



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

 etsinf

Escola Tècnica
Superior d'Enginyeria
Informàtica

Escuela Técnica Superior de Ingeniería Informática
Universidad Politécnica de Valencia

Detección de defectos en objetos en movimiento mediante Redes Neuronales Convolucionales con optimizaciones específicas para hardware NVIDIA

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Haro Armero, Abel

Tutor: Flich Cardo, José
López Rodríguez, Pedro Juan

Curso 2024-2025

Resum

????

Paraules clau: ????, ????????, ????, ??????????????????

Resumen

????

Palabras clave: ????, ???, ??????????????????

Abstract

????

Key words: ????, ????, ????, ??????????????

Índice general

Índice general	v
Índice de figuras	vii
Índice de tablas	vii
<hr/>	
1 Introducción	1
1.1 Motivación	2
1.2 Objetivos	3
1.3 Estructura de la memoria	4
2 Estado del arte	5
2.1 Fundamentos y avances en redes neuronales para visión artificial	5
2.1.1 Fundamentos de la inteligencia artificial	5
2.1.2 Tareas fundamentales en visión por computador	6
2.1.3 Arquitectura y funcionamiento de las CNN	7
2.1.4 Detectores de dos etapas	9
2.1.5 Detectores de una etapa	10
2.1.6 Métricas de evaluación	12
2.2 Aceleradores de procesamiento gráfico	14
2.2.1 Ventajas de las GPUs para modelos de IA	14
2.2.2 NVIDIA en la aceleración de IA	16
2.2.3 Serie Jetson: Dispositivos de IA de bajo consumo	16
2.2.4 TensorRT: Framework de optimización	17
2.3 Seguimiento de objetos en tiempo real	17
2.4 Slicing Aided Hyper Inference	17
3 Análisis del problema	19
4 Diseño e implementación de la solución	21
4.1 Descripción del sistema	21
4.2 Diseño de las etapas del sistema	21
4.3 Segmentación de las etapas del sistema	21
5 Análisis de la solución	23
5.1 Variación de los parámetros	23
5.2 Tipo de segmentación	23
5.3 Talla del modelo	23
5.4 Precisión del modelo	23
5.5 Modo de energía y cores de la CPU	24
5.6 Tamaño de la imagen	24
6 Prueba de concepto	25
6.1 Construcción del entorno	25
6.2 Instalación del entorno	25
7 Conclusiones	27
Bibliografía	29

Apéndices

A	Configuración del sistema	31
A.1	Fase de inicialitzación	31
A.2	Identificación de dispositivos	31
B	??? ?????????????? ????	33

Índice de figuras

1.1	Evolución del interés público en inteligencia artificial según datos de Google Trends (2020-2025)	1
1.2	Proyección del consumo eléctrico de los centros de datos en el mundo	2
2.1	Estructura de un perceptrón multicapa (MLP).	6
2.2	Tareas fundamentales en visión por computador.	7
2.3	Relación entre Machine Learning, Deep Learning, CNN, Computer Vision y Human Vision.	7
2.4	Operación de convolución en una imagen.	8
2.5	Proceso de convolución aplicado a una imagen de un autobús.	8
2.6	Operación de max-pooling en una imagen.	9
2.7	Arquitectura de LeNet-5.	9
2.8	Proceso de búsqueda selectiva aplicado a una imagen.	10
2.9	Arquitectura de R-CNN.	10
2.10	Ejemplo de detección de objetos utilizando YOLO.	11
2.11	Arquitectura de YOLO	12
2.12	Evolución histórica de las características de los microprocesadores (1970-2020).	15

Índice de tablas

2.1	Análisis comparativo de las variantes de YOLO11 considerando precisión, velocidad y complejidad computacional.	12
5.1	Comparación de modelos en términos de inferencia, consumo de energía y potencia.	23

CAPÍTULO 1

Introducción

Durante los últimos años, la inteligencia artificial ha experimentado un crecimiento en popularidad sin precedentes, transformando nuestra capacidad tecnológica con herramientas revolucionarias. Este avance ha sido impulsado por la disponibilidad de grandes volúmenes de datos y el desarrollo de algoritmos avanzados, que han permitido a las máquinas aprender y adaptarse a situaciones complejas. Algunos campos destacados de aplicación incluyen el procesamiento del lenguaje natural, la visión por computador y la robótica. En particular, la visión por computador ha visto un auge significativo, con aplicaciones en áreas como la seguridad, la medicina y la automoción. Este creciente interés se refleja en la evolución del interés público en inteligencia artificial, como muestra la Figura 1.1, basada en datos de Google Trends [3].

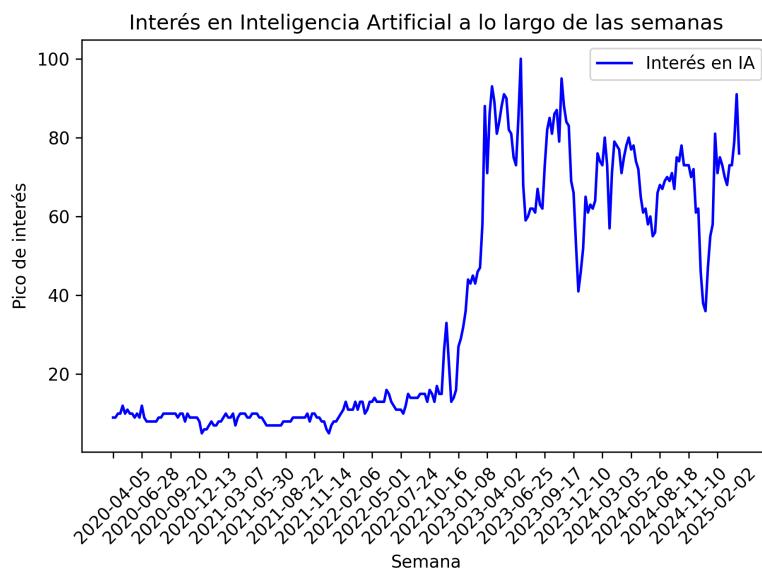


Figura 1.1: Evolución del interés público en inteligencia artificial según datos de Google Trends (2020-2025)

Este progreso ha sido posible gracias a los avances en redes neuronales convolucionales, que han revolucionado la capacidad de los sistemas para detectar y clasificar objetos en imágenes y videos con una gran precisión y velocidad.

Estos algoritmos de visión artificial requieren una potencia computacional significativa tanto para su entrenamiento como para su ejecución. Las CPUs (Unidades Centrales de Procesamiento) tradicionales resultan insuficientes para estas tareas, por lo que la industria ha desarrollado arquitecturas específicas como las GPUs (Unidades de Procesa-

miento Gráfico), TPUs (Unidades de Procesamiento Tensorial) y DLAs (Aceleradores de Aprendizaje Profundo). Estos componentes están optimizados para ejecutar operaciones de entrenamiento e inferencia de manera eficiente, permitiendo implementar sistemas de visión artificial capaces de procesar información visual en tiempo real. Sin embargo, estos aceleradores suelen presentar un consumo energético elevado, lo que plantea importantes retos de eficiencia y sostenibilidad.

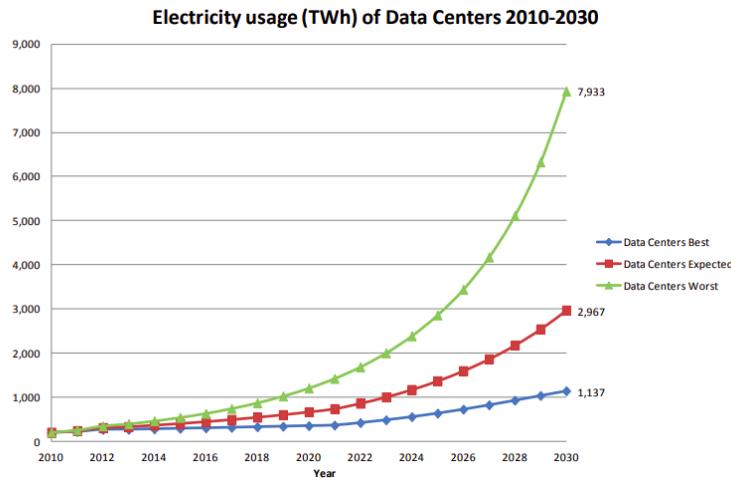


Figura 1.2: Proyección del consumo eléctrico de los centros de datos en el mundo

Como se observa en la Figura 1.2, el consumo eléctrico de los centros de datos en el mundo ha ido aumentando de forma exponencial, lo que plantea un desafío significativo para la sostenibilidad del crecimiento tecnológico [1]. En el peor escenario, esta tendencia podría llevar a un incremento insostenible en la huella de carbono del sector tecnológico, mientras que en el mejor de los casos, la adopción de tecnologías eficientes podría moderar este crecimiento. Este aumento del consumo energético no solo afecta a los centros de datos, sino también a los dispositivos embebidos y móviles, donde la eficiencia energética es crucial para prolongar la vida útil de las baterías y reducir el impacto ambiental.

Para enfrentar estos desafíos, se han desarrollado diversas técnicas de optimización y compresión que reducen el tamaño y la complejidad de los modelos neuronales manteniendo su rendimiento. Paralelamente, han surgido arquitecturas hardware específicamente diseñadas para la inferencia de modelos de aprendizaje profundo en entornos con restricciones energéticas. En este contexto, los dispositivos de la serie Jetson de NVIDIA destacan por ofrecer un equilibrio entre alto rendimiento en tareas de inteligencia artificial y un consumo energético contenido, ideal para aplicaciones embebidas de visión artificial.

La combinación de redes neuronales convolucionales y aceleradores hardware ha permitido la creación de sistemas de visión artificial que pueden detectar y clasificar objetos en movimiento, lo que es esencial en aplicaciones como la vigilancia, la conducción autónoma y la robótica.

1.1 Motivación

Los humanos somos capaces de ver y entender el mundo que nos rodea. Dada una imagen, podemos identificar objetos, reconocer patrones y tomar decisiones basadas en la información visual. Sin embargo, esta capacidad no es innata en las máquinas. La visión por computador es la ciencia que busca dotar a las máquinas de la capacidad de

interpretar y comprender imágenes y videos, emulando la forma en que los humanos percibimos el entorno.

Como se mencionó anteriormente, la inteligencia artificial ha revolucionado la forma en que interactuamos con la tecnología. Se ha convertido en una herramienta esencial para aplicar soluciones innovadoras en una amplia gama de campos. En particular, la visión por computador ha demostrado ser un área de gran potencial. También la existencia de dispositivos de bajo consumo, como los de la serie Jetson de NVIDIA, ha permitido llevar la inteligencia artificial a entornos de edge computing (cálculo en el borde), donde se acerca el procesamiento de datos a la fuente de información. Esto reduce la latencia y el consumo energético. Con todo esto, se abre un abanico de posibilidades para la implementación de sistemas de visión artificial en aplicaciones industriales.

Centrándose en el ámbito industrial, la detección y clasificación de objetos en movimiento es crucial para optimizar procesos, mejorar la seguridad y aumentar la eficiencia. En la mayoría de entornos productivos, la detección de defectos se realiza de forma manual, lo que puede ser ineficiente y propenso a errores. La automatización de este proceso mediante sistemas de visión artificial puede reducir costos, aumentar la precisión y mejorar la calidad del producto final.

La motivación de este trabajo radica en la necesidad de desarrollar un sistema de visión artificial capaz de detectar y clasificar objetos en movimiento en un entorno industrial, específicamente en una cinta transportadora.

1.2 Objetivos

El objetivo principal de este trabajo es desarrollar un sistema de visión artificial capaz de detectar y clasificar objetos en movimiento en una cinta transportadora utilizando redes neuronales convolucionales y aceleradores hardware de bajo consumo. Para lograr este objetivo, se plantean los siguientes objetivos específicos:

- Realizar un estudio del estado del arte en redes neuronales convolucionales, aceleradores hardware de bajo consumo y técnicas avanzadas de optimización para visión artificial.
- Desarrollar un conjunto de datos para el entrenamiento y evaluación del sistema, mediante la captura y etiquetado de imágenes de objetos en movimiento.
- Diseñar, entrenar y validar un modelo de red neuronal convolucional optimizado para la detección y clasificación en tiempo real de defectos en objetos en movimiento.
- Implementar un sistema completo de visión artificial que integre el modelo entrenado con los aceleradores hardware NVIDIA, enfocado en maximizar la eficiencia y minimizar la latencia.
- Analizar los cuellos de botella del sistema, y aplicar técnicas específicas de optimización para mejorar el rendimiento y la eficiencia energética.
- Cuantificar de manera exhaustiva el rendimiento del sistema mediante métricas precisas de exactitud (mAP, precisión, recall), latencia (FPS) y consumo energético (W, J/inferencia).
- Realizar un análisis comparativo sistemático entre diferentes configuraciones de hardware, software y parámetros de optimización para identificar la combinación que ofrezca el mejor equilibrio entre precisión, velocidad y eficiencia energética.

1.3 Estructura de la memoria

????? ?????????????? ?????????????? ?????????????? ?????????????? ??????????????

CAPÍTULO 2

Estado del arte

En este capítulo se realizará un estudio del estado del arte en los diferentes componentes que constituyen la base teórica y técnica de este trabajo. Primero, se examinarán las redes neuronales convolucionales, desde sus fundamentos hasta los modelos más recientes en detección de objetos. A continuación, se analizarán los aceleradores hardware de bajo consumo, con especial énfasis en la arquitectura y capacidades de los dispositivos NVIDIA Jetson. Posteriormente, se estudiarán los algoritmos de seguimiento de objetos en tiempo real, fundamentales para aplicaciones con elementos en movimiento. Finalmente, se explorará la técnica de Slicing Aided Hyper Inference (SAHI), una metodología avanzada para mejorar la detección de objetos pequeños o densamente agrupados. Este marco teórico permitirá contextualizar adecuadamente la solución propuesta para la detección de defectos en objetos en movimiento.

2.1 Fundamentos y avances en redes neuronales para visión artificial

En esta sección se realizará un estudio de las redes neuronales hasta las redes neuronales convolucionales, desde sus fundamentos hasta los modelos más recientes en detección de objetos. Se explicarán los conceptos básicos de las redes neuronales y la evolución de las arquitecturas.

2.1.1. Fundamentos de la inteligencia artificial

La *Inteligencia Artificial* es un campo de estudio que busca desarrollar sistemas capaces de realizar tareas que normalmente requieren inteligencia humana, como el reconocimiento de voz, la toma de decisiones y la comprensión del lenguaje natural. Dentro de este campo, existen diversas subdisciplinas, entre las cuales destacan el *Machine Learning* y el *Deep Learning*.

El *Machine Learning* o aprendizaje automático es una rama de la inteligencia artificial que se centra en el desarrollo de algoritmos y modelos que permiten a las máquinas aprender de los datos y realizar predicciones o tomar decisiones sin ser programadas explícitamente. Este enfoque se basa en la idea de que las máquinas pueden identificar patrones y relaciones en grandes conjuntos de datos, lo que les permite generalizar y adaptarse a nuevas situaciones.

El *Deep Learning* o aprendizaje profundo es una rama del aprendizaje automático que utiliza redes neuronales artificiales con múltiples capas para modelar y resolver problemas complejos. Este enfoque permite aprender representaciones jerárquicas de los datos,

donde cada capa extrae características cada vez más abstractas. Una de las arquitecturas fundamentales es el *Multilayer Perceptron* (MLP) o perceptrón multicapa, que consiste en una red de neuronas artificiales organizadas en al menos tres capas: una de entrada, una o más capas ocultas y una capa de salida, como se muestra en la Figura 2.1 [6]. En un MLP, cada neurona recibe un conjunto de entradas ponderadas por pesos, aplica una función de activación no lineal a la suma de estas entradas ponderadas, y produce una salida que se transmite a la siguiente capa. La capacidad de aprendizaje de estas redes se basa en el algoritmo de retropropagación (backpropagation), que ajusta iterativamente los pesos para minimizar el error entre las predicciones de la red y los valores reales. Esta estructura permite al Deep Learning abordar tareas complejas en visión por computador, procesamiento del lenguaje natural y otros dominios con un alto grado de precisión.

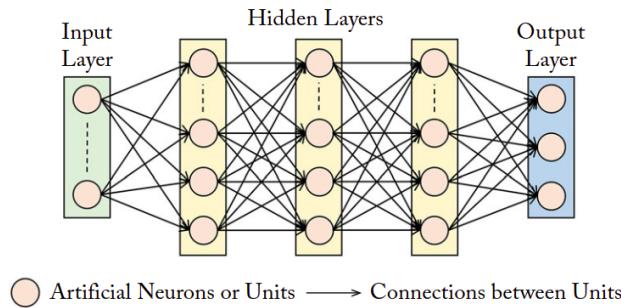


Figura 2.1: Estructura de un perceptrón multicapa (MLP).

2.1.2. Tareas fundamentales en visión por computador

En el ámbito del procesamiento de imágenes mediante técnicas de deep learning, existen diversas tareas con diferentes niveles de complejidad:

1. **Clasificación de imágenes:** Es la tarea más básica, donde la red neuronal asigna una etiqueta a toda la imagen. Por ejemplo, determinar si una imagen contiene un perro, gato o coche. El modelo genera un vector de probabilidades para cada clase posible.
2. **Clasificación con localización:** Además de clasificar el objeto principal, la red también proporciona un cuadro delimitador (bounding box) que indica dónde se encuentra ese objeto en la imagen. Es útil cuando existe un único objeto de interés.
3. **Detección de objetos:** Extiende la tarea anterior para identificar y localizar múltiples objetos en una imagen. Los algoritmos de detección se dividen principalmente en:
 - *Detectores de dos etapas:* Como R-CNN, Fast R-CNN y Faster R-CNN, primero generan propuestas de regiones que podrían contener objetos, y luego clasifican estas regiones. Son más precisos pero computacionalmente más costosos.
 - *Detectores de una etapa:* Como YOLO (You Only Look Once) y SSD (Single Shot MultiBox Detector), que predicen las cajas delimitadoras y las clases directamente en una sola pasada. Son más rápidos aunque tradicionalmente menos precisos.

Ambos enfoques proporcionan para cada objeto detectado su clasificación y cuadro delimitador.

4. Segmentación: Es la tarea más compleja, donde la red no solo identifica y localiza objetos, sino que también asigna una etiqueta a cada píxel de la imagen. Esto permite distinguir entre diferentes objetos y sus contornos, facilitando una comprensión más detallada de la escena.

La Figura 2.2 ilustra estas tareas fundamentales en visión por computador. Para este trabajo, nos centraremos en la tarea de detección de objetos, que es esencial para identificar y clasificar varios objetos en movimiento en un vídeo o imagen.

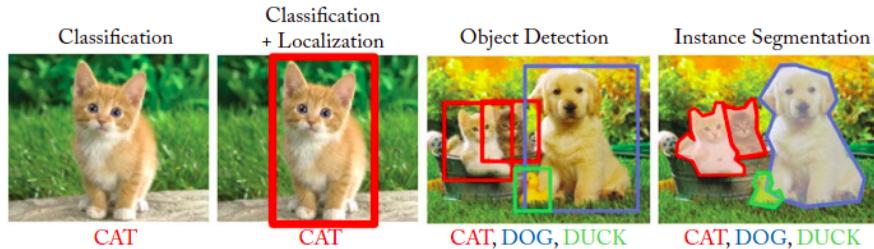


Figura 2.2: Tareas fundamentales en visión por computador.

2.1.3. Arquitectura y funcionamiento de las CNN

Las *Convolutional Neural Networks* (CNN) o redes neuronales convolucionales son un tipo específico de red neuronal profunda. Estas redes están diseñadas para procesar imágenes y extraer características relevantes de manera eficiente, lo que las hace especialmente adecuadas para tareas de visión por computador.

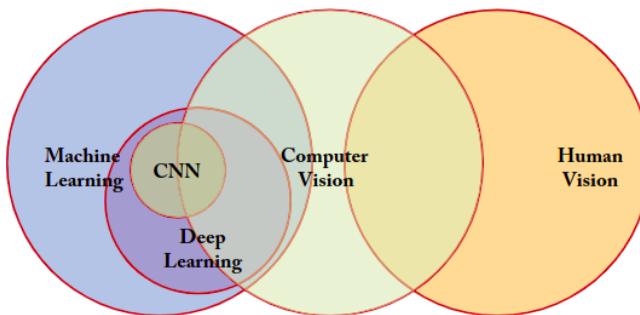


Figura 2.3: Relación entre Machine Learning, Deep Learning, CNN, Computer Vision y Human Vision.

La Figura 2.3 ilustra la relación entre estos conceptos [6]. Las CNN son una subcategoría del Deep Learning, que a su vez es una subcategoría del Machine Learning. Además, las CNN están estrechamente relacionadas con la visión por computador, que busca emular la capacidad de los humanos para interpretar imágenes y vídeos.

Las CNN se inspiran en la forma en que los humanos percibimos el mundo visual. Al igual que nuestro sistema visual, que procesa la información de manera jerárquica, las CNN utilizan capas convolucionales para extraer características de bajo nivel (como bordes y texturas) y capas más profundas para identificar patrones y objetos más complejos. Esta jerarquía de características permite a las CNN aprender representaciones ricas y abstractas de los datos visuales.

La operación de convolución es fundamental en las CNN. Esta operación consiste en aplicar un filtro (o kernel) a una imagen para extraer características locales. El filtro

se desliza sobre la imagen, multiplicando sus valores por los valores de la imagen en cada posición y sumando los resultados. Este proceso genera un mapa de activación que resalta las características relevantes de la imagen.

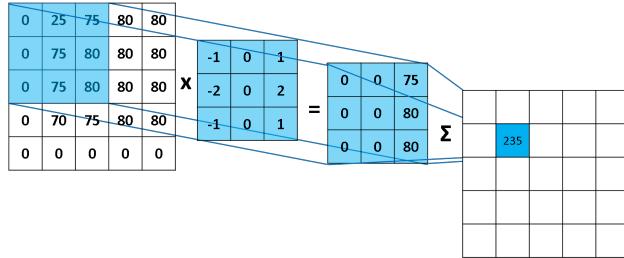


Figura 2.4: Operación de convolución en una imagen.

La Figura 2.4 muestra un ejemplo de la operación de convolución. En este caso, se aplica un filtro de 3x3 a una imagen de entrada, generando un mapa de activación que resalta las características detectadas por el filtro.

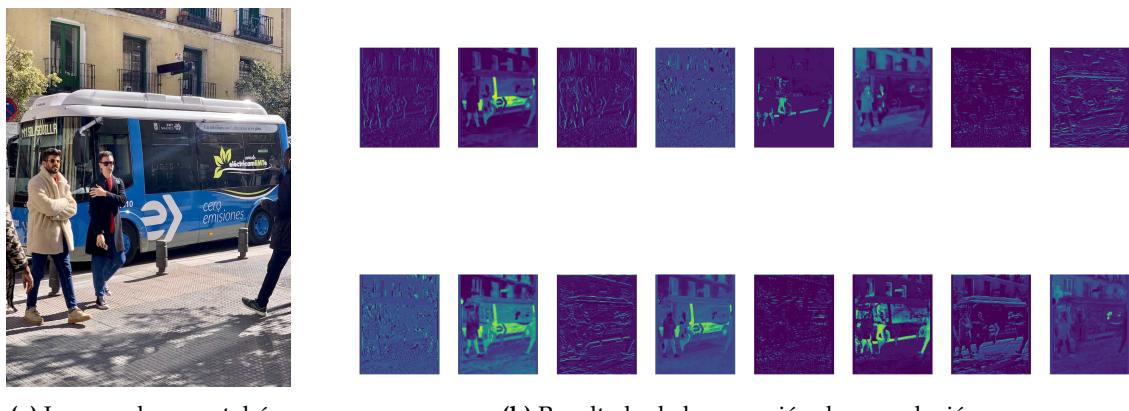


Figura 2.5: Proceso de convolución aplicado a una imagen de un autobús.

La Figura 2.5 ilustra el proceso de la primera convolución del modelo yolo11n [4]. En la parte izquierda se muestra la imagen original de un autobús, mientras que en la parte derecha se presenta el resultado de aplicar la operación de convolución. En este caso, los 16 filtros de la primera capa convolucional han detectado diferentes características de la imagen, como bordes y texturas. Este proceso se repite en múltiples capas, lo que permite a la red aprender representaciones cada vez más complejas de la imagen.

Además de las capas de convolución, las CNN incluyen capas de *pooling* que permiten reducir la dimensionalidad de las características extraídas, ayudando a prevenir el sobreajuste y a mejorar la eficiencia computacional. El *pooling* consiste en aplicar una operación de reducción (como el máximo o la media) a un conjunto de activaciones, lo que permite resumir la información y mantener las características más relevantes. Este proceso se ilustra en la Figura 2.6.

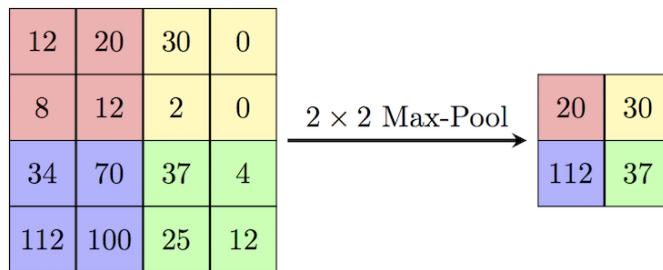


Figura 2.6: Operación de max-pooling en una imagen.

Por último, las CNN incluyen capas completamente conectadas (*fully connected*) al final de la red, que permiten realizar la clasificación final de los objetos detectados. Estas capas toman las características extraídas por las capas convolucionales y las combinan para generar una salida que representa la probabilidad de que un objeto pertenezca a una clase específica.

En la figura 2.7 se muestra la arquitectura de LeNet-5 [8], una de las primeras CNN desarrolladas. Esta red consta de varias capas convolucionales y de pooling, seguidas de capas completamente conectadas. LeNet-5 fue diseñada para la clasificación de dígitos manuscritos y sentó las bases para el desarrollo de arquitecturas más complejas y eficientes en la actualidad.

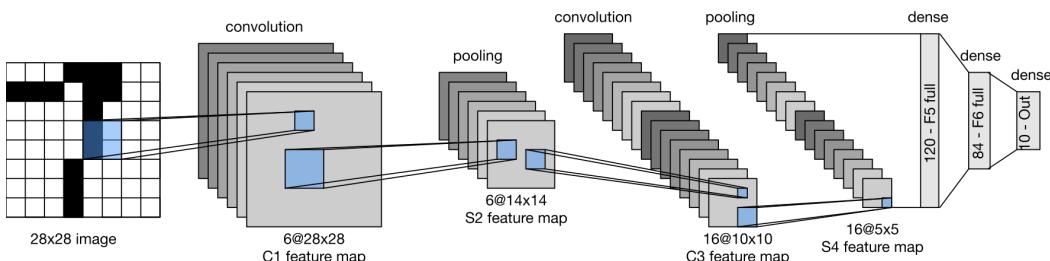


Figura 2.7: Arquitectura de LeNet-5.

2.1.4. Detectores de dos etapas

Los detectores de dos etapas, funcionan mediante un proceso secuencial: primero generan propuestas de regiones (region proposals) que podrían contener objetos y posteriormente clasifican estas regiones. Este enfoque favorece la precisión, aunque generalmente a costa de un mayor tiempo de procesamiento.

La primera arquitectura exitosa de detección de objetos basada en deep learning fue R-CNN (Regions with CNN features) [2]. Este modelo introdujo un enfoque de dos etapas que revolucionó el campo. En su primera fase, R-CNN utiliza un algoritmo de búsqueda selectiva (Selective Search) para generar aproximadamente 2,000 propuestas de regiones que podrían contener objetos. Este algoritmo de búsqueda selectiva divide la imagen en nodos y aristas, e iterativamente agrupa estas regiones en función del color, textura, tamaño y forma hasta que se obtienen las propuestas finales. En la figura 2.8[7] se muestra un ejemplo del resultado del algoritmo de búsqueda selectiva.

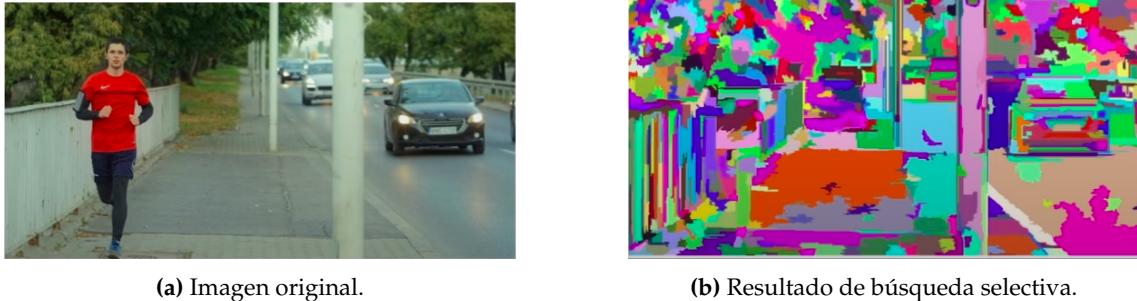


Figura 2.8: Proceso de búsqueda selectiva aplicado a una imagen.

En la segunda fase, cada región propuesta es redimensionada y procesada individualmente por una CNN para extraer características de alto nivel. Estas características alimentan posteriormente a un clasificador SVM (Support Vector Machine) para determinar la categoría del objeto y a un regresor lineal para mejorar la localización del cuadro delimitador. Como se ilustra en la Figura 2.9, este enfoque fue innovador pero computacionalmente costoso, ya que requiere procesar cada propuesta de región de manera independiente.

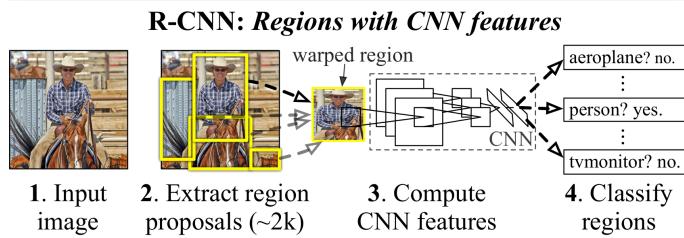


Figura 2.9: Arquitectura de R-CNN.

2.1.5. Detectores de una etapa

En contraste con los detectores de dos etapas, los detectores de una etapa (one-stage detectors) adoptan un enfoque más directo y eficiente. Estos detectores realizan la localización y clasificación de objetos simultáneamente en una sola pasada a través de la red, sin necesidad de un paso intermedio de generación de propuestas.

La arquitectura de los detectores de una etapa procesa la imagen completa una única vez, típicamente mediante una red troncal o *backbone* (generalmente una CNN) para la extracción de características. Estas características son posteriormente procesadas por componentes intermedios (*neck*) y alimentadas a una cabeza de detección (*detection head*) que predice simultáneamente las coordenadas de los cuadros delimitadores y las probabilidades de clase. Esta arquitectura de una etapa prioriza la velocidad de inferencia, resultando idónea para aplicaciones en tiempo real donde la latencia es un factor crítico, aunque pueda suponer una ligera concesión en la precisión máxima. Modelos representativos de este enfoque incluyen SSD (Single Shot MultiBox Detector)[10] y YOLO (You Only Look Once)[11]. Estos han demostrado un equilibrio eficaz entre rapidez y exactitud, permitiendo la detección en tiempo real incluso en dispositivos con recursos computacionales limitados.

YOLO (You Only Look Once) se destaca como una de las arquitecturas de detección de objetos más populares y efectivas. Concebida específicamente para la detección en tiempo real, YOLO introdujo un enfoque unificado que procesa la imagen completa en una sola pasada, realizando la localización y clasificación de forma simultánea. Esta me-

todología ha sido fundamental para su adopción en aplicaciones que requieren alta velocidad de procesamiento.

YOLO procesa la imagen completa de una vez. Divide la imagen en una cuadrícula de $S \times S$ celdas. Cada celda es responsable de detectar los objetos cuyo centro se encuentre dentro de ella. Para cada celda, YOLO predice B cuadros delimitadores (bounding boxes) y puntuaciones de confianza para esos cuadros. La puntuación de confianza indica la probabilidad de que haya un objeto en el cuadro y la precisión de la predicción del cuadro. Al mismo tiempo, predice las probabilidades de clase para cada objeto detectado en la celda.

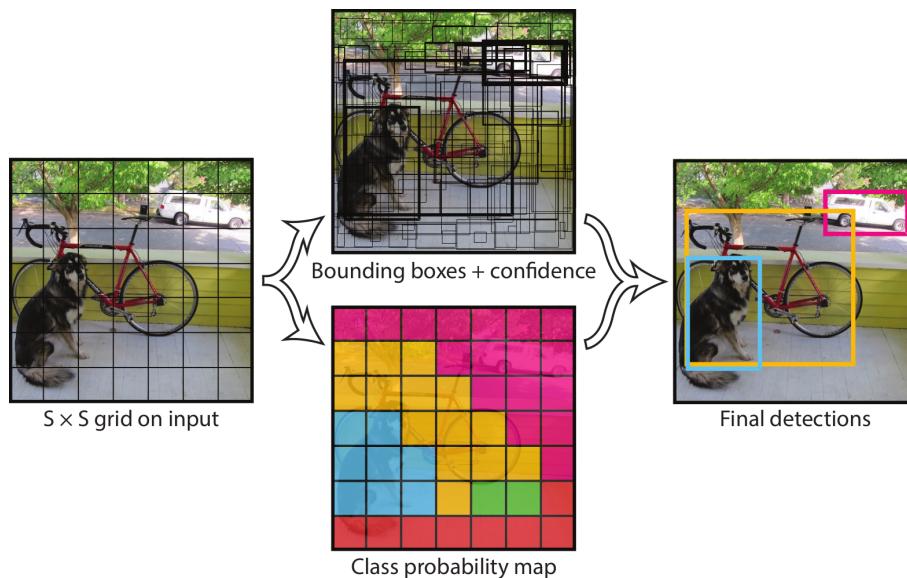


Figura 2.10: Ejemplo de detección de objetos utilizando YOLO.

Como se ilustra en la Figura 2.10[11], YOLO modela la detección como un problema de regresión. Las predicciones del modelo se codifican como un tensor de dimensiones $S \times S \times (B * 5 + C)$, donde el factor 5 corresponde a las coordenadas x, y , ancho, alto y la puntuación de confianza para cada cuadro delimitador, mientras que C representa el número de clases posibles. Por ejemplo, si se utilizan $B = 2$ cuadros delimitadores y $C = 20$ clases, el tensor de salida tendrá dimensiones $S \times S \times (2 * 5 + 20)$.

Una vez generadas todas las predicciones, YOLO implementa un post-procesamiento mediante supresión de no máximos (NMS) para eliminar detecciones redundantes y conservar únicamente las más precisas. Este enfoque unificado permite que YOLO procese imágenes a velocidades significativamente mayores que los detectores de dos etapas, mientras mantiene una precisión competitiva, lo que lo hace ideal para aplicaciones en tiempo real como las que se abordan en este trabajo.

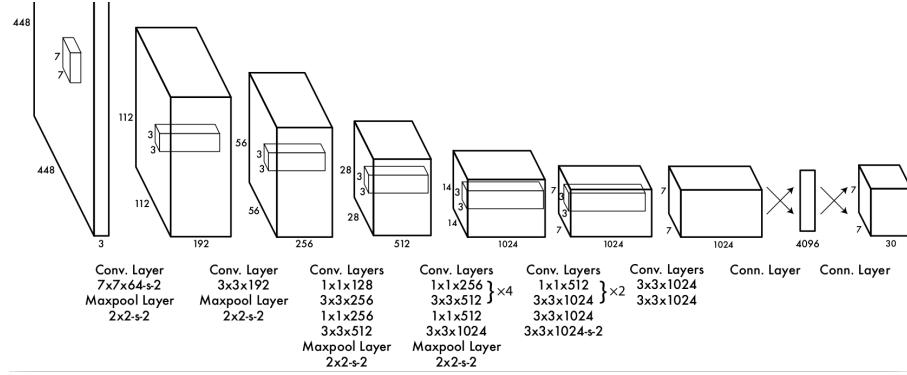


Figura 2.11: Arquitectura de YOLO

En la Figura 2.11 se presenta la arquitectura del modelo primigenio de YOLO [11]. Esta arquitectura se basa en una red neuronal convolucional que extrae características de la imagen de entrada y las procesa a través de varias capas para generar las predicciones finales. A lo largo de los años, se han desarrollado múltiples versiones y mejoras de YOLO, cada una optimizando aspectos como la precisión, la velocidad y la capacidad de detección de objetos pequeños o densamente agrupados.

En este proyecto, se ha seleccionado YOLO11 [4], que representa una mejora y optimización sobre las arquitecturas YOLO precedentes. Dicho modelo forma parte de la librería Ultralytics [5], la cual ofrece una implementación eficaz y sencilla de diversas variantes de YOLO. Las aportaciones de Ultralytics a la comunidad de visión artificial son notables, ya que sus herramientas y recursos agilizan el ciclo de vida (entrenamiento, evaluación, implementación) de los modelos YOLO en múltiples contextos. YOLO11 se ofrece en diversas configuraciones (11n, 11s, 11m, 11l y 11x).

Modelo	Tamaño (px)	Latencia CPU ONNX (ms)	Latencia T4 TRT10 (ms)	mAPval (50-95)	Parámetros / FLOPs (M / B)
YOLO11n	640	56.1 ± 0.8	1.5 ± 0.0	39.5	2.6 / 6.5
YOLO11s	640	90.0 ± 1.2	2.5 ± 0.0	47.0	9.4 / 21.5
YOLO11m	640	183.2 ± 2.0	4.7 ± 0.1	51.5	20.1 / 68.0
YOLO11l	640	238.6 ± 1.4	6.2 ± 0.1	53.4	25.3 / 86.9
YOLO11x	640	462.8 ± 6.7	11.3 ± 0.2	54.7	56.9 / 194.9

Tabla 2.1: Análisis comparativo de las variantes de YOLO11 considerando precisión, velocidad y complejidad computacional.

La Tabla 2.1 presenta un análisis comparativo de las distintas variantes de YOLO11 evaluadas sobre el dataset COCO [9]. Los datos revelan una clara correlación: los modelos más grandes ofrecen mayor precisión (mAP) a costa de mayor complejidad computacional y menor velocidad de procesamiento. Esta escalabilidad permite adaptar la implementación según los requisitos específicos de cada aplicación, balanceando efectivamente el rendimiento con las restricciones computacionales del sistema objetivo.

2.1.6. Métricas de evaluación

Para evaluar el rendimiento de los modelos de detección de objetos, se utilizan métricas específicas que permiten cuantificar la precisión y efectividad del sistema. Entre las métricas más comunes se encuentran:

- **Precisión (Precision):** Mide la proporción de verdaderos positivos (TP) respecto al total de predicciones positivas realizadas por el modelo. Se calcula como:

$$\text{Precision} = \frac{TP}{TP + FP}$$

donde FP representa los falsos positivos. Una alta precisión indica que el modelo realiza pocas predicciones incorrectas.

- **Recall:** Mide la proporción de verdaderos positivos respecto al total de objetos relevantes en la imagen. Se calcula como:

$$\text{Recall} = \frac{TP}{TP + FN}$$

donde FN representa los falsos negativos. Un alto recall indica que el modelo es capaz de detectar la mayoría de los objetos relevantes, aunque pueda incluir algunas predicciones incorrectas.

- **IoU (Intersection over Union):** Mide la superposición entre el cuadro delimitador predicho y el cuadro delimitador real. Se calcula como:

$$\text{IoU} = \frac{\text{Área de intersección}}{\text{Área de unión}}$$

Una IoU alta indica que el modelo ha localizado correctamente el objeto. Generalmente, se considera que una IoU superior a 0.5 indica una detección correcta.

- **mAP (mean Average Precision):** Es una métrica que combina precisión y recall, promediando la precisión a diferentes niveles de recall. Se utiliza para evaluar el rendimiento general del modelo en múltiples clases.

El cálculo del mAP se realiza en varios pasos: primero, para cada clase se calcula la curva PR (Precision-Recall) y se obtiene el Average Precision (AP), que representa el área bajo esta curva. Luego, el mAP se obtiene como la media de todos los AP de las diferentes clases. En detección de objetos, el mAP suele incorporar diferentes umbrales de IoU (Intersection over Union), expresándose como mAP@IoU=0.5 o mAP@IoU=0.5:0.95. Esta métrica es especialmente útil cuando las clases están desequilibradas, ya que da igual importancia a clases minoritarias y mayoritarias. Un valor de mAP cercano a 1 indica un modelo con alta precisión y recall en todas las clases.

- **Latencia:** Mide el tiempo que tarda el modelo en procesar una imagen y generar una predicción se mide en milisegundos (ms) y un valor bajo indica que el modelo es capaz de realizar inferencias rápidamente.
- **FPS (Frames Per Second):** Mide la velocidad de procesamiento del modelo, indicando cuántas imágenes puede procesar por segundo un alto valor de FPS indica que el modelo es capaz de realizar inferencias rápidamente.
- **F1 Score:** Es la media armónica entre precisión y recall, proporcionando un balance entre ambos. Se utiliza para evaluar el rendimiento del modelo en situaciones donde hay un desbalance entre clases. Se calcula como:

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Un F1 Score alto indica que el modelo tiene un buen equilibrio entre precisión y recall, lo que es especialmente importante en aplicaciones donde tanto la detección correcta como la minimización de falsos positivos son críticas.

2.2 Aceleradores de procesamiento gráfico

Un aspecto fundamental para el despliegue eficiente de modelos de inteligencia artificial es el hardware utilizado para su ejecución. A continuación, se analizan los principales aspectos de los aceleradores de procesamiento gráfico y su importancia en aplicaciones de visión artificial.

2.2.1. Ventajas de las GPUs para modelos de IA

A lo largo del desarrollo de la informática, la mejora del hardware ha sido una vía habitual para incrementar el rendimiento de las aplicaciones. Los programas podían ejecutarse más rápido simplemente esperando a que las nuevas generaciones de procesadores ofrecieran mayor potencia de cálculo. Sin embargo, este enfoque ha ido perdiendo efectividad debido a las limitaciones impuestas por el consumo energético y la disipación térmica, tal como describe la Dennard Scaling y el estancamiento de la ley de Moore.

La Dennard Scaling, formulada por Robert Dennard en 1974, establecía que a medida que los transistores se reducían de tamaño, su consumo de energía por unidad de área se mantenía constante. Esto significaba que, al reducir el tamaño de los transistores a la mitad, su área se reducía a un cuarto, pero su consumo de energía por unidad de área permanecía igual. Como resultado, el consumo total de energía se reducía a la mitad, permitiendo aumentar la frecuencia de reloj y el número de transistores sin incrementar significativamente el consumo total de energía. Este principio fue fundamental para el avance exponencial en el rendimiento de los procesadores durante décadas. Sin embargo, a partir de 2005, la Dennard Scaling dejó de cumplirse debido a varios factores físicos fundamentales. A escalas nanométricas, los efectos cuánticos y las fugas de corriente se volvieron significativos, impidiendo que el consumo de energía por unidad de área se mantuviera constante.

Por otro lado, la ley de Moore, formulada por Gordon Moore en 1965, predecía que el número de transistores en un chip se duplicaría aproximadamente cada año, predicción que posteriormente se ajustó a cada dos años. Durante décadas, esta ley se cumplió con notable precisión, permitiendo un crecimiento exponencial en la capacidad de procesamiento. Sin embargo, a medida que los transistores se acercan a escalas atómicas (actualmente en torno a los 3-5 nanómetros), los límites físicos y los desafíos de fabricación han ralentizado significativamente este ritmo de avance. Los costes de investigación y desarrollo para mantener esta tendencia se han disparado, y los beneficios en términos de rendimiento por transistor se han reducido.

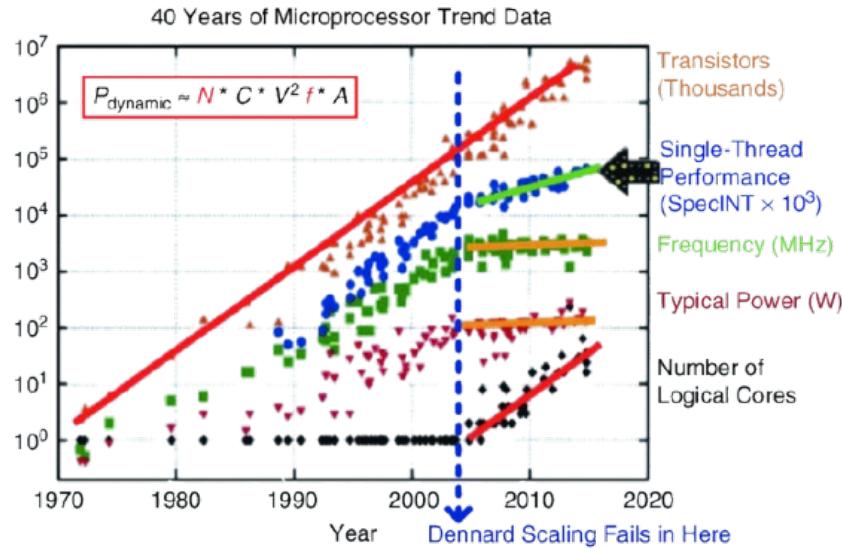


Figura 2.12: Evolución histórica de las características de los microprocesadores (1970-2020).

En la Figura 2.12[12] se ilustra claramente el impacto combinado del fin de la Dennard Scaling y el estancamiento de la ley de Moore. La gráfica muestra cinco métricas fundamentales en escala logarítmica: el número de transistores (línea naranja) que sigue la ley de Moore, el rendimiento de un solo hilo (línea azul), la frecuencia de reloj (línea verde), el consumo de potencia (línea roja) y el número de núcleos lógicos (línea negra). La ecuación de potencia dinámica ($P_{dynamic} \approx N * C * V^2 * f * A$) explica la relación entre el número de transistores (N), la capacitancia (C), el voltaje (V), la frecuencia (f) y el factor de actividad (A).

El punto de inflexión en 2005, marcado como *Dennard Scaling Fails in Here*, marca el momento en que la industria tuvo que cambiar radicalmente su estrategia. La frecuencia de reloj se estancó en torno a los 3-4 GHz, el rendimiento por núcleo comenzó a crecer más lentamente, y como respuesta, se adoptaron dos estrategias principales: el aumento del número de núcleos y la estabilización del consumo de potencia alrededor de los 100W. Este fenómeno ha llevado a la industria a buscar alternativas como las arquitecturas de dominio específico para continuar mejorando el rendimiento de los sistemas computacionales.

Las GPUs (Unidades de Procesamiento Gráfico) representan un claro ejemplo de arquitecturas de dominio específico, diseñadas para maximizar el rendimiento en tareas altamente paralelizables. El diseño de las GPUs sigue el paradigma *many-core*, lo que implica la integración de miles de núcleos de procesamiento relativamente simples en un solo chip. Esta arquitectura permite que las GPUs ejecuten múltiples hilos de manera concurrente, lo que resulta especialmente beneficioso para cargas de trabajo que pueden dividirse en tareas independientes.

El modelo de programación de las GPUs está basado en la ejecución masiva de hilos, organizados en bloques y rejillas (*blocks* y *grids*), según la terminología de CUDA, el modelo de programación desarrollado por NVIDIA. Cada hilo ejecuta la misma función, conocida como *kernel*, pero opera sobre diferentes fragmentos de datos. Este enfoque, conocido como SIMT (Single Instruction, Multiple Threads), permite aprovechar al máximo el paralelismo inherente a muchas aplicaciones de inteligencia artificial, como el entrenamiento y la inferencia de redes neuronales.

Para programar GPUs, se utilizan lenguajes y APIs especializadas como CUDA y OpenCL, que permiten al desarrollador definir explícitamente cómo se distribuyen los datos y las tareas entre los distintos núcleos de la GPU. El éxito en la programación de GPUs depende en gran medida de la capacidad para identificar y explotar el paralelismo de los algoritmos, así como de una gestión eficiente de la memoria, ya que el acceso a la memoria global de la GPU es mucho más lento que el acceso a la memoria compartida entre hilos de un mismo bloque.

- **Arquitectura de procesamiento paralelo:** Las GPUs contienen miles de núcleos más pequeños diseñados para operaciones simultáneas, mientras que las CPUs tienen menos núcleos pero más potentes.
- **Operaciones matriciales eficientes:** Las cargas de trabajo de IA dependen en gran medida de multiplicaciones matriciales, que las GPUs gestionan eficientemente mediante hardware especializado.
- **Mayor ancho de banda de memoria:** Las GPUs ofrecen un ancho de banda de memoria significativamente mayor, crucial para procesar grandes conjuntos de datos en redes neuronales.
- **Instrucciones especializadas:** Las GPUs modernas incluyen núcleos tensoriales específicamente diseñados para acelerar operaciones de aprendizaje profundo.
- **Optimización de rendimiento:** Las GPUs están optimizadas para el procesamiento por lotes en aprendizaje profundo, priorizando el rendimiento sobre la latencia.

2.2.2. NVIDIA en la aceleración de IA

NVIDIA ha desempeñado un papel fundamental en el desarrollo de aceleradores para inteligencia artificial:

- **Pionero en GPGPU:** NVIDIA lideró el movimiento de computación de propósito general en GPUs con la introducción de CUDA en 2006.
- **Desarrollo del ecosistema:** Creó bibliotecas de software completas (cuDNN, cuBLAS) específicamente para aprendizaje profundo.
- **Innovación en hardware:** Desarrolló hardware especializado como los Tensor Cores para operaciones matriciales de IA.
- **Estándar de la industria:** Estableció estándares de facto para hardware y software de aceleración de IA.

2.2.3. Serie Jetson: Dispositivos de IA de bajo consumo

La serie Jetson de NVIDIA representa una familia de dispositivos específicamente diseñados para implementar IA en el borde:

- **Enfoque en computación edge:** Diseñados para el despliegue de IA en el borde donde existen restricciones de energía.
- **Diseño System-on-Chip:** Integra núcleos de CPU ARM, núcleos de GPU NVIDIA y aceleradores especializados.

- **Gama de productos:** Abarca desde dispositivos de nivel básico (Jetson Nano) hasta dispositivos de alto rendimiento (Jetson AGX Orin).
- **Eficiencia energética:** Ofrece un rendimiento por vatio optimizado para aplicaciones embebidas.
- **Plataforma completa:** Proporciona hardware, pila de software y herramientas de desarrollo en un ecosistema unificado.

2.2.4. TensorRT: Framework de optimización

TensorRT es un framework desarrollado por NVIDIA para optimizar la inferencia de modelos de aprendizaje profundo:

- **Optimización de inferencia:** Específicamente diseñado para optimizar modelos entrenados para su despliegue.
- **Optimización de grafos:** Realiza fusión de capas, calibración de precisión y auto-ajuste de kernels.
- **Soporte de cuantización:** Permite precisión INT8/FP16 para reducir la huella de memoria y aumentar el rendimiento.
- **Ajuste específico para plataformas:** Selecciona automáticamente rutas de ejecución óptimas para hardware específico de NVIDIA.
- **Capacidades de integración:** Funciona con los principales frameworks como TensorFlow, PyTorch y ONNX.

2.3 Seguimiento de objetos en tiempo real

Explicación de como funcionan los algoritmos de multi-object tracking (MOT) en tiempo real, filtro de Kalman hasta BYTETrack.

2.4 Slicing Aided Hyper Inference

Explicación de la técnica de Slicing Aided Hyper Inference, como se utiliza para mejorar la precisión de los modelos de detección de objetos y como se aplica en este trabajo.

CAPÍTULO 3

Análisis del problema

????? ?????????????? ?????????????? ?????????????? ?????????????? ??????????????

CAPÍTULO 4

Diseño e implementación de la solución

En este capítulo se explicará la solución propuesta, como se ha diseñado y como se ha implementado.

4.1 Descripción del sistema

Descripción del sistema de visión artificial propuesto, como se ha diseñado y como se ha implementado.

4.2 Diseño de las etapas del sistema

Descripción de las etapas del sistema, como se han diseñado y como se han implementado.

Etapas del sistema:

- **Captura de imágenes:** Descripción de la etapa de captura de imágenes, como se ha diseñado y como se ha implementado.
- **Inferencia:** Descripción de la etapa de inferencia, como se ha diseñado y como se ha implementado.
- **Seguimiento:** Descripción de la etapa de seguimiento, como se ha diseñado y como se ha implementado.
- **Escritura de resultados:** Descripción de la etapa de escritura de resultados, como se ha diseñado y como se ha implementado.

4.3 Segmentación de las etapas del sistema

Tipos de segmentación de las etapas del sistema:

- **No segmentada:** Secuencial
- **Segmentación basada en hilos:** Cada etapa del sistema se ejecuta en un hilo diferente.

- **Segmentación basada en procesos:** Cada etapa del sistema se ejecuta en un proceso diferente.
- **Segmentación basada en hardware:** La etapa de inferencia se ejecuta en GPU, DLA0 y DLA1.
- **Segmentación basada en procesos con memoria compartida:** Cada etapa del sistema se ejecuta en un proceso diferente, pero comparten la memoria.

CAPÍTULO 5

Análisis de la solución

En este capítulo se analizará la solución propuesta variando los parámetros posibles

5.1 Variación de los parámetros

Explicación de los parámetros que se pueden variar en la solución propuesta y su efecto en el rendimiento del sistema.

—PRUEBA—

Model	IoU	CPU_Inference	GPU_Inference	DLA_Inference	CPU_Power	GPU_Power	DLA_Power	CPU_Energy	GPU_Energy	DLA_Energy
YOLOv11-N	0,85	45,2	12,3	15,8	8,2	12,5	6,8	369,64	153,75	107,44
YOLOv11-S	0,87	52,1	14,8	18,2	8,5	13,2	7,1	442,85	195,36	129,22
YOLOv11-M	0,89	68,4	18,2	22,5	9,1	14,8	7,8	622,44	269,36	175,5
YOLOv11-L	0,91	85,6	24,6	28,9	9,8	16,2	8,4	838,88	398,52	242,76

Tabla 5.1: Comparación de modelos en términos de inferencia, consumo de energía y potencia.

—PRUEBA—

5.2 Tipo de segmentación

En esta sección se analizará el rendimiento de la solución propuesta variando el tipo de segmentación de las etapas del sistema con gráficas y tablas.

5.3 Talla del modelo

En esta sección se analizará el rendimiento de la solución propuesta variando la talla del modelo de detección de objetos con gráficas y tablas.

5.4 Precisión del modelo

En esta sección se analizará el rendimiento de la solución propuesta variando la precisión del modelo de detección de objetos con gráficas y tablas.

5.5 Modo de energía y cores de la CPU

En esta sección se analizará el rendimiento de la solución propuesta variando el modo de energía del dispositivo y el número de cores de la CPU con gráficas y tablas.

5.6 Tamaño de la imagen

En esta sección se analizará el rendimiento de la solución propuesta variando el tamaño de la imagen de entrada del modelo con la técnica de Slicing Aided Hyper Inference (SAHI) con gráficas y tablas.

CAPÍTULO 6

Prueba de concepto

Aquí se explicará la implementación de la solución propuesta en el entorno de producción con la cinta transportadora.

6.1 Construcción del entorno

6.2 Instalación del entorno

????? ?????????????? ?????????????? ?????????????? ?????????????? ??????????????

CAPÍTULO 7

Conclusiones

????? ?????????????? ?????????????? ?????????????? ?????????????? ??????????????

Bibliografía

- [1] Anders S. G. Andrae and Tomas Edler. On global electricity usage of communication technology: Trends to 2030. *Challenges*, 6(1):117–157, 2015.
- [2] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2014.
- [3] Google Trends. Interés en inteligencia artificial (2020–2025). <https://trends.google.com/trends/explore?date=2020-04-08%202025-04-08&q=inteligencia%20artificial&hl=es>, 2025. Accedido el 08 abril de 2025.
- [4] Glenn Jocher and Jing Qiu. Ultralytics yolo11, 2024.
- [5] Glenn Jocher, Jing Qiu, and Ayush Chaurasia. Ultralytics YOLO, January 2023.
- [6] Salman Khan, Hossein Rahmani, Syed Afaq Ali Shah, and Mohammed Bennamoun. *A Guide to Convolutional Neural Networks for Computer Vision*. Synthesis Lectures on Computer Vision. Springer Cham, 1 edition, 2018.
- [7] Tushar Kumar. R-cnn explained, 2024. Accedido: 14 de abril de 2025.
- [8] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [9] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.
- [10] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. *SSD: Single Shot MultiBox Detector*, page 21–37. Springer International Publishing, 2016.
- [11] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016.
- [12] Liming Xiu. Time moore: Exploiting moore’s law from the perspective of time. *IEEE Solid-State Circuits Magazine*, 11:39–55, 01 2019.

APÉNDICE A

Configuración del sistema

????? ?????????????? ?????????????? ?????????????? ?????????????? ??????????????

A.1 Fase de inicialización

????? ?????????????? ?????????????? ?????????????? ?????????????? ??????????????

A.2 Identificación de dispositivos

????? ?????????????? ?????????????? ?????????????? ?????????????? ??????????????

APÉNDICE B

??? ?????????????? ????

????? ?????????????? ?????????????? ?????????????? ?????????????? ??????????????