it could be seen that the desired behavior in the feature vectors was achieved.

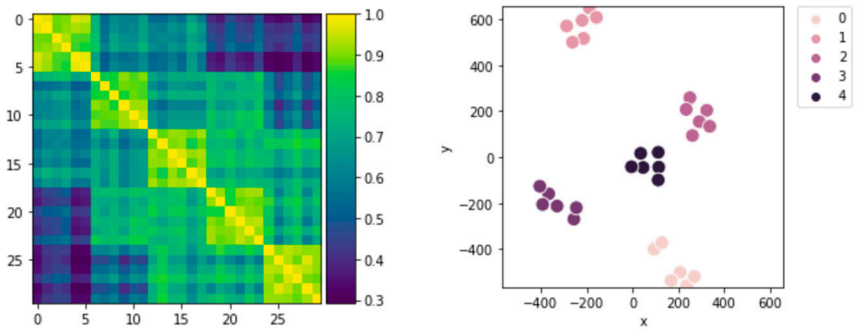

Figure 4. Distance matrix (with cosine similarity) and the projection in a two-dimensional plane of the feature vectors of the last training epoch of the domain-specific CNN.

As mentioned in Section 3.3, the domain-specific CNN was split to train the convolutional block with the contrastive loss function and the fully connected block with the cross-entropy loss function. For the training process, 1000 images of rear views of vehicles of the five "*known classes*" (200 images of each class) were used. A classification accuracy test was performed using 250 images of rear views of vehicles of the five "*known classes*" (50 images of each class) momentarily assuming a closed-set environment to validate that good classification accuracy was being achieved since it is an essential point for OSR. A 90% classification accuracy was reached during this test; more details regarding the data used are presented in Section 3.1.

The next test was to verify that the domain-specific CNN could generate feature vectors spatially close in terms of cosine similarity if they belong to the same class and far apart if they belong to different classes. This behavior was maintained in objects of unknown classes since the detection of objects of new classes and the discovery of their classes depended on this behavior. For this, the testing images, both the nine testing images of "*unknown classes*" shown in Figure 5 on the right and the fifteen images of the

five "*known classes*" shown in Figure 5 on the left, were entered into the convolutional block of the domain-specific CNN to extract their feature vectors. To visualize the results, which are shown in Figure 6, a distance matrix using cosine similarity as the metric was generated, and a two-dimensional projection was performed using linear regression with the two components described in Section 3.4. Figure 6 shows that the domain-specific CNN managed to generate feature vectors close in terms of cosine similarity if they belonged to the same class and distant if they belonged to different classes and managed to maintain such behavior even in objects of "*unknown classes*".

Figure 5. Sample images from the VMMRdb database. Both "*known*" (left) and "*unknown*" (right) classes were used during testing.

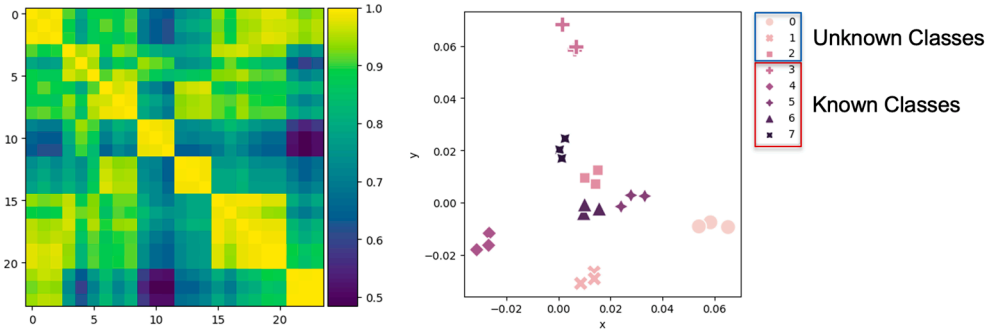

Figure 6. Distance matrix (with cosine similarity) and the projection in a two-dimensional plane of the feature vectors of the samples shown in Figure 5.

Later, the feature vectors of the images used to train the domain-specific CNN and validate its classification accuracy were compressed to two dimensions, and a linear regression was performed on these feature vectors to obtain their projections using the two components described in Section 3.4. With the projections of the 1000 images used to train the domain-specific CNN, we modeled the "*known Classes*" using a GMM, and the distribution of the Gaussians is shown in Figure 7. We then defined a "*known class*" recognition threshold within the GMM with a value of 9.999, using the projections of the 250 images that were used in the domain-specific CNN classification accuracy test.

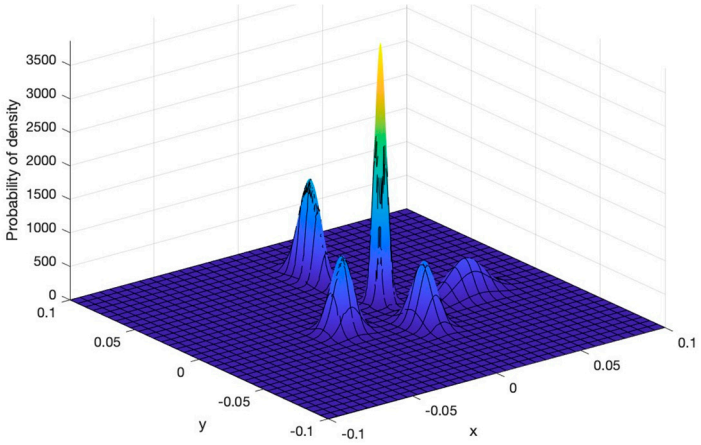

Figure 7. Distribution of the "known classes" using GMM.

Finally, the strategy proposed in Section 3.4 was carried out to detect the objects of new classes and discover their classes. The first step was to enter the two-dimensional projections used in Figure 6, which contain objects of both known and unknown classes (Figure 5), into the GMM to obtain their density probabilities. As its output, the model provided the probability of each object belonging to the known classes, and the threshold allowed us to set a probability limit for known or unknown classes. As can be seen in Table 4, the objects of known classes were correctly identified within their classes, and in the case of the objects of unknown classes, it can be seen that with the limit marked by the threshold, eight of the nine objects were correctly identified as unknown. A clearer representation of the results obtained can be seen in Figure 8, which indicates that the GMM divided the objects as a set of unknown classes that did not pass the threshold as well as the subsets of known classes whose probabilities matched the known class models.

Table 4. Probabilities of the test instances obtained with the GMM. The probabilities that exceeded the threshold (9999) are marked in orange.

	x	y	Real Label		Probability (Class 0)	Probability (Class 1)	Probability (Class 2)	Probability (Class 3)	Probability (Class 4)
0	5.84×10^{-2}	-7.59×10^{-3}	0		1.000×10^{0}	8.152×10^{-27}	7.294×10^{-57}	1.650×10^{-89}	1.483×10^{-73}
1	5.40×10^{-2}	-9.16×10^{-3}	0		1.000×10^{0}	4.920×10^{-22}	6.332×10^{-48}	8.417×10^{-77}	1.826×10^{-67}
2	6.52×10^{-2}	-9.41×10^{-3}	0		1.000×10^{0}	1.412×10^{-33}	3.304×10^{-75}	1.581×10^{-112}	1.321×10^{-88}
3	1.37×10^{-2}	-2.67×10^{-2}	1		4.068×10^{-17}	1.000×10^{0}	6.451×10^{-33}	3.341×10^{-22}	8.823×10^{-29}
4	1.38×10^{-2}	-2.92×10^{-2}	1		6.396×10^{-19}	1.000×10^{0}	1.408×10^{-37}	1.556×10^{-23}	2.039×10^{-31}
5	8.56×10^{-3}	-3.10×10^{-2}	1		1.170×10^{-22}	1.000×10^{0}	3.308×10^{-43}	6.810×10^{-24}	3.034×10^{-27}
6	1.43×10^{-2}	7.13×10^{-3}	2		1.020×10^{-4}	3.865×10^{-6}	9.999×10^{-1}	4.394×10^{-9}	2.252×10^{-11}
7	1.52×10^{-2}	1.24×10^{-2}	2		1.451×10^{-4}	9.982×10^{-9}	9.999×10^{-1}	7.854×10^{-8}	1.672×10^{-11}
8	1.02×10^{-2}	9.51×10^{-3}	2	Known	5.701×10^{-5}	4.896×10^{-6}	9.999×10^{-1}	5.240×10^{-7}	1.290×10^{-8}
9	1.53×10^{-3}	6.82×10^{-2}	3		7.202×10^{-36}	3.243×10^{-35}	5.417×10^{-87}	1.000×10^{0}	4.510×10^{-58}
10	6.30×10^{-3}	5.86×10^{-2}	3		3.311×10^{-28}	6.318×10^{-30}	2.282×10^{-61}	1.000×10^{0}	1.830×10^{-42}
11	6.97×10^{-3}	5.96×10^{-2}	3		2.972×10^{-29}	1.191×10^{-30}	1.055×10^{-63}	1.000×10^{0}	1.461×10^{-43}
12	-2.66×10^{-2}	-1.18×10^{-2}	4		2.727×10^{-28}	2.310×10^{-17}	1.384×10^{-65}	1.521×10^{-56}	1.000×10^{0}
13	-2.68×10^{-2}	-1.64×10^{-2}	4		8.075×10^{-32}	2.380×10^{-17}	1.771×10^{-72}	2.738×10^{-59}	1.000×10^{0}
14	-3.14×10^{-2}	-1.81×10^{-2}	4		3.329×10^{-36}	3.051×10^{-21}	3.378×10^{-87}	2.106×10^{-71}	1.000×10^{0}
15	2.79×10^{-2}	2.76×10^{-3}	5		9.995×10^{-1}	2.116×10^{-6}	5.257×10^{-4}	6.300×10^{-19}	3.457×10^{-21}
16	3.32×10^{-2}	2.46×10^{-3}	5		1.000×10^{0}	2.820×10^{-9}	3.713×10^{-9}	7.638×10^{-27}	2.355×10^{-27}
17	2.41×10^{-2}	-1.54×10^{-3}	5		9.779×10^{-1}	4.606×10^{-3}	1.751×10^{-2}	2.796×10^{-15}	1.285×10^{-18}
18	9.74×10^{-3}	-4.05×10^{-3}	6		4.675×10^{-5}	9.986×10^{-1}	1.350×10^{-3}	2.120×10^{-9}	2.322×10^{-8}
19	1.57×10^{-2}	-2.33×10^{-3}	6		1.208×10^{-2}	4.648×10^{-1}	5.231×10^{-1}	9.067×10^{-10}	1.078×10^{-11}
20	1.01×10^{-2}	-7.46×10^{-4}	6		9.939×10^{-4}	5.405×10^{-1}	4.585×10^{-1}	7.914×10^{-8}	2.899×10^{-7}
21	4.78×10^{-4}	2.02×10^{-2}	7		5.100×10^{-2}	1.262×10^{-4}	1.164×10^{-2}	9.322×10^{-1}	5.056×10^{-3}
22	1.33×10^{-3}	1.69×10^{-2}	7		1.022×10^{-1}	1.459×10^{-3}	6.103×10^{-1}	2.326×10^{-1}	5.334×10^{-2}
23	2.57×10^{-3}	2.45×10^{-2}	7		2.779×10^{-4}	8.184×10^{-8}	1.406×10^{-5}	9.997×10^{-1}	4.287×10^{-7}

Known 5 Groups

[0,1,2,16] [3,4,5] [6,7,8] [9,10,11] [12,13,14]

Unknown 1 Group

[15,17,18,19,20,21,22,23]

Figure 8. Instances grouped according to GMM probabilities and the threshold.

As previously mentioned, the GMM results were the first phase of the strategy and only served as a partial guide in the recognition of objects of unknown classes. In the second phase of the strategy, a clustering algorithm called MOCK was used, which was enhanced with NSGA-II. For the second phase, the 24 feature vectors without projection (288 features) were entered into the clustering algorithm. For execution, the algorithm was run with the parameters shown in Table 5.

Table 5. Parameters for the clustering algorithm MOCK/NSGA-II.

Parameter	Values
Population size (M)	9
Nearest neighbors (L)	2
Number of generations	10

The nine final individuals of the clustering process are shown in Figure 9 in terms of their fitness values, and Figure 10 shows how the vectors were grouped in different structures.

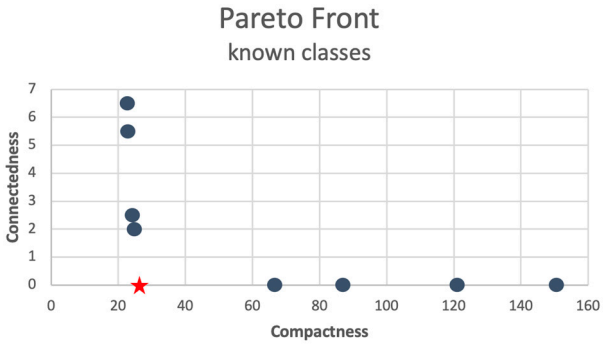

Figure 9. The Pareto front generated using the MOCK clustering algorithm improved with NSGA-II. The individual marked with a red star is the knee point.

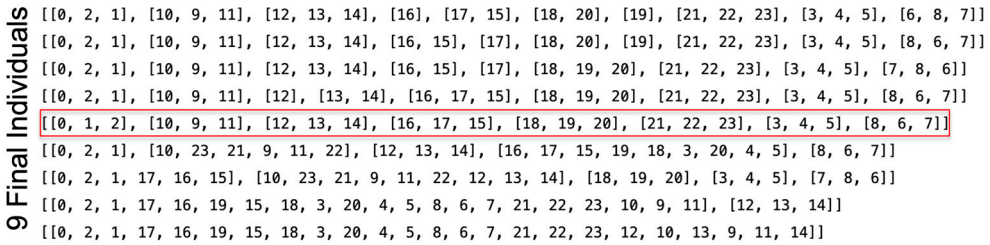

Figure 10. The final nine individuals generated using the MOCK clustering algorithm improved with NSGA-II. The individual marked in red is the knee point.

Subsequently, the comparison described in Section 3.4 was performed to select the "best solution". The individuals generated using the MOCK/NSGA-II algorithm (Figure 10) and the known class subgroups generated using the GMM (Figure 8) were compared. The results of this comparison are shown in Table 6. It can be seen that Solutions 1, 2, 3, and 5 have four structures shared with the subgroups of known classes generated with the GMM. However, since the solution closest to the knee point was selected as the "best solution",

Solution 5 was chosen (marked with *), which in this case, was found to be the knee point, marked in red in Figure 10.

Table 6. Scores obtained from the comparison of the individuals generated using MOCK/NSGA-II and the known class subgroups generated using GMM. The solution marked with * was selected as the "best solution".

	Pareto Frontier Position	Score
Solution 1	1	4
Solution 2	2	4
Solution 3	3	4
Solution 4	4	3
Solution 5 *	**5**	**4**
Solution 6	6	2
Solution 7	7	2
Solution 8	8	1
Solution 9	9	0

Given the "best solution", the four clusters with shared structures with the known class subgroups generated with the GMM were determined as "*known classes*". Then, the clusters containing only objects that the GMM determined as unknown, shown in Figure 8, were determined as "*new classes*". The result of these processes can be seen in Figure 11.

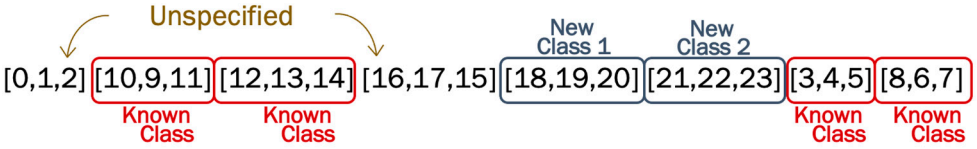

Figure 11. Selection of "*known classes*" and "*new classes*".

Since there were still unspecified clusters as *known* or *new*, we counted the number of known and unknown instances (as determined using the GMM) in the indeterminate clusters and defined the clusters in the same category as that comprising the majority of instances. Thus, we obtained five groups of "*known classes*" and three "*new classes*", as shown in Figure 12. Given the data in Table 4, we can confirm that indeed the vectors of the "new classes" corresponded to the instances of unknown objects and that they were grouped in the same structure as their "unknown class", thus confirming that both the "new classes" of objects of "unknown classes" can indeed be discovered.

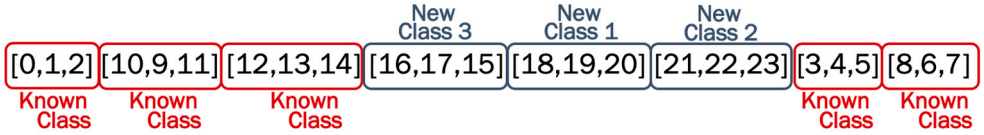

Figure 12. Final selection of "*known classes*" and "*new classes*".

Finally, the objects of known classes were entered into the classification section of the domain-specific CNN where a classification accuracy of 100% was obtained. Given the classification results obtained, we calculated the critical values of true positive (TP), false positive (FP), and false negative (FN) of both known and unknown classes. Subsequently, we calculated the micro-F1 score since it is one of the most commonly used metrics in OSR algorithms. The results obtained are shown in Table 7.

Table 7. Calculation of the micro-F1 score.

	Label	True Positive (TP)	False Positive (FP)	False Negative (FN)	Micro-F1
Known Classes	Chevrolet Silverado 2004	3	0	0	Precision = 1.0
	Ford Explorer 2002	3	0	0	
	Ford Mustang 2000	3	0	0	
	Honda Civic 2002	3	0	0	Recall = 1.0
	Nissan Altima 2005	3	0	0	
Unknown Classes	Unknown Class 1	3	0	0	
	Unknown Class 2	3	0	0	Micro-F1 Score = 1.0
	Unknown Class 3	3	0	0	
	Total	24	0	0	

5. Discussion and Conclusions

The main contribution of this work is to present a strategy to approach the VMMR as an OSR problem that is extended to the discovery of new classes, taking the distribution of feature vectors generated using a domain-specific CNN as the main guideline. This work seeks to highlight the importance of generating domain-specific OSR strategies and the need to apply them to real-world classification/recognition problems such as VMMR in order to obtain classifiers that are not only more accurate but also more robust, as they are prepared to face real-life scenarios. Although we focused on VMMR, the proposed methodology can be used as a benchmark for future domain-specific OSR problems and can be applied to other domains like handwritten digit recognition, chest X-ray classification, etc.

For the development of this work, we considered four main objectives to fulfill the purpose of approaching VMMR as an OSR problem extended for new class discovery. The fulfillment of our first objective could be validated with the results shown in Figure 6, where it can be seen that the CNN designed through the NE process with contrastive loss managed to map within the embedded space the feature vectors close in terms of cosine distance if they belonged to the same classes and far away if they belonged to different classes, maintaining this behavior for both known and unknown classes.

The second objective was described in detail in Section 3.4, which is the theoretical part of the third objective. In the Section 4, the proposed methodology was described step by step, and the experiments carried out validated that the proposed mechanism is able to detect objects of unknown classes and simultaneously discover their classes. One point to highlight is that our strategy is not restricted by training data, as it can be adjusted as these data change. More precisely, by using contrastive learning to train the feature extraction of the domain-specific CNN, the distribution of feature vectors is not only guided by "known classes" but is able to perform a consistent mapping even for objects of "unknown classes", which allows us to effectively detect objects of known classes and discover their classes simultaneously.

From the outset, we decided to employ a CNN not only to exploit the powerful ability of CNNs to extract meaningful features but also because these networks are known to be powerful classifiers. Therefore, since our domain-specific CNN was trained with numerous well-labeled examples, we could rely on its accuracy in classifying instances of known classes. Therefore, the last objective was met by achieving 100% classification accuracy of the images of the known classes in the test set.

Overall, the entire algorithm achieved a micro-F1 score of 1.00 by accurately classifying instances of known classes and effectively discovering the classes of instances whose classes were not included in the training. In a closed-set context, which is where most classification algorithms are developed, all instances of unknown classes would have been classified into some known class, so the model would not have been able to achieve a classification accuracy higher than 62.5% with the test set used in this work since 9 of the 24 test images belonged to unknown classes. The poor classification accuracy in this specific context, which simulates a real-life scenario, would be due to the incomplete knowledge of the world and not due to the classification potential that the classifier could achieve. Therefore, in this work, we proposed to add a mechanism to one of the most used image classifiers such as CNNs in order to detect objects of unknown classes and identify these classes. This highlights the possibility to expand the classification potential of CNNs and increase their robustness to work more effectively in real-life scenarios, thus enabling these classifiers not only to react to queries but also to continue learning even after being trained.

One of the limitations of this work was that due to time constraints, the neuroevolution algorithm was executed only seven times with the specified parameters, and it is left as future work to create a statistically more representative sample of executions and use a parameter calibration algorithm to possibly have better and more efficient results. It is also left as future work to increase the number of "known classes" to be able to classify more models with the domain-specific CNN and apply other OSR strategies to the VMMR problem for a more representative comparison.

Author Contributions: Conceptualization, D.-I.V.-S., H.-G.A.-M. and E.M.-M.; methodology, D.-I.V.-S., H.-G.A.-M. and E.M.-M.; software, D.-I.V.-S.; validation, H.-G.A.-M. and E.M.-M.; formal analysis, D.-I.V.-S.; investigation, D.-I.V.-S.; resources, D.-I.V.-S., H.-G.A.-M. and E.M.-M.; data curation, D.-I.V.-S.; writing—original draft preparation, D.-I.V.-S.; writing—review and editing, D.-I.V.-S., H.-G.A.-M. and E.M.-M.; visualization, D.-I.V.-S.; supervision, H.-G.A.-M. and E.M.-M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: The first author would like to thank the Consejo Nacional de Ciencia y Tecnología (CONACYT), an institution of the Government of Mexico, for the financial support provided through the "Beca Nacional" with CVU 1141251 as part of the Programa de Becas para Estudios de Posgrado.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Naseer, S.; Shah, S.M.A.; Aziz, S.; Khan, M.U.; Iqtidar, K. Vehicle Make and Model Recognition using Deep Transfer Learning and Support Vector Machines. In Proceedings of the IEEE 23rd International Multitopic Conference (INMIC), Bahawalpur, Pakistan, 5–7 November 2020; pp. 1–6. [CrossRef]
2. Agarwal, A.; Shinde, S.; Mohite, S.; Jadhav, S. Vehicle Characteristic Recognition by Appearance: Computer Vision Methods for Vehicle Make, Color, and License Plate Classification. In Proceedings of the IEEE Pune Section International Conference (PuneCon), Pune, India, 15–17 December 2022; pp. 1–6. [CrossRef]
3. Nazemi, A.; Azimifar, Z.; Shafiee, M.J.; Wong, A. Real-Time Vehicle Make and Model Recognition Using Unsupervised Feature Learning. *IEEE Trans. Intell. Transp. Syst.* **2020**, *21*, 3080–3090. [CrossRef]
4. Hassaballah, M.; Kenk, M.A.; Muhammad, K.; Minaee, S. Vehicle Detection and Tracking in Adverse Weather Using a Deep Learning Framework. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 4230–4242. [CrossRef]
5. Hussain, K.F.; Afifi, M.; Moussa, G. A Comprehensive Study of the Effect of Spatial Resolution and Color of Digital Images on Vehicle Classification. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 1181–1190. [CrossRef]
6. Boukerche, A.; Ma, X. A Novel Smart Lightweight Visual Attention Model for Fine-Grained Vehicle Recognition. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 13846–13862. [CrossRef]
7. Fang, J.; Zhou, Y.; Yu, Y.; Du, S. Fine-Grained Vehicle Model Recognition Using A Coarse-to-Fine Convolutional Neural Network Architecture. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 1782–1792. [CrossRef]
8. Masana, M.; Liu, X.; Twardowski, B.; Menta, M.; Bagdanov, A.D.; van de Weijer, J. Class-Incremental Learning: Survey and Performance Evaluation on Image Classification. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 5513–5533. [CrossRef]

9. Hafeez, M.A.; Ul-Hasan, A.; Shafait, F. Incremental Learning of Object Detector with Limited Training Data. In Proceedings of the Digital Image Computing: Techniques and Applications (DICTA), Gold Coast, Australia, 29 November–1 December 2021; pp. 1–8. [CrossRef]
10. Zhang, F. Learning Unsupervised Side Information for Zero-Shot Learning. In Proceedings of the International Conference on Signal Processing and Machine Learning (CONF-SPML), Stanford, CA, USA, 14 November 2021; pp. 325–328. [CrossRef]
11. Li, Y.; Kong, D.; Zhang, Y.; Chen, R.; Chen, J. Representation Learning of Remote Sensing Knowledge Graph for Zero-Shot Remote Sensing Image Scene Classification. In Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Brussels, Belgium, 11–16 July 2021; pp. 1351–1354. [CrossRef]
12. Kezebou, L.; Oludare, V.; Panetta, K.; Agaian, S. Few-Shots Learning for Fine-Grained Vehicle Model Recognition. In Proceedings of the IEEE International Symposium on Technologies for Homeland Security (HST), Boston, MA, USA, 8–9 November 2021; pp. 1–9. [CrossRef]
13. Zhou, F.; Zhang, L.; Wei, W.; Bai, Z.; Zhang, Y. Meta Transfer Learning for Few-Shot Hyperspectral Image Classification. In Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Brussels, Belgium, 11–16 July 2021; pp. 3681–3684. [CrossRef]
14. Scheirer, W.J.; de Rezende Rocha, A.; Sapkota, A.; Boult, T.E. Toward Open Set Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 1757–1772. [CrossRef]
15. Scheirer, W.J.; Jain, L.P.; Boult, T.E. Probability Models for Open Set Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2014**, *36*, 2317–2324. [CrossRef]
16. Rudd, E.M.; Jain, L.P.; Scheirer, W.J.; Boult, T.E. The Extreme Value Machine. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 762–768. [CrossRef]
17. Ribeiro Mendes Júnior, P.; Boult, T.E.; Wainer, J.; Rocha, A. Open-Set Support Vector Machines. *IEEE Trans. Syst. Man Cybern. Syst.* **2022**, *52*, 3785–3798. [CrossRef]
18. Alfarisy, G.A.F.; Malik, O.A.; Hong, O.W. Quad-Channel Contrastive Prototype Networks for Open-Set Recognition in Domain-Specific Tasks. *IEEE Access* **2023**, *11*, 48578–48592. [CrossRef]
19. Bendale, A.; Boult, T. Towards Open World Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 1893–1902. [CrossRef]
20. Wang, Z.; Salehi, B.; Gritsenko, A.; Chowdhury, K.; Ioannidis, S.; Dy, J. Open-World Class Discovery with Kernel Networks. In Proceedings of the IEEE International Conference on Data Mining (ICDM), Sorrento, Italy, 17–20 November 2020; pp. 631–640. [CrossRef]
21. Han, K.; Rebuffi, S.-A.; Ehrhardt, S.; Vedaldi, A.; Zisserman, A. AutoNovel: Automatically Discovering and Learning Novel Visual Categories. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 6767–6781. [CrossRef] [PubMed]
22. Han, K.; Vedaldi, A.; Zisserman, A. Learning to Discover Novel Visual Categories via Deep Transfer Clustering. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 8400–8408. [CrossRef]
23. Geng, C.; Chen, S. Collective Decision for Open Set Recognition. *IEEE Trans. Knowl. Data Eng.* **2022**, *34*, 192–204. [CrossRef]
24. Sun, X.; Yang, Z.; Zhang, C.; Ling, K.-V.; Peng, G. Conditional Gaussian Distribution Learning for Open Set Recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 13477–13486. [CrossRef]
25. Ye, W.; Liu, R.; Li, Y.; Jiao, L. Quantum-inspired evolutionary algorithm for convolutional neural networks architecture search. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, 19–24 July 2020; pp. 1–8. [CrossRef]
26. Liu, J.; Zhou, S.; Wu, Y.; Chen, K.; Ouyang, W.; Xu, D. Block proposal neural architecture search. *IEEE Trans. Image Process.* **2021**, *30*, 15–25. [CrossRef] [PubMed]
27. Zhou, Y.; Gen, G.G.; Yi, Z. A knee-guided evolutionary algorithm for compressing deep neural networks. *IEEE Trans. Cybern.* **2021**, *51*, 1626–1638. [CrossRef]
28. Operiano, K.R.G.; Iba, H.; Pora, W. Neuroevolution architecture backbone for x−ray object detection. In Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI), Canberra, Australia, 1–4 December 2020; pp. 2296–2303. [CrossRef]
29. Hassanzadeh, T.; Essam, D.; Sarker, R. 2D to 3D evolutionary deep convolutional neural networks for medical image segmentation. *IEEE Trans. Med. Imaging* **2021**, *40*, 712–721. [CrossRef]
30. Sun, Y.; Xue, B.; Zhang, M.; Yen, G.G. Evolving deep convolutional neural networks for image classification. *IEEE Trans. Evol. Comput.* **2020**, *24*, 394–407. [CrossRef]
31. Zhang, H.; Jin, Y.; Cheng, R.; Hao, K. Efficient evolutionary search of attention convolutional networks via sampled training and node inheritance. *IEEE Trans. Evol. Comput.* **2021**, *25*, 371–385. [CrossRef]
32. Hu, M.; Wang, W.; Liu, L.; Liu, Y. Apenas: An asynchronous parallel evolution based multi-objective neural architecture search. In Proceedings of the IEEE International Conference on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), Exeter, UK, 17–19 December 2020; pp. 153–159. [CrossRef]
33. Vargas-Hákim, G.; Mezura-Montes, E.; Acosta-Mesa, H. Hybrid encodings for neuroevolution of convolutional neural networks: A case study. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO'21), Online, 10–14 July 2021; Association for Computing Machinery: New York, NY, USA, 2021; pp. 1762–1770. [CrossRef]

34. Vargas-Hákim, G.-A.; Mezura-Montes, E.; Acosta-Mesa, H.-G. A Review on Convolutional Neural Network Encodings for Neuroevolution. *IEEE Trans. Evol. Comput.* **2022**, *26*, 12–27. [CrossRef]
35. Khosla, P.; Teterwak, P.; Wang, C.; Sarna, A.; Tian, Y.; Isola, P.; Maschinot, A.; Liu, C.; Krishnan, D. Supervised contrastive learning. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 18661–18673. [CrossRef]
36. Chen, T.; Kornblith, S.; Norouzi, M.; Hinton, G. A simple framework for contrastive learning of visual representations. In Proceedings of the 37th International Conference on Machine Learning (ICML'20), Online, 13–18 July 2020; pp. 1597–1607.
37. Tafazzoli, F.; Frigui, H.; Nishiyama, K. A Large and Diverse Dataset for Improved Vehicle Make and Model. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, USA, 21–26 July 2017; pp. 874–881. [CrossRef]
38. Hassan, A.; Ali, M.; Durrani, N.M.; Tahir, M.A. An Empirical Analysis of Deep Learning Architectures for Vehicle Make and Model Recognition. *IEEE Access* **2021**, *9*, 91487–91499. [CrossRef]
39. Kristiani, E.; Yang, C.-T.; Huang, C.-Y. iSEC: An Optimized Deep Learning Model for Image Classification on Edge Computing. *IEEE Access* **2020**, *8*, 27267–27276. [CrossRef]
40. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
41. Handl, J.; Knowles, J. An Evolutionary Approach to Multiobjective Clustering. *IEEE Trans. Evol. Comput.* **2007**, *11*, 56–76. [CrossRef]
42. Martínez-Peñaloza, M.; Mezura-Montes, E.; Cruz-Ramírez, N.; Acosta-Mesa, H.; Ríos-Figueroa, H. Improved multi-objective clustering with automatic determination of the number of clusters. *Neural Comput. Appl.* **2017**, *28*, 2255–2275. [CrossRef]
43. Van der Maaten, L.; Hinton, G. Visualizing Data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.

Artículo presentado en el 4to Workshop en New Trends in Computational Intelligence and Applications (CIAPP 2022) y publicado en los workshop proceedings del MICAI 2022

Vehicle make and model recognition with generation of new classes using clustering techniques

Diana-Itzel Vázquez-Santiago[1], Héctor-Gabriel Acosta-Mesa[1], and Efrén Mezura-Montes[1]

[1] Artificial Intelligence Research Institute, Universidad Veracruzana, Veracruz 91097, Mexico

zs21000454@estudiantes.uv.mx,
{heacosta, emezura}@uv.mx

Abstract. One of the main problems faced by supervised learning classification algorithms is scalability. No matter how good their classification accuracy is, they are not able to classify objects for which they were not trained. In this paper we propose a solution to this problem specifically aimed at vehicle make and model recognition. We used a Convolutional Neural Network (CNN) for classification and feature extraction, addressing the scalability problem by using two clustering techniques: K-means and MOCK. For the generation of new classes, we used the feature vectors extracted by the CNN of the images that do not belong to any of the classes with which the model was trained. The results showed that with the learning generated by a CNN it is possible to generate feature vectors with similarities for objects of the same class even if the network was not trained to classify them, which made it possible to generate new classes using unsupervised learning such as clustering.

Keywords: Scalability, CNN, Clustering, MOCK.

1 Introduction

Automatic vehicle make and model recognition aims to offer innovative services to improve the efficiency and safety of transportation networks. Some of these services are intelligent traffic analysis and management, electronic toll collection, emergency vehicle notifications, automatic enforcement of traffic rules, etc. The main problem, in the specific case of this application domain, is that most of the applied approaches for vehicle make and model recognition require large amounts of data to correctly train a model. It is estimated that there are currently more than 3,300 vehicle makes in the world which have added and removed models from the market, modifying the design in each generation and producing different versions of the same vehicle which has made impossible to have a database containing all existing vehicles. This limitation leaves us with a scalability problem that results in vehicles that cannot be classified correctly because the algorithms were not trained to recognize them. In this work, we propose a

feasibly solution to the scalability problem of classification algorithms that use supervised learning by using clustering algorithms to generate new classes using the feature vectors extracted by our classification algorithm. In the state of the art, the scalability problem has been addressed by authors such as Nazemi et al [1] from an anomaly detection approach. Their base system is capable of classify 50 specific vehicle models, to which they added an anomaly detection to identify vehicles that do not belong to any of the 50 classes, to subsequently classify them based on their dimensions within 2 new classes: "Unknown heavy" and "Unknown light". Other authors such as Kezebou et al [4] proposed a Few-Shots Learning approach requiring between 1 and 20 images for the generation of new classes.

2 Methodology

For this work, the VMMRdb database [2] was used since it is one of the most cited in the specialized literature. With the intention of simplifying the problem for analysis, only five classes were used: Dodge Grand Caravan 2005, Ford Explorer 2002, Ford Mustang 2000, Nissan Altima 2005 and Toyota Camry 2007 to train a CNN whose architecture was proposed in the Microsoft technical documentation library [3] with which training and testing times of 2m24s were achieved with accuracies between 90%-95% in 5 epochs.

Since the main objective of this project is to have an algorithm capable of classifying vehicles even if they do not belong to any predefined class, an algorithm was designed to generate new classes from the clustering of images that do not belong to the classes with which the CNN was trained. The feature vectors of the images (extracted with the CNN) were used to perform clustering using 2 algorithms for comparison purposes.

The first clustering algorithm is K-means. The original version of the algorithm was implemented in Python and in the interests of have an additional comparison, a second version was developed using the sklearn library which implements the K-means clustering algorithm. See section 2.3 for more details.

The second clustering algorithm named MOCK employs a multi-objective evolutionary approach named PESA-II. This algorithm attempts to minimize two objectives that are in conflicting with each other (intra-cluster variation and the number of clusters). The concept of Pareto dominance is used to find a set of different non-dominated clustering solutions that achieve a good trade-off between the two objectives. PESA-II's pseudocode is shown in Algorithm 1. See section 2.4 for more details.

Algorithm 1. PESA-II Pseudocode

```
Initialize a random (internal) population IP
Evaluate each member of IP
Initialize the external population EP to the empty set
repeat
    Incorporate nondominated vectors from IP into EP
    Delete the current contents of IP
```

```
    repeat
      With probability Pc, select two parents from EP, where Pc= Crossover probability
      Produce a single child via crossover
      Mutate the child created in the previous step
      With probability 1 – Pc, select one parent
      Mutate the selected parent to produce a child
  until the population IP is filled;
until termination criteria is met;
Return the members of EP as the result
```

2.1 Image preprocessing

The images went through a few processes to fit the model. The first was to segment the images of each class according to the views they showed (front, rear, and side). Due to the scope of the project, we work only with the rear views of the five classes mentioned in Section 2.

Table 1 shows the volume of images (rear views) available per class. Twenty images from each class were chosen for testing and the rest for training, however, as can be seen in Table 1 the volume of images was low and not balanced to adequately train the CNN. To solve these problems, data augmentation and balancing processes were performed on the training set. After these processes, the final number of images for training per class was 1,000 and the twenty images that were originally selected for testing were kept without data enhancement.

The next process was to reduce the dimensions of the images (training and testing) since the CNN architecture required input dimensions of 32x32x3.
With the processes mentioned above, the classification model achieved accuracies between 80%-85%. To improve recognition accuracy the images were cropped to preserve only the region of interest (ROI), which improve the accuracy by up to 10%.

Table 1. Comparison between data volumes per class.

Class	Images (Rear view)	Training (Before augmentation)	Training (After augmentation)	Test (No augmentation)
Dodge Grand Caravan 2005	164	144	1,000	20
Ford Explorer 2002	234	214	1,000	20
Ford Mustang 2000	216	196	1,000	20
Nissan Altima 2005	294	274	1,000	20
Toyota Camry 2007	170	150	1,000	20

2.2 Convolutional Neural Network (CNN)

This project was implemented in Python mainly because of the access to the PyTorch library which provides support for the development of applications related to machine learning, computer vision, natural language processing, etc. More specifically, it has a

base class for all Convolutional Neural Networks modules, which facilitates its implementation and execution.

Initially in this project, it was proposed to work with well-known CNN architectures such as AlexNet or VGG, however, in the first stages, tests were carried out and execution times were time consuming (30min-50min) achieving a maximum accuracy of 60%. Because of this, we chose to use an architecture found in the Microsoft technical documentation library [3], which can be seen in Fig 1. Originally the network was designed to work with the CIFAR10 database, so the input dimensions were 32x32x3. Even with the possible loss of information, it was decided to keep this architecture and resize the images of the VMMRdb database to fit, since even without the data balance, the cropping of the ROI and with the resizing, accuracies of 70% were achieved. The only modification to the architecture was to change the output of the fully connected layer to five (number of classes).

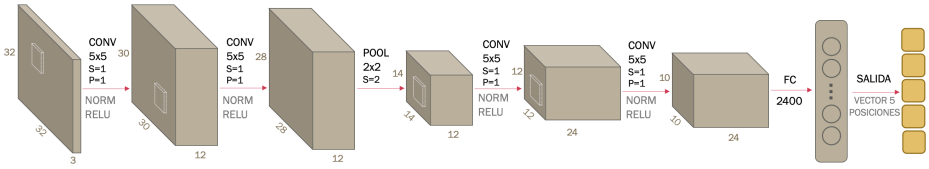

Fig. 1. Architecture of the convolutional neural network used and proposed in the Microsoft technical documentation library [3].

In the training stage, 20-image subsets of the training data were generated and reorganized at each epoch to reduce overfitting. For the update of the network weights during training, the Adam optimizer was used with a learning rate of 0.001 and a weight decrease of 0.0001.

Finally, for the performance evaluation, two indicators were used in each epoch, the first indicator was to evaluate the classification accuracy of the network with the whole test set (20 images as shown in Table 1) and the second indicator was with the Cross Entropy loss function which is mathematically expressed in (1). Where i is the class, c is the number of classes, y_i is the actual class and $\hat{y}_i$ is the predicted class.

$$-\sum_{i=1}^{c} y_i \log \hat{y}_i \tag{1}$$

2.3 K-means Clustering

As mentioned above, the aim of this project is to have an algorithm capable of classifying vehicles even if they do not belong to any predefined class. The first approach implemented for the generation of new classes was K-means clustering. Two implementations of this algorithm were developed. The first was implementing the classical version of the algorithm. The second was using the version developed by sklearn library, initializing the centroids of the clusters with the Lloyd algorithm.

We decided to test the k-means algorithm in its two versions with nine images of three unknown classes to the CNN (three images of each class), to confirm if the clustering algorithms were able to group the images by model. The clustering process was performed using the feature vectors extracted by the CNN, which, as can be seen in Fig 1. had 2,400 features.

2.4 MOCK Clustering with multi-objective evolutionary approach (PESA-II)

The second approach implemented for the generation of new classes was a multi-objective clustering algorithm with automatic determination of the number of clusters (MOCK) optimized with a multi-objective evolutionary algorithm (MOEA), called PESA-II proposed by Corne et al. [5].

The encoding of individuals uses a representation where each individual *g* is made up of *N* genes, *g1,...,gN,* where *N* is the number of data to be clustered and the value *j* assigned to the *i*-th gene represents a union between the *j* and *i* data. A minimum spanning tree (MST) was generated with the Prim algorithm to initialize the population. The MST represented the first individual and for the subsequent generation of the population the (*i* - 1) longest connections were eliminated. In the genotype this was reflected by assigning to itself the gene representing the connection. The decoding of this representation requires the identification of all the subgraphs since the data belonging to each subgraph is assigned to a cluster.

The variation operators used are the uniform crossover and the nearest neighbor mutation operator, which limits the search space since it can only generate connections between the nearest neighbors of the gene being mutated.

In this approach there are two objective functions to be minimized:

1. **Cluster compactness** or global deviation, which is calculated by summing the distances between each datum and its corresponding centroid in a given cluster and is mathematically represented as:

$$Dev(C) = \sum_{C_k \epsilon C} \sum_{i \epsilon C_k} \delta(i, \mu_k) \tag{2}$$

   Where *C* is the set of clusters M_k is the centroid of cluster C_k, *i* is each element of the data set and δ(...) is the Euclidean distance.

2. **Cluster connectivity** which evaluates if the nearest neighbors of an element have been placed in the same cluster as the current element and is mathematically represented as:

$$Conn(C) = \sum_{i=1}^{N} \left(\sum_{j=1}^{L} x_{i,nn_i(j)} \right), \; donde \; x_{r,s} = \begin{cases} \frac{1}{j} \; si \; \nexists C_k : r, s \in C_k \\ 0 \; de \; lo \; contrario \end{cases} \tag{3}$$

   Where *C* is the set of clusters, *N* is the amount of data in the dataset, *L* is the amount of nearest neighbors (user-defined parameter), $nn_i(j)$ is the *j*th nearest neighbor and $x_{r,s}$ is the penalty function. There will only be penalties if any *j*th nearest neighbor is not in the same cluster as the *i*th data.

PESA- II's pseudocode is shown in Algorithm 1 where the use of two populations of solutions can be highlighted: IP which has a fixed size and is responsible for exploring new solutions and EP which has a limited, but not fixed size and has the job of exploiting good solutions since it consists of "niches" distributed over the objective space (Pareto Front). In each generation, the generated solutions (IP) are evaluated and those that are in the objective space are selected to become part of EP, preferring those solutions that occupy less crowded spaces (the "niches" are avoided) to try to completely cover the Pareto Front.

3 Results

To address the problem of scalability of classification algorithms that use supervised learning, it is essential for our proposal to have a classifier with good recognition accuracy to differentiate between images that belong to the predefined classes from those that do not. To test the classification accuracy of our model the CNN implemented and detailed in Section 2.2 was trained and tested 10 times with the 5 classes of the VMMRdb database [2] mentioned in Section 2 which went under the data augmentation and balancing processes mentioned in Section 2.1. Table 2 shows the accuracy results obtained in the 10 training-testing executions.

Table 2. Comparison between Convolutional Neural Network executions. The execution with the most accurate result is highlighted in bold.

Execution	Cross Entropy Loss	Accuracy	Accuracy per class				
			Dodge Grand Caravan 2005	Ford Explorer 2002	Ford Mustang 2000	Nissan Altima 2005	Toyota Camry 2007
1	0.294	**95%**	95%	100%	95%	90%	95%
2	0.290	91%	100%	85%	90%	90%	90%
3	0.295	92%	100%	95%	90%	95%	80%
4	0.313	93%	95%	95%	85%	95%	95%
5	0.300	91%	95%	100%	95%	90%	75%
6	0.311	92%	90%	100%	85%	100%	85%
7	0.247	90%	100%	95%	90%	90%	75%
8	0.313	93%	100%	90%	90%	85%	100%
9	0.303	91%	90%	85%	95%	95%	90%
10	0.321	94%	100%	85%	95%	95%	95%
Average		92.2%	97%	93%	91%	93%	88%

Due to the scope of the project, it is left as future work the implementation of a novelty detection technique to automatically detect images that do not belong to the predefined classes to which the clustering will be applied to generate new classes. Given the above, in order to show that the K-means and MOCK algorithms were able to cluster the images of the unknown classes for the CNN and thus generate new classes, nine images of three unknown models (three of each class) were entered into the CNN:

Ford Ranger 2019, Toyota Prius 2019 and Volkswagen Beetle 2013 which, as expected, generated a misclassification as it was not trained for those classes, however, what was important in this case was to obtain the feature vectors generated by the CNN to enter them into the clustering algorithms.

For the execution of MOCK the parameters were calibrated as follows: Number of generations = 100, Maximum External Population Size = 15, Internal Population Size = 8, L nearest neighbors = 3, Crossover Probability = 0.5. Fig. 2 shows the Pareto Fronts obtained in five executions of the algorithm.

It is important to remember that the clustering process by definition is subjective. Also, it should be considered that each time the CNN is trained, it will learn in a unique way and will be reflected in the weights that are set at the end of the training, therefore, the features extracted after each training will depend on those learned weights. Taking these two points into consideration, five tests of the clustering algorithms were performed using five sets of feature vectors obtained from five executions of the CNN with different weights. Only with one of the sets the desired clustering was achieved by both clustering approaches as shown in Fig. 3. Initially the hypothesis was that MOCK would outperform K-means, however the results showed similar performances where the only advantage shown by MOCK was the automatic determination of the number of clusters. To get a better understanding of the results, five distance matrices of the five sets of feature vectors extracted by the CNN were performed using the Euclidean distance since the same metric was used in the clustering algorithms. This test showed that only in one of the matrices the feature vectors belonging to the same classes were spatially close and it was with that set that the clustering algorithms achieved the desired clustering.

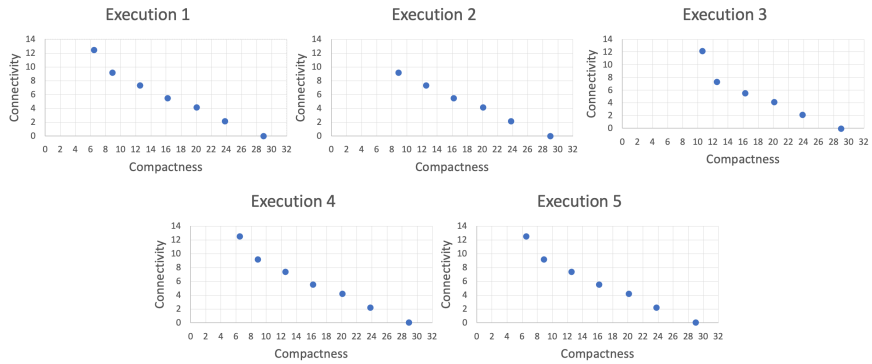

Fig. 2. Pareto fronts obtained in 5 executions of the MOCK algorithm.

Fig. 3. Classification expected and obtained with the K-means and MOCK algorithms.

One point to highlight with the K-means approach is that the addition of the Lloyd algorithm implemented by sklearn library for the initialization of the centroids gave it a great advantage over the classical version that did not achieve the expected classification in any of the executions. Finally, it was noted that in most cases the clustering were related to the predominant shades in the images, which indicates that it may be preferable to work with grayscale images to prevent bias.

4 Conclusions and future work

In this work a feasibly solution to the scalability problem was presented, confirming that it is possible to generate new classes using clustering algorithms to group images based on the feature vectors extracted by the classification model even if the classes are unknown. However, it was observed that the feature vectors belonging to the same classes were not in close regions in terms of Euclidean distance. The research carried out after this project reflected that this distance metric is recommended only to compare points in two or three dimensions. In larger dimensional spaces, all points tend to be far apart, then other measures will be explored such as the cosine distance. Another point to consider is that the feature vectors will depend on the training of the network, specifically the learned weights, which can influence negatively. To achieve a better control, we propose the use of neuroevolution of CNNs with an objective function that measures the consistency of the feature maps. By doing this, we could get feature vectors located spatially close if they belong to the same class, and far away if they belong to different classes. In the literature review this has been achieved using techniques such as Contrastive Loss [7].

Regarding the clustering algorithms, according to authors such as Martínez-Peñaloza et al. [6], better results were achieved using the MOEA NSGA-II compared to the original version of MOCK which uses PESA-II. Since in this work there were no explicit differences between K-means and MOCK, it is proposed to use the optimized version with NSGA-II to try to achieve better results.

Finally, it is proposed as future work to increase the number of classes and perform a novelty detection to automatically recognize vehicles that do not belong to the labels with which the CNN is trained and perform clustering on them.

5 Acknowledgments

The authors would like to thank the Consejo Nacional de Ciencia y Tecnología (CONACYT), an institution of the Government of Mexico, for the financial support provided through the "Beca Nacional" as part of the Programa de Becas para Estudios de Posgrado.

References

1. 1. Nazemi A., Azimifar Z., Shafiee M., Wong A.: Real-Time Vehicle Make and Model Recognition Using Unsupervised Feature Learning. In: IEEE Transactions on Intelligent Transportation Systems, vol. 21, no. 7, pp. 3080-3090. (2020). https://doi.org/10.1109/TITS.2019.2924830.
2. 2. Tafazzoli F., Frigui H., Nishiyama K.: A Large and Diverse Dataset for Improved Vehicle Make and Model. In: IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 874-881 (2017). https://doi.org/10.1109/CVPRW.2017.121
3. 3. Radich, Q., Jenks, A.: Use PyTorch to train your image classification model. In: Microsoft Docs. https://docs.microsoft.com/en-us/windows/ai/windows-ml/tutorials/pytorch-train-model. Last accessed 10 May 2022
4. 4. Kezebou L., Oludare V., Panetta K., Agaian S.: Few-Shots Learning for Fine-Grained Vehicle Model Recognition. In: IEEE International Symposium on Technologies for Homeland Security (HST), pp. 1-9 (2021). https://doi.org/10.1109/HST53381.2021.9619823
5. 5. Corne D., Jerram N., Knowles J., Oates M.: PESA-II: Region-Based Selection in Evolutionary Multiobjective Opti- mization. In: Spector L, Goodman D et al (eds) Proceedings of the genetic evolutionary computation conference 2001, pp 283–290. Morgan Kaufmann, San Francisco (2001). https://dl.acm.org/doi/10.5555/2955239.2955289
6. 6. Martínez-Peñaloza, M., Mezura-Montes, E., Cruz-Ramírez, N., Acosta-Mesa, H., Ríos-Figueroa, H.: Improved multi-objective clustering with automatic determination of the number of clusters. Neural Computing and Applications, 28(8), 2255–2275 (2017). https://doi.org/10.1007/s00521-016-2191-1
7. 7. Wang F., Liu H.: Understanding the Behaviour of Contrastive Loss. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2495-2504 (2021). https://doi.org/10.1109/CVPR46437.2021.00252