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

DETECCIÓN DE MODELOS Y TIPOS DE VEHÍCULOS FRAUDULENTOS EN AUTOPISTAS MEDIANTE APRENDIZAJE PROFUNDO NO SUPERVISADO

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

DIEGO ANDRÉS VEGA ACUÑA

PROFESOR GUÍA:
CARLOS NAVARRO

MIEMBROS DE LA COMISIÓN:
ANDRÉS CABA
PATRICIO VALENZUELA

SANTIAGO DE CHILE
2026

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: DIEGO ANDRÉS VEGA ACUÑA
FECHA: 2026
PROF. GUÍA: CARLOS NAVARRO

DETECCIÓN DE MODELOS Y TIPOS DE VEHÍCULOS FRAUDULENTOS EN AUTOPISTAS MEDIANTE APRENDIZAJE PROFUNDO NO SUPERVISADO

El presente trabajo aborda la viabilidad de clasificar modelos de vehículos a partir de representaciones profundas (*embeddings*) generadas por **Redes Neuronales**, sin utilizar metodos de clasificación tradicionales. El objetivo central es investigar si la estructura semántica de estos *embeddings* es lo suficientemente **robusta y discriminativa** para la **identificación de modelos de vehículos (vistos o no vistos previamente)** mediante técnicas de agrupamiento. Esto busca resolver el desafío de la detección de modelos en escenarios con información limitada.

La aproximación se basa en el **Aprendizaje por Transferencia** y la **Exploración No Supervisada de Representaciones**. Se utilizan **Redes Neuronales Profundas** de visión por computadora (basadas en **CNNs** y **Transformers**) y modelos multimodales de visión y texto, previamente entrenadas con datos a gran escala, y se somete a un ajuste fino (*fine-tuning*) supervisado utilizando conjuntos de datos especializados (en este caso, el dataset *Comprehensive Cars*).

La fase de ajuste fino explora diversas estrategias de optimización que incluyen funciones de pérdida de clasificación y de *metric learning* (como pérdidas contrastivas y angulares), enfocadas en aumentar la separabilidad inter-clase y la compactación intra-clase de los *embeddings*. Esta etapa optimiza los **vectores de características (*embeddings*)** y su espacio latente asociado. Posteriormente, se explora la capacidad de estos *embeddings* para agrupar vehículos del mismo modelo mediante diversos algoritmos de **Agrupamiento (*clustering*)**, con la optimización de hiperparámetros realizada sistemáticamente para maximizar la **pureza del *cluster***.

Los resultados demuestran la **viabilidad y la alta efectividad** del enfoque propuesto. Se validó que los *embeddings* generados son **altamente discriminativos**, permitiendo el **agrupamiento puro/clasificación no supervisada** (una imagen del modelo **X** se agrupa únicamente con otras imagenes del modelo **X**) en el espacio latente. Esto prueba que la **clasificación de modelos puede lograrse de forma no supervisada** a partir de las representaciones profundas, incluso ante modelos no incluidos en la etapa de ajuste fino, superando las limitaciones de los métodos tradicionales.

La investigación establece que las **Representaciones Profundas son una solución robusta** para la identificación de modelos de vehículos mediante *clustering*. Esto tiene implicaciones directas para la **detección de anomalías y fraude** en aplicaciones reales (como autopistas), al permitir la detección de modelos desconocidos o poco representados, sin la necesidad de realizar reentrenamiento.

Dedicada a Matías, mi hermano.

Agradecimientos

Agradezco a mi madre, Claudia Acuña, por su amor incondicional, por su resiliencia y por ser mi ejemplo de esfuerzo diario, amor y bondad.

Agradezco a mi padre, Rodrigo Vega, por ser mi primer maestro en las matemáticas.

Agradezco a mi hermano, Matías Vega, por enseñarme qué es poner al otro por encima de uno mismo y por inculcarme el amor a las matemáticas y las ciencias.

Agradezco a mi tía, Marcela Acuña, por ser una segunda madre para mí y mi hermano, y por ser una compañera de vida para mi madre.

Agradezco a mi tío, Ignacio Badal, por todo el amor que nos entregó a mí y a mi hermano durante nuestra infancia.

Agradezco a mi prima, Magdalena Badal, por incentivarme a ingresar a la Universidad de Chile y por el gran amor que ha mostrado hacia mí, mi hermano y mi madre.

Agradezco a mi prima, Montserrat Badal, por su cariño y entrega incondicional hacia mí y mi familia, además de su profundo amor por los niños, el cual ha marcado mi vida en los últimos años.

Agradezco a mi primo, Ignacio Badal, por ser un gran amigo y compañero de vida.

Agradezco a mi primo, Mateo Badal, por su forma de ser, su pureza y su alegría contagiosa.

Agradezco a mi tía, María Elena Acuña, por su apoyo incondicional y por ser un referente en el pensamiento crítico.

Agradezco a mi tío, Ernesto Contreras, por su gran entrega a la familia y por ser un amigo incondicional.

Agradezco a mi primo, Ernesto Contreras, por compartir conmigo nuestros intereses comunes en la música, las matemáticas y las ciencias, y por ser un ejemplo de esfuerzo académico.

Agradezco a mi prima, Elena Contreras, por su genuino amor por los demás, por su aprecio al arte y por su gran talento en el mismo.

Agradezco a mi tío, Vicente Acuña, por su gran cariño hacia mí, su pasión por el cine, la literatura, la música y las artes en general, además de su intrínseco amor al conocimiento.

Agradezco a mi abuelo, Vicente Acuña, por su fuerza, resiliencia y su entrega al pueblo chileno.

Agradezco a mi abuela, Marily Moenne, por su amor incondicional que permeó a todos mis tíos y primos, además de a mi hermano y a mí. Ella es mi mayor ejemplo de amor y entrega hacia los demás.

Agradezco a mi tío, Felipe Vega, cuyo amor hacia mí y mi hermano ha sido incondicional a lo largo de nuestras vidas.

Agradezco a mis primos, Ignacio Vio, Benjamín Vega y Agustín Vega; aún niños, me han enseñado lo bello de la infancia y la pureza del amor fraternal.

Agradezco a mi abuelo, Ramón Vega, por ser un gran ejemplo de esfuerzo y por su cariño hacia mí y mi hermano.

Agradezco a mi abuela, Alicia Hernández, por su amor incondicional hacia mí y mi her-

mano, y por ser nuestra cuidadora durante la infancia. Sus cuidados y enseñanzas han nutrido mi vida.

Agradezco a mis grandes amistades: Flavio Poblete, Mauricio Valenzuela, Pamela Acuña, Gabriel Raquiman, Francisco Muñoz, Benjamín Rizzardini y José Pablo Muñoz. Su apoyo incondicional, amistad y enseñanzas han sido fundamentales en mi vida.

Agradezco a mis profesores y maestros, quienes han guiado mi camino no solo académico, sino de la vida misma.

Quisiera sobre todo destacar a Antonio Díaz-Araujo, quien ha sido mi gran maestro en cuanto a la disciplina y dinámica de trabajo.

A Carlos Navarro, por su gran apoyo y retroalmentación. Sus comentarios realizados sobre mis primeros resultados fueron cruciales para el desarrollo de este trabajo.

A Andrés Caba, por su gran ayuda, apoyo y disposición, no solo en el presente trabajo si no que también en todos mis años dentro del departamento de Ingieneria Electrica, sobre todo los últimos.

A Javier Ruiz del Solar, quien me transmitió su amor por el área de la Inteligencia Artificial y por ser mi referente en la investigación de la misma.

Agradezco a Paulina Céspedes, por su amor incondicional, su apoyo constante, su entrega y por haberme acompañado en los que han sido los años más desafiantes de mi vida.

Gracias a la vida
Que me ha dado tanto
Me ha dado la risa
Y me ha dado el llanto
Así yo distingo
Dicha de quebranto
Los dos materiales que forman
Mi canto
Y el canto de ustedes que es el mismo canto
Y el canto de todos que es mi propio
Canto

— Violeta Parra, 1966 [1]

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Motivación

La identificación de modelos de vehículos es un desafío que, aunque actualmente explorado en el ámbito académico, no posee una solución definitiva, a pesar de su relevancia en aplicaciones prácticas en los ámbitos de la seguridad y la vigilancia. Uno de los motivos es que los métodos tradicionales de clasificación suelen depender de grandes datos etiquetados y de que las clases a identificar estén presentes en el conjunto de entrenamiento.
Sin embargo, en escenarios del mundo real, como la vigilancia de autopistas, es común enfrentar situaciones donde los modelos de vehículos pueden no haber sido vistos previamente o donde la disponibilidad de datos etiquetados es limitada. Esto plantea la necesidad de desarrollar enfoques que puedan identificar modelos de vehículos de manera efectiva, incluso en estas condiciones desafiantes.
El presente trabajo propone una solución innovadora basada en el uso de representaciones profundas (*embeddings*) generadas por **Redes Neuronales**, combinadas con técnicas de agrupamiento no supervisado (*clustering*). Al aprovechar el aprendizaje por transferencia (*transfer learning*) y la exploración no supervisada de representaciones, **se busca superar las limitaciones de los métodos tradicionales y permitir la identificación de modelos de vehículos, incluso aquellos no presentes en el conjunto de entrenamiento**. Generando así un sistema robusto y adaptable a las condiciones del mundo real.

1.2. Objetivos

1.2.1. Objetivo General

Desarrollar, implementar y evaluar un sistema basado en representaciones profundas y técnicas de agrupamiento no supervisado para la **identificación efectiva de modelos de vehículos** en escenarios con datos limitados y modelos no vistos previamente.

1.2.2. Objetivos Específicos

- Investigar, seleccionar, utilizar y comparar diversas arquitecturas de **Redes Neuronales** adecuadas para la generación de representaciones profundas de vehículos.
- Implementar un proceso de ajuste fino (*fine-tuning*) supervisado utilizando conjuntos de datos especializados para optimizar la calidad de los *embeddings*.

- Investigar, utilizar y comparar diversas técnicas de agrupamiento no supervisado para agrupar e identificar modelos de vehículos a partir de las representaciones profundas generadas.
- Proponer e implementar un sistema de clasificación no supervisada de modelos de vehículos basado en las representaciones profundas y técnicas de agrupamiento.
- Analizar la efectividad del sistema propuesto en términos cuantitativos y cualitativos, considerando la "pureza" del *cluster* (agrupamiento de un único modelo) y la capacidad de generalización del sistema en casos de modelos no vistos previamente.

1.3. Hipótesis

La combinación de **Representaciones Profundas Multimodales** (Visión y Texto, basadas en **Transformers**) optimizadas mediante *Metric Learning* y técnicas de *clustering* **no supervisado** (**con auto-exploración de hiperparámetros**) resulta en *embeddings* con una **mayor calidad discriminativa** y **robustez de generalización** (*Zero-Shot*/*Few-Shot*) que las representaciones generadas por modelos unimodales (visión pura), lo que permite alcanzar una **mayor pureza de agrupamiento** de modelos de vehículos (vistos y no vistos) y, por lo tanto, una **identificación más efectiva de modelos de vehículos** en general.

1.4. Estructura del Documento

El documento se organiza en los siguientes capítulos:

- **Capítulo 1: Introducción** - Se presenta la motivación, los objetivos y la hipótesis del estudio.
- **Capítulo 2: Marco Teórico y Estado del Arte** - Se presentan los conceptos fundamentales de la Inteligencia Artificial, el Aprendizaje Automático y el Aprendizaje Profundo, así como una revisión del estado del arte en la identificación de vehículos y técnicas de agrupamiento.
- **Capítulo 3: Metodología** - Se detalla la metodología empleada en el desarrollo del proyecto, incluyendo la selección de datos, las arquitecturas de **redes neuronales** utilizadas, las técnicas de ajuste fino, los algoritmos de agrupamiento implementados y los experimentos realizados.
- **Capítulo 4: Resultados y Discusión** - Se presentan los resultados obtenidos a partir de los diferentes experimentos realizados, junto con un análisis crítico de su desempeño y comparaciones entre los enfoques probados.
- **Capítulo 5: Conclusiones** - Se resumen los hallazgos principales del estudio, se discuten las implicaciones prácticas y se sugieren posibles direcciones para futuras investigaciones en el área.

Capítulo 2

Marco Teórico y Estado del Arte

2.1. Inteligencia Artificial y Aprendizaje Automático

La Inteligencia Artificial (*Artificial Intelligence*) es la disciplina de las ciencias de la computación dedicada al diseño de sistemas capaces de realizar tareas que normalmente requieren inteligencia humana, tales como el razonamiento, la resolución de problemas, el aprendizaje, la percepción y la toma de decisiones [2].
Sus principales subcampos incluyen:

- **Procesamiento de Lenguaje Natural (PLN):** comprensión y generación de texto o voz.
- **Representación del conocimiento:** codificación de información del mundo para la inferencia automática.
- **Razonamiento automatizado:** derivación de conclusiones a partir de premisas.
- **Aprendizaje automático (Machine Learning):** algoritmos que mejoran su desempeño a partir de datos sin programación explícita [3].
- **Visión por computadora:** interpretación y análisis de imágenes y vídeo.
- **Robótica:** control e interacción de agentes físicos en entornos del mundo real.

Dentro de la IA, el aprendizaje automático ha adquirido una relevancia particular debido al incremento masivo de datos disponibles y la necesidad inherente de extraer patrones útiles de manera eficiente.
Se reconocen tres paradigmas principales [3]:

1. **Supervisado:** el modelo aprende a mapear entradas a salidas etiquetadas.
2. **No supervisado:** el modelo descubre estructuras o patrones en datos no etiquetados.
3. **Por refuerzo:** un agente optimiza una señal de recompensa mediante interacción con un entorno.

2.2. Aprendizaje Profundo (*Deep Learning*)

El aprendizaje profundo (*Deep Learning*) es un subcampo del aprendizaje automático que emplea **redes neuronales** con múltiples capas ocultas para modelar representaciones jerárquicas directamente a partir de datos no estructurados, como imágenes, audio o texto [3]. A diferencia del aprendizaje automático tradicional, el aprendizaje profundo no requiere la extracción manual de características, lo que ha sido un factor clave en la evolución de la visión por computadora, donde anteriormente la generación de descriptores (*features*) constituía un cuello de botella [4].
La transición de la ingeniería manual de características a la capacidad de aprendizaje automático de representaciones constituye un cambio de paradigma fundamental. En el contexto de la identificación granular de vehículos, donde distinguir entre modelos requiere detectar sutilezas visuales complejas, el aprendizaje profundo permite descubrir características discriminatorias de manera automática, superando las limitaciones y sesgos del diseño manual.
A modo de ilustración, la relación jerárquica entre *Artificial Intelligence*, *Machine Learning* y *Deep Learning* se presenta en la Figura 2.1, la cual presenta un diagrama de *Euler* que delimita el alcance de cada disciplina.

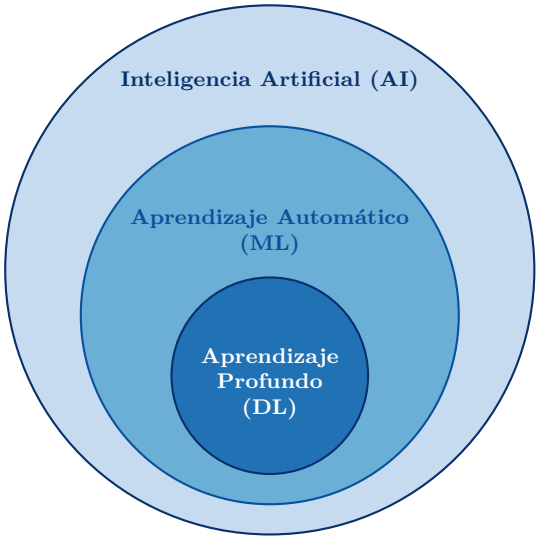


Figura 2.1: Relación jerárquica entre disciplinas. El Aprendizaje Profundo es un subconjunto del Aprendizaje Automático, el cual a su vez es parte de la Inteligencia Artificial.

2.3. Conceptos Fundamentales de Redes Neuronales Profundas

Una red neuronal se compone de neuronas interconectadas organizadas en capas. Cada neurona recibe una o más entradas, realiza una suma ponderada de estas, aplica una función de activación y produce una salida [2, 3].

2.3.1. Neurona Artificial

$$y = f\left(\sum_{i=1}^{n} w_i \cdot x_i + b\right) \tag{2.1}$$

La Ecuación 2.1 presenta la fórmula que representa la operación básica de una neurona artificial, en donde la salida y es el resultado de aplicar una función de activación no lineal f a una combinación lineal de las entradas. Aquí, x_i representa las señales de entrada, w_i son los pesos sinápticos que ponderan la importancia de cada entrada, y b es el sesgo (*bias*) que permite desplazar el umbral de activación.
Esta estructura matemática se visualiza en la Figura 2.2, destacando el flujo desde la recepción de señales ponderadas hasta la generación de la salida.

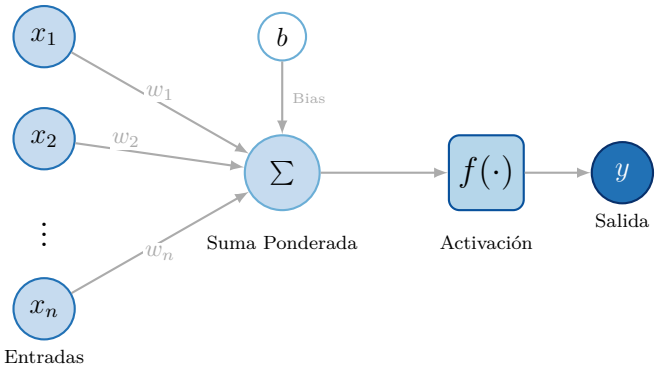


Figura 2.2: Representación esquemática de una neurona artificial. Las entradas x_i son ponderadas por w_i, sumadas junto con el sesgo b, y procesadas por una función de activación f para producir la salida y.

2.3.2. Funciones de Activación

Las funciones de activación introducen no linealidad en la red, permitiéndole aprender patrones complejos [4].
Algunas de las funciones de activación más comunes son:

Función de Rectificación Lineal Unitaria (ReLU)

$$f(x) = \max(0, x) \tag{2.2}$$

La función **ReLU** es la más utilizada en las capas ocultas de las **Redes Neuronales** modernas. Mantiene la salida en cero para cualquier entrada negativa y devuelve la entrada misma para cualquier valor positivo. Esto introduce la no linealidad necesaria de manera computacionalmente eficiente. Su rango es $[0, \infty)$.

Función Sigmoide

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

La función **Sigmoide** (o logística) comprime cualquier valor real de entrada en el rango $(0, 1)$. Históricamente, fue popular en las capas ocultas, pero sufre del problema del

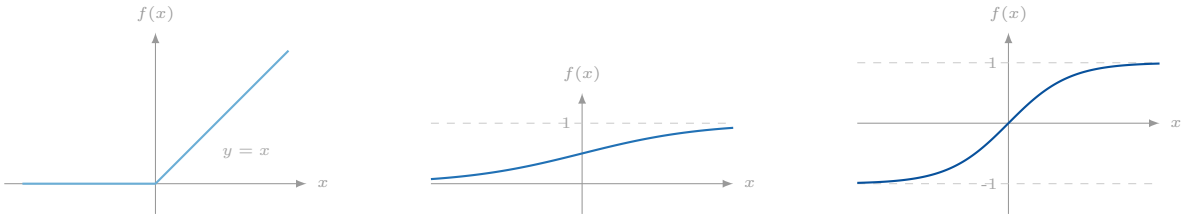
gradiente desvaneciente para valores muy grandes o muy pequeños de la entrada. Actualmente es más utilizada en la capa de salida para problemas de clasificación binaria.

Función Tangente Hiperbólica (Tanh)

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.4}$$

Similar a la **Sigmoide**, la función **Tanh** también es una función sigmoidal, pero comprime los valores de entrada en el rango $(-1, 1)$. Al estar centrada en cero, a menudo converge más rápido que la **Sigmoide** durante el entrenamiento, ya que los gradientes tienen una media de cero.

El comportamiento de estas funciones es fundamental para la dinámica de aprendizaje de la red. La Figura 2.3 muestra gráficamente sus perfiles de activación, ilustrando sus rangos de salida y zonas de saturación.



(a) **ReLU**. (b) **Sigmoide**. (c) **Tangente Hiperbólica**.

Figura 2.3: Perfiles de activación de las funciones más comunes.

2.3.3. Operación de Convolución

$$O(i,j) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I(i+m, j+n) \cdot K(m,n) \tag{2.5}$$

La operación de convolución consiste en deslizar un filtro (*Kernel*) sobre la imagen de entrada [4]. Para una imagen I y un filtro K de tamaño $M \times N$, la salida $O(i,j)$ se calcula como la suma ponderada de los valores de la región cubierta por el filtro en la imagen de entrada, tal como se define en la Ecuación 2.5. Parámetros clave incluyen el *Stride* (paso de desplazamiento) y el *Padding* (relleno de bordes), los cuales determinan las dimensiones espaciales de salida. La Figura 2.4 ilustra conceptualmente este proceso, mostrando cómo una región local de la entrada se procesa mediante el filtro para producir un único valor en el mapa de características de salida.

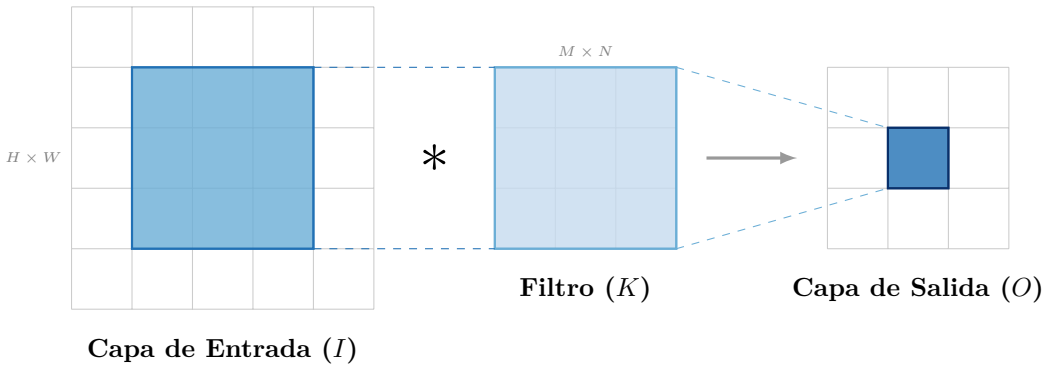


Figura 2.4: Representación esquemática de la operación de convolución 2D.

2.3.4. Operaciones de *Pooling*

Las capas de *pooling* reducen la dimensionalidad espacial, disminuyendo la carga computacional y controlando el sobreajuste [3].
Dos técnicas comunes son:

Max Pooling

$$O(i,j) = \max_{(m,n)\in\mathcal{R}} I(i+m, j+n) \tag{2.6}$$

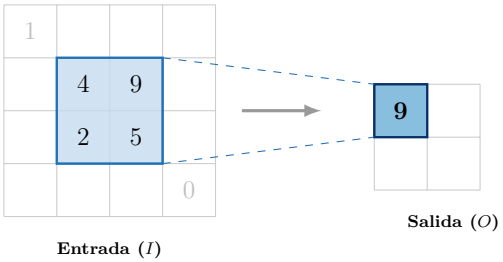
Selecciona el valor máximo en la ventana (región $\mathcal{R}$), preservando características prominentes.

Average Pooling

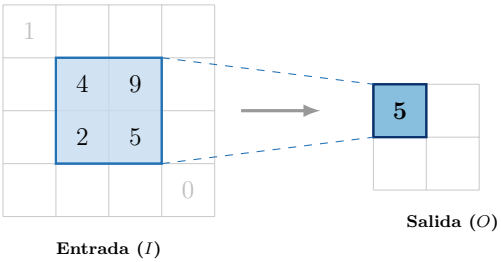
$$O(i,j) = \frac{1}{|\mathcal{R}|} \sum_{(m,n)\in\mathcal{R}} I(i+m, j+n) \tag{2.7}$$

Calcula el promedio de valores en la ventana (región $\mathcal{R}$).

La Figura 2.5 ilustra ambas operaciones, mostrando cómo una región de la entrada se reduce a un solo valor en la salida.



(a) *Max Pooling* (Máximo: 9)



(b) *Average Pooling* (Promedio: 5)

Figura 2.5: Representación esquemática de las operaciones de *Pooling*.

2.4. Arquitecturas Transformer en Visión por Computadora

Originalmente propuestos para el Procesamiento de Lenguaje Natural (*Natural Language Processing*) [5], los **Transformers** han revolucionado la visión por computadora. Su núcleo es el mecanismo de **autoatención (*self-attention*)**, que pondera dinámicamente la influencia de cada parte de la entrada sobre las demás, permitiendo capturar relaciones globales de largo alcance, superando así las limitaciones locales de las **CNNs**.

2.4.1. Mecanismo de Autoatención

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.8}$$

La autoatención calcula una puntuación de relevancia utilizando vectores de **Query** (Q), **Key** (K) y **Value** (V) [5]. La Ecuación 2.8 describe este proceso, donde las puntuaciones se normalizan mediante la función *softmax* y se utilizan para ponderar los valores V.
La Figura 2.6 ilustra el flujo de datos en el mecanismo de autoatención.

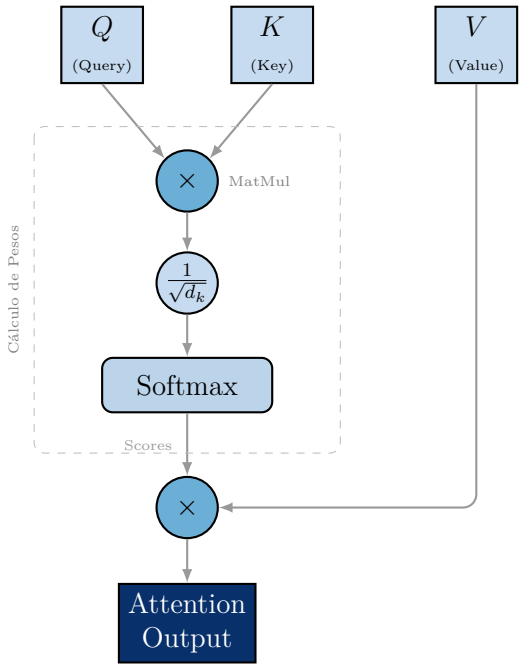


Figura 2.6: Diagrama esquemático del Mecanismo de Autoatención (*Scaled Dot-Product*). Elaboración propia basada en [5].

2.4.2. Autoatención Multi-Cabeza

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \tag{2.9}$$

Comunmente, la autoatención es utilizada con multiples "cabezas", lo que es conocido como **autoatención multi-cabeza**. Lo anterior permite al modelo atender a múltiples subes-

pacios de representación simultáneamente, enriqueciendo su capacidad para comprender relaciones complejas entre parches distantes de un texto, imagen o audio. La Ecuación 2.9 describe este proceso, donde cada *head* representa una instancia independiente del mecanismo de autoatención, y sus salidas se concatenan y proyectan para formar la salida final.
La Figura 2.7 ilustra el flujo de datos en la autoatención multi-cabeza.

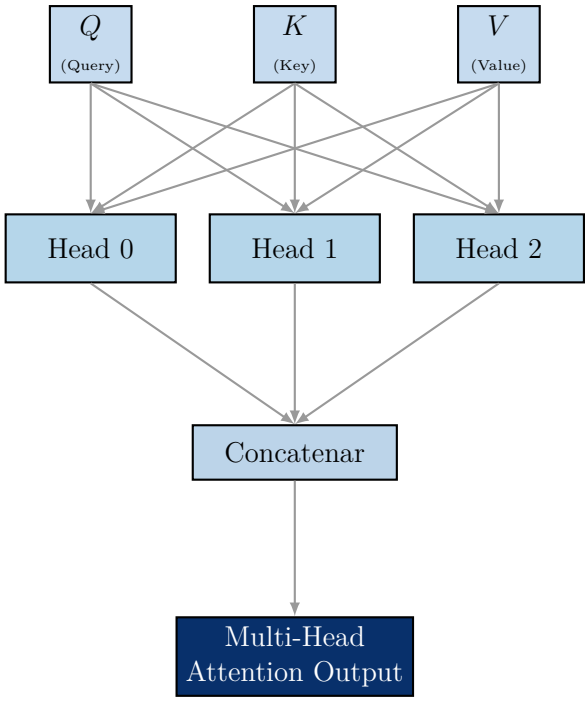


Figura 2.7: Diagrama esquemático de la Autoatención Multi-Cabeza. Elaboración propia basada en [5].

2.4.3. Vision Transformers (ViT)

$$Z_0 = [x_{cls}; x_1E; x_2E, ..., x_NE] + E_{pos} \tag{2.10}$$

Los **Vision Transformers (ViT)** adaptan la autoatención multi-cabeza al dominio de las imágenes, dividiéndolas en parches fijos, linealizándolos en *tokens* y procesándolos con un codificador **Transformer** estándar [6].
La secuencia de entrada se define según la Ecuación 2.10, donde x_{cls} es el token de clasificación cuya salida final representa la **imagen** completa en el espacio latente. Esta capacidad de atención global resulta crítica para diferenciar modelos de vehículos visualmente similares.
La Figura 2.8 presenta un esquema del proceso de los **ViT**: desde la división de la imagen en parches hasta la proyección lineal y el procesamiento por el codificador **Transformer**.

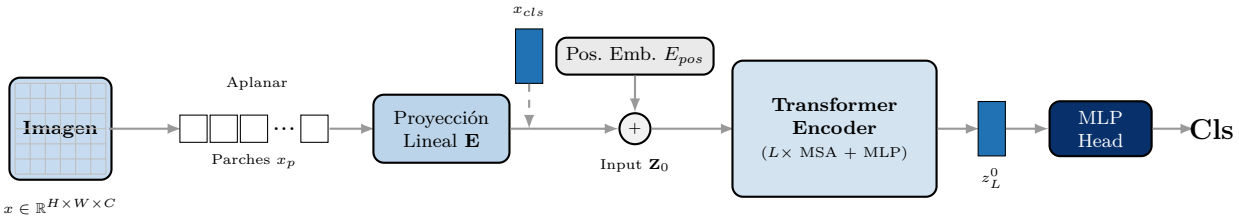


Figura 2.8: Arquitectura del **Vision Transformer (ViT)**. Elaboración propia basada en la arquitectura propuesta por [6].

2.5. Modelos Multimodales y Aprendizaje Contrastivo

La convergencia entre la visión por computadora y el procesamiento del lenguaje natural ha dado lugar a **modelos multimodales** capaces de aprender representaciones conjuntas de imágenes y texto [4]. Estos modelos aprovechan la supervisión natural disponible en grandes volúmenes de pares imagen-texto en la web.

2.5.1. CLIP (Contrastive Language-Image Pre-training)

CLIP es una arquitectura fundamental que entrena dos codificadores: uno para imágenes (que puede ser basado en una **CNN** o un **ViT**) y otro para texto (basado en **Transformer**) [7]. El objetivo del entrenamiento es predecir qué descripción de texto corresponde a qué imagen dentro de un lote.
La Figura 2.9 muestra el paradigma de entrenamiento de **CLIP**, donde ambos codificadores proyectan sus entradas a un espacio latente compartido para maximizar la similitud de los pares correctos.

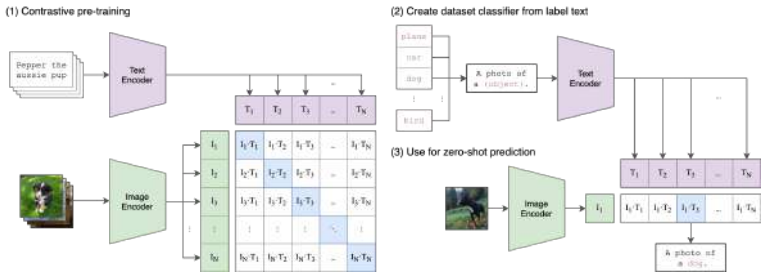


Figura 2.9: Arquitectura oficial de **CLIP** presentando los codificadores de imagen y texto junto al espacio latente compartido. Adaptado de [7]. Licencia CC BY 4.0.

$$\mathcal{L} = -\frac{1}{2N}\sum_{i=1}^{N}\left(\log\frac{\exp(I_i\cdot T_i/\tau)}{\sum_{j=1}^{N}\exp(I_i\cdot T_j/\tau)} + \log\frac{\exp(T_i\cdot I_i/\tau)}{\sum_{j=1}^{N}\exp(T_j\cdot I_i/\tau)}\right) \tag{2.11}$$

La Ecuación 2.11 formaliza la pérdida de entropía cruzada simétrica utilizada en el entrenamiento de **CLIP**. En ella, I_i y T_i son los *embeddings* normalizados, por lo que su producto punto equivale directamente a la *similitud coseno*. La optimización minimiza la divergencia en ambas direcciones (imagen $\rightarrow$ texto y texto $\rightarrow$ imagen) ajustada por el parámetro de temperatura τ. Esta estricta alineación del espacio latente visual con el semántico es lo que

dota a **CLIP** de capacidades *Zero-Shot* robustas, permitiéndole generar representaciones discriminativas incluso para clases no vistas durante el entrenamiento, una característica que resulta fundamental para la identificación de clases desconocidas, en este trabajo: modelos de vehículares.

2.6. Transferencia de Aprendizaje y Extracción de Características

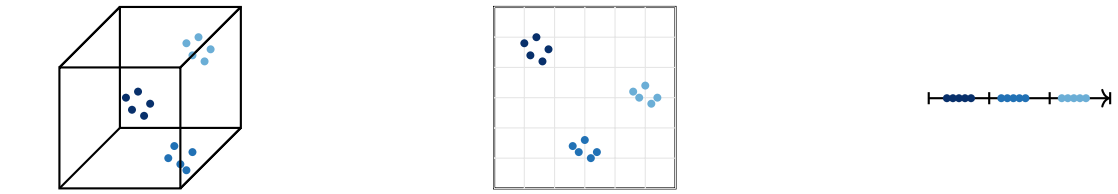
El **Aprendizaje por Transferencia (*Transfer Learning*)** es una técnica donde un modelo entrenado en una tarea con grandes volúmenes de datos (ej. ImageNet [8]) se reutiliza como punto de partida para una tarea relacionada con menos datos disponibles [9].

2.6.1. *Embeddings* Profundos

Los modelos preentrenados actúan como potentes extractores de características. Al eliminar la capa de clasificación final, se obtiene un vector denso de alta dimensión conocido como ***embedding***. Estos vectores codifican la información semántica y visual de la entrada: imágenes visualmente similares producen *embeddings* cercanos en el espacio vectorial. La calidad de estos *embeddings*, provenientes de arquitecturas como **ResNet**, **ViT** o **CLIP**, es el fundamento que habilita el uso de técnicas de agrupamiento no supervisado, dotando a datos brutos (puramente matemáticos) de una representación manejable para la detección de patrones [10].

2.7. Técnicas de Reducción de Dimensionalidad

Los *embeddings* generados por redes profundas suelen poseer una alta dimensionalidad (ej. 512, 768 o 2048 dimensiones). Para mejorar la eficiencia computacional y mitigar la "maldición de la dimensionalidad" en el agrupamiento posterior, se emplean técnicas de reducción que buscan preservar la estructura latente de los datos.
La Figura 2.10 ilustra conceptualmente el objetivo de estas técnicas: proyectar datos complejos de alta dimensión a un espacio de menor dimensión (ej. 3D a 2D a 1D) manteniendo la estructura de vecindad.



(a) Espacio Original ($\mathbb{R}^3$) (b) Proyección ($\mathbb{R}^2$) (c) Latente ($\mathbb{R}^1$)

Figura 2.10: Visualización conceptual del proceso de reducción de dimensionalidad preservando la estructura de vecindad.

Las técnicas utilizadas para el desarrollo de este trabajo incluyen:

PCA (Análisis de Componentes Principales) [11]

$$\begin{aligned} \Sigma &= \frac{1}{N-1}\sum_{i=1}^{N}(x_i-\bar{x})(x_i-\bar{x})^T \\ \Sigma W &= \Lambda W \\ Y &= XW_k \end{aligned} \tag{2.12}$$

Técnica lineal cuyo objetivo es proyectar los datos en un espacio de menor dimensión maximizando la varianza. Como se observa en la Ecuación 2.12, el proceso inicia con el cálculo de la matriz de covarianza Σ, seguido de su descomposición para obtener W (que contiene los vectores propios). Finalmente, W_k es la matriz truncada con los k componentes principales que transforman los datos originales X en la proyección Y.

t-SNE (*t-distributed Stochastic Neighbor Embedding*) [12]

$$\begin{aligned} p_{j|i} &= \frac{\exp(-\|x_i-x_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|x_i-x_k\|^2/2\sigma_i^2)} \\ q_{ij} &= \frac{(1+\|y_i-y_j\|^2)^{-1}}{\sum_{k\neq l}(1+\|y_k-y_l\|^2)^{-1}} \\ C &= \sum_i KL(P_i||Q_i) = \sum_i\sum_j p_{j|i}\log\frac{p_{j|i}}{q_{j|i}} \end{aligned} \tag{2.13}$$

Técnica no lineal probabilística presentada en la Ecuación 2.13. Primero define la similitud en alta dimensión ($p_{j|i}$) mediante distribuciones Gaussianas, y la similitud en baja dimensión (q_{ij}) utilizando una distribución t-Student. El objetivo es minimizar la divergencia de Kullback-Leibler (KL) (C) entre ambas distribuciones, lo que permite preservar las vecindades locales mientras separa los clústeres globales.

UMAP (Uniform Manifold Approximation and Projection) [13]

$$\begin{aligned} w_{ij} &= \exp(-(d(x_i,x_j)-\rho_i)/\sigma_i) \\ C_{UMAP} &= \sum_{i,j}\left[w_{ij}\log\left(\frac{w_{ij}}{w'_{ij}}\right)+(1-w_{ij})\log\left(\frac{1-w_{ij}}{1-w'_{ij}}\right)\right] \end{aligned} \tag{2.14}$$

Técnica basada en topología que optimiza la preservación de la estructura mediante la función de pérdida de entropía cruzada descrita en la Ecuación 2.14. Aquí, w_{ij} representa la probabilidad de conexión en alta dimensión (basada en la distancia $d(x_i,x_j)$ y los parámetros locales ρ_i, σ_i), mientras que w'_{ij} representa la probabilidad en baja dimensión. La función C_{UMAP} equilibra la atracción entre vecinos (primer término) y la repulsión entre no-vecinos (segundo término).

En la Figura 2.11 se presentan las proyecciones bidimensionales obtenidas mediante **PCA**, **t-SNE** y **UMAP** en el dataset Digits [14]. Se observa cómo cada técnica preserva diferentes aspectos de la estructura de los datos: **PCA** mantiene la varianza global, mientras que **t-SNE** y **UMAP** enfatizan las relaciones locales entre puntos similares.

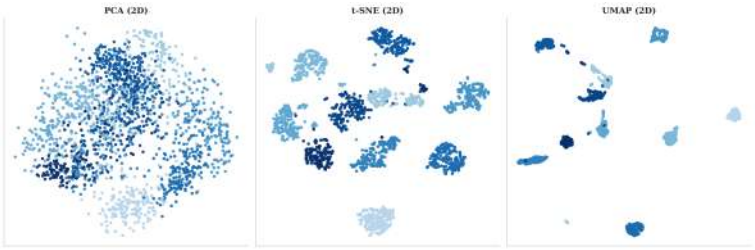


Figura 2.11: Representaciones bidimensionales del dataset Digits [14] obtenidas mediante los métodos **PCA**, **t-SNE** y **UMAP**.

2.8. Técnicas de Agrupamiento (*Clustering*)

El *clustering* agrupa objetos de tal manera que los miembros de un mismo grupo son más similares entre sí que con los de otros grupos [15].
En las siguientes subsecciones se describen los algoritmos de agrupamiento utilizados en este trabajo, categorizados según su enfoque metodológico.

2.8.1. Algoritmos Basados en Centroides y Jerárquicos

K-Means [16]

$$\begin{aligned} \text{Asignación:} \quad & S_i^{(t)} = \{x_p : \|x_p - \mu_i^{(t)}\|^2 \leq \|x_p - \mu_j^{(t)}\|^2, \forall j\} \\ \text{Actualización:} \quad & \mu_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j \end{aligned} \tag{2.15}$$

Algoritmo iterativo que alterna entre dos pasos fundamentales descritos en la Ecuación 2.15. Primero, en la fase de asignación, cada punto x_p se vincula al clúster S_i cuyo centroide μ_i es el más cercano. Segundo, en la fase de actualización, se recalculan los centroides como el promedio aritmético de los puntos asignados a cada clúster. Este ciclo se repite hasta la convergencia.

Agglomerative Clustering [17]

$$d(C_u \cup C_v, C_w) = \min\{d(C_u, C_w), d(C_v, C_w)\} \tag{2.16}$$

Enfoque jerárquico ascendente que inicia con cada punto como un clúster individual y los fusiona iterativamente. La Ecuación 2.16 muestra la regla de actualización de distancias para el criterio de *Single Linkage* (Enlace Simple): la distancia entre un nuevo clúster fusionado ($C_u \cup C_v$) y cualquier otro clúster C_w es la mínima distancia que tenían sus componentes originales. Esto permite al algoritmo seguir la "conectividad" de los datos, formando dendrogramas.

2.8.2. Algoritmos Basados en Densidad

DBSCAN [18]

$$N_\epsilon(p) = \{q \in D \mid \text{dist}(p,q) \leq \epsilon\}$$
$$\text{Tipo}(p) = \begin{cases} \text{Núcleo} & \text{si } |N_\epsilon(p)| \geq \text{MinPts} \\ \text{Borde} & \text{si } |N_\epsilon(p)| < \text{MinPts} \wedge p \in N_\epsilon(\text{Núcleo}) \\ \text{Ruido} & \text{en otro caso} \end{cases} \tag{2.17}$$

Algoritmo basado en la clasificación de puntos según su densidad local, como se detalla en la Ecuación 2.17. Se define una vecindad N_ϵ alrededor de un punto p. Si esta vecindad contiene suficientes puntos ($MinPts$), p inicia un clúster (Núcleo). El algoritmo expande los clústeres conectando núcleos vecinos y asignando puntos de borde, dejando los puntos aislados como ruido.

OPTICS [19]

$$\text{CD}(o) = \text{dist}(o, o_{\text{MinPts}})$$
$$\text{RD}(p,o) = \max(\text{CD}(o), \text{dist}(p,o)) \tag{2.18}$$

Generalización de DBSCAN para densidades variables. En lugar de producir una agrupación fija, calcula un ordenamiento basado en dos valores: la Distancia de Núcleo (CD), que es la distancia al $MinPts$-ésimo vecino más cercano, y la Distancia de Alcanzabilidad (RD) mostrada en la Ecuación 2.18. La RD indica cuán "facil" es alcanzar el punto p desde el núcleo o, permitiendo visualizar la estructura de clústeres como valles en un gráfico de alcanzabilidad.

HDBSCAN [20]

$$d_{\text{mreach}}(a,b) = \max\{\text{core}_k(a), \text{core}_k(b), d(a,b)\}$$
$$\text{core}_k(x) = d(x, \text{k-vecino}(x)) \tag{2.19}$$

Extensión jerárquica robusta de **DBSCAN**. El paso fundamental del algoritmo, mostrado en la Ecuación 2.19, es la transformación del espacio mediante la **Distancia de Alcanzabilidad Mutua** (d_{mreach}). Esta métrica empuja los puntos en regiones de baja densidad (donde core_k es alto) alejándolos entre sí, mientras mantiene cercanos a los puntos en zonas de alta densidad. Sobre este nuevo espacio métrico, el algoritmo construye un árbol de expansión mínima (MST) para simplificar la jerarquía de clústeres.

En la Figura 2.12 se comparan los resultados de varios algoritmos de *clustering* aplicados al dataset Digits [14] tras un preprocesamiento con **PCA** a 30 componentes y visualización con **UMAP** 2D. Se observa cómo cada método identifica diferentes estructuras en los datos, reflejando sus enfoques únicos para definir y detectar clústeres.

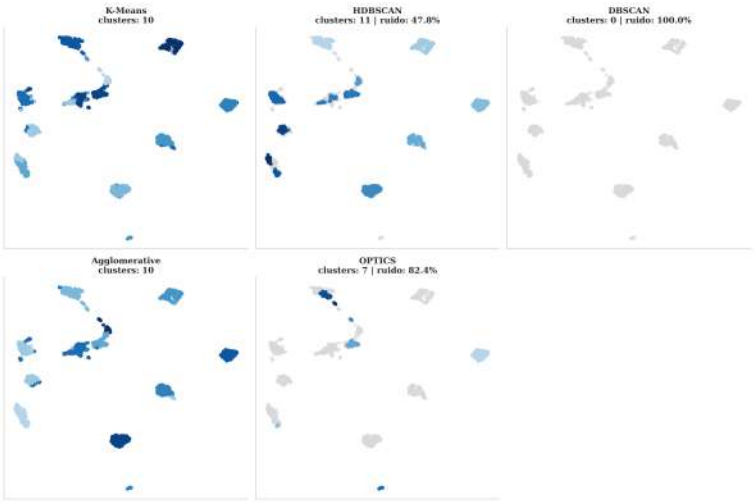


Figura 2.12: Comparación de métodos de *clustering* aplicada al dataset Digits [14].

2.9. Métricas de Evaluación de *Clustering*

La evaluación en escenarios no supervisados presenta desafíos únicos [21]. Se distinguen dos tipos de métricas utilizadas en esta investigación, las cuales fueron utilizadas para la optimización automática de los hiperparámetros de los algoritmos de *clustering*.

2.9.1. Métricas Internas (Sin Etiquetas)

Utilizadas para evaluar la calidad de la estructura sin conocer la verdad fundamental.

Índice de Silueta (Silhouette Score) [22]

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{2.20}$$

Mide la calidad de agrupamiento para cada punto i en un rango de $[-1, 1]$. Como se define en la Ecuación 2.20, se compara la cohesión $a(i)$ (distancia media intra-clúster) contra la separación $b(i)$ (distancia media al clúster vecino más cercano). Valores cercanos a 1 indican que la muestra está lejos de los clústeres vecinos.

Índice de Davies-Bouldin [23]

$$\text{DB} = \frac{1}{k} \sum_{i=1}^{k} \max_{j \neq i} \left(\frac{\sigma_i + \sigma_j}{d(\mu_i, \mu_j)} \right) \tag{2.21}$$

Evalúa la relación entre la dispersión intra-clúster y la separación inter-clúster. La Ecuación 2.21 busca el peor caso de similitud para cada clúster i respecto a otro clúster j. Aquí, σ representa la dispersión promedio de los puntos respecto a su centroide y $d(\mu_i, \mu_j)$ es la distancia entre los centroides. Un valor menor indica una mejor partición (grupos compactos y separados).

Índice de Calinski-Harabasz [24]

$$\text{CH} = \frac{\text{Tr}(B_k)}{\text{Tr}(W_k)} \times \frac{N-k}{k-1} \tag{2.22}$$

Conocido como el criterio de razón de varianza. La Ecuación 2.22 calcula la razón entre la dispersión inter-clúster (representada por la traza de la matriz B_k) y la dispersión intra-clúster (traza de la matriz W_k), normalizada por el número de muestras N y grupos k. Un valor más alto corresponde a clústeres mejor definidos y separados.

2.9.2. Métricas Externas (Con Etiquetas)

Utilizadas para validar la eficacia final del sistema comparando los resultados contra un conjunto de datos etiquetado de control (*Ground Truth*).

Adjusted Rand Score (ARI) [25]

$$\text{ARI} = \frac{\text{RI} - E[\text{RI}]}{\max(\text{RI}) - E[\text{RI}]} \tag{2.23}$$

Mide la similitud entre las asignaciones predichas y las etiquetas reales, corregida por el azar. El índice base (RI) calcula la proporción de pares de puntos que son tratados consistentemente en ambas particiones. La ecuación 2.23 ajusta este valor restando el valor esperado $E[\text{RI}]$ (agrupamiento aleatorio), resultando en una métrica donde 0.0 indica aleatoriedad y 1.0 coincidencia perfecta.

Normalized Mutual Information (NMI) [26]

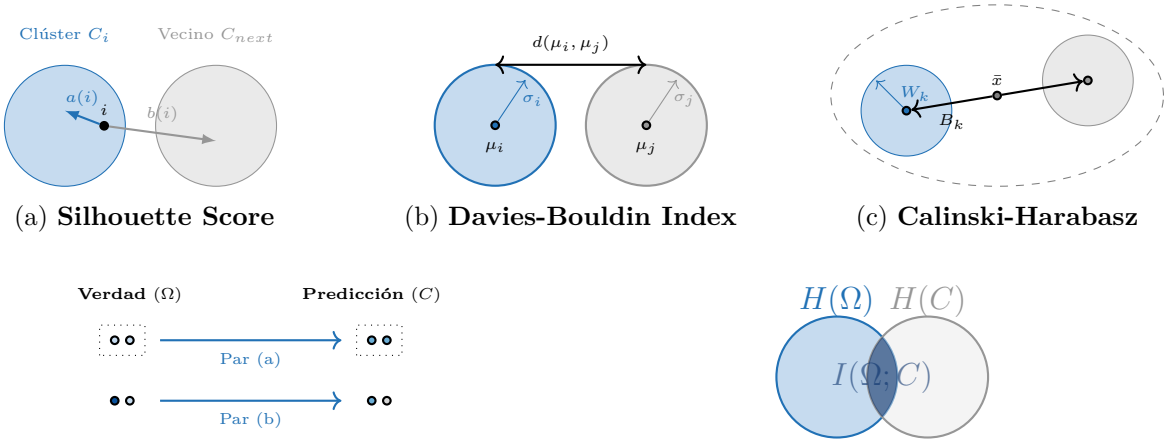
$$\text{NMI}(\Omega, C) = \frac{2 \cdot I(\Omega; C)}{H(\Omega) + H(C)} \tag{2.24}$$

Cuantificación basada en la teoría de la información. La Ecuación 2.24 mide la Información Mutua $I(\Omega; C)$ entre las etiquetas verdaderas Ω y los clústeres C, normalizada por la media aritmética de sus entropías H. Este valor indica cuánto reduce el conocimiento de la asignación de clústeres la incertidumbre sobre las clases verdaderas.

En la figura 2.13 se ilustran conceptualmente las métricas internas y externas de evaluación de *clustering* descritas previamente.

2.10. Adaptación de Dominio y *Fine-Tuning*

Dado que los modelos base se entrenan en dominios generales, es crucial adaptarlos al dominio específico de vehículos para mitigar el *domain shift* [9].
El proceso de *Fine-Tuning* implica descongelar total o parcialmente los pesos de la red pre-entrenada y continuar el entrenamiento con datos específicos a una tasa de aprendizaje reducida. En modelos multimodales como **CLIP**, este ajuste puede realizarse en el codificador de imagen, en el de texto, o en ambos, ya sea de forma conjunta o secuencial [27].



(a) **Silhouette Score** (b) **Davies-Bouldin Index** (c) **Calinski-Harabasz**

(d) **Adjusted Rand Index** (ARI) (e) **Normalized Mutual Information** (NMI)

Figura 2.13: Representación esquemática de las métricas de evaluación.

2.10.1. Funciones de Pérdida Supervisadas

Para potenciar la capacidad discriminativa de los *embeddings* y optimizarlos para el *clustering* no supervisado, se exploró un amplio espectro de **funciones de pérdida supervisadas**. Estas pérdidas se dividen en dos categorías principales:

Pérdida de Clasificación Tradicional

$$L_{CE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log(\hat{y}_{i,c}) \tag{2.25}$$

Se utilizó la **Cross-Entropy Loss** [28] para establecer una base de rendimiento (*baseline*) en la tarea de clasificación de modelos. Esta pérdida optimiza las probabilidades de salida en la última capa de la red. Donde en la Ecuación 2.25 N es el número de muestras, C es el número de clases, $y_{i,c}$ es la etiqueta de verdad fundamental (1 si la clase c es la correcta para la muestra i, 0 en caso contrario) y $\hat{y}_{i,c}$ es la probabilidad predicha para la clase c.
Sin embargo, esta pérdida no garantiza la estructura óptima del espacio de *embeddings* para el *clustering*.

Pérdidas de *Metric Learning* Diseñadas específicamente para mejorar la calidad del espacio latente, forzando explícitamente la **compactación intra-clase** y la **separabilidad inter-clase** de los *embeddings* [29]. Las pérdidas exploradas incluyen:

Contrastive Loss [30]

$$L_C = \sum_i \left(y_i d^2 + (1-y_i)\max(0, m-d)^2\right) \tag{2.26}$$

Pérdida que opera minimizando la distancia (d) entre pares positivos ($y_i = 1$) y maximizando la distancia (sujeta a un margen m) entre pares negativos ($y_i = 0$).

NTXent Loss (Normalized Temperature-Scaled Cross-Entropy) [31]

$$L_{\text{NTXent}} = -\frac{1}{N}\sum_{i=1}^{N}\log\left(\frac{e^{z_i \cdot z_{\text{pos}}/\tau}}{\sum_{j\in\text{All}} e^{z_i \cdot z_j/\tau}}\right) \tag{2.27}$$

Es una forma de pérdida contrastiva ampliamente utilizada en modelos como **CLIP**, que se enfoca en alinear *embeddings* positivos y dispersar negativos utilizando la similitud del coseno y una temperatura τ.

Triplet Loss [32]

$$L_{\text{Triplet}} = \max(0, d(\mathbf{a}, \mathbf{p}) - d(\mathbf{a}, \mathbf{n}) + \alpha) \tag{2.28}$$

Busca asegurar que la distanci en el espacio latente de los *embeddings* de un anclaje ($\mathbf{a}$) y un positivo ($\mathbf{p}$) sea menor que la distancia entre el anclaje y un negativo ($\mathbf{n}$) por un margen fijo α.

ArcFace Loss [29]

$$L_{\text{ArcFace}} = -\frac{1}{N}\sum_{i=1}^{N}\log\left(\frac{e^{s\cos\left(m+\theta_{y_i}\right)}}{e^{s\cos\left(m+\theta_{y_i}\right)} + \sum_{j\neq y_i} e^{s\cos(\theta_j)}}\right) \tag{2.29}$$

Una pérdida basada en el margen angular que penaliza el ángulo (θ) entre el *embedding* de la muestra y el centroide de la clase correcta, forzando límites de decisión claros con un factor de escala s y margen m.

Multi-Similarity Loss [33]

$$L_{\text{MS}} = \frac{1}{N}\sum_{i}\left[\frac{1}{\beta}\log\left(1 + \sum_{p\in P_i} e^{-\beta(S_{i,p}-\lambda)}\right) + \frac{1}{\alpha}\log\left(1 + \sum_{n\in N_i} e^{\alpha(S_{i,n}-\lambda)}\right)\right] \tag{2.30}$$

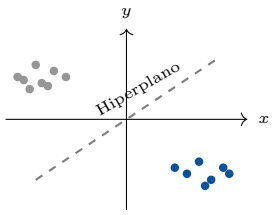
Perdida avanzada que asigna pesos dinámicamente a pares de *embeddings* positivos y negativos difíciles.

Si bien la **Cross-Entropy Loss** establece una base de rendimiento en clasificación, su formulación no garantiza la estructura geométrica óptima del espacio latente requerida para tareas no supervisadas. En contraste, las funciones de *Metric Learning* moldean explícitamente la distribución de los *embeddings*, favoreciendo la compacidad intra-clase y la separabilidad inter-clase. En consecuencia, la evaluación comparativa de estas funciones resulta crítica para determinar la estrategia de entrenamiento (*fine-tuning*) idónea, pues la calidad de las representaciones vectoriales aprendidas impacta directamente en la eficacia del agrupamiento final para la identificación vehicular [7].

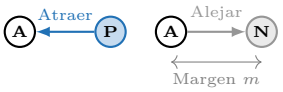
La Figura 2.14 resume visualmente las diferencias conceptuales entre las funciones de pérdida estudiadas.

2.11. Aplicaciones de IA en Identificación Vehicular

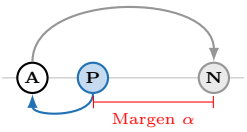
La identificación precisa de vehículos ha evolucionado desde enfoques puramente visuales hacia arquitecturas profundas y multimodales. A continuación, se analiza cómo los mecanis-



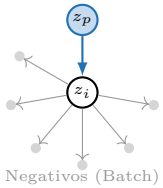
(a) **Cross-Entropy** (Clasificación)



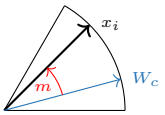
(b) **Contrastive Loss**



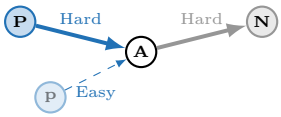
(c) **Triplet Loss**



(d) **NT-Xent Loss**



(e) **ArcFace Loss**



(f) **Multi-Similarity Loss**

Figura 2.14: Comparación visual de funciones de pérdida.

mos de atención y los modelos de visión-lenguaje han transformado este campo.

2.11.1. Mecanismos de Atención en Vision Transformers

La literatura reciente demuestra la eficacia de los **Transformers**, sus representaciones profundas (*embeddings*) y el *clustering* en tareas vehiculares complejas. Metodologías como **Patch-to-Cluster attention** han mejorado significativamente la interpretabilidad de los **ViTs** al agrupar parches visuales semánticamente relacionados [34], mientras que enfoques como **MD-ViT** han abordado la re-identificación vehicular aprovechando detalles de múltiples niveles de abstracción [35].

2.11.2. Discriminación de Grano Fino y *Near-Duplicates*

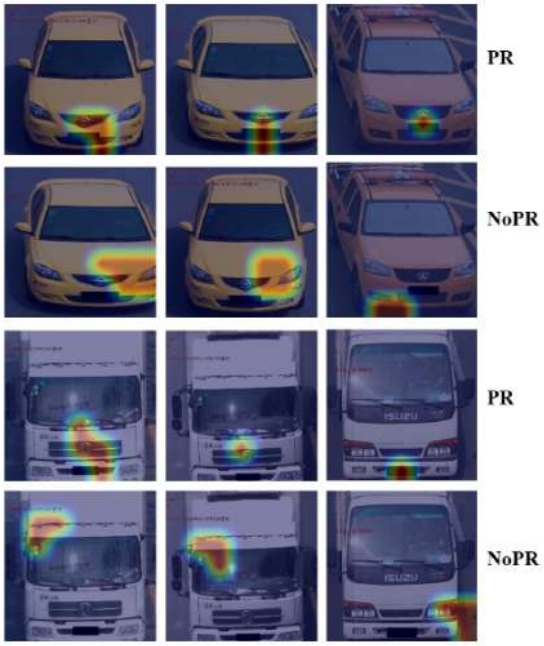
Un desafío fundamental en este dominio es la distinción entre instancias visualmente casi idénticas, fenómeno conocido como *near-duplicates*. Estas diferencias suelen presentarse entre vehículos del mismo modelo que varían sutilmente debido al año de fabricación, nivel de equipamiento (*trim level*) o decoraciones personalizadas (calcomanías, inspecciones). He et al. [36] abordaron esta problemática específica, demostrando que las características globales extraídas por **CNNs** estándar a menudo resultan insuficientes para capturar estas sutilezas.

En su trabajo *Part-regularized Near-duplicate Vehicle Re-identification*, proponen que la clave para diferenciar estas instancias reside en forzar a la red a focalizar su atención en partes locales distintivas —como los faros, la parrilla frontal o el diseño de las ventanas— en lugar de depender únicamente de la geometría global del vehículo. Esta hipótesis es crucial para el presente estudio, pues sugiere que las redes neuronales poseen la capacidad intrínseca de discriminar entre sub-clases de vehículos. Sin embargo, esta capacidad discriminativa depende intrínsecamente de que el mecanismo de atención posea la granularidad suficiente para ponderar características locales críticas sobre la topología global.

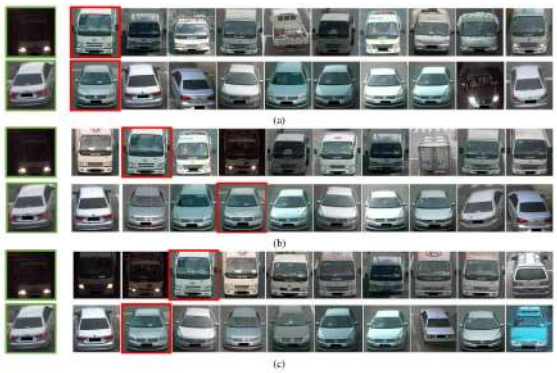
Como se evidencia en la Figura 2.15, las redes entrenadas con regularización de partes logran activar mapas de atención más nítidos sobre zonas discriminativas, mejorando la recuperación de instancias correctas frente a modelos base.

(a) Desafío visual: pares *near-duplicates*.

(b) Extracción de regiones locales clave.



(c) Mapas de atención: PR vs NoPR.



(d) Comparativa de resultados de recuperación.

Figura 2.15: Análisis de la re-identificación de vehículos *near-duplicates* mediante atención local. (2.15.a) Ejemplos de vehículos casi idénticos donde la diferenciación depende de detalles sutiles (calcomanías, parrillas). (2.15.b) Detección de partes locales discriminativas (faros, ventanas) mediante la red propuesta. (2.15.c) Mapas de calor (*Attention Maps*) comparando el modelo base (NoPR) vs. el regularizado (PR); note cómo PR focaliza nítidamente la atención en características únicas como logotipos o parrillas. (2.15.d) Resultados cualitativos de recuperación (Ranking); el modelo propuesto (filas inferiores) logra corregir errores de falsos positivos presentes en el baseline. Adaptado de [36]. ©2019 IEEE.

2.11.3. Modelos de Visión-Lenguaje (VLM) en Re-Identificación

Una evolución natural de los métodos anteriores es la **Re-Identificación Vehicular basada en modelos VLM**, donde se explota la capacidad multimodal para mejorar la robustez frente a cambios de perspectiva o iluminación. Un referente clave es el trabajo de **CLIP-ReID** [37], que propone explotar las representaciones visuales y lingüísticas conjuntas sin necesidad de etiquetas de texto concretas durante la inferencia.

En esta misma línea, otros trabajos han explorado el **aprendizaje auto-supervisado** para refinar los *embeddings*. El método **SVLL-ReID** [38], por ejemplo, integra la auto-supervisión lingüística y visual en dos etapas, logrando que los *prompts* aprendibles y los rasgos visuales sean mutuamente más discriminativos. De igual manera, se ha validado el uso de pseudo-etiquetas de texto generadas automáticamente para entrenar modelos **VLM** robustos [39].

La Figura 2.16 ilustra cómo **CLIP-ReID** transfiere el conocimiento multimodal de **CLIP** a la tarea de identificación específica.

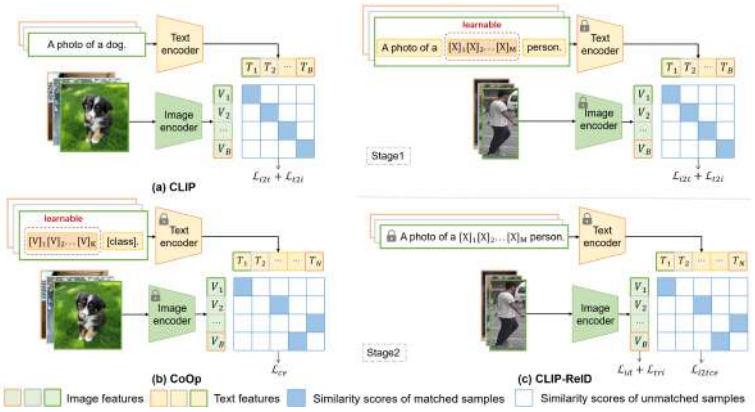


Figura 2.16: Arquitectura del método **CLIP-ReID** que optimiza la proyección visual utilizando *tokens* de texto aprendibles. Adaptado de [37]. Licencia CC BY 4.0.

2.12. De la Lectura de Patentes (ANPR) al Reconocimiento de Modelos (VMR)

Si bien el Reconocimiento Automático de Patentes (**ANPR**) es una tecnología madura, la identificación precisa del modelo del vehículo (*Vehicle Model Recognition* o **VMR**) representa un desafío de orden superior debido a la gran variabilidad intra-clase y la, a veces sutil, variabilidad inter-clase.

2.12.1. Evolución hacia el Reconocimiento de Atributos

Los sistemas ANPR clásicos se limitaban a la segmentación y reconocimiento óptico de caracteres (OCR). Sin embargo, la integración de redes neuronales convolucionales (**CNN**) permitió extender estas capacidades hacia la clasificación de atributos visuales (color, tipo de vehículo, marca) [40].

Esta evolución establece un precedente teórico sólido para la presente investigación: si las arquitecturas profundas logran extraer características robustas para leer patentes en condi-

ciones de iluminación variable, oclusiones y baja resolución, es inferible que modelos más avanzados —como los **Transformers** (**ViT**)— poseen la capacidad de capturar las características de grano fino necesarias para discriminar entre modelos específicos de vehículos, superando las limitaciones de las **CNN** tradicionales en tareas de *fine-grained classification*.

2.12.2. Soluciones Comerciales y sus Limitaciones

En el ámbito industrial, existen soluciones que han llevado el aprendizaje profundo a entornos productivos. Estas herramientas validan la factibilidad técnica de la identificación vehicular, aunque operan bajo paradigmas mayoritariamente supervisados y propietarios:

Jenoptik [41] Implementa algoritmos de clasificación propietarios que integran la lectura de matrículas con el reconocimiento de características del vehículo. Su enfoque demuestra que la fusión de datos visuales mejora la tasa de acierto en condiciones ambientales adversas.

ParkPow [42] Ofrece sistemas de gestión basados en *Deep Learning* para la clasificación de vehículos. Su arquitectura permite inferir la marca y el tipo de carrocería, aunque su dependencia de bases de datos pre-entrenadas limita su flexibilidad ante modelos nuevos o poco comunes.

Sighthound ALPR+ [43] Utiliza redes neuronales optimizadas para el procesamiento en tiempo real, capaces de extraer atributos vehiculares simultáneamente a la lectura de la patente. Representa el estado del arte en eficiencia computacional para sistemas de vigilancia.

2.13. Brecha en el Estado del Arte y Oportunidad de Investigación

El análisis integral de los fundamentos teóricos y el estado del arte expuesto en este capítulo revela una convergencia tecnológica significativa. La revisión de arquitecturas avanzadas como los **Transformers** y modelos multimodales como **CLIP**, junto con técnicas de proyección de variedades (*Manifold Learning*) como **UMAP** y **t-SNE**, sugiere que *las representaciones latentes actuales poseen una riqueza semántica suficiente para discriminar sutilezas visuales sin necesidad de supervisión explícita*. Esta base teórica es la que sustenta la hipótesis de que es factible transitar desde enfoques rígidos hacia sistemas adaptativos.

Respecto al panorama industrial, si bien existen soluciones comerciales robustas, estas suelen operar bajo paradigmas que requieren la definición previa de las clases a identificar. Esta dependencia de catálogos cerrados impone restricciones operativas críticas: la dificultad de escalar ante la constante aparición de nuevos modelos vehiculares y la incapacidad inherente de gestionar el reconocimiento de instancias "desconocidas" (*Closed-Set limitation*).

Paralelamente, la literatura académica reciente ha comenzado a desafiar estas limitaciones. Investigaciones de vanguardia en re-identificación vehicular y aprendizaje métrico han demostrado que es posible extraer características de grano fino mediante mecanismos de atención y aprovechar el conocimiento multimodal para mejorar la generalización. Estos trabajos previos proporcionan *insights* valiosos y validan empíricamente que los modelos profundos pueden ir más allá de la simple clasificación, capturando la identidad visual única de los objetos.

En este contexto, la presente memoria se plantea como una culminación de estas líneas de investigación. Se propone *un cambio de paradigma hacia un enfoque* ***no supervisado y multimodal*** *que sintetiza la capacidad descriptiva de los modelos VLM con la exploración estructural del clustering jerárquico*. Esta propuesta busca resolver la limitación del *Open-Set Recognition*, demostrando que la integración de estas tecnologías permite no solo identificar lo conocido, sino también descubrir y segregar automáticamente nuevas clases de vehículos, dotando al sistema de una capacidad de generalización inédita en aplicaciones de monitoreo vehicular.

Capítulo 3

Metodología

El presente capítulo detalla la metodología empleada para el desarrollo de un sistema capaz de identificar modelos de vehículos utilizando representaciones profundas y técnicas de agrupamiento no supervisado. El enfoque metodológico combina técnicas avanzadas de visión por computadora y aprendizaje automático, estructurándose en un flujo de trabajo secuencial que abarca desde la ingestión y curación de datos hasta la validación de la capacidad de generalización en escenarios de mundo abierto.

La estrategia de investigación se divide en cuatro etapas fundamentales: (1) La definición de una estrategia de datos robusta para manejar el desbalance de clases y simular condiciones reales; (2) El ajuste fino (*fine-tuning*) de diversas arquitecturas de redes de *deep learning* (unimodales y multimodales) para especializar sus representaciones; (3) La optimización de técnicas de reducción de dimensionalidad para destilar la estructura latente; y (4) La implementación y evaluación de algoritmos de *clustering* capaces de operar sin conocer el número de clases *a priori*.

3.1. Implementación

El diseño del proyecto se basa en la programación modular y orientada a objetos (OOP), lo que resulta en un *pipeline* estructurado en varias etapas independientes pero interconectadas.

El repositorio del proyecto está organizado en módulos clave que separan la lógica de procesamiento de datos, la definición de arquitecturas de modelos y los flujos de ejecución (*pipelines*). A continuación, en la Figura 3.1, se detalla la estructura de directorios y la función de cada componente:

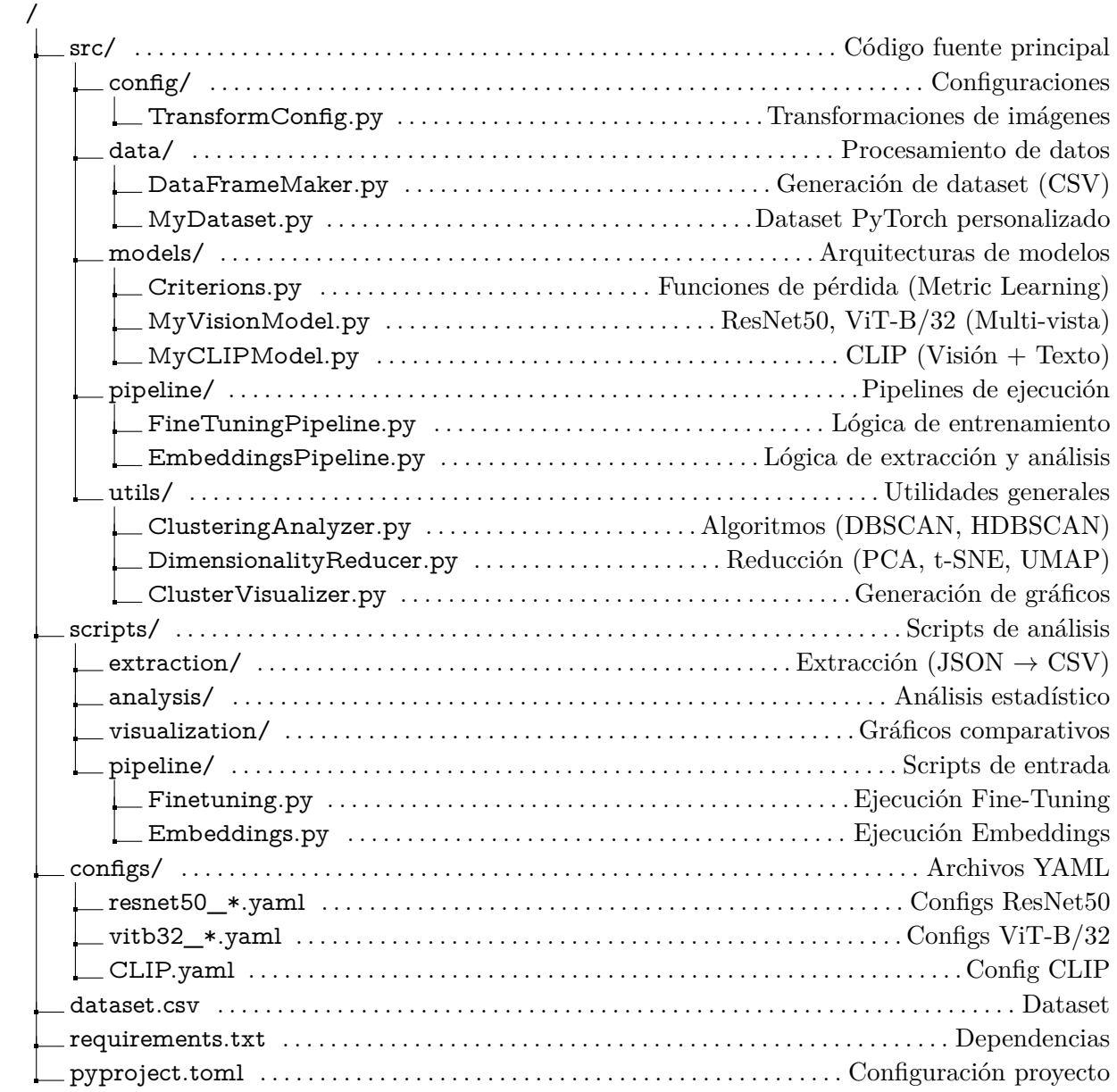


Figura 3.1: Estructura de directorios y organización del código fuente del proyecto.

Cabe destacar que carpetas auxiliares como `results/`, `dataset/image/` y `.venv/` han sido omitidas del control de versiones para mantener la ligereza del repositorio, gestionándose de manera local-remota a traves de *Google Drive*.

3.1.1. Herramientas Computacionales

Para asegurar la reproducibilidad, eficiencia y escalabilidad de los experimentos, se seleccionó un conjunto de herramientas de software e infraestructura modernas para el *stack* tecnológico. A continuación, se describen las tecnologías clave utilizadas:

Python Lenguaje de programación principal del proyecto. Su extenso ecosistema de librerías para ciencia de datos e Inteligencia Artificial lo convierte en la base fundamental para la implementación de los *pipelines* de procesamiento y entrenamiento.

PyTorch *Framework* de aprendizaje profundo utilizado para la definición, entrenamiento y evaluación de las redes neuronales. Se seleccionó por su flexibilidad, su capacidad de diferenciación automática (**Autograd**) y su eficiencia en el uso de aceleración por GPU.

Scikit-learn Librería empleada para la implementación de algoritmos clásicos de *Machine Learning*. En este trabajo, es fundamental para las etapas de reducción de dimensionalidad (**PCA**) y para la ejecución de algoritmos de *clustering* y cálculo de métricas de validación.

Optuna Framework de optimización de hiperparámetros automatizado de nueva generación. En esta investigación, se utilizó su algoritmo de muestreo bayesiano (**TPE**) y sus estrategias de poda (*pruning*) para la búsqueda eficiente de las configuraciones óptimas en los algoritmos de reducción de dimensionalidad y *clustering*.

UV Gestor de paquetes y proyectos para Python, escrito en Rust. Se utilizó para gestionar las dependencias y entornos virtuales del proyecto de manera determinista y veloz, reemplazando a herramientas tradicionales como `pip` o `poetry` para optimizar los tiempos de configuración.

Git y GitHub Sistema de control de versiones distribuido y plataforma de alojamiento. Se utilizaron para mantener el historial de cambios del código fuente, facilitar el desarrollo modular y permitir la gestión remota del proyecto.

Google Cloud Platform (GCP) Infraestructura en la nube utilizada para la ejecución de los experimentos computacionalmente costosos. Específicamente, se aprovisionaron 2 instancias de Máquinas Virtuales (VM) una equipada con GPU para acelerar el proceso de *fine-tuning* de los modelos profundos y otra equipada con una CPU de alto rendimiento para la ejecución del análisis de los *embeddings* profundos (etapas de *reducción de dimensionalidad* y *clustering*). Las especificaciones técnicas de las instancias se presentan en la Tabla 3.1.

Tabla 3.1: Especificaciones técnicas de las instancias de Google Cloud Platform utilizadas en los experimentos.

Instancia	CPU	Memoria RAM	GPU	Almacenamiento
GPU-VM	n1-standard-4 (4 CPU virtuales)	15 GB	NVIDIA Tesla T4 (16 GB de VRAM)	100 GB SSD
CPU-VM	c4-standard-16 (16 CPU virtuales)	60 GB	N/A	100 GB SSD

JSON y CSV Formatos de intercambio de datos. `JSON` se utilizó para el almacenamiento estructurado de configuraciones y metadatos de los experimentos, mientras que `CSV` se empleó para la persistencia tabular de los resultados métricos, facilitando su posterior lectura y análisis estadístico.

3.1.2. Etapa 0: Estrategia de Datos y Preprocesamiento

Antes de ejecutar las etapas de entrenamiento, se implementó un flujo de trabajo riguroso para la curación, transformación y estructuración de los datos. Esta etapa es crítica para garantizar que los modelos no solo aprendan a clasificar imágenes limpias, sino que sean robustos ante las variaciones inherentes a las imágenes reales y a la distribución desbalanceada de las clases.

3.1.2.1. Generación y Curación del Dataset

El conjunto de datos *Comprehensive Cars* (CompCars [44]) presenta una estructura compleja con múltiples vistas no relevantes para el proposito de esta investigación (e.g., vistas aéreas o laterales). Para normalizar el acceso a los datos, se implementó el módulo `DataFrameMaker`. Este módulo consolida los metadatos dispersos y genera un archivo maestro CSV que actúa como fuente de verdad para el entrenamiento, filtrando exclusivamente las imágenes con puntos de vista **frontal** (*Front*) y **trasero** (*Rear*).

El esquema de datos resultante, diseñado para facilitar la carga eficiente en los *pipelines* de PyTorch, incluye los siguientes campos clave:

- `id`: Identificador único de la imagen.
- `image_path`: Ruta relativa al archivo de imagen.
- `released_year`: Año de lanzamiento, crucial para distinguir generaciones.
- `viewpoint`: Punto de vista normalizado (Front/Rear).
- `bbox`: Coordenadas del cuadro delimitador para recortes precisos.
- `make`, `model`, `type`: Jerarquía de etiquetas del vehículo.

La Tabla 3.2 contiene un extracto del dataset final utilizado en los experimentos.

Tabla 3.2: Extracto del dataset final utilizado en los experimentos. Corresponde a instancias del modelo *Skoda Haorui* de diferentes años y vistas.

id	image_path	released_year	viewpoint	bbox	make	model	type
fc8f3f81654df1	dataset/image/95/915/2013/fc8f3f81654df1.jpg	2013	front	[179.0, 143.0, 733.0, 527.0]	Skoda	Haorui	sedan
04b1b1e64d2f08	dataset/image/95/915/2014/04b1b1e64d2f08.jpg	2014	front	[73.0, 51.0, 838.0, 618.0]	Skoda	Haorui	sedan
e0e8974bb29b5b	dataset/image/95/915/2014/e0e8974bb29b5b.jpg	2014	rear	[97.0, 47.0, 810.0, 626.0]	Skoda	Haorui	sedan
66cde8f8470235	dataset/image/95/915/2012/66cde8f8470235.jpg	2012	front	[146.0, 118.0, 775.0, 588.0]	Skoda	Haorui	sedan
ff28f37fd6f1f3	dataset/image/95/915/2012/ff28f37fd6f1f3.jpg	2012	rear	[114.0, 76.0, 669.0, 527.0]	Skoda	Haorui	sedan
4b35c9433a9132	dataset/image/95/915/2009/4b35c9433a9132.jpg	2009	front	[91.0, 50.0, 780.0, 627.0]	Skoda	Haorui	sedan
75d8b969af28b7	dataset/image/95/915/2009/75d8b969af28b7.jpg	2009	rear	[63.0, 70.0, 842.0, 623.0]	Skoda	Haorui	sedan

3.1.2.2. Definición de Clases y Manejo de Long-Tail

Las clases se definieron mediante la combinación del modelo y el año de lanzamiento; por ejemplo, “Audi_A4_2010” y “Audi_A4_2015” constituyen clases distintas.

En la Tabla 3.3 se presenta un resumen estadístico de la distribución de las imágenes del dataset final. Es importante destacar que, para la configuración **Both** (Ambas Vistas), se reporta el universo de clases que poseen intersección de vistas (existencia simultánea en frontal y trasera), contabilizando el total de imágenes disponibles en dichas clases.

Tabla 3.3: Estadísticas generales del dataset por viewpoint. La fila Both representa la unión de imágenes de las clases compartidas.

Viewpoint	Clases	Imágenes	Media	Mediana	Desv. Est.	Rango
Front	3994	18,431	4.6	3.0	3.94	1–31
Rear	3689	13,513	3.7	3.0	3.06	1–20
Both	3598	31,073	3.8	3.0	3.43	1–20

Como se puede apreciar, uno de los desafíos principales del dataset es su distribución de cola larga (*long-tail*), donde una gran cantidad de clases posee muy pocas imágenes. Si bien se evaluaron distintas estrategias de balanceo, la solución más robusta y consistente fue implementar un filtrado basado en un umbral mínimo de imágenes (`min_images`), gestionado dinámicamente por el módulo `MyDataset`.

El criterio de filtrado establece que una clase se considera **Regular** únicamente si satisface el umbral `min_images` en la(s) vista(s) seleccionada(s) (Frontal, Trasera o Ambas). Específicamente, al utilizar la configuración de ambas vistas, el requisito numérico `min_images` debe cumplirse simultáneamente en ambas perspectivas. Esta restricción garantiza la existencia de suficientes **pares completos** para conformar lotes (*batches*) de entrenamiento balanceados, evitando así el desbalance modal. Adicionalmente, en el contexto de los modelos multimodales, esto facilita el uso de etiquetas textuales que describan explícitamente que la entrada corresponde a un par de imágenes.

Se definieron dos categorías de clases:

Clases Regulares ($\geq$ `min_images`) Clases con suficientes muestras (en cada vista requerida) para garantizar una división estadísticamente válida entre entrenamiento, validación y prueba. Estas son las únicas utilizadas para el ajuste fino supervisado.

Clases One-Shot ($<$ `min_images`) Clases con escasa representación que son excluidas del entrenamiento y reservadas exclusivamente para el conjunto de prueba. Esto permite evaluar la capacidad de generalización del sistema ante modelos nunca antes vistos (*Zero-Shot* o *Few-Shot*).

El impacto de este umbral (`min_images`) fue una variable de estudio en sí misma. La Figura 3.2 ilustra la distribución de imágenes disponibles por clase. Para la fila inferior ("Ambas Vistas"), se grafica la cantidad de **pares efectivos** (el mínimo entre imágenes frontales y traseras).

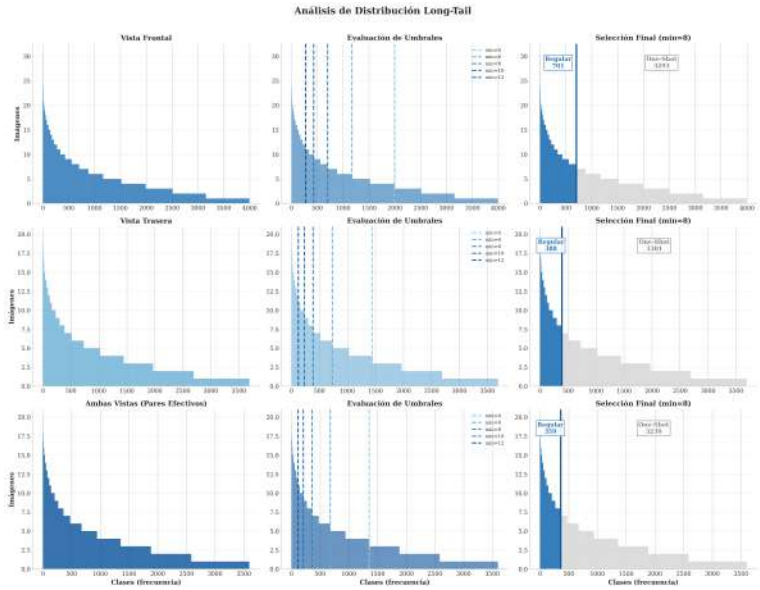


Figura 3.2: Análisis de distribución de clases y efecto del filtrado. Las filas representan las configuraciones de vista (Frontal, Trasera y Ambas). La primera columna corresponde a la distribución de las clases original sin filtrado. La segunda columna presenta los distintos umbrales estudiados y su efecto en la separación en clases Regulares y *One-Shot*. La tercera columna presenta el detalle de la distribución basal utilizada, obtenida al usar `min_images = 8`.

En la Tabla 3.4 se presentan las estadísticas resumidas de la distribución de clases resultante tras aplicar la lógica de filtrado.

Tabla 3.4: Distribución de clases según umbral min_images. Para la columna **Both**, se aplica el criterio de intersección lógica (AND) para contar pares válidos.

	Front			Rear			Both		
Umbral	**Reg.**	**One-S.**	**%**	**Reg.**	**One-S.**	**%**	**Reg.**	**One-S.**	**%**
4	1992	2002	49.9 %	1444	2245	39.1 %	1356	2242	37.7 %
6	1168	2826	29.2 %	730	2959	19.8 %	674	2924	18.7 %
8	701	3293	17.6 %	388	3301	10.5 %	359	3239	10.0 %
10	430	3564	10.8 %	229	3460	6.2 %	206	3392	5.7 %
12	275	3719	6.9 %	123	3566	3.3 %	113	3485	3.1 %

3.1.2.3. Transformaciones y Aumento de Datos

Un punto crucial de la implementación es la decisión de utilizar siempre los pesos preentrenados en **ImageNet** como punto de partida. Esta elección asegura que todos los modelos, tanto unimodales como multimodales, partan de una base de conocimiento común, permitiendo comparaciones justas y evitando sesgos introducidos por inicializaciones aleatorias. Para mantener la compatibilidad con estos pesos, las transformaciones aseguran siempre 3 canales de entrada; incluso al aplicar la conversión a escala de grises, la imagen se replica en los canales RGB para que las capas convolucionales iniciales operen correctamente.

En la Figura 3.3 se ilustra el flujo de preprocesamiento aplicado a cada imagen durante el entrenamiento.

Figura 3.3: Flujo de preprocesamiento aplicado a cada imagen durante el entrenamiento.

Con el objetivo de simular las condiciones adversas típicas de los sistemas reales, se implementó una estrategia de aumento de datos agresiva mediante el módulo `TransformConfig`.

Las técnicas incluyen de *data augmentation* utilizas incluyen:

Geometría Rotaciones, traslaciones y *Random Horizontal Flip*.

Apariencia *Color Jitter* (brillo/contraste), *Gaussian Blur* y conversión a escala de grises.

Regularización *Random Erasing* para forzar al modelo a no depender de características visuales completas.

En la Figura 3.4 se presentan ejemplos visuales de las transformaciones aplicadas.

Figura 3.4: Ejemplos de la aplicación de data augmentation sobre una imagen del dataset (imagen frontal del vehículo Skoda, modelo Haorui del año 2012). Se observa la variedad de transformaciones geométricas y de apariencia implementadas para robustecer el entrenamiento.

Es importante mencionar que la aplicación de estas transformaciones es dinámica y estocástica, generando variaciones únicas en cada época de entrenamiento para maximizar la diversidad del conjunto de datos.

3.1.3. Etapa 1: Ajuste Fino Supervisado (*Fine-Tuning*)

El objetivo de esta etapa es especializar las representaciones genéricas de los modelos pre-entrenados en el dominio de grano fino de los modelos vehiculares.

3.1.3.1. Selección de Arquitecturas

La selección de modelos se fundamentó en el estudio original de **CLIP** [7] para permitir una comparativa directa y rigurosa.

Modelos Base (ResNet50, ViT-B/32 y CLIP (ViT-B/32)) Fueron seleccionados por ser las arquitecturas más livianas evaluadas en el *paper* original de **CLIP**, estableciendo una línea base de eficiencia.

Modelos de Comparación (ResNet101, ViT-B/16 y CLIP (ViT-B/16)) Versiones de mayor complejidad utilizadas para evaluar si el aumento en profundidad o resolución justifica la carga computacional adicional.

El módulo `MyVisionModel` adapta dinámicamente estas arquitecturas, reemplazando la capa final (*head*) por una capa identidad para el aprendizaje métrico o una capa lineal para la clasificación, según el objetivo del experimento. Por otra parte, el módulo `MyCLIPModel` gestiona la integración del modelo multimodal **CLIP**, permitiendo la extracción conjunta de características visuales y textuales, y un ajuste fino *fine-tuning* por capas y fases.

La Tabla 3.5 detalla las dimensiones del espacio latente (*embedding size*) para cada arquitectura utilizada.

Tabla 3.5: Dimensiones de los vectores de características generados por las distintas arquitecturas.

Arquitectura	**Tipo**	**Dimensión del Embedding (D)**
ResNet50	CNN	2048
ResNet101	CNN	2048
ViT-B/32	Transformer	768
ViT-B/16	Transformer	768
CLIP (ViT-B/32)	Multimodal (basada en Transformer)	512
CLIP (ViT-B/16)	Multimodal (basada en Transformer)	512

3.1.3.2. Estrategia de Entrenamiento

El entrenamiento es orquestado por el módulo `FineTuningPipeline`. Dependiendo de la arquitectura, se aplicaron estrategias diferenciadas:

Modelos de Visión Pura Se aplicó un muestreo estándar para el entrenamiento con **Cross-Entropy**. Sin embargo, para el paradigma de **Metric Learning**, se utilizó una estrategia de muestreo $P \times K$ implementada mediante un *wrapper* personalizado de `MPerClassSampler`. Esto construye cada lote (*batch*) seleccionando P clases y K imágenes por clase, garantizando la presencia de agrupaciones fijas dentro de cada iteración, lo que es crucial para el aprendizaje métrico efectivo.

Modelos Multimodales (CLIP) Se diseñó una estrategia por fases para preservar el conocimiento del lenguaje. Se experimentó congelando selectivamente el codificador de imagen o de texto, y aplicando *layer-wise freezing* (congelamiento progresivo de capas) para determinar la configuración óptima que adapta el modelo al dominio vehicular sin perder su capacidad semántica.

El pipeline implementa técnicas avanzadas de regularización y optimización, incluyendo:

Optimización Diferenciada Separación de parámetros para aplicar *Weight Decay* únicamente a los pesos de convolución/lineales, excluyendo sesgos y capas de normalización.

Precisión Mixta Uso de *Automatic Mixed Precision* (AMP) con `GradScaler` para reducir el consumo de memoria y acelerar el entrenamiento en GPU.

Estabilidad Uso de *Gradient Clipping*, *WarmUp* y *Early Stopping*.

Respecto a las funciones de perdida de *Metric Learning* seleccionadas, estas se implementaron a traves de *wrappers* específicos en el módulo `MetricLearningLosses` de la librería `pytorch-metric-learning`, asegurando implementaciones numéricamente estables de:

Triplet Loss y Contrastive Loss Seleccionadas por su capcacidad de forzar distancias distancias relativas en el espacio Euclidiano.

ArcFace Seleccionada por su capacidad de introducir márgenes angulares que mejoran la discriminación.

NT-Xent y Multi-Similarity Seleccionadas para optimizar relaciones dentro del lote mediante aprendizaje contrastivo.

Mientras que la función de perdida de clasificación estandar utilizada (**Cross Entropy Loss**) fue implementada utilizando la versión nativa de PyTorch.

La Figura 3.5 ilustra la arquitectura modular de `MyVisionModel`.

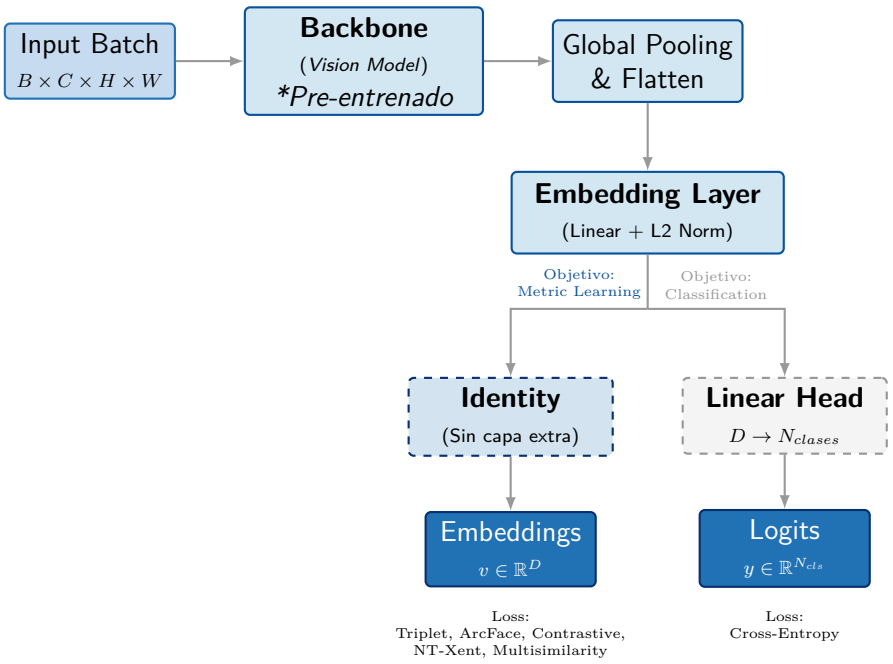


Figura 3.5: Arquitectura modular de `MyVisionModel`. El modelo comparte un *backbone* y una capa de *embedding* común, pero adapta dinámicamente su capa final (*Head*) según el objetivo de entrenamiento.

Para los modelos multimodales, se diseñó una lógica de generación de descripciones asimétrica. Durante el **entrenamiento**, se utiliza una descripción completa (ej. *The frontal*

view image of a BMW X5 vehicle from 2015 type SUV) para maximizar el aprendizaje de relaciones visuales-textuales. Sin embargo, durante la **validación y prueba**, se utiliza una descripción parcial (ej. "*The frontal view image of a BMW vehicle type SUV*"), ocultando intencionalmente el modelo específico. Esta restricción simula el escenario real donde el sistema desconoce el modelo exacto que debe identificar y obliga a la red a inferir la identidad basándose en las representaciones profundas aprendidas.

La Figura 3.6 ilustra este proceso de etiquetado asimétrico.

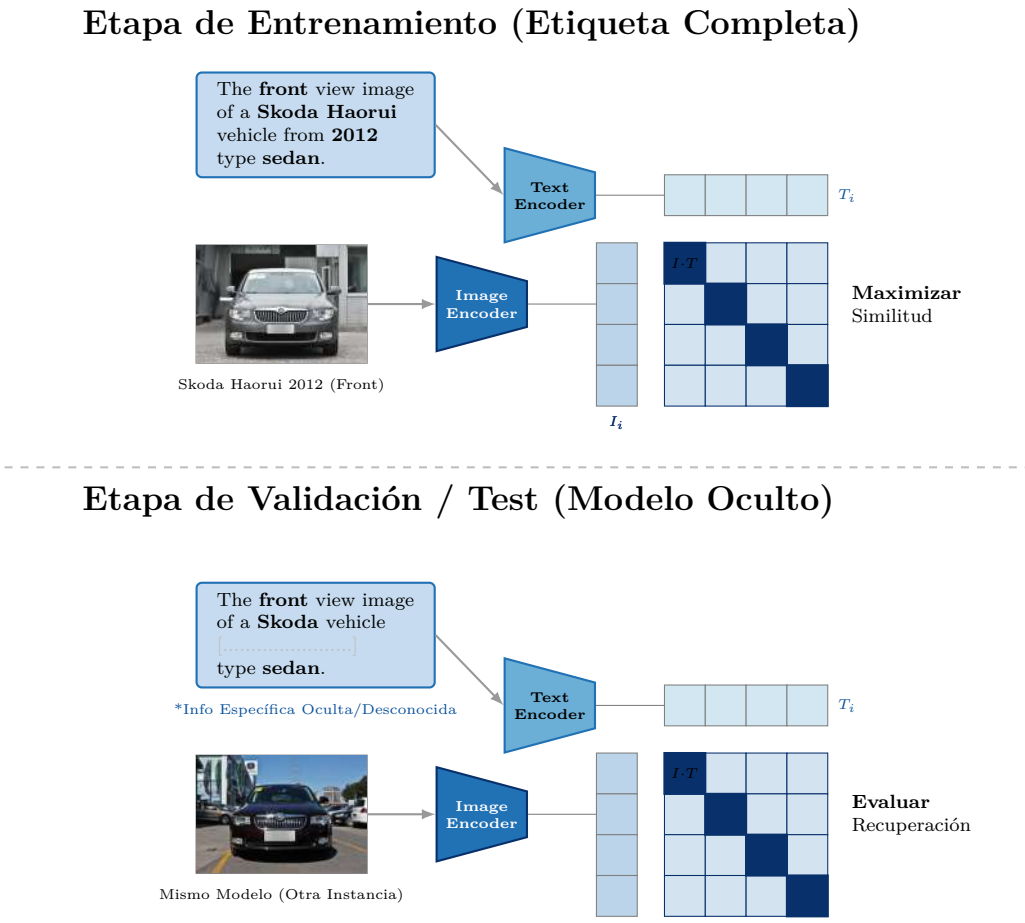


Figura 3.6: Diagrama de la estrategia asimétrica. (Arriba) Entrenamiento con etiquetas completas e imagen base. (Abajo) Evaluación simulando *Zero-Shot*, con una imagen diferente del mismo modelo y etiqueta de texto parcial.

Finalmente, la Figura 3.7 presenta la estrategia de entrenamiento secuencial implementada para **CLIP**.

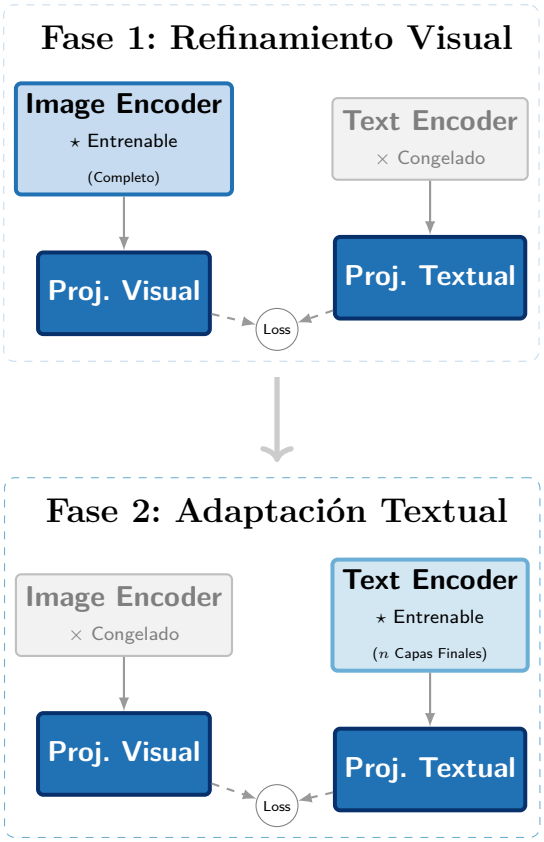


Figura 3.7: Estrategia de entrenamiento secuencial. (⋆) Indica componentes activos. (×) Indica componentes congelados.

Los Pseudocódigos A.1, A.2 y A.3 resumen la lógica de los ciclos de entrenamiento implementados para los modelos de visión pura y multimodales.

3.1.4. Etapa 2: Optimización de la Reducción de Dimensionalidad

Los *embeddings* generados por los modelos profundos poseen una alta dimensionalidad. Trabajar directamente en este espacio presenta dos desafíos críticos para el agrupamiento no supervisado: la "maldición de la dimensionalidad", que vuelve ineficaces las métricas de distancia tradicionales, y el elevado costo computacional. Por lo tanto, el objetivo de esta etapa es destilar la estructura latente, eliminando ruido y densificando la información semántica para facilitar la formación de clústeres compactos.

Esta etapa se aplicó sistemáticamente tanto a los *embeddings* base (*baseline*) como a los ajustados (*fine-tuned*) para realizar una comparación y analsis exhaustivo del impacto de la reducción de dimensionalidad en ambos escenarios.

3.1.4.1. Motivación y Selección de Algoritmos

Los algoritmos seleccionados para esta etapa corresponden a los definidos en el marco teórico y estado del arte. Su elección se sustenta en sus propiedades complementarias para la preparación de los datos:

PCA (Principal Component Analysis) Fue seleccionado por su capacidad de capturar la estructura global de los datos maximizando la varianza. Se implementó una variante de

PCA Incremental para gestionar eficientemente el uso de memoria cuando el volumen del dataset excede las capacidades de la RAM (lotes de 1000 muestras), permitiendo escalabilidad.

t-SNE y UMAP Fueron seleccionados por su capacidad para preservar la estructura local y las relaciones de vecindad no lineales (*Manifold Learning*), lo cual es crítico para distinguir modelos de vehículos visualmente similares que residen cerca en el espacio latente.

3.1.4.2. Estrategia de Optimización con Optuna

La eficacia de **PCA** ha sido ampliamente estudiada en diveras aplicaciones; por regla general se selecciona un valor a partir de la varianza explicada. En este caso dicho enfoque no es suficiente. Por otra parte, la eficacia de **t-SNE** y **UMAP** es altamente sensible a sus hiperparámetros (e.g., *perplexity* o *n_neighbors*). Una configuración subóptima puede crear clústeres artificiales o colapsar estructuras relevantes. Para mitigar esto, se implementó una búsqueda automatizada utilizando la librería **Optuna**.

El proceso de optimización se configuró de la siguiente manera:

Muestreador Se utilizó el **TPE (Tree-structured Parzen Estimator)**, un algoritmo bayesiano que modela la probabilidad de mejora y converge más rápido que la búsqueda aleatoria.

Presupuesto Se ejecutaron 50 intentos (*trials*) por algoritmo, utilizando **Median Pruning** para detener prematuramente aquellas configuraciones que no mostraban promesas de mejora en las primeras etapas.

Rangos de Búsqueda Los hiperparámetros se acotaron dinámicamente según el tamaño del dataset (N) y la dimensionalidad original (D) para asegurar que los valores sean matemáticamente válidos. El detalle de los rangos y distribuciones utilizadas se presenta en la Tabla 3.6.

Tabla 3.6: Espacio de búsqueda de hiperparámetros para algoritmos de reducción de dimensionalidad optimizados con Optuna. N representa el número de muestras y D el número de características (dimensiones) originales.

Algoritmo	**Hiperparámetro**	**Rango de Búsqueda / Configuración**
PCA	`n_components`	$[\max(16, \min(32, \min(D, N)/4)), \min(\min(D, N), 0.9D)]$
	`batch_size`	$\min(1000, N//10)$ (Incremental)
t-SNE	`n_components`	$\{2, 3\}$
	`perplexity`	$[\min(5, N/100), \min(150, (N-1)/2)]$
	`learning_rate`	$[10.0, 1000.0]$ (Logarítmico)
	`max_iter`	$[500, 3000]$
	`early_exaggeration`	$[4.0, 36.0]$
UMAP	`n_components`	$[2, 50]$
	`n_neighbors`	$[\min(5, N/100), \min(300, N-1)]$
	`min_dist`	$[0.0, 0.8]$
	`learning_rate`	$[0.1, 3.0]$
	`metric`	{'euclidean', 'cosine', 'manhattan'}
	`n_epochs`	$[200, 1500]$
	`negative_sample_rate`	$[2, 15]$

Función Objetivo

$$Score = 0.7 \cdot \text{Norm}(Silhouette) + 0.3 \cdot Trustworthiness \tag{3.1}$$

Para evaluar la calidad de la reducción en un contexto no supervisado utilizando algoritmos no lineales (**t-SNE** y **UMAP**), se diseñó una métrica híbrida presentada en la Ecuación 3.1. Esta función busca un equilibrio entre la separación de grupos y la fidelidad de la proyección: el **Silhouette Score** favorece configuraciones que producen grupos densos y bien separados, mientras que la **Confianza (Trustworthiness)** penaliza las proyecciones que introducen falsos vecinos. Cabe destacar que para **PCA**, al ser una transformación lineal global, se optimizó maximizando únicamente el *Silhouette Score*.

El Pseudocódigo A.4 describe la lógica implementada para la búsqueda de la mejor proyección.

3.1.5. Etapa 3: Agrupamiento No Supervisado (*Clustering*)

Esta etapa constituye el núcleo de la validación de la hipótesis: determinar si la estructura latente de los *embeddings* posee la calidad suficiente para permitir una **clasificación no supervisada** de los modelos de vehículos. A diferencia de enfoques clásicos particionales como **K-Means**, que requieren conocer el número de clases a priori, esta investigación priorizó algoritmos capaces de descubrir la estructura natural de los datos sin suposiciones previas sobre la cantidad de categorías.

3.1.5.1. Motivación y Selección de Algoritmos

La selección de los algoritmos se fundamentó en la necesidad de operar en un escenario de "mundo abierto", donde el sistema desconoce la cantidad total de modelos de vehículos (clases) a los que será expuesto. Por este motivo, se descartó **K-Means** debido a su dependencia crítica del parámetro k y su tendencia a forzar clústeres de forma esférica.

En su lugar, se optó por una estrategia comparativa dominada por métodos basados en densidad, contrastada con un enfoque jerárquico de varianza mínima:

DBSCAN Se seleccionó como el algoritmo base por su capacidad probada para identificar clústeres de formas arbitrarias y su robustez ante el ruido (*outliers*), características esenciales dado que los *embeddings* visuales pueden presentar irregularidades en el espacio latente.

HDBSCAN y OPTICS Se incluyeron como evoluciones necesarias para mitigar las limitaciones de **DBSCAN**. **HDBSCAN** permite adaptarse a clústeres de densidades variables (común en datasets heterogéneos de vehículos), mientras que **OPTICS** se utilizó para evaluar la estructura de alcanzabilidad, permitiendo detectar agrupamientos anidados.

Agglomerative Clustering A diferencia de los tres anteriores (basados en densidad), este algoritmo se incluyó como punto de comparación. Al utilizar el criterio de enlace de Ward, se agrupa basándose en la minimización de la varianza interna. Su inclusión permite contrastar si la naturaleza de los datos vehiculares responde mejor a una separación por densidad o a una cohesión por varianza mínima.

3.1.5.2. Estrategia de Optimización con Optuna

La capacidad de estos algoritmos para descubrir la estructura correcta depende críticamente de sus hiperparámetros. Para evitar sesgos manuales, se implementó una búsqueda automatizada utilizando **Optuna**.

El proceso de optimización se configuró de la siguiente manera:

Muestreador Se utilizó el **TPE (Tree-structured Parzen Estimator)** para muestrear eficientemente el espacio de búsqueda.

Presupuesto Se ejecutaron **250 intentos (trials)** por algoritmo para asegurar la convergencia en un espacio de búsqueda complejo.

Rangos de Búsqueda Los hiperparámetros se acotaron dinámicamente según el tamaño del dataset (N) para asegurar su validez. El detalle de los rangos se presenta en la Tabla 3.7.

Tabla 3.7: Espacio de búsqueda de hiperparámetros para algoritmos de clustering optimizados con Optuna. N representa el número de muestras del dataset.

Algoritmo	**Hiperparámetro**	**Rango de Búsqueda / Configuración**
DBSCAN	`eps`	$[\min(0.001, P_5(dist)), \min(20.0, P_{95}(dist))]$
	`min_samples`	$[1, \max(15, N/30)]$
HDBSCAN	`min_cluster_size`	$[2, \min(20, N/30)]$
	`min_samples`	$[1, 15]$
	`cluster_selection_epsilon`	$[0.0, 1.0]$
	`cluster_selection_method`	{'eom', 'leaf'}
OPTICS	`min_samples`	$[2, \max(15, N/30)]$
	`xi`	$[0.005, 0.15]$
	`min_cluster_size`	$[\max(2, N/250), \min(100, N/15)]$
Agglomerative	`distance_threshold`	$[1.0, 50.0]$
	`linkage`	{'ward', 'complete', 'average', 'single'}

Función Objetivo

$$Score = 0.35 \cdot \overline{Sil} + 0.25 \cdot \overline{CH} + 0.25 \cdot ARI + 0.15 \cdot NMI \tag{3.2}$$

Se diseñó una función objetivo compuesta para maximizar la calidad del agrupamiento. Aunque el objetivo final es no supervisado, durante la etapa de optimización y validación se aprovecharon las etiquetas disponibles para guiar la búsqueda hacia configuraciones semánticamente coherentes. La Ecuación 3.2 combina métricas internas como el **Silhouette Score** ($\overline{Sil}$) y el **Calinski-Harabasz** ($\overline{CH}$) normalizados, con métricas externas como el **Adjusted Rand Index** (ARI) y la **Normalized Mutual Information** (NMI). Esta ponderación (60 % estructura, 40 % verdad fundamental) penaliza configuraciones degeneradas (e.g., ruido $> 40\,\%$ o un solo clúster gigante).

El Pseudocódigo A.5 describe la lógica implementada para la búsqueda de los mejores parámetros de agrupamiento. La Figura B.1 ilustra el flujo de trabajo implementado con Optuna para la optimización tanto de la reducción de dimensionalidad como del clustering.

3.1.5.3. Definición de Tipología de Clústeres

Para evaluar cualitativamente el éxito del agrupamiento, se definieron formalmente tres tipos de clústeres en función de la homogeneidad de sus etiquetas verdaderas (utilizadas solo para evaluación *a posteriori*):

Cluster Puro Aquel formado exclusivamente por instancias de una única clase de vehículo (Pureza = 1.0). Este es el caso ideal, donde el algoritmo logra aislar perfectamente un modelo específico en el espacio latente.

Cluster Mixto Aquel que contiene instancias de múltiples clases sin que ninguna predomine significativamente. Estos clústeres indican una confusión en el espacio de características, usualmente entre modelos visualmente idénticos (e.g., diferentes años de una misma generación).

Cluster Dominante Un subtipo de clúster mixto donde una clase representa la mayoría absoluta de las instancias (e.g., 90 % Clase A, 10 % Ruido/Otras). Estos son considerados éxitos parciales, ya que permiten una identificación con alta probabilidad.

(a) Ejemplo de clúster puro. Todas las imágenes corresponden a un *Skoda Haorui* del año 2013.

(b) Ejemplo de clúster mixto. Se observan múltiples modelos similares agrupados. Las imágenes corresponden a los modelos *Skoda VisionC* del 2014, *Skoda Haorui* del 2014 y *Skoda Haorui* del 2012.

(c) Ejemplo de clúster dominante. Tres imágenes pertenecen al modelo *Skoda Haorui* del año 2013, mientras que la restante corresponde al mismo modelo pero del año 2012.

Figura 3.8: Ejemplos de los distintos tipos de clusters definidos para la evaluación cualitativa del agrupamiento. De izquierda a derecha: 3.8.a Clúster Puro, 3.8.b Clúster Mixto, 3.8.c Clúster Dominante.

3.2. Diseño Experimental

Para validar la hipótesis de investigación y evaluar la efectividad de las representaciones profundas en la identificación de modelos vehiculares, se diseñó una serie de cuatro experimentos secuenciales. Cada experimento se centra en aislar una variable crítica del sistema (arquitectura, estrategia de *fine-tuning*, cantidad de datos y capacidad de generalización), manteniendo constantes los demás parámetros para asegurar la comparabilidad de los resultados.

A continuación, se detallan las configuraciones y objetivos de cada experimento.

3.2.1. Experimento 1: Establecimiento de Línea Base Unimodal

Objetivo Determinar el rendimiento base de las arquitecturas de visión por computadora tradicionales (**CNN** vs. **Transformers**) y cuantificar la ganancia obtenida al cambiar de un objetivo de clasificación pura a uno de *Metric Learning*. Este experimento actúa como la base de comparación para validar la supuesta superioridad de los modelos multimodales posteriores.

Configuración Se entrenaron y evaluaron 2 variantes de modelos unimodales utilizando un umbral fijo de 8 imágenes por clase (`min_images=8`). Se compararon los *embeddings* "*Out of the box*" (*Baseline*) contra los re-entrenados (*fine-tuned*).

Metodología Se definieron cuatro configuraciones experimentales, combinando dos arquitecturas (**ResNet50** y **ViT-B/32**) con dos objetivos de entrenamiento (Clasificación y *Metric Learning*). La Tabla 3.8 resume las configuraciones utilizadas.

Tabla 3.8: Configuraciones para la evaluación de modelos de visión pura. Se seleccionaron ResNet50 y ViT-B/32 por ser los *backbones* visuales utilizados en la implementación original de CLIP, permitiendo una comparación directa.

ID	Arquitectura	Paradigma	Loss Function
E1.1	ResNet50	CNN	Cross-Entropy (Clasificación)
E1.2	ResNet50	CNN	Contrastive / Triplet / NT-Xent / ArcFace / Multisimilarity (Metric Learning)
E1.3	ViT-B/32	Transformer	Cross-Entropy (Clasificación)
E1.4	ViT-B/32	Transformer	Contrastive / Triplet / NT-Xent / ArcFace / Multisimilarity (Metric Learning)

Cada configuración se entrenó utilizando valores de hiperparámetros diferenciados, selecciondos manualmente tras una revisión bibliografica exhaustiva, un análisis del conjunto de datos y varios ensayos. Los hiperparámetros específicos se detallan en el Apéndice B.

3.2.2. Experimento 2: Dinámica de Adaptación en Modelos Multimodales (CLIP)

Objetivo Analizar la plasticidad requerida por los codificadores de **CLIP** para adaptarse al dominio vehicular. Se parte de la premisa de que el componente visual requiere una adaptación profunda (*hard fine-tuning*) debido a las sutilezas de los modelos de autos, mientras que el componente textual podría degradarse si se modifica excesivamente (*catastrophic forgetting*).

Configuración Utilizando la arquitectura **CLIP-ViT-B/32** y fijando los hiperparámetros de optimización (*Learning Rate*, *Weight Decay*, *Batch Size*, *Epochs*).

Metodología Se realizaron dos sub-estudios:

- **E2.A - Descongelamiento Progresivo:** Se entrenó el modelo descongelando progresivamente las capas del *Image Encoder* y el *Text Encoder* desde la última (la más cercana a la salida) hasta la primera (la más cercana a la entrada), en pasos de 1 a 12 capas. Se evaluó la calidad de los clústeres resultantes (Puros vs. Mixtos) para encontrar el punto óptimo de profundidad de entrenamiento.
- **E2.B - Entrenamiento Secuencial:** A partir de los hallazgos de E2.A, se evaluaron dos estrategias de ajuste por fases (*phase fine-tuning*). Se contrastó el rendimiento de optimizar primero el codificador visual (manteniendo el textual congelado) seguido de un refinamiento del codificador textual, **frente a la secuencia inversa**. Esta comparación permitió determinar el orden de adaptación más eficaz, el cual fue estandarizado para los experimentos subsiguientes.

3.2.3. Experimento 3: Sensibilidad al Volumen de Datos

Objetivo Evaluar la robustez del modelo **CLIP-ViT-B/32** ante la escasez de datos, simulando escenarios de mayor restricción. Este experimento busca determinar el límite inferior de datos necesarios para que el agrupamiento no supervisado siga siendo efectivo.

Configuración Manteniendo la mejor configuración de entrenamiento obtenida en el Experimento 2, se varió el umbral de filtrado del dataset (`min_images`).

Metodología Se generaron cuatro escenarios de entrenamiento con distinta densidad de información:

- **Escenario Crítico:** 4 y 6 imágenes por clase.
- **Escenario Base:** 8 imágenes por clase (Estándar del estudio).
- **Escenario Rico:** 10 y 12 imágenes por clase.

3.2.4. Experimento 4: Escalabilidad y Generalización (Testing Final)

Objetivo Validar si el aumento en la capacidad del modelo (mayor profundidad o resolución) se traduce en una mejora del agrupamiento y, crucialmente, evaluar el desempeño final en clases nunca vistas (*Zero-Shot*).

Configuración Se utilizaron arquitecturas de mayor capacidad.

Metodología El experimento se dividió en dos fases:

- **E4.A Escalado de Arquitectura:** Se realizo una nueva busqueda de hiperparámetros para los algoritmos de reducción de dimensionalidad y *clustering* sobre las nuevas arquitecturas de mayor capacidad para verificar si los algoritmos son robustos frente a la escala de las arquitecturas.

- **E4.B Evaluación de Generalización (*Open-Set Testing*):** Se seleccionarán los modelos con mejor desempeño (Top-3) para la evaluación final sobre el conjunto de **Clases One-Shot** (clases excluidas del entrenamiento). Se evaluará el agrupamiento variando la proporción de clases desconocidas inyectadas en el sistema (0 %, 25 %, 50 %, 75 % y 100 % del set de prueba) para medir la degradación del rendimiento ante la novedad pura.

3.3. Protocolo de Evaluación

Dado que los algoritmos de agrupamiento no supervisado carecen de una función de pérdida directa comparable a la exactitud (*accuracy*) en clasificación supervisada, la evaluación del desempeño se estructura en tres dimensiones complementarias: métricas cuantitativas de calidad de clúster, inspección visual de la topología latente y análisis de generalización.

A continuación se define el criterio de éxito para cada dimensión.

3.3.1. Métricas Cuantitativas y Criterio de Éxito

Para objetivar la calidad de los agrupamientos obtenidos en los Experimentos 1, 2 y 3, se utilizan las métricas definidas teóricamente en la Sección 2.9. La evaluación se jerarquiza de la siguiente manera:

1. **Métricas Primarias (Validación Externa):** Dado que se dispone de las etiquetas reales (*ground truth*) para fines de validación, el **Adjusted Rand Index (ARI)** y la **Normalized Mutual Information (NMI)** constituyen los indicadores principales. Se considera que una configuración supera al estado del arte si logra un incremento positivo ($\Delta > 0$) en **ARI** respecto a la línea base (resultados obtenidos utilizando los *embeddings baseline*) y mantiene la estabilidad en **NMI**.

2. **Métrica de Pureza (Cluster Purity):** Para el análisis específico de identificación vehicular, se reporta la pureza promedio de los clústeres, calculada como el porcentaje de clústeres puros generados.

3. **Métricas Secundarias (Validación Interna):** El **Silhouette Score** se utiliza como métrica auxiliar para monitorear la compacidad geométrica de los *embeddings*, aunque se subordina a las métricas externas debido a que una alta compacidad no siempre implica una separación semántica correcta en clases de grano fino.

3.3.2. Evaluación Cualitativa (Inspección Visual)

Para complementar los indicadores numéricos, se generan proyecciones bidimensionales utilizando **UMAP**. La evaluación cualitativa busca identificar patrones que las métricas globales pueden enmascarar, tales como:

1. La fragmentación de una misma clase de vehículo en múltiples sub-clústeres (por ejemplo, debido a cambios de iluminación o color).

2. La fusión incorrecta de modelos visualmente similares (e.g., diferentes generaciones de un mismo sedán).

3. La proporción de clústeres mixtos.

4. Presencia de Clases Complicadas: Identificación de clases que resultan sistemáticamente fragmentadas en múltiples clústeres, lo que indicaría una dificultad intrínseca de los datos para ser agrupados o una limitación en la representación aprendida.

3.3.3. Plan Metodológico de Evaluación Final

La evaluación final del sistema no se limita a reportar un único número, sino que sigue un plan riguroso diseñado para caracterizar el comportamiento del modelo en profundidad.

3.3.3.1. Optimización Compuesta

Para cada experimento (1 al 3), el desempeño reportado corresponde a la mejor configuración encontrada tras ejecutar **50 trials** para la reducción de dimensionalidad y **250 trials** para el *clustering* mediante Optuna. Esto asegura que la comparación entre modelos (e.g., **ResNet** vs **CLIP**) sea justa, evaluando cada uno en su punto óptimo de operación y no bajo hiperparámetros arbitrarios.

3.3.3.2. Formulación de la Métrica de Éxito

$$\mathcal{S} = w_1 \cdot P_{avg} + w_2 \cdot \left(1 - \frac{|N - n|}{N}\right) \tag{3.3}$$

Para comparar objetivamente los resultados entre experimentos y determinar el mejor modelo del estudio (Top-3: Mejor **CLIP**, Mejor **ViT**, Mejor **ResNet**), se utiliza una métrica de éxito ponderada ($\mathcal{S}$) que prioriza linealmente la pureza del agrupamiento sobre la consistencia estructural (Ecuación 3.3). Donde P_{avg} es la pureza promedio de los clústeres, N es el número real de clases y n es el número de clústeres obtenidos. Los pesos w_1 y w_2 (con $w_1 + w_2 = 1$) se ajustan para reflejar la prioridad del diseño: se prefiere obtener clústeres altamente puros aunque su número total difiera ligeramente del real ($w_1 > w_2$).

3.3.3.3. Evaluación de Generalización (Zero-Shot)

Para el Experimento 4, se utilizan los mejores modelos seleccionados en la fase anterior (sin re-optimizar hiperparámetros) para evaluar el conjunto de prueba *One-Shot*. Se calcula la tasa de degradación de la métrica $\mathcal{S}$ al comparar el rendimiento en clases vistas frente a clases no vistas (0 %, 25 %, 50 %, 75 %, 100 % de inyección de clases desconocidas).

$$D_{\mathcal{S}} = \frac{\mathcal{S}_{vistas} - \mathcal{S}_{novistas}}{\mathcal{S}_{vistas}} \tag{3.4}$$

Donde $D_{\mathcal{S}}$ representa la degradación relativa de la métrica de éxito $\mathcal{S}$ al enfrentarse a clases no vistas.
Un valor bajo de $D_{\mathcal{S}}$ indica que el sistema ha aprendido descriptores visuales generalizables (e.g., forma de los faros) en lugar de memorizar clases específicas, validando así la hipótesis de que las representaciones profundas permiten la identificación de modelos vehiculares en un escenario de mundo abierto.

Capítulo 4

Resultados y Discusión

En este capítulo se presentan los resultados experimentales obtenidos a partir de la metodología descrita anteriormente. El análisis se centra en evaluar la calidad de los espacios latentes generados para la tarea de agrupamiento (*clustering*) de vehículos, la capacidad de discriminación entre modelos de automóviles y la eficiencia computacional de los sistemas propuestos.

4.1. Experimento 1: Establecimiento de Línea Base Unimodal

El objetivo de este primer experimento es establecer una línea base sólida mediante la evaluación exhaustiva de configuraciones unimodales (solo visión). A continuación, se analiza el impacto de la arquitectura de la red troncal (*backbone*), el objetivo de aprendizaje, las funciones de pérdida específicas y la integración de las vistas frontal y trasera en la conformación de los grupos de vehículos.

4.1.1. Comparativa de Arquitectura y Objetivos de Aprendizaje

En primera instancia, se evalúa el rendimiento de las dos arquitecturas propuestas: ResNet50 (CNN) y ViT-B/32 (Transformer), bajo dos paradigmas de aprendizaje: Clasificación Supervisada y *Metric Learning*.

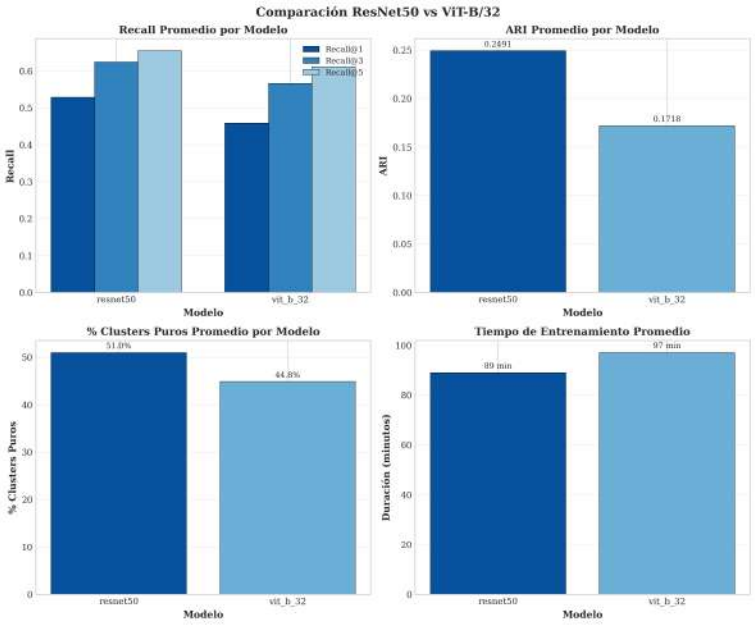


Figura 4.1: Comparativa de arquitecturas visuales: ResNet50 vs ViT-B/32. Se observa que ResNet50 supera consistentemente a ViT-B/32 en métricas de cohesión local (Recall@k), calidad de clustering (ARI) y porcentaje de clusters puros.

Como se evidencia en la Figura 4.1, la arquitectura **ResNet50 supera consistentemente a ViT-B/32** en todas las métricas de evaluación del espacio latente. Específicamente, ResNet50 alcanza un ARI promedio de 0.2605 frente al 0.1718 de ViT-B/32. Aunque la diferencia en pureza de clusters es menor (49.4 % vs 44.8 %), la capacidad de ResNet para estructurar globalmente el espacio es superior.

Estos resultados sugieren que el sesgo inductivo inherente a las Redes Neuronales Convolucionales (invarianza a la traslación y jerarquía de características locales) resulta más beneficioso para capturar las líneas de diseño, la geometría de la carrocería y detalles específicos (faros, llantas, parrillas) que la atención global de los *Vision Transformers*. La naturaleza *data-hungry* de ViT implica que, sin un pre-entrenamiento masivo o un volumen de datos significativamente mayor, no logra diferenciar tan eficazmente las sutiles variaciones

entre modelos de autos (e.g., distintos años de un mismo modelo o *facelifts*) como la CNN. Además, ResNet50 demostró ser más eficiente en tiempo de entrenamiento promedio (88.8 min vs 97.0 min).

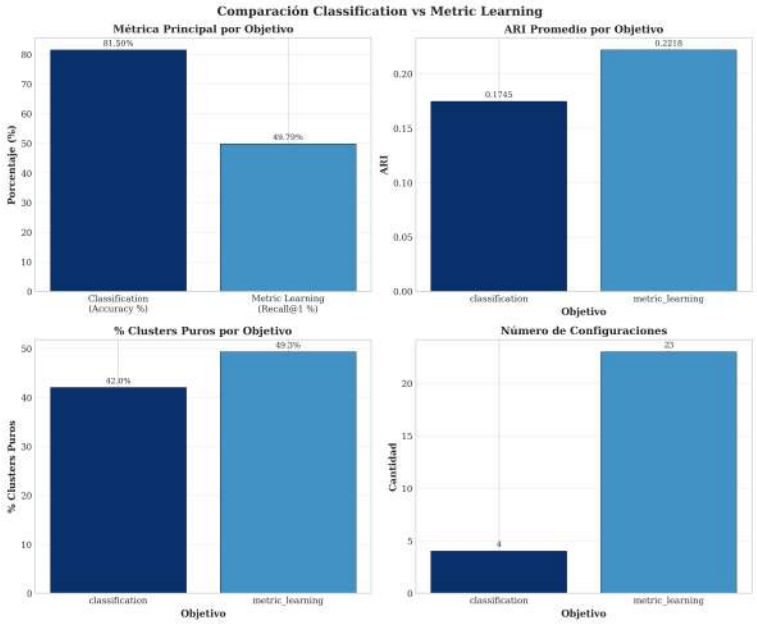


Figura 4.2: Comparativa de objetivos de aprendizaje: Clasificación vs Metric Learning. Se observa que el enfoque de Metric Learning genera espacios latentes mucho más ricos y aptos para el agrupamiento que la Clasificación.

Respecto al objetivo de aprendizaje, la Figura 4.2 revela una clara ventaja del aprendizaje métrico. Si bien el enfoque de **Clasificación** obtiene una exactitud (*Accuracy*) alta en la tarea supervisada (81.5 % promedio), su desempeño al generar *embeddings* para clustering es deficiente en comparación con **Metric Learning**. El ARI promedio (0.22 vs 0.17) y el porcentaje de clusters puros (48.1 % vs 42.0 %) son superiores en los modelos entrenados con aprendizaje métrico. Esto valida la hipótesis de que, para tareas de agrupamiento no supervisado de vehículos, optimizar directamente las distancias en el espacio de características compacta mejor las clases intra-varianza (mismo modelo de auto) que la optimización de hiperplanos de decisión.

4.1.2. Impacto de las Funciones de Pérdida (*Loss Functions*)

Profundizando en el paradigma de *Metric Learning*, se analiza el comportamiento de las distintas funciones de pérdida y su efecto en la topología de los grupos.

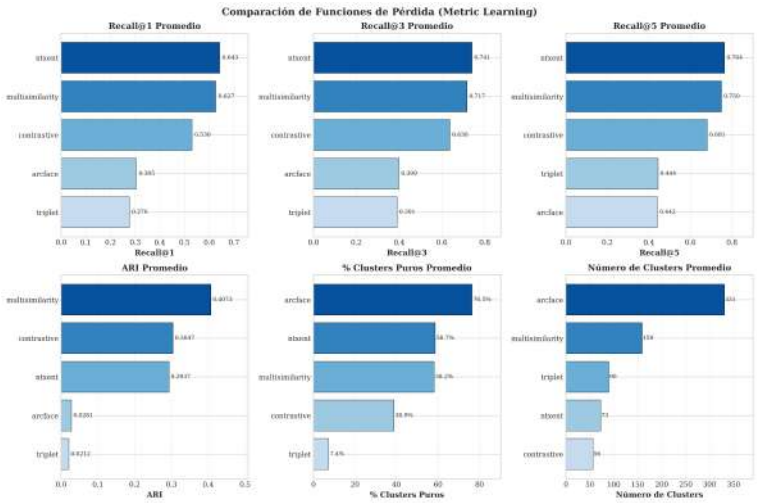
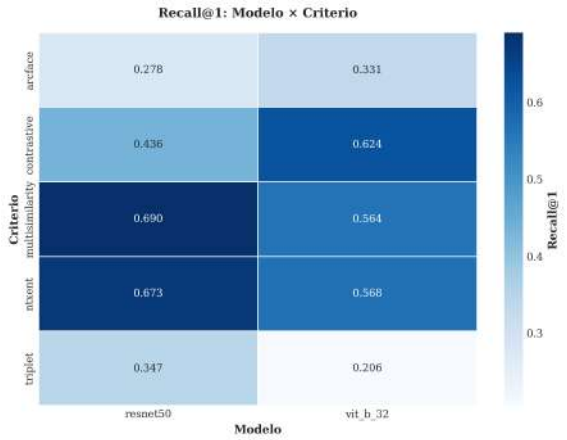


Figura 4.3: Comparativa de funciones de pérdida dentro del paradigma de Metric Learning. MultiSimilarity y NTXent dominan las métricas de cohesión, mientras que ArcFace muestra un comportamiento peculiar con alta pureza pero bajo ARI.

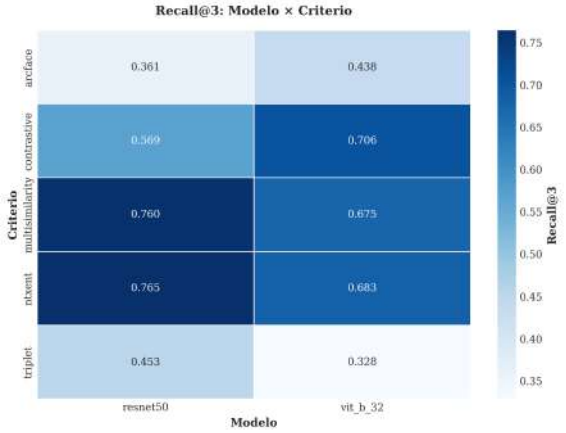
La Figura 4.3 destaca a **MultiSimilarity** y **NTXent** como las funciones de pérdida más robustas, dominando los rankings de Recall@1, Recall@3, Recall@5 y ARI. Estas pérdidas logran equilibrar la atracción de positivos y la repulsión de negativos, generando clusters cohesivos de modelos de autos.

Un hallazgo notable es el comportamiento de **ArcFace**. Esta función logra el porcentaje más alto de clusters puros (superando el 90 % en varias configuraciones), pero paradójicamente presenta un ARI y Recall extremadamente bajos. Este fenómeno indica que ArcFace tiende a **sobre-fragmentar** el espacio latente. Al forzar márgenes angulares muy estrictos, el modelo separa una misma identidad (modelo de auto) en múltiples sub-clusters aislados (ver Figura 4.6). Por ejemplo, podría estar separando el mismo vehículo por color, ángulo o pequeñas variaciones de equipamiento en clusters distintos. Estos sub-clusters son altamente puros, pero al estar artificialmente separados, el algoritmo de clustering no logra unificarlos como un solo modelo de vehículo, penalizando severamente el ARI. Por otro lado, *Triplet Loss* muestra el rendimiento más pobre.

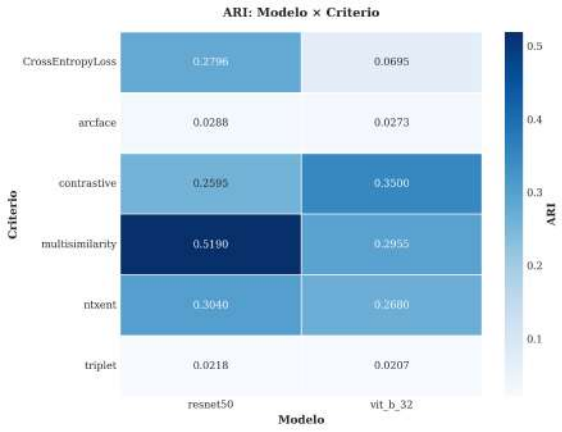
Para un desglose detallado del rendimiento por cada combinación de modelo y criterio, se presentan los mapas de calor en la Figura 4.4.



(a) Heatmap de Recall@1.



(b) Heatmap de Recall@3.



(c) Heatmap de ARI.

Figura 4.4: Análisis detallado del impacto de las funciones de pérdida en las métricas *Recall* (proxy de vecindad) y **ARI**. Se confirma la superioridad de las combinaciones que utilizan ResNet50 con MultiSimilarity o NTXent.

4.1.3. Integración de Información Multivista

Se evaluó la hipótesis de si la incorporación de la vista trasera del vehículo añade información discriminativa relevante para el agrupamiento.

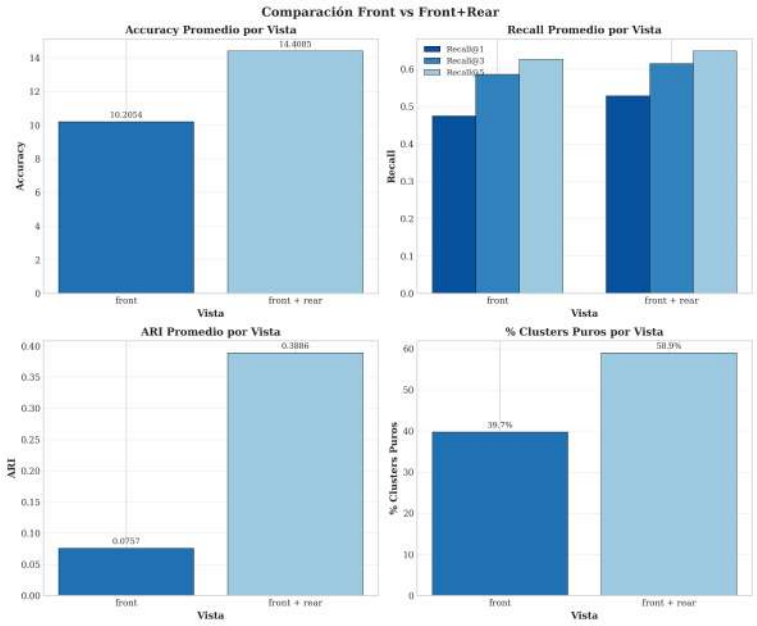


Figura 4.5: Comparativa de configuraciones de vistas: Solo Frontal vs Frontal + Trasera. La combinación de ambas vistas mejora significativamente la definición de los grupos.

Como se observa en la Figura 4.5, la configuración **Front + Rear** es estrictamente superior a utilizar solo la vista frontal. El aumento en Recall@1 (de 0.48 a 0.52 promedio) y ARI (de 0.07 a 0.38) es significativo. Esto confirma que la vista trasera aporta características visuales únicas (diseño de luces traseras, forma del maletero, salidas de escape, emblemas) que permiten resolver ambigüedades presentes cuando autos de distintas marcas o modelos comparten un diseño frontal similar o genérico. Esta información adicional es crítica para desambiguar modelos y construir un espacio latente con mayor separabilidad.

4.1.4. Calidad del Espacio Latente y Topología de *Clusters*

Más allá de las métricas de rendimiento, es crucial analizar la topología de los agrupamientos generados.

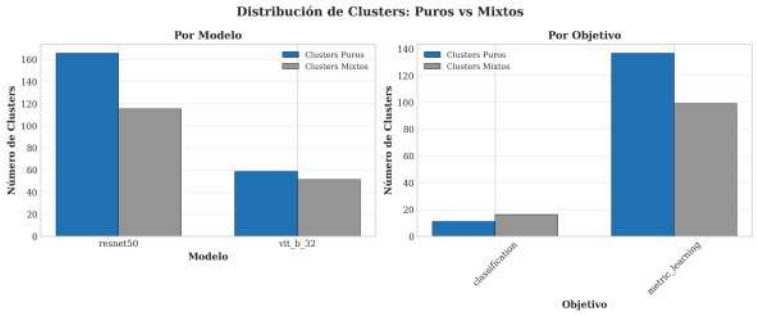


Figura 4.6: Distribución de clústeres puros vs mixtos para distintas configuraciones. Metric Learning genera una cantidad significativamente mayor de clústeres puros en comparación con Clasificación.

La Figura 4.6 ilustra la distribución de clusters. El paradigma de *Metric Learning* genera del orden de 130-125 clusters puros frente a 95-100 mixtos, mientras que Clasificación apenas logra generar 10-15 clusters puros. Esto demuestra que los modelos de clasificación tienden a colapsar múltiples modelos de vehículos en regiones confusas del espacio, mientras que el aprendizaje métrico logra aislar las identidades visuales con mayor éxito.

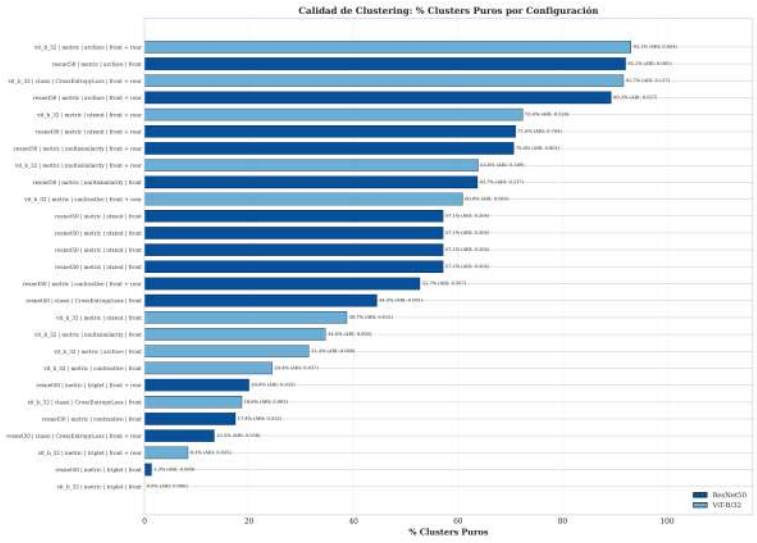


Figura 4.7: Ranking detallado de calidad de clustering (% Clusters Puros) para todas las configuraciones evaluadas.

El ranking de calidad (Figura 4.7) y el análisis de clases problemáticas (Figura 4.8) ofrecen una visión complementaria. Existe una correlación negativa entre el ARI y el número de clases solapadas.

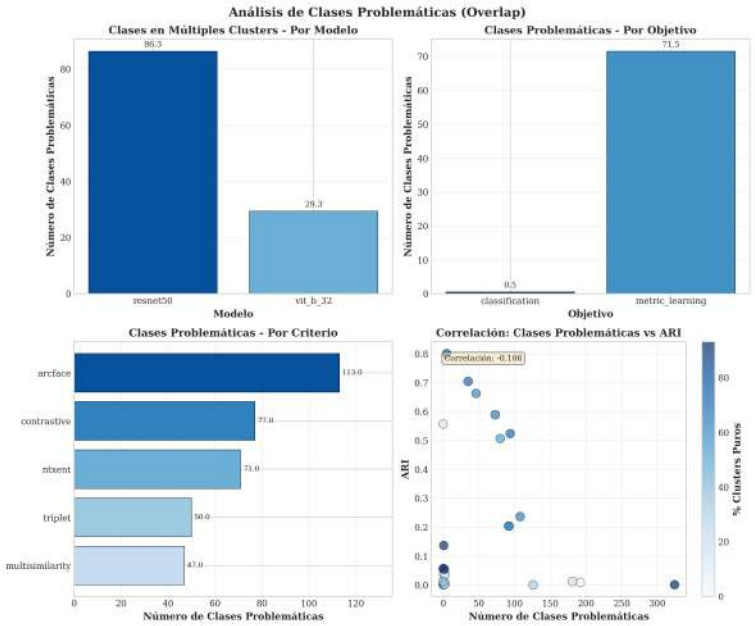


Figura 4.8: Análisis de clases problemáticas con solapamiento en el espacio latente. Se observa una correlación entre un bajo ARI y un alto número de clases compartidas entre clusters.

Es interesante notar que, aunque ViT tiene un rendimiento general menor, presenta menos clases problemáticas promedio (29.3) que ResNet (86.3). Esto sugiere que los *embeddings* de ViT son altamente distintivos cuando logran formar un cluster, pero el modelo falla en agrupar correctamente la totalidad de las variaciones de un mismo modelo de auto (bajo Recall). Por el contrario, ResNet captura mejor la varianza intra-clase (distintos colores o acabados del mismo auto), aunque a costa de un mayor solapamiento en los bordes de decisión entre modelos parecidos.

4.1.5. Eficiencia Computacional

Finalmente, se evalúa el costo computacional de las configuraciones.

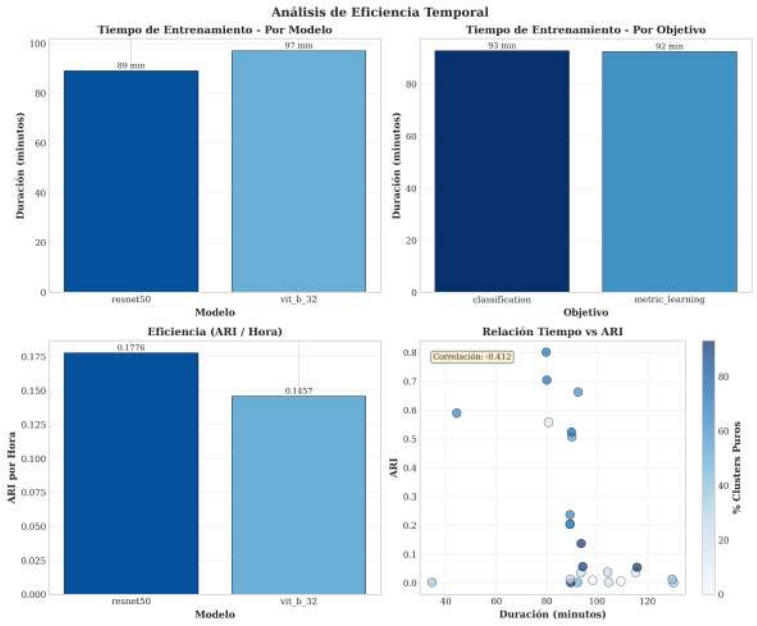


Figura 4.9: Eficiencia computacional comparativa entre ResNet50 y ViT-B/32. Se observa que ResNet50 no solo ofrece mejor rendimiento, sino también una mayor eficiencia en términos de ARI por hora de entrenamiento.

La Figura 4.9 muestra que ResNet50 no solo obtiene mejores resultados, sino que es más rápido de entrenar. Al calcular una métrica de eficiencia (ARI por hora), ResNet50 obtiene una mejor relación frente a ViT-B/32. Dada la correlación negativa observada entre tiempo de entrenamiento y desempeño (-0.412), se concluye que modelos más complejos o lentos no necesariamente garantizan mejores representaciones para este dataset de vehículos.

4.1.6. Mejores Configuraciones y Selección Final

A modo de síntesis, la Figura 4.10 presenta las 5 mejores configuraciones según distintas métricas.

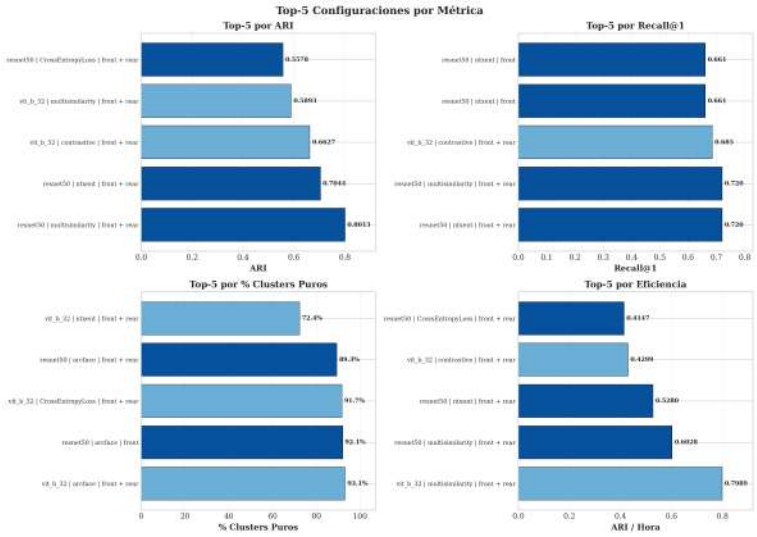


Figura 4.10: Comparativa de las 5 mejores configuraciones del Experimento 1. Se observa que las configuraciones basadas en ResNet50 y Metric Learning dominan el ranking.

El análisis conjunto de ARI, Recall@1 y Eficiencia señala una clara tendencia ganadora. Las configuraciones que ocupan los primeros lugares combinan consistentemente:

1. **Arquitectura:** ResNet50.
2. **Objetivo:** Metric Learning.
3. **Pérdida:** MultiSimilarity.
4. **Vistas:** Front + Rear.

Basado en los resultados expuestos, se selecciona la configuración **ResNet50 con MultiSimilarity Loss utilizando vistas Frontales y Traseras** como la línea base unimodal óptima y codificador visual principal. Sin embargo, si bien esta configuración establece una barra de rendimiento alta, sigue dependiendo puramente de características visuales y patrones geométricos. Dada la distribución de cola larga del conjunto de datos y los desafíos de la aplicación real (donde el sistema enfrentará modelos desconocidos), esta dependencia exclusiva de la visión puede comprometer la robustez de la generalización ante clases nunca vistas (*Zero-Shot*). En consecuencia, a continuación se explora el modelo multimodal **CLIP**, con el objetivo de determinar si la integración de etiquetas textuales junto a los *embeddings* visuales logra generar un espacio latente semánticamente más denso, capaz de superar las limitaciones de la configuración puramente visual.

4.2. Experimento 2: Dinámica de Adaptación en Modelos Multimodales (CLIP)

Habiendo establecido que una arquitectura ResNet50 optimizada mediante aprendizaje métrico constituye un estándar alto en el dominio visual, este segundo experimento explora el potencial de los modelos multimodales **CLIP**. El objetivo central es determinar la metodología óptima para adaptar un modelo pre-entrenado en pares imagen-texto genéricos a la tarea específica y de grano fino de la clasificación vehicular no supervisada.

A continuación, se presentan los resultados de los estudios de descongelamiento progresivo (*Progressive Unfreezing*) y del orden de las fases de entrenamiento. Se analiza secuencialmente el impacto de los componentes individuales, la progresión del rendimiento por capas, la calidad estructural de los clústeres resultantes y la eficiencia computacional. Es importante destacar que, para garantizar la comparabilidad directa entre todas las configuraciones de este experimento, se utilizó un umbral fijo de 8 imágenes por clase (`min_images=8`) y la arquitectura **CLIP-ViT-B/32**.

4.2.1. Estudio de Descongelamiento Progresivo

4.2.1.1. Predominancia del Componente Visual sobre el Textual

El primer nivel de análisis se centra en el rendimiento aislado de ambos codificadores. La Figura 4.11 presenta la comparación de los resultados promedio asociados al ajuste exclusivo de cada componente.

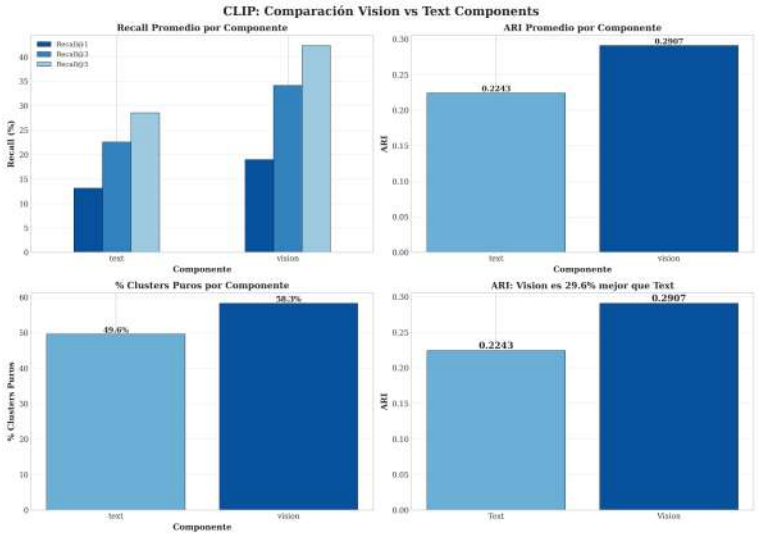


Figura 4.11: Comparación global entre componentes de Visión y Texto. Se observa una ventaja sistemática del ajuste visual en métricas de Recall, ARI y porcentaje de clústeres puros.

Los resultados evidencian una asimetría funcional crítica: el ajuste del **Image Encoder** proporciona una ganancia significativamente mayor en todas las métricas presentadas respecto al ajuste del **Text Encoder**. Aunque la magnitud de la diferencia podría parecer moderada en términos absolutos, es evidente que la adaptación visual impulsa con mayor fuerza el **ARI** y la "pureza" de los *clusters* generados. Esto confirma que la discriminación de modelos vehiculares reside primariamente en características visuales (patrones geométricos y de textura)

más que en la semántica textual pura. No obstante, los resultados sugieren el beneficio potencial de realizar un ajuste conjunto a ambas ramas, aunque priorizando estratégicamente la componente visual.

4.2.1.2. Progresión del Rendimiento según Profundidad (Recall y ARI)

Una vez identificada la preponderancia del componente visual, resulta necesario comprender la dinámica de mejora conforme se aumenta la profundidad del entrenamiento (número de capas descongeladas).

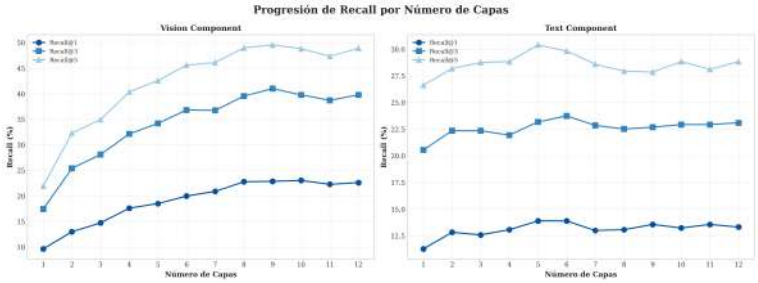


Figura 4.12: Progresión de las métricas de Recall (@1, @3, @5) en función del número de capas entrenables.

La Figura 4.12 muestra cómo la capacidad de recuperación (*Recall*) escala con la profundidad. En el componente de Visión, se aprecia una pendiente positiva constante y pronunciada, lo que indica que las capas más profundas (cercanas a la entrada) contienen información relevante que, al ser ajustada, refina la capacidad del modelo para encontrar vecinos correctos en el espacio latente. En contraste, la componente de Texto exhibe una pendiente menos pronunciada que alcanza su punto de saturación o máximo rendimiento en torno al descongelamiento de 6 capas; el ajuste de capas posteriores no presenta mejoras significativas en esta métrica.

Figura 4.13: Progresión del ARI y punto de máximo rendimiento. El ajuste profundo permite alcanzar los valores máximos de calidad de agrupamiento.

Complementariamente, la Figura 4.13 destaca el punto de máximo rendimiento en ARI. A diferencia de tareas más simples donde el ajuste de las últimas capas ("Heads") suele ser suficiente, aquí se observa que el máximo global se alcanza al descongelar una porción significativa de la red: específicamente, 9 capas para la componente de Visión y 10 capas para la componente Textual. Esto sugiere que la adaptación exitosa al dominio vehicular requiere una reconfiguración profunda de las características aprendidas previamente.

4.2.1.3. Directa: Visión vs. Texto por Capas

Para visualizar la brecha de rendimiento en cada etapa de profundidad, se presenta una comparación directa capa a capa en la Figura 4.14.

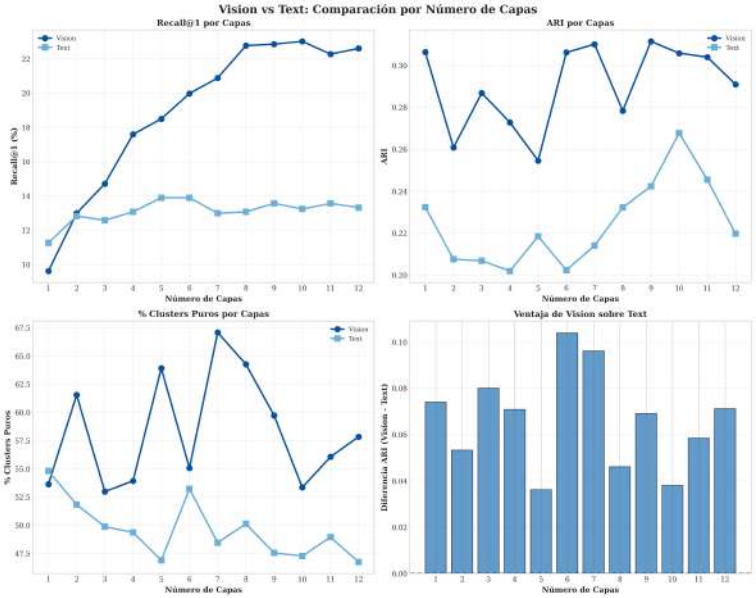


Figura 4.14: Comparativa directa capa a capa entre Visión y Texto. La brecha de rendimiento (cuadrante inferior derecho) se ensancha conforme aumenta la profundidad del entrenamiento, alcanzando un *sweet spot* al descongelar 6 y 7 capas.

El análisis detallado revela que la influencia de las componentes es lineal principalmente para la métrica de *Recall* (evaluada durante el *fine-tuning*), mientras que para las métricas de **ARI** y porcentaje de clústeres puros, las curvas presentan un comportamiento más complejo. Específicamente, para el caso del **ARI**, el descongelamiento de la componente visual genera picos de rendimiento óptimo al liberar 1, 6, 7 y 9 capas. Por su parte, en la componente textual, los mejores resultados se concentran en las capas profundas: 8, 9, 10 y 11.

En cuanto al porcentaje de clústeres puros, los mejores desempeños visuales se generan al descongelar 2, 5, 7 y 8 capas, mientras que en texto destacan las capas 1, 2, 6 y 8. El gráfico de diferencia porcentual (cuadrante inferior derecho) ratifica la ventaja sistemática del descongelamiento de capas visuales por sobre las textuales, mostrando una dominancia visual constante (valores positivos) a lo largo de toda la arquitectura.

4.2.1.4. Correlaciones y Mapas de Calor

Para validar estadísticamente la relación entre la profundidad del entrenamiento y el desempeño del modelo, se analizaron los coeficientes de correlación y se generaron mapas de calor del espacio de búsqueda.

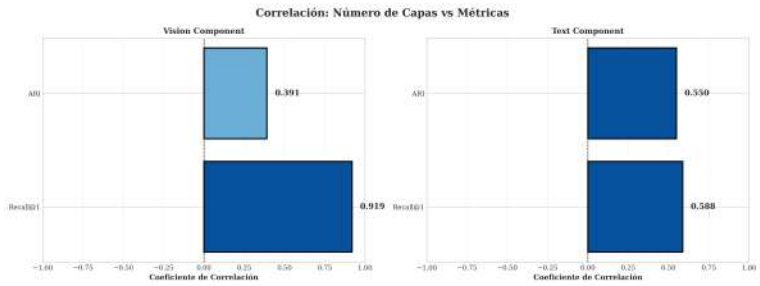


Figura 4.15: Coeficientes de correlación entre el número de capas y las métricas de desempeño. Existe una fuerte correlación positiva en el componente visual.

La Figura 4.15 muestra una correlación positiva fuerte (> 0.39 e incluso cercana a 0.91) para la componente Visual y una correlación fija entorno a 5.5 para el caso de la componente Textual. Resulta de particular interés notar que el ajuste fino de la componente Visual produce mejoras significativas en las métricas de vecindad inmediata (Recall@1) y tambien en la estructura global (ARI). En contrapartida, la componente textual presenta valores de correlación más bajos en ambas métricas, lo que sugiere que, aunque su impacto absoluto es menor, el ajuste profundo de la componente textual tiene una influencia determinante en la coherencia global de los embeddings y la conformación de los clústeres.

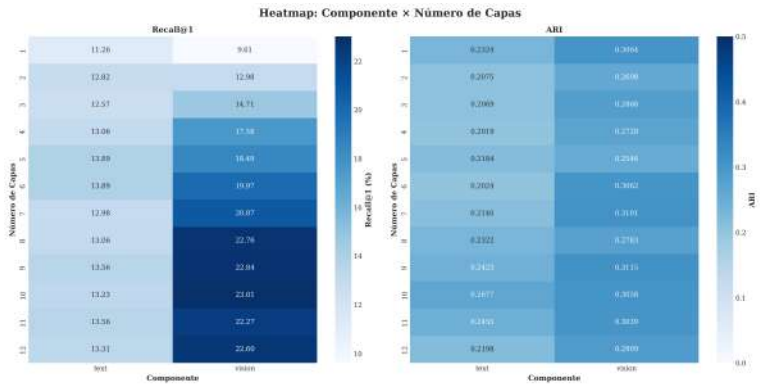


Figura 4.16: Mapa de calor de rendimiento. Se observa claramente cómo las zonas de alto rendimiento (colores intensos) se concentran en el componente visual con alto número de capas y que de manera general el ajuste de la componente visual presenta una mayor adaptación al dominio.

El mapa de calor de la Figura 4.16 permite identificar visualmente la "zona óptima" de operación. La región correspondiente a *Vision* con $9-10$ capas presenta consistentemente los valores más altos tanto en Recall@1 como en ARI. La componente textual, si bien muestra una menor varianza y un rendimiento global inferior, exhibe picos locales de rendimiento al descongelar 5, 9 y 11 capas.

4.2.1.5. Impacto en la Calidad Estructural del Clustering

La Figura 4.17 desglosa el impacto de la profundidad del ajuste en la topología de los *clusters* resultantes.

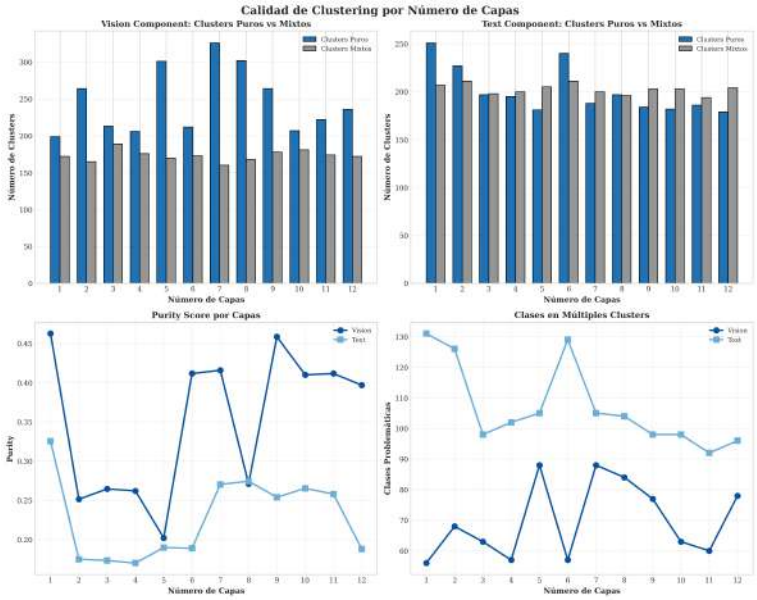


Figura 4.17: Calidad estructural del clustering. A mayor profundidad en el componente visual, aumenta la cantidad de clústeres puros y disminuye drásticamente el número de clases problemáticas (solapadas).

El análisis revela que el ajuste profundo del componente visual tiene la mayor implicancia en la calidad estructural. El *fine-tuning* de la rama visual otorga al espacio de *embeddings* una definición y nitidez superior, permitiendo la generación de una mayor cantidad de clústeres puros frente a mixtos en comparación con la componente textual, una tendencia que se mantiene constante independientemente del número de capas descongeladas. Adicionalmente, el **Purity Score** es sistemáticamente mayor y la cantidad de *clases problemáticas* (aquellas dispersas en múltiples clústeres) es siempre menor en los resultados derivados del ajuste visual.

4.2.1.6. Eficiencia y Ranking de Configuraciones

Finalmente, se contrasta el rendimiento obtenido con el costo computacional para evaluar la viabilidad de las distintas estrategias.

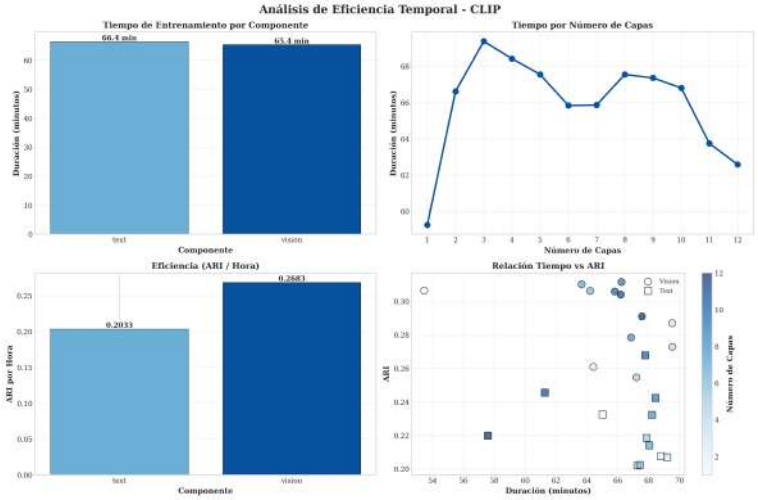


Figura 4.18: Análisis de eficiencia temporal (ARI por hora). Aunque el entrenamiento profundo consume más tiempo, la ganancia en ARI compensa el costo, manteniendo una eficiencia competitiva.

La Figura 4.18 ilustra que el tiempo de ejecución promedio por componente es sustancialmente similar. Sin embargo, el tiempo asociado al número de capas presenta una curva de crecimiento no lineal. A pesar del mayor costo temporal de las configuraciones profundas, la eficiencia (medida como ganancia de ARI por hora de cómputo) resulta superior para el componente visual, justificando la inversión de recursos.

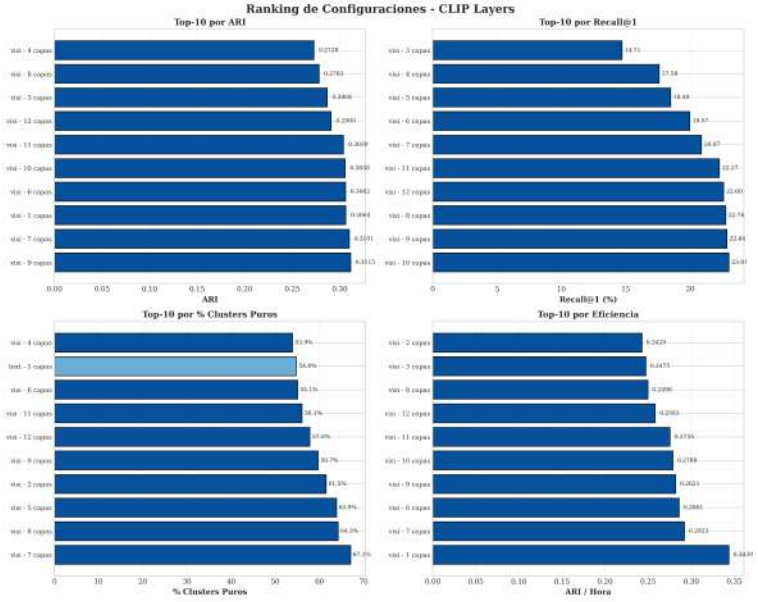


Figura 4.19: Ranking de las mejores configuraciones del Experimento 2.A. Las configuraciones de visión profunda dominan todos los indicadores clave (ARI, Recall, Pureza).

El ranking final presentado en la Figura 4.19 es concluyente: las 10 mejores configuraciones corresponden invariablemente al componente visual con un alto número de capas. Específicamente, las tres mejores configuraciones en términos de métricas globales se sitúan en el rango de 8 a 9 capas descongeladas.

4.2.1.7. Síntesis y Selección para los Siguientes Experimentos

El análisis integral de los resultados del Experimento 2.A permite extraer una conclusión fundamental: si bien existe una clara dominancia de la componente Visual en términos de ganancia bruta de rendimiento, la componente Textual desempeña un rol complementario no despreciable en la estructuración global del espacio latente (evidenciado por sus altas correlaciones con el ARI). Ambas componentes aportan información valiosa; la visión captura la morfología y textura necesaria para la discriminación fina, mientras que el texto preserva la coherencia semántica. Por lo tanto, la estrategia óptima no consiste en elegir una sobre la otra, sino en integrar el ajuste de ambas.

Dada la mayor influencia de la componente visual, se puede intuir que priorizar su ajuste fino —estableciendo primero una base visual sólida— generará un punto de partida más robusto antes de refinar las relaciones semánticas con el texto. Esta hipótesis sobre la secuencialidad del entrenamiento será el foco del estudio en la siguiente sección.

Para los experimentos subsiguientes, y basándose en la maximización de la métrica de éxito definida en la Ecuación 3.3, se establecen como valores fijos el descongelamiento de **12 capas del Image Encoder** y **9 capas del Text Encoder**. Esta configuración busca maximizar la capacidad del modelo para capturar tanto las sutilezas visuales críticas como las relaciones semánticas necesarias para una identificación vehicular robusta.

4.2.2. Estudio del Orden de Fases de Entrenamiento

Basado en el hallazgo de la sección anterior, donde se identificó la predominancia del componente visual en la adaptación al dominio, esta etapa evalúa la hipótesis de la secuencialidad. El objetivo es determinar si el orden en que se ajustan los componentes afecta la convergencia final del modelo y la calidad del espacio latente. Se contrastan dos estrategias de *fine-tuning* secuencial (*Two-stage Fine-tuning*):

1. **Visión → Texto:** Optimización profunda del codificador visual (Fase 1), seguida de un refinamiento del codificador textual (Fase 2).
2. **Texto → Visión:** Optimización del codificador textual (Fase 1), seguida del ajuste del codificador visual (Fase 2).

Para este estudio, se mantuvieron fijos los hiperparámetros de profundidad seleccionados en E2.A (12 capas visuales, 9 textuales) y el umbral de datos (`min_images=8`).

4.2.2.1. Comparativa Global y Ganancia Respecto a la Línea Base

La evaluación comienza contrastando el rendimiento final de ambas estrategias y su mejora respecto al modelo sin entrenamiento (*baseline*).

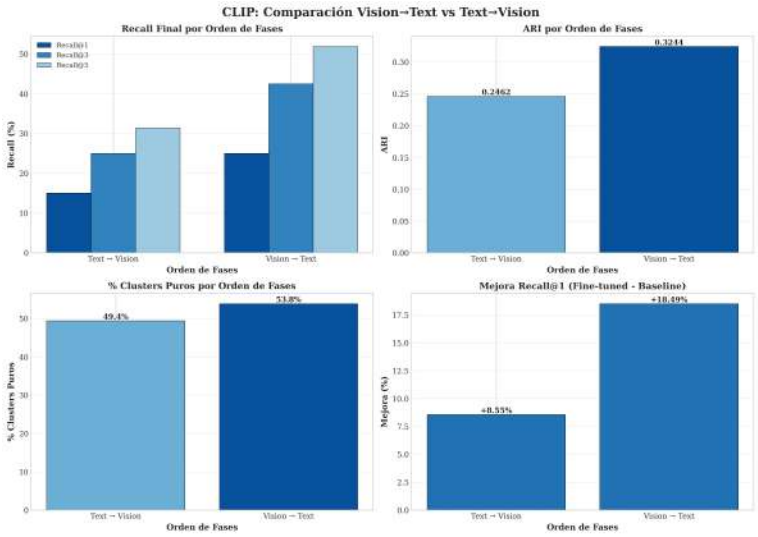


Figura 4.20: Comparación del rendimiento global según el orden de fases. La estrategia Vision → Text supera a la inversa en todas las métricas clave, destacando especialmente en la mejora del Recall@1.

La Figura 4.20 establece un claro ganador: la estrategia **Visión → Texto**. Se observa una ventaja sistemática en todas las métricas, siendo especialmente notable la diferencia en la mejora del **Recall@1** y el **ARI** (cuadrantes superiores). Esto sugiere que establecer primero una representación geométrica robusta mediante el ajuste visual facilita una alineación semántica más efectiva en la etapa posterior.

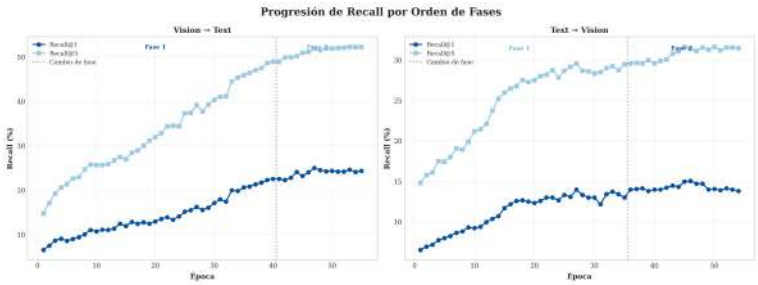


Figura 4.21: Curvas de progresión de Recall durante el entrenamiento secuencial. Nótese la estabilidad y crecimiento constante en la estrategia Vision → Text, versus la latencia en el aprendizaje de la estrategia inversa.

La dinámica temporal del aprendizaje se detalla en la Figura 4.21. La estrategia **Visión → Texto** muestra una progresión estable y pronunciada desde el inicio. Por el contrario, la estrategia inversa (**Texto → Visión**) exhibe un comportamiento subóptimo: el valor de su pendiente es menor y el valor final alcanzado también.

4.2.2.2. Dinámica de Aprendizaje y Contribución por Fase

Es crucial descomponer no solo el resultado final, sino entender el aporte marginal de cada etapa del proceso secuencial.

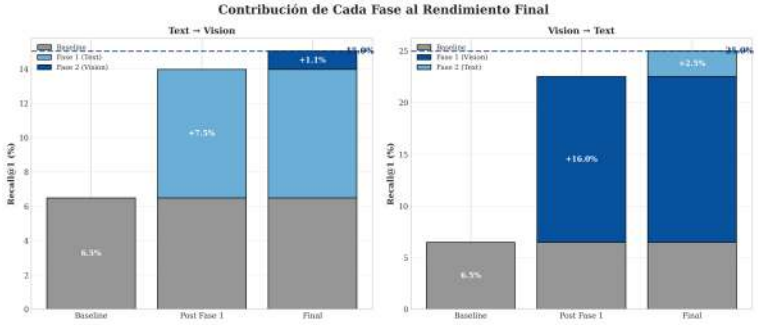


Figura 4.22: Contribución marginal de cada fase al rendimiento final. En la estrategia óptima (Vision → Text), la visión construye la base principal y el texto aporta un refinamiento adicional. En la inversa, la fase de texto aporta una ganancia mínima.

Este fenómeno se cuantifica explícitamente en la Figura 4.22. El gráfico de barras apiladas revela la anatomía del éxito: en la configuración **Visión → Texto** (Figura derecha), la Fase 1 aporta la mayor parte de la mejora estructural, mientras que la Fase 2 suma un refinamiento porcentual valioso. En la configuración inversa, la Fase 1 tiene una contribución menor, lo que confirma que el texto debe actuar como un mecanismo de soporte sobre una base visual sólida, y no como el cimiento primario del ajuste fino.

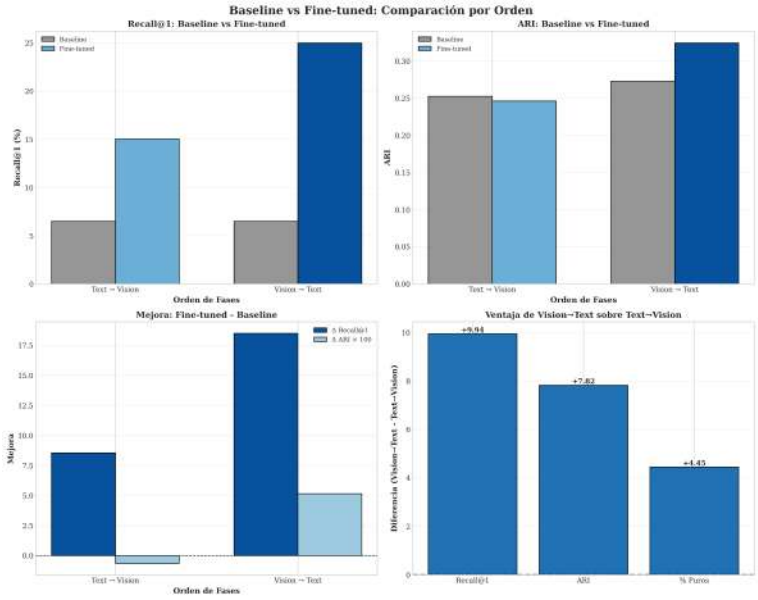


Figura 4.23: Comparativa de mejora respecto al Baseline. La estrategia Vision → Text maximiza el delta de mejora tanto en Recall@1 como en ARI.

La Figura 4.23 refuerza este análisis al comparar el incremento neto respecto al estado inicial. Se evidencia que la estrategia **Visión → Texto** no solo obtiene métricas finales más

altas, sino que maximiza el aprovechamiento de los datos de entrenamiento, logrando una diferencia de mejora significativamente mayor en todas las dimensiones evaluadas.

4.2.2.3. Calidad Estructural del Clustering

Para verificar la robustez del espacio latente generado, se analizan las métricas de agrupamiento y la topología de los clústeres resultantes.

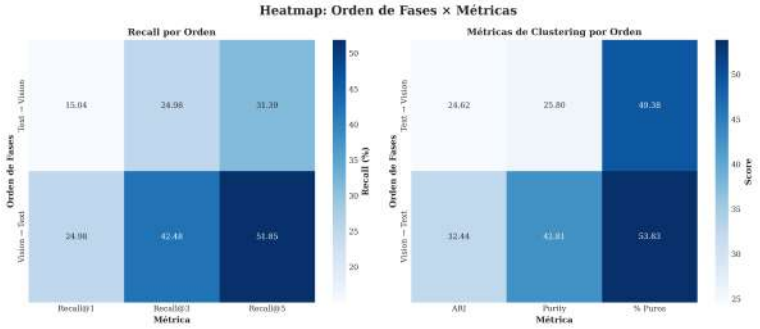


Figura 4.24: Mapa de calor de métricas globales. La intensidad de color confirma la superioridad integral de la configuración Vision → Text tanto en recuperación como en calidad de clústeres.

El mapa de calor de la Figura 4.24 permite una visualización integral del rendimiento. La fila correspondiente a **Vision → Text** presenta consistentemente valores más altos (tonos más oscuros) en todas las columnas, abarcando tanto métricas de recuperación (*Recall*) como métricas estructurales del *clustering* (ARI, Purity y porcentaje de clústeres puros).

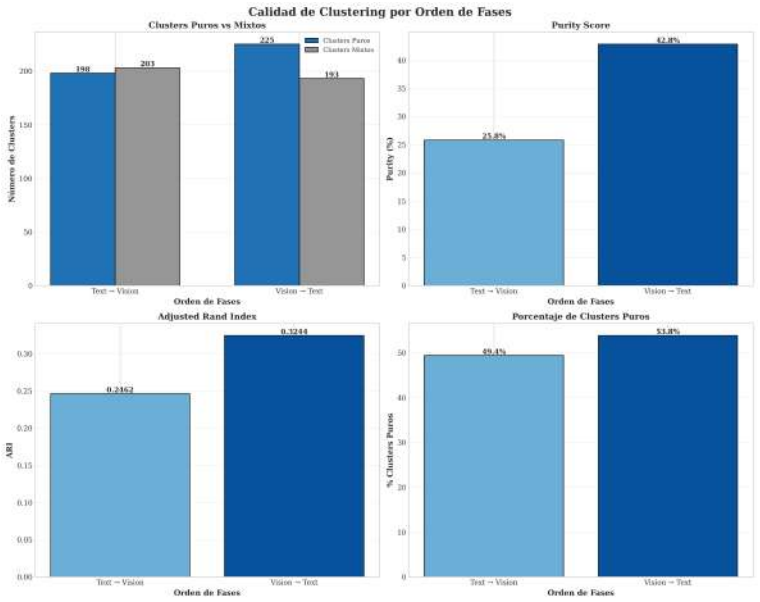


Figura 4.25: Detalle de la calidad de clustering. La estrategia Vision → Text maximiza el número de clústeres puros (barra azul oscuro) y minimiza la fragmentación.

La Figura 4.25 desglosa la composición de los grupos generados. La estrategia **Visión → Texto** produce una mayor cantidad absoluta de clústeres puros frente a mixtos. Esto indica que el modelo logra aislar con mayor precisión las identidades vehiculares en el espacio vectorial, reduciendo la ambigüedad entre modelos similares.

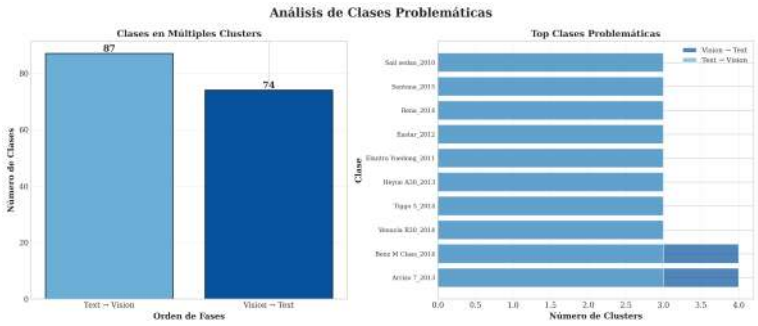


Figura 4.26: Análisis de clases problemáticas. La estrategia Vision → Text reduce significativamente el número de clases que se fragmentan o solapan en múltiples clústeres.

Un indicador crítico de la consistencia del modelo es la reducción de la confusión inter-clase. La Figura 4.26 demuestra que el orden **Visión → Texto** resulta en un menor número de "clases problematicas" (aquellas dispersas en múltiples clústeres). Esto valida que la coherencia semántica aportada por el refinamiento textual final ayuda a compactar las representaciones de una misma clase, evitando su fragmentación.

4.2.2.4. Eficiencia y Clasificación Final

Finalmente, se contrasta el rendimiento obtenido con el costo computacional para determinar el retorno de inversión de cada estrategia.

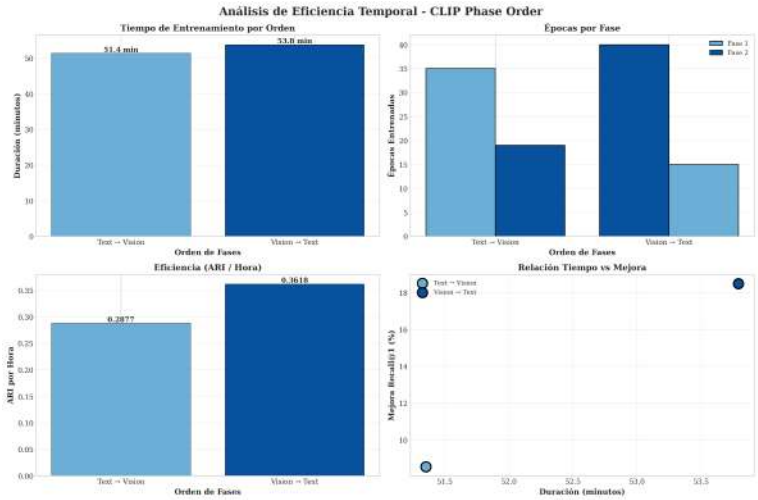


Figura 4.27: Análisis de eficiencia temporal. Aunque los tiempos brutos son similares, la estrategia Vision → Text ofrece una mayor eficiencia (ARI/Hora).

La Figura 4.27 confirma que no existe una penalización temporal significativa por elegir el orden correcto; ambos procesos consumen tiempos de cómputo similares. Sin embargo, la eficiencia calculada como ARI ganado por hora de entrenamiento favorece claramente a la secuencia **Visión → Texto**, ofreciendo una mejor relación costo-beneficio.

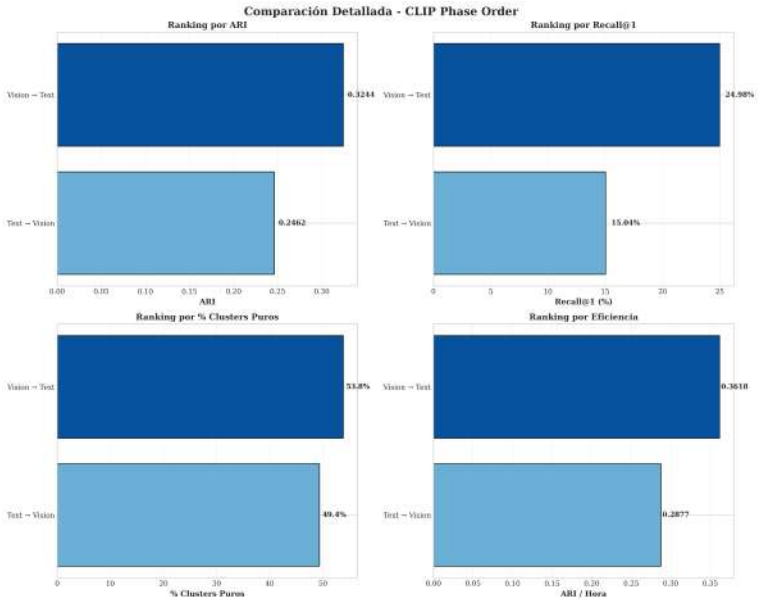


Figura 4.28: Ranking final de configuraciones. La estrategia Vision → Text ocupa el primer lugar indiscutible en todos los indicadores de desempeño.

El ranking global presentado en la Figura 4.28 sintetiza los hallazgos: la configuración **Visión → Texto** domina indiscutiblemente en ARI, Recall y Pureza, consolidándose como

la metodología superior.

4.2.2.5. Conclusión del Experimento 2 y Estandarización

Los resultados de los sub-estudios 2.A y 2.B permiten establecer un protocolo de entrenamiento optimizado para modelos multimodales en el dominio vehicular. Se concluye que:

1. La adaptación profunda del componente visual es el motor principal del rendimiento (evidenciado en 2.A).
2. El componente textual es beneficioso y complementario, pero su contribución es efectiva únicamente cuando se aplica sobre representaciones visuales ya especializadas (evidenciado en 2.B, Figura 4.22).
3. El orden secuencial **Visión** $\rightarrow$ **Texto** maximiza la sinergia entre ambas modalidades, evitando la degradación de características que ocurre en el orden inverso.

En consecuencia, para los Experimentos 3 (Sensibilidad a Datos) y 4 (Generalización), se estandariza el uso de la estrategia **Visión** $\rightarrow$ **Texto**, aplicando un descongelamiento de 12 capas para el codificador visual y 9 para el textual.

4.3. Experimento 3: Sensibilidad al Volumen de Datos

Tras establecer en el Experimento 2 que la estrategia secuencial **Visión → Texto** maximiza el rendimiento del modelo multimodal, el presente experimento somete dicha configuración óptima a pruebas de estrés variando la disponibilidad de datos. El objetivo es determinar el umbral mínimo de información necesario para que el sistema mantenga una capacidad de agrupamiento no supervisado efectiva y cuantificar la degradación del rendimiento en escenarios de extrema escasez (*Few-Shot*). Se evaluaron cinco escenarios definidos por el umbral de imágenes mínimas por clase: $k \in \{4, 6, 8, 10, 12\}$, manteniendo constantes los hiperparámetros de entrenamiento.

4.3.1. Impacto en el Rendimiento de Recuperación y Agrupamiento

El primer nivel de análisis se centra en cómo la cantidad de muestras por clase influye en las métricas globales de desempeño.

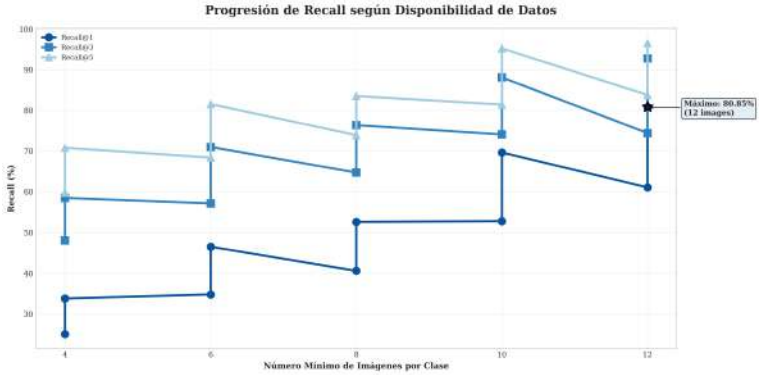


Figura 4.29: Progresión del Recall (@1, @3, @5) en función del umbral de imágenes mínimas.

La Figura 4.29 ilustra la relación entre la disponibilidad de datos y la capacidad de recuperación (*Recall*). Se evidencia un comportamiento de crecimiento no lineal, utilizando una mayor cantidad de imágenes se obtienen mejores valores de *Recall*, además, usar los valores asociados al utilizar vistas frontales o mixtas poseen un comportamiento peculiar en donde se logra apreciar que al aumentar el numero de imágenes solo frontales y comparar el recall con el resultado del valor anterior con ambas vistas, se obtiene un resultado inferior, lo que sugiere que utilizar ambas vistas es más beneficioso en casos donde las imágenes son escasas.

Figura 4.30: Progresión del ARI respecto a la disponibilidad de datos. A diferencia del Recall, la calidad estructural del agrupamiento (ARI) muestra una sensibilidad más lineal frente al aumento de datos en los escenarios iniciales.

Por su parte, la Figura 4.30 muestra la evolución del **ARI**. Es notable que, incluso en el escenario más crítico (4 imágenes), el modelo logra establecer una estructura de agrupamiento coherente superior al azar. A medida que la cantidad de imágenes usadas para el entrenamiento aumentan, tambien lo hace el valor del ARI, alcanzando su máximo: 0.86 al utilizar 12 imágenes como filtro del dataset.

4.3.2. Comparativa de Escenarios: Críticos vs. Ricos

Para entender las diferencias cualitativas entre operar con escasez o abundancia de datos, se desglosan las métricas por escenario específico.

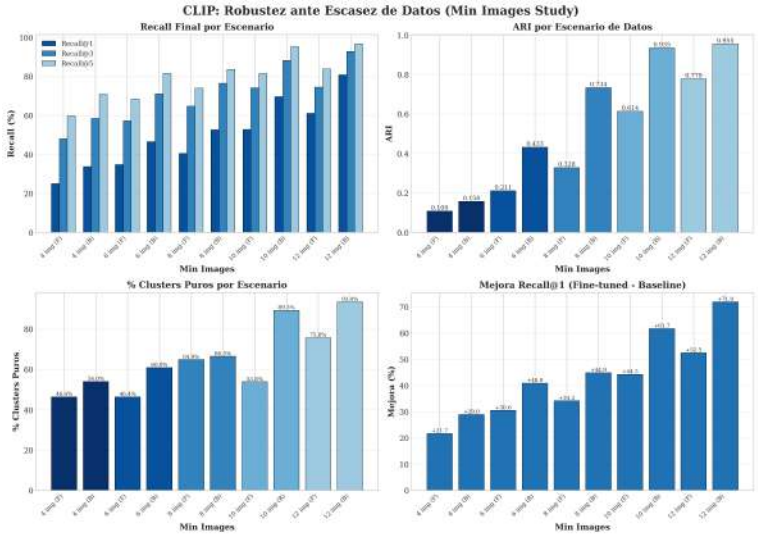


Figura 4.31: Comparativa detallada de métricas por escenario. Se contrasta el desempeño en Recall, ARI y porcentaje de clústeres puros desde el escenario crítico (4 img) hasta el escenario rico (12 img).

La Figura 4.31 revela que todas las métricas aumentan al utilizar más datos. Esto implica que el modelo se beneficia de la diversidad y cantidad de ejemplos para construir representaciones más robustas. Es importante notar que la mejora de todas las metricas no es lineal, y que además, el paso de 8 a 10 imágenes genera un salto más pronunciado en comparación con otros intervalos, lo que sugiere un efecto de umbral donde cierta cantidad de datos es necesaria para capturar la variabilidad intrínseca de las clases vehiculares.

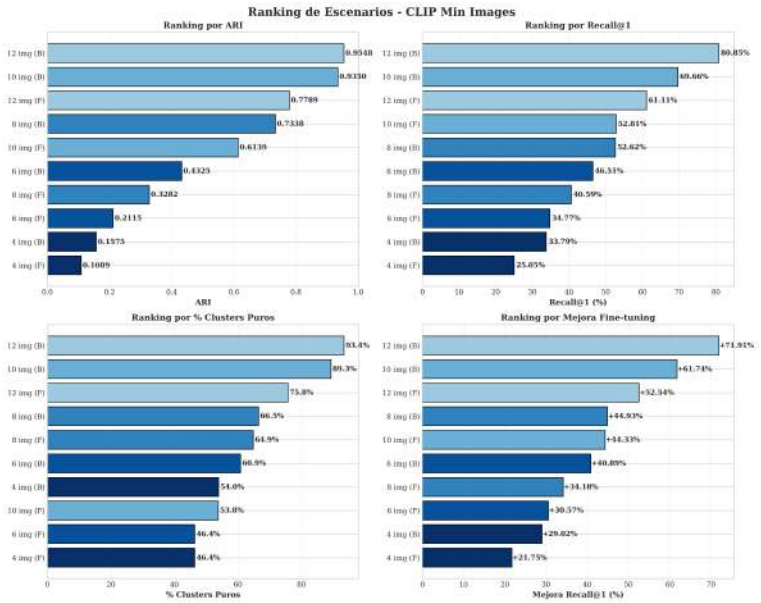


Figura 4.32: Ranking de escenarios según múltiples indicadores de desempeño. Los escenarios con mayor cantidad de datos dominan las métricas absolutas, pero los escenarios intermedios (8-10) ofrecen un balance competitivo.

El ranking presentado en la Figura 4.32 confirma la premisa de "más datos es mejor" (idea altamente estudiada en la arquitectura **Transfomer**), situando consistentemente al escenario de 12 imágenes en el primer lugar. No obstante, la diferencia con el escenario base de 8 imágenes no es prohibitiva, validando la elección de este último como un estándar representativo para los experimentos previos.

4.3.3. Mejora respecto a la Línea Base y Correlaciones

Es fundamental verificar si el proceso de *fine-tuning* aporta valor incluso cuando los datos son escasos, o si sería preferible utilizar el modelo *Zero-Shot* original en dichas situaciones.

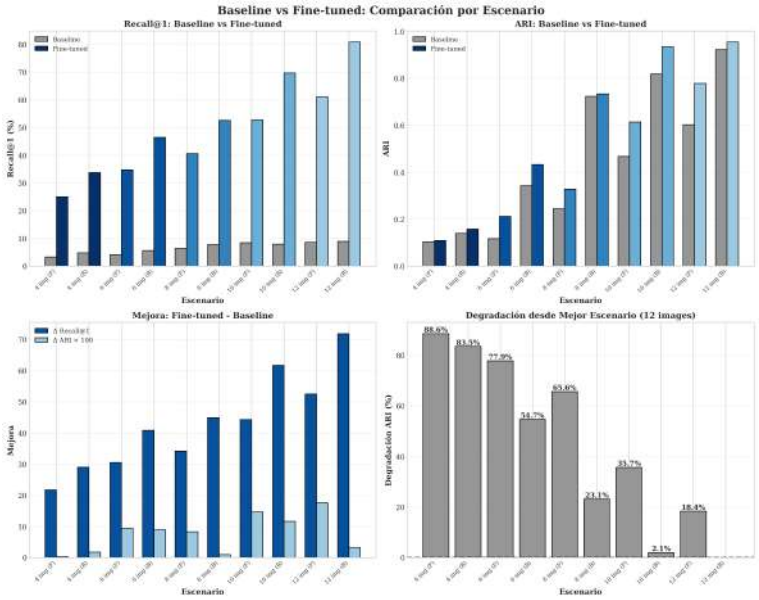


Figura 4.33: Comparación de mejora: Baseline vs Fine-tuned por escenario. Incluso con solo 4 imágenes, el ajuste fino logra superar significativamente al modelo base, demostrando la capacidad de adaptación de CLIP en régimen de Few-Shot.

La Figura 4.33 es concluyente: el ajuste fino proporciona una mejora dramática sobre la línea base (*baseline*) en todos los escenarios, sin excepción. La ganancia relativa asociada es mayor a medida que aumentan los datos. Siendo notorio el salto al utilizar 10 o 12 imágenes, donde el modelo alcanza los mejores niveles de desempeño.

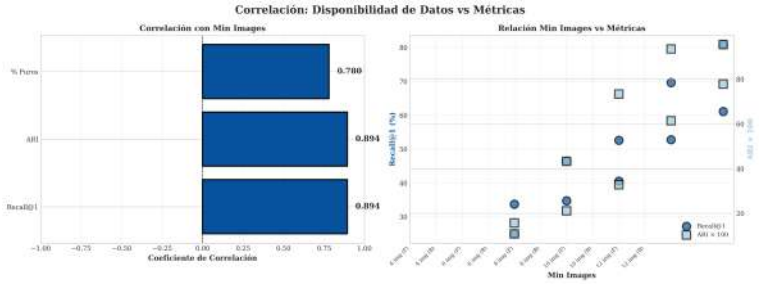


Figura 4.34: Matriz de correlación entre el número de imágenes (Min Images) y las métricas de desempeño. Existe una correlación positiva fuerte con el Recall y el ARI, validando la hipótesis de sensibilidad al volumen de datos.

El análisis de correlación (Figura 4.34) cuantifica la dependencia estadística. Se observa una correlación positiva fuerte (≥ 0.78) entre `min_images` y las métricas de rendimiento (Recall, ARI). Esto confirma que la variable "volumen de datos" es un predictor lineal del éxito del sistema dentro del rango estudiado.

4.3.4. Calidad Estructural del Espacio Latente

Más allá de los números, se evalúa cómo la cantidad de datos moldea la topología de los clústeres.

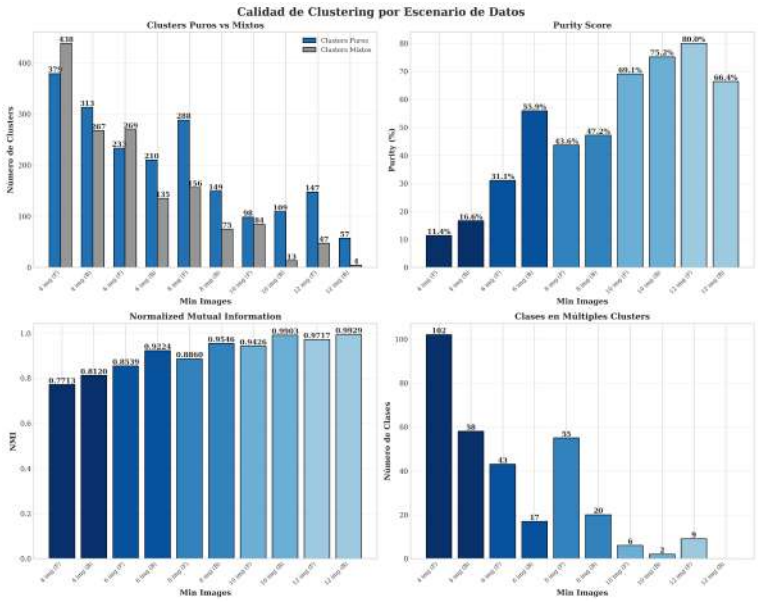


Figura 4.35: Calidad estructural del clustering por escenario. A medida que aumentan los datos, disminuye la cantidad de clústeres mixtos y se reduce el solapamiento entre clases (Overlapping Classes).

La Figura 4.35 muestra que la principal ventaja de disponer de más datos radica en la reducción de la ambigüedad. En los escenarios de 10 y 12 imágenes, el número de **Clases con Solapamiento** (clases que el modelo confunde y mezcla en un mismo clúster) disminuye drásticamente. Con pocos datos (4-6 imágenes), el espacio latente es más "ruidoso", propiciando la fusión indebida de modelos visualmente similares (e.g., sedanes genéricos de la misma marca).

4.3.5. Dinámica de Entrenamiento y Eficiencia de Datos

Finalmente, se analiza el costo de entrenar con más datos y la eficiencia del aprendizaje.

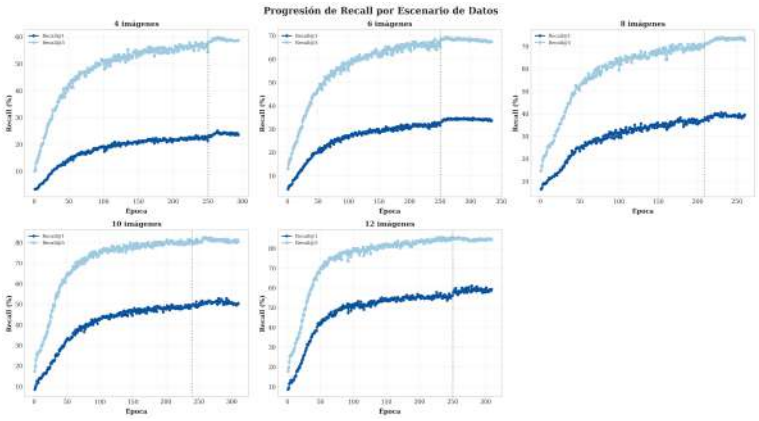


Figura 4.36: Curvas de aprendizaje (Recall) durante el entrenamiento. Los escenarios con más datos muestran una curva más estable y un punto de partida más alto, convergiendo a valores superiores.

Las curvas de la Figura 4.36 indican que la estabilidad del entrenamiento mejora con el volumen de datos. Los escenarios de 4 y 6 imágenes presentan mayor volatilidad entre épocas, mientras que los escenarios de 10 y 12 imágenes exhiben una convergencia monótona y suave.

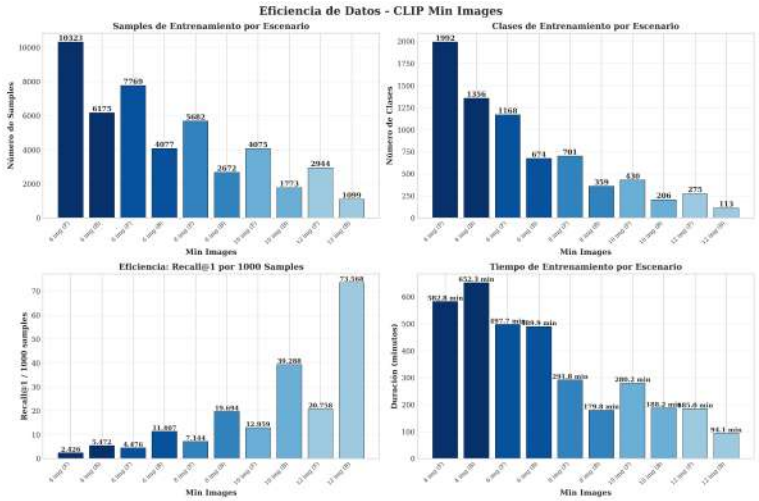


Figura 4.37: Análisis de eficiencia de datos. Se muestra el Recall obtenido por cada 1000 muestras de entrenamiento. Los escenarios de pocos datos resultan ser extremadamente eficientes, extrayendo más rendimiento por unidad de información.

Paradójicamente, como revela la Figura 4.37, los escenarios "pobres" son los más eficientes. La métrica de **Recall por 1000 samples** es máxima en el escenario de 4 imágenes y decrece a medida que se añaden datos. Esto sugiere que, aunque el rendimiento absoluto mejor con más datos, el modelo es capaz de extraer valor significativo incluso en condiciones de escasez, optimizando el uso de la información disponible, lo que resalta la versatilidad del enfoque basado en **CLIP** para adaptarse a diferentes regímenes de datos.

4.3.6. Síntesis del Experimento 3

Los resultados permiten concluir que el sistema propuesto basado en **CLIP** es altamente robusto a la escasez de datos. Aunque el rendimiento absoluto escala positivamente con el volumen de información (alcanzando su pico en 12 imágenes), el sistema es capaz de generar agrupamientos coherentes y puros con tan solo 6 u 8 imágenes por modelo. Se identifica el umbral de **8 imágenes** (utilizado como base en este estudio) como un punto de equilibrio óptimo (*sweet spot*) entre el esfuerzo de recolección de datos y la calidad del reconocimiento, validando la metodología empleada en los experimentos anteriores. A pesar de lo anterior, para el experimento de escalabilidad se opto por utilizar 10 imágenes por modelo, con el fin de maximizar el rendimiento en un escenario más exigente.

4.4. Experimento 4: Escalabilidad y Generalización (Testing Final)

Habiendo optimizado las estrategias de entrenamiento y validado la robustez ante la escasez de datos, la fase final de la experimentación busca responder dos interrogantes críticas para el despliegue en el mundo real: ¿Es la capacidad del modelo unimodal (profundidad y resolución) un factor limitante para la calidad del agrupamiento? y ¿Cómo se comporta el sistema ante la aparición de modelos de vehículos completamente desconocidos (*Open-Set*)?

Este experimento se divide en dos etapas: una centrada en el escalado de la arquitectura visual y otra enfocada en la evaluación de generalización *Zero-Shot* sobre el conjunto de prueba reservado.

4.4.1. Escalado de Arquitectura Visual

En esta etapa, se replicaron las mejores configuraciones de entrenamiento obtenidas previamente para los modelos de visión pura (Aprendizaje Métrico) sobre arquitecturas de mayor capacidad. Se comparan los modelos base contra sus versiones escaladas: **ResNet50 vs. ResNet101** (mayor profundidad) y **ViT-B/32 vs. ViT-B/16** (mayor resolución de parches).

4.4.1.1. Saturación en Arquitecturas Convolucionales

El primer hallazgo surge al analizar el escalado en redes convolucionales.

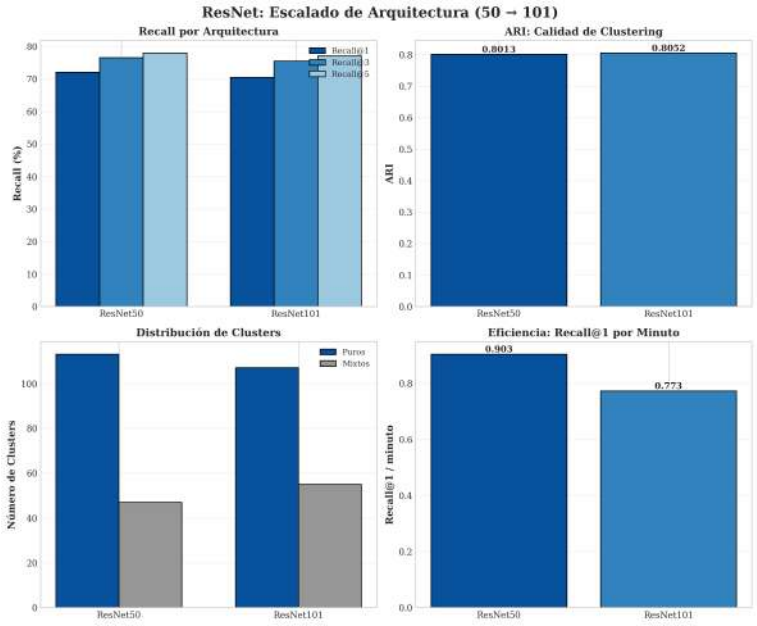


Figura 4.38: Comparativa de escalado en CNNs: ResNet50 vs ResNet101. No se observan ganancias significativas al aumentar la profundidad; de hecho, ResNet50 mantiene una ligera ventaja en eficiencia y pureza.

Como evidencia la Figura 4.38, el paso de ResNet50 a ResNet101 no se traduce en una mejora del rendimiento. Las métricas de recuperación y ARI permanecen estancadas e incluso, en ciertos casos, ResNet50 supera marginalmente a su versión más profunda. Esto sugiere que, para el dominio específico de identificación vehicular de grano fino con este volumen de datos, la arquitectura CNN alcanza un punto de **saturación de capacidad** con 50 capas. La estructura inductiva de la convolución parece haber extraído el máximo provecho de las características locales disponibles, y añadir más parámetros solo incrementa el costo computacional sin aportar mayor discriminación semántica.

4.4.1.2. Impacto de la Resolución en Transformers

En contraste con las CNNs, los *Vision Transformers* exhiben una sensibilidad crítica a la escala, específicamente al tamaño de los parches de entrada.

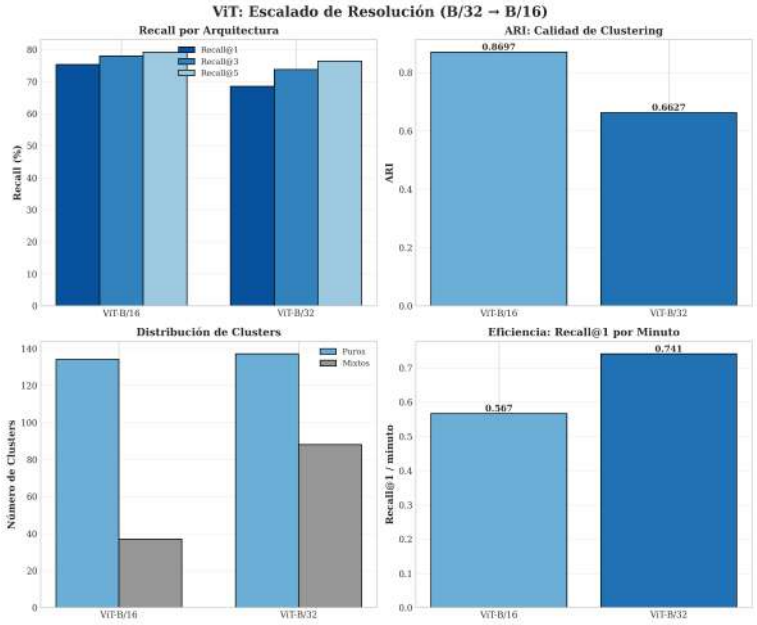


Figura 4.39: Comparativa de escalado en Transformers: ViT-B/32 vs ViT-B/16. La reducción del tamaño de parche (aumentando la secuencia de entrada) genera un salto cualitativo en el ARI, pasando de ~ 0.66 a ~ 0.86.

La Figura 4.39 muestra una mejora dramática al pasar de ViT-B/32 a ViT-B/16. El **ARI** experimenta un salto cualitativo, elevándose de niveles cercanos a 0.66 hasta 0.86. Al analizar la distribución de clústeres, se observa que la cantidad de clústeres mixtos disminuye significativamente, mientras que la cantidad de clústeres puros se mantiene estable. Esto indica que la mayor resolución espacial (procesando parches de 16×16 píxeles en lugar de 32×32) permite al mecanismo de atención capturar detalles sutiles —como la forma de los faros o logotipos— que antes se perdían, reduciendo la confusión entre modelos similares sin sacrificar la cohesión. Aunque la eficiencia disminuye debido al cómputo cuadrático de la atención, la ganancia en calidad de agrupamiento justifica plenamente el uso de ViT-B/16.

4.4.1.3. Mantenimiento de CLIP-ViT-B/32

Respecto a la arquitectura multimodal, se determinó mantener la configuración base **CLIP-ViT-B/32**, descartando el escalado para este estudio. Esta decisión se fundamenta en dos razones críticas:

1. **Superioridad Empírica:** Como se observa en la comparativa general (Figura 4.40), la configuración CLIP-ViT-B/32 ya demuestra una superioridad sistemática frente a las demás arquitecturas base, alcanzando métricas que satisfacen los requerimientos operativos sin necesidad de escalar.

2. **Eficiencia Arquitectónica:** Como se establece en el trabajo original de Radford et al. [7], la arquitectura ViT-B/32 representa un punto óptimo de eficiencia y rendimiento para tareas de *zero-shot*, siendo el estándar de referencia más robusto antes de incurrir en costos computacionales prohibitivos.

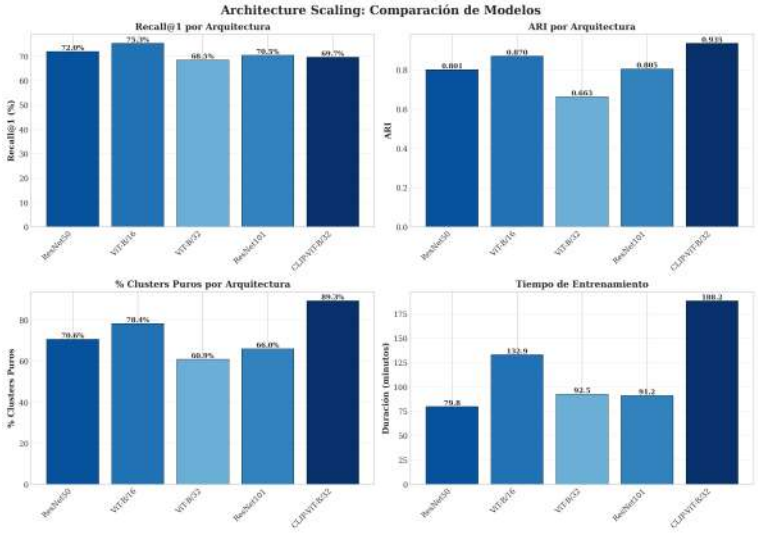


Figura 4.40: Comparativa general de arquitecturas. CLIP-ViT-B/32 se mantiene competitivo globalmente frente a los modelos unimodales escalados.

4.4.1.4. Rankings Globales y Selección de Modelos

Figura 4.41: Rankings globales de arquitectura. Se observa una jerarquía clara donde los Transformers escalados (ViT-B/16) lideran en ARI, mientras CLIP destaca en métricas complementarias.

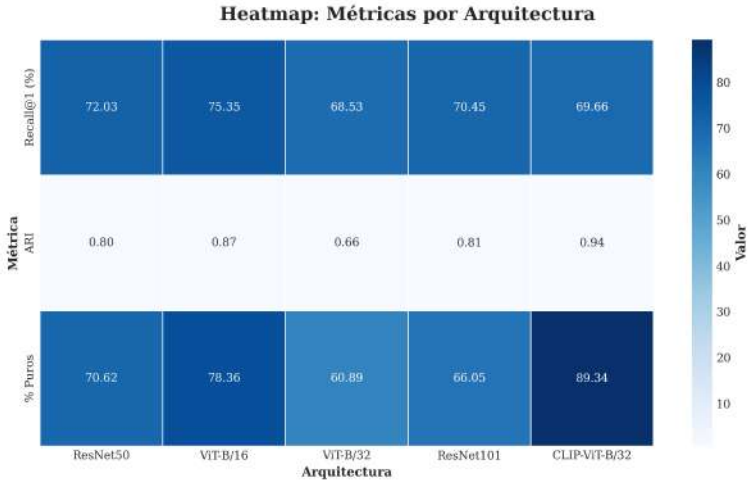


Figura 4.42: Mapa de calor de rendimiento. El ARI mejora consistentemente con el escalado en Vision Transformers, mientras que CLIP y ResNet50 mantienen un balance sólido.

El análisis global (Figuras 4.41 y 4.42) permite establecer una jerarquía de rendimiento clara:

1. **Nivel Superior:** ViT-B/16 (Visión Pura) logra métricas estructurales sobresalientes (ARI > 0.8), demostrando que la granularidad de la atención es el factor más determinante en ausencia de texto.

2. **Nivel Competitivo:** CLIP-ViT-B/32 y ResNet50 ofrecen un balance sólido. CLIP destaca por su capacidad de generalización semántica y ResNet50 por su eficiencia.

3. **Nivel Inferior:** ResNet101 no logra justificar su costo, quedando relegado por la saturación de la arquitectura.

4.4.1.5. Selección para Testing Final

Basado en estos resultados, se seleccionan los siguientes tres modelos representativos para la evaluación de generalización final:

- **ResNet50:** Como el mejor exponente de las CNNs.
- **ViT-B/16:** Como el modelo de visión pura más potente tras el escalado.
- **CLIP-ViT-B/32:** Como la propuesta multimodal definitiva y eficiente.

4.4.2. Evaluación de Generalización (*Open-Set Testing*)

Tras validar el desempeño en las clases regulares, se procedió a evaluar la robustez del sistema en un escenario de "estrés" de datos. El conjunto de datos completo consta de un total de **3598 clases**, las cuales presentan un desbalance extremo: **359 clases regulares** (utilizadas durante el entrenamiento y validación estándar) y **3239 clases *One-Shot*** (clases de muy baja frecuencia, con menos de 8 imágenes, que simulan la "cola larga" de la distribución real).

Este experimento consiste en un análisis de sensibilidad donde se inyectan progresivamente las clases *One-Shot* al conjunto de prueba, variando su proporción desde un 0 % (solo clases regulares) hasta un 100 % (el dataset completo con las 3239 clases de baja frecuencia incluidas). El objetivo es medir cómo se degrada la calidad del agrupamiento a medida que el sistema debe gestionar un número masivo de categorías con información visual mínima.

4.4.2.1. Degradación del Rendimiento ante la Cola Larga

La incorporación de clases *One-Shot* introduce un desafío doble: aumenta exponencialmente el número de clústeres a identificar y reduce la densidad de información por clúster.

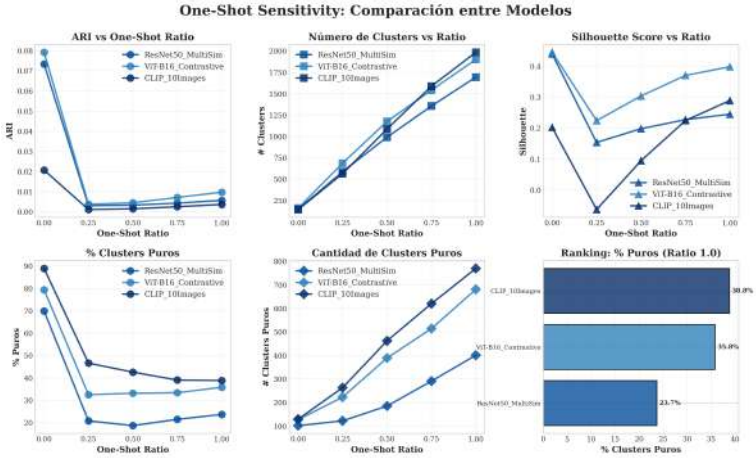


Figura 4.43: Curvas de sensibilidad de la Pureza de Clústeres ante la incorporación de clases One-Shot. Se observa la degradación del rendimiento desde el escenario base (solo regulares) hasta el escenario completo (Ratio 1.0). CLIP (línea azul oscura) demuestra la mayor estabilidad.

La Figura 4.43 ilustra la trayectoria de degradación para los tres mejores modelos.

- **Escenario Base (Ratio 0.0):** Cuando se evalúa exclusivamente sobre las clases regulares, **CLIP** demuestra una superioridad notable con una pureza del **89.0 %**, seguido por **ViT-B/16** (79.4 %) y **ResNet50** (69.9 %).
- **Transición a la Cola Larga:** A medida que se agregan las clases de baja frecuencia, el rendimiento decae en todos los modelos. Sin embargo, la pendiente de degradación es distinta. Al llegar al escenario completo (**Ratio 1.0**, con 3239 clases *One-Shot* añadidas), **CLIP** logra estabilizarse manteniendo una pureza del **38.8 %**.
- **Comparativa Final:** En el escenario más exigente, **CLIP** supera a **ViT-B/16** (35.8 %) y mantiene una ventaja considerable sobre **ResNet50**, que cae hasta un **23.7 %** de pureza.

Estos resultados indican que, aunque todos los modelos sufren ante la escasez de datos por clase, los modelos basados en **Transformers** (y especialmente los multimodales) conservan una capacidad superior para estructurar el espacio latente incluso cuando la información visual es mínima.

4.4.2.2. Análisis de la Estructura de Clústeres en el Dataset Completo

Para entender cómo los modelos organizan esta masiva cantidad de clases minoritarias, se analiza la distribución de clústeres en el punto de máxima entropía (Ratio 1.0).

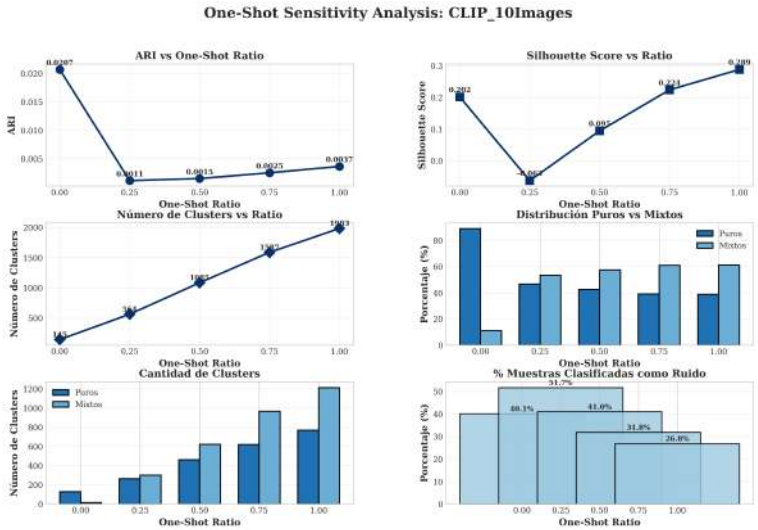


Figura 4.44: Distribución de clústeres para CLIP-ViT-B/32 con el dataset completo. El modelo logra identificar exitosamente 770 clústeres puros (barra azul sólida) en un entorno dominado por clases de baja frecuencia.

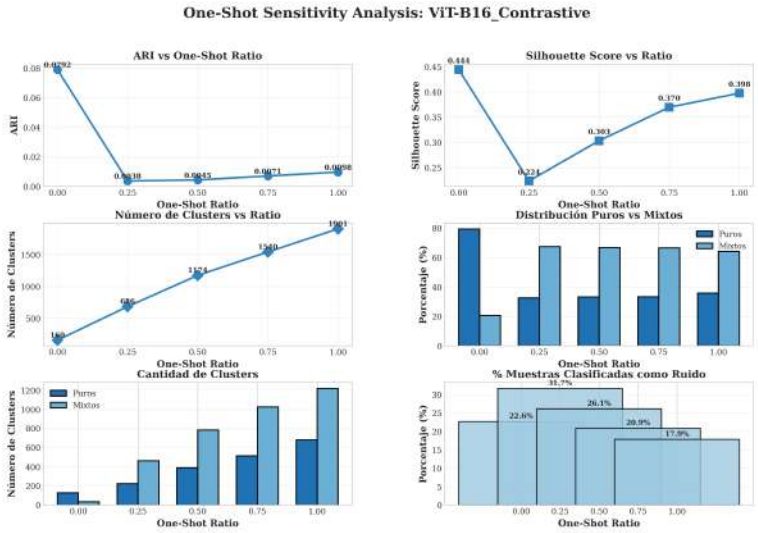


Figura 4.45: Distribución de clústeres para ViT-B/16 (Contrastive). Genera 681 clústeres puros, mostrando una capacidad competitiva pero inferior a CLIP en la resolución de clases difíciles.

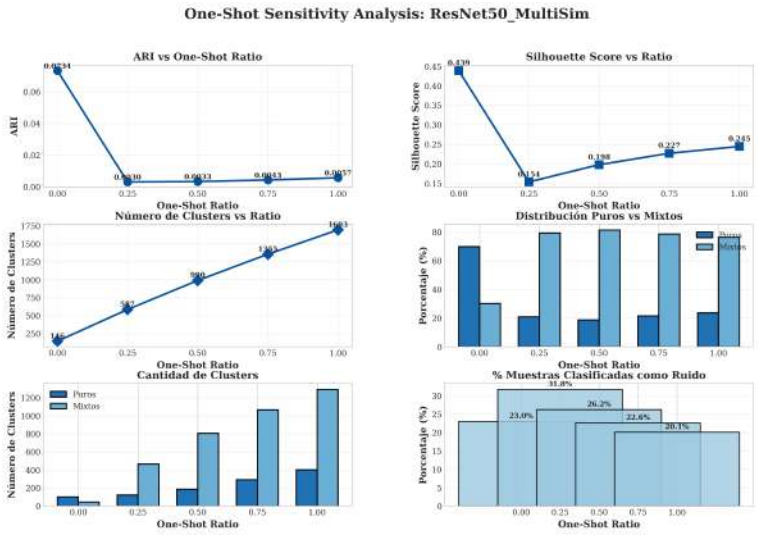


Figura 4.46: Distribución de clústeres para ResNet50 (MultiSimilarity). La mayoría de las clases One-Shot terminan fusionadas en clústeres mixtos (barra gris) o dispersas como ruido.

Las Figuras 4.44, 4.45 y 4.46 presentan el desglose final:

1. **CLIP (Figura 4.44):** Es el modelo que mejor gestiona la "cola larga", generando la mayor cantidad de clústeres puros (**770**) y manteniendo una proporción de ruido controlada. Esto sugiere que su pre-entrenamiento multimodal le permite encontrar características distintivas incluso en ejemplos únicos.

2. **ViT-B/16 (Figura 4.45):** Se mantiene como una alternativa robusta con **681** clústeres puros, aunque muestra una tendencia ligeramente mayor a mezclar clases visualmente similares (1220 clústeres mixtos vs 1213 de CLIP) a pesar de tener menos pureza total.

3. **ResNet50 (Figura 4.46):** Muestra dificultades significativas para escalar a este volumen de clases, produciendo solo **401** clústeres puros. La gran mayoría de las clases *One-Shot* son absorbidas en clústeres mixtos (76.3 % de mezcla), evidenciando que las características aprendidas por la CNN no tienen la granularidad suficiente para separar miles de clases con pocas muestras.

4.4.2.3. Conclusión del Testing Final

El análisis de sensibilidad confirma que la arquitectura **CLIP** no solo es superior en condiciones ideales, sino que es la más resiliente ante la distribución de datos real de "cola larga". Su capacidad para mantener cerca del 40 % de pureza en un dataset dominado por más de 3000 clases de baja frecuencia valida su idoneidad para sistemas de identificación vehicular en el mundo real, donde la aparición de modelos raros o con pocos datos es la norma y no la excepción.

Capítulo 5

Conclusiones

5.1. Conclusiones del Estudio

La presente investigación ha logrado validar la hipótesis central planteada: la utilización de representaciones profundas (*embeddings*), optimizadas mediante técnicas de aprendizaje métrico y explotadas a través de algoritmos de agrupamiento no supervisado, constituye una solución robusta y efectiva para la identificación de modelos de vehículos, superando las limitaciones de los clasificadores supervisados tradicionales en escenarios de información limitada.

A partir de los experimentos realizados y el análisis exhaustivo de los resultados, se desprenden las siguientes conclusiones principales:

- **Viabilidad del Enfoque No Supervisado:** Se ha demostrado que es posible identificar modelos de vehículos con alta precisión sin necesidad de una capa de clasificación final entrenada para clases específicas. La estructura semántica del espacio latente, cuando es optimizada correctamente, permite que los vehículos del mismo modelo se agrupen naturalmente, facilitando su identificación mediante *clustering*.

- **Superioridad de las Arquitecturas Multimodales:** El modelo **CLIP** se posicionó consistentemente como la arquitectura más competente, superando a los modelos de visión pura (**ViT** y **ResNet50**). La incorporación de información semántica textual durante el pre-entrenamiento dota al modelo de una capacidad de generalización superior, permitiéndole distinguir matices visuales sutiles entre modelos de vehículos que resultan indistinguibles para redes entrenadas únicamente con imágenes.

- **Resiliencia ante la "Cola Larga" y Escenarios *Few-Shot*:** Uno de los hallazgos más significativos es la robustez de **CLIP** en escenarios de datos desbalanceados. Mientras que las arquitecturas tradicionales como ResNet50 degradan su rendimiento drásticamente al enfrentar clases con pocas muestras (clases *One-Shot* o *Few-Shot*), CLIP logró mantener una alta pureza en los agrupamientos, identificando correctamente modelos raros o poco representados. Esto valida su idoneidad para aplicaciones reales donde la distribución de vehículos es naturalmente de cola larga.

- **Efectividad del *Fine-Tuning* con Aprendizaje Métrico:** La estrategia de ajuste fino utilizando funciones de pérdida contrastivas y angulares demostró ser fundamental para compactar las clases en el espacio latente y aumentar la separabilidad entre ellas.

Este paso es crítico para transformar un modelo pre-entrenado genérico en un extractor de características especializado en el dominio vehicular.

- **Calidad del Embedding como Métrica Principal:** El estudio evidenció que las métricas tradicionales de entrenamiento supervisado (como *Accuracy* o *Recall*) no siempre correlacionan directamente con la capacidad de agrupamiento no supervisado. La evaluación basada en la "pureza del clúster" y la coherencia del espacio latente resultó ser un indicador más fiel del rendimiento real del sistema en la tarea de descubrimiento de modelos.

En síntesis, el sistema propuesto ofrece una alternativa flexible y escalable para la vigilancia y análisis vehicular, eliminando la necesidad de reentrenar modelos cada vez que aparece un nuevo tipo de vehículo en el mercado, y trasladando la complejidad del problema desde la clasificación supervisada hacia la calidad de la representación y el agrupamiento de datos.

5.2. Trabajos Futuros

A pesar de los resultados prometedores, la investigación abre nuevas interrogantes y avenidas de exploración que podrían potenciar aún más el desempeño y la aplicabilidad del sistema. A continuación, se proponen líneas de trabajo futuro:

- **Exploración con Imágenes No Etiquetadas:** Investigar el uso de técnicas de aprendizaje auto-supervisado (*Self-Supervised Learning*) utilizando grandes volúmenes de imágenes de vehículos sin ninguna etiqueta (ni textual ni de clase). Esto permitiría aprovechar la inmensa cantidad de datos visuales disponibles en la web o en cámaras de seguridad sin el costo de la curación manual.

- **Sistemas de Recuperación (*Retrieval*):** Evaluar directamente la capacidad de los *embeddings* en tareas de recuperación de imágenes basada en contenido (*Content-Based Image Retrieval*). Esto implicaría medir la precisión del sistema al buscar "vehículos similares" a una imagen de consulta, lo cual tiene aplicaciones directas en forense digital y búsqueda de vehículos sospechosos.

- **Análisis de Identidad y Re-identificación:** Explorar si la granularidad de los *embeddings* permite llegar a un nivel de identificación de instancia única (Re-identificación vehicular), distinguiendo no solo el modelo, sino el vehículo específico (basándose en detalles como adhesivos, daños o accesorios), o por el contrario, forzar la invarianza a estos detalles para robustecer la clasificación de modelo.

- **Ingeniería de Etiquetas Textuales (*Prompt Engineering*):** En el contexto del *fine-tuning* de modelos multimodales, estudiar el impacto de la redacción y el nivel de detalle de las etiquetas textuales. Experimentar con descripciones más ricas (incluyendo año, tipo de carrocería, país de origen) versus etiquetas simples, para determinar cómo el lenguaje influye en la estructuración del espacio visual.

- **Nuevos Algoritmos de Reducción y *Clustering*:** Evaluar algoritmos de reducción de dimensionalidad y agrupamiento más recientes o específicos para variedades de alta dimensión, que puedan preservar mejor la topología local y global de los datos sin requerir una búsqueda de hiperparámetros tan exhaustiva.

- **Arquitecturas Híbridas:** Investigar el diseño de arquitecturas que combinen explícitamente las fortalezas de las redes convolucionales (para la extracción de texturas y patrones locales de bajo nivel) con los *Transformers* (para la comprensión semántica global), buscando un equilibrio óptimo entre precisión en los detalles y abstracción conceptual.

Bibliografía

[1] V. Parra, "Gracias a la vida." en *Las Últimas Composiciones*, 1966. Santiago, Chile: RCA Victor.

[2] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ, USA: Pearson, 3rd ed., 2009.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[4] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 2nd ed., 2022.

[5] A. Vaswani *et al.*, "Attention is all you need," *arXiv preprint*, 2017.

[6] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint*, 2020.

[7] A. Radford, J. W. Kim, S. Chen, G. Hallacy, D. Amodei, S. Amodei, and I. Sutskever, "Learning transferable visual models from natural language supervision," *arXiv preprint arXiv:2103.00020*, 2021.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 248–255, IEEE, 2009.

[9] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[10] A. Babenko *et al.*, "Neural codes for image retrieval," *arXiv preprint*, 2014.

[11] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, pp. 37–52, 8 1987.

[12] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *J. Mach. Learn. Res.*, vol. 9, no. 86, pp. 2579–2605, 2008.

[13] L. McInnes, J. Healy, N. Saul, and L. Großberger, "Umap: Uniform manifold approximation and projection," *J. Open Source Softw.*, vol. 3, no. 29, p. 861, 2018.

[14] Scikit-learn, "Digits dataset." https://scikit-learn.org/stable/datasets/toy_dataset.html#digits-dataset, 2011. Accessed: 2025-12-01.

[15] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, 1999.

[16] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Stat. Probab.*, vol. 1, pp. 281–297, 1967.

[17] D. Müllner, "Modern hierarchical, agglomerative clustering algorithms," *arXiv preprint*

arXiv:1109.2378, Sep. 2011. https://arxiv.org/abs/1109.2378.

[18] M. Ester *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Discov. Data Min. (KDD)*, pp. 226–231, 1996.

[19] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," *ACM SIGMOD Record*, vol. 28, pp. 49–60, Jun 1999.

[20] R. J. G. B. Campello *et al.*, "Density-based clustering based on hierarchical density estimates," in *Adv. Knowl. Discov. Data Min.*, vol. 7819 of *Lect. Notes Comput. Sci.*, pp. 160–172, 2013.

[21] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *J. Intell. Inf. Syst.*, vol. 17, no. 2/3, pp. 107–145, 2001.

[22] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Appl. Math.*, vol. 20, pp. 53–65, 1987.

[23] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-1, no. 2, pp. 224–227, 1979.

[24] T. Calinski and J. Harabasz, "A dendrite method for cluster analysis," *Communication in Statistics- Theory and Methods*, vol. 3, pp. 1–27, 1 1974.

[25] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, pp. 193–218, Dec. 1985.

[26] I. S. Dhillon, S. Mallela, and D. S. Modha, "Information-theoretic co-clustering," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, (New York, NY, USA), p. 89–98, Association for Computing Machinery, 2003.

[27] F.-L. Chen, D.-Z. Zhang, M.-L. Han, X.-Y. Chen, J. Shi, S. Xu, and B. Xu, "Vlp: A survey on vision-language pre-training," *Machine Intelligence Research*, vol. 20, p. 38–56, Jan. 2023.

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

[29] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, pp. 5962–5979, Jun 2021.

[30] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, pp. 1788–1795, 2006.

[31] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International Conference on Machine Learning (ICML)*, pp. 1597–1607, 2020.

[32] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," *arXiv preprint*, 2015.

[33] X. Wang, R. Ouyang, K. Liu, F. Wang, Y. Ouyang, T. Zhu, X. Wang, and Z. Yin, "Multi-similarity loss with general pair weighting for deep metric learning," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5022–5030, 2019.

[34] R. Grainger *et al.*, "Paca-vit: Learning patch-to-cluster attention in vision transformers," *arXiv preprint*, 2022.

[35] H. Taleb and C. Wang, "Transformer-based vehicle re-identification with multiple details," in *Proc. 5th Int. Conf. Image, Video Process. Artif. Intell. (IVPAI)*, p. 22, 2024.

[36] B. He, J. Li, Y. Zhao, and Y. Tian, "Part-regularized near-duplicate vehicle re-identification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3997–4005, 2019.

[37] S. Li, L. Sun, and Q. Li, "Clip-reid: Exploiting vision-language model for image re-identification without concrete text labels," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, pp. 1405–1413, 2023.

[38] B. Wang, Y. Liang, L. Cai, H. Huang, and H. Zeng, "Image re-identification: Where self-supervision meets vision-language learning," 2024.

[39] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, "Learning to prompt for vision-language models," *International Journal of Computer Vision*, vol. 130, p. 2337–2348, July 2022.

[40] R. Laroca *et al.*, "A robust real-time automatic license plate recognition based on the yolo detector," *arXiv preprint*, 2018.

[41] Jenoptik, "ANPR: Automatic number plate recognition." https://www.jenoptik.com/products/civil-security/anpr. Accessed: 2025-06-30.

[42] ParkPow, "Parking Management Software | ParkPow." https://parkpow.com/, 2020. Accessed: 2025-06-30.

[43] Sighthound, "Sighthound ALPR+ | Advanced license plate recognition & vehicle analytics." https://www.sighthound.com/products/alpr. Accessed: 2025-06-30.

[44] L. Yang, P. Luo, C. C. Loy, and X. Tang, "A large-scale car dataset for fine-grained categorization and verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Anexos

Anexo A. Pseudocódigos

A.1. Algoritmos de *Fine-Tuning*

A.1.1. Modelos de Visión Unimodal

Código A.1: Ciclo de Entrenamiento para Modelos de Visión (Clasificación).

```
input: DataLoader D (Muestreo Estándar), Modelo M, Optimizador opt, Loss L (Cross-
    ↪ Entropy)
output: Modelo Optimizado M
begin
   for epoch in 1 to TotalEpochs do
      for batch (images, labels) in D do
         opt.zero_grad()

         // Forward: Backbone -> Embeddings -> Classification Head
         embeddings ← M.backbone(images)
         logits ← M.head(embeddings)

         // Calcular pérdida de clasificación (Cross-Entropy) sobre los logits
         loss ← L(logits, labels)

         // Backward pass con escalado de gradientes (AMP)
         scaler.scale(loss).backward()
         scaler.step(opt)
         scaler.update()
      end

      // Validación basada en Exactitud (Accuracy)
      Validar_Accuracy(M, D_val)
   end
   return M
end
```

A.1.2. Modelos Multimodales (CLIP)

Código A.2: Ciclo de Entrenamiento para Modelos de Visión (Metric Learning).

```
input: DataLoader D (Muestreo P × K), Modelo M, Optimizador opt, Loss L
output: Modelo Optimizado M
begin
    for epoch in 1 to TotalEpochs do
        for batch (images, labels) in D do
            opt.zero_grad()

            // Forward: Obtener embeddings aplanados (batch, dim)
            embeddings ← M.forward(images)

            // Calcular pérdida en el espacio de embeddings
            loss ← L(embeddings, labels)

            // Backward pass con escalado de gradientes (AMP)
            scaler.scale(loss).backward()
            scaler.step(opt)
            scaler.update()
        end

        // Validación basada en recuperación (Recall@K) sobre embeddings normalizados
        Validar_KNN(M, D_val)
    end
    return M
end
```

A.1.3. Entrenamiento por Fases para CLIP

Código A.3: Ciclo de Entrenamiento por Fases para CLIP.

```
input: DataLoader D, ModeloCLIP M, Loss L (NT-Xent/CLIPLoss)
// Fases configurables: TextEncoder, VisionLayers, Projections
begin
    for phase in Fases_Configuradas do
        Congelar_Capas_No_Activas(M, phase)
        opt ← ConfigurarOptimizador(phase)

        for epoch in 1 to EpochsPerPhase do
            for batch (images, texts, labels) in D do
                // Forward multimodal
                img_emb, text_emb ← M.forward(images, texts)

                // Pérdida Contrastiva simétrica (Imagen <-> Texto)
                loss ← L(img_emb, text_emb)

                loss.backward()
                opt.step()
            end
        end
    end
    return M
```

```
end
```

A.2. Algoritmos de optimización de hiperparámetros

A.2.1. Optimización de Reducción de Dimensionalidad

Código A.4: Optimización de Reducción de Dimensionalidad.

```
input: Embeddings E, Algoritmo A, Trials T
output: Mejores Parámetros θ*
begin
    study ← crear\_estudio(direction='maximize')

    for i in 1 to T do
        θ ← sugerir\_hiperparametros(trial, A)

        // Reducción (Manejo de memoria eficiente)
        if A es PCA and N > 10000 then
            E_red ← IncrementalPCA(E, θ)
        else
            E_red ← A(E, θ)
        end

        // Evaluación
        if N_muestras < 2 return -1

        sil ← SilhouetteScore(E_red)

        if A in [t-SNE, UMAP] then
            // Calcular fidelidad de vecindad para métodos no lineales
            trust ← Trustworthiness(E, E_red)
            // Score compuesto ponderado
            score ← 0.7 · ((sil + 1)/2) + 0.3 · trust
        else
            score ← (sil + 1)/2
        end

        study.report(score)
    end
    return study.best_params
end
```

A.2.2. Optimización de Clustering

Código A.5: Optimización de Clustering.

```
input: Embeddings E, Etiquetas L_true, Algoritmo A, Trials T
output: Mejores Parámetros θ*
begin
    study ← crear\_estudio(direction='maximize')
```

```
    for i in 1 to T do
        θ ← sugerir\_hiperparametros(trial, A)

        // Clustering
        clusterer ← CrearClusterer(A, θ)
        L_pred ← clusterer.fit_predict(E)

        // Validaciones de sanidad (Evitar degeneración)
        n_clusters ← unique(L_pred)
        n_ruido ← count(L_pred == −1)

        if n_clusters < 2 or n_clusters > 0.6N return -1
        if A in [DBSCAN, HDBSCAN] and n_ruido > 0.4N return -1

        // Cálculo de Métricas
        sil ← Silhouette(E, L_pred)
        ch ← CalinskiHarabasz(E, L_pred)
        ari ← AdjustedRand(L_true, L_pred)
        nmi ← NormalizedMutualInfo(L_true, L_pred)

        // Normalización y Score Compuesto
        score ← 0.35 · norm(sil) + 0.25 · norm(ch) + 0.25 · max(0, ari) + 0.15 · nmi

        study.report(score)
    end
    return study.best_params
end
```

Anexo B. Diagrama de Optuna

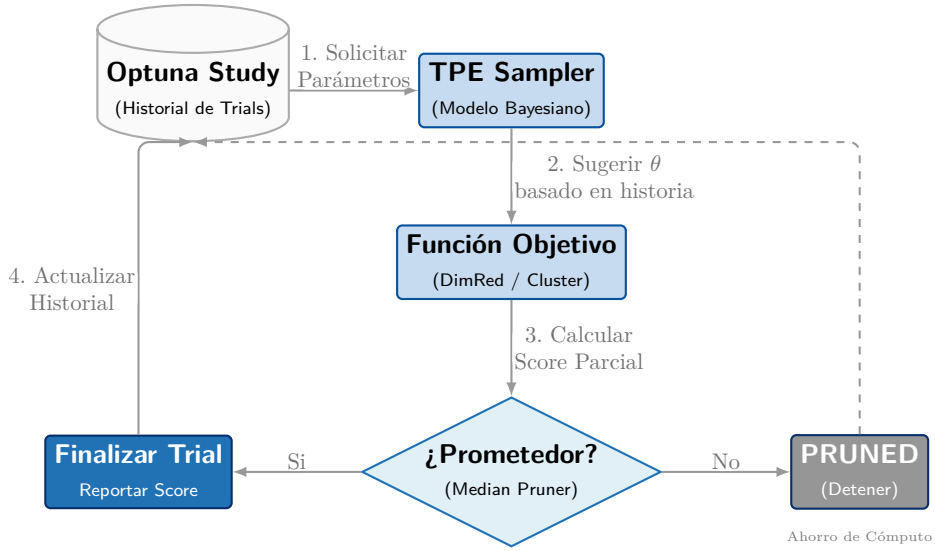


Figura B.1: Ciclo de optimización bayesiana con Optuna. El muestreador TPE sugiere hiperparámetros basándose en el historial del estudio, mientras que el pruner detiene prematuramente los intentos de bajo rendimiento.

Anexo C. Repositorio del código fuente

El código fuente desarrollado para la experimentación y análisis de resultados está disponible en un repositorio público de GitHub, accesible en la siguiente dirección: comp-cars-deep-learning-analysis.

Anexo D. Resultados Completos de la Memoria

El directorio remoto contenido en Google Drive contiene todos los artefactos generados durante la ejecución de los experimentos, incluyendo modelos entrenados, datos crudos de evaluación, análisis estadísticos y visualizaciones.