Universidad Nacional de Ingeniería

Maestría en Ciencias de la Computación

Entrenamiento con Paralelismo en la Arquitectura MTCNN

Informe del curso:

Visión Computacional Distribuida

Docente:

Dr. Manuel Quispe

Autores:

Michell Alvarez

Silvana Rosas

Lima – Perú

2026

Índice

Lista de tablas

Lista de figuras

1. Introducción

El presente trabajo se basa en el modelo *Multi-task Cascaded Convolutional Neural Networks* (MTCNN), una arquitectura de detección facial que emplea un enfoque jerárquico en tres etapas, procesando la imagen de manera progresiva desde un nivel grueso hasta un nivel fino. Este enfoque permite identificar rostros de forma eficiente y precisa, combinando una alta velocidad de inferencia con una elevada capacidad de refinamiento.

Desde el punto de vista computacional, el sistema se orienta a la optimización de la arquitectura, haciendo uso del paralelismo por lotes durante el entrenamiento, con un *batch size* de 16, lo que permite un mejor aprovechamiento de los recursos computacionales disponibles.

La primera etapa corresponde a la *Proposal Network* (P-Net), cuya función es detectar rápidamente regiones candidatas a contener rostros. Esta red genera posibles cuadros delimitadores y aplica el proceso de supresión de no-máximos (*Non-Maximum Suppression*, NMS) para eliminar detecciones redundantes. Posteriormente, la *Refine Network* (R-Net) se encarga de refinar estas propuestas iniciales, filtrando falsos positivos y mejorando la regresión de las cajas delimitadoras. Finalmente, la *Output Network* (O-Net) realiza el refinamiento final de las detecciones y predice cinco puntos de referencia faciales, correspondientes a los ojos, la nariz y las comisuras de la boca.

A nivel arquitectónico, el modelo reemplaza filtros convolucionales de gran tamaño por filtros de 3×3, lo que reduce el costo computacional y, al mismo tiempo, incrementa la profundidad de la red, mejorando su capacidad descriptiva. Esta estrategia permite alcanzar un mejor rendimiento y menores tiempos de ejecución en comparación con métodos previos.

El entrenamiento del modelo se formula como un problema de aprendizaje multi-tarea, integrando simultáneamente tres objetivos: (i) la clasificación de regiones como rostro o no rostro, (ii) la regresión de los cuadros delimitadores y (iii) la localización de puntos clave faciales. Esta formulación conjunta permite que las redes compartan representaciones internas, contribuyendo a una detección facial más precisa.

2. Trabajos relacionados

El trabajo Zhang et al., 2025 demuestra que el muestreo por lotes consciente del concepto mejora significativamente el proceso de preentrenamiento en modelos de visión–lenguaje. De forma complementaria, el estudio Chen et al., 2023 señala que el paralelismo automático, particularmente en la forma de procesamiento distribuido por lotes, constituye un componente clave para escalar modelos de visión profunda. En base a estos antecedentes, se implementó un esquema de procesamiento de imágenes utilizando paralelismo con cuatro núcleos de ejecución.

Para obtener el tiempo de procesamiento se consideró:

a) **Archivo de configuración externa:**
La configuración del proyecto se estructuró en los siguientes grupos:

- **General:** definición de las rutas de almacenamiento de las imágenes.
- **Detector:** uso del algoritmo *MTCNN* para la detección facial.
- **Aumento:** generación de datos aumentados, considerando la creación de 30 imágenes por cada imagen original, variando tamaño, escala y rotaciones.
- **Logging:** registro detallado de tiempos y acciones en el archivo `procesamiento.log`.

b) **Aumento de la muestra de imágenes:**

- Obtención de los *bounding boxes* de las imágenes originales mediante el detector *MTCNN*.
- Aplicación de técnicas de aumento de datos utilizando la librería *Albumentations*.

c) **Traza de acciones (logger):**
Implementación de un sistema de registro que permite monitorear las acciones ejecutadas y medir los tiempos de procesamiento en cada etapa del flujo.

Los tipos de paralelismo implementados durante el procesamiento fueron los siguientes:

a) **Paralelismo por funciones (*ThreadPool*):**
El modelo de multihilo permite que un proceso ejecute múltiples subprocesos de forma simultánea, compartiendo memoria y recursos. No obstante, su rendimiento se ve limitado por el *Global Interpreter Lock* (GIL) de Python, el cual restringe la ejecución de bytecode a un único subproceso a la vez, reduciendo su eficacia en tareas intensivas en CPU. Para esta implementación se utilizó el módulo `concurrent.futures.ThreadPoolExecutor`.

b) **Paralelismo por lotes (*ProcessPool*):**
El tamaño de los bloques de procesamiento se calculó de manera proporcional al número de núcleos disponibles. Este enfoque presenta el siguiente compromiso:

- **Batches grandes:** mayor rendimiento computacional, a costa de un mayor consumo de memoria.
- **Batches pequeños:** menor uso de memoria, pero incremento en el tiempo total de ejecución.

c) **Paralelismo asíncrono (*async*):**
Modelo de programación no bloqueante en el que las tareas se gestionan mediante un *event loop*. Se utilizó la instrucción `await asyncio.gather(*tareas)`, siendo este enfoque especialmente adecuado para tareas de tipo *I/O-bound*, tales como la lectura de imágenes desde disco, consultas a bases de datos o el envío de resultados a través de la red.

3. Metodología

3.1. Comparativa de tiempos de ejecución de algoritmos de detección facial bajo distintos esquemas de preprocesamiento

A partir de las imágenes del repositorio https://github.com/MVTLab-Uni/FacesDetectionComparison, se realizó una comparativa de los tiempos totales de ejecución de cuatro algoritmos de detección facial: *MTCNN*, *DLIB*, *DNN* y *HAAR*. Dichos algoritmos fueron evaluados en dos escenarios: **Base**, correspondiente a un conjunto de imágenes con preprocesamiento estándar, y **Mejorado**, que considera un preprocesamiento acotado. En todos los casos, se observa una disminución notable en los tiempos de ejecución tras la aplicación de los filtros de procesamiento más efectivos, lo que evidencia una mejora significativa en el rendimiento computacional de los modelos evaluados.

El algoritmo *MTCNN* presenta la mayor reducción en el tiempo de ejecución, disminuyendo de 14.85 s a 3.01 s, seguido por los métodos *DLIB* y *DNN*, los cuales exhiben reducciones de tiempo de magnitud similar. Por su parte, el algoritmo *HAAR*, que ya presentaba el menor tiempo de ejecución en el escenario base, muestra una mejora adicional al reducir su tiempo de 1.74 s a 0.35 s. En base a estos resultados, se evidencia que las técnicas de mejoramiento de imagen acotado, tales como la corrección de brillo, la reducción de ruido y la normalización del contorno facial, no solo favorecen una detección más estable, sino que además optimizan el rendimiento temporal de todos los detectores faciales analizados.

Tabla 1: Comparativa de tiempos de ejecución de algoritmos de detección facial

Algoritmo	**Base (s)**	**Mejorado (s)**
MTCNN	14.85	3.01
DLIB	8.54	1.76
DNN	4.67	0.96
HAAR	1.74	0.35

3.2. Paralelismo en fase de entrenamiento de P-Net

El objetivo principal del entrenamiento de la red *P-Net* (*Proposal Network*) es aprender a generar de manera eficiente regiones candidatas (*proposals*) donde exista probabilidad de presencia de un rostro, operando a múltiples escalas y con un costo computacional reducido. La red *P-Net* no realiza la detección final de rostros, sino que actúa como la primera etapa del proceso, priorizando un alto *recall*, permitiendo la presencia de falsos positivos y garantizando una ejecución rápida, lo cual replica fielmente su rol dentro del *pipeline* original de *MTCNN*.

Para el entrenamiento se utilizó el dataset *WIDER FACE*, específicamente el subconjunto *WIDER_SELECTED*, compuesto por aproximadamente 4 275 imágenes anotadas con información de rutas de imagen, número de rostros y coordenadas de los cuadros delimitadores. Como simplificación metodológica y con fines académicos, se seleccionó únicamente el *bounding box* de mayor área por imagen. Esta decisión se justifica en que *P-Net* aprende una localización aproximada de los rostros, reduce la ambigüedad asociada a la presencia de múltiples caras en una misma imagen y disminuye la complejidad computacional, sin afectar la validez del enfoque cuando el objetivo es replicar el *pipeline* y no alcanzar resultados *state-of-the-art*.

Durante la fase de preparación de los datos, cada imagen se carga conservando su tamaño original y posteriormente se redimensiona a 128×128 píxeles, normalizando los valores de los píxeles al rango $[0, 1]$. Las anotaciones de los cuadros delimitadores, originalmente expresadas en coordenadas absolutas, se transforman a un formato normalizado dividiendo cada componente por el ancho o el alto de la imagen correspondiente. Este procedimiento es fundamental para garantizar la independencia respecto al tamaño de la imagen y asegurar la compatibilidad con el entrenamiento convolucional. Las etiquetas de entrenamiento consideran una salida de clasificación binaria, donde se emplea únicamente la etiqueta

positiva (rostro), y una salida de regresión correspondiente a los desplazamientos normalizados del *bounding box*.

Dado que *P-Net* es una arquitectura completamente convolucional (*fully convolutional*), no produce una única predicción por imagen, sino mapas de salida bidimensionales. En consecuencia, las etiquetas de clasificación se replican en un mapa de tamaño $(59, 59, 1)$ y las etiquetas de regresión en un mapa de tamaño $(59, 59, 4)$, simulando el comportamiento de ventanas deslizantes descrito en el método original.

La arquitectura del modelo implementado replica el diseño propuesto en *MTCNN*. Está compuesta por capas convolucionales de tamaño reducido (3×3 y 1×1), funciones de activación *PReLU* y una única capa de *max pooling*. El modelo no incluye capas completamente conectadas, acepta entradas de tamaño variable y contiene un número reducido de parámetros (aproximadamente 6.6K), lo que garantiza una elevada velocidad de inferencia.

El entrenamiento se realiza bajo un esquema de aprendizaje multitarea, combinando de forma simultánea la función de pérdida de clasificación y la función de pérdida de regresión de los cuadros delimitadores. Para la tarea de clasificación se emplea la pérdida de *Binary Cross-Entropy*, mientras que para la regresión se utiliza la pérdida de *Huber* (*Smooth L1*), debido a su robustez frente a valores atípicos y su mayor estabilidad en comparación con la pérdida L_2. La función de pérdida total asigna un mayor peso a la clasificación que a la regresión, priorizando inicialmente la correcta detección de regiones candidatas y posteriormente el refinamiento de su localización.

La optimización se llevó a cabo utilizando el optimizador *Adam*, con una tasa de aprendizaje de 1×10^{-4}, un tamaño de lote de 16 y un total de 15 épocas de entrenamiento, empleando una GPU *NVIDIA A100* en *Google Colab*. Estos parámetros conservadores permiten un entrenamiento estable y una convergencia adecuada. Finalmente, durante el proceso se almacenó un *checkpoint* por época en formato `.keras`, lo que posibilita la continuidad del entrenamiento, la reutilización del modelo entrenado para la generación de *proposals* y la reproducibilidad completa del experimento.

3.3. Procesamiento de resultados de P-Net para obtener DATASET para RNET

El proceso inicia con la imagen original del dataset *WIDER_ SELECTED*, la cual presenta una resolución arbitraria y cuenta con anotaciones *ground truth* en formato (x, y, w, h). Dichas imágenes son procesadas por una red *P-Net* (*Proposal Network*) previamente entrenada, la cual corresponde a una arquitectura completamente convolucional aplicada sobre una pirámide de escalas, con el objetivo de manejar adecuadamente las variaciones en el tamaño de los rostros. Para cada escala, la *P-Net* genera de forma simultánea un mapa de clasificación (rostro / no-rostro) y un mapa de regresión de cuadros delimitadores.

A partir de estos mapas se realiza la generación de propuestas (*proposals*) por imagen, donde se seleccionan únicamente aquellas posiciones cuyo puntaje de confianza supera un umbral predefinido (*SCORE_ THRESHOLD*). Estas posiciones se transforman a coordenadas correspondientes a la imagen original y se refinan mediante los desplazamientos obtenidos del mapa de regresión, produciendo inicialmente miles de propuestas caracterizadas por sus coordenadas $[x_1, y_1, x_2, y_2]$ y su respectivo puntaje de confianza.

Posteriormente, se aplica el algoritmo de *Supresión No Máxima* (*Non-Maximum Suppression, NMS*), el cual compara todas las propuestas entre sí mediante el cálculo del *Intersection over Union* (IoU). Aquellas propuestas redundantes cuyo IoU supera un umbral preestablecido son descartadas, conservándose únicamente las propuestas con mayor puntuación. Como resultado de esta etapa, se obtiene un conjunto reducido y depurado de propuestas por imagen.

Las propuestas supervivientes generadas por la red *P-Net* son evaluadas frente a las anotaciones reales mediante el cálculo de la *Intersection over Union* (IoU) con respecto al *ground truth*, lo que permite su posterior etiquetado supervisado. En función del valor de la IoU, las propuestas se clasifican en tres categorías: positivas (IoU $\geq 0{,}5$), parciales o *part faces* ($0{,}3 \leq$ IoU $< 0{,}5$) y negativas (IoU $<$

0,3). Esta etapa no implica un proceso de entrenamiento, sino que corresponde exclusivamente a una fase de análisis y construcción del conjunto de datos que será utilizado posteriormente para el entrenamiento de la red *R-Net*.

Con el fin de garantizar robustez y reproducibilidad, se implementó un mecanismo de persistencia segura por imagen, mediante el cual los resultados intermedios se almacenan en archivos `.npz` que contienen las cajas delimitadoras generadas, los puntajes de confianza asociados y la ruta relativa de cada imagen. De manera adicional, el progreso del procesamiento se registra en el archivo `processed.txt`, lo que permite reanudar ejecuciones interrumpidas, evitar reprocesamientos innecesarios y soportar ejecuciones prolongadas en entornos computacionales como *Google Colab*.

Finalmente, el proceso da como resultado un dataset estructurado y balanceado, listo para ser utilizado en el entrenamiento de la red *R-Net*. Dicho conjunto está compuesto por parches de tamaño 24×24 píxeles y sus respectivas etiquetas de clasificación y regresión de cuadros delimitadores, constituyendo la entrada directa para la siguiente etapa de la arquitectura en cascada. Todo el flujo descrito se ilustra de manera esquemática en la **Figura 2**.

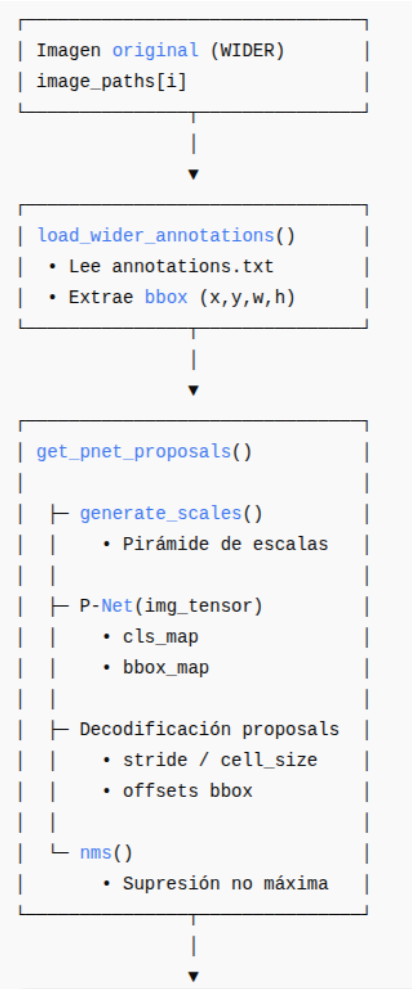


Figura 1: Primera parte del flujo del proceso de generación de propuestas mediante P-Net

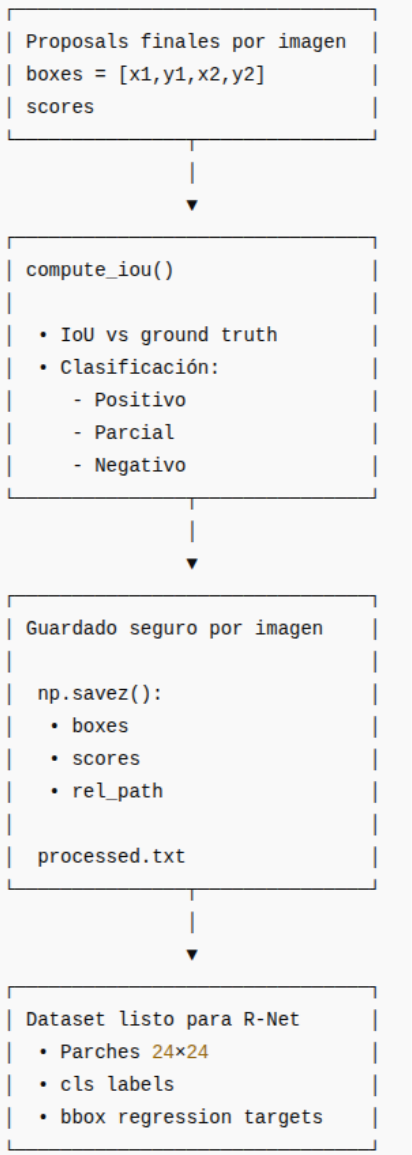


Figura 2: Segunda parte del flujo del proceso de generación de propuestas mediante P-Net

Tabla 2: Parámetros y criterios aplicados en la generación y análisis de propuestas de P-Net

Variable / Criterio	**Valor utilizado**	**Etapa**	**Función principal**	**Resultado obtenido**
`score_thresh`	0.7	P-Net	Filtrado inicial por confianza	Reduce propuestas de muy baja probabilidad
Pirámide de escalas	Multiescala	P-Net	Detección a diferentes tamaños	Generación exhaustiva de candidatos
NMS por escala	0.7	Post-P-Net	Supresión local	Elimina redundancia dentro de cada escala
NMS global	0.7	Post-P-Net	Supresión no máxima final	Reduce solapamientos entre escalas
IoU ≥ 0.5	Positivo	Dataset R-Net	Etiquetado estricto	95 proposals positivas
$0{,}3 \leq$ IoU < 0.5	Part face	Dataset R-Net	Etiquetado intermedio	423 proposals parciales
IoU < 0.3	Negativo	Dataset R-Net	Etiquetado negativo	951 314 proposals negativas
Total de proposals analizadas	–	Análisis	Tamaño del dataset	111 082 proposals
IoU mínimo	–	Análisis	Límite inferior	0.0000
IoU máximo	–	Análisis	Calidad máxima	0.7250
IoU medio	–	Análisis	Indicador global	0.0013

En la **Figura 3** se presenta el resumen de capas de un modelo convolucional ligero, representativo de la red *P-Net* del esquema *MTCNN*, diseñado para la detección inicial de rostros. El modelo recibe como entrada una imagen RGB de tamaño variable y aplica una secuencia de capas convolucionales (*Conv2D*) intercaladas con funciones de activación *PReLU*, lo que permite capturar características espaciales relevantes manteniendo un bajo costo computacional. Asimismo, se incluye una capa de *MaxPooling2D* con el objetivo de reducir la resolución espacial y aumentar la robustez frente a variaciones de escala.

La arquitectura se bifurca finalmente en dos salidas:

- **Rama de clasificación (*cls*):** predice la probabilidad de presencia de un rostro.
- **Rama de regresión (*bbox*):** estima los desplazamientos del cuadro delimitador.

En conjunto, la arquitectura es compacta, con un número reducido de parámetros, eficiente desde el punto de vista computacional y adecuada para su aplicación como primera etapa en un esquema de detección en cascada, donde se prioriza la velocidad de inferencia y el filtrado temprano de regiones no relevantes.

Layer (type)	Output Shape	Param #	Connected to
input_layer_3 (InputLayer)	(None, None, None, 3)	0	-
conv2d_9 (Conv2D)	(None, None, None, 10)	280	input_layer_3[0]…
p_re_lu_9 (PReLU)	(None, None, None, 10)	10	conv2d_9[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, None, None, 10)	0	p_re_lu_9[0][0]
conv2d_10 (Conv2D)	(None, None, None, 16)	1,456	max_pooling2d_3[…
p_re_lu_10 (PReLU)	(None, None, None, 16)	16	conv2d_10[0][0]
conv2d_11 (Conv2D)	(None, None, None, 32)	4,640	p_re_lu_10[0][0]
p_re_lu_11 (PReLU)	(None, None, None, 32)	32	conv2d_11[0][0]
cls (Conv2D)	(None, None, None, 1)	33	p_re_lu_11[0][0]
bbox (Conv2D)	(None, None, None, 4)	132	p_re_lu_11[0][0]

Figura 3: Resumen de capas del P-Net de MTCNN

4. Resultados

Las métricas observadas durante el entrenamiento de la red *P-Net* se resumen a continuación:

- **Exactitud de clasificación (*cls_ accuracy*):** valor cercano a 1,0.
- **Pérdida de regresión (*bbox_ loss*):** comportamiento decreciente y estable a lo largo de las épocas.
- **Convergencia:** rápida, alcanzada aproximadamente a las 10 épocas de entrenamiento.

El comportamiento observado es coherente con lo esperado para la red *P-Net*, el cual se caracteriza por:

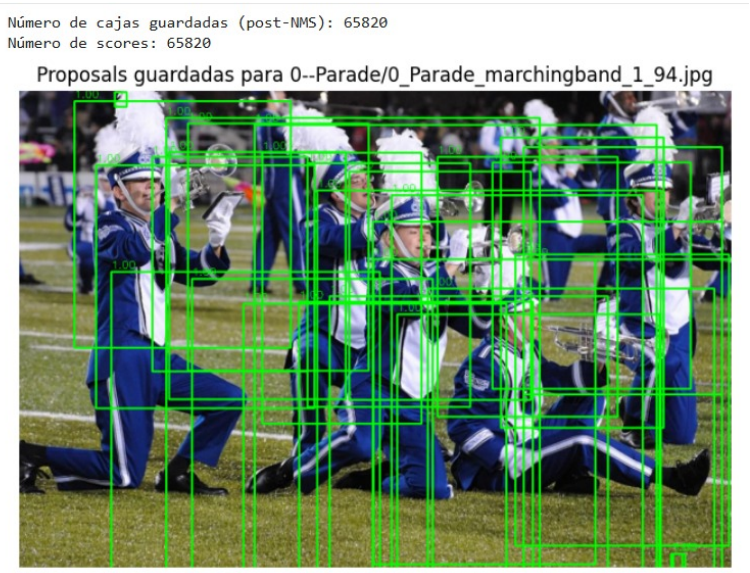
- Alta confianza en la identificación de regiones candidatas.
- Localización aproximada (*coarse localization*) de los rostros.
- Ausencia de sobreajuste crítico durante el proceso de entrenamiento.

Tabla 3: Comparación de tiempos de entrenamiento de la red P-Net con y sin paralelismo

Época	**Sin paralelismo (s)**	**Con paralelismo (s)**
1	222.06	31.71
2	31.22	24.31
3	31.65	24.06
4	31.08	24.17
5	31.17	24.08
6	31.00	24.20
7	31.03	23.96
8	30.96	23.67
9	31.11	23.79
10	30.88	23.84
11	31.10	23.95
12	30.96	23.74
13	31.31	23.77
14	31.04	23.84
15	30.93	24.05
Total	**657.61**	**367.25**

Nota. Los tiempos están expresados en segundos. El paralelismo reduce significativamente el tiempo total de entrenamiento de la red P-Net.

En la **Figura 4** se presenta el resultado de la aplicación de la red *P-Net* sobre una imagen de prueba. La subfigura **(a)** corresponde a la imagen original, mientras que la subfigura **(b)** muestra las propuestas generadas por *P-Net*, representadas mediante múltiples *bounding boxes*. Se observa que la red produce un número elevado de propuestas con alta confianza, cubriendo de forma densa las regiones visualmente relevantes. Este comportamiento es coherente con el diseño de *P-Net*, cuya finalidad es maximizar la sensibilidad y asegurar un alto *recall* en la etapa inicial de la arquitectura en cascada, delegando la eliminación de falsas detecciones a las etapas posteriores del modelo *MTCNN*.



(a) Imagen original

(b) Imagen con *bounding boxes*

Figura 4: Resultados de la red P-Net sobre la imagen de prueba A

(a) Imagen original

(b) Imagen con *bounding boxes*

Figura 5: Resultados de la red P-Net sobre la imagen de prueba B

5. Conclusiones

En este trabajo se implementó la etapa *P-Net* del modelo *MTCNN* y se desarrolló el proceso completo de generación y análisis de propuestas con el objetivo de construir el conjunto de entrenamiento para la red *R-Net*. Mediante el uso de una pirámide multiescala, un umbral de confianza de 0.7 y la aplicación de supresión no máxima tanto por escala como a nivel global (NMS = 0,7), se analizaron un total de 111 082 propuestas generadas a partir de imágenes del dataset *WIDER FACE*, reproduciendo fielmente la metodología descrita en el artículo original.

Las propuestas fueron evaluadas utilizando la métrica *Intersection over Union* (IoU) frente a las anotaciones originales, obteniéndose 95 propuestas positivas (IoU $\geq$ 0,5), 423 propuestas parciales (0,3 $\leq$ IoU $<$ 0,5) y una gran mayoría de 951 314 propuestas negativas (IoU $<$ 0,3). Esta distribución altamente desbalanceada es característica de un enfoque orientado al *recall*. El valor medio de IoU de 0.0013 confirma la exploración exhaustiva del espacio de la imagen, mientras que el IoU máximo alcanzado de 0.7250 demuestra que la red *P-Net* es capaz de generar propuestas correctamente alineadas con los rostros reales.

La implementación de paralelismo en el entrenamiento de la red **P-Net** reduce de manera significativa el tiempo total de ejecución, pasando de 657.61 s sin paralelismo a 367.25 s con paralelismo, lo que representa una disminución aproximada del 44 %. Se observa que, sin paralelismo, la primera época presenta un tiempo considerablemente mayor (222.06 s), evidenciando un alto *overhead* inicial asociado a la carga de datos y a la inicialización del modelo; sin embargo, a partir de la segunda época los tiempos se estabilizan alrededor de 31 s por época. En contraste, el entrenamiento con paralelismo muestra tiempos más bajos y homogéneos, situándose consistentemente en el rango de 23–24 s por época, lo que indica una ejecución más eficiente y estable.

Estos resultados justifican la necesidad de las etapas posteriores del *pipeline MTCNN*, en las cuales la red *R-Net* se encarga de reducir drásticamente el número de candidatos mediante un refinamiento progresivo de las regiones propuestas y una clasificación más precisa.

6. Referencias

Referencias

Chen, X., Li, H., Zhang, Y., et al. (2023). A Survey on Auto-Parallelism of Large-Scale Deep Learning Training. *IEEE Transactions on Parallel and Distributed Systems*. https://doi.org/10.1109/TPDS.2023.10163912

Zhang, Y., Wang, Z., Li, Y., et al. (2025). Concept-Aware Batch Sampling Improves Language-Image Pretraining. *arXiv preprint arXiv:2511.20643*. https://arxiv.org/pdf/2511.20643

Anexos

Anexo 1: Código fuente y scripts

El código fuente utilizado para el desarrollo del sistema, incluyendo scripts de preprocesamiento y entrenamiento, se encuentra disponible en el siguiente repositorio de GitHub:

https://github.com/silvanalorens/MTCNN_Training_Bach.git

Este repositorio contiene los archivos necesarios para la reproducción de los experimentos presentados en este informe.

Anexo 2: Dataset utilizado

El conjunto de datos utilizado en este estudio corresponde al dataset *Selected frontal faces of Wider Dataset*, disponible públicamente en la plataforma Kaggle. Mecionado dataset puede descargarse desde la siguiente ruta oficial:

https://www.kaggle.com/datasets/alirezakay/wider-selected/data

Anexo 3: Evolución de las métricas de entrenamiento de la red P-Net sin el uso de paralelismo

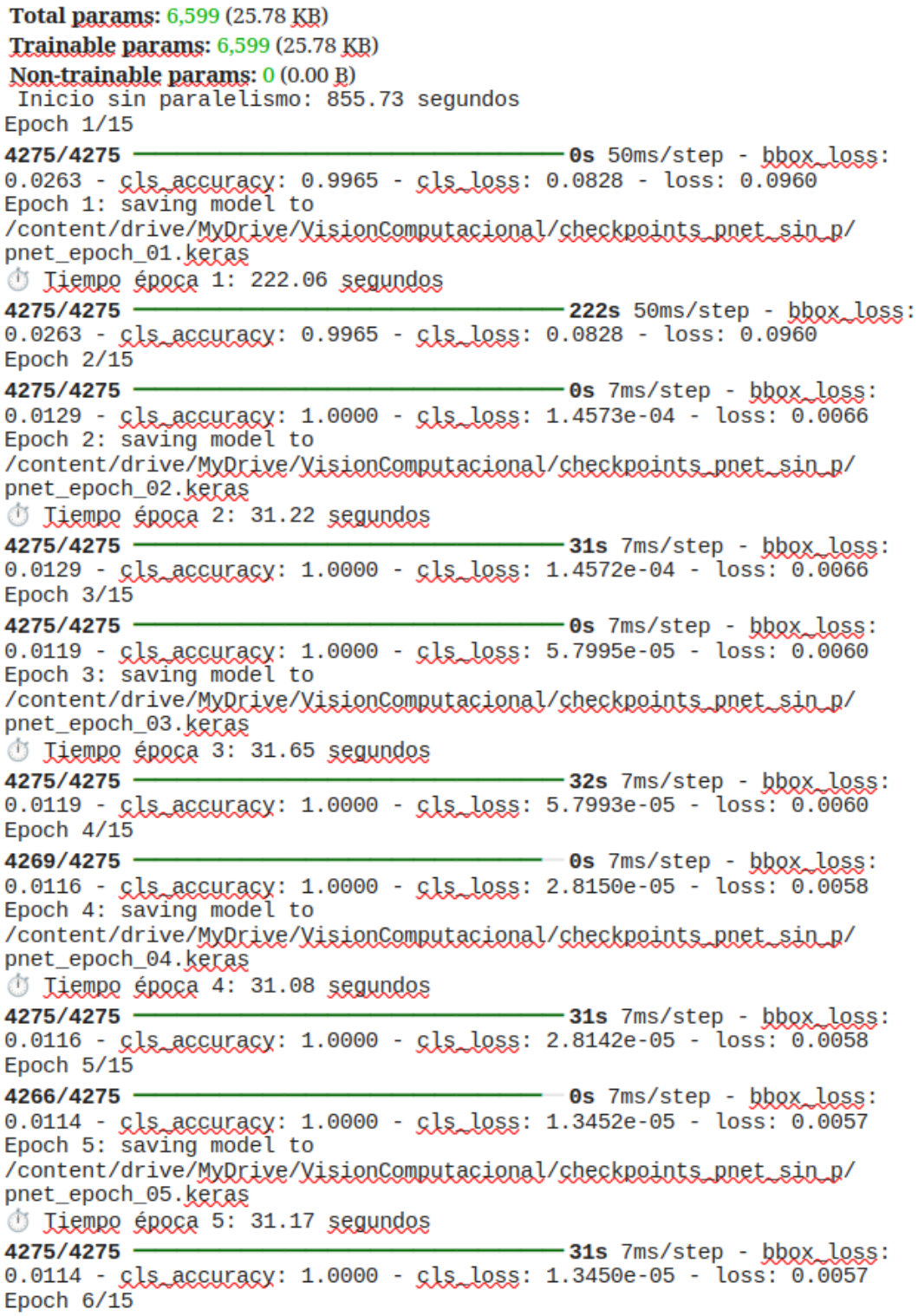


Figura 6: Inicio del entrenamiento sin paralelismo

```
4270/4275 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - bbox_loss:
0.0113 - cls_accuracy: 1.0000 - cls_loss: 9.3299e-06 - loss: 0.0057
Epoch 6: saving model to
/content/drive/MyDrive/VisionComputacional/checkpoints_pnet_sin_p/
pnet_epoch_06.keras
⏱ Tiempo época 6: 31.00 segundos
4275/4275 ━━━━━━━━━━━━━━━━━━━━ 31s 7ms/step - bbox_loss:
0.0113 - cls_accuracy: 1.0000 - cls_loss: 9.3300e-06 - loss: 0.0057
Epoch 7/15
4265/4275 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - bbox_loss:
0.0114 - cls_accuracy: 1.0000 - cls_loss: 7.6231e-06 - loss: 0.0057
Epoch 7: saving model to
/content/drive/MyDrive/VisionComputacional/checkpoints_pnet_sin_p/
pnet_epoch_07.keras
⏱ Tiempo época 7: 31.03 segundos
4275/4275 ━━━━━━━━━━━━━━━━━━━━ 31s 7ms/step - bbox_loss:
0.0114 - cls_accuracy: 1.0000 - cls_loss: 7.6224e-06 - loss: 0.0057
Epoch 8/15
4268/4275 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - bbox_loss:
0.0112 - cls_accuracy: 1.0000 - cls_loss: 6.0764e-06 - loss: 0.0056
Epoch 8: saving model to
/content/drive/MyDrive/VisionComputacional/checkpoints_pnet_sin_p/
pnet_epoch_08.keras
⏱ Tiempo época 8: 30.96 segundos
4275/4275 ━━━━━━━━━━━━━━━━━━━━ 31s 7ms/step - bbox_loss:
0.0112 - cls_accuracy: 1.0000 - cls_loss: 6.0762e-06 - loss: 0.0056
Epoch 9/15
4258/4275 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - bbox_loss:
0.0112 - cls_accuracy: 1.0000 - cls_loss: 4.4316e-06 - loss: 0.0056
Epoch 9: saving model to
/content/drive/MyDrive/VisionComputacional/checkpoints_pnet_sin_p/
pnet_epoch_09.keras
⏱ Tiempo época 9: 31.11 segundos
4275/4275 ━━━━━━━━━━━━━━━━━━━━ 31s 7ms/step - bbox_loss:
0.0112 - cls_accuracy: 1.0000 - cls_loss: 4.4318e-06 - loss: 0.0056
Epoch 10/15
4275/4275 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - bbox_loss:
0.0109 - cls_accuracy: 1.0000 - cls_loss: 3.1006e-06 - loss: 0.0055
Epoch 10: saving model to
/content/drive/MyDrive/VisionComputacional/checkpoints_pnet_sin_p/
pnet_epoch_10.keras
⏱ Tiempo época 10: 30.88 segundos
4275/4275 ━━━━━━━━━━━━━━━━━━━━ 31s 7ms/step - bbox_loss:
0.0109 - cls_accuracy: 1.0000 - cls_loss: 3.1006e-06 - loss: 0.0055
Epoch 11/15
4259/4275 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - bbox_loss:
0.0110 - cls_accuracy: 1.0000 - cls_loss: 2.0891e-06 - loss: 0.0055
Epoch 11: saving model to
/content/drive/MyDrive/VisionComputacional/checkpoints_pnet_sin_p/
pnet_epoch_11.keras
⏱ Tiempo época 11: 31.10 segundos
4275/4275 ━━━━━━━━━━━━━━━━━━━━ 31s 7ms/step - bbox_loss:
0.0110 - cls_accuracy: 1.0000 - cls_loss: 2.0896e-06 - loss: 0.0055
```

Figura 7: Estabilización de las métricas por época sin paralelismo

```
Epoch 12/15
4270/4275 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - bbox_loss:
0.0110 - cls_accuracy: 1.0000 - cls_loss: 1.4574e-06 - loss: 0.0055
Epoch 12: saving model to
/content/drive/MyDrive/VisionComputacional/checkpoints_pnet_sin_p/
pnet_epoch_12.keras
⏱ Tiempo época 12: 30.96 segundos
4275/4275 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 31s 7ms/step - bbox_loss:
0.0110 - cls_accuracy: 1.0000 - cls_loss: 1.4576e-06 - loss: 0.0055
Epoch 13/15
4272/4275 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - bbox_loss:
0.0111 - cls_accuracy: 1.0000 - cls_loss: 1.0513e-06 - loss: 0.0056
Epoch 13: saving model to
/content/drive/MyDrive/VisionComputacional/checkpoints_pnet_sin_p/
pnet_epoch_13.keras
⏱ Tiempo época 13: 31.31 segundos
4275/4275 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 31s 7ms/step - bbox_loss:
0.0111 - cls_accuracy: 1.0000 - cls_loss: 1.0514e-06 - loss: 0.0056
Epoch 14/15
4264/4275 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - bbox_loss:
0.0111 - cls_accuracy: 1.0000 - cls_loss: 8.6530e-07 - loss: 0.0055
Epoch 14: saving model to
/content/drive/MyDrive/VisionComputacional/checkpoints_pnet_sin_p/
pnet_epoch_14.keras
⏱ Tiempo época 14: 31.04 segundos
4275/4275 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 31s 7ms/step - bbox_loss:
0.0111 - cls_accuracy: 1.0000 - cls_loss: 8.6541e-07 - loss: 0.0055
Epoch 15/15
4261/4275 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - bbox_loss:
0.0111 - cls_accuracy: 1.0000 - cls_loss: 7.1614e-07 - loss: 0.0056
Epoch 15: saving model to
/content/drive/MyDrive/VisionComputacional/checkpoints_pnet_sin_p/
pnet_epoch_15.keras
⏱ Tiempo época 15: 30.93 segundos
4275/4275 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 31s 7ms/step - bbox_loss:
0.0111 - cls_accuracy: 1.0000 - cls_loss: 7.1643e-07 - loss: 0.0056
 Fin sin paralelismo: 855.73 segundos
Tiempos por época sin paralelismo: [222.06001167900013,
31.221922790000008, 31.65320944999985, 31.0843933000001,
31.17371164699989, 31.004655357000047, 31.02982041900009,
30.964421588999812, 31.110487221000085, 30.87820049900006,
31.100216706000083, 30.961538748000066, 31.30548339799998,
31.035560127000053, 30.93251936200022]
 Tiempo total entrenamiento sin paralelismo: 657.61 segundos
```

Figura 8: Comportamiento final del entrenamiento sin paralelismo

Anexo 4: Evolución de las métricas de entrenamiento de la red P-Net incluyendo el uso de paralelismo

```
Total params: 6,599 (25.78 KB)
Trainable params: 6,599 (25.78 KB)
Non-trainable params: 0 (0.00 B)
 Inicio con paralelismo: 1795.68 segundos
Epoch 1/15
268/268 ━━━━━━━━━━━━━━━━━━━━ 0s 99ms/step - bbox_loss:
0.1382 - cls_accuracy: 0.9997 - cls_loss: 0.2861 - loss: 0.3552
Epoch 1: saving model to
/content/drive/MyDrive/VisionComputacional/checkpoints_pnet_con_p/
pnet_epoch_01.keras
⏱ Tiempo época 1: 31.71 segundos
268/268 ━━━━━━━━━━━━━━━━━━━━ 32s 100ms/step - bbox_loss:
0.1380 - cls_accuracy: 0.9997 - cls_loss: 0.2856 - loss: 0.3546
Epoch 2/15
266/268 ━━━━━━━━━━━━━━━━━━━━ 0s 91ms/step - bbox_loss:
0.0200 - cls_accuracy: 1.0000 - cls_loss: 0.0163 - loss: 0.0263
Epoch 2: saving model to
/content/drive/MyDrive/VisionComputacional/checkpoints_pnet_con_p/
pnet_epoch_02.keras
⏱ Tiempo época 2: 24.31 segundos
268/268 ━━━━━━━━━━━━━━━━━━━━ 24s 90ms/step - bbox_loss:
0.0200 - cls_accuracy: 1.0000 - cls_loss: 0.0162 - loss: 0.0262
Epoch 3/15
267/268 ━━━━━━━━━━━━━━━━━━━━ 0s 90ms/step - bbox_loss:
0.0178 - cls_accuracy: 1.0000 - cls_loss: 0.0044 - loss: 0.0133
Epoch 3: saving model to
/content/drive/MyDrive/VisionComputacional/checkpoints_pnet_con_p/
pnet_epoch_03.keras
⏱ Tiempo época 3: 24.06 segundos
268/268 ━━━━━━━━━━━━━━━━━━━━ 24s 90ms/step - bbox_loss:
0.0178 - cls_accuracy: 1.0000 - cls_loss: 0.0044 - loss: 0.0133
Epoch 4/15
267/268 ━━━━━━━━━━━━━━━━━━━━ 0s 90ms/step - bbox_loss:
0.0156 - cls_accuracy: 1.0000 - cls_loss: 0.0020 - loss: 0.0098
Epoch 4: saving model to
/content/drive/MyDrive/VisionComputacional/checkpoints_pnet_con_p/
pnet_epoch_04.keras
⏱ Tiempo época 4: 24.17 segundos
268/268 ━━━━━━━━━━━━━━━━━━━━ 24s 90ms/step - bbox_loss:
0.0156 - cls_accuracy: 1.0000 - cls_loss: 0.0020 - loss: 0.0098
Epoch 5/15
267/268 ━━━━━━━━━━━━━━━━━━━━ 0s 90ms/step - bbox_loss:
0.0143 - cls_accuracy: 1.0000 - cls_loss: 0.0010 - loss: 0.0082
Epoch 5: saving model to
/content/drive/MyDrive/VisionComputacional/checkpoints_pnet_con_p/
pnet_epoch_05.keras
⏱ Tiempo época 5: 24.08 segundos
268/268 ━━━━━━━━━━━━━━━━━━━━ 24s 90ms/step - bbox_loss:
0.0143 - cls_accuracy: 1.0000 - cls_loss: 0.0010 - loss: 0.0082
Epoch 6/15
```

Figura 9: Inicio del entrenamiento con paralelismo

```
266/268 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 90ms/step - bbox_loss:
0.0137 - cls_accuracy: 1.0000 - cls_loss: 6.1070e-04 - loss: 0.0075
Epoch 6: saving model to
/content/drive/MyDrive/VisionComputacional/checkpoints_pnet_con_p/
pnet_epoch_06.keras
⏱ Tiempo época 6: 24.20 segundos
268/268 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24s 90ms/step - bbox_loss:
0.0137 - cls_accuracy: 1.0000 - cls_loss: 6.0951e-04 - loss: 0.0075
Epoch 7/15
268/268 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 89ms/step - bbox_loss:
0.0132 - cls_accuracy: 1.0000 - cls_loss: 3.9133e-04 - loss: 0.0070
Epoch 7: saving model to
/content/drive/MyDrive/VisionComputacional/checkpoints_pnet_con_p/
pnet_epoch_07.keras
⏱ Tiempo época 7: 23.96 segundos
268/268 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24s 89ms/step - bbox_loss:
0.0132 - cls_accuracy: 1.0000 - cls_loss: 3.9115e-04 - loss: 0.0070
Epoch 8/15
266/268 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 88ms/step - bbox_loss:
0.0130 - cls_accuracy: 1.0000 - cls_loss: 2.8977e-04 - loss: 0.0068
Epoch 8: saving model to
/content/drive/MyDrive/VisionComputacional/checkpoints_pnet_con_p/
pnet_epoch_08.keras
⏱ Tiempo época 8: 23.67 segundos
268/268 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24s 88ms/step - bbox_loss:
0.0130 - cls_accuracy: 1.0000 - cls_loss: 2.8932e-04 - loss: 0.0068
Epoch 9/15
268/268 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 88ms/step - bbox_loss:
0.0128 - cls_accuracy: 1.0000 - cls_loss: 2.1719e-04 - loss: 0.0066
Epoch 9: saving model to
/content/drive/MyDrive/VisionComputacional/checkpoints_pnet_con_p/
pnet_epoch_09.keras
⏱ Tiempo época 9: 23.79 segundos
268/268 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24s 88ms/step - bbox_loss:
0.0128 - cls_accuracy: 1.0000 - cls_loss: 2.1709e-04 - loss: 0.0066
Epoch 10/15
268/268 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 88ms/step - bbox_loss:
0.0124 - cls_accuracy: 1.0000 - cls_loss: 1.7309e-04 - loss: 0.0064
Epoch 10: saving model to
/content/drive/MyDrive/VisionComputacional/checkpoints_pnet_con_p/
pnet_epoch_10.keras
⏱ Tiempo época 10: 23.84 segundos
268/268 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24s 89ms/step - bbox_loss:
0.0124 - cls_accuracy: 1.0000 - cls_loss: 1.7301e-04 - loss: 0.0064
Epoch 11/15
266/268 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 89ms/step - bbox_loss:
0.0125 - cls_accuracy: 1.0000 - cls_loss: 1.4038e-04 - loss: 0.0064
Epoch 11: saving model to
/content/drive/MyDrive/VisionComputacional/checkpoints_pnet_con_p/
pnet_epoch_11.keras
⏱ Tiempo época 11: 23.95 segundos
268/268 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24s 89ms/step - bbox_loss:
0.0125 - cls_accuracy: 1.0000 - cls_loss: 1.4017e-04 - loss: 0.0064
```

Figura 10: Estabilización de las métricas por época con paralelismo

```
Epoch 12/15
267/268 ━━━━━━━━━━━━━━━━━━━━ 0s 88ms/step - bbox_loss:
0.0123 - cls_accuracy: 1.0000 - cls_loss: 1.1228e-04 - loss: 0.0063
Epoch 12: saving model to
/content/drive/MyDrive/VisionComputacional/checkpoints_pnet_con_p/
pnet_epoch_12.keras
⏱ Tiempo época 12: 23.74 segundos
268/268 ━━━━━━━━━━━━━━━━━━━━ 24s 88ms/step - bbox_loss:
0.0123 - cls_accuracy: 1.0000 - cls_loss: 1.1221e-04 - loss: 0.0063
Epoch 13/15
267/268 ━━━━━━━━━━━━━━━━━━━━ 0s 88ms/step - bbox_loss:
0.0121 - cls_accuracy: 1.0000 - cls_loss: 9.9268e-05 - loss: 0.0062
Epoch 13: saving model to
/content/drive/MyDrive/VisionComputacional/checkpoints_pnet_con_p/
pnet_epoch_13.keras
⏱ Tiempo época 13: 23.77 segundos
268/268 ━━━━━━━━━━━━━━━━━━━━ 24s 88ms/step - bbox_loss:
0.0121 - cls_accuracy: 1.0000 - cls_loss: 9.9196e-05 - loss: 0.0062
Epoch 14/15
267/268 ━━━━━━━━━━━━━━━━━━━━ 0s 89ms/step - bbox_loss:
0.0123 - cls_accuracy: 1.0000 - cls_loss: 8.6738e-05 - loss: 0.0063
Epoch 14: saving model to
/content/drive/MyDrive/VisionComputacional/checkpoints_pnet_con_p/
pnet_epoch_14.keras
⏱ Tiempo época 14: 23.84 segundos
268/268 ━━━━━━━━━━━━━━━━━━━━ 24s 89ms/step - bbox_loss:
0.0123 - cls_accuracy: 1.0000 - cls_loss: 8.6677e-05 - loss: 0.0063
Epoch 15/15
268/268 ━━━━━━━━━━━━━━━━━━━━ 0s 89ms/step - bbox_loss:
0.0120 - cls_accuracy: 1.0000 - cls_loss: 7.8428e-05 - loss: 0.0061
Epoch 15: saving model to
/content/drive/MyDrive/VisionComputacional/checkpoints_pnet_con_p/
pnet_epoch_15.keras
⏱ Tiempo época 15: 24.05 segundos
268/268 ━━━━━━━━━━━━━━━━━━━━ 24s 89ms/step - bbox_loss:
0.0120 - cls_accuracy: 1.0000 - cls_loss: 7.8393e-05 - loss: 0.0061
 fin con paralelismo: 2162.93 segundos
Tiempos por época con paralelismo: [31.71295191099989,
24.312938671999973, 24.0590494139999, 24.16821197199988,
24.082411072000014, 24.19581231699999, 23.962002727000026,
23.674122549999993, 23.789074510999853, 23.842976671000088,
23.94932153800005, 23.744464105000134, 23.772425569999996,
23.84065473999999, 24.05438274800008]
 Tiempo total entrenamiento: 367.25 segundos
```

Figura 11: Comportamiento final del entrenamiento con paralelismo