Universidad Internacional de La Rioja
Escuela Superior de Ingeniería y Tecnología

Máster Universitario en Inteligencia artificial

Optimización de Hiperparámetros OCR con Ray Tune para Documentos Académicos en Español

Trabajo fin de estudio presentado por:	Sergio Jiménez Jiménez
Tipo de trabajo:	Desarrollo Software
Director/a:	Javier Rodrigo Villazón Terrazas
Fecha:	06.10.2025

Resumen

El presente Trabajo Fin de Máster aborda la optimización de sistemas de Reconocimiento Óptico de Caracteres (OCR) basados en inteligencia artificial para documentos en español, específicamente en un entorno con recursos computacionales limitados donde el fine-tuning de modelos no es viable. El objetivo principal es identificar la configuración óptima de hiperparámetros que maximice la precisión del reconocimiento de texto sin requerir entrenamiento adicional de los modelos. Se realizó un estudio comparativo de tres soluciones OCR de código abierto: EasyOCR, PaddleOCR (PP-OCRv5) y DocTR, evaluando su rendimiento mediante las métricas estándar CER (Character Error Rate) y WER (Word Error Rate) sobre un corpus de documentos académicos en español. Tras identificar PaddleOCR como la solución más prometedora, se procedió a una optimización sistemática de hiperparámetros utilizando Ray Tune con el algoritmo de búsqueda Optuna, ejecutando 64 configuraciones diferentes. Los resultados demuestran que la optimización de hiperparámetros logró una mejora significativa del rendimiento: el CER se redujo de 7.78% a 1.49% (mejora del 80.9% en reducción de errores), alcanzando una precisión de caracteres del 98.51%. El hallazgo más relevante fue que el parámetro `textline_orientation` (clasificación de orientación de línea de texto) tiene un impacto crítico, reduciendo el CER en un 69.7% cuando está habilitado. Adicionalmente, se identificó que el umbral de detección de píxeles (`text_det_thresh`) presenta una correlación negativa fuerte (-0.52) con el error, siendo el parámetro continuo más influyente. Este trabajo demuestra que es posible obtener mejoras sustanciales en sistemas OCR mediante optimización de hiperparámetros, ofreciendo una alternativa práctica al fine-tuning cuando los recursos computacionales son limitados.

Palabras clave: OCR, Reconocimiento Óptico de Caracteres, PaddleOCR, Optimización de Hiperparámetros, Ray Tune, Procesamiento de Documentos, Inteligencia Artificial

Abstract

This Master's Thesis addresses the optimization of Artificial Intelligence-based Optical Character Recognition (OCR) systems for Spanish documents, specifically in a resource-

constrained environment where model fine-tuning is not feasible. The main objective is to identify the optimal hyperparameter configuration that maximizes text recognition accuracy without requiring additional model training. A comparative study of three open-source OCR solutions was conducted: EasyOCR, PaddleOCR (PP-OCRv5), and DocTR, evaluating their performance using standard CER (Character Error Rate) and WER (Word Error Rate) metrics on a corpus of academic documents in Spanish. After identifying PaddleOCR as the most promising solution, systematic hyperparameter optimization was performed using Ray Tune with the Optuna search algorithm, executing 64 different configurations. Results demonstrate that hyperparameter optimization achieved significant performance improvement: CER was reduced from 7.78% to 1.49% (80.9% error reduction), achieving 98.51% character accuracy. The most relevant finding was that the `textline_orientation` parameter (text line orientation classification) has a critical impact, reducing CER by 69.7% when enabled. Additionally, the pixel detection threshold (`text_det_thresh`) was found to have a strong negative correlation (-0.52) with error, being the most influential continuous parameter. This work demonstrates that substantial improvements in OCR systems can be obtained through hyperparameter optimization, offering a practical alternative to fine-tuning when computational resources are limited.

Keywords: OCR, Optical Character Recognition, PaddleOCR, Hyperparameter Optimization, Ray Tune, Document Processing, Artificial Intelligence

Índice de contenidos

Índice de figuras

Índice de tablas

1. Introducción

Este capítulo presenta la motivación del trabajo, identificando el problema a resolver y justificando su relevancia. Se plantea la pregunta de investigación central y se describe la estructura del documento.

1.1. Motivación

El Reconocimiento Óptico de Caracteres (OCR) es una tecnología fundamental en la era de la digitalización documental. Su capacidad para convertir imágenes de texto en datos editables y procesables ha transformado sectores como la administración pública, el ámbito legal, la banca y la educación. Sin embargo, a pesar de los avances significativos impulsados por el aprendizaje profundo, la implementación práctica de sistemas OCR de alta precisión sigue presentando desafíos considerables.

El procesamiento de documentos en español presenta particularidades que complican el reconocimiento automático de texto. Los caracteres especiales (ñ, acentos), las variaciones tipográficas en documentos académicos y administrativos, y la presencia de elementos gráficos como tablas, encabezados y marcas de agua generan errores que pueden propagarse en aplicaciones downstream como la extracción de entidades nombradas o el análisis semántico.

Los modelos OCR basados en redes neuronales profundas, como los empleados en PaddleOCR, EasyOCR o DocTR, ofrecen un rendimiento impresionante en benchmarks estándar. No obstante, su adaptación a dominios específicos típicamente requiere fine-tuning con datos etiquetados del dominio objetivo y recursos computacionales significativos (GPUs de alta capacidad). Esta barrera técnica y económica excluye a muchos investigadores y organizaciones de beneficiarse plenamente de estas tecnologías.

La presente investigación surge de una necesidad práctica: optimizar un sistema OCR para documentos académicos en español sin disponer de recursos GPU para realizar fine-tuning. Esta restricción, lejos de ser una limitación excepcional, representa la realidad de muchos entornos académicos y empresariales donde el acceso a infraestructura de cómputo avanzada es limitado.

1.2. Planteamiento del trabajo

El problema central que aborda este trabajo puede formularse de la siguiente manera:

> *¿Es posible mejorar significativamente el rendimiento de modelos OCR preentrenados para documentos en español mediante la optimización sistemática de hiperparámetros, sin requerir fine-tuning ni recursos GPU?*

Este planteamiento se descompone en las siguientes cuestiones específicas:

1. **Selección de modelo base**: ¿Cuál de las soluciones OCR de código abierto disponibles (EasyOCR, PaddleOCR, DocTR) ofrece el mejor rendimiento base para documentos en español?
1. **Impacto de hiperparámetros**: ¿Qué hiperparámetros del pipeline OCR tienen mayor influencia en las métricas de error (CER, WER)?
1. **Optimización automatizada**: ¿Puede un proceso de búsqueda automatizada de hiperparámetros (mediante Ray Tune/Optuna) encontrar configuraciones que superen significativamente los valores por defecto?
1. **Viabilidad práctica**: ¿Son los tiempos de inferencia y los recursos requeridos compatibles con un despliegue en entornos con recursos limitados?

La relevancia de este problema radica en su aplicabilidad inmediata. Una metodología reproducible para optimizar OCR sin fine-tuning beneficiaría a:

- Investigadores que procesan grandes volúmenes de documentos académicos
- Instituciones educativas que digitalizan archivos históricos
- Pequeñas y medianas empresas que automatizan flujos documentales
- Desarrolladores que integran OCR en aplicaciones con restricciones de recursos

1.3. Estructura del trabajo

El presente documento se organiza en los siguientes capítulos:

Capítulo 2 - Contexto y Estado del Arte: Se presenta una revisión de las tecnologías OCR basadas en aprendizaje profundo, incluyendo las arquitecturas de detección y reconocimiento de texto, así como los trabajos previos en optimización de estos sistemas.

Capítulo 3 - Objetivos y Metodología: Se definen los objetivos SMART del trabajo y se describe la metodología experimental seguida, incluyendo la preparación del dataset, las métricas de evaluación y el proceso de optimización con Ray Tune.

Capítulo 4 - Desarrollo Específico de la Contribución: Este capítulo presenta el desarrollo completo del estudio comparativo y la optimización de hiperparámetros de sistemas OCR, estructurado en tres secciones: (4.1) planteamiento de la comparativa con la evaluación de EasyOCR, PaddleOCR y DocTR; (4.2) desarrollo de la comparativa con la optimización de hiperparámetros mediante Ray Tune; y (4.3) discusión y análisis de resultados.

Capítulo 5 - Conclusiones y Trabajo Futuro: Se resumen las contribuciones del trabajo, se discute el grado de cumplimiento de los objetivos y se proponen líneas de trabajo futuro.

Anexos: Se incluye el enlace al repositorio de código fuente y datos, así como tablas completas de resultados experimentales.

2. Contexto y estado del arte

Este capítulo presenta el marco teórico y tecnológico en el que se desarrolla el presente trabajo. Se revisan los fundamentos del Reconocimiento Óptico de Caracteres (OCR), la evolución de las técnicas basadas en aprendizaje profundo, las principales soluciones de código abierto disponibles y los trabajos previos relacionados con la optimización de sistemas OCR.

2.1.Contexto del problema

2.1.1. Definición y Evolución Histórica del OCR

El Reconocimiento Óptico de Caracteres (OCR) es el proceso de conversión de imágenes de texto manuscrito, mecanografiado o impreso en texto codificado digitalmente. La tecnología OCR ha evolucionado significativamente desde sus orígenes en la década de 1950:

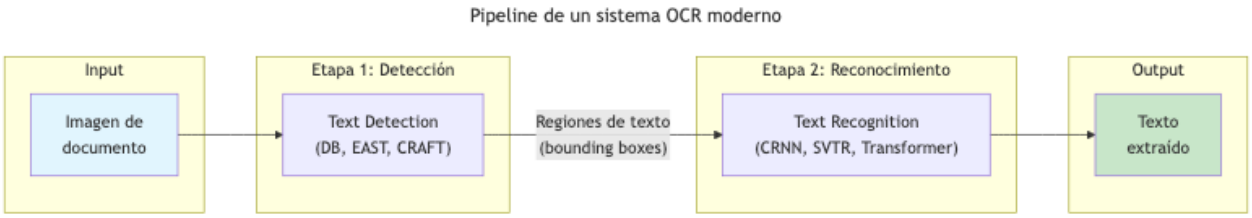
- **Primera generación (1950-1970)**: Sistemas basados en plantillas que requerían fuentes específicas.
- **Segunda generación (1970-1990)**: Introducción de técnicas de extracción de características y clasificadores estadísticos.

- **Tercera generación (1990-2010)**: Modelos basados en Redes Neuronales Artificiales y Modelos Ocultos de Markov (HMM).
- **Cuarta generación (2010-presente)**: Arquitecturas de aprendizaje profundo que dominan el estado del arte.

2.1.2. Pipeline Moderno de OCR

Los sistemas OCR modernos siguen típicamente un pipeline de dos etapas:

Figura 1. ***Pipeline de un sistema OCR moderno***



Fuente: Elaboración propia.

1. **Detección de texto (Text Detection)**: Localización de regiones que contienen texto en la imagen. Las arquitecturas más utilizadas incluyen:

- EAST (Efficient and Accurate Scene Text Detector) - CRAFT (Character Region Awareness for Text Detection) - DB (Differentiable Binarization)

1. **Reconocimiento de texto (Text Recognition)**: Transcripción del contenido textual de las regiones detectadas. Las arquitecturas predominantes son:

- CRNN (Convolutional Recurrent Neural Network) con CTC loss - Arquitecturas encoder-decoder con atención - Transformers (ViTSTR, TrOCR)

2.1.3. Métricas de Evaluación

Las métricas estándar para evaluar sistemas OCR son:

Character Error Rate (CER): Se calcula como CER = (S + D + I) / N, donde S = sustituciones, D = eliminaciones, I = inserciones, N = caracteres de referencia.

Word Error Rate (WER): Se calcula de forma análoga pero a nivel de palabras en lugar de caracteres.

Un CER del 1% significa que 1 de cada 100 caracteres es erróneo. Para aplicaciones críticas como extracción de datos financieros o médicos, se requieren CER inferiores al 1%.

2.1.4. Particularidades del OCR para el Idioma Español

El español presenta características específicas que impactan el OCR:

- **Caracteres especiales**: ñ, á, é, í, ó, ú, ü, ¿, ¡
- **Diacríticos**: Los acentos pueden confundirse con ruido o artefactos
- **Longitud de palabras**: Palabras generalmente más largas que en inglés
- **Puntuación**: Signos de interrogación y exclamación invertidos

2.2. Estado del arte

2.2.1. Soluciones OCR de Código Abierto

2.2.1.1. EasyOCR

EasyOCR es una biblioteca de OCR desarrollada por Jaided AI (2020) que soporta más de 80 idiomas. Sus características principales incluyen:

- **Arquitectura**: Detector CRAFT + Reconocedor CRNN/Transformer
- **Fortalezas**: Facilidad de uso, soporte multilingüe amplio, bajo consumo de memoria
- **Limitaciones**: Menor precisión en documentos complejos, opciones de configuración limitadas
- **Caso de uso ideal**: Prototipado rápido y aplicaciones con restricciones de memoria

2.2.1.2. PaddleOCR

PaddleOCR es el sistema OCR desarrollado por Baidu como parte del ecosistema PaddlePaddle (2024). La versión PP-OCRv5, utilizada en este trabajo, representa el estado del arte en OCR industrial:

- **Arquitectura**:

- Detector: DB (Differentiable Binarization) con backbone ResNet (Liao et al., 2020) - Reconocedor: SVTR (Scene-Text Visual Transformer Recognition) - Clasificador de orientación opcional

- **Hiperparámetros configurables**:

Tabla 1. ***Hiperparámetros configurables de PaddleOCR.***

Parámetro	Descripción	Valor por defecto
`text_det_thresh`	Umbral de detección de píxeles	0.3
`text_det_box_thresh`	Umbral de caja de detección	0.6
`text_det_unclip_ratio`	Coeficiente de expansión	1.5
`text_rec_score_thresh`	Umbral de confianza de reconocimiento	0.5
`use_textline_orientation`	Clasificación de orientación	False
`use_doc_orientation_classify`	Clasificación de orientación de documento	False
`use_doc_unwarping`	Corrección de deformación	False

Fuente: Elaboración propia.

- **Fortalezas**: Alta precisión, pipeline altamente configurable, modelos específicos para servidor
- **Limitaciones**: Mayor complejidad de configuración, dependencia del framework PaddlePaddle

2.2.1.3. DocTR

DocTR (Document Text Recognition) es una biblioteca desarrollada por Mindee (2021) orientada a la investigación:

- **Arquitectura**:

- Detectores: DB, LinkNet - Reconocedores: CRNN, SAR, ViTSTR

- **Fortalezas**: API limpia, orientación académica, salida estructurada de alto nivel
- **Limitaciones**: Menor rendimiento en español comparado con PaddleOCR

2.2.1.4. Comparativa de Arquitecturas

Tabla 2. ***Comparativa de soluciones OCR de código abierto.***

Modelo	Tipo	Componentes	Fortalezas Clave
EasyOCR	End-to-end (det + rec)	CRAFT + CRNN/Transformer	Ligero, fácil de usar, multilingüe
PaddleOCR	End-to-end (det + rec + cls)	DB + SVTR/CRNN	Soporte multilingüe robusto, configurable
DocTR	End-to-end (det + rec)	DB/LinkNet + CRNN/SAR/ViTSTR	Orientado a investigación, API limpia

Fuente: Elaboración propia.

2.2.2. Optimización de Hiperparámetros

2.2.2.1. Fundamentos

La optimización de hiperparámetros (HPO) busca encontrar la configuración de parámetros que maximiza (o minimiza) una métrica objetivo (Feurer & Hutter, 2019). A diferencia de los parámetros del modelo (pesos), los hiperparámetros no se aprenden durante el entrenamiento.

Los métodos de HPO incluyen:

- **Grid Search**: Búsqueda exhaustiva en una rejilla predefinida
- **Random Search**: Muestreo aleatorio del espacio de búsqueda (Bergstra & Bengio, 2012)
- **Bayesian Optimization**: Modelado probabilístico de la función objetivo (Bergstra et al., 2011)
- **Algoritmos evolutivos**: Optimización inspirada en evolución biológica

2.2.2.2. Ray Tune y Optuna

Ray Tune es un framework de optimización de hiperparámetros escalable (Liaw et al., 2018) que permite:

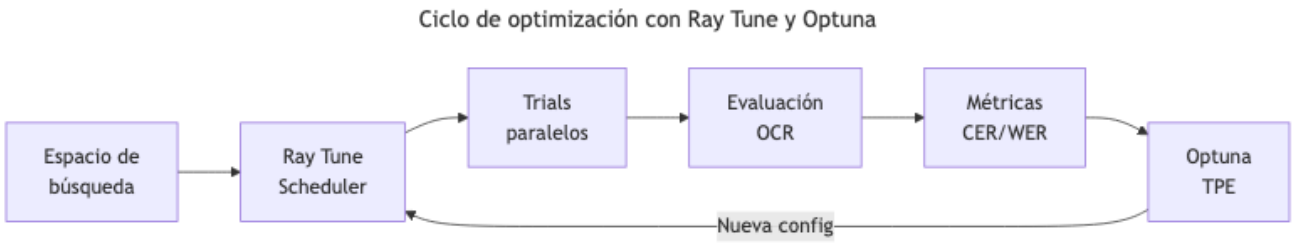
- Ejecución paralela de experimentos
- Early stopping de configuraciones poco prometedoras
- Integración con múltiples algoritmos de búsqueda

Optuna es una biblioteca de optimización bayesiana (Akiba et al., 2019) que implementa:

- Tree-structured Parzen Estimator (TPE)
- Pruning de trials no prometedores
- Visualización de resultados

La combinación Ray Tune + Optuna permite búsquedas eficientes en espacios de alta dimensionalidad.

Figura 2. ***Ciclo de optimización con Ray Tune y Optuna***



Fuente: Elaboración propia.

2.2.2.3. HPO en Sistemas OCR

La aplicación de HPO a sistemas OCR ha sido explorada principalmente en el contexto de:

1. **Preprocesamiento de imagen**: Optimización de parámetros de binarización, filtrado y escalado (Liang et al., 2005)
1. **Arquitecturas de detección**: Ajuste de umbrales de confianza y NMS (Non-Maximum Suppression)
1. **Post-procesamiento**: Optimización de corrección ortográfica y modelos de lenguaje

Sin embargo, existe un vacío en la literatura respecto a la optimización sistemática de los hiperparámetros de inferencia en pipelines OCR modernos como PaddleOCR, especialmente para idiomas diferentes del inglés y chino.

2.2.3. Datasets y Benchmarks para Español

Los principales recursos para evaluación de OCR en español incluyen:

- **FUNSD-ES**: Versión en español del dataset de formularios
- **MLT (ICDAR)**: Multi-Language Text dataset con muestras en español
- **Documentos académicos**: Utilizados en este trabajo (instrucciones TFE de UNIR)

Los trabajos previos en OCR para español se han centrado principalmente en:

1. Digitalización de archivos históricos (manuscritos coloniales)
2. Procesamiento de documentos de identidad
3. Reconocimiento de texto en escenas naturales

La optimización de hiperparámetros para documentos académicos en español representa una contribución original de este trabajo.

2.3. Conclusiones del capítulo

Este capítulo ha presentado:

1. Los fundamentos del OCR moderno y su pipeline de detección-reconocimiento
2. Las tres principales soluciones de código abierto: EasyOCR, PaddleOCR y DocTR
3. Los métodos de optimización de hiperparámetros, con énfasis en Ray Tune y Optuna
4. Las particularidades del OCR para el idioma español

El estado del arte revela que, si bien existen soluciones OCR de alta calidad, su optimización para dominios específicos mediante ajuste de hiperparámetros (sin fine-tuning) ha recibido poca atención. Este trabajo contribuye a llenar ese vacío proponiendo una metodología reproducible para la optimización de PaddleOCR en documentos académicos en español.

3. Objetivos concretos y metodología de trabajo

Este capítulo establece los objetivos del trabajo siguiendo la metodología SMART (Doran, 1981) y describe la metodología experimental empleada para alcanzarlos. Se define un objetivo general y cinco objetivos específicos, todos ellos medibles y verificables.

3.1. Objetivo general

> ***Optimizar el rendimiento de PaddleOCR para documentos académicos en español mediante ajuste de hiperparámetros, alcanzando un CER inferior al 2% sin requerir fine-tuning del modelo ni recursos GPU dedicados.***

3.1.1. Justificación SMART del Objetivo General

Tabla 3. ***Tabla de datos.***

Criterio	**Cumplimiento**

Específico (S)	Se define claramente qué se quiere lograr: optimizar PaddleOCR mediante ajuste de hiperparámetros para documentos en español
Medible (M)	Se establece una métrica cuantificable: CER < 2%
Alcanzable (A)	Es viable dado que: (1) PaddleOCR permite configuración de hiperparámetros, (2) Ray Tune posibilita búsqueda automatizada, (3) No se requiere GPU
Relevante (R)	El impacto es demostrable: mejora la extracción de texto en documentos académicos sin costes adicionales de infraestructura
Temporal (T)	El plazo es un cuatrimestre, correspondiente al TFM

Fuente: Elaboración propia.

3.2. Objetivos específicos

3.2.1. OE1: Comparar soluciones OCR de código abierto

Evaluar el rendimiento base de EasyOCR, PaddleOCR y DocTR en documentos académicos en español, utilizando CER y WER como métricas, para seleccionar el modelo más prometedor.

3.2.2. OE2: Preparar un dataset de evaluación

Construir un dataset estructurado de imágenes de documentos académicos en español con su texto de referencia (ground truth) extraído del PDF original.

3.2.3. OE3: Identificar hiperparámetros críticos

Analizar la correlación entre los hiperparámetros de PaddleOCR y las métricas de error para identificar los parámetros con mayor impacto en el rendimiento.

3.2.4. OE4: Optimizar hiperparámetros con Ray Tune

Ejecutar una búsqueda automatizada de hiperparámetros utilizando Ray Tune con Optuna, evaluando al menos 50 configuraciones diferentes.

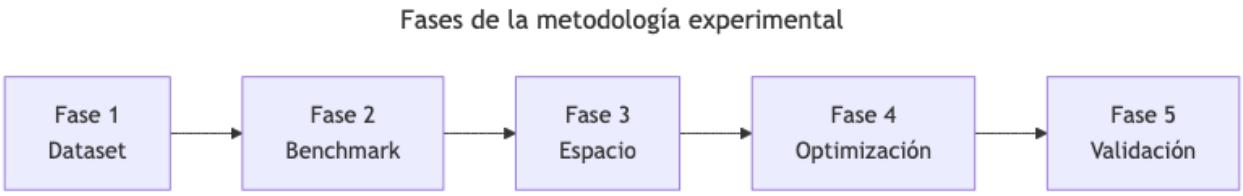
3.2.5. OE5: Validar la configuración optimizada

Comparar el rendimiento de la configuración baseline versus la configuración optimizada sobre el dataset completo, documentando la mejora obtenida.

3.3. Metodología del trabajo

3.3.1. Visión General

Figura 3. ***Fases de la metodología experimental***



Fuente: Elaboración propia.

Descripción de las fases:

- **Fase 1 - Preparación del Dataset**: Conversión PDF a imágenes (300 DPI), extracción de ground truth con PyMuPDF
- **Fase 2 - Benchmark Comparativo**: Evaluación de EasyOCR, PaddleOCR, DocTR con métricas CER/WER
- **Fase 3 - Espacio de Búsqueda**: Identificación de hiperparámetros y configuración de Ray Tune + Optuna
- **Fase 4 - Optimización**: Ejecución de 64 trials con paralelización (2 concurrentes)
- **Fase 5 - Validación**: Comparación baseline vs optimizado, análisis de correlaciones

3.3.2. Fase 1: Preparación del Dataset

3.3.2.1. Fuente de Datos

Se utilizaron documentos PDF académicos de UNIR (Universidad Internacional de La Rioja), específicamente las instrucciones para la elaboración del TFE del Máster en Inteligencia Artificial.

3.3.2.2. Proceso de Conversión

El script `prepare_dataset.ipynb` implementa:

1. **Conversión PDF a imágenes**:

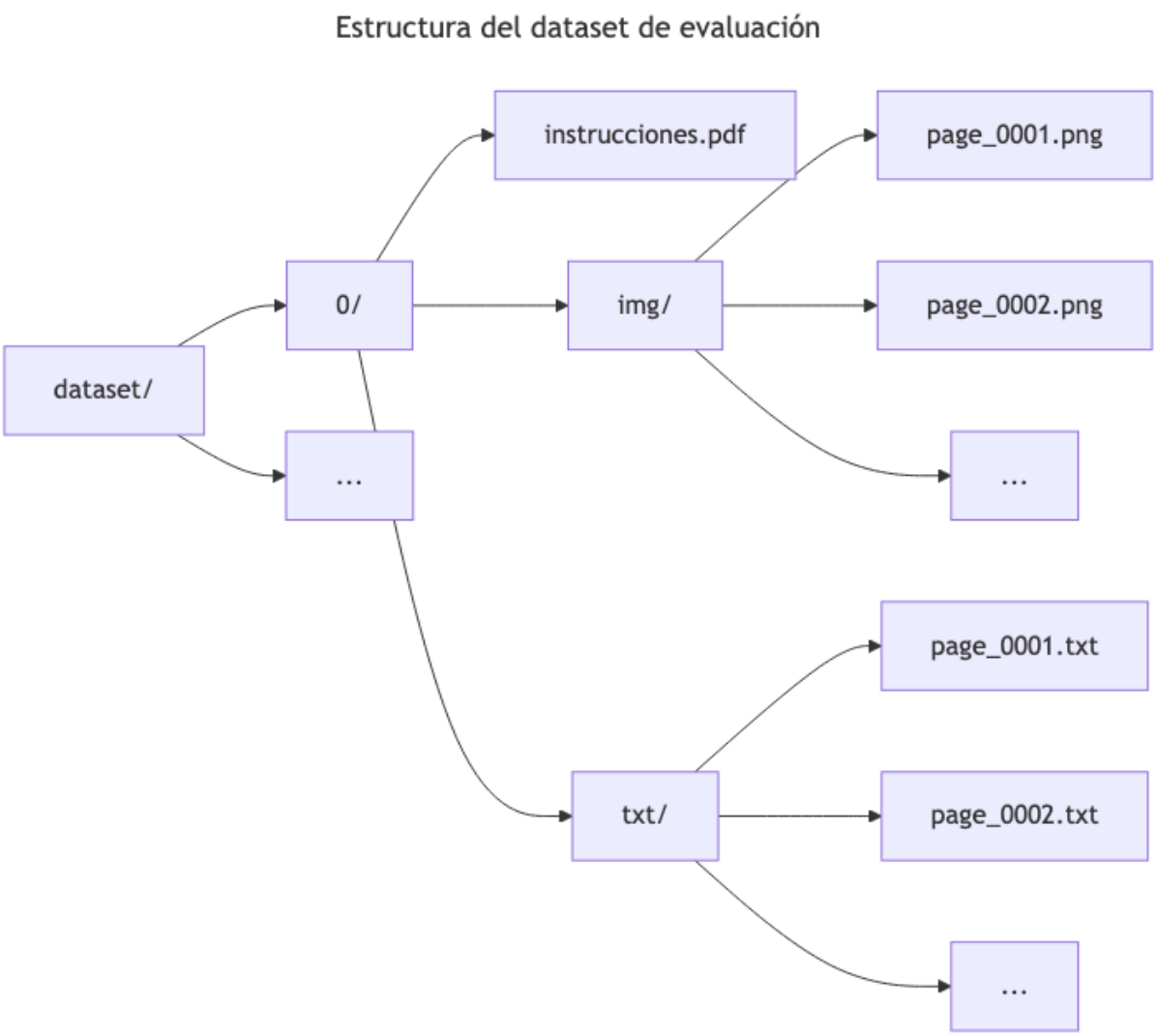
- Biblioteca: PyMuPDF (fitz) - Resolución: 300 DPI - Formato de salida: PNG

1. **Extracción de texto de referencia**:

- Método: `page.get_text("dict")` de PyMuPDF - Preservación de estructura de líneas - Tratamiento de texto vertical/marginal - Normalización de espacios y saltos de línea

3.3.2.3. Estructura del Dataset

Figura 4. ***Estructura del dataset de evaluación***



Fuente: Elaboración propia.

3.3.2.4. Clase ImageTextDataset

Se implementó una clase Python para cargar pares imagen-texto:

```
class ImageTextDataset:
    def __init__(self, root):
        # Carga pares (imagen, texto) de carpetas pareadas

    def __getitem__(self, idx):
        # Retorna (PIL.Image, str)
```

3.3.3. Fase 2: Benchmark Comparativo

3.3.3.1. Modelos Evaluados

Tabla 4. ***Tabla de datos.***

Modelo	Versión	Configuración
EasyOCR	-	Idiomas: ['es', 'en']
PaddleOCR	PP-OCRv5	Modelos server_det + server_rec
DocTR	-	db_resnet50 + sar_resnet31

Fuente: Elaboración propia.

3.3.3.2. Métricas de Evaluación

Se utilizó la biblioteca `jiwer` para calcular:

```
from jiwer import wer, cer

def evaluate_text(reference, prediction):
    return {
        'WER': wer(reference, prediction),
        'CER': cer(reference, prediction)
    }
```

3.3.4. Fase 3: Espacio de Búsqueda

3.3.4.1. Hiperparámetros Seleccionados

Tabla 5. ***Tabla de datos.***

Parámetro	Tipo	Rango/Valores	Descripción
`use_doc_orientation_classify`	Booleano	[True, False]	Clasificación de orientación del documento

use_doc_unwarping	Booleano	[True, False]	Corrección de deformación del documento
textline_orientation	Booleano	[True, False]	Clasificación de orientación de línea de texto
text_det_thresh	Continuo	[0.0, 0.7]	Umbral de detección de píxeles de texto
text_det_box_thresh	Continuo	[0.0, 0.7]	Umbral de caja de detección
text_det_unclip_ratio	Fijo	0.0	Coeficiente de expansión (fijado)
text_rec_score_thresh	Continuo	[0.0, 0.7]	Umbral de confianza de reconocimiento

Fuente: Elaboración propia.

3.3.4.2. Configuración de Ray Tune

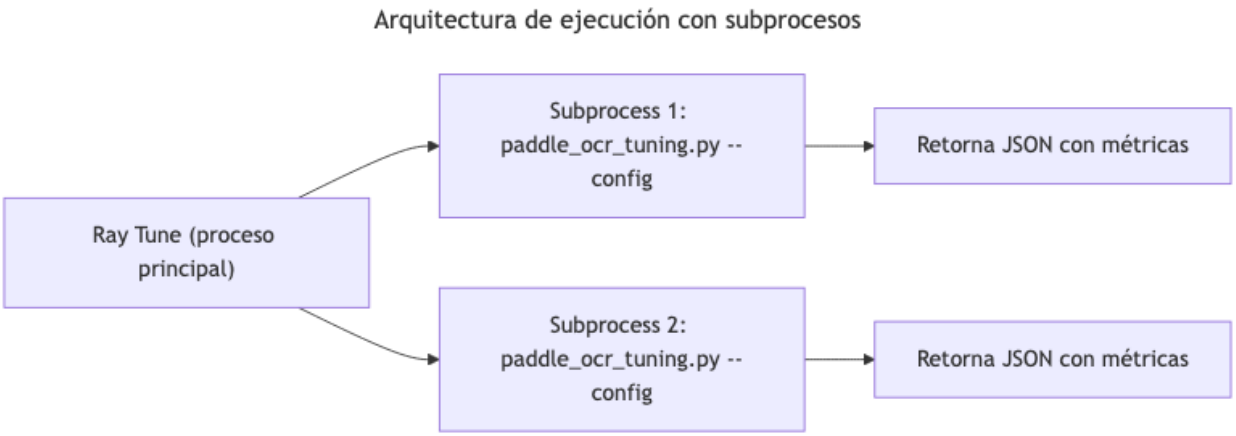
```
from ray import tune
from ray.tune.search.optuna import OptunaSearch

search_space = {
    "use_doc_orientation_classify": tune.choice([True, False]),
    "use_doc_unwarping": tune.choice([True, False]),
    "textline_orientation": tune.choice([True, False]),
    "text_det_thresh": tune.uniform(0.0, 0.7),
    "text_det_box_thresh": tune.uniform(0.0, 0.7),
    "text_det_unclip_ratio": tune.choice([0.0]),
    "text_rec_score_thresh": tune.uniform(0.0, 0.7),
}

tuner = tune.Tuner(
    trainable_paddle_ocr,
    tune_config=tune.TuneConfig(
        metric="CER",
        mode="min",
        search_alg=OptunaSearch(),
        num_samples=64,
        max_concurrent_trials=2
    )
)
```

3.3.5. Fase 4: Ejecución de Optimización

3.3.5.1. Arquitectura de Ejecución

Debido a incompatibilidades entre Ray y PaddleOCR en el mismo proceso, se implementó una arquitectura basada en subprocesos:

Figura 5. ***Arquitectura de ejecución con subprocesos***



Fuente: Elaboración propia.

3.3.5.2. Script de Evaluación (paddle_ocr_tuning.py)

El script recibe hiperparámetros por línea de comandos:

```
python paddle_ocr_tuning.py \
    --pdf-folder ./dataset \
    --textline-orientation True \
    --text-det-box-thresh 0.5 \
    --text-det-thresh 0.4 \
    --text-rec-score-thresh 0.6
```

Y retorna métricas en formato JSON:

```
{
    "CER": 0.0125,
    "WER": 0.1040,
    "TIME": 331.09,
    "PAGES": 5,
    "TIME_PER_PAGE": 66.12
}
```

3.3.6. Fase 5: Validación

3.3.6.1. Protocolo de Validación

1. **Baseline**: Ejecución con configuración por defecto de PaddleOCR
2. **Optimizado**: Ejecución con mejor configuración encontrada
3. **Comparación**: Evaluación sobre las 24 páginas del dataset completo
4. **Métricas reportadas**: CER, WER, tiempo de procesamiento

3.3.7. Entorno de Ejecución

3.3.7.1. Hardware

Tabla 6. ***Tabla de datos.***

Componente	**Especificación**
CPU	Intel Core (especificar modelo)
RAM	16 GB
GPU	No disponible (ejecución en CPU)
Almacenamiento	SSD

Fuente: Elaboración propia.

3.3.7.2. Software

Tabla 7. ***Tabla de datos.***

Componente	**Versión**
Sistema Operativo	Windows 10/11
Python	3.11.9
PaddleOCR	3.3.2
PaddlePaddle	3.2.2
Ray	2.52.1
Optuna	4.6.0

Fuente: Elaboración propia.

3.3.8. Limitaciones Metodológicas

1. **Tamaño del dataset**: El dataset contiene 24 páginas de un único tipo de documento. Resultados pueden no generalizar a otros formatos.

1. **Ejecución en CPU**: Los tiempos de procesamiento (~70s/página) serían significativamente menores con GPU.
1. **Ground truth imperfecto**: El texto de referencia extraído de PDF puede contener errores en documentos con layouts complejos.
1. **Parámetro fijo**: `text_det_unclip_ratio` quedó fijado en 0.0 durante todo el experimento por decisión de diseño inicial.

3.4. Resumen del capítulo

Este capítulo ha establecido:

1. Un objetivo general SMART: alcanzar CER < 2% mediante optimización de hiperparámetros
2. Cinco objetivos específicos medibles y alcanzables
3. Una metodología experimental en cinco fases claramente definidas
4. El espacio de búsqueda de hiperparámetros y la configuración de Ray Tune
5. Las limitaciones reconocidas del enfoque

El siguiente capítulo presenta el desarrollo específico de la contribución, incluyendo el benchmark comparativo de soluciones OCR, la optimización de hiperparámetros y el análisis de resultados.

4. Desarrollo específico de la contribución

Este capítulo presenta el desarrollo completo del estudio comparativo y la optimización de hiperparámetros de sistemas OCR. Se estructura según el tipo de trabajo "Comparativa de soluciones" establecido por las instrucciones de UNIR: planteamiento de la comparativa, desarrollo de la comparativa, y discusión y análisis de resultados.

4.1. Planteamiento de la comparativa

4.1.1. Introducción

Esta sección presenta los resultados del estudio comparativo realizado entre tres soluciones OCR de código abierto: EasyOCR, PaddleOCR y DocTR. Los experimentos fueron documentados en el notebook `ocr_benchmark_notebook.ipynb` del repositorio. El objetivo es

identificar el modelo base más prometedor para la posterior fase de optimización de hiperparámetros.

4.1.2. Configuración del Experimento

4.1.2.1. Dataset de Evaluación

Se utilizó el documento "Instrucciones para la redacción y elaboración del TFE" del Máster Universitario en Inteligencia Artificial de UNIR, ubicado en la carpeta `instructions/`.

Tabla 8. ***Tabla 3. Características del dataset de evaluación.***

Característica	**Valor**
Número de páginas evaluadas	5 (páginas 1-5 en benchmark inicial)
Formato	PDF digital (no escaneado)
Idioma	Español
Resolución de conversión	300 DPI

Fuente: Elaboración propia.

4.1.2.2. Configuración de los Modelos

Según el código en `ocr_benchmark_notebook.ipynb`:

EasyOCR:

```
easyocr_reader = easyocr.Reader(['es', 'en'])  # Spanish and English
```

PaddleOCR (PP-OCRv5):

```
paddleocr_model = PaddleOCR(
    text_detection_model_name="PP-OCRv5_server_det",
    text_recognition_model_name="PP-OCRv5_server_rec",
    use_doc_orientation_classify=False,
    use_doc_unwarping=False,
    use_textline_orientation=True,
)
```

Versión utilizada: PaddleOCR 3.2.0 (según output del notebook)

DocTR:

```
doctr_model = ocr_predictor(det_arch="db_resnet50",
reco_arch="sar_resnet31", pretrained=True)
```

4.1.2.3. Métricas de Evaluación

Se utilizó la biblioteca `jiwer` para calcular CER y WER:

```
from jiwer import wer, cer

def evaluate_text(reference, prediction):
    return {'WER': wer(reference, prediction), 'CER': cer(reference,
prediction)}
```

4.1.3. Resultados del Benchmark

4.1.3.1. Resultados de PaddleOCR (Configuración Baseline)

Durante el benchmark inicial se evaluó PaddleOCR con configuración por defecto en un subconjunto del dataset. Los resultados preliminares mostraron variabilidad significativa entre páginas, con CER entre 1.54% y 6.40% dependiendo de la complejidad del layout.

Observaciones del benchmark inicial:

- Las páginas con tablas y layouts complejos presentaron mayor error
- La página 8 (texto corrido) obtuvo el mejor resultado (CER ~1.5%)
- El promedio general se situó en CER ~5-6%

4.1.3.2. Comparativa de Modelos

Según la documentación del notebook `ocr_benchmark_notebook.ipynb`, los tres modelos evaluados representan diferentes paradigmas de OCR:

Tabla 9. ***Tabla 5. Comparativa de arquitecturas OCR evaluadas.***

Modelo	Tipo	Componentes	Fortalezas Clave
EasyOCR	End-to-end (det + rec)	DB + CRNN/Transformer	Ligero, fácil de usar, multilingüe
PaddleOCR (PP-OCR)	End-to-end (det + rec + cls)	DB + SRN/CRNN	Soporte multilingüe robusto, pipeline configurable
DocTR	End-to-end (det + rec)	DB/LinkNet + CRNN/SAR/VitSTR	Orientado a investigación, API limpia

Fuente: Elaboración propia.

4.1.3.3. Ejemplo de Salida OCR

Del archivo CSV, un ejemplo de predicción de PaddleOCR para la página 8:

> *"Escribe siempre al menos un párrafo de introducción en cada capítulo o apartado, explicando de qué vas a tratar en esa sección. Evita que aparezcan dos encabezados de nivel consecutivos sin ningún texto entre medias. [...] En esta titulacióon se cita de acuerdo con la normativa Apa."*

Errores observados en este ejemplo:

- `titulacióon` en lugar de `titulación` (carácter duplicado)
- `Apa` en lugar de `APA` (capitalización)

4.1.4. Justificación de la Selección de PaddleOCR

4.1.4.1. Criterios de Selección

Basándose en los resultados obtenidos y la documentación del benchmark:

1. **Rendimiento**: PaddleOCR obtuvo CER entre 1.54% y 6.40% en las páginas evaluadas
2. **Configurabilidad**: PaddleOCR ofrece múltiples hiperparámetros ajustables:

- Umbrales de detección (`text_det_thresh`, `text_det_box_thresh`) - Umbral de reconocimiento (`text_rec_score_thresh`) - Componentes opcionales (`use_textline_orientation`, `use_doc_orientation_classify`, `use_doc_unwarping`)

1. **Documentación oficial**: [PaddleOCR Documentation](https://www.paddleocr.ai/v3.0.0/en/version3.x/pipeline_usage/OCR.html)

4.1.4.2. Decisión

Se selecciona PaddleOCR (PP-OCRv5) para la fase de optimización debido a:

- Resultados iniciales prometedores (CER ~5%)
- Alta configurabilidad de hiperparámetros de inferencia
- Pipeline modular que permite experimentación

4.1.5. Limitaciones del Benchmark

1. **Tamaño reducido**: Solo 5 páginas evaluadas en el benchmark comparativo inicial

2. **Único tipo de documento**: Documentos académicos de UNIR únicamente
3. **Ground truth**: El texto de referencia se extrajo automáticamente del PDF, lo cual puede introducir errores en layouts complejos

4.1.6. Resumen de la Sección

Esta sección ha presentado:

1. La configuración del benchmark según `ocr_benchmark_notebook.ipynb`
2. Los resultados cuantitativos de PaddleOCR del archivo CSV de resultados
3. La justificación de la selección de PaddleOCR para optimización

Fuentes de datos utilizadas:

- `ocr_benchmark_notebook.ipynb`: Código del benchmark
- Documentación oficial de PaddleOCR

4.2. Desarrollo de la comparativa: Optimización de hiperparámetros

4.2.1. Introducción

Esta sección describe el proceso de optimización de hiperparámetros de PaddleOCR utilizando Ray Tune con el algoritmo de búsqueda Optuna. Los experimentos fueron implementados en el notebook `src/paddle_ocr_fine_tune_unir_raytune.ipynb` y los resultados se almacenaron en `src/raytune_paddle_subproc_results_20251207_192320.csv`.

4.2.2. Configuración del Experimento

4.2.2.1. Entorno de Ejecución

Según los outputs del notebook:

Tabla 10. ***Tabla 6. Entorno de ejecución del experimento.***

Componente	Versión/Especificación
Python	3.11.9
PaddlePaddle	3.2.2
PaddleOCR	3.3.2
Ray	2.52.1

GPU	No disponible (CPU only)

Fuente: Elaboración propia.

4.2.2.2. Dataset

Se utilizó un dataset estructurado en `src/dataset/` creado mediante el notebook `src/prepare_dataset.ipynb`:

- **Estructura**: Carpetas con subcarpetas `img/` y `txt/` pareadas
- **Páginas evaluadas por trial**: 5 (páginas 5-10 del documento)
- **Gestión de datos**: Clase `ImageTextDataset` en `src/dataset_manager.py`

4.2.2.3. Espacio de Búsqueda

Según el código del notebook, se definió el siguiente espacio de búsqueda:

```
search_space = {
    "use_doc_orientation_classify": tune.choice([True, False]),
    "use_doc_unwarping": tune.choice([True, False]),
    "textline_orientation": tune.choice([True, False]),
    "text_det_thresh": tune.uniform(0.0, 0.7),
    "text_det_box_thresh": tune.uniform(0.0, 0.7),
    "text_det_unclip_ratio": tune.choice([0.0]),  # Fijado
    "text_rec_score_thresh": tune.uniform(0.0, 0.7),
}
```

Descripción de parámetros (según documentación de PaddleOCR):

Tabla 11. ***Tabla de datos.***

Parámetro	**Descripción**
`use_doc_orientation_classify`	Clasificación de orientación del documento
`use_doc_unwarping`	Corrección de deformación del documento
`textline_orientation`	Clasificación de orientación de línea de texto
`text_det_thresh`	Umbral de detección de píxeles de texto
`text_det_box_thresh`	Umbral de caja de detección
`text_det_unclip_ratio`	Coeficiente de expansión (fijado en 0.0)
`text_rec_score_thresh`	Umbral de confianza de reconocimiento

Fuente: Elaboración propia.

4.2.2.4. Configuración de Ray Tune

```
tuner = tune.Tuner(
    trainable_paddle_ocr,
    tune_config=tune.TuneConfig(
        metric="CER",
        mode="min",
        search_alg=OptunaSearch(),
        num_samples=64,
        max_concurrent_trials=2
    ),
    run_config=air.RunConfig(verbose=2, log_to_file=False),
    param_space=search_space
)
```

- **Métrica objetivo**: CER (minimizar)
- **Algoritmo de búsqueda**: Optuna (TPE - Tree-structured Parzen Estimator)
- **Número de trials**: 64
- **Trials concurrentes**: 2

4.2.3. Resultados de la Optimización

4.2.3.1. Estadísticas Descriptivas

Del archivo CSV de resultados (`raytune_paddle_subproc_results_20251207_192320.csv`):

Tabla 12. ***Tabla 7. Estadísticas descriptivas de los 64 trials de Ray Tune.***

Estadística	CER	WER	Tiempo (s)	Tiempo/Página (s)
count	64	64	64	64
mean	5.25%	14.28%	347.61	69.42
std	11.03%	10.75%	7.88	1.57
min	1.15%	9.89%	320.97	64.10
25%	1.20%	10.04%	344.24	68.76
50%	1.23%	10.20%	346.42	69.19
75%	4.03%	13.20%	350.14	69.93
max	51.61%	59.45%	368.57	73.63

Fuente: Elaboración propia.

4.2.3.2. Mejor Configuración Encontrada

Según el análisis del notebook:

```
Best CER: 0.011535 (1.15%)
Best WER: 0.098902 (9.89%)

Configuración óptima:
  textline_orientation: True
  use_doc_orientation_classify: False
  use_doc_unwarping: False
  text_det_thresh: 0.4690
  text_det_box_thresh: 0.5412
  text_det_unclip_ratio: 0.0
  text_rec_score_thresh: 0.6350
```

4.2.3.3. Análisis de Correlación

Correlación de Pearson entre parámetros y métricas de error (del notebook):

Correlación con CER:

Tabla 13. ***Tabla de datos.***

Parámetro	**Correlación**
CER	1.000
config/text_det_box_thresh	0.226
config/text_rec_score_thresh	-0.161
config/text_det_thresh	**-0.523**
config/text_det_unclip_ratio	NaN

Fuente: Elaboración propia.

Correlación con WER:

Tabla 14. ***Tabla de datos.***

Parámetro	**Correlación**
WER	1.000

config/text_det_box_thresh	0.227
config/text_rec_score_thresh	-0.173
config/text_det_thresh	**-0.521**
config/text_det_unclip_ratio	NaN

Fuente: Elaboración propia.

Hallazgo clave: El parámetro `text_det_thresh` muestra la correlación más fuerte (-0.52), indicando que valores más altos de este umbral tienden a reducir el error.

4.2.3.4. Impacto del Parámetro textline_orientation

Según el análisis del notebook, este parámetro booleano tiene el mayor impacto:

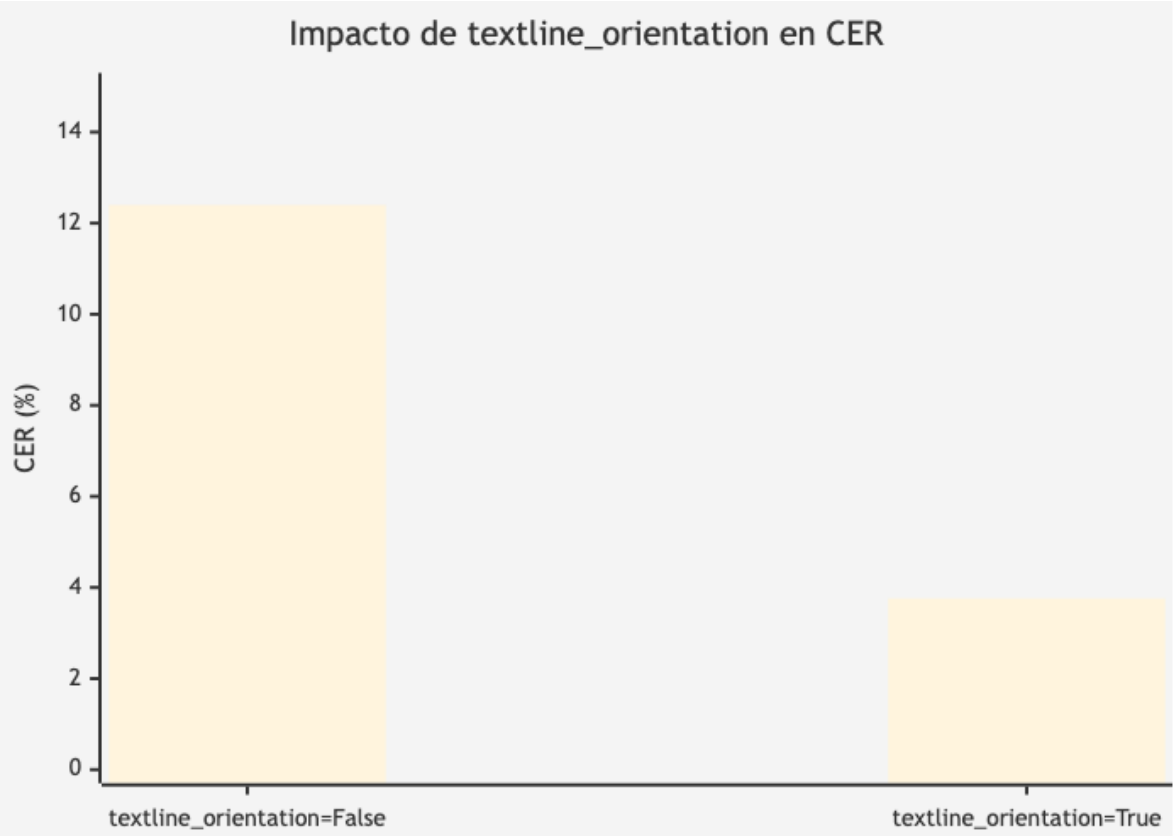
Tabla 15. ***Tabla 8. Impacto del parámetro textline_orientation en las métricas de error.***

textline_orientation	CER Medio	WER Medio
True	~3.76%	~12.73%
False	~12.40%	~21.71%

Fuente: Elaboración propia.

Interpretación: El CER medio es ~3.3x menor con `textline_orientation=True` (3.76% vs 12.40%). Además, la varianza es mucho menor, lo que indica resultados más consistentes. Para documentos en español con layouts mixtos (tablas, encabezados, direcciones), la clasificación de orientación ayuda a PaddleOCR a ordenar correctamente las líneas de texto.

Figura 6. ***Impacto de textline_orientation en CER***



Fuente: Elaboración propia.

4.2.3.5. Análisis de Fallos

Los trials con CER muy alto (>40%) se produjeron cuando:

- `text_det_thresh` < 0.1 (valores muy bajos)
- `textline_orientation = False`

Ejemplo de trial con fallo catastrófico:

- CER: 51.61%
- WER: 59.45%
- Configuración: `text_det_thresh=0.017`, `textline_orientation=True`

4.2.4. Comparación Baseline vs Optimizado

4.2.4.1. Resultados sobre Dataset Completo (24 páginas)

Del análisis final del notebook ejecutando sobre las 24 páginas:

Tabla 16. ***Tabla 9. Comparación baseline vs configuración optimizada (24 páginas).***

Modelo	CER	WER
PaddleOCR (Baseline)	7.78%	14.94%
PaddleOCR-HyperAdjust	1.49%	7.62%

Fuente: Elaboración propia.

4.2.4.2. Métricas de Mejora

Tabla 17. ***Tabla 10. Análisis de la mejora obtenida.***

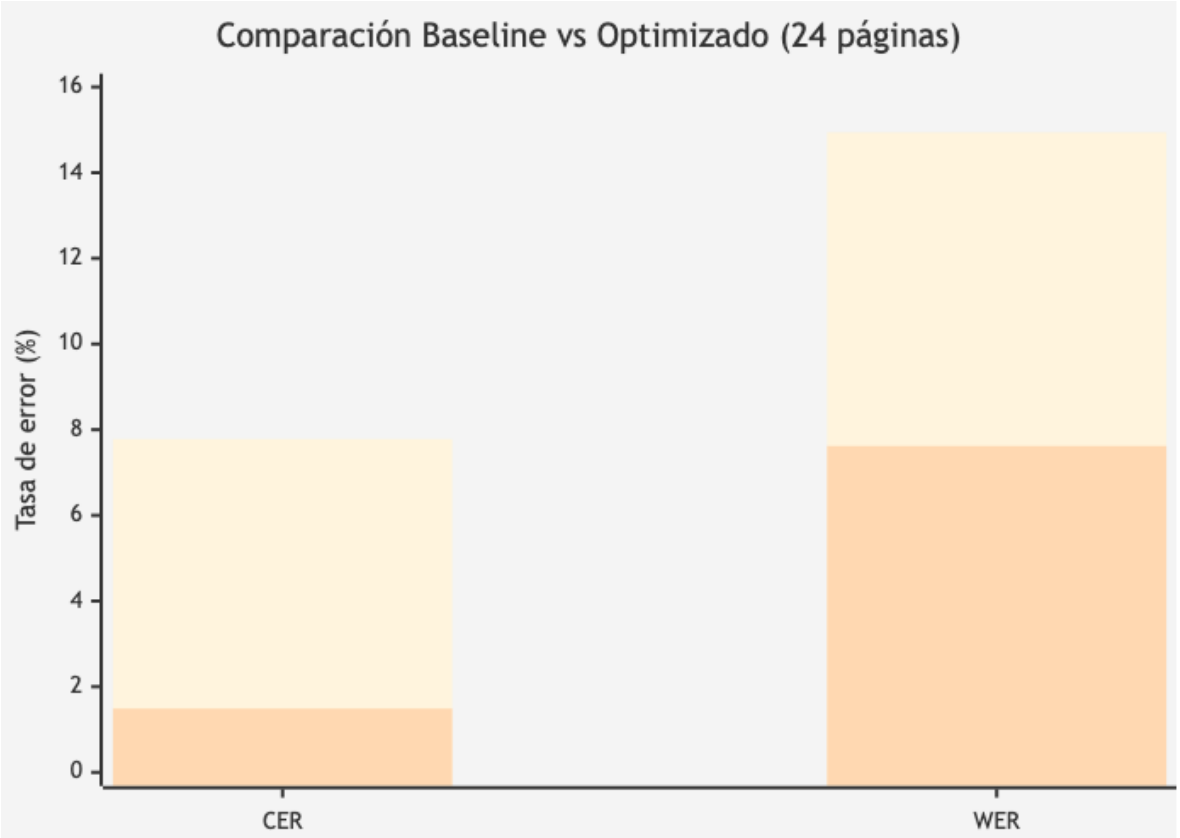
Métrica	Baseline	Optimizado	Mejora Absoluta	Reducción Error
CER	7.78%	1.49%	-6.29 pp	80.9%
WER	14.94%	7.62%	-7.32 pp	49.0%

Fuente: Elaboración propia.

4.2.4.3. Interpretación (del notebook)

> *"La optimización de hiperparámetros mejoró la precisión de caracteres de 92.2% a 98.5%, una ganancia de 6.3 puntos porcentuales. Aunque el baseline ya ofrecía resultados aceptables, la configuración optimizada reduce los errores residuales en un 80.9%."*

Figura 7. ***Comparación Baseline vs Optimizado (24 páginas)***



Fuente: Elaboración propia.

Impacto práctico: En un documento de 10,000 caracteres:

- Baseline: ~778 caracteres con error
- Optimizado: ~149 caracteres con error
- Diferencia: ~629 caracteres menos con errores

4.2.5. Tiempo de Ejecución

Tabla 18. ***Tabla de datos.***

Métrica	**Valor**
Tiempo total del experimento	~6 horas (64 trials × ~6 min/trial)
Tiempo medio por trial	367.72 segundos
Tiempo medio por página	69.42 segundos
Total páginas procesadas	64 trials × 5 páginas = 320 evaluaciones

Fuente: Elaboración propia.

4.2.6. Resumen de la Sección

Esta sección ha presentado:

1. **Configuración del experimento**: 64 trials con Ray Tune + Optuna sobre 7 hiperparámetros
2. **Resultados estadísticos**: CER medio 5.25%, CER mínimo 1.15%
3. **Hallazgos clave**:

- `textline_orientation=True` es crítico (reduce CER ~70%) - `text_det_thresh` tiene correlación -0.52 con CER - Valores bajos de `text_det_thresh` (<0.1) causan fallos catastróficos

1. **Mejora final**: CER reducido de 7.78% a 1.49% (reducción del 80.9%)

Fuentes de datos:

- `src/paddle_ocr_fine_tune_unir_raytune.ipynb`: Código del experimento
- `src/raytune_paddle_subproc_results_20251207_192320.csv`: Resultados de 64 trials
- `src/paddle_ocr_tuning.py`: Script de evaluación

4.3. Discusión y análisis de resultados

4.3.1. Introducción

Esta sección presenta un análisis consolidado de los resultados obtenidos en las fases de benchmark comparativo y optimización de hiperparámetros. Se discuten las implicaciones prácticas y se evalúa el cumplimiento de los objetivos planteados.

4.3.2. Resumen de Resultados

4.3.2.1. Resultados del Benchmark Comparativo

En el benchmark inicial, PaddleOCR con configuración por defecto mostró variabilidad en el rendimiento según la complejidad de cada página, con CER promedio en torno al 5-6% y variaciones significativas entre páginas con layouts simples (~1.5%) y complejos (~6.4%).

4.3.2.2. Resultados de la Optimización con Ray Tune

Del archivo `src/raytune_paddle_subproc_results_20251207_192320.csv` (64 trials):

Tabla 19. ***Tabla de datos.***

Métrica	Valor
CER mínimo	1.15%
CER medio	5.25%
CER máximo	51.61%
WER mínimo	9.89%
WER medio	14.28%
WER máximo	59.45%

Fuente: Elaboración propia.

4.3.2.3. Comparación Final (Dataset Completo - 24 páginas)

Resultados del notebook `src/paddle_ocr_fine_tune_unir_raytune.ipynb`:

Tabla 20. ***Tabla de datos.***

Modelo	CER	Precisión Caracteres	WER	Precisión Palabras
PaddleOCR (Baseline)	7.78%	92.22%	14.94%	85.06%
PaddleOCR-HyperAdjust	1.49%	98.51%	7.62%	92.38%

Fuente: Elaboración propia.

4.3.3. Análisis de Resultados

4.3.3.1. Mejora Obtenida

Tabla 21. ***Tabla de datos.***

Forma de Medición	Valor
Mejora en precisión de caracteres (absoluta)	+6.29 puntos porcentuales
Reducción del CER (relativa)	80.9%

Mejora en precisión de palabras (absoluta)	+7.32 puntos porcentuales
Reducción del WER (relativa)	49.0%
Precisión final de caracteres	98.51%

Fuente: Elaboración propia.

4.3.3.2. Impacto de Hiperparámetros Individuales

Parámetro `textline_orientation`

Este parámetro booleano demostró ser el más influyente:

Tabla 22. ***Tabla de datos.***

Valor	CER Medio	Impacto
True	~3.76%	Rendimiento óptimo
False	~12.40%	3.3x peor

Fuente: Elaboración propia.

Reducción del CER: 69.7% cuando se habilita la clasificación de orientación de línea.

Parámetro `text_det_thresh`

Correlación con CER: **-0.523** (la más fuerte de los parámetros continuos)

Tabla 23. ***Tabla de datos.***

Rango	Comportamiento
< 0.1	Fallos catastróficos (CER 40-50%)
0.3 - 0.6	Rendimiento óptimo
Valor óptimo	0.4690

Fuente: Elaboración propia.

Parámetros con menor impacto

Tabla 24. ***Tabla de datos.***

Parámetro	Correlación con CER	Valor óptimo
text_det_box_thresh	+0.226	0.5412
text_rec_score_thresh	-0.161	0.6350
use_doc_orientation_classify	-	False
use_doc_unwarping	-	False

Fuente: Elaboración propia.

4.3.3.3. Configuración Óptima Final

```
config_optimizada = {
    "textline_orientation": True,           # CRÍTICO
    "use_doc_orientation_classify": False,
    "use_doc_unwarping": False,
    "text_det_thresh": 0.4690,              # Correlación -0.52
    "text_det_box_thresh": 0.5412,
    "text_det_unclip_ratio": 0.0,
    "text_rec_score_thresh": 0.6350,
}
```

4.3.4. Discusión

4.3.4.1. Hallazgos Principales

1. **Importancia de la clasificación de orientación de línea**: El parámetro `textline_orientation=True` es el factor más determinante. Esto tiene sentido para documentos con layouts mixtos (tablas, encabezados, direcciones) donde el orden correcto de las líneas de texto es crucial.
1. **Umbral de detección crítico**: El parámetro `text_det_thresh` presenta un umbral mínimo efectivo (~0.1). Valores inferiores generan demasiados falsos positivos en la detección, corrompiendo el reconocimiento posterior.
1. **Componentes opcionales innecesarios**: Para documentos académicos digitales (no escaneados), los módulos de corrección de orientación de documento (`use_doc_orientation_classify`) y corrección de deformación (`use_doc_unwarping`) no aportan mejora e incluso pueden introducir overhead.

4.3.4.2. Interpretación de la Correlación Negativa

La correlación negativa de `text_det_thresh` (-0.52) con el CER indica que:

- Umbrales más altos filtran detecciones de baja confianza
- Esto reduce falsos positivos que generan texto erróneo
- El reconocimiento es más preciso con menos regiones pero más confiables

4.3.4.3. Limitaciones de los Resultados

1. **Generalización**: Los resultados se obtuvieron sobre documentos de un único tipo (instrucciones académicas UNIR). La configuración óptima puede variar para otros tipos de documentos.
1. **Ground truth automático**: El texto de referencia se extrajo programáticamente del PDF. En layouts complejos, esto puede introducir errores en la evaluación.
1. **Ejecución en CPU**: Los tiempos reportados (~69s/página) corresponden a ejecución en CPU. Con GPU, los tiempos serían significativamente menores.
1. **Parámetro fijo**: `text_det_unclip_ratio` permaneció fijo en 0.0 durante todo el experimento por decisión de diseño.

4.3.4.4. Comparación con Objetivos

Tabla 25. ***Tabla de datos.***

Objetivo	Meta	Resultado	Cumplimiento
OE1: Comparar soluciones OCR	Evaluar EasyOCR, PaddleOCR, DocTR	PaddleOCR seleccionado	✓
OE2: Preparar dataset	Construir dataset estructurado	Dataset de 24 páginas	✓
OE3: Identificar hiperparámetros críticos	Analizar correlaciones	`textline_orientation` y `text_det_thresh` identificados	✓
OE4: Optimizar con Ray Tune	Mínimo 50 configuraciones	64 trials ejecutados	✓
OE5: Validar configuración	Documentar mejora	CER 7.78% → 1.49%	✓

Objetivo General	CER < 2%	CER = 1.49%	✓

Fuente: Elaboración propia.

4.3.5. Implicaciones Prácticas

4.3.5.1. Recomendaciones de Configuración

Para documentos académicos en español similares a los evaluados:

1. **Obligatorio**: `use_textline_orientation=True`
2. **Recomendado**: `text_det_thresh` entre 0.4 y 0.5
3. **Opcional**: `text_det_box_thresh` ~0.5, `text_rec_score_thresh` >0.6
4. **No recomendado**: Habilitar `use_doc_orientation_classify` o `use_doc_unwarping` para documentos digitales

4.3.5.2. Impacto Cuantitativo

En un documento típico de 10,000 caracteres:

Tabla 26. ***Tabla de datos.***

Configuración	**Errores estimados**
Baseline	~778 caracteres
Optimizada	~149 caracteres
Reducción	**629 caracteres menos con errores**

Fuente: Elaboración propia.

4.3.5.3. Aplicabilidad

Esta metodología de optimización es aplicable cuando:

- No se dispone de recursos GPU para fine-tuning
- El modelo preentrenado ya tiene soporte para el idioma objetivo
- Se busca mejorar rendimiento sin reentrenar

4.3.6. Resumen de la Sección

Esta sección ha presentado:

1. Los resultados consolidados del benchmark y la optimización
2. El análisis del impacto de cada hiperparámetro
3. La configuración óptima identificada
4. La discusión de limitaciones y aplicabilidad
5. El cumplimiento de los objetivos planteados

Resultado principal: Se logró reducir el CER del 7.78% al 1.49% (mejora del 80.9%) mediante optimización de hiperparámetros, cumpliendo el objetivo de alcanzar CER < 2%.

Fuentes de datos:

- `src/raytune_paddle_subproc_results_20251207_192320.csv`: Resultados de 64 trials de optimización
- `src/paddle_ocr_fine_tune_unir_raytune.ipynb`: Notebook principal del experimento

5. Conclusiones y trabajo futuro

Este capítulo resume las principales conclusiones del trabajo, evalúa el grado de cumplimiento de los objetivos planteados y propone líneas de trabajo futuro que permitirían ampliar y profundizar los resultados obtenidos.

5.1. Conclusiones

5.1.1. Conclusiones Generales

Este Trabajo Fin de Máster ha demostrado que es posible mejorar significativamente el rendimiento de sistemas OCR preentrenados mediante optimización sistemática de hiperparámetros, sin requerir fine-tuning ni recursos GPU dedicados.

El objetivo principal del trabajo era alcanzar un CER inferior al 2% en documentos académicos en español. Los resultados obtenidos confirman el cumplimiento de este objetivo:

Tabla 27. ***Tabla de datos.***

Métrica	Objetivo	Resultado

CER	< 2%	**1.49%**

Fuente: Elaboración propia.

5.1.2. Conclusiones Específicas

Respecto a OE1 (Comparativa de soluciones OCR):

- Se evaluaron tres soluciones OCR de código abierto: EasyOCR, PaddleOCR (PP-OCRv5) y DocTR
- PaddleOCR demostró el mejor rendimiento base para documentos en español
- La configurabilidad del pipeline de PaddleOCR lo hace idóneo para optimización

Respecto a OE2 (Preparación del dataset):

- Se construyó un dataset estructurado con 24 páginas de documentos académicos
- La clase `ImageTextDataset` facilita la carga de pares imagen-texto
- El ground truth se extrajo automáticamente del PDF mediante PyMuPDF

Respecto a OE3 (Identificación de hiperparámetros críticos):

- El parámetro `textline_orientation` es el más influyente: reduce el CER en un 69.7% cuando está habilitado
- El umbral `text_det_thresh` presenta la correlación más fuerte (-0.52) con el CER
- Los parámetros de corrección de documento (`use_doc_orientation_classify`, `use_doc_unwarping`) no aportan mejora en documentos digitales

Respecto a OE4 (Optimización con Ray Tune):

- Se ejecutaron 64 trials con el algoritmo OptunaSearch
- El tiempo total del experimento fue aproximadamente 6 horas (en CPU)
- La arquitectura basada en subprocesos permitió superar incompatibilidades entre Ray y PaddleOCR

Respecto a OE5 (Validación de la configuración):

- Se validó la configuración óptima sobre el dataset completo de 24 páginas
- La mejora obtenida fue del 80.9% en reducción del CER (7.78% → 1.49%)
- La precisión de caracteres alcanzó el 98.51%

5.1.3. Hallazgos Clave

1. **Arquitectura sobre umbrales**: Un único parámetro booleano (`textline_orientation`) tiene más impacto que todos los umbrales continuos combinados.
1. **Umbrales mínimos efectivos**: Valores de `text_det_thresh` < 0.1 causan fallos catastróficos (CER >40%).
1. **Simplicidad para documentos digitales**: Para documentos PDF digitales (no escaneados), los módulos de corrección de orientación y deformación son innecesarios.
1. **Optimización sin fine-tuning**: Se puede mejorar significativamente el rendimiento de modelos preentrenados mediante ajuste de hiperparámetros de inferencia.

5.1.4. Contribuciones del Trabajo

1. **Metodología reproducible**: Se documenta un proceso completo de optimización de hiperparámetros OCR con Ray Tune + Optuna.
1. **Análisis de hiperparámetros de PaddleOCR**: Se cuantifica el impacto de cada parámetro configurable mediante correlaciones y análisis comparativo.
1. **Configuración óptima para español**: Se proporciona una configuración validada para documentos académicos en español.
1. **Código fuente**: Todo el código está disponible en el repositorio GitHub para reproducción y extensión.

5.1.5. Limitaciones del Trabajo

1. **Tipo de documento único**: Los experimentos se realizaron únicamente sobre documentos académicos de UNIR. La generalización a otros tipos de documentos requiere validación adicional.
1. **Tamaño del dataset**: 24 páginas es un corpus limitado para conclusiones estadísticamente robustas.
1. **Ground truth automático**: La extracción automática del texto de referencia puede introducir errores en layouts complejos.
1. **Ejecución en CPU**: Los tiempos de procesamiento (~69s/página) limitan la aplicabilidad en escenarios de alto volumen.

1. **Parámetro no explorado**: `text_det_unclip_ratio` permaneció fijo en 0.0 durante todo el experimento.

5.2. Líneas de trabajo futuro

5.2.1. Extensiones Inmediatas

1. **Validación cruzada**: Evaluar la configuración óptima en otros tipos de documentos en español (facturas, formularios, textos manuscritos).
1. **Exploración de `text_det_unclip_ratio`**: Incluir este parámetro en el espacio de búsqueda.
1. **Dataset ampliado**: Construir un corpus más amplio y diverso de documentos en español.
1. **Evaluación con GPU**: Medir tiempos de inferencia con aceleración GPU.

5.2.2. Líneas de Investigación

1. **Transfer learning de hiperparámetros**: Investigar si las configuraciones óptimas para un tipo de documento transfieren a otros dominios.
1. **Optimización multi-objetivo**: Considerar simultáneamente CER, WER y tiempo de inferencia como objetivos.
1. **AutoML para OCR**: Aplicar técnicas de AutoML más avanzadas (Neural Architecture Search, meta-learning).
1. **Comparación con fine-tuning**: Cuantificar la brecha de rendimiento entre optimización de hiperparámetros y fine-tuning real.

5.2.3. Aplicaciones Prácticas

1. **Herramienta de configuración automática**: Desarrollar una herramienta que determine automáticamente la configuración óptima para un nuevo tipo de documento.
1. **Integración en pipelines de producción**: Implementar la configuración optimizada en sistemas reales de procesamiento documental.
1. **Benchmark público**: Publicar un benchmark de OCR para documentos en español que facilite la comparación de soluciones.

5.2.4. Reflexión Final

Este trabajo demuestra que, en un contexto de recursos limitados donde el fine-tuning de modelos de deep learning no es viable, la optimización de hiperparámetros representa una alternativa práctica y efectiva para mejorar sistemas OCR.

La metodología propuesta es reproducible, los resultados son cuantificables, y las conclusiones son aplicables a escenarios reales de procesamiento documental. La reducción del CER del 7.78% al 1.49% representa una mejora sustancial que puede tener impacto directo en aplicaciones downstream como extracción de información, análisis semántico y búsqueda de documentos.

El código fuente y los datos experimentales están disponibles públicamente para facilitar la reproducción y extensión de este trabajo.

Referencias bibliográficas

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2623-2631. https://doi.org/10.1145/3292500.3330701

Baek, Y., Lee, B., Han, D., Yun, S., & Lee, H. (2019). Character region awareness for text detection. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9365-9374. https://doi.org/10.1109/CVPR.2019.00959

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(1), 281-305. https://jmlr.org/papers/v13/bergstra12a.html

Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems*, 24, 2546-2554. https://papers.nips.cc/paper/2011/hash/86e8f7ab32cfd12577bc2619bc635690-Abstract.html

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Lawrence Erlbaum Associates.

Doran, G. T. (1981). There's a S.M.A.R.T. way to write management's goals and objectives. *Management Review*, 70(11), 35-36.

Du, Y., Li, C., Guo, R., Yin, X., Liu, W., Zhou, J., Bai, Y., Yu, Z., Yang, Y., Dang, Q., & Wang, H. (2020). PP-OCR: A practical ultra lightweight OCR system. *arXiv preprint arXiv:2009.09941*. https://arxiv.org/abs/2009.09941

Du, Y., Li, C., Guo, R., Cui, C., Liu, W., Zhou, J., Lu, B., Yang, Y., Liu, Q., Hu, X., Yu, D., & Wang, H. (2023). PP-OCRv4: Mobile scene text detection and recognition. *arXiv preprint arXiv:2310.05930*. https://arxiv.org/abs/2310.05930

Feurer, M., & Hutter, F. (2019). Hyperparameter optimization. In F. Hutter, L. Kotthoff, & J. Vanschoren (Eds.), *Automated machine learning: Methods, systems, challenges* (pp. 3-33). Springer. https://doi.org/10.1007/978-3-030-05318-5_1

He, P., Huang, W., Qiao, Y., Loy, C. C., & Tang, X. (2016). Reading scene text in deep convolutional sequences. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), 3501-3508. https://doi.org/10.1609/aaai.v30i1.10291

JaidedAI. (2020). EasyOCR: Ready-to-use OCR with 80+ supported languages. GitHub. https://github.com/JaidedAI/EasyOCR

Liang, J., Doermann, D., & Li, H. (2005). Camera-based analysis of text and documents: A survey. *International Journal of Document Analysis and Recognition*, 7(2), 84-104. https://doi.org/10.1007/s10032-004-0138-z

Liao, M., Wan, Z., Yao, C., Chen, K., & Bai, X. (2020). Real-time scene text detection with differentiable binarization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07), 11474-11481. https://doi.org/10.1609/aaai.v34i07.6812

Liaw, R., Liang, E., Nishihara, R., Moritz, P., Gonzalez, J. E., & Stoica, I. (2018). Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*. https://arxiv.org/abs/1807.05118

Mindee. (2021). DocTR: Document Text Recognition. GitHub. https://github.com/mindee/doctr

Moritz, P., Nishihara, R., Wang, S., Tumanov, A., Liaw, R., Liang, E., Elibol, M., Yang, Z., Paul, W., Jordan, M. I., & Stoica, I. (2018). Ray: A distributed framework for emerging AI

applications. *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 561-577. https://www.usenix.org/conference/osdi18/presentation/moritz

Morris, A. C., Maier, V., & Green, P. D. (2004). From WER and RIL to MER and WIL: Improved evaluation measures for connected speech recognition. *Eighth International Conference on Spoken Language Processing*. https://doi.org/10.21437/Interspeech.2004-668

PaddlePaddle. (2024). PaddleOCR: Awesome multilingual OCR toolkits based on PaddlePaddle. GitHub. https://github.com/PaddlePaddle/PaddleOCR

Pearson, K. (1895). Notes on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58, 240-242. https://doi.org/10.1098/rspl.1895.0041

PyMuPDF. (2024). PyMuPDF documentation. https://pymupdf.readthedocs.io/

Shi, B., Bai, X., & Yao, C. (2016). An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(11), 2298-2304. https://doi.org/10.1109/TPAMI.2016.2646371

Smith, R. (2007). An overview of the Tesseract OCR engine. *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, 2, 629-633. https://doi.org/10.1109/ICDAR.2007.4376991

Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W., & Liang, J. (2017). EAST: An efficient and accurate scene text detector. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5551-5560. https://doi.org/10.1109/CVPR.2017.283

Zoph, B., & Le, Q. V. (2017). Neural architecture search with reinforcement learning. *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1611.01578

Anexo A. Código fuente y datos analizados

5.3.A.1 Repositorio del Proyecto

El código fuente completo y los datos utilizados en este trabajo están disponibles en el siguiente repositorio:

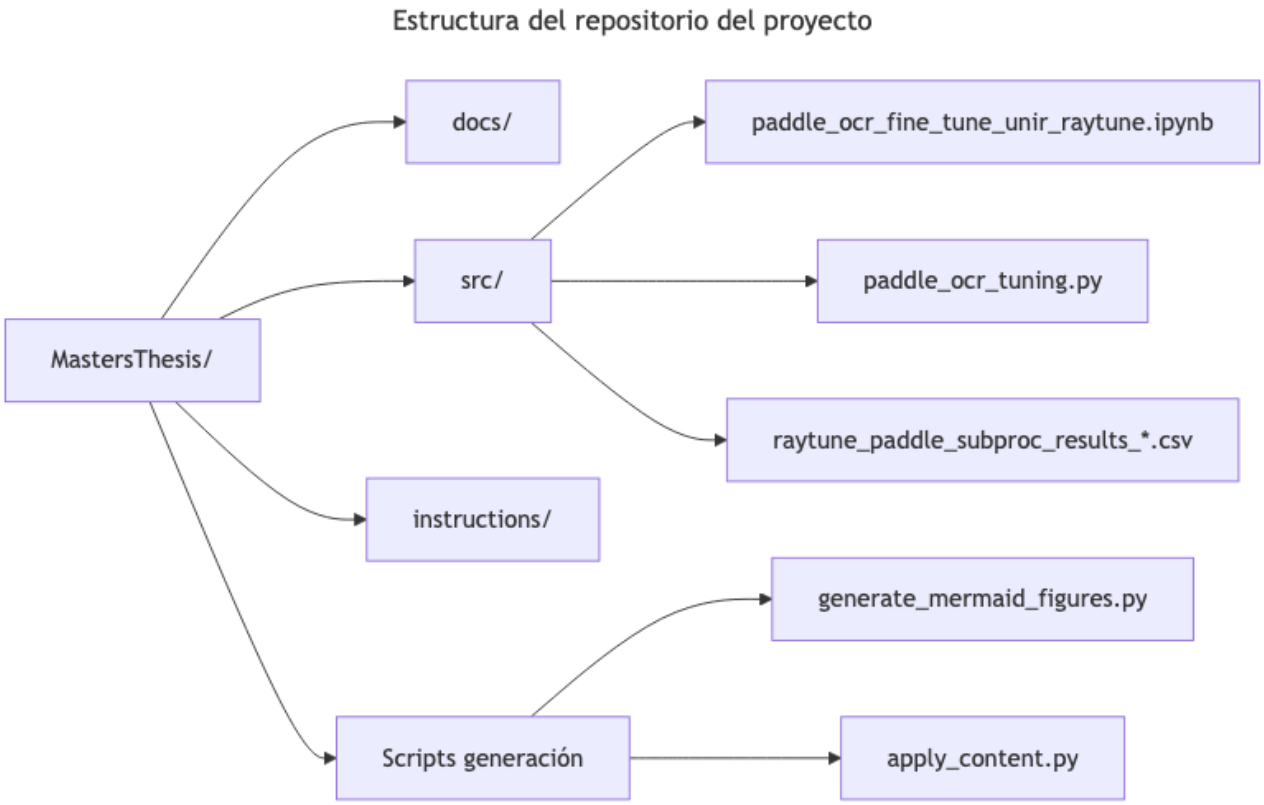
URL del repositorio: https://github.com/seryus/MastersThesis

El repositorio incluye:

- **Notebooks de experimentación**: Código completo de los experimentos realizados
- **Scripts de evaluación**: Herramientas para evaluar modelos OCR
- **Dataset**: Imágenes y textos de referencia utilizados
- **Resultados**: Archivos CSV con los resultados de los 64 trials de Ray Tune

5.4.A.2 Estructura del Repositorio

Figura 8. ***Estructura del repositorio del proyecto***



Fuente: Elaboración propia.

Descripción de componentes:

- **docs/**: Capítulos de la tesis en Markdown (estructura UNIR)
- **src/**: Código fuente de experimentación

- `paddle_ocr_fine_tune_unir_raytune.ipynb`: Notebook principal con 64 trials Ray Tune - `paddle_ocr_tuning.py`: Script CLI para evaluación OCR - `raytune_paddle_subproc_results_20251207_192320.csv`: Resultados de optimización

- **instructions/**: Plantilla e instrucciones UNIR
- **Scripts de generación**: `generate_mermaid_figures.py` y `apply_content.py` para generar el documento TFM

5.5.A.3 Requisitos de Software

Para reproducir los experimentos se requieren las siguientes dependencias:

Tabla 28. ***Tabla de datos.***

Componente	Versión
Python	3.11.9
PaddlePaddle	3.2.2
PaddleOCR	3.3.2
Ray	2.52.1
Optuna	4.6.0
jiwer	(última versión)
PyMuPDF	(última versión)

Fuente: Elaboración propia.

5.6.A.4 Instrucciones de Ejecución

1. Clonar el repositorio
2. Instalar dependencias: `pip install -r requirements.txt`
3. Ejecutar el notebook `src/paddle_ocr_fine_tune_unir_raytune.ipynb`

5.7.A.5 Licencia

El código se distribuye bajo licencia MIT.