Escuela
Politécnica
Superior

Adaptación al dominio de imágenes 360 para la ayuda a la conducción de motos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor:
Alfonso Ruiz Martínez
Tutor/es:
Marcelo Saval Calvo
Antonio Javier Gallego Sánchez

Diciembre 2023

Universitat d'Alacant
Universidad de Alicante

Adaptación al dominio de imágenes 360 para la ayuda a la conducción de motos

Trabajo Fin de Grado

Autor
Alfonso Ruiz Martínez

Tutor/es
Marcelo Saval Calvo
Departamento de Tecnología Informática y Computación
Antonio Javier Gallego Sánchez
Departamento de Lenguajes y Sistemas Informáticos

Grado en Ingeniería Informática

Escuela
Politécnica
Superior

Universitat d'Alacant
Universidad de Alicante

ALICANTE, Diciembre 2023

Justificación y Objetivos

En este trabajo de fin de grado se estudiará e implementará un sistema de ayuda a la conducción en motocicletas mediante visión por computador y *deep learning*, aplicando la técnica de adaptación al dominio no supervisada. Para la percepción del entorno se utilizarán datos 360 grados que permiten ver de forma simultánea todo el entorno de la motocicleta. Se realizará un planteamiento general y se abordará una problemática concreta como es la detección de vehículos cercanos a la moto.

Índice general

Índice de figuras

Índice de Códigos

1 Introducción

El transporte desempeña uno de los papeles más fundamentales en nuestras vidas, debido a que no solo importa cómo llegamos a un lugar. El transporte tiene mucho más trasfondo (Mundial, 2023), es fundamental para respaldar el crecimiento económico, la creación de empleo y la conexión de las personas con los servicios esenciales, como la atención de salud o la educación.

Los vehículos autónomos han emergido como la vanguardia de la innovación, marcando una nueva era del transporte. Estos son medios de transporte que cuentan con sistemas informáticos instalados, que actúan como sistemas guía capaces de imitar las capacidades del ser humano a la hora de conducir. La autonomía de los vehículos está definida por cinco niveles distintos, según la sociedad de ingenieros automotrices (INTERNATIONAL, 2021), como podemos observar en la Figura 1.1. Los niveles van desde el nivel 0, sin automoción, hasta el nivel 5, con un nivel completo de automoción donde el humano no interviene. Destacar que del nivel 0 al nivel 2 el conductor tiene la responsabilidad del control, mientras que del nivel 3 al nivel 5 es el vehículo quien tiene el control.

Estos vehículos utilizan una variedad de tecnologías, incluyendo sensores, cámaras, radares, GPS y sistemas de aprendizaje automático, para detectar el entorno y tomar decisiones en tiempo real. La tecnología ha avanzado significativamente en las últimas décadas, pero todavía hay muchos desafíos técnicos y sociales que deben ser abordados para que la autonomía total se convierta en una realidad cotidiana. Los vehículos autónomos son capaces de percibir el entorno que los rodea y, en base a un análisis, aplicar la mejor decisión posible que sea pertinente.

La motivación para continuar con la investigación y el desarrollo de las tecnologías de los vehículos autónomos viene dado por múltiples desafíos, como son los siguientes:

- **La seguridad vial:** debido a que se espera que los vehículos autónomos puedan reducir significativamente el número de accidentes de tráfico, además de evitando colisiones, ayudando a reducir la fatiga y el estrés de los conductores. Esto es muy importante, ya que el factor humano es el causante del 90% de los accidentes de tráfico. En el año 2022 aumentó el número de víctimas mortales con respecto al año anterior en las carreteras españolas (DGT, 2022).

- **Ahorro económico:** si conseguimos reducir el número de accidentes, también estaremos ahorrando en seguros de vehículos, gastos de mantenimiento y reparaciones a causa de dichos percances.

- **Cambio climático:** los vehículos autónomos podrían ser más eficientes en términos de consumo de combustible y emisiones de gases de efecto invernadero. Este hecho ayudaría a reducir la contaminación del medio ambiente y aportaría en la lucha contra el cambio climático.

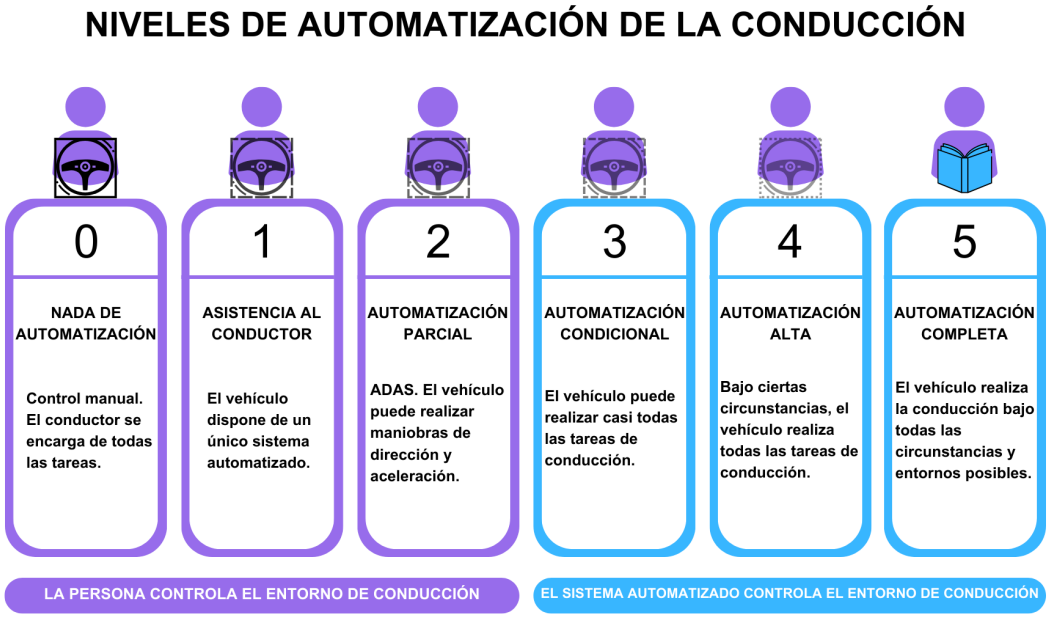


Figura 1.1: Niveles de autonomía en conducción autónoma

- **Desarrollo tecnológico:** la investigación y el desarrollo de la tecnología pueden conducir a avances en otras áreas tecnológicas, como nuevos sensores, desarrollo de IA y sistemas de control, etc.
- **Smart cities:** ciudades capaces de utilizar la tecnología de la información y comunicación, con el objetivo de crear mejores infraestructuras para una gran cantidad de ciudadanos. Según el estudio Cities in Motion (IESE, 2016), el 70% de la población mundial vivirá en las ciudades. Por lo tanto, estas necesitarán dar las mejores prestaciones y recursos posibles al alcance.

Centrándonos en el número de víctimas mortales en carretera debido a accidentes de tráfico provocados por el factor humano, es un hecho que en los últimos años ha habido un aumento significativo en la demanda social por los vehículos con asistencia a la conducción. Estos datos se vuelven aún más alarmantes cuando miramos el caso de las motos. En España se producen alrededor de 400 muertes al año, una cifra muy alta si la comparamos con la de los turismos (José Manuel Abad y Alameda, 2018).

Este es el principal motivo por el cual nació la motivación para la investigación y desarrollo de un sistema de ayuda a la conducción en las motos, puesto que podría ser muy útil para mejorar la seguridad a la hora de conducir y así poder contribuir la reducción de accidentes. La investigación se centra en un modelo de IA que aplica adaptación al dominio y es capaz de detectar los turismos desde cualquier ángulo, o lo que es lo mismo, utilizando imágenes 360º como entrada de datos de entrada.

1.1 Objetivos

El principal objetivo de Trabajo Final de Grado (TFG) es desarrollar un sistema de detección de coches mediante el uso de IA utilizando datos 360º para la ayuda a la conducción en motocicletas.

A lo largo de este proyecto, tenemos tanto objetivos académicos, como objetivos del proyecto.

Objetivos académicos:

- Familiarizarse con la metodología científica, artículos científicos y la terminología del área de estudio.
- Aprender sobre Python y Pytorch.
- Aprender y aplicar conocimientos sobre el aprendizaje profundo.
- Aprender y aplicar conocimientos sobre el ámbito de la adaptación al dominio.
- Mejorar la capacidad de documentación y comunicación.

Objetivos del proyecto:

- Realizar una revisión de los trabajos relacionados con sistemas de ayuda a la conducción en motocicletas.
- Proponer una solución a la detección de turismos: aparcados y en movimiento.
- Tratamiento y procesamiento de datos 360º con IA.
- Implementación de la solución.
- Evaluación del sistema.
- Documentación completa del proyecto realizado.

1.2 Estructura del proyecto

Para facilitar su lectura, el contenido de este documento se encuentra organizado en capítulos y secciones.

A continuación se describe la estructura que vamos a seguir.

- ***Capítulo 1: Introducción*** ⇒ Capítulo introductorio que abarca los fundamentos del proyecto y describe los objetivos que se pretenden alcanzar.
- ***Capítulo 2: Marco teórico*** ⇒ Dedicada a proporcionar una base teórica general del ámbito en el que se mueve el problema a resolver.
- ***Capítulo 3: Metodología*** ⇒ Explicación más detallada del método que se va a realizar para resolver el problema planteado, así como los objetivos a abordar.

- ***Capítulo 4: Desarrollo*** ⇒ Describe el proceso seguido para realizar el proyecto descrito y las herramientas y librerías utilizadas.
- ***Capítulo 5: Experimentos*** ⇒ Analiza los diferentes experimentos realizados y los resultados obtenidos por el modelo.
- ***Capítulo 6: Conclusiones y trabajo futuro*** ⇒ Detalla el conjunto de reflexiones obtenidas en cuanto al estudio de los resultados de las pruebas del proyecto, así como de posibles trabajos futuros a abordar.

2 Marco Teórico

Este capítulo trata de proporcionar una base teórica general del ámbito en el que se mueve el problema a resolver. El objetivo es poner en contexto los conceptos necesarios para entender el trabajo realizado, desde el funcionamiento de los sistemas de ayuda a la conducción, hasta conocimientos sobre IA.

2.1 Sistemas de ayuda a la conducción

Los sistemas de ayuda a la conducción permiten prevenir muchos de los desastres comentados anteriormente, como por ejemplo evitar colisiones, reducir la fatiga y el estrés, también ofrecen una asistencia con una variedad de tecnologías que ayuda al conductor y pueden contar con elementos de seguridad activa (ayudan a prevenir accidentes o a reducir su gravedad) o seguridad pasiva (ayudan a minimizar los daños sobre los ocupantes en caso de accidente). Además, pueden mejorar la eficiencia de la conducción y reducir el estrés del conductor.

Otro factor que ha contribuido a la popularidad de los sistemas de asistencia a la conducción es el rápido avance de la tecnología. Los sistemas están cada vez más avanzados, haciendo que sean más efectivos y accesibles para los consumidores. Se han vuelto muy comunes en los vehículos modernos e incluso ha causado un cambio de preferencia en los consumidores. Dispuestos incluso a pagar más por sistemas de seguridad que ayuden a prevenir accidentes y proteger a sus seres queridos y haciendo que se valore más la seguridad activa a la hora de decantarse en la elección de un vehículo. Este aumento en la demanda es debido a una mayor concienciación sobre los riesgos de la conducción, puesto que, la sociedad es cada vez más consciente de sus peligros y de la importancia de la seguridad vial.

Entre los sistemas de ayuda a la conducción se plantean numerosos desafíos y aspectos que son un gran reto a resolver. Algunos de los más importantes son los siguientes:

1. **Sistemas de detección:** Uno de los mayores desafíos es el desarrollo de tener sistemas con tecnología de detección y percepción muy precisa del entorno del vehículo. Los vehículos autónomos deben ser capaces de detectar y reconocer objetos en su entorno, como peatones, ciclistas, otros vehículos, obstáculos en la carretera, reconocimiento de las señales y semáforos con una precisión extremadamente alta.

2. **Toma de decisiones:** La toma de decisiones debe ser rápida y precisa basada en la información obtenida por los sensores sobre el entorno, para así asegurar una conducción segura y eficiente.

3. **Ciberseguridad:** Otro gran aspecto a tener en cuenta es debido a que este tipo de vehículos están conectados a la nube y entre sí. Por eso es crucial protegerlos y asegurar que no sean vulnerables a ataques maliciosos que intenten acceder a toda esa información.

4. **Regulación y normativas:** Las regulaciones y leyes varían de un país a otro y eso puede afectar a los vehículos. Además, existe la preocupación sobre quien sería el responsable en caso de accidentes que involucren este tipo de transporte autónomo.

5. **Ética:** Hay que tener en cuenta también que los vehículos deben ser capaces de tomar decisiones en situaciones difíciles, como por ejemplo en situaciones de emergencia donde se requiere la elección entre salvar la vida del conductor o de los peatones. Por lo tanto, es importante también abordar el aspecto ético y moral de la conducción autónoma.

2.1.1 Sistemas avanzados de ayuda a la conducción

Actualmente es muy común encontrarnos diferentes tipos de ADAS. Dichos sistemas funcionan combinando grandes cantidades de datos y la ayuda de un software, permitiendo que la tecnología sea capaz de resolver ciertas situaciones más rápidamente que un ser humano. Analiza la información, reconoce el entorno y determina cómo reaccionar ante él. Algunos de los ADAS más importantes que presentan los vehículos en el mercado en la actualidad son:

- **Control de Crucero Adaptativo (ACC):** El ACC es un sistema de control que ajusta automáticamente la velocidad para mantener una distancia de seguridad de los vehículos de enfrente. Está compuesto por diferentes tipos de sistemas como son: el sistema de frenado de emergencia que advirte al conductor o da apoyo al frenado si hay riesgo alto de colisión; los sistemas multisensoriales que se encargan de integrar la información para mejorar la seguridad del vehículo; los sistemas predictivos que son los encargados de modificar la velocidad en base a predicciones sobre el comportamiento del resto de vehículos del entorno realizando ajustes tempranos y moderados con respecto al comportamiento que se prevé.

- **Alerta Aviso de Colisión (FCW):** La FCW advierte al conductor mediante un pitido o mensaje cuando otro vehículo, peatón u obstáculo aparece en nuestro camino.

- **Alerta de Cambio de Carril (LDW):** La LDW indica con un indicador sonoro, luminoso o con vibración, que nos estamos saliendo del carril sin haber activado el intermitente.

- **Asistente de Visión Nocturna (NVA):** El NVA se activa en la oscuridad y muestra todo lo que hay en la vía, si se cruza una persona o un animal, el sistema lo mostrará y advertirá para esquivarlo.

- **Sistema de Mantenimiento de Carril (LKAS):** El LKAS detecta si el vehículo se desvía del carril de forma no intencionada y alerta al conductor mediante señales visuales y sonoras. Si no se corrige la trayectoria, el sistema actúa sobre la dirección con un ligero movimiento de volante. No funciona cuando están activos los intermitentes o cuando se vence la fuerza ejercida por el mando automático ya que interpreta que se está realizando el cambio de forma voluntaria.

- **Sistema de Detección de Fatiga (DDD):** El DDD avisa al conductor si pierde la concentración al volante o si ha superado el tiempo de conducción sin detenerse.

- **Alerta de Objetos en el Punto Ciego (BSD):** La BSD detecta los vehículos que se acercan por el ángulo muerto detrás (o al lado) e informa al conductor mediante luces, sonidos o vibraciones.
- **Lector de Señales de Tráfico (TSR):** El TSR es capaz de reconocer las señales de tráfico durante la circulación. Generalmente solo reconocen las señales de límite de velocidad, aunque algunos ya detectan otras restricciones.
- **Frenado de Emergencia Autónomo (AEB):** El AEB ayuda a evitar o mitigar los efectos de la colisión detectando vehículos en movimiento (o detenidos) y advirtiendo al conductor con anticipación, también es capaz de frenar hasta determinada velocidad.
- **Alerta por Paso de un Peatón (PCW):** La PCW hace que el conductor tenga muy presente la presencia de un peatón en la trayectoria del vehículo, tratando de evitar así los atropellos.

Como podemos ver en la Figura 2.1 la primera aparición de ADAS en la historia fue a partir de 1990, más concretamente fueron el ACC y la FCW, ambas implementadas solamente en coches. Un par de años más tarde se implementó la LDW en 1992. Al principio, fueron desarrollando sistemas más simples, que consistían en alertas o avisos al conductor. Sin embargo, años más tarde, con el uso de cámaras, aparecieron la NVA por el año 2000 y el LKAS en 2003. A partir del 2007, ya se desarrollaron algunos sistemas más, como pueden ser el DDD o la BSD. En el 2009 apareció el TSR y el AEB. Además, a principios de 2010 se desarrolló la PCW. No es hasta el 2021, que han aparecido las primera ADAS que podemos encontrar en las motos, y al igual que en los coches, una de ellas es el ACC y la otra es la BSD, como se muestra en verde en la Figura 2.1.

Figura 2.1: Evolución histórica de los ADAS. En rojo los relativos a coches y en verde a motos.

Es evidente la gran diferencia entre la evolución de las ADAS de coches y motos, como se ha visto en la Figura 2.1. Esta distinción no es debida a la falta de tecnología en los últimos años para adaptar los dispositivos como cámaras y sensores a las motos, más bien, es una consecuencia a la complejidad de la intervención de un tercero en la velocidad y dirección de un vehículo donde no existe sujeción ninguna, además, hay que tener en cuenta que la carrocería está compuesta por los pasajeros. Otros motivos pueden ser la falta de componentes electrónicos que las integran, en comparación a los coches. También hay que tener en cuenta que una moto no siempre circula de forma perpendicular a la carretera y no siempre obtendremos la información desde la misma posición. Por último, recordar que la demanda no es tanta como la que existe en el mundo del automóvil, lo que complica aún más

la investigación y el desarrollo de estos sistemas a falta del movimiento de grandes cantidades de capital.

2.1.2 Sensores

Un vehículo no podría obtener la cualidad de autónomo, ni podría utilizar estos sistemas de ayuda a la conducción, sin una manera de percibir el entorno. Para ello necesita servirse de sensores como los que podemos apreciar en la Figura 2.2, que lo doten de una detallada reproducción cartográfica del terreno para que el avance en la navegación sea normal. Este hardware está compuesto principalmente por:

- **Cámara:** Pueden llevar un tipo de cámaras de poca amplitud y buena calidad de imagen para la conducción, y otro tipo con menor calidad y una amplitud de hasta 120º para maniobras como aparcar, en las que necesitan de menos profundidad y más amplitud de visión. Se pueden situar delante, detrás y a los laterales.
- **Radar:** Complementan a las cámaras cuando hay poca visión como por ejemplo por la noche o en túneles. Funcionan transmitiendo ondas electromagnéticas por pulsos, cuando impactan en un objeto, vuelven al sensor, proporcionando la velocidad y el lugar del objeto. Radares y cámaras son suficientes para los niveles más bajos de autonomía, pero no cubren todas las situaciones sin un conductor.
- **Ultrasónico:** Utilizan ondas sonoras de alta frecuencia para detectar objetos. Estos sensores emiten pulsos de sonido (en una frecuencia indetectable para los oídos humanos) que se reflejan en los objetos cercanos. Un receptor detecta las ondas reflejadas y calcula la distancia desde su vehículo al objeto, utilizando el efecto doppler. Su uso es muy común a la hora de realizar estacionamientos, ya que estos sensores informan al conductor mediante una señal (un pitido o una imagen) de la proximidad de un obstáculo, normalmente, en la parte trasera del vehículo.
- **Sistema Global de Navegación por Satélite (GNSS):** Es una constelación de satélites que transmite rangos de señales utilizados para el posicionamiento y localización en cualquier parte del planeta. Estos permiten determinar las coordenadas geográficas y la altitud de un punto dado como resultado de la recepción de señales provenientes de satélites para fines de navegación.
- **Laser Imaging Detection and Ranging (LIDAR):** Dispositivo que determina la distancia de un emisor láser a un objeto o superficie, utilizando un haz de láser pulsado. Funciona igual con condiciones altas o bajas de luminosidad. Gracias a combinar el uso de los sensores anteriores con el LIDAR, el sistema de ACC puede ajustar la velocidad del vehículo de manera mucho más precisa en función a las condiciones del entorno.

2.1.3 Recopilación del medio

Para poder tener una conducción fluida a la hora de la circulación, necesitamos un sistema capaz de adaptarse de manera inteligente en un entorno cambiante, como escenarios de tráfico complicados, marcas de carriles o el tráfico que pasa por la carretera. La conducción autónoma

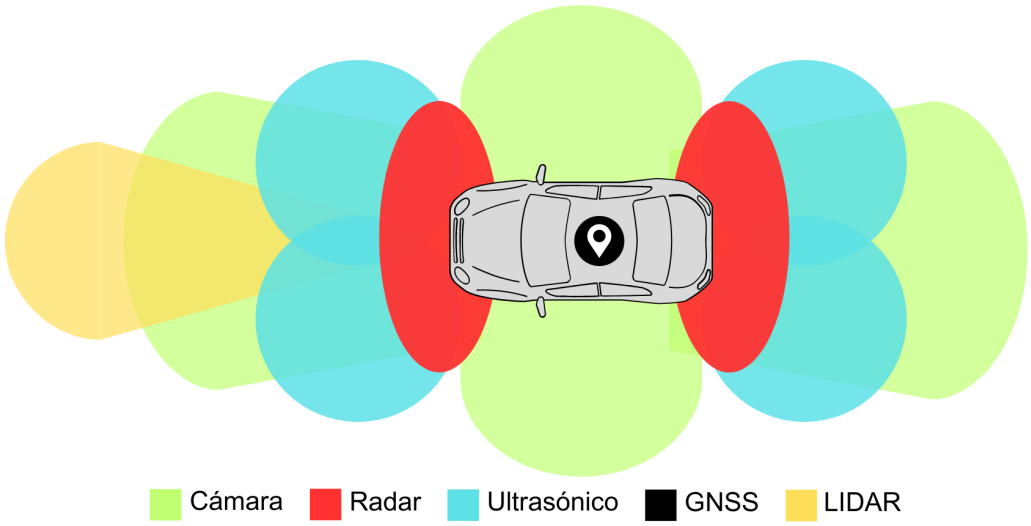


Figura 2.2: Sensores.

necesita saber de otros vehículos y pasajeros para poder determinar el contexto en la carretera y decidir cómo comportarse en una situación determinada.

Necesitamos primero recopilar los datos del entorno con sensores instalados en los vehículos, Sección 2.1.2. Cada vehículo tiene unas características distintas como la forma o el tamaño y sus sensores se colocan en diferentes lugares. La percepción de los datos que reciben también se recoge de manera diferente, pero con la ayuda de algoritmos de ML, estos datos se procesan para extraer características precisas con las que interpretar el entorno. Estos datos pueden ser desde carriles y barreras hasta señales de tráfico y marcas en el pavimento que brindan información importante para el funcionamiento de los sistemas de los vehículos.

2.1.4 Criterio toma de decisiones

Con los datos ya obtenidos y procesados, la salida de la toma de decisiones puede clasificarse en dos tipos: comandos de control de bajo nivel como la aceleración o la velocidad, y conductas de alto nivel como cambio de carril, adelantamiento, mantenimiento de carril.

El propósito de un sistema de toma de decisiones es generar el conjunto de mejores decisiones que al implementarse aporten seguridad a los seres humanos y una estrategia de conducción muy fiable. Es necesario formular una serie de criterios de diseño para lograr una mejor decisión, como son: buena actuación en tiempo real; un balance entre conducción segura y eficiencia (siendo la seguridad más importante); generar una decisión correcta y razonable; comodidad en la conducción (estabilidad en la conducción, pocos movimientos bruscos); alta capacidad de detección de fallos.

A la hora de elegir como se realiza la toma de decisiones hay que tener en cuenta muchos factores para lograr un sistema más completo, como pueden ser:

- Información del entorno cincundante.
- Información de las normas de tráfico locales.
- Resultados de la planificación de la ruta.
- Historial de resultados de la toma de decisiones.
- Ética de la conducción.

Tras tener en cuenta esto, el siguiente paso es saber en que escenarios aplicar qué tipo de decisiones.

2.1.5 Toma de decisiones

A la hora de realizar la toma de decisiones con los datos ya listos, podemos utilizar métodos clásicos y métodos basados en ML.

Los métodos clásicos se pueden subdividir en tres categorías: métodos basados en reglas, métodos de optimización y métodos probabilísticos. Los métodos que están basados en reglas dependen de una base de datos construida de acuerdo con numerosas leyes de tránsito, experiencia de conducción y conocimiento de conducción y las estrategias tienen en consideración los diferentes estados de los vehículos. Los métodos que están basados en optimización generalmente se basan en una recompensa o una función de utilidad para generar resultados de decisión. Los métodos probabilísticos generan resultados de comportamiento basados en la teoría de la probabilidad en matemáticas. Un modelo probabilístico debe establecerse para la determinación de comportamientos.

Los métodos basados en el ML, permiten a los vehículos dominar las capacidades de toma de decisiones similares a las humanas, a través de una gran cantidad de datos de entrenamiento, como SVM, AdaBoost, etc. También pueden utilizar modelos para aprender características de los datos y generar resultados de clasificación o regresión. Actualmente los regresores son más comunes a la hora de las tomas de decisiones. El objetivo de estos es aprender estrategias para maximizar los rendimientos probando varios comportamientos. Los comportamientos de los agentes se pueden ajustar de acuerdo con funciones de recompensa.

2.2 Inteligencia Artificial

En esta sección vamos a dar un punto de vista general sobre la IA y el funcionamiento de esta para saber, más adelante, como aplicarla en sistemas de ayuda a la conducción. En la Figura 2.3, se puede apreciar el diagrama representativo de la comparación entre IA, ML y DL.

La IA es la capacidad de las máquinas para dar soluciones a problemas según una serie de datos, aprender de los datos y utilizar lo aprendido en la toma de decisiones tal y como lo haría un ser humano (Rouhiainen, 2018).

Existen unos principios básicos que se deben de cumplir para una administración responsable de la IA (García, 2019). Deben de: beneficiar a las personas y al planeta, impulsando el crecimiento inclusivo, sostenible y el bienestar; diseñarse respetando el estado de derecho, los derechos humanos, los valores democráticos y la diversidad; existir una transparencia y

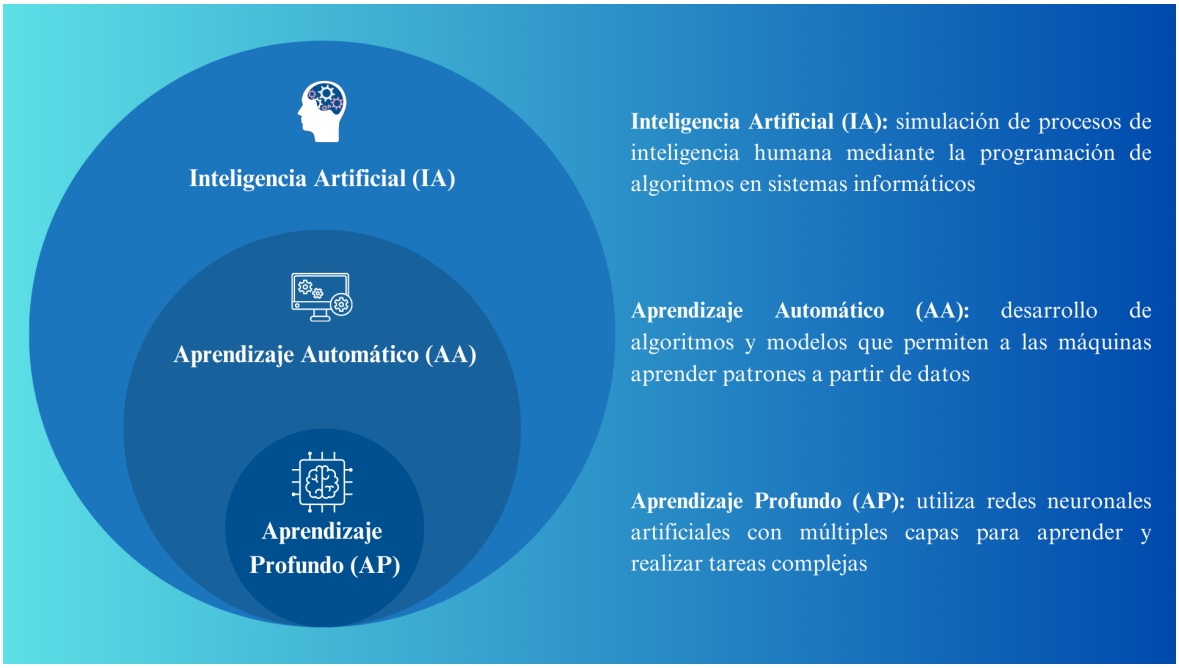


Figura 2.3: Diagrama de IA, ML y DL.

divulgación responsable; funcionar de manera sólida y segura y los riesgos potenciales deben evaluarse y gestionarse continuamente.

Dentro de la IA existen dos tipos, la IA Débil y la IA Fuerte. La IA Débil, también conocida como IA Estrecha, son sistemas diseñados para realizar tareas específicas y limitadas, como pueden ser el reconocimiento de voz, la identificación de imágenes o la traducción de idiomas. Actualmente la mayoría de ADAS se componen de varias IA Débiles. Por otro lado, la IA Fuerte, también conocida como IA general, tiene como objetivo crear máquinas inteligentes similares a la mente humana a partir de información y experiencias, progresando constantemente y mejorando sus habilidades a lo largo del tiempo.

2.2.1 Aprendizaje automático

El ML o *Machine Learning* es una rama de la IA (ver Figura 2.3), que se centra en el desarrollo de algoritmos y modelos que permite a las máquinas, a diferencia de la programación convencional, a aprender patrones a partir de los datos (ver Figura 2.4). Además, el ML utiliza técnicas estadísticas y computacionales para mejorar su rendimiento en una tarea a medida que se expone a una cantidad mayor de datos.

El ML está compuesto por diferentes tipos de modelos de aprendizaje y usa distintas técnicas algorítmicas (Buskirk y cols., 2018)(Qiu y cols., 2016). Pero como se puede observar en la Figura 2.5, todos comparten unos pasos comunes que siguen para obtener los resultados. Para comenzar se plantea un problema en el cual se precisa de ML para resolverse. Se realiza un estudio sobre como solucionarlo y se obtienen unos datos. Dichos datos son los que se utilizarán como entrada al modelo, con ellos se comienza a entrenar al modelo. Tras el entrenamiento, para comprobar la exactitud o rendimiento de este, se pone a prueba con

Figura 2.4: Programación tradicional vs. ML.

el conjunto de datos de test, son nuevos datos que el modelo no ha visto con anterioridad, y se analizan los resultados obtenidos. Para terminar, se realiza un análisis de dichos datos y a raíz de las conclusiones obtenidas, si no se satisface el problema propuesto, se vuelve a entrenar el modelo realizando cambios en cualquiera de las fases anteriores o, en cambio, sí se obtienen resultados satisfactorios y se pone en funcionamiento dicho modelo.

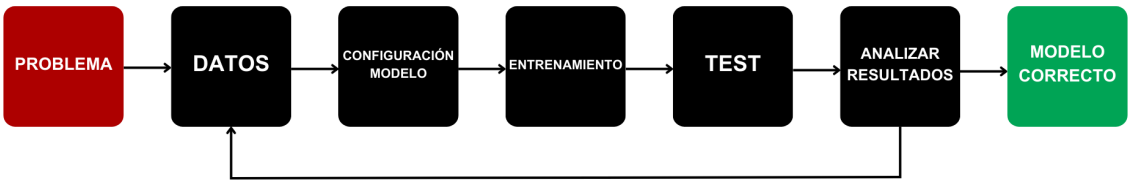


Figura 2.5: Pasos funcionamiento de un modelo de ML.

Dependiendo de la naturaleza de los datos y el resultado deseado, se puede utilizar cualquiera de los cuatro tipos de modelos de aprendizaje: supervisado, semi-supervisado, no supervisado y por refuerzo (Morales y Escalante, 2022). Dentro de cada modelo, se pueden aplicar distintos algoritmos.

- **Aprendizaje supervisado:** En el aprendizaje supervisado el modelo aprende con el ejemplo. Se le proporciona al algoritmo de ML un conjunto de datos conocido que incluye las entradas y salidas deseadas y el algoritmo debe encontrar un conjunto de parámetros para determinar cómo llegar de esas entradas a las salidas. Mientras el observador conoce las respuestas correctas al problema, el algoritmo identifica patrones en los datos, aprende de las observaciones y hace predicciones. El algoritmo realiza unas predicciones que son evaluadas para realizar correcciones, este proceso continúa hasta que el algoritmo alcanza un alto nivel de exactitud/rendimiento.
- **Aprendizaje semi-supervisado:** El aprendizaje semi-supervisado es similar al super-

visado, pero utiliza datos etiquetados y no etiquetados. Consiste en utilizar la información de los datos no etiquetados para mejorar el modelo inducido a partir de los datos etiquetados. Los datos etiquetados son esencialmente información que tiene etiquetas significativas para que el algoritmo pueda entender los datos, mientras que los datos no etiquetados carecen de esa información. Mediante esta combinación, los algoritmos de ML pueden aprender a etiquetar datos no etiquetados.

- **Aprendizaje no supervisado:** En el aprendizaje no supervisado, el algoritmo estudia los datos para identificar patrones. No hay etiquetas ni ningún elemento que brinde más información que los datos. Debido a este motivo, la máquina determina las correlaciones y relaciones analizando la información de los datos proporcionados. Lo que intenta es organizar los datos de alguna manera para describir su estructura. Esto puede significar agrupar los datos en clusters o disponerlos de forma que aparezcan organizados.

- **Aprendizaje por refuerzo:** El aprendizaje por refuerzo se centra en procesos de aprendizaje regimentados, en los que se proporciona a un algoritmo de ML un conjunto de acciones, parámetros y valores finales. Al definir las reglas, el algoritmo de ML intenta explorar distintas opciones y posibilidades, supervisando y evaluando cada resultado para determinar cuál es el óptimo. El aprendizaje por refuerzo enseña a la máquina por el método de ensayo y error. Aprende de experiencias pasadas y empieza a adaptar su enfoque en respuesta a la situación para lograr el mejor resultado posible.

Los algoritmos de ML están diseñados básicamente para realizar clasificaciones, encontrar patrones, proyectar resultados y tomar decisiones fundamentadas. Los algoritmos pueden usarse uno o combinarse varios a la vez para lograr la mayor exactitud posible cuando se trata de datos complejos y más impredecibles.

2.2.2 Aprendizaje profundo

El DL o *Deep Learning* es una subcategoría del ML (ver Figura 2.3), que utiliza ANN con múltiples capas, para aprender y realizar tareas complejas. Estas redes imitan la estructura y el funcionamiento del cerebro humano, permitiendo el procesamiento de información en niveles jerárquicos, lo que facilita la extracción de características y una toma de decisiones más precisa y avanzada. Consisten en un grupo de unidades, llamadas neuronas artificiales (Figura 2.6), conectadas entre sí para transmitirse señales. La información viaja a través de la ANN para proporcionar valores de salida.

Las neuronas artificiales son una unidad de cálculo que intentan representar el comportamiento de una neurona del cerebro humano. El resultado del cálculo de dicha neurona consiste en realizar una suma ponderada de las entradas, multiplicando el peso de la conexión entre la entrada y la neurona, seguida de la aplicación de una función de activación, como se muestra en la Figura 2.6.

Las neuronas artificiales se organizan en vectores lineales llamados capas. Existen capas de entrada, capas de salida y capas ocultas. Puede haber ninguna o varias capas ocultas. El diseño de la topología de la red implica determinar el número de neuronas de cada capa, el número de capas de la red, la ruta de las conexiones de las capas y el recorrido de las conexiones entre los nodos (Zou y cols., 2009). Normalmente, esos factores se fijan inicialmente y se optimizan tras varios experimentos. En la figura 2.7 se puede ver un ejemplo de topología de una ANN.

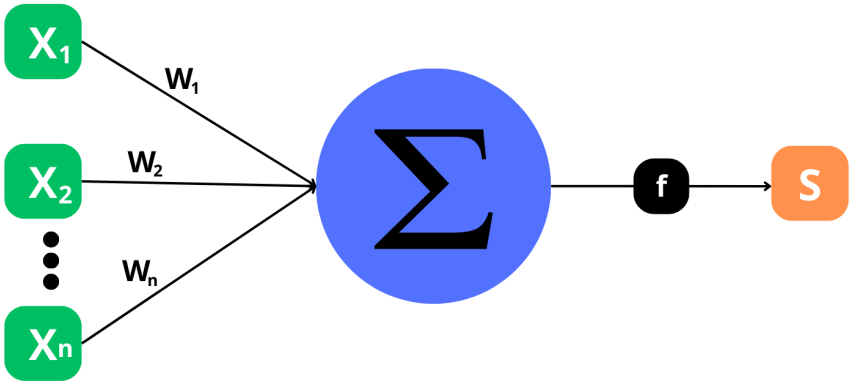


Figura 2.6: Neurona artificial.

Para realizar este aprendizaje, normalmente, se intenta minimizar una función de pérdida que evalúa la red en su totalidad. Una vez que se ha aplicado un patrón a la entrada de la red como estímulo, este se propaga desde la primera capa a través de las capas siguientes de la red, hasta generar una salida (*forward pass*).

Para realizar una evaluación de la predicción del resultado obtenido, la señal de salida se compara con la salida deseada y se calcula una señal de error para cada una de las salidas, a esto se le conoce como función de pérdida. Dichas salidas de error se propagan hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de la capa oculta que contribuyen directamente al resultado. Los valores de los pesos de las neuronas se van actualizando buscando reducir el valor de la función de pérdida. A esto se le conoce como retropropagación (o *backpropagation*) y permite que la información del costo fluya hacia atrás a través de la red para calcular el gradiente. Sin embargo, las neuronas de la capa oculta solo reciben una fracción de la señal total del error, basándose aproximadamente en la contribución relativa que haya aportado cada neurona a la salida original. Este proceso se repite capa por capa hasta que todas las neuronas de la red hayan recibido una señal de error que describa su contribución relativa al error total.

Para que la red aprenda correctamente se deben pasar numerosos ejemplos que abarquen el mayor espectro de casos posibles.

Este proceso es importante, debido a que, a medida que se entrena la red, las neuronas de las capas intermedias se organizan a sí mismas, de tal modo que, las distintas neuronas aprenden a reconocer distintas características de la entrada. Después del entrenamiento, cuando se les presente un patrón arbitrario de entrada que contenga ruido o que esté incompleto, las neuronas de la capa oculta de la red responderán con una salida activa si la nueva entrada contiene un patrón que se asemeje a aquella característica que las neuronas individuales hayan aprendido a reconocer durante su entrenamiento.

Inicialmente se utilizaban ANN más simples, como Máquina de vectores de soporte (SVM) o Perceptrón Multicapa (MLP) entre otros, pero según se han ido desarrollando, su topología también ha ido evolucionando. A continuación, se pueden apreciar dos de los tipos de redes más importantes en la actualidad para tratamiento de datos visuales.

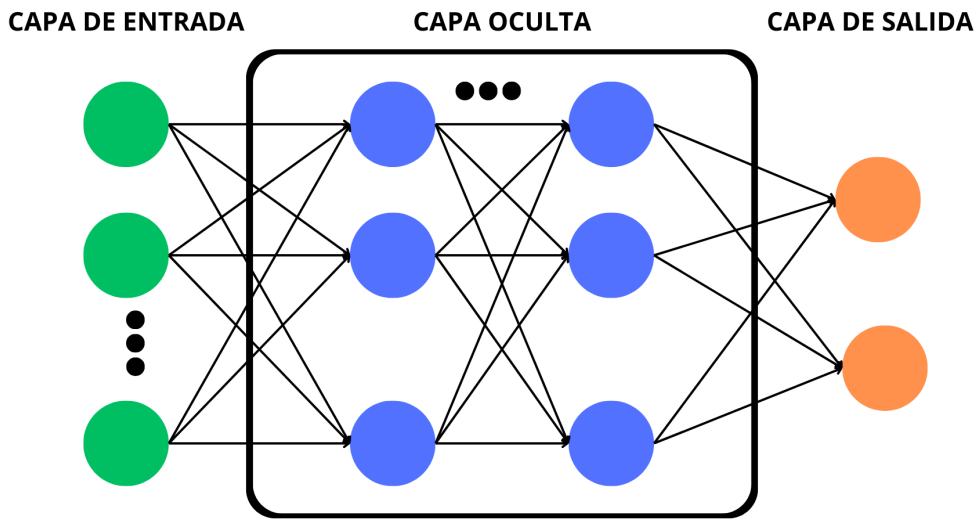


Figura 2.7: Topología de ANN.

- **CNN:** La CNN está especializada en procesar datos en forma de matriz, como las imágenes. Una imagen digital es una representación de datos visuales, compuesta por píxeles dispuestos en una rejilla con valores que indican el brillo y color de cada píxel. En la Figura 2.8 se puede observar un ejemplo de topología de una CNN. A continuación, se van a explicar las capas más importantes que componen dicha arquitectura.

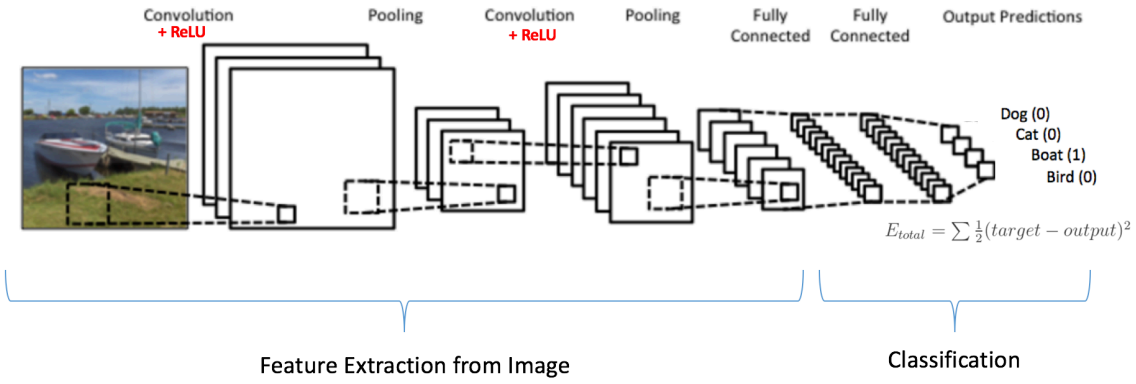


Figura 2.8: Ejemplo de topología de una CNN.

  - **Capa convolucional:** Realiza un producto escalar entre dos matrices, donde una matriz es el conjunto de parámetros aprendibles (kernel) y la otra es una porción restringida del campo receptivo. El kernel es más pequeño pero más profundo. Funciona desplazándose por la altura y anchura de la imagen, generando un mapa

de activación bidimensional. El objetivo es extraer características locales a una región de la imagen.

- **Capa de no linealidad:** Como la convolución es una operación lineal y las imágenes distan mucho de ser lineales, las capas de no linealidad suelen colocarse directamente después de la capa convolucional para introducir la no linealidad en el mapa de activación. Las operaciones no lineales más populares son sigmoide, tangente hiperbólica y la Unidad Lineal Rectificada (ReLU).
- **Capa de agrupación:** Sustituye la salida de la red en determinados lugares obteniendo una estadística resumida de las salidas cercanas. Además de reducir el tamaño espacial de la representación, disminuyendo la cantidad de cálculos y pesos, el proceso también aporta invarianza de translación, permitiendo reconocer objetos independientemente de su ubicación. Las más habituales so agrupamientos al máximo valor y al valor medio, *max pooling* y *average pooling* respectivamente.
- **Capa completamente conectada o densa:** Las neuronas de esta capa tienen conectividad total con todas las neuronas de la capa anterior y posterior. Por eso puede calcularse, como es habitual, mediante una multiplicación matricial seguida de un sesgo. Esto ayuda a mapear la representación entre la entrada y la salida.

- **RNN:** La RNN utiliza datos secuenciales o datos de series temporales, ya que toma información de entradas anteriores para influir en la entrada y salida actuales, como se puede ver en su topología en la Figura 2.9. Dicha salida depende de los elementos anteriores dentro de la secuencia. Además, comparten parámetros en cada capa de la red y el mismo parámetro de peso.

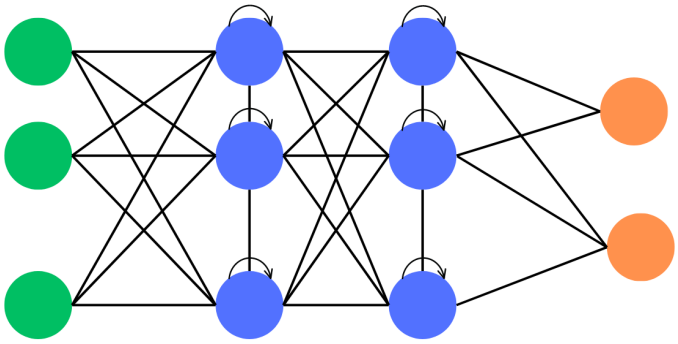


Figura 2.9: Ejemplo de topología de una RNN.

2.2.2.1 Problemas

Como en todos los campos, el de DL no es una excepción y también existen problemas o limitaciones. El más importante es la existencia de una dependencia muy fuerte hacia los datos. Esto es debido a que el modelo necesita una gran cantidad de datos para comprender

los patrones existentes. Un factor a tener en cuenta, es que la escala del modelo y el tamaño de la cantidad de datos necesaria tienen una relación casi lineal (Tan y cols., 2018). Para un problema concreto, el espacio expresivo del modelo debe ser lo suficientemente grande como para descubrir los patrones bajo los datos.

La recopilación de datos es compleja y costosa, por lo que resulta extremadamente difícil crear un conjunto de datos anotados a gran escala y de alta calidad. Además, incluso si se obtiene un conjunto de datos de entrenamiento, por costoso que sea, es muy fácil que este quede obsoleto, por lo que no puede aplicarse eficazmente a las nuevas tareas.

2.2.3 Aprendizaje por transferencia

La aparición de nuevos retos trae consigo la investigación y desarrollo para abordarlos y para el problema mencionado anteriormente sobre los datos, surgió el TL. El TL o *Transfer Learning* es la mejora del aprendizaje en una tarea nueva mediante la transferencia de conocimientos de una tarea relacionada que ya se ha aprendido (Olivas y cols., 2009).

Cuando entrenamos un modelo desde cero, tenemos un conjunto de datos del que extraemos características y tenemos una función que asigna valores sobre nuestro conjunto de etiquetas. Cuando se le da un conjunto de datos suficientemente grande, aprende a realizar una tarea específica. Sin embargo, cuando se le encarga resolver un nuevo problema, no puede recurrir a ningún conocimiento adquirido previamente. En su lugar, un algoritmo convencional necesita un segundo conjunto de datos para iniciar un nuevo proceso de aprendizaje. ES por ello que se recurre a la técnica de TL. Dicha técnica consiste en utilizar los pesos de una red entrenada para inicializar los pesos de la misma arquitectura de red que se emplea en otra tarea similar, ver Figura 2.10. Esto permite reducir el tiempo de entrenamiento y la cantidad de datos necesitados.

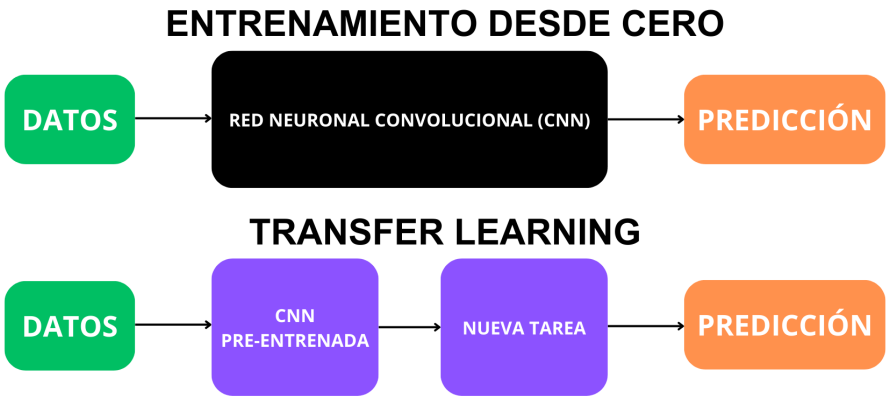


Figura 2.10: Comparación entre un modelo que entrena desde cero y uno que utiliza TL.

2.2.4 Adaptación al dominio

El TL es la idea básica de explotar la información relacionada para aprender, mientras que la Adaptación al Dominio (DA) es específica para casos en los que las tareas[1] fuente y objetivo son las mismas. Gracias a esto, un modelo entrenado en un conjunto de datos no necesita volver a entrenarse en un nuevo conjunto. En su lugar, el modelo preentrenado puede ajustarse para ofrecer un rendimiento óptimo con los nuevos datos.

La DA es una técnica que permite mejorar el rendimiento de un modelo en un dominio[2] objetivo, que no contiene suficientes datos etiquetados, utilizando los conocimientos aprendidos por el modelo en otro dominio relacionado con datos etiquetados adecuados. Por ejemplo, en el caso de este TFG se quieren reconocer vehículos con imágenes 360. Así que se plantea utilizar una red que detecte vehículos en imágenes y adaptar el dominio de entrada para que haga la misma tarea con nuestros datos 360. El mecanismo de adaptación de dominio consiste en descubrir los factores latentes comunes a los dominios fuente y objetivo y adaptarlos para reducir el desajuste marginal y condicional.

Al igual que se han visto los tipos del ML en la Sección 2.2.1, también se puede clasificar la DA en alguna de las tres categorías siguientes:

- **Adaptación al Dominio Supervisada (SDA):** En la SDA los datos del dominio objetivo están completamente anotados.
- **Adaptación al Dominio Semi-Supervisada (SSDA):** En la SSDA solo se etiquetan unas pocas muestras de datos en el dominio objetivo.
- **Adaptación al Dominio No Supervisada (UDA):** En la UDA el dominio objetivo no tiene etiquetas, salvo las del conjunto de test, que se utilizan para evaluar las predicciones del modelo en el conjunto de datos del objetivo.

Para un problema en el cual nombramos la dependencia hacia los datos, SDA no parece la solución ideal, debido al uso de dos conjuntos de datos completamente etiquetados, sin embargo, pueden funcionar de forma óptima sin la necesidad de utilizar una enorme cantidad de datos. Por otro lado, se puede pensar que SSDA puede llegar a tener mejor rendimiento debido al etiquetado de menos datos, pero aún así, se ha comprobado que se empeora el rendimiento sin realizar ciertas modificaciones en la red. Y por último, UDA aborda este desafío permitiendo que el modelo generalice de manera más efectiva a través de una vasta cantidad de datos, sin la necesidad del uso de etiquetas para el dominio objetivo, minimizando la brecha entre estos dominios.

[1] Una tarea es un trabajo que hay que hacer o emprender

[2] Un dominio son todos los valores que pueden (que tienen sentido dado el contexto) entrar en una función

3 Metodología

En este capítulo se va a explicar más detalladamente el método a evaluar como propuesta de solución al problema planteado durante este trabajo. Constará de una descripción de la metodología seguida para poner en contexto la implementación.

Para detectar coches mediante visión por computador con datos 360 necesitamos una cantidad enorme de datos etiquetados. Puesto que son datos propios no tenemos el *ground truth* y etiquetar todo (24 fps) es inviable, así que se utilizará la técnica de UDA. Para llevar a cabo esto se va a utilizar un modelo que la aplica.

3.1 Estructura del modelo

El modelo o *framework* que se utiliza como base para el desarrollo de la investigación es *Adaptive Teacher* (Li y cols., 2022). La estructura de este, como se puede ver en la Figura 3.1, consta de dos módulos: el modelo profesor del dominio objetivo y el modelo estudiante de comunicación entre dominios.

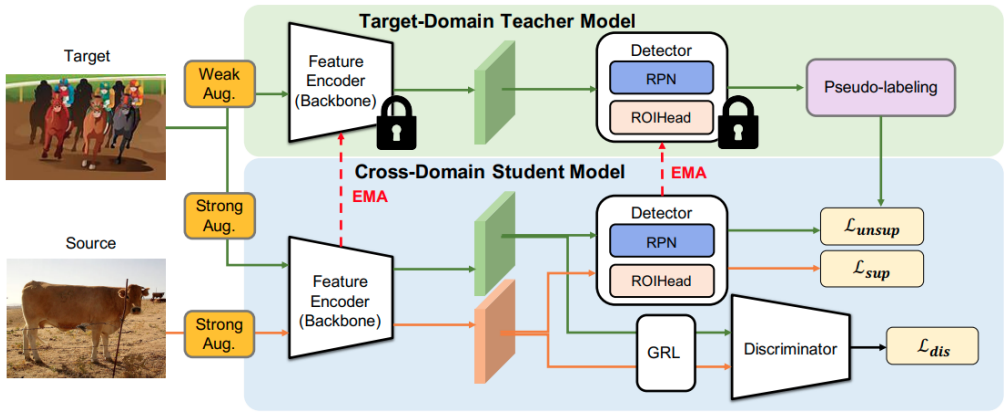


Figura 3.1: Framework Adaptive Teacher. Imagen extraída de Li y cols. (2022).

El modelo de profesor tiene en cuenta las imágenes del dominio objetivo con aumentos débiles, como volteos y recortes horizontales aleatorios, mientras que el modelo del estudiante tiene en cuenta las imágenes con aumentos fuertes, como añadir aleatoriamente fluctuaciones de color, escala de grises y parches de recorte, en ambos dominios (Liu y cols., 2021). Para entrenar el modelo utilizamos dos estrategias de aprendizaje, las cuales son el aprendizaje mutuo profesor-estudiante y el aprendizaje adversarial.

Por un lado, el aprendizaje mutuo consiste en que varios algoritmos aprendan de diferentes fuentes y compartan su conocimiento entre ellos, para así que todos los participantes puedan mejorar sus resultados de clasificación y su exactitud de predicción. Por el otro lado, el

aprendizaje adversarial es un método cuyo objetivo es la generación y detección de entradas engañosas, su objetivo es engañar a los modelos y la detección de esto.

Para comenzar, se entrena el detector de objetos con los datos etiquetados de origen, inicializándose junto al codificador de características.

En la fase de aprendizaje mutuo, se duplica el detector de objetos inicializado en dos detectores idénticos, uno para el modelo profesor y el otro para el modelo estudiante. Durante el entrenamiento, mientras que el estudiante actualiza los conocimientos aprendidos para el profesor a través de la Media Móvil Exponencial (EMA), el modelo profesor genera pseudo-etiquetas para entrenar al estudiante. De manera iterativa, van mejorando las pseudo-etiquetas para el entrenamiento del estudiante. Además, se aplica el aprendizaje adversarial utilizando el discriminador y la Capa Inversa del Gradiente (GRL) para el aprendizaje adaptativo con el fin de alinear las distribuciones entre ambos dominios en el modelo del estudiante.

En dicho modelo, la pérdida supervidasa (Lsup) y la pérdida no supervisada (Lunsup) sirven para el aprendizaje del codificador y del detector, mientras que, la pérdida adversarial (Ldis) está hecha para actualizar el codificador de características y el discriminador.

3.2 Montaje del modelo

Debido a la naturaleza de esta investigación, la cual esta basada en la detección de coches, los elementos más importantes de la arquitectura a tener en cuenta a la hora de la configuración son el detector y el codificador.

3.2.1 Detector

Se ha de tener en cuenta, como se ha visto en la Figura 3.1, este modelo no tiene una topología simple. Como se ha comentado en el apartado anterior, Sección 3.1, los dos modelos utilizan un detector duplicado.

Este detector no funciona como un clasificador mostrado en la Sección 2.2.2 con una topología básica de CNN debido a su composición por una Red Regional Propuesta (RPN) (Ren y cols., 2015) y por una Región De Interés (ROI) (He y cols., 2017). Por lo tanto, podemos decir que el tipo de modelo utilizado es una Faster R-CNN. A continuación se dará una visión sobre este tipo de redes.

3.2.1.1 Algoritmos R-CNN

Para dar contexto sobre este tipo de modelos, primero hay que explicar que es una R-CNN.

R-CNN (Liang y Hu, 2015) es un tipo de modelo de DL utilizado en visión por computador que está especialmente diseñado para la detección de objetos. Su objetivo original y principal es definir los bordes de un objeto dentro de imágenes de entrada del modelo.

Como se aprecia en la Figura 3.2, primero se proponen regiones generadas por una función, como pueden ser *bounding boxes*. Estas regiones son extraídas de la imagen y redimensionadas para ser clasificadas por la CNN. Por último, los *bounding boxes* son refinados por una SVM que es entrenada usando las características de la CNN.

Fast R-CNN (Girshick, 2015) es más eficiente que su antecesora y tiene una topología distinta, presentando una serie de diferencias (ver Figura 3.3). Este modelo también comienza por una proposición de regiones generadas por una función, pero, a diferencia del anterior,

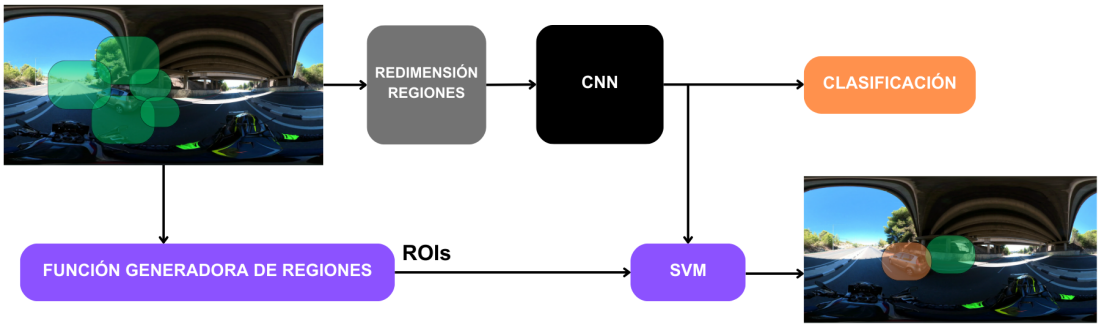


Figura 3.2: Topología de una R-CNN.

Fast R-CNN procesa la imagen entera. Además, en vez de clasificar cada una, agrupa las características de la CNN correspondiente a cada una de las regiones. Y, por último, los cuadros delimitadores son refinados por capas de regresión.

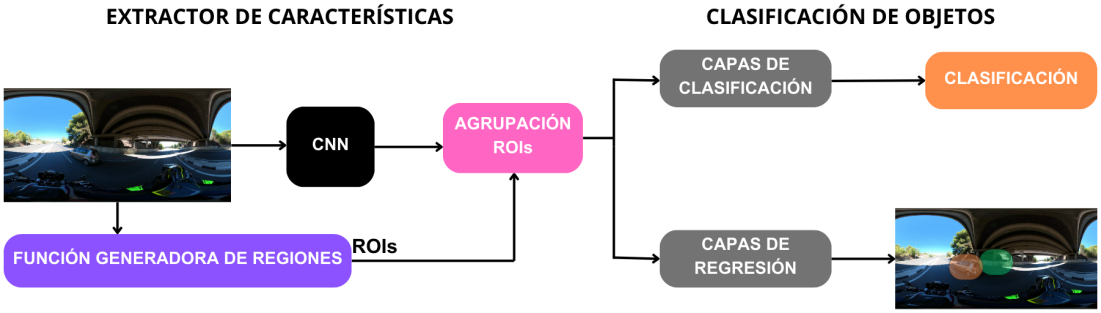


Figura 3.3: Topología de una Fast R-CNN.

Faster R-CNN (Ren y cols., 2015) es el último tipo que compone esta familia de algoritmos. Como su nombre indica, es la más eficiente de las tres. Como se puede ver en la Figura 3.4 y, a diferencia de las dos anteriores, Faster R-CNN añade una RPN para generar las regiones propuestas directamente en la red, en lugar de utilizar otro algoritmo.

3.2.2 Codificador de características

En este caso, se utiliza una CNN como extractor de características, conocida también como la columna vertebral (o *backbone*) de nuestro modelo, para la detección de objetos.

Debido a su correcto funcionamiento en tareas sencillas de manera independiente, estas redes suelen ser utilizadas para la extracción de características en los modelos más complicados.

Existen muchas arquitecturas populares de CNN que se pueden utilizar para este tipo de tareas. Pero la que se ha escogido para este TFG sobre la detección de vehículos en imágenes

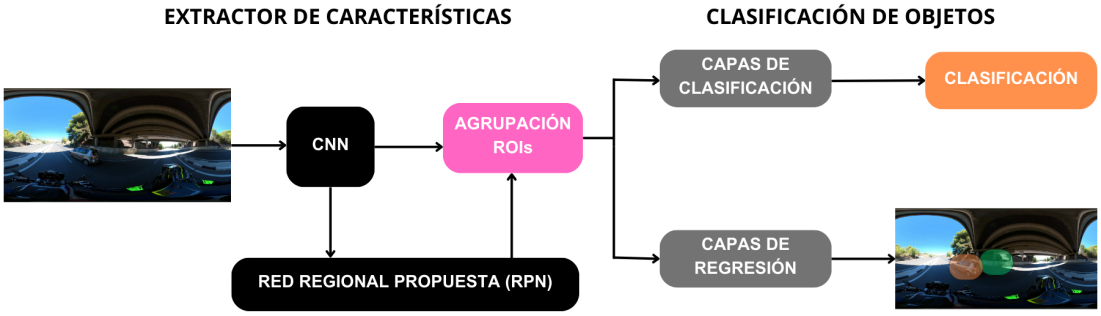


Figura 3.4: Topología de una Faster R-CNN.

360º es la VGG-16.

3.2.2.1 Modelo VGG-16

Como se acaba de comentar, VGG-16 (Simonyan y Zisserman, 2014) no solo es un tipo de CNN, sino que, además, está considerado como uno de los mejores modelos de visión por computador. Es capaz de clasificar imágenes pertenecientes a mil tipos diferentes de clases obteniendo muy buenos resultados.

Recibe este nombre debido a que su composición se basa en dieciséis capas con pesos. Más concretamente, está formada por trece capas convolucionales, cinco capas de agrupación (o *max pooling*) y tres capas completamente conectadas. Lo cual tiene como resultado veintiuna capas, pero solo dieciséis son capas con pesos. Dicha arquitectura del modelo se puede ver representada en la Figura 3.5.

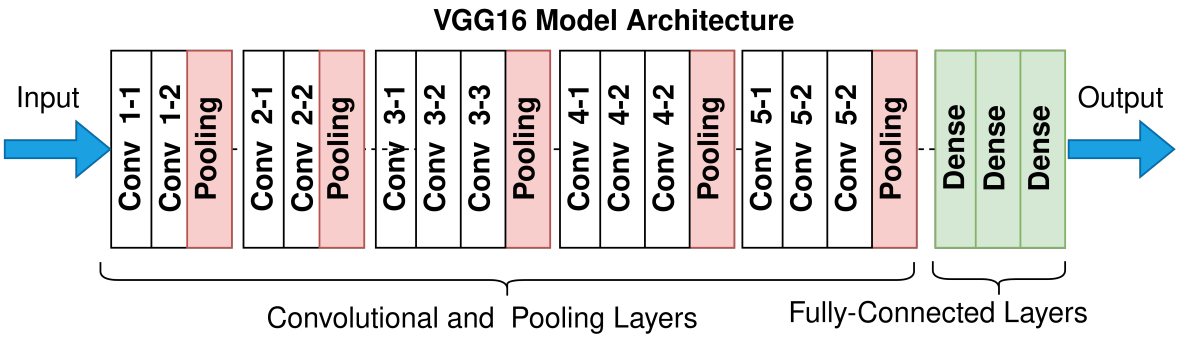


Figura 3.5: Arquitectura del modelo VGG-16.

4 Desarrollo

En este capítulo se va a explicar y desarrollar el proyecto realizado, desde el uso de las diferentes herramientas utilizadas, hasta el proceso de realizar dicho trabajo, partiendo de la recolección de datos, hasta la puesta en marcha del modelo.

4.1 Herramientas

Antes de explicar el desarrollo seguido para la realización del programa, se va a realizar una contextualización sobre las diferentes herramientas utilizadas y el uso que han tenido.

4.1.1 Lenguaje de programación

El lenguaje de programación con el que se ha realizado el proyecto y sobre el cual se ha creado el entorno virtual para el control de las librerías es Python, más concretamente, su versión Python 3.8, debido a su compatibilidad con las librerías empleadas por el modelo *Adaptive Teacher*. Además, gracias a su simplicidad y a su legibilidad, la sintaxis de Python es concisa y fácil de entender, haciéndolo un lenguaje de programación ideal para el desarrollo de la IA.

4.1.2 Generación de datos

Tanto para el proceso de recolección y obtención de datos brutos, hasta para la tarea de convertirlos en los datos que queremos utilizar para nuestro modelo, han sido varias herramientas de las que hemos tenido que disponer.

4.1.2.1 Cámara 360

Para la obtención de datos brutos del medio, se ha utilizado una cámara 360. Más concretamente se ha predispuesto del uso de una cámara modelo INSTA360 ONE X2. Dicho modelo cuenta con las siguientes especificaciones:

- Apertura: F2.0
- Equivalente a focal de 35mm: 7.2mm
- ISO: 100-3200
- Valor de exposición: 4EV
- Resolución de Vídeo:
- Formato de Vídeo: INSV

4.1.2.2 Edición de vídeo

Es necesario un preprocesamiento de los datos brutos, para poder obtener los datos que queremos pasarle a nuestro modelo. Este proceso ha sido realizado mediante una herramienta de edición de vídeo que procesa datos INSV. El programa seleccionado es Insta360 STUDIO 2023, el cual permite editar vídeos tomados con la cámara 360 y con el formato mencionado, ha utilizada para marcar los puntos clave del vídeo y realizar los cortes que queremos para quedarnos como resultado final los datos limpios.

4.1.2.3 FFmpeg

Para realizar la conversión de vídeo a imagen, se ha utilizado del uso de la librería **FFmpeg**. Este programa es capaz de procesar datos multimedia, tanto vídeo, como audio. Gracias al uso de dicha librería, se han obtenido las imágenes (o fotogramas) necesarias por parte de cada vídeo.

4.1.2.4 Labelme

El etiquetado de los datos es un apartado fundamental si estamos hablando de un proyecto en el ámbito en la IA. Para realizar dicho etiquetado mediante polígonos sobre la superficie de los coches en las distintas imágenes de nuestro conjunto de datos, se ha utilizado la librería labelme (Wada y cols., 2021). Labelme es una herramienta de código abierto, desarrollada en Python para ayudar a la anotación poligonal manual de imágenes para la detección de objetos, clasificación y segmentación.

4.1.2.5 Labelme2coco

La librería Labelme2coco (fcakyon, 2023) nos proporciona la capacidad de transformar las etiquetas de todo un directorio, creadas con Labelme, a un fichero en formato COCO[1]. Este formato es un estándar para los datos de entrenamiento y evaluación en tareas de detección de objetos.

4.1.3 Librerías del modelo

A la hora de utilizar el modelo propuesto para el desarrollo de la investigación, es necesario el uso de las diferentes librerías que lo componen para utilizarlo. Además, hay que tener en cuenta las versiones específicas y que sean compatibles entre sí para lograr compilar el proyecto sin ningún problema.

4.1.3.1 PyTorch

PyTorch es una librería de código abierto hecha en Python, basada a su vez en la biblioteca de Torch, enfocada a problemas de visión artificial y procesamiento de lenguajes naturales. Realiza cálculos numéricos mediante programación de tensores, lo que facilita su aplicación al desarrollo de aplicaciones de DL. Debido a su sencillez y la aceleración que proporciona a

[1]COCO es un formato para especificar conjuntos de datos de detección y segmentación de objetos a gran escala

través de su uso en la Unidad de Procesamiento Gráfico (GPU), la convierte en una de las mejores elecciones posibles a la hora de entrenar modelos. Más concretamente, se va a utilizar la versión de PyTorch 1.7.1 para este proyecto.

4.1.3.2 CUDA

Arquitectura Unificada de Dispositivos de Cómputo (CUDA) es una plataforma de cálculo paralelo y un modelo de programación creado por NVIDIA. CUDA ayuda a acelerar las aplicaciones aprovechando la potencia de los aceleradores de la GPU, junto a lecturas dispersas y una memoria compartida. Se va a utilizar la versión 11.0 para este proyecto.

4.1.3.3 OpenCV

Librería de Visión por Computador de Código Abierto (OpenCV) es una librería de visión por computador y software de ML de código abierto. OpenCV fue creada para proporcionar una infraestructura común para las aplicaciones sobre IA y cuenta con muchos algoritmos optimizados especializados en la detección y clasificación de objetos.

4.1.3.4 Detectron2

Detectron2 (Wu y cols., 2019) es una librería de visión por computador desarrollada por Facebook, la cual nos permite crear fácilmente modelos de detección de objetos, segmentación de instancias, detector de puntos clave y segmentación panóptica. También se puede destacar que está basada en PyTorch. Se va a utilizar la versión 0.3 del Detectron2 para este proyecto.

4.1.3.5 Adaptive Teacher

Adaptive Teacher es la librería del modelo que se va a utilizar para la resolución de la problemática planteada a lo largo de este trabajo. Su funcionamiento está explicado en la Sección 3.1. Dicha librería está creada mediante el uso de Detectron2, OpenCV, CUDA, PyTorch y Python.

4.2 Procedimiento del proyecto

Una vez contextualizadas las herramientas utilizadas en el trabajo y una visión más amplia sobre el tema, se va a comentar el procedimiento seguido para el desarrollo del proyecto. Dicho desarrollo consta de dos partes, la generación de los datos (referente a los datos) y la búsqueda, instalación y configuración de las distintas librerías con sus diferentes versiones (referente al modelo), en la Figura 4.1 se puede observar el diagrama del procedimiento seguido.

Debido a la variedad de procesos a seguir para realizar este programa, se ha necesitado el uso de distintos entornos, e incluso sistemas operativos, donde trabajar e ir avanzando con cada parte del proyecto. Debido a que para poder utilizar las mismas librerías pero para diferentes programas, se necesitan versiones distintas, puesto que cada programa tiene su propia configuración independiente.

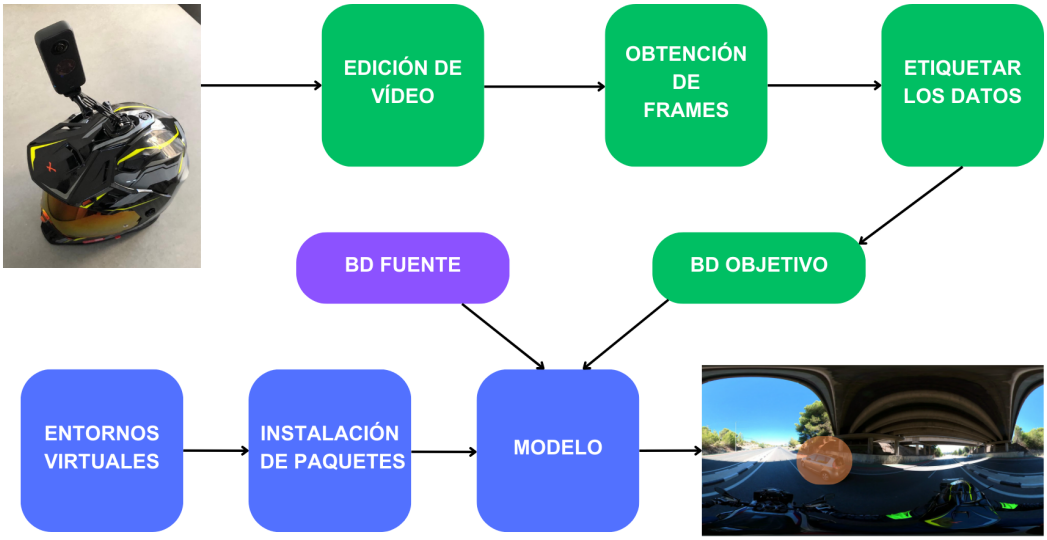


Figura 4.1: Diagrama del desarrollo del proyecto.

4.2.1 Generación de datos

Los datos son uno de los factores más importantes para el correcto funcionamiento de un modelo. Es por ello que se han llevado a cabo diferentes tareas para la obtención de estos. Para comenzar, y una vez configurada la cámara 360 (INSTA360 ONE X2), se ha acoplado a un soporte en el casco de una moto. Se ha puesto en funcionamiento por diferentes áreas de la ciudad de Benidorm, las cuales podemos clasificar en autovía, áreas interurbanas y áreas urbanas. En total se filmaron unos 50 minutos de vídeo.

Mediante el programa de edición de vídeo (Insta360 STUDIO 2023) y tras un par de cortes para limpiar información innecesaria, se extrajeron diferentes fragmentos de los cuales se puede sumar el tiempo de cada área, obteniendo así 4 minutos de autovía, 16 minutos de área interurbana y 30 minutos de área urbana.

Estas dos primeras partes del proceso están realizadas en el sistema operativo Windows 10, debido a que los programas usados, no tenían soporte para otro sistema. Debido a que la librería de Detectron2 solo funciona en ubuntu, el trabajo restante está realizado en el sistema operativo Ubuntu 20.04.

Una vez obtenidos los vídeos limpios y organizados se realiza la construcción de la base de datos que vamos a utilizar como dominio objetivo, a la cual nos referiremos como Moto360. Para la creación del conjunto de datos se utiliza la herramienta ffmpeg, con la que se obtienen las imágenes correspondientes de cada vídeo.

Ya con la base de datos creada y aunque se está utilizando un modelo de UDA, también es necesario etiquetar imágenes. Haciendo uso de la herramienta Labelme y mediante un etiquetado poligonal, se obtienen las anotaciones de cada uno de los diferentes fotogramas que componen el conjunto test. Además, para el correcto funcionamiento del modelo, también

se crean unas etiquetas para el conjunto de entrenamiento denominadas anotaciones ficticias o *dummy annotations*. Estas anotaciones no aportan ninguna información extra sobre la imagen, tan solo agrupan el nombre de cada imagen junto a su resolución y un id. A diferencia de un modelo de ML tradicional, no es necesario el etiquetado de ambos conjuntos, solamente se ha de etiquetar el conjunto de test.

Debido a que la librería en la que está basada el modelo es Detectron2, para la correcta interpretación de las anotaciones, la base de datos necesita un etiquetado estándar como por ejemplo COCO, que es el que se ha utilizado en este caso. Con la ayuda del programa Labelme2coco se ha obtenido un archivo en formato json con todas las etiquetas de las imágenes que componen el conjunto de test.

4.2.2 Librerías del modelo

Con la base de datos lista para el funcionamiento, queda realizar la búsqueda e instalación de las librerías y la preparación del modelo para su puesta en marcha, teniendo en cuenta la correcta interpretación de su estructura, los hiperparámetros y la correcta inyección de los datos.

Tras una búsqueda sobre librerías que aplicaran la problemática de UDA y un estudio sobre estas, se seleccionó el modelo *Adaptive Teacher*, el cual se ha ido nombrando a lo largo de este trabajo. El motivo de esta elección es debido a los resultados favorables que obtiene dicho modelo sobre una problemática parecida a la que se trata de resolver en este trabajo, siendo esta la *Comprensión de escenas con niebla con datos sintéticos* (Sakaridis y cols., 2018). La problemática que se plantea es la detección de distintos tipos de elementos en las carreteras bajo una capa de niebla, creada por ordenador, que simula situaciones de visibilidad reducida. El problema es abordado mediante el uso de la técnica de UDA donde el dominio fuente son unas imágenes normales de unas carreteras y el dominio objetivo está formado por las mismas imágenes pero con niebla.

En base a esta elección de modelo, se han determinado las diferentes librerías necesarias y las diferentes versiones de las mismas para un correcto funcionamiento. La base sobre la que se sostiene es el entorno sobre el que se va a ejecutar, el cual va a tener la versión de Python 3.8, como se ha comentado anteriormente. Es muy importante el control de las versiones de las librerías, debido a que Python es un lenguaje en constante avance que cada poco tiempo desarrolla y lanza una nueva versión.

Una vez ya con el entorno creado y para continuar con la instalación, es el turno de las librerías que se encargan de realizar los cálculos numéricos mediante el uso de la GPU de forma paralela, o lo que es lo mismo, realizar los entrenamientos sobre el modelo. Dichas librerías con sus respectivas versiones son PyTorch 1.7.1 y CUDA 11.0.

Para un correcto funcionamiento de la librería en la que se basa el modelo y no obtener problemas con las versiones, se instala OpenCV, seguido una instalación de la librería wheel. Estas dos son utilizadas para la visión por computador debido su especialización en algoritmos optimizados para la detección de objetos. Tras esto y para poder poner en funcionamiento el modelo, se instala la librería Detectron2 versión 0.3 compatible con nuestras versiones de PyTorch y de CUDA.

Por último, con las librerías instaladas y sin problemas de compatibilidad de versiones, se modifica el código para que el modelo reciba como entrada las dos bases de datos que se van a utilizar para la obtención de resultados.

5 Resultados

Este capítulo va a tratar los experimentos realizados con el modelo desarrollado, centrándose en una evaluación de los resultados obtenidos. También se van a analizar las bases de datos que sirven como datos de entrada al modelo, a especificar el entorno de pruebas donde se realiza el entrenamiento y la evaluación y a explicar los hiperparámetros utilizados.

5.1 Bases de datos

Los datos de entrada al modelo se agrupan en conjuntos de datos, también conocidos como bases de datos. Por lo general, a la hora de utilizar un modelo se utiliza una única base de datos, la cual se divide en un conjunto de entrenamiento y un conjunto de test para evaluar a la red, como está explicado en la Sección 2.2.1.

En este caso y debido al uso de la técnica de UDA, se utilizan dos bases de datos. El conjunto de datos que actúa como dominio fuente tiene las dos particiones etiquetadas y el conjunto de datos que actúa como dominio objetivo solo tiene la partición de test etiquetada.

5.1.1 CityScapes - Dominio Fuente

Para el dominio fuente se va a utilizar la base de datos *CityScapes* (Cordts y cols., 2016). Esta base de datos está compuesta por 5000 imágenes tomadas en 27 ciudades distintas. Además, las imágenes tienen una resolución de 2048x1024 píxeles.

Las anotaciones que contiene este conjunto de datos no se limita simplemente a los coches, puesto que está compuesto por 33 clases distintas, todos relacionados con elementos que te puedes encontrar en la vía pública durante la conducción. Sin embargo, en los experimentos nos limitaremos al uso exclusivo de las etiquetas de los turismos (id 26).

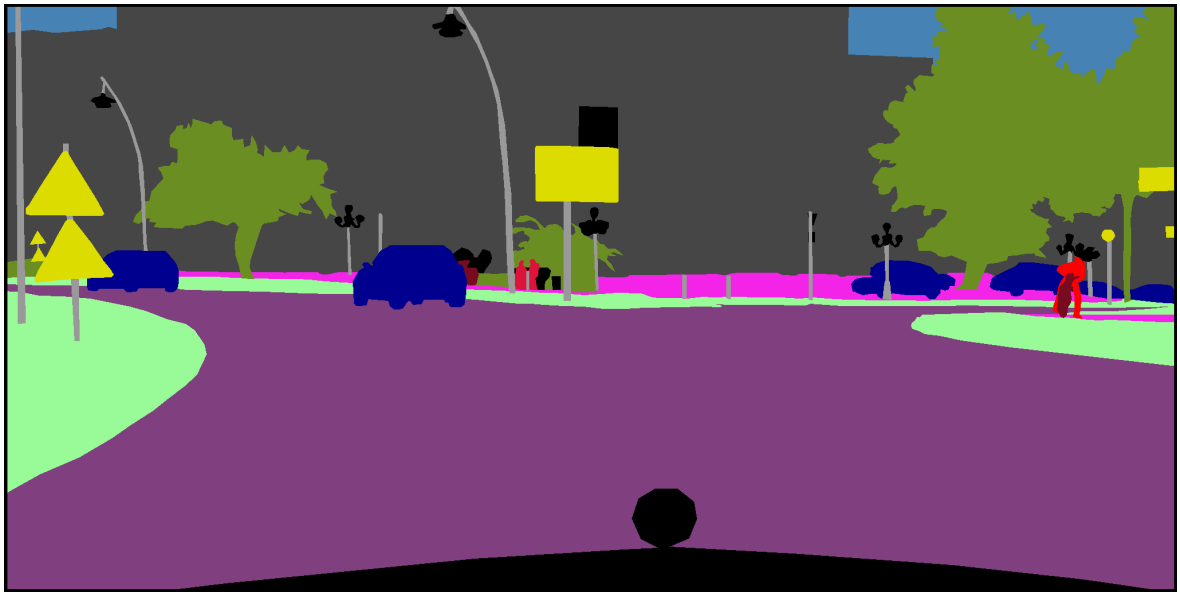
La organización del conjunto de datos en el proyecto tiene la estructura dividida en dos bloques, que a su vez están divididos en otros dos como se puede ver en la figura 5.1.

Figura 5.1: Estructura de *CityScapes* en el proyecto.

Esta primera división realizada en el conjunto de datos está hecha para separar en dos directorios las imágenes de sus anotaciones. Por un lado, la carpeta *leftImg8bit* contiene los

fotogramas captados directamente de las calles de las ciudades. Por otro lado, la carpeta *gtFine* contiene las anotaciones *fine* que se encuentran codificadas en ficheros con formato json que contienen los polígonos individuales de los objetos. Además, se pueden encontrar las mismas imágenes pero representadas por los polígonos y coloreadas por los distintos tipos de clases que representan. En la Figura 5.2 se puede ver un ejemplo de imagen que se utiliza como datos de entrada del modelo captada por la cámara en la calle, Figura 5.2a, y un ejemplo de la representación de las anotaciones de dicha imagen, Figura 5.2b.

(a) Imagen de una calle de Aquisgrán



(b) Anotaciones de la imagen

Figura 5.2: Ejemplo de imagen de *CityScapes*.

La segunda división, agrupa los datos de los conjuntos de entrenamiento y los conjuntos de test. Las distintas particiones de los conjuntos están hechas de forma que: el entrenamiento contiene 3325 imágenes pertenecientes a 20 ciudades distintas y el test contiene 1675 imágenes pertenecientes a 7 ciudades distintas.

5.1.2 Moto360 - Dominio Objetivo

Como dominio objetivo se va a utilizar el conjunto de datos creado a partir de las grabaciones de la cámara 360, comentado en la Sección 4.2.1. La base de datos Moto360 está compuesta por un conjunto de 2012 fotogramas. Cada fotograma tiene un tamaño de 5760x2880 píxeles.

Las imágenes se pueden clasificar en tres tipos distintos de áreas: autovía, interurbana y urbana. La cantidad de elementos que componen cada categoría está relacionada de forma directa con respecto a la duración del vídeo de cada una de ellas. Obteniendo para cada área el número de 162 imágenes de autovía, 646 imágenes de área interurbana y 1204 imágenes de autovía.

Al contrario que el dominio fuente, esta base de datos solo tiene etiquetada los coches, de manera que no hay que hacer ningún cambio en el código. Y su estructura, como se puede apreciar en la Figura 5.3, contiene dos archivos con formato json que guardan la información sobre las etiquetas en formato COCO de ambos conjuntos y dos directorios para diferenciar el conjunto de entrenamiento y el de test.

Figura 5.3: Estructura de Moto360 en el proyecto

Para la organización de los dos directorios, hay que realizar una división del conjunto de datos, obteniendo así el conjunto de entrenamiento y el conjunto de test. El conjunto de entrenamiento está formado por 1962 imágenes y el conjunto de test formado por 50 imágenes. Al igual que para obtener los fotogramas se ha utilizado la proporción de áreas con respecto a la duración del vídeo, se ha mantenido dicha proporción a la hora de crear este conjunto de test. Contiene 4 imágenes de autovía, 16 imágenes de área interurbana y 30 imágenes de área urbana.

En la Figura 5.4 se puede ver el ejemplo de entrada al modelo de una imagen captada por la cámara 360, en un área urbana y con sus respectivas etiquetas indicando la posición de los coches.

Figura 5.4: Ejemplo de imagen etiquetada de Moto360

5.2 Entorno de pruebas

El entorno donde se han ejecutado las diferentes pruebas hechas sobre el modelo, están realizadas en una máquina en local, la cual cuenta con las especificaciones siguientes:

- **CPU:** AMD Ryzen 5 5600X 6-Core Processor
- **RAM:** 2x16 GB DDR4
- **GPU:** NVIDIA GeForce RTX 4070

5.3 Hiperparámetros

Los hiperparámetros son parámetros cuyos valores se establecen antes de que comience el entrenamiento del modelo. Son los encargados de tareas como controlar la estructura interna de la red, la rapidez con la que aprende el modelo o de cómo se ajustan los pesos internamente. Los modelos de DL pueden tener desde pocos, hasta cientos de hiperparámetros. Es importante una buena elección de estos para poder ayudar a que la red aprenda como resolver la tarea de manera exacta y eficiente.

El modelo con el que se han realizado las pruebas tiene varios grupos de hiperparámetros que afectan a distintas partes del modelo:

- **Configuración base:** Especifica la ubicación de un archivo de configuración que sirve como base y puede contener configuraciones comunes que se comparten entre varios experimentos.
- **Configuración del modelo:** Define la arquitectura del modelo, la columna vertebral (o *backbone*), la RPN, la ROI y otros aspectos relacionados con la estructura del modelo.

- **Configuración del solucionador:** Contiene parámetros relacionados con la optimización, la programación de la tasa de aprendizaje y el número máximo de iteraciones durante el entrenamiento.
- **Configuración del cargador de datos:** Define parámetros relacionados con la carga de datos, como el porcentaje de datos supervisados utilizados durante el entrenamiento.
- **Configuración de las bases de datos:** Especifica las bases de datos utilizadas como dominio fuente y dominio objetivo y si se utilizan conjuntos de datos cruzados.
- **Configuración de adaptación al dominio:** Contiene parámetros relacionados con la adaptación de dominio no supervisada, como el peso de la pérdida de los datos etiquetados y sin etiquetar o la actualización del modelo del profesor.
- **Configuración de pruebas:** Indica la frecuencia de la evaluación durante el entrenamiento del modelo.

5.4 Métricas

La métrica con la que vamos a evaluar los experimentos en este trabajo va a ser la Precisión Media (AP). La AP es una métrica popular para medir la precisión y el rendimiento promedio en modelos de detección de objetos. Por lo tanto, la AP, mide la precisión de un modelo para detectar objetos en una imagen. La fórmula general para el cálculo de la precisión es la siguiente:

$$\text{Precisión} = \frac{\text{Verdaderos Positivos}}{\text{Verdaderos Positivos} + \text{Falsos Positivos}}$$

Esta métrica se utiliza en el ámbito de la evaluación de conjuntos de datos COCO. Tiene la palabra promedio en el nombre debido a que calcula la media de la precisión de todas las clases, pero en nuestro caso, solo evaluamos la detección de la clase coches. Por lo general, una red con una AP alta, localiza con precisión las ubicaciones de los objetos y los clasifica en sus respectivas clases de manera correcta. En nuestro caso, una puntuación alta, significa que se han ubicado con precisión los coches que aparecen en las imágenes.

5.5 Experimentos

Para poder evaluar qué modelo es el que mejores resultados obtiene, se realizan varios experimentos en el entorno local, ajustando diferentes hiperparámetros.

Debido a la naturaleza limitada y escasa de los recursos (económicos y temporales), es imposible realizar todas las combinaciones posibles para evaluar cual es la que mejores resultados obtiene. Por ello, es importante identificar cuales son los que al ajustarlos pueden ofrecer mejoras más significativas sobre el modelo.

La configuración base genérica utilizada que comparten los diferentes experimentos es la siguiente:

Código 5.1: Archivo de configuración base de la red

```
1 _BASE_: "./Base-RCNN-C4.yaml"
2 MODEL:
```

```
  META_ARCHITECTURE: "DAobjTwoStagePseudoLabGeneralizedRCNN"
  BACKBONE:
    NAME: "build_vgg_backbone"
  MASK_ON: False
  RESNETS:
    DEPTH: 101
  PROPOSAL_GENERATOR:
    NAME: "PseudoLabRPN"
  RPN:
    IN_FEATURES: ["vgg4"]
  ROI_HEADS:
    NAME: "StandardROIHeadsPseudoLab"
    LOSS: "CrossEntropy"
    NUM_CLASSES: 1
    IN_FEATURES: ["vgg4"]
  ROI_BOX_HEAD:
    NAME: "FastRCNNConvFCHead"
    NUM_FC: 2
    POOLER_RESOLUTION: 7
SOLVER:
  LR_SCHEDULER_NAME: "WarmupTwoStageMultiStepLR"
  STEPS: (60000, 80000, 90000, 360000)
  FACTOR_LIST: (1, 1, 1, 1, 1)
  MAX_ITER: 100000
  IMG_PER_BATCH_LABEL: 1
  IMG_PER_BATCH_UNLABEL: 1
  BASE_LR: 0.04
DATALOADER:
  SUP_PERCENT: 100.0
DATASETS:
DATASETS:
  CROSS_DATASET: True
  TRAIN_LABEL: ("cityscapes_fine_instance_seg_train",)
  TRAIN_UNLABEL: ("moto_train",)
  TEST: ("moto_test",)
SEMISUPNET:
  Trainer: "ateacher"
  BBOX_THRESHOLD: 0.8
  TEACHER_UPDATE_ITER: 1
  BURN_UP_STEP: 20000
  EMA_KEEP_RATE: 0.9996
  UNSUP_LOSS_WEIGHT: 0.5
  SUP_LOSS_WEIGHT: 1.0
  DIS_TYPE: "vgg4"
TEST:
  EVAL_PERIOD: 1000
```

Debido al funcionamiento de la red, el evaluador del modelo estudiante de comunicación entre dominios muestra los resultados de este cuando se ha cumplido el número de iteraciones indicado en el hiperparámetro *burn_up_step*. Es por eso que el eje X comienza en este número.

5.5.1 Reducción de imágenes

Una técnica común utilizada en DL es redimensionar el tamaño de las imágenes durante el entrenamiento y el test, para que los modelos se ejecuten más rápido. Una imagen con el doble de tamaño requiere que la red aprenda cuatro veces más la cantidad de píxeles, lo cual suma tiempo y recursos.

Debido a la gran resolución que tienen las imágenes del conjunto de datos Moto360, se ha probado a aplicar esta técnica. Se han realizado dos experimentos, uno dejando las imágenes sin aplicar ninguna reducción, con una resolución de 5760x2880 píxeles y otro aplicando una

reducción y obteniendo una resolución de 1333x800 píxeles.

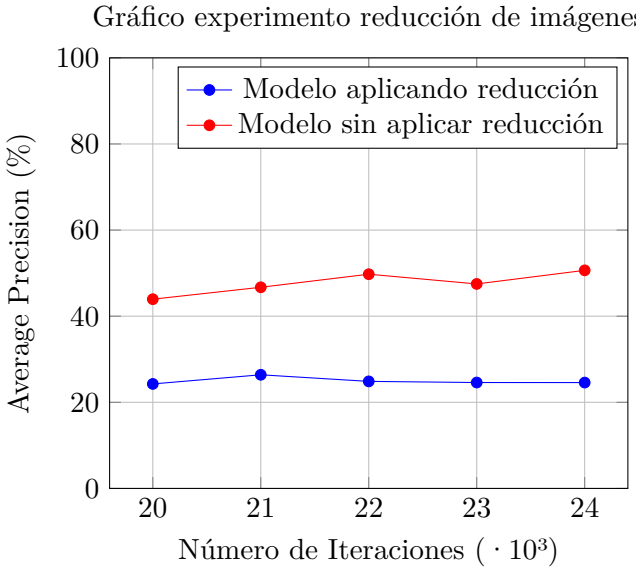


Figura 5.5: Resultados con y sin aplicar reducción de imágenes

Los resultados de dichos experimentos se puede apreciar en la Figura 5.5. Con un primer vistazo, se puede ver la gran diferencia existente entre los dos modelos entrenados y evaluados. El modelo entrenado con la reducción de imágenes tiene la mitad de precisión a lo largo de las cinco mil iteraciones. Con estos valores obtenidos, se mantendrá la resolución normal de las imágenes, puesto que aunque ahorre tiempo de entrenamiento, se sacrifica mucho rendimiento. Este resultado puede ser debido a la naturaleza de las imágenes de la cámara 360 puesto que los coches alejados de la moto por delante o por detrás se ven más pequeños que con una cámara normal, y si a eso le sumas la reducción, obtienes que el modelo no termina de obtener buenos resultados.

5.5.2 Burn up step

El *burn up step* se refiere al número de pasos iniciales donde el modelo se adapta más intensamente a los datos, tanto supervisados, como los no supervisados, estableciendo una base sólida antes de realizar los ajustes.

Este hiperparámetro tiene suma importancia, puesto que es el encargado de inicializar la red de la manera más efectiva antes de realizar los ajustes más en detalle, permitiendo así que el modelo se familiarice con los datos.

Además, un valor bien ajustado en este elemento puede mejorar la capacidad de generalizar datos no vistos por la red. Sin embargo, un valor elevado puede provocar un sobreajuste (o *overfitting*) y uno pequeño puede reducir la influencia sobre los datos no etiquetados y que el modelo no llegue a adaptarse de una manera correcta.

Por ello, para obtener un valor que se ajuste lo mejor posible al modelo, se han realizado dos experimentos. Los experimentos tienen la configuración base comentada, excepto el valor del hiperparámetro *burn_up_step*. El primer experimento está realizado poniendo a prueba

su funcionamiento con un valor de 10000 iteraciones, por otro lado, el segundo experimento está realizado poniendo a prueba su funcionamiento con un valor de 20000.

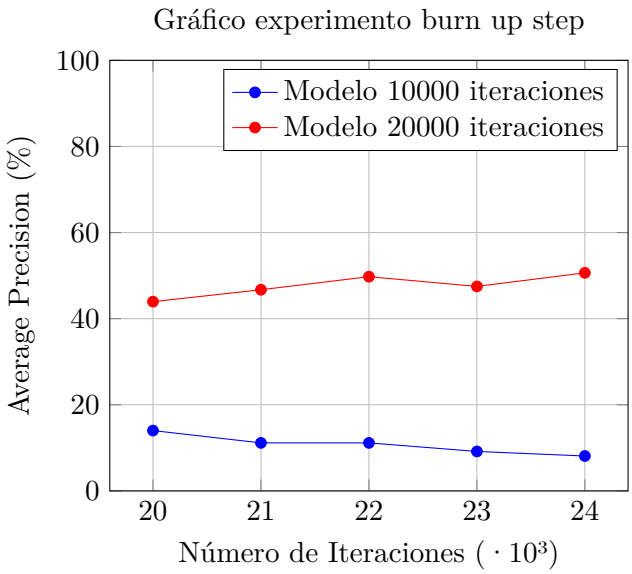


Figura 5.6: Resultados de ajustar el hiperparámetro burn up step a 10000 y 20000 iteraciones

Los resultados obtenidos, como se pueden ver en la Figura 5.6, dejan claro a simple vista cual es la mejor opción. El modelo que realiza el *burn_up_step* de 10000 iteraciones no solo tiene poca precisión al principio, sino que, además, cada vez tiene menos. Por otro lado, se puede apreciar que el modelo de 20000 iteraciones comienza con una buena precisión y también continúa aumentando con el paso de las iteraciones.

5.5.3 Peso de pérdida no supervisada

El peso de pérdida no supervisada (o *unsup_loss_weight*) controla la ponderación relativa de la pérdida generada por las muestras no etiquetadas durante el entrenamiento. Permite controlar la influencia relativa de los datos no etiquetados en la actualización de pesos.

Al ajustar este hiperparámetro, puedes regular la importancia de la información proporcionada por los datos no etiquetados para mejorar la generalización del modelo.

Un valor adecuado contribuye a representaciones más robustas y útiles y a la adaptación a las particularidades de los datos no etiquetados. Por una parte, un valor mayor dará más peso a las muestras no etiquetadas, si se tiene una confianza sobre los datos no etiquetados puede ser beneficioso. Por otra parte, un menor valor dará menos importancia a estas muestras, lo cuál también puede ser positivo si se sospecha que las muestras puedan contener ruido.

Para evaluar este hiperparámetro se han realizado tres experimentos, todos tienen la configuración base, excepto el campo *unsup_loss_weight*, el cual es 0.25, 0.5 o 1.

A la hora de evaluar los resultados obtenidos del experimento, como se puede ver en la Figura 5.7, son valores muy parejos. No obstante, aunque en un primer vistazo general cueste decantarnos por un modelo en concreto, no todos tienen el mismo rendimiento. Para comenzar, el modelo con 0.25 tiene un buen desarrollo en la primera mitad, pero no acaba

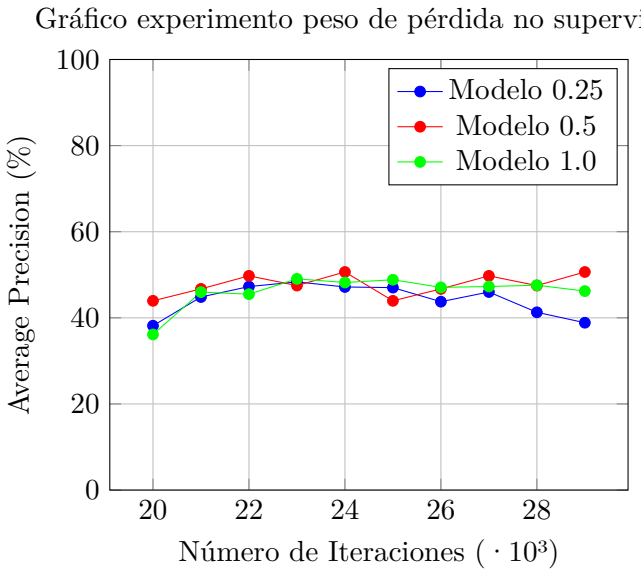


Figura 5.7: Comparación entre tres modelos con el peso de pérdida no supervisada diferente

de adaptarse con el paso de las iteraciones y acaba disminuyendo su precisión. Seguido de este, el modelo 1.0, comienza con una mala precisión pero va aumentando y se mantiene muy constante, sin mejoras aparentes, pero tampoco empeora. Por último, el modelo 0.5, comienza con el mejor rendimiento y, a pesar de una caía en la mitad de las iteraciones, acaba siendo el que más precisión tiene. Con esta información, se puede decir que realmente no es rentable ajustar mucho este hiperparámetro en este modelo, ya que aunque hayan diferencias entre los experimentos, no son muy grandes.

5.5.4 Tasa de aprendizaje base

La tasa de aprendizaje base (o *base_lr*) indica el valor de la tasa de aprendizaje (*learning rate*) del modelo durante el entrenamiento. La tasa de aprendizaje controla el tamaño de los pasos que el algoritmo toma durante el proceso de actualización de los pesos del modelo. Por lo tanto, ya que afecta a la rapidez con la que la red aprende, la optimización del modelo depende de forma directa de este hiperparámetro.

Por una parte, un valor alto puede acelerar la convergencia del modelo, sobretodo al principio del entrenamiento, sin embargo, también puede que ocurra lo contrario y que la red diverja. Por otra parte, un valor bajo, puede proporcionar estabilidad y prevenir problemas, pero puede que el modelo requiera un número mayor de iteraciones para poder tener un rendimiento óptimo.

Por lo tanto, una tasa de aprendizaje bien ajustada es crucial para garantizar un modelo sin oscilaciones ni divergencias. Para obtener dicho ajuste y partiendo de la configuración base, se han realizado tres experimentos modificando el valor del elemento *base_lr*.

Al modificar el valor del *base_lr* se obtienen resultados diferentes, como se puede apreciar en la Figura 5.8. De manera general se puede observar como el modelo 0.1 tiene mucha menos precisión en comparación con los otros dos y de manera individual tiene una precisión muy

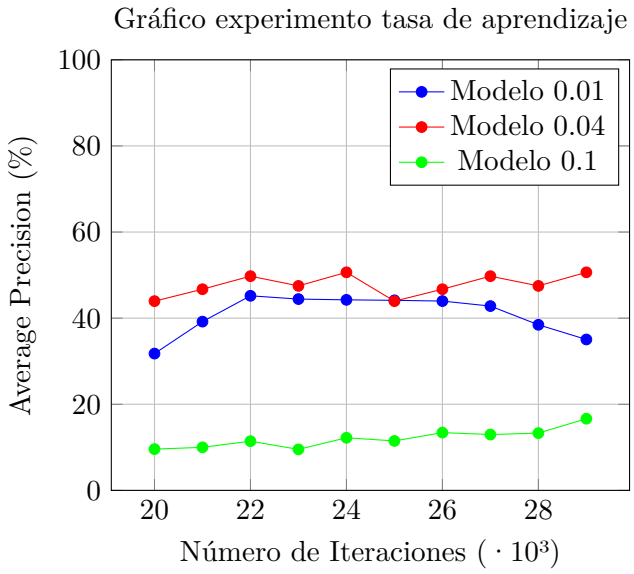


Figura 5.8: Evaluación de tres modelos comparando distintas tasas de aprendizaje

mala y por lo tanto, diverge. Por otro lado, el modelo 0.01, parte con una precisión baja, la cual aumenta con el paso de las iteraciones, pero, llegados a un punto comienza a disminuir y termina con el mismo porcentaje que con el que comenzó. Por último, el modelo 0.04, obtiene un mejor rendimiento en todas las etapas que se ha evaluado el modelo, salvo en una que comparte precisión, y además, sigue mejorando y eso significa que está convergiendo.

5.5.5 Conclusión

Tras realizar los diferentes experimentos, se puede concluir con que el modelo que, en general, ha sacado un mejor rendimiento con respecto a los otros modelos entrenados, es el modelo configurado como modelo base, Figura 5.1. La mejor precisión media obtenida por este modelo es en la iteración 24000, donde obtiene un 50%.

6 Conclusiones

En este trabajo se ha desarrollado una investigación y todo el proceso completo de la puesta en marcha de un modelo en el ámbito de la visión artificial con imágenes 360.

La motivación surgió a raíz de conocer el número de víctimas mortales en motos y la necesidad de reducir este número. Tras una investigación en el marco teórico, se propuso resolver esta problemática mediante el uso de DL, más concretamente, con la aplicación de la UDA. A su vez, este método, planteó un desafío crucial sobre la manera de abordar la dependencia de las anotaciones sobre los conjuntos de datos a la hora de conseguir que una ANN aprenda a realizar una tarea concreta. A la hora de proponer una metodología para la resolución de estos obstáculos, se realizó una búsqueda sobre diferentes tipos de modelos, para al final utilizar la red *Adaptive Teacher*, debido a los buenos resultados obtenidos en el aprendizaje de tareas parecidas. La puesta en marcha del desarrollo abarcó todos los pasos, desde la obtención, tratamiento y etiquetado (del conjunto test) de los datos con la cámara 360, pasando por la instalación de todas las librerías necesarias con sus versiones correspondientes para evitar los conflictos de compatibilidad, hasta la puesta en marcha del modelo con nuestros datos. Por último, se realizaron varios experimentos para ajustar los valores de los hiperparámetros con mayor relevancia.

Los experimentos que se llevaron a cabo consisten en el uso de la técnica de redimensionar las imágenes, utilizada comúnmente en modelos CNN, y el ajuste de valores de varios hiperparámetros como son el *burn up step*, el peso de pérdida no supervisada y la tasa de aprendizaje. El objetivo de de estos es conseguir un aumento del rendimiento y de la precisión, para conseguir los mejores resultados posible por parte del modelo dentro de las limitaciones de los recursos. La técnica de reducir los datos de entrada obtuvo como resultado que se mejoraba el tiempo de ejecución del modelo, pero debido a la composición de las imágenes 360, la precisión empeoraba considerablemente. Con el ajuste de los diferentes hiperparámetros, no solo se consiguió una precisión promedia del 50% con un modelo entrenado, sino también se llegó a la conclusión de qué hiperparámetros eran más influyentes sobre los resultados para poder justificar una mayor inversión de recursos en su ajuste.

Además de obtener una buena precisión en el modelo, a nivel de proyecto, también se han obtenido buenos resultados. Se han logrado completar los objetivos propuestos, tanto los académicos, como a nivel de proyecto. En concreto, cabe destacar los conocimientos obtenidos sobre DL, la comprensión de artículos científico, Python y el control de librerías y entornos virtuales.

Como posible trabajo futuro, sería interesante poner el modelo en marcha en una situación de conducción real y realizar experimentos con una moto sobre su rendimiento. Dentro de estos experimentos se podría: probar en qué tipo de carreteras (autovía, área interurbana o área urbana) o lugares funciona mejor, como afecta a su rendimiento bajo qué condiciones meteorológicas se conduzca, si influye el momento del día o la congestión del tráfico, etc.

Finalmente, si se obtienen buenos resultados de los experimentos realizados fuera del la-

boratorio, significa que se puede utilizar como base para la creación de ADAS. Por ejemplo, se podría conectar a un piloto en la moto (que emita una luz o un sonido), que se active y avise al conductor poniéndole alerta cuando un coche se acerque peligrosamente poniendo en riesgo su integridad física. De esta manera, poder lograr el objetivo inicial y contribuir a la reducción del número de accidentes.

Bibliografía

AENOR. (1997). *norma une 50136:1997.* Descargado de http://docubib.uc3m.es/CURSOS/Documentos_cientificos/Normas%20y%20directrices/UNE_50136=ISO%207144.pdf

Akyildiz, I. F., Pompili, D., y Melodia, T. (2005). Underwater acoustic sensor networks: research challenges. *Ad hoc networks*, *3*(3), 257–279.

Balsamiq. (s.f.). https://balsamiq.com/. (Accessed: 2018-04-10)

BOE. (2012, marzo). *Resolución de 7 de marzo de 2012, de la universidad de alicante, por la que se publica el plan de estudios de graduado en ingeniería multimedia.* BOE, 22 marzo de 2012. Descargado de http://www.boe.es/boe/dias/2012/03/22/pdfs/BOE-A-2012-4008.pdf

Buskirk, T. D., Kirchner, A., Eck, A., y Signorino, C. S. (2018, 2 de 1). An introduction to machine learning methods for survey researchers. *Survey Practice*, *11*(1). doi: 10.29115/SP-2018-0004

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., … Schiele, B. (2016, 4). The cityscapes dataset for semantic urban scene understanding.

DGT. (2022). Siniestros de tráfico durante 2022. *DGT*. Descargado de https://www.dgt.es/comunicacion/notas-de-prensa/1.145-personas-fallecieron-en-siniestros-de-trafico-durante-2022/

fcakyon. (2023, marzo). *wkentaro/labelme2coco: v0.2.4.* Descargado de https://pypi.org/project/labelme2coco/

García, S. M. (2019). Ética e inteligencia artificial. *https://media.iese.edu/research/pdfs/ST-0522.pdf*.

Girshick, R. (2015). Fast r-cnn. En *2015 ieee international conference on computer vision (iccv)* (p. 1440-1448). doi: 10.1109/ICCV.2015.169

He, K., Gkioxari, G., Dollár, P., y Girshick, R. (2017, 3). Mask r-cnn.

Heinz, M., Carsten, y Hoffmann, J. (2014, March). *The listings package, march 2014.* http://texdoc.net/texmf-dist/doc/latex/listings/listings.pdf. Descargado 12/12/2014, de http://texdoc.net/texmf-dist/doc/latex/listings/listings.pdf

IESE. (2016). *Iese cities in motion index* (Tesis Doctoral no publicada). University of Navarra.

INTERNATIONAL, S. (2021, 4). Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. *Mobilus, Vehicle Electrification*. Descargado de `https://www.sae.org/standards/content/j3016_202104/`

José Manuel Abad, J. G., y Alameda, D. (2018). El riesgo de que aumenten las muertes en moto. *El País*. Descargado de `https://elpais.com/politica/2018/05/23/sepa_usted/1527088171_417726.html`

Li, Y.-J., Dai, X., Ma, C.-Y., Liu, Y.-C., Chen, K., Wu, B., … Vajda, P. (2022). Cross-domain adaptive teacher for object detection. En *Ieee conference on computer vision and pattern recognition (cvpr)*.

Liang, M., y Hu, X. (2015, June). Recurrent convolutional neural network for object recognition. En *Proceedings of the ieee conference on computer vision and pattern recognition (cvpr)*.

Liu, Y.-C., Ma, C.-Y., He, Z., Kuo, C.-W., Chen, K., Zhang, P., … Vajda, P. (2021, 2). Unbiased teacher for semi-supervised object detection.

Morales, E. F., y Escalante, H. J. (2022). Chapter 6 - a brief introduction to supervised, unsupervised, and reinforcement learning. En A. A. Torres-García, C. A. Reyes-García, L. Villaseñor-Pineda, y O. Mendoza-Montoya (Eds.), *Biosignal processing and classification using computational learning and intelligence* (p. 111-129). Academic Press. Descargado de `https://www.sciencedirect.com/science/article/pii/B9780128201251000178` doi: https://doi.org/10.1016/B978-0-12-820125-1.00017-8

Mundial, B. (2023). *El transporte sostenible.* Descargado de `https://www.bancomundial.org/es/topic/transport/overview#:~:text=El%20transporte%20es%20fundamental%20para,estos%20beneficios%20no%20se%20materializan`

Newton, I. (1846). *Newton's principia: the mathematical principles of natural philosophy.* First American edition.

Olivas, E. S., Guerrero, J. D. M., Sober, M. M., Benedito, J. R. M., y Lopez, A. J. S. (2009). *Handbook of research on machine learning applications and trends: Algorithms, methods and techniques - 2 volumes.* Hershey, PA: Information Science Reference - Imprint of: IGI Publishing.

Qiu, J., Wu, Q., Ding, G., Xu, Y., y Feng, S. (2016, 12). A survey of machine learning for big data processing. *EURASIP Journal on Advances in Signal Processing*, *2016*, 67. doi: 10.1186/s13634-016-0355-x

Ren, S., He, K., Girshick, R., y Sun, J. (2015, 6). Faster r-cnn: Towards real-time object detection with region proposal networks.

Rosenbaum, J. (1988). *Bulk acoustic wave theory and devices.* Artech House on Demand.

Rouhiainen, L. (2018). Inteligencia artificial. *Madrid: Alienta Editorial*, 20–21.

Sakaridis, C., Dai, D., y Van Gool, L. (2018, Sep). Semantic foggy scene understanding with synthetic data. *International Journal of Computer Vision*, *126*(9), 973–992. Descargado de https://doi.org/10.1007/s11263-018-1072-8

Shaw, M., y Garlan, D. (1996). *Software architecture: perspectives on an emerging discipline* (Vol. 1). Prentice Hall Englewood Cliffs.

Simonyan, K., y Zisserman, A. (2014, 9). Very deep convolutional networks for large-scale image recognition.

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., y Liu, C. (2018). A survey on deep transfer learning. En V. Kůrková, Y. Manolopoulos, B. Hammer, L. Iliadis, y I. Maglogiannis (Eds.), *Artificial neural networks and machine learning – icann 2018* (pp. 270–279). Cham: Springer International Publishing.

Villa, D. (2008, 4). *Latex: Listados de código cómodos y resultones con listings.* http://crysol.org/es/node/909. Descargado 12/12/2014, de http://crysol.org/es/node/909

Wada, K., mpitid, Buijs, M., N., Z. C., , Kubovčík, B. M., … Toft, H. (2021, noviembre). *wkentaro/labelme: v4.6.0.* Zenodo. Descargado de https://doi.org/10.5281/zenodo.5711226 doi: 10.5281/zenodo.5711226

Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., y Girshick, R. (2019). *Detectron2.* Descargado de https://github.com/facebookresearch/detectron2

Zou, J., Han, Y., y So, S.-S. (2009). Overview of artificial neural networks. En D. J. Livingstone (Ed.), *Artificial neural networks: Methods and applications* (pp. 14–22). Totowa, NJ: Humana Press. Descargado de https://doi.org/10.1007/978-1-60327-101-1_2 doi: 10.1007/978-1-60327-101-1_2

Lista de Acrónimos y Abreviaturas

ACC	Control de Crucero Adaptativo.
ADAS	Sistema Avanzado de Ayuda a la Conducción.
AEB	Frenado de Emergencia Autónomo.
ANN	Red Neuronal Artificial.
AP	Precisión Media.
BSD	Alerta de Objetos en el Punto Ciego.
CNN	Red Neuronal Convolucional.
CUDA	Arquitectura Unificada de Dispositivos de Cómputo.
DA	Adaptación al Dominio.
DDD	Sistema de Detección de Fatiga.
DL	Apredizaje Profundo.
EMA	Media Móvil Exponencial.
FCW	Alerta Aviso de Colisión.
GNSS	Sistema Global de Navegación por Satélite.
GPU	Unidad de Procesamiento Gráfico.
GRL	Capa Inversa del Gradiente.
IA	Inteligencia Artificial.
LDW	Alerta de Cambio de Carril.
LIDAR	Laser Imaging Detection and Ranging.
LKAS	Sistema de Mantenimiento de Carril.
ML	Aprendizaje Automático.
MLP	Perceptrón Multicapa.
NVA	Asistente de Visión Nocturna.
OpenCV	Librería de Visión por Computador de Código Abierto.
PCW	Alerta por Paso de un Peatón.
R-CNN	Red Neuronal Convolucional basada en Regiones.
ReLU	Unidad Lineal Rectificada.
RNN	Red Neuronal Recurrente.
ROI	Región De Interés.
RPN	Red Regional Propuesta.
SDA	Adaptación al Dominio Supervisada.
SSDA	Adaptación al Dominio Semi-Supervisada.
SVM	Máquina de vectores de soporte.
TFG	Trabajo Final de Grado.
TL	Aprendizaje por Transferencia.

TSR Lector de Señales de Tráfico.
UDA Adaptación al Dominio No Supervisada.