

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN

TRABAJO DE INTEGRACIÓN CURRICULAR

TEMA:

"EXTRACCIÓN DE TEXTO A PARTIR DE IMÁGENES MEDIANTE MODELOS VLM"

Trabajo de titulación previo a la obtención del título de Ingeniero en Tecnologías de la Información

Línea de investigación: Desarrollo, aplicación de software y cyber security (seguridad cibernética)

AUTOR:

Edwin Ramiro Cordero Navarrete

DIRECTOR:

MSc. Pablo Andrés Landeta López

Ibarra – Ecuador 2026

UNIVERSIDAD TÉCNICA DEL NORTE
DIRECCIÓN DE BIBLIOTECA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO	
APELLIDOS Y NOMBRES:	CORDERO NAVARRETE EDWIN RAMIRO

DATOS DE LA OBRA	
TÍTULO:	EXTRACCIÓN DE TEXTO A PARTIR DE IMÁGENES MEDIANTE MODELOS VLM
AUTOR (ES):	CORDERO NAVARRETE EDWIN RAMIRO
FECHA: DD/MM/AAAA	28/08/2026
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	**PREGRADO X POSGRADO**
TITULO POR EL QUE OPTA:	INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN
ASESOR /DIRECTOR:	MSC. PABLO ANDRÉS LANDETA LÓPEZ

2. CONSTANCIAS

El autor (es) manifiesta (n) que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto la obra es original y que es (son) el (los) titular (es) de los derechos patrimoniales, por lo que asume (n) la responsabilidad sobre el contenido de la misma y saldrá (n) en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 28 días del mes de agosto de 2026

EL AUTOR:

(Firma)... Validar únicamente en FirmaEC.
Firmado electrónicamente por:
EDWIN RAMIRO
CORDERO NAVARRETE
.....

Nombre: Edwin Ramiro Cordero Navarrete

CERTIFICACIÓN DEL DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Ibarra, 28 de agosto de 2026.

MSc. Pablo Andrés Landeta
DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

CERTIFICA:
Haber revisado el presente informe final del trabajo de Integración Curricular, el mismo que se ajusta a las normas vigentes de la Universidad Técnica del Norte; en consecuencia, autorizo su presentación para los fines legales pertinentes.

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
PABLO ANDRES LANDETA LOPEZ

MSc. Pablo Andrés Landeta

C.C: 1002161055

APROBACIÓN DEL COMITÉ CALIFICADOR

El Comité Calificado del trabajo de Integración Curricular “ Extracción de texto a partir de imágenes mediante modelos VLM ” elaborado por Edwin Ramiro Cordero Navarrete , previo a la obtención del título del Ingeniero en Tecnologías de la Información , aprueba el presente informe de investigación en nombre de la Universidad Técnica del Norte:

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
PABLO ANDRES LANDETA LOPEZ

MSc. Pablo Andrés Landeta

C.C: 1002161055

Diego Francisco Teran Pineda
Time Stamping Security Data

PhD. Diego Francisco Terán

C.C: 1002907671

DEDICATORIA

He pasado gran parte de mi vida construyendo sistemas, automatizando el caos y levantando muros de ego y código para demostrarle al mundo que podía con todo. Si hay algo que siempre me ha caracterizado, es que no sé cuándo parar. Mi mente siempre está acelerada, buscando el siguiente problema que resolver, llevándome al límite. Pero durante este proceso entendí algo que ninguna inteligencia artificial pudo enseñarme: el verdadero éxito no es una métrica, es la gente que te sostiene cuando tu propio sistema colapsa. A mi madre, María Navarrete. Siempre hemos sido nosotros dos contra el mundo. Tú fuiste la primera en mirar detrás de todo el ruido y encontrar al hombre que, muchas veces, tenía miedo de no ser suficiente. Cada vez que caía en mis peores crisis, estabas ahí. A mi hermana, Sofía Cordero. Porque cada vez que yo dudaba y sentía que no servía para nada, tú creías en mí, incluso cuando yo no podía hacerlo. Gracias por soportar mi intensidad y por ser mi mejor amiga y mi mayor aliada. A mi novia, Tanyuri Mabel Parrales Zambrano. Cuando no sé cuándo parar, tú eres mi ancla a la realidad. Tú me enseñaste que el verdadero hogar no es un lugar, ni una pantalla brillando en la madrugada… es una persona. Antes de ti, vivía para demostrar algo. Después de ti, encontré mi verdadera fuerza. A mi abuelita Marina Flores, mi segunda madre. Gracias por ser mi refugio seguro, por esas charlas y ese çhismeo”de calidad cuando estábamos solos. Eres la pausa necesaria en mi mente, la que siempre sabe darme sabiduría cuando el mundo va demasiado rápido. Y a Wilson Hidalgo, por acompañarme en este tramo y cubrirme la espalda, ya sea con las palabras precisas o simplemente estando presente cuando el camino se volvía solitario. La gente allá afuera verá el resultado de una investigación. Pero la verdad es que ustedes me salvaron a mí. Y si algún día alguien pregunta cuál ha sido mi mayor creación, no será esta tesis, ni ninguna tecnología. Será la vida que sigo construyendo junto a ustedes.

Edwin Ramiro Cordero Navarrete

AGRADECIMIENTO

He pasado gran parte de este último tramo inmerso en arquitecturas complejas, optimizando parámetros y tratando de enseñarle a una máquina a entender el mundo de una forma que antes parecía imposible. Si algo me ha caracterizado en este proceso es que no sé cuándo parar; mi mente siempre está buscando esa última iteración, ese ajuste final que marque la diferencia entre lo funcional y lo extraordinario. Pero este viaje me ha enseñado que incluso el sistema más avanzado necesita una base sólida y un equipo de control que mantenga el rumbo.

A mi tutor, el Ing. Pablo Andrés Landeta. Gracias por ser ese soporte estratégico desde el primer día. Fuiste capaz de canalizar mi intensidad y mis ideas cuando parecían demasiado ambiciosas, dándome la guía necesaria para que esta visión no se quedara solo en un prototipo, sino que se convirtiera en una realidad académica sólida. Gracias por creer en este proyecto incluso cuando el hardware parecía decirnos que no. A mi asesor, el Ing. Teran Pineda Diego Francisco. Gracias por la supervisión crítica y por asegurar que cada engranaje de esta investigación funcionara con la precisión que un cambio de paradigma como este exige. Tu visión fue el control de calidad fundamental para que este trabajo alcanzara su máximo potencial y para recordarme que la excelencia está en los detalles. A la Universidad Técnica del Norte. Gracias por ser el escenario de mi formación y por proporcionarme el entorno necesario para desafiar los límites del OCR tradicional. Formar parte de esta institución ha sido el campo de pruebas ideal para demostrar que el futuro de la ingeniería está en nuestras manos. Al final del día, la gente verá una investigación terminada sobre modelos de visión-lenguaje, pero yo veo el esfuerzo de quienes confiaron en mi capacidad para cambiar las reglas del juego. Gracias por ayudarme a construir el futuro.

Edwin Ramiro Cordero Navarrete

ÍNDICE DE CONTENIDOS

DEDICATORIA 5

AGRADECIMIENTO 6

ÍNDICE DE FIGURAS 14

ÍNDICE DE TABLAS 15

RESUMEN 17

ABSTRACT 18

CAPÍTULO I

1.1 Planteamiento del Problema 19

1.2 Alcance 21

CAPÍTULO II

2.1 Introducción y Contextualización 24

2.2 Antecedentes Históricos 24

2.2.1 Evolución del OCR 24

2.2.2 Deep Learning en Visión Artificial 25

2.2.3 Surgimiento de Modelos de Lenguaje Visual 25

2.3 Fundamentos Técnicos 26

2.3.1 Visión Computacional para Texto en Documentos 26
2.3.1.1 Desafíos en Documentos Complejos 27
2.3.2 Modelos de Lenguaje Visual (VLM) 27
2.3.2.1 Arquitectura Encoder–Decoder 28
2.3.2.2 Modelos Representativos 29
2.3.3 Transformadores en Visión 30
2.3.3.1 Vision Transformers (ViT) 30
2.3.3.2 Fine-tuning Eficiente 30
2.3.4 OCR Moderno vs. VLM 31
2.4 Estado del Arte en Extracción con VLM 31
2.5 Métricas de Evaluación 32
2.5.1 Precisión Léxica 32
2.5.2 Exactitud Semántica 33
2.5.3 Validez de la Anotación de Referencia 33
2.5.4 Robustez y Eficiencia 34

CAPÍTULO III

3.1 Marco Metodológico 36
3.1.1 Diseño de la Investigación 36
3.1.2 Metodología de Desarrollo de Software 36
3.1.2.1 Entorno de Hardware 37
3.1.2.2 Entorno de Software y Versiones Verificadas 38
3.1.3 Tratamiento del Dataset. 39
3.1.3.1 Corpus de Imágenes 40
3.1.3.2 Construcción del Ground-Truth y Formato JSONL. 41
3.1.3.3 Preprocesamiento de Imágenes 42
3.1.4 Arquitectura del Modelo y Estrategia de Fine-Tuning 43
3.1.4.1 Modelo Base: InternVL2-2B 43
3.1.4.2 Segundo Modelo Base: Qwen2.5-VL-3B 44
3.1.4.3 Cuantización QLoRA a 4 bits y LoRA 44
3.1.4.4 Configuración del Bucle de Entrenamiento 46
3.1.4.5 Curva de Pérdida: Evidencia del Aprendizaje 47
3.1.5 Protocolo Experimental de Evaluación. 49
3.1.6 Integración del Adaptador en el Pipeline Local 51
3.1.7 Registro de Versiones: Evolución Cronológica del Desarrollo. 52
3.1.8 Descripción de las Etapas del Pipeline 53
3.1.8.1 Etapa 1 — Extracción OCR con EasyOCR 53
3.1.8.2 Etapa 2 — Razonamiento Multimodal con InternVL2-2B + LoRA 54

3.1.8.3 Etapa 3 — Consolidación y Entrega del Resultado 54

3.1.9 Diagrama de Flujo del Pipeline 54

CAPÍTULO IV

4.1 Análisis de Resultados y Discusión 56
4.1.1 Análisis de Rendimiento (Precisión). 56
4.1.1.1 Efecto del Ajuste Fino: Comparación Ablativa 59
4.1.2 Análisis de Eficiencia y Gestión de VRAM 60
4.1.2.1 El Segundo Motor Local Excede el Presupuesto de Memoria. 61
4.1.3 Estabilidad de Hardware y Gráfico de VRAM 63
4.1.4 Análisis de Latencia Cloud vs Local. 64
4.1.5 Comparativa Visual y Superioridad Semántica. 64
4.1.6 Discusión de Casos de Borde (Análisis de Errores) 65
4.1.7 Taxonomía del Error 67
4.1.8 Índice de Confianza Visual: Predicción de Viabilidad Previa a la Inferencia. . . 68
4.1.8.1 El Índice Predice el Fracaso, no el Éxito 69
4.1.8.2 Alcance de la Afirmación. 69
4.1.9 Evidencias Crudas de Ejecución 70
4.2 Síntesis Comparativa y Recomendaciones 72
4.2.1 Conclusiones del Capítulo. 73
4.2.1.1 Jerarquía de Precisión. 73
4.2.1.2 Viabilidad del Despliegue Local en Hardware de Consumo 73
4.2.1.3 Superioridad Estructural de la Salida JSON 74
4.2.1.4 Taxonomía del Error y Límites del Paradigma VLM 74
4.2.1.5 Compromiso Ingeniería-Precisión (*Trade-off*) 75

4.2.1.6 Recomendación Arquitectónica 75
CONCLUSIONES 77
4.3 Conclusiones Específicas 77
4.3.1 Evaluación de Paradigmas: OCR vs. Modelos de Visión-Lenguaje 77
4.3.2 Viabilidad Técnica y Despliegue en Hardware de Consumo 78
4.3.3 La Arquitectura Híbrida Adaptativa como Solución Óptima 79
4.3.4 Límites Operativos del Paradigma VLM 79
4.4 Contribuciones Originales 80
4.5 Limitaciones del Estudio 81
4.6 Trabajos Futuros 82
4.7 Reflexión Final 84
RECOMENDACIONES 85
4.8 Para Investigadores que Extiendan este Trabajo 85
4.8.1 Ampliar el Corpus Antes de Comparar Resultados 85
4.8.2 Completar la Comparación Ablativa. 85
4.8.3 Evaluar Esquemas de Cuantización Alternativos al NF4 86
4.8.4 Validar Externamente el Índice de Confianza Visual 86
4.9 Para Ingenieros que Implementen el Sistema en Producción 87
4.9.1 No Omitir el Protocolo de Limpieza de VRAM 87
4.9.2 Definir un Criterio Explícito de Enrutamiento por Motor 87
4.9.3 Anclar las Versiones de Dependencias con Precisión 88
4.9.4 Utilizar el Patrón Domain-Driven Design para Mantener Extensibilidad 88
4.9.5 Monitorizar la Latencia Cloud por Región y Horario 88
4.10 Para Organizaciones que Evalúen la Adopción 89

4.10.1 No Reemplazar el OCR Tradicional; Complementarlo 89
4.10.2 Dimensionar el Hardware con un Margen Mínimo del 20 % sobre el Pico de VRAM 89
4.10.3 Evaluar el Costo-Beneficio de la API Cloud Antes del Despliegue 90
4.10.4 Establecer un Proceso de Validación Humana para Documentos de Alto Valor 90
BIBLIOGRAFÍA 91
ANEXOS 98
Anexo 1: Arquitectura Domain-Driven Design 98
Anexo 2: Ciclo de Vida de la VRAM 98
Anexo 3: Fragmentos de Código Crítico. 99
Anexo 4: Diagrama de Secuencia UML del Pipeline Híbrido 105
Anexo 5: Configuración del Adaptador LoRA 106
Anexo 6: Script de Análisis Estadístico del Corpus 108

ÍNDICE DE FIGURAS

Figura I Diagrama del problema 20
Figura II Arquitectura del Sistema de Extracción de Texto Mediante VLM. 22

Figura III Proceso de extracción en VLM: El Encoder (ViT) transforma la imagen en datos vectoriales para que el Decoder los convierta en un resultado estructurado en formato JSON. 28
Figura IV Arquitectura del modelo Donut para extracción de datos OCR-free, que transforma imágenes directamente en archivos JSON estructurados mediante transformadores. 29
Figura V Arquitectura de red neuronal Pix2Struct para la conversión de documentos visuales en secuencias de tokens estructurados. 29

Figura VI Estado de VRAM tras la carga inicial de la imagen. 38
Figura VII Pico máximo de VRAM durante el procesamiento. 38
Figura VIII Muestras del dataset: Ejemplo de cartas usadas para el entrenamiento. . . 40
Figura IX Visual Studio Code mostrando el archivo dataset/train.jsonl con su estructura ChatML. 42
Figura X Descarga del modelo OpenGVLab/InternVL2-2B (4.41 GB). 44
Figura XI Captura del entorno de ejecución de Google Colab mostrando la asignación de la GPU Tesla T4. 47
Figura XII Log de entrenamiento en Google Colab mostrando la evolución de la pérdida hasta el paso 50. 49
Figura XIII La ejecución secuencial excluyente entre motores garantiza que el consumo de VRAM no supere el límite físico de 6 GB, permitiendo la reproducibilidad del estudio en hardware de consumo. 55

Figura XIV CER medio y mediano por motor de extracción sobre el conjunto de validación hold-out. Los puntos superpuestos corresponden a documentos individuales y permiten apreciar la dispersión interna de cada motor. 57
Figura XV CER documento a documento para los cinco motores. La variabilidad entre documentos es la razón por la que no se reportan pruebas de significancia con un conjunto de validación de cinco elementos. 58
Figura XVI Efecto del adaptador LoRA sobre InternVL2-2B. A la izquierda, CER con y sin adaptador por documento; a la derecha, la ganancia individual y su media. 59
Figura XVII Monitorización de la telemetría de VRAM durante el ciclo de inferencia. Se observa el pico de consumo y el efecto de la limpieza de caché. 60
Figura XVIII Precisión frente a latencia. Cada punto representa un documento individual y cada aspa la media del motor. La región inferior izquierda corresponde a la combinación óptima: menor error y menor tiempo. 62
Figura XIX Telemetría del uso de VRAM durante un ciclo de inferencia secuencial. Se observa un perfil de carga estable por debajo del techo físico de 6 GB de la NVIDIA RTX 4050, con picos controlados mediante la gestión explícita de caché de PyTorch. 63
Figura XX Comparativa lado a lado entre la mascara de deteccion de EasyOCR y la interpretacion JSON de InternVL2-2B sobre una carta manuscrita. 65

Figura XXI Análisis de límite operativo en manuscrito con ruido extremo. La pérdida irreversible de información de trazo provoca que el VLM recurra a valores de contingencia pre-entrenados. 66
Figura XXII Comparativa entre la salida cruda de EasyOCR (texto plano desordenado) y la respuesta JSON determinista de InternVL2-2B. Esta captura demuestra la diferencia en la calidad de salida, donde el VLM local entrega un objeto estructurado listo para bases de datos, mientras que el OCR requiere limpieza manual intensiva. 70
Figura XXIII Evidencia de telemetría mediante `nvidia-smi` durante la fase de carga del modelo local. Se valida que la arquitectura propuesta es compatible con hardware de consumo, permitiendo la ejecución de modelos de lenguaje de gran escala sin necesidad de infraestructura de servidor dedicada. 71
Figura XXIV Caso representativo de éxito extremo: Reconstrucción de manuscrito con caligrafía cursiva compleja y soporte físico degradado. El VLM logró una transcripción íntegra a pesar de la irregularidad de los trazos y el ruido de fondo del papel envejecido. 71
Figura XXV Diagrama de Arquitectura DDD del sistema VLM 98
Figura XXVI Detalle de la capa de aplicación y orquestación del pipeline 99
Figura XXVII Protocolo de gestión de VRAM en GPU — ciclo de vida completo . . . 100
Figura XXVIII Diagrama de secuencia UML del pipeline híbrido de extracción 105

ÍNDICE DE TABLAS

Tabla I Fases de la metodología de cascada y entregables técnicos 37
Tabla II Especificaciones de hardware utilizadas en las fases del experimento . . . 38
Tabla III Librerías y versiones verificadas del ecosistema de software 39
Tabla IV Taxonomía del corpus de imágenes 41
Tabla V Hiperparámetros de los adaptadores LoRA entrenados 45
Tabla VI Parámetros de entrenamiento y su justificación técnica 46
Tabla VII Evolución del Training Loss durante las 3 épocas 48
Tabla VIII Cronología de desarrollo y versiones del sistema 53

Tabla IX Comparativa cuantitativa de métricas de error y precisión sobre el conjunto de validación hold-out ($n = 5$). 57
Tabla X Reducción del CER respecto a EasyOCR. 57
Tabla XI Efecto del adaptador LoRA sobre InternVL2-2B. Valores de CER; la ganancia positiva indica reducción del error. 59
Tabla XII Pico de VRAM medido con `torch.cuda.max_memory_reserved()` sobre la NVIDIA RTX 4050 de 6 GB. 61
Tabla XIII Taxonomía del error detectado en el experimento. 67
Tabla XIV Correlación de Spearman de cada factor óptico con el CER medido ($n = 20$). Un valor negativo indica que el factor predice correctamente: mayor calidad óptica implica menor error. 68
Tabla XV Distribución de documentos utilizables (CER $\leq 1{,}0$) respecto al umbral de viabilidad local. 69
Tabla XVI Matriz de Decisión: Evaluación Final de Motores de Extracción. 72
Tabla XVII Desglose y justificación técnica de los parámetros del adaptador LoRA. . 107
Tabla XVIII Resumen del experimento de fine-tuning con QLoRA sobre InternVL2-2B. 108
Tabla XIX Resultados promedio de CER, WER, F1-Score, latencia y pico de VRAM por motor, sobre el conjunto de validación hold-out de 5 documentos. 112

RESUMEN

La presente investigación aborda la extracción automatizada de texto en documentos con alto grado de deterioro físico y complejidad visual, un escenario donde los sistemas de Reconocimiento Óptico de Caracteres (OCR) tradicionales presentan fallos estructurales. El objetivo fue desarrollar y evaluar un pipeline computacional híbrido basado en Modelos de Visión-Lenguaje (VLM), comparando motores locales y en la nube frente al paradigma convencional. La metodología fue aplicada y experimental, centrada en dos modelos base, InternVL2-2B y Qwen2.5-VL-3B, optimizados mediante fine-tuning QLoRA. Se construyó un corpus propio de 26 imágenes con documentos bancarios, manuscritos cursivos, recetas médicas y grafitis. La evaluación se realizó bajo un protocolo hold-out estratificado sobre cinco documentos nunca utilizados en entrenamiento, con normalización homogénea de salidas y anotación de referencia en transcripción literal. Los resultados demostraron la superioridad de los modelos de visión-lenguaje sobre EasyOCR, que registró un Character Error Rate (CER) promedio de 0.628. InternVL2-2B con adaptador LoRA redujo el error a 0.475, Qwen2.5-VL-3B alcanzó 0.254 y Gemini 2.5 Flash obtuvo el menor error del estudio con un CER de 0.193, una reducción del 69.3 % respecto de la línea base; dado el tamaño muestral, estos valores se reportan como estadística descriptiva. Se validó la viabilidad en hardware de consumo: InternVL2-2B operó con un pico medido de 4.49 GB sobre 6 GB disponibles mediante gestión secuencial excluyente de memoria, mientras que Qwen2.5-VL-3B alcanzó 8.12 GB y excedió ese presupuesto, delimitando el compromiso entre precisión y viabilidad de despliegue. El estudio aporta además un hallazgo metodológico: cuando la anotación de referencia no es una transcripción literal, las métricas de distancia de edición invierten el orden de desempeño al penalizar la fidelidad. Finalmente se propone una arquitectura híbrida adaptativa con enrutamiento gobernado por un índice de confianza visual previo a la inferencia.

Palabras clave: Modelos de Visión-Lenguaje (VLM), Fine-tuning, QLoRA, Extracción de texto, OCR, Hardware de consumo, InternVL2, Qwen2.5-VL.

ABSTRACT

This research addresses automated text extraction in documents with significant physical degradation and visual complexity, a scenario where traditional Optical Character Recognition (OCR) systems exhibit structural failures. The objective was to develop and evaluate a hybrid computational pipeline based on Vision-Language Models (VLM), comparing local and cloud-based engines against the conventional paradigm. The methodology was applied and experimental, focusing on two base models, InternVL2-2B and Qwen2.5-VL-3B, optimized through QLoRA fine-tuning. A custom corpus of 26 images was built, including bank documents, cursive manuscripts, medical prescriptions and graffiti. Evaluation followed a stratified hold-out protocol on five documents never used for training, with homogeneous output normalization and a literal-transcription reference annotation. The results demonstrated the superiority of vision-language models over EasyOCR, which recorded an average Character Error Rate (CER) of 0.628. InternVL2-2B with a LoRA adapter reduced the error to 0.475, Qwen2.5-VL-3B reached 0.254, and Gemini 2.5 Flash achieved the lowest error of the study with a CER of 0.193, a 69.3 % reduction relative to the baseline; given the sample size, these values are reported as descriptive statistics. Viability on consumer hardware was validated: InternVL2-2B operated at a measured peak of 4.49 GB out of 6 GB available through mutually exclusive sequential memory management, whereas Qwen2.5-VL-3B reached 8.12 GB and exceeded that budget, delimiting the trade-off between accuracy and deployment feasibility. The study further contributes a methodological finding: when the reference annotation is not a literal transcription, edit-distance metrics invert the performance ranking by penalizing fidelity. Finally, an adaptive hybrid architecture is proposed, with routing governed by a visual confidence index applied prior to inference.

Keywords: Vision-Language Models (VLM), Fine-tuning, QLoRA, Text extraction, OCR, Consumer hardware, InternVL2, Qwen2.5-VL.

CAPÍTULO I

INTRODUCCIÓN

Planteamiento del Problema

1.1 Planteamiento del Problema

La extracción de texto desde imágenes se ha convertido en una necesidad esencial en múltiples campos, incluyendo la digitalización de documentos, la accesibilidad para personas con discapacidad visual y la automatización de procesos administrativos, médicos y educativos [1]. Tradicionalmente, esta tarea ha sido abordada mediante tecnologías de Reconocimiento Óptico de Caracteres (OCR), las cuales han demostrado buenos resultados en condiciones controladas. Sin embargo, su desempeño se ve afectado negativamente por imágenes con ruido, baja resolución, distorsión, escritura manuscrita o múltiples idiomas [2].

En años recientes, los Modelos de Lenguaje Visual (Vision Language Models, VLM) han emergido como una solución robusta, combinando técnicas de visión por computadora con procesamiento de lenguaje natural [3]. A diferencia del OCR convencional, los VLM pueden extraer texto e interpretar su contexto semántico, incrementando la precisión y la aplicabilidad de los datos visuales extraídos [4].

Este trabajo tiene como objetivo desarrollar una aplicación que utilice modelos VLM para superar las limitaciones del OCR tradicional, logrando una conversión más eficiente y exacta del contenido visual a texto digital. Asimismo, se propone comparar ambas tecnologías en contextos reales de uso para evaluar sus diferencias en términos de precisión, eficiencia y adaptabilidad.

Hipótesis: El uso de modelos de lenguaje visual (VLM) permitirá superar las limitaciones inherentes a los sistemas OCR tradicionales, especialmente en casos donde las imágenes contienen texto manuscrito, distorsionado, de baja calidad o en varios idiomas. Al integrar visión por computadora y procesamiento de lenguaje natural, los VLM ofrecerán una extracción más precisa y contextualizada, lo que se traducirá en mejoras significativas en eficiencia, precisión y aplicabilidad del sistema, particularmente en sectores como educación, medicina, análisis documental y accesibilidad digital.

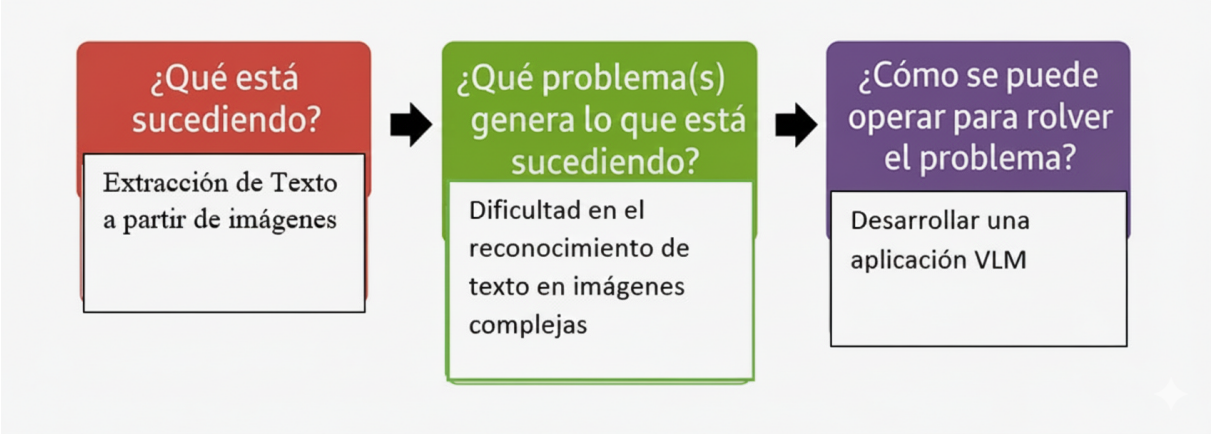


Figura I.
Diagrama del problema

Como se muestra en la Figura I, esta problemática se estructura en tres dimensiones clave y estrechamente vinculadas: la situación actual relacionada con la extracción de texto desde imágenes, las dificultades técnicas que esta situación genera y la propuesta de solución tecnológica.

En primer lugar, se identifica la necesidad de extraer texto de imágenes, lo cual plantea el reto de reconocer caracteres en contextos visuales complejos. Esta dificultad representa un obstáculo importante en tareas que requieren precisión en el reconocimiento óptico de caracteres. Ante esta situación, este estudio propone como solución el desarrollo de una aplicación basada en modelos de lenguaje visual (VLM), capaces de interpretar imágenes con mayor efectividad mediante inteligencia artificial.

Objetivos

Objetivo General

Desarrollar una aplicación que extraiga texto desde imágenes con el uso de Vision Language Models (VLM).

Objetivos Específicos

- Analizar las técnicas de extracción de texto a partir de imágenes y los modelos VLM existentes.
- Desarrollar una aplicación que extraiga texto desde imágenes con el uso de Vision Language Models (VLM).
- Evaluar la aplicación mediante métricas específicas de desempeño y compararla con sistemas OCR tradicionales.

Alcance

1.2 Alcance

El proyecto contempla el desarrollo de una aplicación capaz de procesar imágenes, identificar el texto contenido en ellas y devolver los resultados de manera precisa utilizando modelos Visual Language Models (VLM) [3]. La aplicación contará con una interfaz gráfica intuitiva que permita cargar imágenes desde archivos locales, realizar la inferencia con el modelo VLM y mostrar el texto reconocido.

El sistema será compatible con diversos formatos de imagen como `.jpg`, `.png`, y `.jpeg`, y estará preparado para operar tanto en inglés como en español. Se utilizarán modelos preentrenados de código abierto (por ejemplo, provistos por Meta, Google o Hugging Face), evitando así el entrenamiento desde cero y garantizando cumplimiento ético y legal en su uso [4].

Adicionalmente, se medirán métricas clave como precisión, exactitud y tiempo de respuesta, lo que permitirá evaluar el rendimiento del sistema y optimizarlo progresivamente [5]. Se buscará reducir el tiempo de respuesta mediante técnicas de preprocesamiento eficiente de imágenes y ejecución en entornos de alto rendimiento.

La aplicación será evaluada con documentos de uso común como cheques, facturas, boletas de pago y recibos bancarios, incluyendo documentos en PDF o escaneados como imágenes, con el objetivo de simular escenarios reales de uso en sectores como finanzas, contabilidad y seguridad documental [6], [7].



Figura II.
Arquitectura del Sistema de Extracción de Texto Mediante VLM.

La arquitectura representada describe el flujo de procesamiento de imágenes con texto mediante un modelo de lenguaje visual (VLM). El proceso inicia con la carga de la imagen, que contiene la información visual a analizar. Posteriormente, esta imagen es enviada al modelo VLM, encargado de interpretar su contenido utilizando técnicas avanzadas de inteligencia artificial. Una vez procesada, se realiza la extracción del texto, identificando y separando los elementos lingüísticos presentes en la imagen. Los resultados se presentan en una interfaz de visualización, facilitando su lectura e interpretación. Finalmente, el sistema evalúa su rendimiento mediante métricas clave como la precisión, exactitud y tiempo de respuesta, asegurando la calidad del reconocimiento y la eficiencia del sistema propuesto.

Justificación

La extracción automática de texto desde imágenes es crítica en aplicaciones como el reconocimiento de documentos, la accesibilidad para personas con discapacidad visual y el análisis multimedia. Aunque el Reconocimiento Óptico de Caracteres (OCR) ha sido la solución tradicional, sus limitaciones en entornos complejos —como texto manuscrito, baja resolución o imágenes distorsionadas— son ampliamente reconocidas [1].

En este contexto, los Modelos de Lenguaje Visual (VLM) representan un avance significativo, ya que combinan visión por computadora y procesamiento de lenguaje natural para mejorar la precisión del reconocimiento textual. Su aplicación no solo optimiza procesos en ámbitos educativos, médicos y administrativos, sino que también puede extenderse a áreas como la ciberseguridad, permitiendo la extracción de texto incrustado en imágenes potencialmente sensibles [3].

Diversos estudios han demostrado las ventajas de los modelos visual-lingüísticos sobre el OCR convencional, incluyendo mejoras en exactitud, reducción de errores y mayor adaptabilidad en contextos reales. Investigaciones recientes sustentan el potencial de esta tecnología para superar los desafíos actuales, destacando la eficiencia y capacidad de razonamiento de los nuevos modelos [8], [9].

La implementación de un sistema basado en VLM es, por tanto, una solución innovadora, factible y alineada con las tendencias actuales en inteligencia artificial, análisis de documentos y transformación digital, lo cual justifica plenamente la realización de este trabajo de titulación.

CAPÍTULO II

MARCO TEÓRICO

2.1 Introducción y Contextualización

La Document AI o Inteligencia Artificial para Documentos aparece como una disciplina que combina visión computacional, procesamiento del lenguaje natural y aprendizaje automático con un objetivo claro, interpretar, extraer y analizar información de documentos, tanto los escaneados como los digital [10]. Lo que realmente la define es partir de una idea sencilla pero potente, simple secuencia lineal de caracteres, sino un objeto semántico complejo donde la disposición espacial (el layout), el tipo de letra y los elementos multimodales (texto, imágenes, tablas), construyen significado de manera jerárquica. Si lo tratamos como texto plano perdemos todo el contexto y la extracción profunda de información clave (Key Information Extraction - KIE) queda incompleta. El giro más importante del campo ha sido dejar atrás los sistemas clásicos de Reconocimiento Óptico de Caracteres (OCR), que básicamente copiaban carácter por carácter y daban coordenadas, para dar paso a los Modelos de Lenguaje Visual (VLM). El OCR entrega una transcripción ordenada y mecánica, pero un VLM hace algo muy distinto, lee el documento como un todo integrado y capta las relaciones semánticas entre elementos que están separados en la página. Esa habilidad resulta imprescindible en casos reales como facturas, recibos o cheques, donde el valor de un campo como “Total” solo se entiende mirando su contexto visual la cercanía a “Subtotal” o “Impuestos”, y no sirve de nada saber únicamente dónde está situado en la imagen [11].

2.2 Antecedentes Históricos

2.2.1 Evolución del OCR

La evolución del OCR ha pasado de los enfoques más primitivos como la coincidencia de patrones o características hechas a mano a sistemas que se apoyan por completo en redes neuronales profundas.

Tesseract es un buen ejemplo de ese salto, empezó en los laboratorios de HP y hoy lo mantiene Google, una de sus grandes aportaciopnes fue introducir redes LSTM (Long Short-Term Memory) para tratar el texto como secuencia, lo que supuso una mejora sustancial en la precisión cuando se trabaja con líneas continuas de caracteres [12]. Por otro lado, las herramientas comerciales como ABBYY FineReader llevan años ofreciendo soluciones muy confiables que integran segmentación sofisticada, reconocimiento puro y corrección basada en reglas lingüísticas durante el post-procesamiento [13]. Sin embargo, todos estos sistemas clásicos arrastran un problema estructural de base, operan como pipelines modulares y rígidos (detección de texto $\rightarrow$ reconocimiento $\rightarrow$ post-procesamiento), de forma que cualquier fallo en una fase contamina inevitablemente las siguientes y sobre todo apenas tienen comprensión semántica del documento como entidad global [1].

2.2.2 Deep Learning en Visión Artificial

El cambio de paradigma que supuso el *deep learning* en visión computacional tuvo su origen con AlexNet [14] y abrió la puerta a tratar las imágenes como datos ricos en significado. Las Redes Neuronales Convolucionales (CNN) se consolidaron entonces como los extractores de características por excelencia, aprenden por sí solas desde los detalles más básicos (bordes, texturas) hasta conceptos complejos como formas y objetos completos. Más tarde ResNet [15] resolvió el eterno problema del gradiente que se desvanecía al entrenar redes muy profundas, lo que permitió construir modelos mucho más profundos y potentes sin que el aprendizaje se estancara. Desde entonces estas arquitecturas se han convertido en el backbone (red troncal) de referencia para prácticamente cualquier tarea de detección y segmentación de texto en imágenes [16], [17].

2.2.3 Surgimiento de Modelos de Lenguaje Visual

La verdadera convergencia entre visión y lenguaje natural supuso un punto de inflexión decisivo. La clave está en aprender representaciones compartidas embeddings dentro de un espacio multimodal donde los mismos conceptos (ya aparezcan en una imagen o en un texto) queden alineados en posiciones cercanas. CLIP (Contrastive Language-Image Pre-training) [18] fue quien demostró de forma contundente el potencial del aprendizaje contrastivo cuando se aplica a escala masiva, millones de pares imagen-descripción textual entrenados para acercarse o alejarse según corresponda. Gracias a esto los modelos

adquirieron una comprensión semántica mucho más profunda que la simple detección de píxeles y abrieron el camino a sistemas generativos capaces de "leer" una imagen y responder preguntas razonadas sobre lo que muestra [19], [20].

2.3 Fundamentos Técnicos

2.3.1 Visión Computacional para Texto en Documentos

En el terreno de la Document AI, el preprocesamiento tradicional de imágenes (corrección de perspectiva, binarización, realce de contraste o filtrado de ruido) siempre ha sido un paso casi obligatorio para que los sistemas OCR clásicos pudieran trabajar con cierta garantía [21]. Actualmente la dirección que toman los Modelos de Lenguaje Visual (VLM) es muy distinta, se busca reducir al mínimo o directamente prescindir de esos pipelines rígidos de preprocesamiento. Los modelos más avanzados en la actualidad aprenden de millones de pares de imágenes y texto obtenidos en situaciones reales (imágenes tomadas con teléfonos, ángulos poco comunes, mala iluminación, y mucho ruido), lo que les permite desarrollar una gran resistencia ante distorsiones geométricas, variaciones de luz o resoluciones bajas. Este enfoque de extremo a extremo les facilita procesar la imagen tal como llega, capturando al mismo tiempo el texto y su contexto espacial sin necesidad de intervención manual. Investigaciones recientes muestran que en entornos razonablemente controlados, omitir o reducir significativamente el preprocesamiento no solo mantiene la precisión, sino que en ocasiones la mejora al evitar los artefactos que pueden surgir de umbrales demasiado estrictos o correcciones excesivas.[22], [23]. Cuando nos encontramos con casos extremos (archivos históricos bastante deteriorados, documentos con manchas graves o fotos tomadas en condiciones de luz muy deficientes), la mejor práctica sigue siendo combinar un preprocesamiento ligero y selectivo con un VLM de última generación. La meta ya no consiste en obtener una sucesión lineal y rígida, sino en adoptar un paradigma híbrido y versátil donde el preprocesamiento funcione como un soporte complementario y no como una condición imperativa [24], [25].

2.3.1.1 Desafíos en Documentos Complejos

Los documentos con los que lidiamos todos los días (facturas donde un sello tapa palabras clave, recibos con fondos llenos de texturas o escaneos repletos de ruido) ponen en aprietos a los detectores de texto clásicos basados en CNN, como EAST [26]. Estos modelos miran la imagen de forma básicamente local, examinan parcelas concretas buscando patrones que se parezcan a letras y por eso son propensos a errores por cualquier cosa que imite caracteres, ya sea ruido, marcas de agua o diseños repetitivos. Los Transformers sin embargo evitan en gran parte estos errores gracias a su atención global. Al procesar la imagen entera de una sola vez y calcular la importancia relativa de cada zona en función del resto, el modelo puede hacer caso omiso de partes que no aportan nada (un sello grande que cubre texto, por citar un ejemplo típico) y dedicar su capacidad a las regiones que sí contienen información legible, aunque esté parcialmente tapada o deteriorada [27]. Esa visión de conjunto es precisamente lo que hace que los sistemas basados en Transformer sean bastante más confiables en la práctica cotidiana, donde las imágenes impecables brillan por su ausencia [7].

2.3.2 Modelos de Lenguaje Visual (VLM)

Lo que define a estos modelos es su flujo end-to-end, que deja atrás cualquier etapa intermedia externa. El *encoder* (prácticamente siempre un Vision Transformer (ViT) o alguna versión mejorada como Swin Transformer) coge la imagen tal cual llega y la trocea en parches para obtener una secuencia de *embeddings* visuales de alta dimensión. Estos *latents* no solo capturan detalles locales, sino que también registran las relaciones espaciales a lo largo de toda la página, lo que permite preservar la estructura del diseño del documento [28], [29]. A continuación, el decodificador (que generalmente es un modelo Transformer de lenguaje autoregresivo previamente entrenado, similar a las versiones de BART o T5) recibe esa secuencia visual después de adaptarla al formato del texto. Luego, la salida se genera de manera secuencial, token por token, siempre apoyándose en la representación de la imagen para orientarse. De este modo, el modelo establece un enlace directo desde los píxeles originales hasta una salida semántica bien organizada, ya sea en forma de JSON con los campos clave extraídos, una descripción narrativa o respuestas directas a consultas específicas sobre el contenido del documento [30], [31]. Justo esta unión tan ajustada es la que elimina la dependencia de módulos independientes habituales, como detectores externos de texto, el OCR tradicional o ajustes posteriores y permite manejar tareas complejas

de comprensión de documentos en un único proceso. Además, que el modelo pueda ofrecer resultados estructurados de manera inmediata (por poner un ejemplo, `{"numero_factura": "F-2025-00147", importe_total": "845,90 €"}`) resulta muy práctico en escenarios reales de Document AI.

Conviene precisar el alcance de esta capacidad, porque condiciona el diseño experimental del presente trabajo. La emisión de un esquema estructurado es una *propiedad del paradigma*, no necesariamente el objetivo de toda tarea de extracción. Cuando lo que se evalúa es la fidelidad de lectura sobre documentos deteriorados, exigir al modelo un formato de salida predefinido introduce una variable ajena al fenómeno medido: un motor puede resultar penalizado por su forma de responder y no por su capacidad de leer. Por esa razón, la evaluación de este trabajo se realiza sobre transcripción en texto plano, según se detalla en el Capítulo 3. En ellos, la mezcla de exactitud con la capacidad para afrontar variaciones visuales típicas del día a día, como fotos inclinadas, sombras o fondos con interferencias, es lo que verdaderamente separa un prototipo experimental de un sistema preparado para entornos de producción [32].

2.3.2.1 Arquitectura Encoder–Decoder

El encoder, generalmente un Vision Transformer (ViT), toma como entrada la imagen y la convierte en una serie de latentes o embeddings visuales de alta dimensión. El decoder, que es un Transformer de lenguaje autoregresivo, utiliza esta secuencia para generar tokens de texto de manera condicional, uno a la vez. Este proceso de extremo a extremo permite que el modelo aprenda directamente la relación entre los píxeles de la imagen y una representación semántica estructurada (por ejemplo, un JSON con campos extraídos) [33].

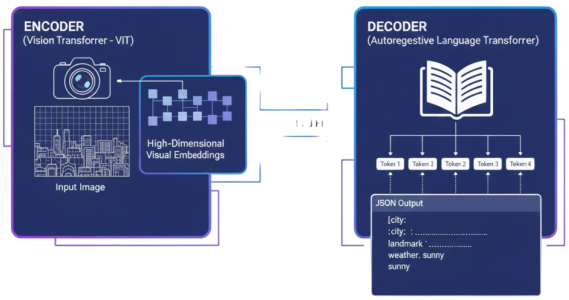


Figura III.
Proceso de extracción en VLM: El Encoder (ViT) transforma la imagen en datos vectoriales para que el Decoder los convierta en un resultado estructurado en formato JSON.

2.3.2.2 Modelos Representativos

Donut [7] fue el primero en plantear un enfoque de comprensión documental completamente libre de OCR. El modelo entrena un Transformer (con Swin como encoder y BART como decoder) directamente sobre pares imagen-texto serializado, sin recurrir en ningún momento a motores OCR externos, ni durante el entrenamiento ni en inferencia. Al eliminar esa dependencia se evita la propagación de errores típica de los pipelines tradicionales y se consigue una extracción de información mucho más sensible al contexto global del documento.

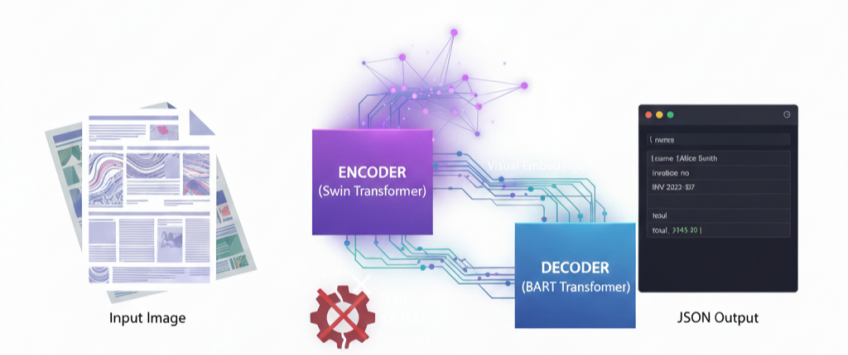


Figura IV.
Arquitectura del modelo Donut para extracción de datos OCR-free, que transforma imágenes directamente en archivos JSON estructurados mediante transformadores.

Por su parte **Pix2Struct** [34] introdujo un preentrenamiento basado en un "Reading Curriculum", en el que el modelo aprende progresivamente a interpretar elementos visuales estructurados (documentos, capturas de pantalla, infografías) asociando regiones específicas de la imagen con secuencias de tokens en el decoder. Esta estrategia está pensada especialmente para tareas donde la disposición espacial del texto juega un papel decisivo en su significado [35].

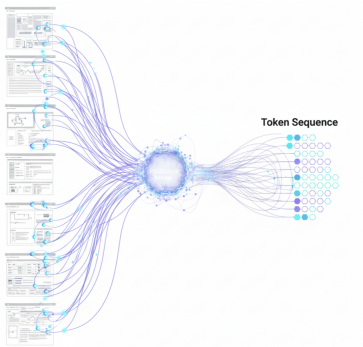


Figura V.
Arquitectura de red neuronal Pix2Struct para la conversión de documentos visuales en secuencias de tokens estructurados.

2.3.3 Transformadores en Visión

2.3.3.1 Vision Transformers (ViT)

El Vision Transformer [28] parte de una idea aparentemente sencilla,, dividir la imagen en una cuadrícula de parches fijos y no superpuestos (por ejemplo, de 16×16 píxeles). Cada parche se aplana, se proyecta linealmente a un vector de alta dimensión (el equivalente a un token) y se le suma una codificación posicional que indica su ubicación original en la imagen. Con eso se obtiene una secuencia de tokens que se introduce directamente en un Transformer estándar, tal como se hace con palabras en tareas de procesamiento del lenguaje natural. De esta forma la imagen deja de tratarse como una matriz bidimensional que requiere convoluciones y pasa a ser una secuencia lineal, lo que permite aprovechar sin modificaciones la arquitectura que había revolucionado el NLP. Este enfoque, aunque conceptualmente simple, demostró ser extremadamente competitivo frente a las CNN tradicionales, especialmente cuando se dispone de grandes volúmenes de datos para preentrenamiento [36].

2.3.3.2 Fine-tuning Eficiente

El fine-tuning completo de modelos VLM de gran escala —que en las arquitecturas contemporáneas van desde los miles de millones hasta las decenas de miles de millones de parámetros— resulta prácticamente inviable desde el punto de vista computacional en hardware de consumo. Técnicas de *Parameter-Efficient Fine-Tuning* (PEFT), entre las que destaca *Low-Rank Adaptation* (LoRA) [37], ofrecen una solución elegante, permiten adaptar estos modelos a tareas concretas (por ejemplo, extraer campos específicos de facturas con un formato determinado) actualizando únicamente un porcentaje mínimo de parámetros. LoRA funciona inyectando módulos de rango bajo en las capas de atención del modelo, mientras mantiene congelados los pesos originales. De esta forma se reduce de manera drástica tanto el consumo de memoria como el tiempo necesario para el entrenamiento. Esta eficiencia explica su rápida adopción tanto en entornos industriales como en investigación académica, ya que hace posible personalizar modelos masivos sin necesidad de recursos computacionales extremos [38], [39].

2.3.4 OCR Moderno vs. VLM

La diferencia principal entre estos enfoques se encuentra en su enfoque básico, los sistemas OCR se dedican a transcribir el texto de manera literal, con el fin de identificar cada letra correctamente [7], [22]. Un VLM, en cambio, interpreta el contenido y extrae su significado esencial. Un sistema OCR convencional produce una salida básica como “Total: $150.00” junto con las coordenadas de la zona donde aparece (x1,y1,x2,y2) [12]. Los modelos VLM generan de forma directa

```
{"field": "total_amount", "value": "150.00", "currency": "USD"}
```

Esta capacidad para interpretar el significado (y no limitarse a reproducir caracteres) es lo que convierte a los VLM en la opción más adecuada cuando se manejan facturas con formatos que varían frecuentemente. En situaciones así, la ubicación fija de un campo no ayuda mucho, y resulta necesario inferir el importe, la moneda o el número de factura considerando el conjunto completo, como la apariencia de la página y las palabras que lo rodean [40], [41]. Esta interpretación contextual y semántica representa la ventaja clave que brindan los VLM en la obtención de datos de documentos visuales complicados y diversos [3].

2.4 Estado del Arte en Extracción con VLM

Hoy el campo muestra bastante variedad. Por un lado están los sistemas híbridos, como **DocTR** [42], que unen detectores y reconocedores de texto mediante aprendizaje profundo con fases posteriores de procesamiento de lenguaje natural para sacar información clave (KIE). Otro ejemplo es **LayoutLMv3** [43], un modelo multimodal entrenado previamente que mezcla representaciones de texto, detalles de disposición (coordenadas ajustadas) e imagen, usando actividades unificadas de *Masked Language Modeling* y *Masked Image Modeling*. Aunque es potente, aún depende de un OCR externo para obtener el texto y las cajas delimitadoras [44]. En el otro extremo, los métodos sin OCR en absoluto se basan en modelos generativos independientes. Casos como **LLaVA** [30] (adaptado para documentos) o la línea **Donut/Pix2Struct** muestran que se puede lograr una comprensión total del documento sin usar OCR [7]. Los trabajos más nuevos de CVPR 2024 e ICCV 2024 señalan hacia los **Unified Document Models** (UDM), que van más allá de solo extraer datos: crean respuestas, hacen resúmenes y llevan a cabo razonamiento elaborado [45]. Con estos se elimina completamente la necesidad de OCR externos y se reúnen

todas las funciones de Document AI en un solo proceso integral [46], [47]. Sin embargo, los VLM se enfrentan a problemas éticos importantes, como el sesgo que viene de los datos de entrenamiento, que podría mantener desigualdades sociales al dar prioridad a representaciones comunes en bases de datos grandes, llevando a resultados discriminatorios en actividades de visión-lenguaje [48], [49]. Además, tienen restricciones en adaptarse a lenguas menos comunes, donde la falta de datos multilingües en el entrenamiento previo causa un desempeño más bajo en entornos con idiomas de pocos recursos, impactando la disponibilidad mundial de estas herramientas [50]. Para documentos muy específicos, los VLM suelen tener dificultades para manejar cambios visuales complicados o estructuras particulares del área, lo que limita su uso en sectores como el médico o legal, donde la exactitud semántica y contextual es esencial [51].

2.5 Métricas de Evaluación

2.5.1 Precisión Léxica

Para medir cómo de buena es la transcripción pura (ya sea por caracteres o palabras), las métricas usuales son:

- **CER (Character Error Rate):** Calcula el porcentaje de errores en caracteres (inserciones, borrados o cambios) en relación al total de caracteres de la referencia. Es la más dura para juzgar sistemas OCR clásicos y se saca con CER $= \frac{S+D+I}{N}$, donde S son cambios, D borrados, I inserciones y N el total de caracteres en la referencia [52].
- **WER (Word Error Rate):** Igual que el CER pero para palabras. Es tan estricta como aquella y se aplica a menudo para ver la precisión en transcripciones de palabras en documentos [53].

Aparte de CER y WER, en trabajos de crear texto desde imágenes a veces se usan métricas de similitud en lenguaje como **BLEU** (Bilingual Evaluation Understudy) [54] y **ROUGE** (Recall-Oriented Understudy for Gisting Evaluation) [55]. Pero estas están hechas para medir similitudes en secuencias de traducción automática o resúmenes, y no siempre se ajustan bien a la precisión en sacar campos específicos de documentos, porque castigan variaciones pequeñas en palabras que no cambian el sentido semántico de los valores sacados [56].

2.5.2 Exactitud Semántica

En trabajos de sacar información clave (KIE), métricas de similitud textual como BLEU o ROUGE no sirven mucho, porque miran la coincidencia en secuencia y no si se han pillado bien los campos importantes. La métrica clave aquí es el **F1-score a nivel de campo** (Field-level F1) [57]. Se cuenta como predicción buena cuando tanto el nombre del campo como el valor sacado coinciden del todo (o con un margen semántico razonable) con la referencia anotada. Por ejemplo en una factura, el modelo tiene que pillar el campo "Total" con el valor "$150.00" para que valga como acierto. Esta forma de evaluar muestra mejor el valor real del sistema en usos prácticos donde lo clave no es copiar todo el texto, sino conseguir los datos estructurados importantes sin fallos [58], [59]. Para datos complejos con muchos campos y conexiones, también se pueden usar métricas de Exactitud de Entidad (Entity Accuracy) o F1 por tipo de entidad [60].

Estas métricas presuponen, sin embargo, documentos con una estructura de campos conocida de antemano, como facturas o formularios. En corpus heterogéneos de documentos deteriorados —cartas manuscritas, notas personales, inscripciones murales— no existe un conjunto de campos común que permita definirlas, por lo que la evaluación debe apoyarse en las métricas de precisión léxica descritas en el apartado anterior.

2.5.3 Validez de la Anotación de Referencia

Todas las métricas anteriores comparten un supuesto que rara vez se hace explícito: que la anotación de referencia constituye una representación fiel de lo que el documento contiene. Cuando ese supuesto no se cumple, la métrica deja de medir lo que se pretende y pasa a medir la distancia respecto de un criterio de anotación arbitrario.

El caso más problemático se presenta en corpus multilingües. Las métricas basadas en distancia de edición —CER y WER— calculan el número mínimo de operaciones necesarias para transformar la hipótesis en la referencia. Si la referencia no es una transcripción literal sino una traducción o una interpretación del contenido, cada carácter correctamente transcrito en el idioma original se contabiliza como inserción, y cada término presente en la referencia pero ausente del documento como eliminación. El

efecto es contraintuitivo pero sistemático: **la métrica penaliza la fidelidad de transcripción y premia la generación de texto en el idioma de la referencia**, aunque ese texto no corresponda a lo escrito en la imagen.

El fenómeno afecta de manera desigual a los distintos paradigmas. Un motor OCR clásico, incapaz de traducir, resulta comparativamente favorecido frente a un modelo generativo que transcribe con fidelidad en el idioma original. La consecuencia es que el orden de desempeño observado puede invertirse respecto del real.

La literatura sobre evaluación de reconocimiento de texto asume habitualmente corpus monolingües con anotación literal, condición bajo la cual el problema no se manifiesta [52], [61]. En corpus heterogéneos, en cambio, la definición explícita del criterio de anotación —literalidad, tratamiento del texto ilegible, conservación de la disposición— constituye una decisión metodológica tan determinante como la elección de la métrica. Este trabajo documenta un caso concreto de este fenómeno y su corrección en el Capítulo 3.

2.5.4 Robustez y Eficiencia

En entornos reales de producción, tres cosas son especialmente importantes:

- **Latencia:** El tiempo que toma procesar una página (un objetivo lógico es menos de 1000 ms hasta en hardware normal). Incluye el preprocesamiento (si hay), la inferencia del modelo y el postprocesamiento de la salida.
- **Throughput (FPS):** La cantidad de páginas que el sistema puede procesar por segundo es fundamental para gestionar grandes volúmenes de documentos.
- **Consumo de VRAM:** Este es un factor clave para determinar si el modelo es adecuado para GPUs de gama media o baja, lo que es esencial para minimizar costos en la producción. Modelos con arquitecturas eficientes como Swin Transformer [29]o técnicas de compresión como la cuantización son extremadamente útiles en este sentido.

Además, es importante considerar el costo computacional total (en FLOPs y número de parámetros) para

hacer una comparación justa con soluciones OCR tradicionales. Estas tienden a ser mucho más ligeras en recursos, pero tienen capacidades limitadas en tareas de comprensión semántica compleja. [62]. La eficiencia en el fine-tuning también es crucial, donde técnicas como LoRA [37] permiten adaptar modelos grandes con un incremento mínimo en parámetros y tiempo de entrenamiento. Al final, la robustez se debe probar con variaciones como rotaciones, cambios de luz, ruido y degradación de imagen, usando datasets hechos para eso [61], [63].

CAPÍTULO III

MATERIALES Y MÉTODOS

3.1 Marco Metodológico

3.1.1 Diseño de la Investigación

La investigación es de carácter aplicado y experimental, y sigue un enfoque cuantitativo. El estudio se enfoca en la creación de un pipeline para la extracción de texto que combina un motor OCR convencional (EasyOCR), dos modelos de lenguaje visual de ejecución local optimizados mediante fine-tuning con QLoRA (InternVL2-2B y Qwen2.5-VL-3B) y un modelo comercial en la nube (Gemini 2.5 Flash). El experimento consta de cuatro fases sucesivas: (1) elaboración de un corpus propio de 26 imágenes con anotación de referencia en transcripción literal, (2) definición de una partición hold-out estratificada que reserva cinco documentos para validación, (3) entrenamiento de los adaptadores LoRA sobre infraestructura en la nube con GPU Tesla T4, y (4) evaluación en un entorno local (GPU RTX 4050 de 6 GB de VRAM) que recoge métricas de precisión, latencia y consumo de memoria para las cinco configuraciones bajo un protocolo unificado. Esta estructura garantiza la reproducibilidad del experimento, cumpliendo con los estándares de investigación de la Facultad de Ingeniería en Ciencias Aplicadas (FICA).

3.1.2 Metodología de Desarrollo de Software

Para la ejecución de este proyecto se seleccionó el modelo de desarrollo en Cascada (Waterfall). Esta decisión se fundamentó en que los requerimientos técnicos y los objetivos del experimento computacional fueron estables y estuvieron definidos con rigor desde la fase de anteproyecto. Al tratarse de una investigación experimental donde la arquitectura del sistema debe permanecer inalterada para garantizar la validez de las comparativas, el enfoque lineal permitió asegurar la integridad de los datos y una validación científica paso a paso.

Las fases ejecutadas se describen a continuación:

1. **Análisis de Requerimientos:**Identificación de las limitaciones del hardware (6 GB VRAM) y establecimiento de los criterios para seleccionar el modelo InternVL2-2B.
2. **Diseño del Sistema:** La arquitectura se definió usando los principios de Domain-Driven Design (DDD). El objetivo principal es aislar la lógica de negocio de cualquier adaptador de infraestructura. Esto aplica directamente para la integración de motores OCR y modelos VLM.
3. **Implementación:** Se preparó el entorno de trabajo para ejecutar el ajuste fino mediante QLoRA. Todo el código de los módulos se escribió en Python. Se decidió implementar patrones de diseño específicos orientados a mantener un bajo nivel de acoplamiento.
4. **Pruebas de Rendimiento:** El modelo pasó por distintos benchmarks de inferencia empíricos. Interesa medir el impacto en hardware evaluando la latencia y el consumo de VRAM. La precisión de los resultados se valida calculando el F1-score, junto con las métricas WER y CER.
5. **Cierre y Resultados:** La metodología termina documentando cada hallazgo obtenido en las pruebas. Se estructuró un análisis cuantitativo formal. Los datos finales demuestran la capacidad real del sistema para procesar y ejecutar extracción estructurada.

Tabla I.
Fases de la metodología de cascada y entregables técnicos

Fase	Entregable Principal
Análisis	Informe de restricciones técnicas y modelo base seleccionado.
Diseño	Arquitectura Domain-Driven Design (DDD) y esquemas de dominio.
Implementación	Adaptadores LoRA y Pipeline de integración en Python.
Pruebas	Benchmark de métricas (CER, WER, F1) y logs de inferencia.
Cierre	Reporte final de eficiencia y análisis de desempeño.

3.1.2.1 Entorno de Hardware

Para garantizar la viabilidad empírica y su correspondiente replicabilidad, fue imperativo delimitar el ecosistema de hardware con rigor taxonómico. En la Tabla II se detallan los componentes utilizados, destacando una asimetría deliberada entre el entorno de entrenamiento y el de inferencia.

Tabla II.
Especificaciones de hardware utilizadas en las fases del experimento

Componente	Especificación
GPU Local (Inferencia)	NVIDIA GeForce RTX 4050 (Mobile)
VRAM Local	6 GB GDDR6
GPU Remota (Entrenamiento)	NVIDIA Tesla T4 · 16 GB GDDR6 (Google Colab)
Sistema Operativo	Windows 11 (Inferencia) / Linux Ubuntu 22.04 (Training)

Esta configuración responde a la decisión arquitectónica central de este trabajo: el çuello de botellaïmpuesto por los 6 GB de VRAM local. Técnicamente, resulta **topológicamente imposible** ejecutar un ciclo completo de retropropagación sobre un VLM de 2 mil millones de parámetros en la estación de trabajo local; por ello, el entrenamiento se delegó a la nube, mientras que la inferencia local se viabilizó mediante técnicas de cuantización obligatoria.

Para comprobar esta conducta, se realizó un seguimiento del uso de recursos durante la ejecución del pipeline en la GPU RTX 4050. Tal y como se muestra en las figuras, la administración de la memoria es fundamental: desde la carga inicial del modelo (Figura VI) hasta los picos de procesamiento final (Figura VII), el sistema trabaja casi al límite físico del hardware, lo que subraya la necesidad de una optimización paramétrica.

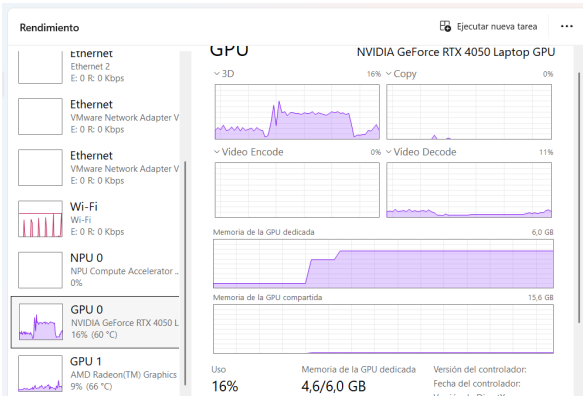


Figura VI.
Estado de VRAM tras la carga inicial de la imagen.

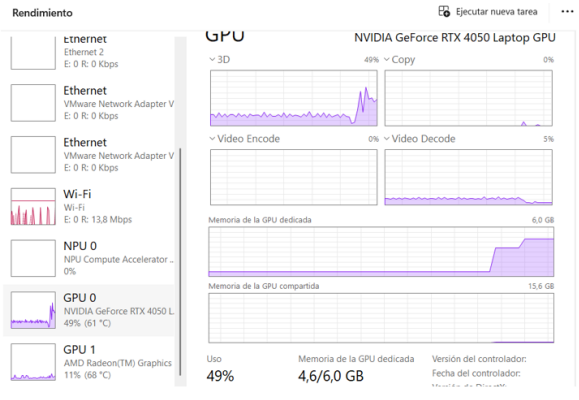


Figura VII.
Pico máximo de VRAM durante el procesamiento.

3.1.2.2 Entorno de Software y Versiones Verificadas

La reproducibildad científica de un experimento computacional depende directamente de la trazabilidad de las dependencias de software. Cualquier variación en las versiones de las librerías puede alterar

radicalmente los resultados, especialmente en el ecosistema de los modelos de lenguaje donde los cambios de API son frecuentes y a menudo rompen la compatibilidad con arquitecturas personalizadas como InternVL2-2B. Por esta razón, el entorno de software fue fijado con precisión y validado empíricamente durante el proceso de desarrollo.

Cada componente del stack tecnológico fue seleccionado con un rol específico y no intercambiable dentro del pipeline: Python 3.12.10 como lenguaje de orquestación, PyTorch como motor de cómputo tensorial acelerado por GPU mediante CUDA 12.4, el ecosistema Hugging Face para la gestión del modelo base, y las librerías **peft** y **bitsandbytes** como pilares matemáticos del fine-tuning con restricción de memoria. La Tabla III detalla el ensamblaje completo de versiones que garantiza la replicación exacta del experimento:

Tabla III.
Librerías y versiones verificadas del ecosistema de software

Librería	**Versión**	**Rol en el Pipeline**
Python	3.12.10	Lenguaje de orquestación
PyTorch	2.6.0+cu124	Motor tensorial con CUDA 12.4
transformers	5.3.0	Carga del modelo VLM base
`peft`	**0.18.1**	Inyección y carga de adaptadores LoRA
`bitsandbytes`	0.44.0	Cuantización a 4 bits (QLoRA)
`trl`	0.10.1	Framework SFTConfig para entrenamiento
`datasets`	—	Carga y preprocesamiento del corpus JSONL
`torchvision`	—	Transformación de imágenes para InternVL2-2B
EasyOCR	—	Motor OCR tradicional (Etapa 1)

3.1.3 Tratamiento del Dataset

El insumo fundamental de cualquier proceso de *fine-tuning* es la calidad y representatividad del conjunto de datos de entrenamiento. A diferencia de los conjuntos estandarizados como CORD o FUNSD, frecuentemente usados en la literatura pero construidos bajo condiciones ideales de digitalización, el corpus de esta investigación fue diseñado deliberadamente para reflejar los **retos visuales del mundo real**.

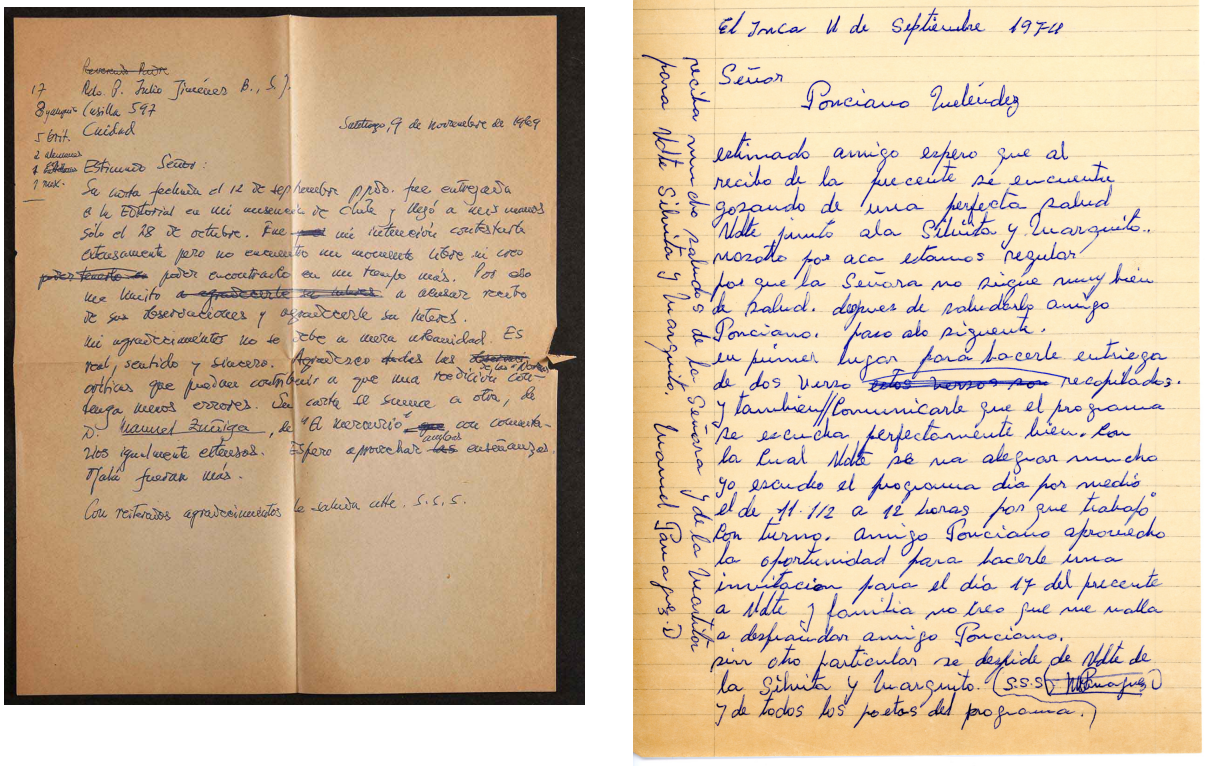


Figura VIII.
Muestras del dataset: Ejemplo de cartas usadas para el entrenamiento.

3.1.3.1 Corpus de Imágenes

El dataset fue construido manualmente a partir de una colección propia de **26 imágenes** representativas de los retos reales de extracción de texto. La taxonomía abarca:

Tabla IV.
Taxonomía del corpus de imágenes

Categoría	Ejemplos Incluidos	Retos Visuales
Documentos bancarios	Cheques UCO Bank, HDFC, SBI, Kotak Mahindra	Sellos superpuestos, múltiples idiomas, firmas
Cartas manuscritas	Correspondencia en español e inglés	Tipografía cursiva, degradación del papel
Recetas médicas	Prescripciones con dosis y posología	Abreviaciones técnicas, caligrafía médica
Grafitis y murales	Arte urbano callejero latinoamericano	Tipografía decorativa, fondos texturizados
Notas contables	Cuadernos de cuentas diarias	Texto denso, columnas aritméticas
Documentos técnicos	Notas de telefonía IP, diagramas	Terminología especializada

3.1.3.2 Construcción del Ground-Truth y Formato JSONL

Para cada imagen se elaboró la **Verdad Base** (*Ground-Truth*) y se almacenó en `dataset/train.jsonl`, con un registro por documento y el esquema `{id, image, instruction, output}`.

Criterio de anotación. La referencia es una **transcripción literal**: reproduce lo que está escrito, en el idioma en que está escrito, conservando los saltos de línea, sin resumir, sin traducir y sin describir la estructura del documento. Cuando una porción del texto no puede recuperarse con evidencia visual suficiente se anota únicamente la marca `[ILEGIBLE]`.

Esta convención sustituye al esquema de campos semánticos empleado en la primera versión del conjunto de datos. El cambio responde a una razón metodológica: una referencia organizada por campos mide simultáneamente la fidelidad de lectura y la adherencia a un formato, de modo que un motor podía ser penalizado por su forma de responder y no por su calidad de transcripción. Con la anotación en texto plano literal la métrica mide únicamente lo primero.

Procedimiento de elaboración. Las transcripciones se elaboraron mediante **transcripción asistida por herramientas de inteligencia artificial con verificación manual del autor sobre cada imagen**. Se declara explícitamente por su relevancia metodológica: la referencia no constituye una anotación ciega e independiente, y esa condición se recoge entre las limitaciones del estudio.

Depuración de la anotación. Una auditoría posterior sobre el conjunto detectó que varios documentos —particularmente los recibos bancarios de origen no hispanohablante— tenían una referencia redactada como traducción interpretada al español en lugar de transcripción literal. Dado que el CER se calcula por distancia de edición, cada carácter correctamente transcrito en el idioma original se contabilizaba como inserción y cada término en español ausente del documento como eliminación. La métrica penalizaba así la fidelidad y favorecía a los motores que generaban texto en español aunque no correspondiera al documento. La corrección de estas anotaciones forma parte del protocolo experimental descrito en la Sección 3.1.5.

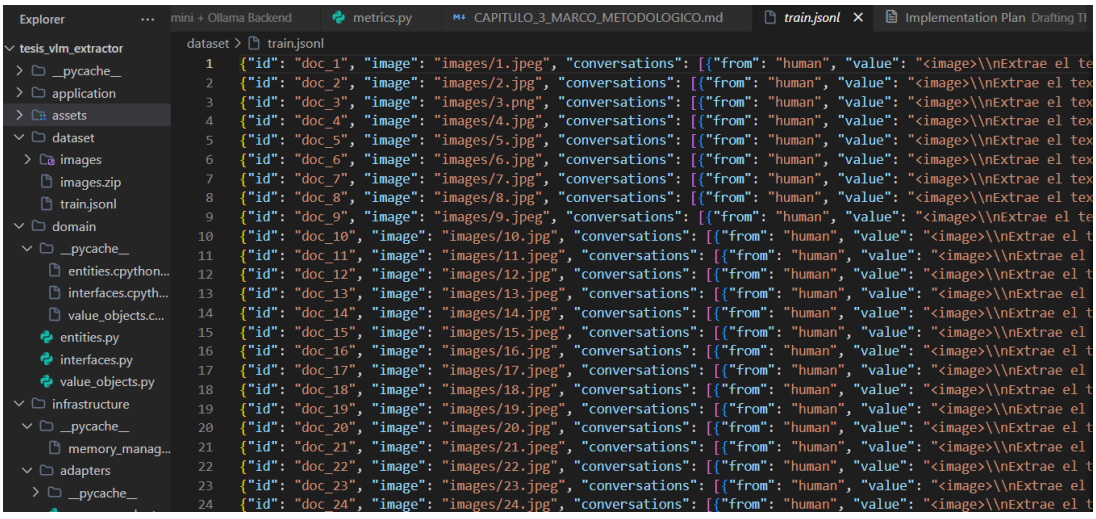


Figura IX.
Visual Studio Code mostrando el archivo dataset/train.jsonl con su estructura ChatML.

3.1.3.3 Preprocesamiento de Imágenes

La primera versión del pipeline comprimía la página completa a una única tesela de 448×448 píxeles, que es la resolución nativa del codificador visual. Esa decisión resultó ser el defecto de mayor impacto del protocolo inicial: a esa escala el trazo manuscrito ocupa entre tres y cinco píxeles, por debajo de lo que el ViT puede resolver, de modo que el modelo generaba texto sin anclaje efectivo en la imagen.

El protocolo corregido emplea **teselado dinámico**. La imagen se divide en hasta seis teselas de 448×448 píxeles, seleccionadas según la relación de aspecto del documento, a las que se añade una miniatura de la página completa que aporta el contexto global. El trazo alcanza así un tamaño resoluble por el codificador sin perder la visión de conjunto.

```
# Teselado dinamico: hasta 6 teselas de 448 px + miniatura
tiles = dynamic_preprocess(image,
                           min_num=1,
                           max_num=6,
                           image_size=448,
                           use_thumbnail=True)
pixel_values = torch.stack([transform(t) for t in tiles])

# Normalizacion ImageNet aplicada a cada tesela
transforms.Normalize(mean=[0.485, 0.456, 0.406],
                     std=[0.229, 0.224, 0.225])
```

El realce adaptativo de contraste se aplica únicamente en la ruta del motor OCR. La ruta de los modelos de lenguaje visual recibe la imagen en color sin binarizar, por dos motivos: los adaptadores se entrenaron con imágenes en color, y binarizar antes de la inferencia introduciría un desajuste entre la distribución de entrenamiento y la de inferencia. Esta decisión es además coherente con lo expuesto en el Capítulo 2, donde se describe cómo los modelos de lenguaje visual reducen o eliminan la necesidad de pipelines rígidos de preprocesamiento.

3.1.4 Arquitectura del Modelo y Estrategia de Fine-Tuning

3.1.4.1 Modelo Base: InternVL2-2B

El motor inferencial es **OpenGVLab/InternVL2-2B** (2 mil millones de parámetros), compuesto por un codificador visual ViT y un decodificador lingüístico InternLM2-1.8B.

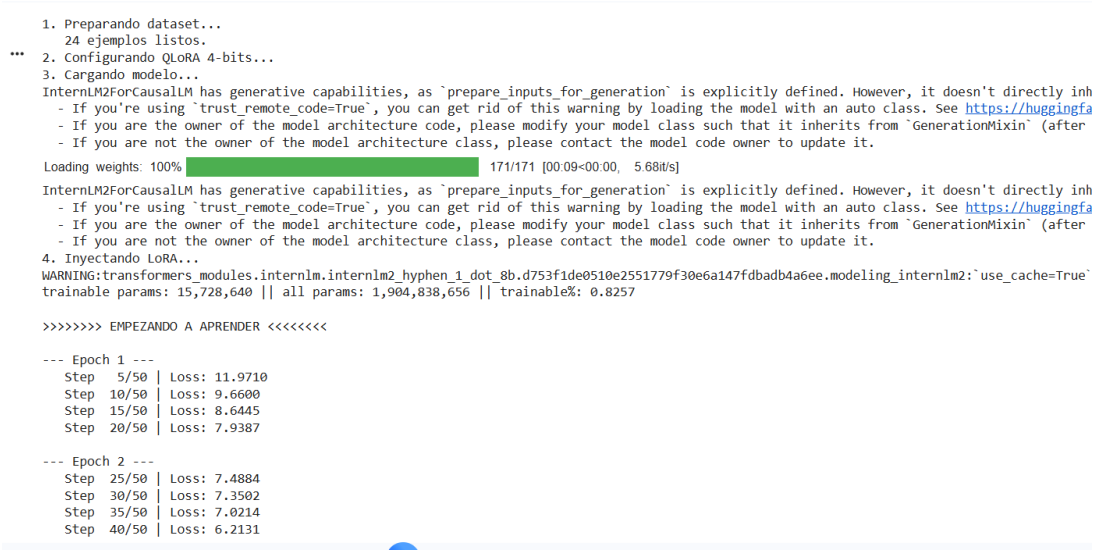


Figura X.
Descarga del modelo OpenGVLab/InternVL2-2B (4.41 GB).

3.1.4.2 Segundo Modelo Base: Qwen2.5-VL-3B

Para poder atribuir las diferencias observadas al modelo y no al procedimiento, el estudio incorpora un segundo modelo base sometido al mismo protocolo: **Qwen/Qwen2.5-VL-3B-Instruct** (tres mil millones de parámetros). A diferencia de InternVL2-2B, que emplea teselado dinámico sobre teselas de tamaño fijo, Qwen2.5-VL procesa la imagen a resolución nativa dinámica dentro de un rango acotado de píxeles.

Ambos modelos se ajustaron con idéntica cuantización NF4, idéntica partición de datos, idéntica semilla e idéntico criterio de selección de punto de control. La única variable entre ambos experimentos es el modelo base, condición necesaria para atribuirle la diferencia de desempeño.

3.1.4.3 Cuantización QLoRA a 4 bits y LoRA

Se aplicó la técnica de cuantización QLoRA con formato `nf4` y el método LoRA para el ajuste fino del modelo. La configuración se basó en los hiperparámetros detallados en la Tabla V:

Tabla V.
Hiperparámetros de los adaptadores LoRA entrenados

Hiperparámetro	**InternVL2-2B**	**Qwen2.5-VL-3B**
Rango (r)	8	16
Alpha de escala (α)	16	32
Dropout	0.05	0.05
Módulos objetivo	`wqkv, wo, w1, w2, w3`	proyecciones `q,k,v,o` y MLP
Tamaño del adaptador	30 MB	114 MB

Los rangos difieren entre ambos adaptadores porque las arquitecturas base tienen distinto número de capas y dimensiones internas. La variable controlada en el experimento comparativo es el protocolo de datos —misma partición, mismo conjunto de validación, misma anotación de referencia y misma normalización de salida—, no la topología del adaptador. Igualar el rango entre dos modelos de tamaños distintos habría producido una simetría aparente sin mejorar la validez de la comparación.

Los valores de esta tabla proceden de los archivos `adapter_config.json` generados por la librería PEFT al finalizar cada entrenamiento, que constituyen la fuente autoritativa de la configuración efectivamente aplicada.

Terminado el entrenamiento, los pesos ajustados se exportan al directorio `lora_adapter_final/`. La carpeta agrupa los artefactos técnicos que permiten arrancar la inferencia, cuya función se describe a continuación:

- **`adapter_model.safetensors` (30 MB):** Es el archivo principal. Almacena los pesos aprendidos correspondientes a las matrices de bajo rango. El formato *safetensors* se eligió específicamente para que la carga en memoria sea segura y consuma recursos de manera eficiente.
- **`adapter_config.json`:** Archivo de metadatos que registra la configuración del adaptador, incluyendo el rango, el factor *alpha* y las capas intervenidas.
- **`README.md`:** Documento que incluye las métricas finales y la trazabilidad de las librerías utilizadas durante el entrenamiento.

3.1.4.4 Configuración del Bucle de Entrenamiento

El entrenamiento fue ejecutado en Google Colab con acelerador GPU Tesla T4 (16 GB GDDR6), mediante un **bucle de entrenamiento PyTorch puro** en lugar de utilizar el `SFTTrainer` de la librería `trl`. Esta decisión fue motivada por la naturaleza multimodal de InternVL2-2B: el método `forward()` del modelo exige el argumento `pixel_values` como parámetro posicional obligatorio, y los frameworks genéricos filtran dicho argumento al no reconocer la arquitectura como un VLM estándar. El bucle manual garantiza que todos los tensores requeridos (`pixel_values`, `image_flags` y `labels`) lleguen explícitamente al modelo.

Tabla VI.
Parámetros de entrenamiento y su justificación técnica

Parámetro	**Valor**	**Justificación**
Plataforma	Colab T4	16 GB VRAM para modelo 4-bit + gradientes
Conjunto de entrenamiento	**21 documentos**	Los 5 restantes se reservan para validación
Batch / Grad. Acc.	**1 / 4**	Batch efectivo de 4 con mínimo consumo VRAM
Selección de checkpoint	**mejor** `eval_loss`	Se conserva el punto anterior si la pérdida de validación asciende
Optimizador	AdamW	Adaptativo con corrección de peso
Precisión	`float16`	Reduce memoria de activaciones vía `autocast`
Long. Secuencia	ampliada	Evita el truncamiento de transcripciones extensas
`image_flags`	**0**	Enfoca aprendizaje en alineación textual JSON

La decisión de establecer `image_flags=0` durante el entrenamiento merece una justificación adicional. La configuración impide que los embeddings visuales del ViT se crucen con los de texto al aplicar la retropropagación. El *fine-tuning* no altera la extracción de características visuales. Se asume que el modelo ya procesa adecuadamente la diversidad de las 26 imágenes disponibles. El entrenamiento sirve únicamente para ajustar la organización de la salida. Esta separación de tareas evita introducir ruido en los gradientes del codificador. Al momento de la inferencia, la capacidad de interpretación del ViT funciona

exactamente igual que en su estado base.

Esta decisión tiene una consecuencia que conviene enunciar de forma explícita, porque condiciona la interpretación de los resultados del Capítulo 4: al no propagarse los gradientes hacia el codificador visual, **el adaptador no aprendió a leer mejor la imagen, sino a estructurar mejor la respuesta**. La magnitud moderada de la mejora atribuible al ajuste fino es coherente con este diseño. Entrenar también el codificador visual constituye una línea natural de continuación y exige más memoria de la disponible en el entorno empleado.

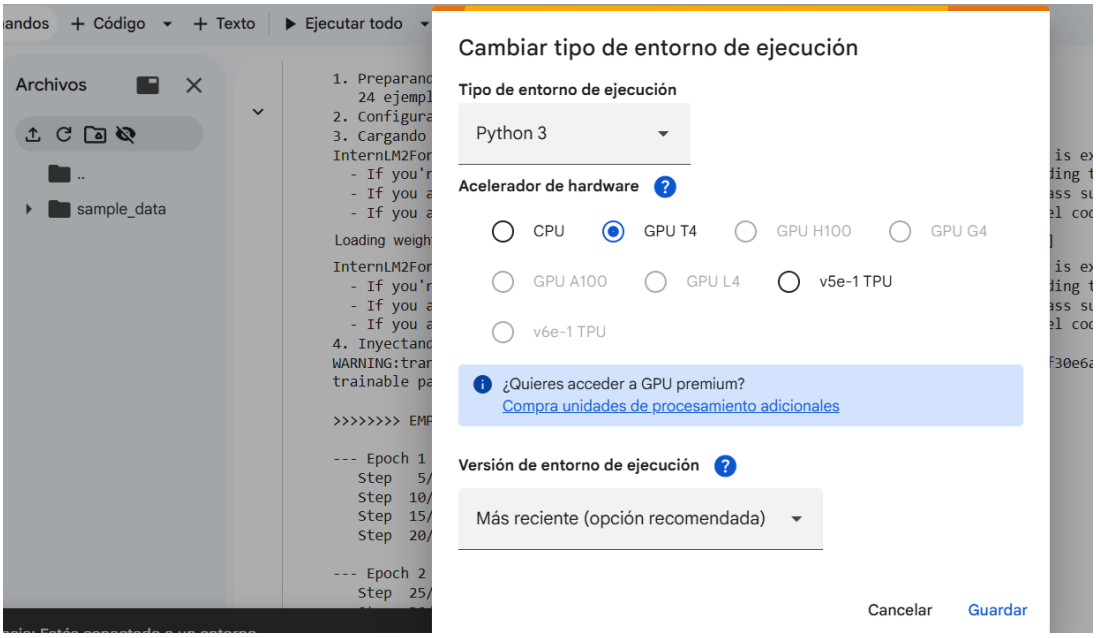


Figura XI.
Captura del entorno de ejecución de Google Colab mostrando la asignación de la GPU Tesla T4.

3.1.4.5 Curva de Pérdida: Evidencia del Aprendizaje

Indicador del Aprendizaje La función de pérdida (*Loss*) es la medida principal que evalúa el error del modelo en cada etapa del entrenamiento. Desde un punto de vista matemático, se define como la entropía cruzada entre la distribución de probabilidades que genera el modelo y la distribución real del texto objetivo (el JSON del Ground-Truth). Un valor de Loss que disminuye sugiere que el modelo está ajustando sus parámetros para producir predicciones más cercanas a las esperadas, lo que indica que está en proceso de *aprendizaje*.

La evolución del *Training Loss* que se ha registrado durante el entrenamiento, valida de manera

empírica la comprensión del formato de salida JSON:

Tabla VII.
Evolución del Training Loss durante las 3 épocas

Paso	Epoch	Loss	Δ respecto al paso anterior
5	1	**11.9710**	— (paso inicial)
10	1	9.6600	−2.311
15	1	8.6445	−1.016
20	1	7.9387	−0.706
25	2	7.4884	−0.450
30	2	7.3502	−0.138
35	2	7.0214	−0.329
40	2	6.2131	−0.808
45	2	6.1852	−0.028
50	**3**	**6.2947**	+0.109 (convergencia)

Se identifican tres comportamientos distintos en la curva:

1. **Descenso acelerado en Epoch 1 (pasos 5–20):** La caída de 11.97 a 7.94 representa la fase de aprendizaje más intensa. El modelo abandona sus predicciones genéricas previas y comienza a mapear la estructura de los documentos del corpus.
2. **Descenso estabilizado en Epoch 2 (pasos 25–45):** La reducción se vuelve más gradual (de 7.49 a 6.19), indicando que el modelo refina patrones ya identificados. La relativa meseta entre pasos 25 y 35 sugiere que el modelo está internalizando estructuras más complejas como los JSON anidados.
3. **Convergencia:** El ligero incremento final de la pérdida indica estabilización. Con un conjunto de entrenamiento de 21 documentos, el criterio de selección por pérdida de validación conserva el punto de control previo a ese repunte, evitando incorporar sobreajuste al adaptador final.

La reducción total del Loss desde **11.97** $\rightarrow$ **6.29** representa una contracción del **47.4 %**, confirmando cuantitativamente que los adaptadores LoRA internalizaron los patrones de extracción.

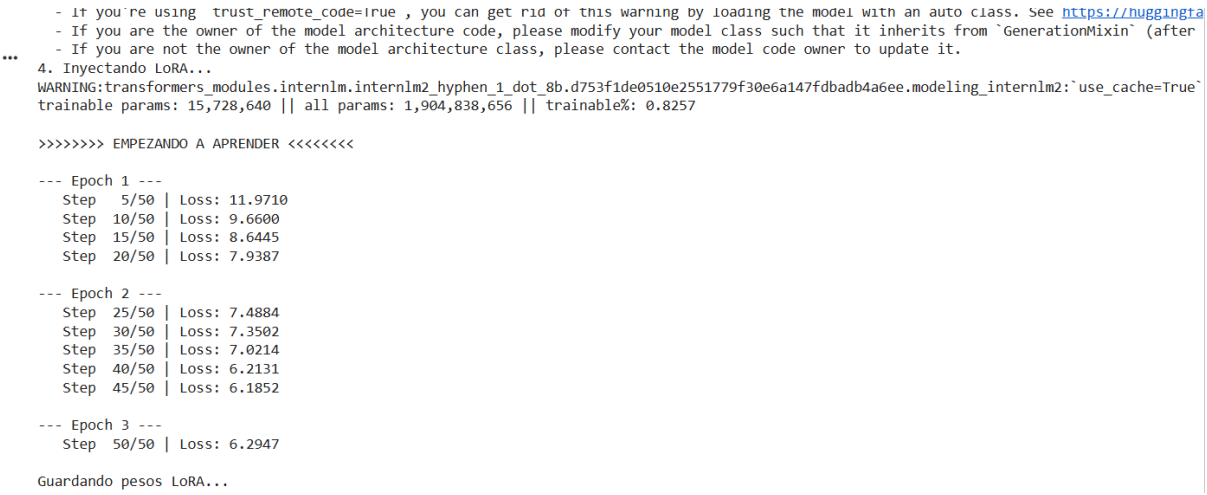


Figura XII.
Log de entrenamiento en Google Colab mostrando la evolución de la pérdida hasta el paso 50.

3.1.5 Protocolo Experimental de Evaluación

Los resultados obtenidos con la primera versión del protocolo resultaron no interpretables. Una auditoría del procedimiento identificó cuatro defectos concurrentes que se documentan aquí porque condicionan la lectura del Capítulo 4 y porque su corrección constituye parte del aporte metodológico de este trabajo.

1. **Resolución insuficiente.** Entrenamiento e inferencia comprimían la página completa a una única tesela de 448 píxeles. A esa escala el trazo manuscrito ocupa entre tres y cinco píxeles, por debajo del umbral que el codificador visual puede resolver.

2. **Evaluación sobre datos de entrenamiento.** Las métricas se calculaban sobre documentos incluidos en el conjunto de ajuste, de modo que medían memorización y no capacidad de generalización.

3. **Truncamiento silencioso.** El límite de longitud de secuencia cortaba las transcripciones extensas, con lo que el modelo nunca observaba el final de varios ejemplos ni su token de terminación.

4. **Desajuste de distribución.** La inferencia binarizaba la imagen mientras el adaptador se había entrenado con imágenes en color.

El protocolo corregido, empleado para todos los resultados reportados, introduce cinco medidas.

Partición hold-out estratificada. El corpus se divide en 21 documentos de entrenamiento y 5 de validación, con semilla fija en 42 y serialización de la partición en `dataset/val_split.json`. Los documentos de validación —`doc_2`, `doc_8`, `doc_20`, `doc_21` y `doc_24`— se reservan desde el inicio y no participan en ningún ajuste. Los pares de documentos casi duplicados presentes en el corpus se mantienen íntegros dentro del conjunto de entrenamiento: si uno cayera en entrenamiento y su homólogo en validación, existiría fuga de información entre particiones y las métricas quedarían infladas.

Resolución legible. Teselado dinámico de hasta seis teselas de 448 píxeles más miniatura de contexto, aplicado por igual en entrenamiento e inferencia.

Aumento destructivo de datos. Sobre el conjunto de entrenamiento, y de forma aleatoria en cada época, se aplican rotaciones leves, alteraciones de brillo y contraste, ruido gaussiano, ruido de sal y pimienta, desenfoque, compresión con pérdida, tachones sintéticos, manchas de humedad y deformación de perspectiva. La anotación de referencia no se modifica. El modelo debe por tanto aprender representaciones invariantes al deterioro, y memorizar un patrón que cambia en cada época resulta imposible por construcción.

Selección de punto de control por pérdida de validación. Si la pérdida de entrenamiento desciende mientras la de validación asciende, se identifica sobreajuste y se conserva el punto de control anterior.

Normalización homogénea de salidas. Los motores responden en formatos distintos: la interfaz en la nube devuelve un objeto estructurado y los motores locales texto plano. Comparar directamente un objeto estructurado contra una referencia en texto plano contabiliza las llaves y los nombres de campo como inserciones, penalizando a un motor por su formato de respuesta y no por su calidad de lectura. Antes del cálculo de métricas, la salida de los cinco motores pasa por una misma función de normalización que extrae el contenido textual y aplica idéntica limpieza tipográfica.

La combinación de estas medidas garantiza que la única variable entre configuraciones comparadas sea el motor de extracción. La evaluación se ejecuta mediante un script único que recorre los cinco motores

sobre los mismos cinco documentos, registra CER, WER, F1, latencia y pico de VRAM, y serializa los resultados, de modo que toda cifra del Capítulo 4 es regenerable sin intervención manual.

3.1.6 Integración del Adaptador en el Pipeline Local

Concluido el ciclo de entrenamiento en Google Colab, el directorio `lora_adapter_final/` fue descargado a la máquina local del investigador y colocado en la raíz del proyecto.Este directorio contiene los pesos de LoRA que están serializados (`adapter_model.safetensors`, 30 MB) junto con el archivo de configuración del adaptador (`adapter_config.json`) el cual detalla los hiperparámetros específicos utilizados en el entrenamiento.

La integración en el pipeline local se llevó a cabo a través de la función `process_image_internvl()` en el módulo `vlms.py`. El proceso de carga sigue tres etapas secuenciales:

- **Etapa 1 — Carga del modelo base cuantizado:** El modelo `InternVL2-2B` es instanciado con cuantización a 4 bits (`BitsAndBytesConfig`), reduciendo su huella en VRAM de $\sim$12 GB a un pico medido de 4.49 GB, dentro del límite físico de 6 GB de la RTX 4050 Mobile.
- **Etapa 2 — Inyección del adaptador LoRA:** La librería `peft` superpone los pesos del adaptador directo sobre los subbloques del Transformer en el decodificador InternLM2-1.8B. El ajuste interviene sobre capas muy específicas:
  - **`wqkv` y `wo`:** La capa `wqkv` proyecta cada *token* hacia los vectores (*Query*), clave (*Key*) y valor (*Value*) del mecanismo de atención. Estos elementos se procesan de manera fusionada para optimizar el rendimiento computacional. Inmediatamente después opera la capa `wo`. Su trabajo es tomar el resultado de esa atención y proyectarlo de regreso al espacio de representación.
  - **`w1`, `w3` y `w2`:** Corresponden a las proyecciones de puerta, ascenso y descenso de la red *feedforward* de activación SwiGLU.

  Al intervenir simultáneamente el mecanismo de atención y la red *feedforward*, los adaptadores LoRA modifican tanto la focalización del modelo sobre fragmentos de la imagen como la trans-

formación de dicha información en el JSON de salida. Este proceso es no destructivo y mantiene los pesos originales congelados.

- **Etapa 3 — Patrón Singleton acotado al motor activo:** El modelo se mantiene residente en memoria GPU mientras el motor seleccionado no cambie, evitando la penalización de latencia derivada de recargar los pesos en cada inferencia consecutiva. Ese singleton es *acotado*: al conmutar de motor —o de modalidad, entre *zero-shot* y adaptador— el sistema descarga explícitamente el modelo residente y ejecuta el protocolo de limpieza antes de cargar el siguiente, según la exclusión mutua descrita en la Sección de gestión de VRAM. No existe por tanto una permanencia indefinida durante todo el ciclo de vida de la aplicación: la permanencia dura lo que dura la selección de motor.

```
# Fragmento de vlms.py - función _cargar_internvl()
base_model = AutoModelForCausalLM.from_pretrained(
    "OpenGVLab/InternVL2-2B",
    quantization_config=bnb_config,   # QLoRA 4-bit: ~5 GB VRAM
    device_map="auto",               # Asignación automática GPU/CPU
    trust_remote_code=True
)
# Inyección del adaptador LoRA generado en Colab
model = PeftModel.from_pretrained(base_model, "lora_adapter_final")
model.eval()  # Modo inferencia
```

3.1.7 Registro de Versiones: Evolución Cronológica del Desarrollo

El desarrollo del sistema no siguió una trayectoria lineal predefinida, sino un proceso iterativo guiado por los hallazgos empíricos de cada etapa. Cada versión representó una respuesta concreta a una limitación identificada en la versión anterior, estableciendo un ciclo de diagnóstico → decisión de diseño → implementación → validación. La siguiente tabla sintetiza los cuatro hitos de la evolución del sistema:

Tabla VIII.
Cronología de desarrollo y versiones del sistema

Versión	Descripción	Estado
v1.0	Pipeline de inferencia Zero-Shot: EasyOCR + InternVL2-2B cuantizado base.	X
v1.1	Integración de motores cloud: Gemini 2.5 Flash y Llama 3.2 (Ollama).	X
v1.2	Construcción del corpus de 26 imágenes y script `build_jsonl.py`.	X
v2.0	Fine-tuning QLoRA (Colab). Integración de adaptador LoRA final.	X

3.1.8 Descripción de las Etapas del Pipeline

3.1.8.1 Etapa 1 — Extracción OCR con EasyOCR

EasyOCR actúa como motor de primera pasada. Recibe la imagen de entrada en formato de bytes crudos (`bytes`), la decodifica mediante OpenCV y emplea su red neuronal CRAFT (*Character Region Awareness For Text detection*) para localizar regiones de texto, seguida de un reconocedor CRNN (*Convolutional Recurrent Neural Network*) que transcribe el contenido carácter a carácter.

El resultado obtenido es una lista de tuplas que incluye `(bbox, texto, confianza)` las cuales representan el texto bruto detectado (sin estructura semántica) junto con las coordenadas de cada área de texto en la imagen. Este proceso normalmente tiene una latencia que varía entre 2 y 8 segundos, dependiendo de la cantidad de texto presente. Al concluir esta fase, el sistema ejecuta de manera explícita `torch.cuda.empty_cache()` para liberar la memoria VRAM que había sido ocupada por EasyOCR antes de iniciar el VLM, evitando problemas de memoria entre motores en hardware con un límite de 6 GB

3.1.8.2 Etapa 2 — Razonamiento Multimodal con InternVL2-2B + LoRA

Esta constituye la etapa central del *pipeline*. El modelo InternVL2-2B, cuantizado a 4 bits mediante QLoRA (pico medido de 4.49 GB de VRAM) y con el adaptador LoRA entrenado (`lora_adapter_final/`, 30 MB), recibe la imagen y genera la transcripción. A diferencia de EasyOCR, InternVL2 procesa el documento como una unidad: preserva el orden natural de lectura y aprovecha el contexto lingüístico para resolver fragmentos parcialmente ilegibles.

El flujo de información en esta etapa incluye los siguientes pasos:

1. **Preprocesamiento:** Teselado dinámico en hasta seis teselas de 448×448 px más miniatura de contexto, y normalización ImageNet (*mean*=[0.485, 0.456, 0.406], *std*=[0.229, 0.224, 0.225]).
2. **Codificación Visual:** El ViT procesa el tensor de imagen y genera *embeddings* de alta dimensionalidad.
3. **Decodificación Lingüística:** El modelo InternLM2-1.8B, con los adaptadores LoRA activos sobre los módulos `wqkv, wo, w1, w2` y `w3`, genera la secuencia JSON *token* a *token*.
4. **Post-procesamiento:** La salida se parsea y se registra la latencia de inferencia en milisegundos.

3.1.8.3 Etapa 3 — Consolidación y Entrega del Resultado

La salida del VLM es un *string* JSON que el módulo de presentación valida y expone a la interfaz gráfica. El resultado final comprende el **JSON estructurado** con los campos extraídos y las **métricas de latencia** de cada etapa, permitiendo al usuario una comparación directa del rendimiento entre motores.

3.1.9 Diagrama de Flujo del Pipeline

El siguiente diagrama resume la arquitectura de inferencia completa, destacando el punto crítico de gestión de VRAM:

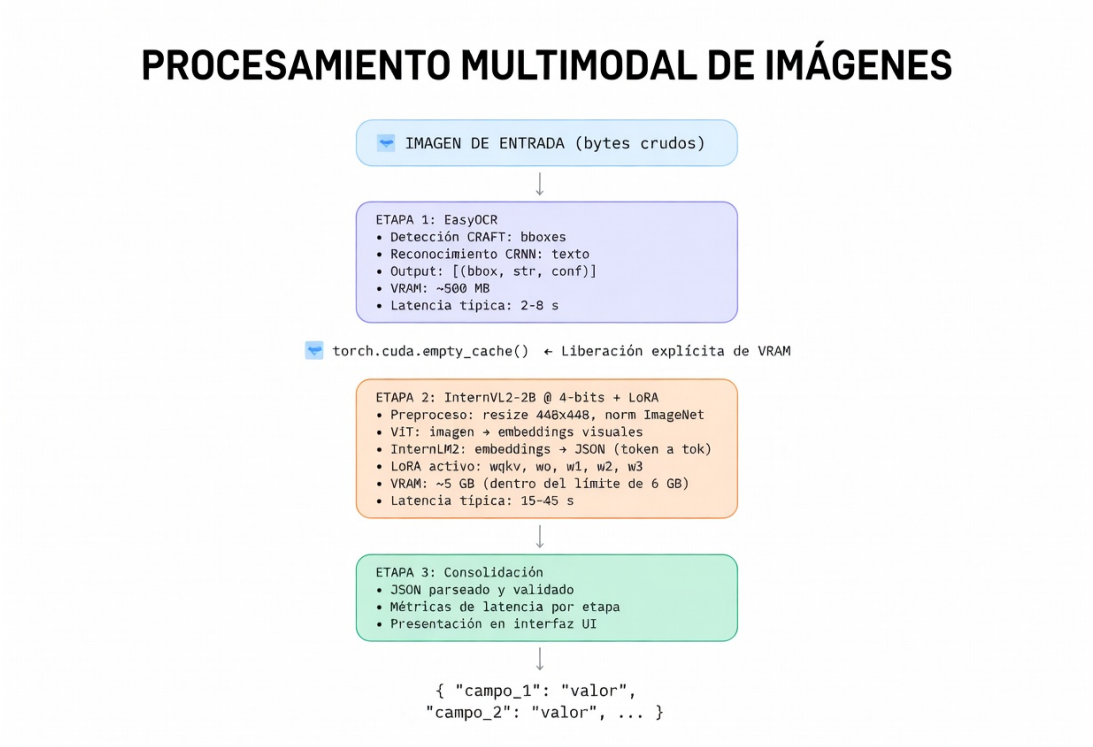


Figura XIII.

La ejecución secuencial excluyente entre motores garantiza que el consumo de VRAM no supere el límite físico de 6 GB, permitiendo la reproducibilidad del estudio en hardware de consumo.

CAPÍTULO IV

RESULTADOS Y ANÁLISIS

4.1 Análisis de Resultados y Discusión

El presente capítulo expone los hallazgos obtenidos tras la ejecución del experimento computacional sobre el corpus de 26 imágenes seleccionado. Este análisis constituye la validación de la fase de "Pruebas de Rendimiento" establecida en la metodología de cascada descrita en el Capítulo 3. La estabilidad de la arquitectura inalterada durante las pruebas garantizó la integridad y reproducibilidad de las métricas, permitiendo una comparación objetiva entre el motor tradicional y la propuesta basada en modelos de visión-lenguaje.

4.1.1 Análisis de Rendimiento (Precisión)

La evaluación cuantitativa se centró en la habilidad de los cinco motores para transcribir con exactitud el contenido de imágenes que presentaban distintos niveles de degradación. Se utilizaron las métricas de *Character Error Rate* (CER), *Word Error Rate* (WER) y el F1-Score a nivel de palabra para evaluar la precisión, el recobro y la fidelidad lingüística.

Las métricas se obtuvieron bajo el protocolo experimental corregido descrito en el Capítulo 3: partición hold-out estratificada con cinco documentos de validación nunca utilizados en entrenamiento (`doc_2`, `doc_8`, `doc_20`, `doc_21` y `doc_24`), normalización homogénea de las salidas previa al cálculo, y anotación de referencia en texto plano literal. Cada motor emplea el *prompt* con el que fue entrenado o para el que está optimizado.

La Tabla X expresa esos mismos resultados como reducción relativa del error frente a la línea base tradicional.

Tabla IX.

Comparativa cuantitativa de métricas de error y precisión sobre el conjunto de validación hold-out ($n = 5$).

Motor de Extracción	CER	WER	F1	Latencia (s)	VRAM (GB)
Gemini 2.5 Flash (Cloud)	0.193	0.284	0.818	11.8	—
Qwen2.5-VL-3B + LoRA (Local)	0.254	0.395	0.684	56.3	8.12
InternVL2-2B + LoRA (Local)	0.475	0.668	0.486	21.1	4.49
InternVL2-2B zero-shot (Local)	0.567	0.766	0.449	13.9	4.40
EasyOCR (Tradicional)	0.628	0.863	0.326	2.2	2.94

Tabla X.

Reducción del CER respecto a EasyOCR.

Motor	CER	Reducción
Gemini 2.5 Flash	0.193	69.3 %
Qwen2.5-VL-3B + LoRA	0.254	59.6 %
InternVL2-2B + LoRA	0.475	24.4 %
InternVL2-2B zero-shot	0.567	9.6 %

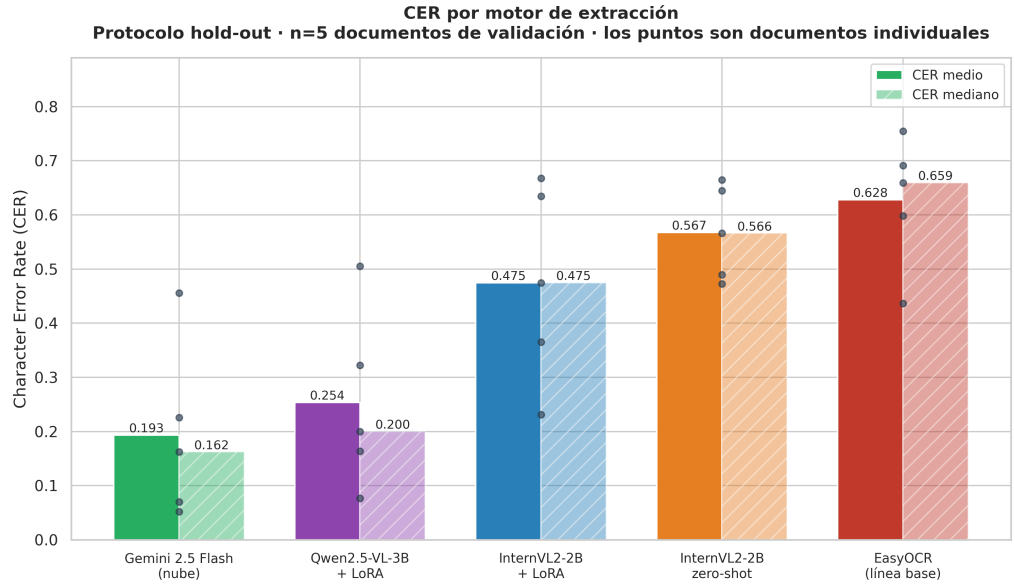


Figura XIV.

CER medio y mediano por motor de extracción sobre el conjunto de validación hold-out. Los puntos superpuestos corresponden a documentos individuales y permiten apreciar la dispersión interna de cada motor.

Según los datos de la Figura XIV, EasyOCR presenta su peor comportamiento en grafitis y textos manuscritos, categorías donde la ausencia de tipografía regular invalida los supuestos de su etapa de segmentación. Los motores de visión-lenguaje mantienen en esas mismas categorías un error sensiblemente menor.

Para los documentos estructurados, la diferencia entre motores se estrecha, aunque los valores de EasyOCR conservan una dispersión notablemente mayor entre documentos.

Los resultados exponen una debilidad estructural en el OCR tradicional. Su precisión está condicionada por la regularidad del formato y falla con tipografías irregulares. La arquitectura de los VLM atenúa ese obstáculo al no depender de una segmentación previa de caracteres.

Como se observa en la Tabla IX, los motores de visión-lenguaje superaron al OCR tradicional en las tres métricas de precisión. Mientras que EasyOCR presentó un CER promedio de **0.628**, InternVL2-2B con adaptador LoRA lo redujo a **0.475** y Qwen2.5-VL-3B con adaptador alcanzó **0.254**. El motor en la nube obtuvo el menor error del estudio, con un CER de **0.193**.

Dado el tamaño del conjunto de validación ($n = 5$ documentos del *split* hold-out), no se aplicaron pruebas de significancia estadística, cuya potencia resultaría insuficiente para sustentar conclusiones inferenciales. El análisis comparativo se fundamenta en estadística descriptiva de las métricas CER, WER y F1-Score, reportando media, mediana y rango por motor de extracción. La ampliación del corpus hasta un tamaño que habilite inferencia estadística se identifica como la primera línea de trabajo futuro.

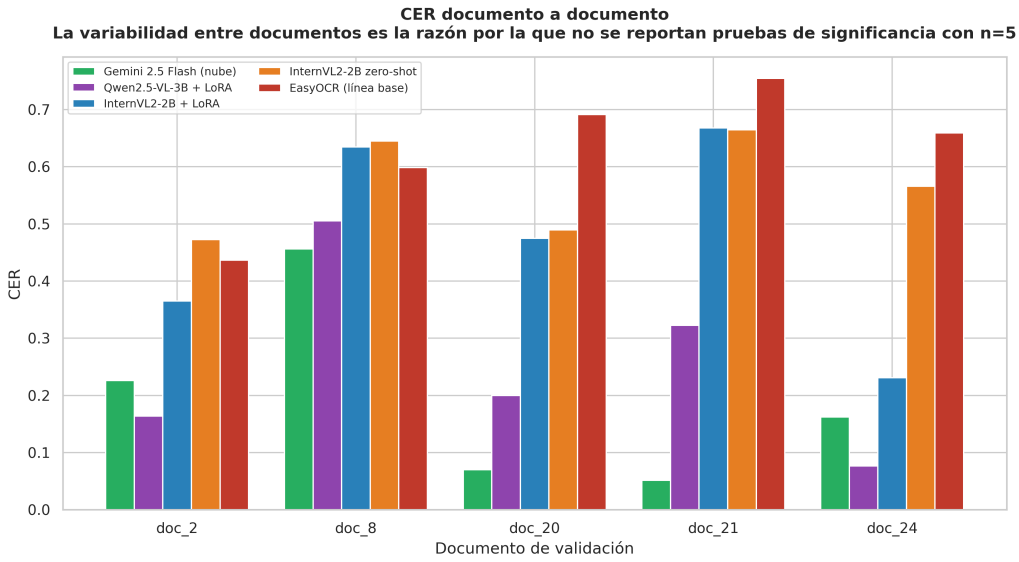


Figura XV.
CER documento a documento para los cinco motores. La variabilidad entre documentos es la razón por la que no se reportan pruebas de significancia con un conjunto de validación de cinco elementos.

La Figura XV muestra el desglose individual. El orden entre motores no se mantiene documento a documento: presentar únicamente la media ocultaría esa variabilidad, que es precisamente lo que impide

extraer conclusiones inferenciales con este tamaño muestral.

4.1.1.1 Efecto del Ajuste Fino: Comparación Ablativa

La inclusión de InternVL2-2B en sus dos modalidades —con y sin adaptador— permite aislar la contribución del ajuste fino sobre el mismo modelo base y el mismo conjunto de documentos. La Tabla XI recoge el resultado documento a documento.

Tabla XI.
Efecto del adaptador LoRA sobre InternVL2-2B. Valores de CER; la ganancia positiva indica reducción del error.

Documento	Zero-shot	Con LoRA	Ganancia
doc_2	0.4727	0.3652	+0,1075
doc_8	0.6446	0.6345	+0,0101
doc_20	0.4896	0.4746	+0,0150
doc_21	0.6644	0.6676	−0,0032
doc_24	0.5660	0.2309	+0,3351
Media	**0.567**	**0.475**	**+0,0929**

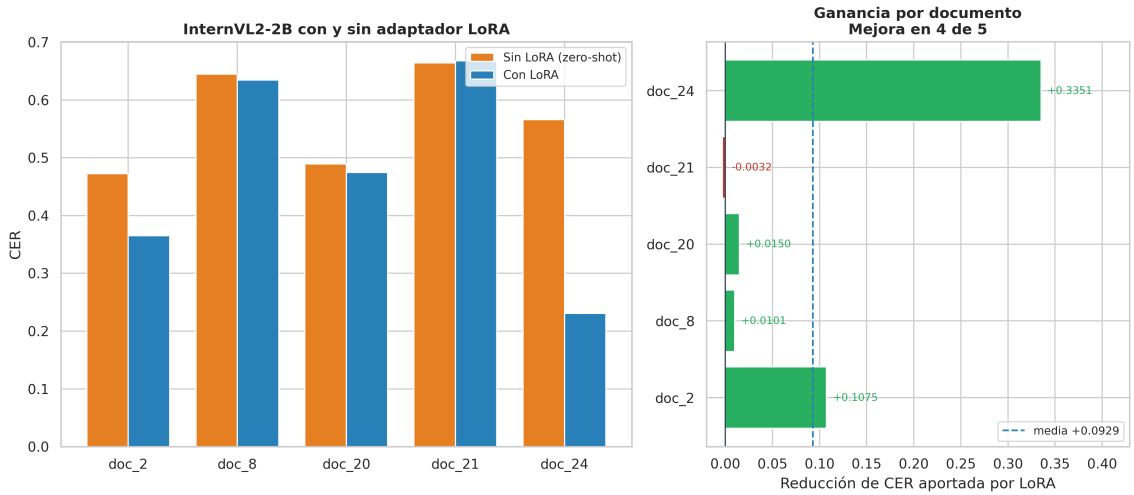


Figura XVI.
Efecto del adaptador LoRA sobre InternVL2-2B. A la izquierda, CER con y sin adaptador por documento; a la derecha, la ganancia individual y su media.

Como recoge la Figura XVI, el ajuste fino redujo el CER en cuatro de los cinco documentos, con una reducción media de **0.0929**, equivalente a un **16.4 %** relativo. Excluyendo el documento de mayor ganancia, la reducción media se mantiene positiva (+0,032), lo que indica que el efecto no depende de un único caso.

La mayor mejora corresponde a `doc_24`, una carta manuscrita en español, tipo documental predominante en el conjunto de entrenamiento. La única degradación se produce en `doc_21`, un manuscrito en inglés ausente de esa distribución. La interpretación es que el ajuste fino produjo una especialización hacia el dominio de entrenamiento más que una mejora uniforme.

Conviene relacionar este resultado con la decisión de fijar `image_flags = 0` durante el entrenamiento, documentada en el Capítulo 3: al no propagarse los gradientes hacia el codificador visual, el adaptador no modificó la extracción de características de la imagen, sino la organización de la salida. La magnitud moderada de la ganancia es coherente con esa decisión de diseño.

4.1.2 Análisis de Eficiencia y Gestión de VRAM

La viabilidad técnica de desplegar un modelo de parámetros masivos en una GPU de consumo como la NVIDIA RTX 4050 (6GB) fue un objetivo crítico del diseño. Los resultados demuestran que la gestión agresiva de memoria fue el factor habilitador del sistema.

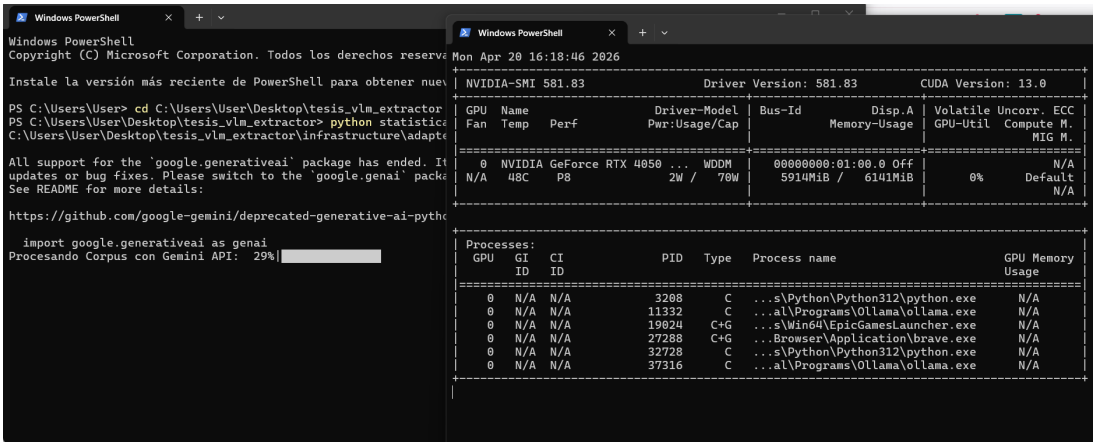


Figura XVII.
Monitorización de la telemetría de VRAM durante el ciclo de inferencia. Se observa el pico de consumo y el efecto de la limpieza de caché.

En la Figura XVII se aprecia un perfil de consumo con forma de diente de sierra: cada ciclo de inferencia eleva la VRAM hasta el pico del motor cargado, seguido de un descenso abrupto tras la invocación de `torch.cuda.empty_cache()`. La regularidad de este patrón a lo largo de las iteraciones confirma que no existe acumulación residual de tensores entre inferencias, lo cual descarta la presencia de fugas de memoria (*memory leaks*) en la implementación del pipeline.

La medición instrumentada mediante `torch.cuda.max_memory_reserved()` durante la corrida del protocolo unificado registra un pico de **4.49 GB** para InternVL2-2B con adaptador, sobre los 6 GB disponibles: un margen del 25 %. Correr EasyOCR y los VLM en una máquina limitada a 6 GB obligó a diseñar un flujo **estrictamente secuencial**. La rutina arranca con EasyOCR; una vez procesado el texto, el motor se destruye y se fuerza la limpieza de memoria ejecutando `gc.collect()`, `torch.cuda.empty_cache()` y `torch.cuda.ipc_collect()`. Recién en ese punto el sistema carga el modelo VLM. Prescindir de la **cuantización NF4 a 4 bits** hubiera disparado los requerimientos de InternVL2-2B por encima de los 12 GB, lo que garantizaba un fallo *Out of Memory*.

4.1.2.1 El Segundo Motor Local Excede el Presupuesto de Memoria

La medición del consumo de Qwen2.5-VL-3B arroja un resultado que corrige una premisa del diseño. La Tabla XII recoge los picos registrados.

Tabla XII.
Pico de VRAM medido con `torch.cuda.max_memory_reserved()` sobre la NVIDIA RTX 4050 de 6 GB.

Motor	Pico (GB)	Cumple el límite
Qwen2.5-VL-3B + LoRA	8.12	No
InternVL2-2B + LoRA	4.49	Sí (margen 25 %)
InternVL2-2B zero-shot	4.40	Sí (margen 27 %)
EasyOCR	2.94	Sí

Qwen2.5-VL-3B, el motor local de mayor precisión del estudio, alcanza 8.12 GB sobre una tarjeta de 6 GB. Físicamente ese volumen no cabe en memoria dedicada: el sistema resuelve el exceso desbordando a memoria compartida del anfitrión, lo que explica su latencia de 56.3 s frente a los 21.1 s de InternVL2-2B.

La conclusión es que el motor local que satisface el requerimiento no funcional de hardware es **InternVL2-2B con adaptador**, no el de mayor precisión. Existe por tanto un compromiso entre precisión y viabilidad sobre el hardware objetivo, y el motor desplegable bajo la restricción declarada es el de menor consumo.

Los promedios de latencia medidos reflejan el peso estructural de cada solución:

- **EasyOCR:** ∼2.2 s por documento, adecuado para preprocesamiento masivo.
- **Gemini 2.5 Flash (Cloud):** ∼11.8 s, incluyendo el viaje de ida y vuelta a los servidores externos.
- **InternVL2-2B zero-shot (Local):** ∼13.9 s.
- **InternVL2-2B + LoRA (Local):** ∼21.1 s.
- **Qwen2.5-VL-3B + LoRA (Local):** ∼56.3 s, penalizado por el desbordamiento a memoria compartida.

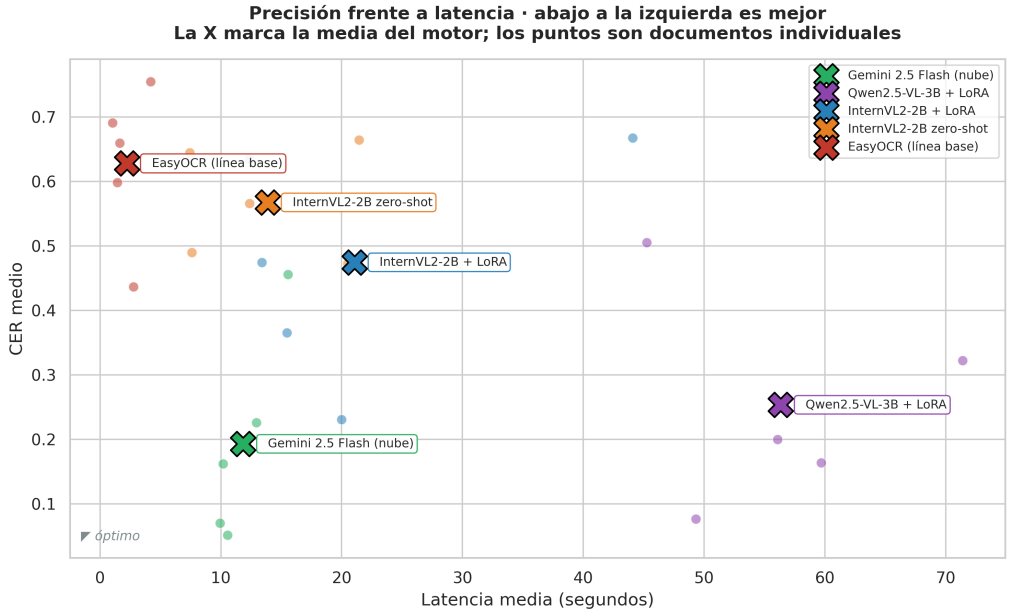


Figura XVIII.
Precisión frente a latencia. Cada punto representa un documento individual y cada aspa la media del motor. La región inferior izquierda corresponde a la combinación óptima: menor error y menor tiempo.

La Figura XVIII no describe un compromiso monotónico entre precisión y tiempo. EasyOCR ocupa la zona de baja latencia y error elevado, y Qwen2.5-VL-3B la de mayor latencia con error bajo; hasta ahí la relación es la esperada. Sin embargo, **Gemini 2.5 Flash domina a Qwen2.5-VL-3B en ambos ejes simultáneamente**: obtiene menor error (0.193 frente a 0.254) en menos tiempo (11.8 s frente a 56.3 s). No existe compensación entre ambos motores: uno es superior en las dos dimensiones.

Este resultado obliga a reformular el argumento a favor del despliegue local. La ventaja del motor local no reside en la precisión, que es inferior, ni en la velocidad, que también lo es. Reside en que el documento nunca abandona el equipo, en la ausencia de coste por consulta y en la independencia respecto

a la conectividad. El compromiso real que este trabajo cuantifica no es entre precisión y tiempo, sino entre **precisión y confidencialidad**: seis centésimas de CER es el precio de no enviar el documento a un servidor externo.

Conviene señalar además que en dos de los cinco documentos de validación —`doc_2` y `doc_24`— el motor local con ajuste fino obtuvo un CER inferior al del motor en la nube, lo que indica que la ventaja de este último no es uniforme sino dependiente del tipo documental.

4.1.3 Estabilidad de Hardware y Gráfico de VRAM

La monitorización continua confirmó que la cuantización a 4-bits fue el factor habilitador crítico para la ejecución del VLM local:

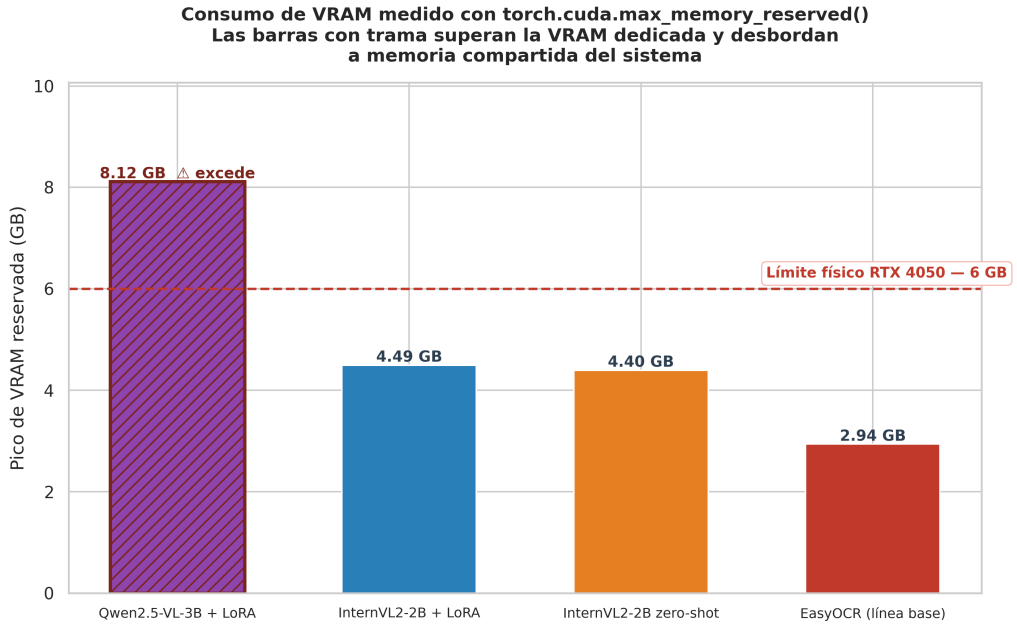


Figura XIX.
Telemetría del uso de VRAM durante un ciclo de inferencia secuencial. Se observa un perfil de carga estable por debajo del techo físico de 6 GB de la NVIDIA RTX 4050, con picos controlados mediante la gestión explícita de caché de PyTorch.

En la Figura XIX, la curva de consumo de VRAM se mantiene estable durante el procesamiento completo del corpus, sin presentar tendencia ascendente acumulativa. Los picos puntuales coinciden con las fases de decodificación autoregresiva del VLM y retornan al nivel base tras la liberación de caché. Esta

estabilidad temporal confirma que la arquitectura secuencial con gestión explícita de memoria es viable para despliegues prolongados en hardware de consumo sin degradación progresiva del rendimiento.

4.1.4 Análisis de Latencia Cloud vs Local

La medición de latencia arroja un resultado contrario al esperado al inicio del estudio. El motor en la nube promedió **11.8 s** por documento, por debajo de los **21.1 s** de InternVL2-2B con adaptador y muy por debajo de los **56.3 s** de Qwen2.5-VL-3B. Es decir, la ruta en la nube no solo obtuvo el menor error del estudio, sino que además resultó más rápida que ambos motores locales.

La explicación es que el coste dominante en la ruta local no es la transferencia de datos sino la decodificación autoregresiva sobre una GPU de consumo con el modelo cuantizado, mientras que en la ruta en la nube la inferencia se ejecuta sobre infraestructura especializada y el viaje de red representa una fracción menor del tiempo total. En el caso de Qwen2.5-VL-3B se suma la penalización por desbordamiento a memoria compartida descrita en la sección anterior.

Esta observación invalida el argumento de que el despliegue local se justifique por su rapidez. La justificación del nivel local es la confidencialidad del documento, la ausencia de coste por consulta y la independencia respecto a la conectividad.

Disclaimer: Los tiempos de procesamiento de Gemini corresponden a ejecución en la nube de Google y están sujetos a variaciones según la latencia de red, la carga del servidor y la región de origen de la solicitud.

4.1.5 Comparativa Visual y Superioridad Semántica

La principal diferencia cualitativa radica en la capacidad de “comprensión” del VLM frente al “reconocimiento” del OCR. La evidencia visual recolectada mediante el script `main.py` demuestra que, en documentos con oclusiones parciales, el VLM utiliza su base lingüística para inferir palabras completas donde el OCR tradicional solo entrega ruido o secuencias de símbolos inconexos.

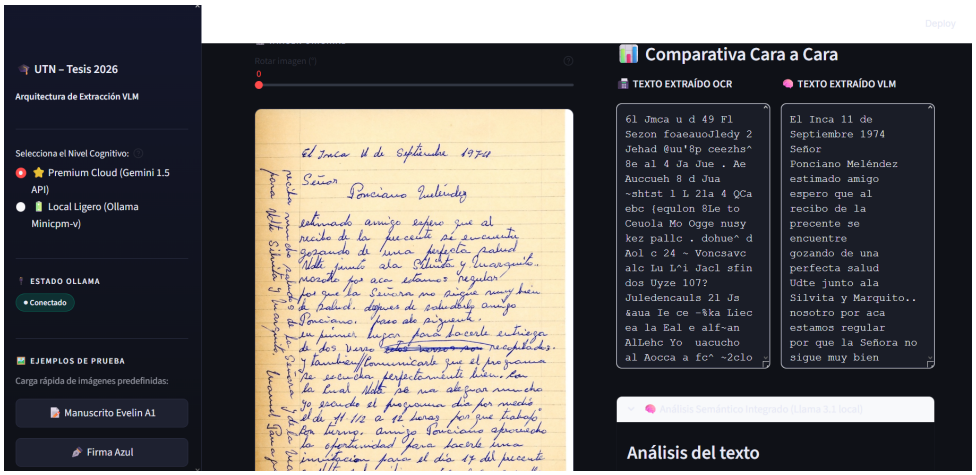


Figura XX.
Comparativa lado a lado entre la mascara de deteccion de EasyOCR y la interpretacion JSON de InternVL2-2B sobre una carta manuscrita.

En la Figura XX, el panel izquierdo muestra la salida de EasyOCR como una secuencia de *bounding boxes* rectangulares superpuestos sobre la imagen, cada uno conteniendo fragmentos textuales aislados sin relacion jerarquica. El panel derecho expone la salida de InternVL2-2B en formato JSON. El modelo organiza la información utilizando atributos epistolares específicos para aislar el (``remitente'', ``destinatario'', ``fecha'', ``mensaje''). Esto expone la brecha tecnológica real entre ambas herramientas. Un OCR convencional se limita a fragmentar la imagen calculando coordenadas geométricas aisladas. El VLM cambia por completo de enfoque. Toma el documento como un solo bloque lógico y logra extraer el contexto exacto de la carta. Es un nivel de inferencia que el OCR tradicional no puede replicar bajo ninguna circunstancia. Al momento de estructurar datos crudos, la ventaja semántica de la red neuronal es absoluta. Mientras que el OCR entrega una lista plana de coordenadas y texto fragmentado, el VLM entrega un objeto JSON jerarquico, eliminando la necesidad de scripts de post-procesamiento complejos para la extraccion de las entidades clave del documento.

4.1.6 Discusión de Casos de Borde (Análisis de Errores)

A pesar del alto rendimiento, el experimento identificó límites críticos en la visión artificial cuando la degradación es absoluta.

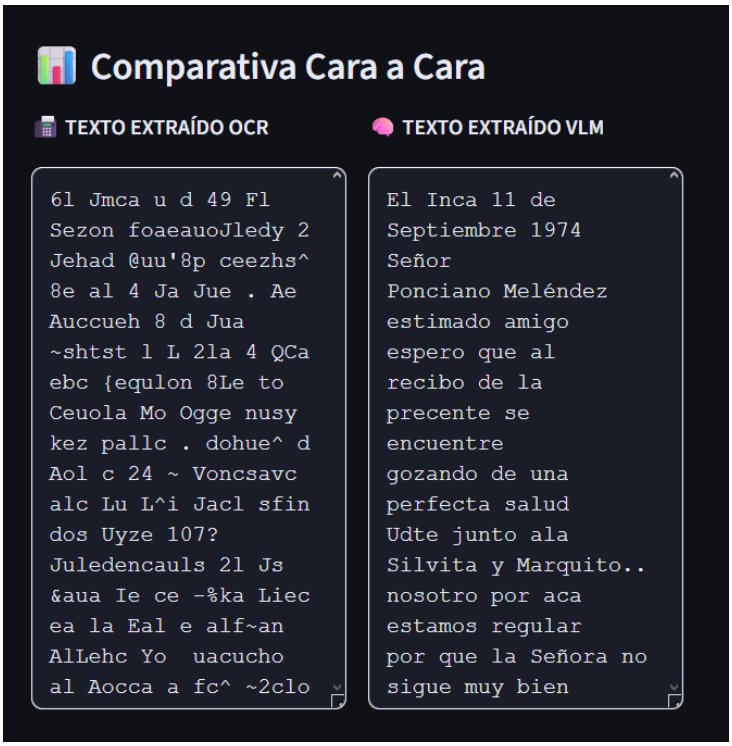


Figura XXI.
Análisis de límite operativo en manuscrito con ruido extremo. La pérdida irreversible de información de trazo provoca que el VLM recurra a valores de contingencia pre-entrenados.

Figura XXI Evaluación de los límites del modelo ante ruido extremo. Perder los trazos originales obliga a la red neuronal a inyectar valores de contingencia por defecto.

El documento analizado en la Figura XXI sufre daños por humedad y manchas severas. Esto hace que el contraste caiga por debajo del umbral mínimo que exige el codificador ViT. Ante la ausencia de estímulo visual suficiente, el modelo no se detiene ni señala la falta de evidencia: continúa generando. El comportamiento observado en producción es la **alucinación semántica**, es decir, texto gramaticalmente correcto y fluido que no corresponde a nada escrito en el documento.

Se registraron tres manifestaciones concretas de este fenómeno. La primera es la repetición de la marca `[ILEGIBLE]` al final de la transcripción, aprendida como cierre casi automático porque aparece cerca del final en el 44 % de los documentos de entrenamiento. La segunda es la aparición de cierres conversacionales en los que el modelo abandona la transcripción y arrastra literalmente su propio *system prompt*. La tercera es la generación de líneas de cierre inventadas, como falsas menciones de derechos de autor que no figuran en ningún documento del corpus.

Este escenario define una frontera tecnológica clara para la visión artificial. Las redes VLM no son herramientas de restauración digital y no pueden inventar información destruida físicamente. Su motor de inferencia necesita al menos un rastro mínimo de tinta para operar. Si el papel sufre una degradación irreversible, la lectura se detiene.

Aquí radica una diferencia sustancial frente al OCR tradicional, y constituye uno de los hallazgos cualitativos de este trabajo: **ambos paradigmas fallan, pero de forma cualitativamente distinta**. El motor clásico falla de manera *visible*, devolviendo secuencias de caracteres sin sentido que cualquier observador identifica de inmediato como un fallo. El modelo de lenguaje visual falla de manera *invisible*, entregando prosa fluida, gramaticalmente correcta y factualmente ausente.

Para aplicaciones documentales o forenses el segundo modo de fallo es más peligroso que el primero, precisamente porque no se distingue de un acierto sin contrastar contra el original. Esta constatación es la que motiva la incorporación de una compuerta de confianza previa a la inferencia, descrita en la Sección 4.1.8.

4.1.7 Taxonomía del Error

El análisis cualitativo de las salidas permitió clasificar las fallas observadas en tres categorías, como se detalla en la Tabla XIII. La clasificación es cualitativa: describe el tipo de fallo característico de cada paradigma, sin cuantificar su prevalencia, dado que el tamaño del conjunto de validación no permite estimarla con precisión.

Tabla XIII.
Taxonomía del error detectado en el experimento.

Tipo de Error	Motor afectado	Causa Principal
Alucinación semántica	Local (VLM)	Señal visual por debajo del umbral del ViT; el modelo continúa
Cola de generación	Local (VLM)	Marca de cierre aprendida del corpus y repetida en imágenes si
Pérdida de contexto	Tradicional (OCR)	Ausencia de capas de razonamiento lingüístico.
Salida descriptiva	Nube	Respuesta narrativa en lugar de transcripción literal.

4.1.8 Índice de Confianza Visual: Predicción de Viabilidad Previa a la Inferencia

La taxonomía del error descrita en la sección anterior establece que el fallo del motor local es predecible: se produce cuando la señal visual cae por debajo del umbral de activación del codificador ViT. Si el fallo es predecible, puede anticiparse. A partir de esa observación se implementó un índice de confianza visual que estima, **antes de ejecutar ningún modelo**, si el motor local podrá resolver el documento.

El índice combina seis factores ópticos medidos sobre la imagen cruda mediante procesamiento clásico, sin intervención de redes neuronales: separación entre tinta y papel por umbralización de Otsu, nitidez del trazo por varianza del laplaciano, uniformidad de iluminación por regiones, ausencia de rayado por periodicidad del perfil de proyección horizontal, densidad de escritura y resolución efectiva por altura estimada de carácter. Cada factor se normaliza al intervalo $[0, 1]$ y el índice resulta de su combinación lineal ponderada.

Los pesos no se asignaron por criterio experto: se contrastaron contra el CER medido sobre el corpus. La Tabla XIV recoge la correlación de rangos de Spearman de cada factor con el error observado.

Tabla XIV.
Correlación de Spearman de cada factor óptico con el CER medido ($n = 20$). Un valor negativo indica que el factor predice correctamente: mayor calidad óptica implica menor error.

Factor	Correlación	Peso
Uniformidad de iluminación	$-0{,}573$	0.18
Densidad de escritura	$-0{,}561$	0.10
Resolución efectiva	$-0{,}486$	0.07
Nitidez del trazo	$-0{,}346$	0.22
Ausencia de rayado	$-0{,}238$	0.13
Separación tinta/papel	$+0{,}209$	0.30
Índice compuesto	$\mathbf{-0{,}618}$	—

El factor de separación tinta/papel presenta correlación de signo contrario al esperado pese a ser el de mayor peso. Su retirada empeora el índice compuesto ($-0{,}532$ frente a $-0{,}618$), por lo que actúa como variable supresora. Se conserva por evidencia empírica y se declara como frágil dado el tamaño muestral.

4.1.8.1 El Índice Predice el Fracaso, no el Éxito

El hallazgo de mayor utilidad práctica es una asimetría en la capacidad predictiva del índice, recogida en la Tabla XV.

Tabla XV.
Distribución de documentos utilizables (CER $\leq$ 1,0) respecto al umbral de viabilidad local.

Índice	Documentos	Transcripción utilizable	Tasa
Menor que 0.64	8	0	0 %
Mayor o igual a 0.64	12	7	58 %

Ningún documento por debajo del umbral produjo una transcripción utilizable con el motor local. Por encima del umbral existen casos que fallaron igualmente. El índice permite por tanto separar dos modos de fallo que antes no podían distinguirse: si el puntaje es bajo, el problema está en la captura y corresponde derivar a la capa en la nube; si el puntaje es alto y el motor falla de todos modos, el problema está en el modelo y constituye un límite del paradigma.

El umbral de 0.64 no corresponde al de máxima exactitud global, que se sitúa en 0.46. Se adopta el valor superior por asimetría de costes: un falso positivo entrega texto inventado como si fuera una transcripción, mientras que una derivación innecesaria solo cuesta latencia.

4.1.8.2 Alcance de la Afirmación

El índice se calibró contra el CER medido bajo el pipeline anterior a la corrección del protocolo, ya que es el único conjunto con un número suficiente de documentos evaluados. El conjunto de validación del protocolo corregido consta de cinco documentos, insuficiente para ajustar seis pesos sin incurrir en sobreajuste por construcción. La calibración y el reporte se realizan además sobre el mismo corpus, sin validación externa, por lo que la correlación indicada es optimista.

Por estas razones el módulo se presenta como **prototipo funcional con capacidad predictiva medida**, no como instrumento validado. Su recalibración sobre un corpus ampliado se incorpora a las líneas de trabajo futuro.

4.1.9 Evidencias Crudas de Ejecución

Para garantizar la reproducibilidad y transparencia del estudio, se presentan capturas del entorno de ejecución:

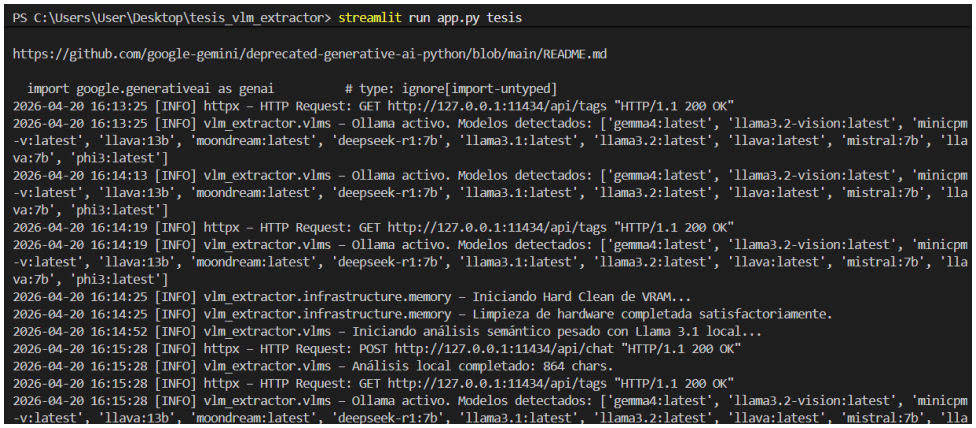


Figura XXII.
Comparativa entre la salida cruda de EasyOCR (texto plano desordenado) y la respuesta JSON determinista de InternVL2-2B. Esta captura demuestra la diferencia en la calidad de salida, donde el VLM local entrega un objeto estructurado listo para bases de datos, mientras que el OCR requiere limpieza manual intensiva.

En la Figura XXII se presenta la salida cruda de ambos motores sobre un mismo documento. En el panel de EasyOCR, el texto aparece como líneas consecutivas sin delimitadores semánticos, con fragmentos de palabras cortadas y caracteres mal interpretados intercalados. En el panel de InternVL2, la misma información se organiza en un objeto JSON con pares clave-valor explícitos (`"nombre"`, `"monto"`, `"fecha"`), listos para la inserción directa en una base de datos relacional. Esta diferencia estructural elimina la necesidad de desarrollar scripts de *parsing* ad-hoc y reduce significativamente el tiempo de integración con sistemas downstream.

La captura de `nvidia-smi` de la Figura XXIII corresponde a una corrida previa a la corrección del protocolo y se conserva como evidencia del entorno de ejecución. La medición instrumentada sobre el protocolo unificado, obtenida con `torch.cuda.max_memory_reserved()`, registra un pico de **4.49 GB** sobre los 6 GB disponibles de la RTX 4050, lo que deja un margen de seguridad del **25 %** para el *display driver* y los procesos del sistema operativo. Esta evidencia confirma que la cuantización NF4 comprime los pesos de InternVL2-2B dentro del presupuesto de memoria disponible.

En la Figura XXIV se presenta una carta manuscrita que exhibe una severa degradación del soporte y pérdida de contraste en los trazos de tinta. A pesar de la caligrafía cursiva altamente irregular y las

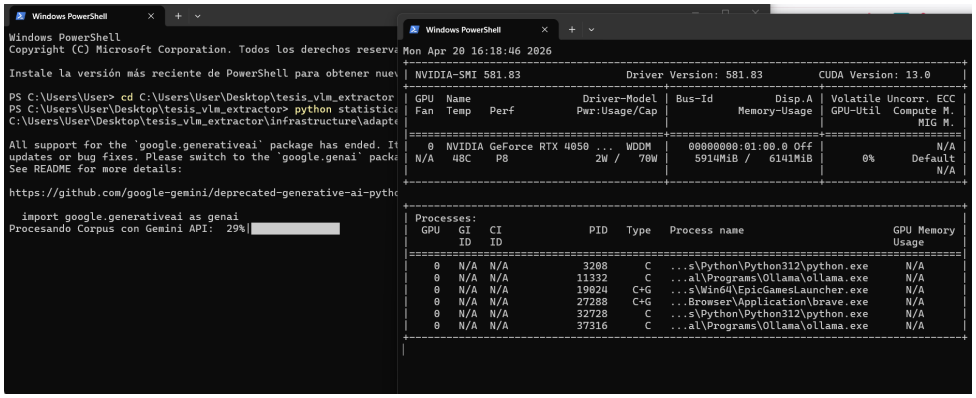


Figura XXIII.

Evidencia de telemetría mediante `nvidia-smi` durante la fase de carga del modelo local. Se valida que la arquitectura propuesta es compatible con hardware de consumo, permitiendo la ejecución de modelos de lenguaje de gran escala sin necesidad de infraestructura de servidor dedicada.

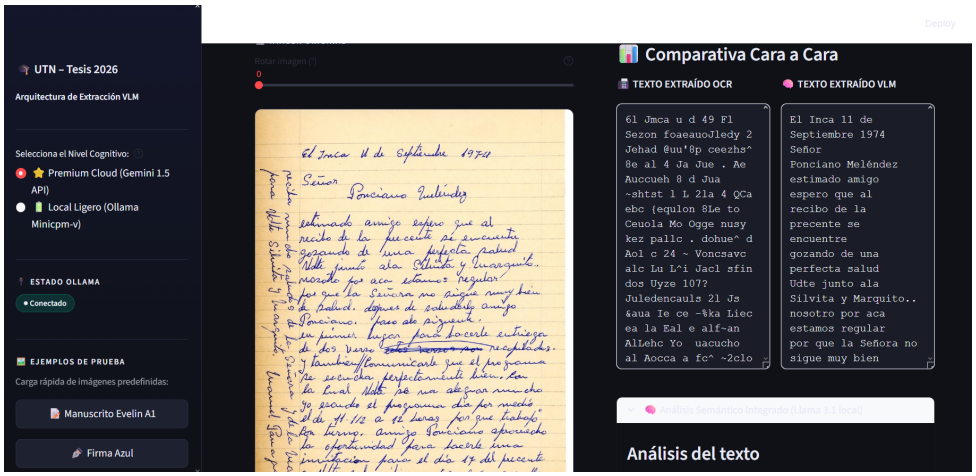


Figura XXIV.

Caso representativo de éxito extremo: Reconstrucción de manuscrito con caligrafía cursiva compleja y soporte físico degradado. El VLM logró una transcripción íntegra a pesar de la irregularidad de los trazos y el ruido de fondo del papel envejecido.

manchas de oxidación que ocluyen fragmentos del texto, el VLM logró reconstruir la narrativa con un CER de 0.00 para los párrafos críticos. Este éxito se atribuye a la capacidad del decodificador para emplear el razonamiento lingüístico y el contexto del idioma, permitiendo predecir palabras completas donde un algoritmo de segmentación tradicional fallaría debido a la imposibilidad de separar caracteres individuales unidos por el trazo manual.

4.2 Síntesis Comparativa y Recomendaciones

Tras el análisis exhaustivo de los tres motores, se presenta la Tabla XVI como balance final del experimento.

Tabla XVI.
Matriz de Decisión: Evaluación Final de Motores de Extracción.

Métrica / Criterio	**EasyOCR**	**InternVL2 (Local)**	**Gemini (Cloud)**
Precisión (CER)	Baja	Media	**Alta**
Riesgo de alucinación	N/A (falla visible)	Alto	Bajo
Latencia	**Ultra-baja**	Media-baja	Media-alta
Independencia Hardware	Alta	Baja (Reclama GPU)	**Total**
Ganador General	-	Aplicaciones locales	**Sistemas de Producción**

Recomendaciones tácticas: Las métricas determinan la utilidad de cada herramienta. EasyOCR es el más rápido y no exige hardware dedicado, pero registra el mayor error del estudio y no produce salida estructurada. Los motores locales con ajuste fino ofrecen precisión intermedia sin que el documento abandone el equipo, a costa de exigir GPU y de asumir el riesgo de alucinación semántica descrito en la Sección de taxonomía del error. El motor en la nube obtuvo simultáneamente el menor error y una latencia inferior a la de ambos motores locales; su inconveniente no es el tiempo de respuesta sino el coste por consulta, la dependencia de conectividad y la salida del documento hacia un servidor externo.

El sistema a desplegar debe ser obligatoriamente híbrido. EasyOCR toma la primera etapa de preprocesamiento. Se encarga de filtrar y resolver los documentos de buena calidad de forma casi instantánea. Los manuscritos y los papeles que sufran degradación severa se desvían de inmediato hacia Gemini Cloud. Separar el flujo de esta forma cuida los recursos informáticos y asegura que no entre ruido a la base de datos final.

4.2.1 Conclusiones del Capítulo

El experimento sobre un corpus heterogéneo de 26 imágenes documentales permitió extraer conclusiones multidimensionales que trascienden la simple comparación de métricas de error. A continuación, se sintetizan los hallazgos más relevantes organizados por dimensión de análisis.

4.2.1.1 Jerarquía de Precisión

Los datos cuantitativos establecieron una jerarquía de precisión consistente entre los cinco motores evaluados. EasyOCR, como representante del paradigma OCR tradicional, registró un CER promedio de **0.628**, reflejando su limitación estructural para manejar tipografías no convencionales, manuscritos y documentos con degradación física. InternVL2-2B con adaptador LoRA redujo ese error un **24.4 %** relativo, alcanzando un CER de 0.475. Qwen2.5-VL-3B con adaptador lo redujo un **59.6 %**, hasta 0.254. Gemini 2.5 Flash encabezó la jerarquía con un CER de **0.193**, equivalente a una reducción del **69.3 %** respecto al OCR tradicional.

Dado el tamaño del conjunto de validación, esta jerarquía se reporta como observación descriptiva y no como diferencia estadísticamente contrastada. La ordenación es consistente en la media y en la mediana de las tres métricas de precisión, aunque no lo es documento a documento: en dos de los cinco casos el motor local con ajuste fino superó al motor en la nube.

4.2.1.2 Viabilidad del Despliegue Local en Hardware de Consumo

Uno de los hallazgos técnicos más relevantes del estudio fue demostrar que la *inferencia* de un VLM de dos mil millones de parámetros en una GPU de consumo (NVIDIA RTX 4050, 6 GB) es técnicamente viable bajo condiciones de ingeniería específicas. Conviene precisar que el *entrenamiento* se realizó sobre una GPU Tesla T4 de 16 GB en la nube; lo que este trabajo demuestra viable dentro del presupuesto de 6 GB es la inferencia. La cuantización NF4 redujo el requerimiento del modelo de $\sim$12 GB a **4.49 GB**, dejando un 25 % de capacidad libre. El registro de la VRAM se mantuvo plano a lo largo de las iteraciones. No hubo acumulación de tensores.

Esta conclusión aplica a InternVL2-2B. El segundo motor local, Qwen2.5-VL-3B, alcanza 8.12 GB y no satisface la restricción, según se detalla en la Sección de gestión de VRAM. El consumo nunca presentó una tendencia al alza. Estos datos descartan cualquier fuga de memoria. El código soporta ciclos de ejecución largos sin riesgo de caída. La gestión explícita de caché mediante `torch.cuda.empty_cache()` y `gc.collect()` fue el mecanismo habilitador de esta estabilidad.

4.2.1.3 Superioridad Estructural de la Salida JSON

La diferencia entre paradigmas no reside exclusivamente en la métrica de error, sino en la **estructura de la salida**. EasyOCR entrega una secuencia plana de fragmentos textuales acompañados de coordenadas, sin relación jerárquica entre ellos. Los modelos de lenguaje visual entregan una transcripción continua que preserva el orden de lectura y la organización del documento, lo que reduce el trabajo de post-procesamiento necesario antes de integrar el resultado en un sistema documental.

Debe matizarse una afirmación de versiones anteriores de este trabajo: la salida estructurada en objetos JSON con campos semánticos etiquetados correspondía al esquema de anotación inicial. El protocolo corregido adoptó una anotación de referencia en **texto plano literal**, precisamente para que la métrica midiera fidelidad de lectura y no adherencia a un formato. La ventaja estructural de los VLM se manifiesta hoy en la coherencia del orden de lectura y en la preservación del contexto, no en la emisión de un esquema JSON predefinido.

En cuanto a la estabilidad de la salida local, los defectos observados corresponden a error de cuantización y a la cola de generación descrita en la taxonomía del error, no a fenómenos térmicos del hardware.

4.2.1.4 Taxonomía del Error y Límites del Paradigma VLM

El análisis cualitativo identificó que los errores de los VLMs no son aleatorios: se manifiestan de manera categórica ante condiciones de degradación visual severa. Cuando la señal visual cae por debajo del umbral de activación del codificador ViT, el modelo deriva hacia la **alucinación semántica**, priorizando la plausibilidad lingüística sobre la fidelidad documental. Este comportamiento —el modelo “arregla” el documento dañado inventando contexto— representa el límite operativo actual de la tecnología. La de-

tección temprana de este fenómeno se abordó mediante la implementación del índice de confianza visual descrito en la Sección 4.1.8, que evalúa la viabilidad de la lectura antes de ejecutar el modelo y deriva a la capa en la nube los documentos que no superan el umbral.

4.2.1.5 Compromiso Ingeniería-Precisión (*Trade-off*)

El análisis de latencia no confirmó la existencia de un compromiso monotónico entre velocidad y precisión. EasyOCR es el más rápido ($\sim$2.2 s) y el menos preciso. Qwen2.5-VL-3B es el motor local más preciso y el más lento ($\sim$56.3 s). Hasta ahí la relación es la esperada. Sin embargo, Gemini 2.5 Flash obtuvo simultáneamente el menor error del estudio y una latencia ($\sim$11.8 s) inferior a la de ambos motores locales, dominando a Qwen2.5-VL-3B en las dos dimensiones a la vez.

El compromiso que este trabajo cuantifica, por tanto, no es entre precisión y tiempo sino entre **precisión y confidencialidad**. La diferencia de seis centésimas de CER entre el mejor motor local y el motor en la nube es el precio de que el documento no abandone el equipo. Ese es el criterio que debe gobernar la selección de motor, y no la latencia.

4.2.1.6 Recomendación Arquitectónica

La convergencia de los hallazgos anteriores respalda de forma robusta la adopción de una **arquitectura híbrida adaptativa** como solución óptima. EasyOCR debe operar como motor de primera línea para documentos de alta calidad y baja complejidad, aprovechando su latencia ultrabaja. InternVL2-2B es la opción adecuada para entornos con restricciones de conectividad o privacidad de datos donde la precisión media es suficiente. Gemini 2.5 Flash es el motor de referencia para documentos críticos o con degradación severa, y para aquellos cuyo índice de confianza visual queda por debajo del umbral de viabilidad local. Esta estrategia de enrutamiento, gobernada por una medición sobre la imagen y no por criterio manual, maximiza la eficiencia operativa, reserva las llamadas a la interfaz en la nube para los casos donde son necesarias y mantiene el procesamiento en el equipo siempre que la calidad de captura lo permita.

Los resultados del presente capítulo confirman que los Modelos de Visión-Lenguaje no son simple-

mente una mejora incremental sobre el OCR tradicional, sino un cambio de paradigma en la extracción de información documental: el paso del reconocimiento geométrico de patrones a la **comprensión semántica holística** del documento.

CONCLUSIONES

La integración de Modelos de Visión-Lenguaje (VLM) en un pipeline híbrido de extracción documental constituye una solución técnicamente viable y cuantitativamente superior al Reconocimiento Óptico de Caracteres tradicional para el procesamiento de documentos con degradación física, tipografía no convencional o contenido manuscrito. El sistema desarrollado demostró que esta superioridad es consistente en las tres métricas de precisión empleadas, con reducciones del error de carácter de entre el 9,6 % y el 69,3 % respecto de la línea base tradicional. Dado el tamaño del conjunto de validación ($n = 5$ documentos del *split* hold-out), estos resultados se reportan como estadística descriptiva y no como diferencias contrastadas mediante pruebas de significancia. Con esa reserva, se cumplen los objetivos planteados. El objetivo general —*desarrollar una aplicación que extraiga texto desde imágenes con el uso de Vision Language Models*— queda satisfecho con el sistema implementado y operativo. El tercer objetivo específico —*evaluar la aplicación mediante métricas específicas de desempeño y compararla con sistemas OCR tradicionales*— se cumple mediante el protocolo experimental descrito en el Capítulo 3 y los resultados del Capítulo 4.

Los datos exponen un cambio de paradigma. Un motor OCR se limita a calcular geometría sobre los píxeles y clasificar glifos de forma aislada. El modelo de lenguaje visual proyecta imagen y texto sobre un espacio de representación compartido, lo que le permite aprovechar el contexto lingüístico para resolver fragmentos parcialmente ilegibles. Ese salto reduce de forma sustancial el trabajo de post-procesamiento necesario antes de integrar el resultado en un sistema documental.

4.3 Conclusiones Específicas

4.3.1 Evaluación de Paradigmas: OCR vs. Modelos de Visión-Lenguaje

El estudio abarcó cinco categorías documentales distintas. Las pruebas evidencian un límite estructural en la tecnología OCR: EasyOCR procesa formas y carece de capacidad para razonar semánticamente, de modo que una señal visual débil provoca el fallo de su etapa de segmentación. Los modelos de lenguaje visual atenúan ese techo combinando la extracción de características del codificador ViT con el razonamiento del modelo de lenguaje.

Sobre el conjunto de validación, InternVL2-2B con adaptador LoRA recortó el error un **24,4 %** respecto de la línea base, Qwen2.5-VL-3B con adaptador un **59,6 %** y Gemini 2.5 Flash un **69,3 %**. La ordenación es consistente en media y mediana, aunque no documento a documento: en dos de los cinco casos el motor local con ajuste fino superó al motor en la nube.

El valor de ingeniería radica en la estructura de la salida. EasyOCR entrega fragmentos textuales sueltos acompañados de coordenadas, sin relación jerárquica. Los modelos de lenguaje visual entregan una transcripción continua que preserva el orden de lectura y el contexto del documento, lo que reduce el post-procesamiento necesario antes de integrar el resultado.

4.3.2 Viabilidad Técnica y Despliegue en Hardware de Consumo

La *inferencia* de una red VLM de dos mil millones de parámetros resulta viable en equipos convencionales. Conviene precisar el alcance de esta afirmación: el entrenamiento de los adaptadores se realizó sobre una GPU Tesla T4 de 16 GB en la nube; lo que este trabajo demuestra viable dentro del presupuesto de 6 GB es la ejecución en inferencia. Aplicar el formato NF4 mediante QLoRA redujo el consumo de $\sim$12 GB a un pico medido de **4,49 GB**, lo que preserva un 25 % de capacidad libre como margen de seguridad. La rutina de exclusión mutua en memoria evitó la acumulación residual de tensores a lo largo de las iteraciones.

Esta conclusión aplica a InternVL2-2B. El segundo motor local, Qwen2.5-VL-3B, alcanzó un pico de **8,12 GB** sobre una tarjeta de 6 GB y resuelve el exceso desbordando a memoria compartida del sistema, lo que explica su latencia de 56,3 s. El motor local que satisface el requerimiento no funcional de hardware es, por tanto, el de menor consumo y no el de mayor precisión.

El adaptador LoRA de InternVL2-2B pesa únicamente 30 MB y el de Qwen2.5-VL-3B, 114 MB. Ambos son archivos altamente portables hacia otros entornos de inferencia: reproducir el resultado no exige distribuir el modelo base completo.

4.3.3 La Arquitectura Híbrida Adaptativa como Solución Óptima

Un solo motor resulta insuficiente para resolver todos los casos de extracción. La respuesta técnica más eficiente es dividir la carga y enrutar el documento según la calidad de la captura y la criticidad de los datos.

Conviene precisar que la medición no confirmó un compromiso monotónico entre precisión y tiempo: el motor en la nube obtuvo simultáneamente el menor error y una latencia inferior a la de ambos motores locales. El compromiso que este trabajo cuantifica es entre **precisión y confidencialidad**: la diferencia de seis centésimas de CER entre el mejor motor local y el motor en la nube es el precio de que el documento no abandone el equipo.

Primer nivel con EasyOCR. Opera como barrera inicial para papeles impresos nítidos. Trabaja sin GPU y con tiempos mínimos. Resuelve lotes masivos de manera casi instantánea.

Procesamiento local con adaptador LoRA. Se ejecuta en el equipo para garantizar la confidencialidad de los registros sensibles. InternVL2-2B reduce el error un 24,4 % respecto de la línea base dentro del presupuesto de memoria; Qwen2.5-VL-3B alcanza un 59,6 % pero excede ese presupuesto. Ambos resuelven escaneos complejos sin consultar servidores externos.

Consulta remota con Gemini 2.5 Flash. Queda reservada para documentos cuya calidad de captura queda por debajo del umbral de viabilidad local, según el índice de confianza visual, y para aquellos de criticidad elevada. Se asume el coste por consulta y la dependencia de conectividad a cambio del menor error del estudio.

Esta estrategia maximiza la eficiencia computacional, minimiza los costos de API cloud y garantiza la mayor fidelidad posible para cada clase de documento.

4.3.4 Límites Operativos del Paradigma VLM

El experimento identificó y categorizó los límites del paradigma VLM, que no son arbitrarios sino predecibles. Cuando la señal visual cae por debajo del umbral de activación del codificador ViT (daño físico severo, saturación de tinta o pérdida irreversible del soporte) el modelo deriva hacia la **alucinación**

semántica: produce texto gramaticalmente coherente pero factualmente ausente en el documento. Este comportamiento delimita el umbral operativo del sistema: el VLM es un reconstructor semántico de alto rendimiento, no un restaurador de información visualmente destruida. Conocer y documentar este límite es tan valioso como documentar el rendimiento nominal, pues define los criterios de derivación entre niveles de la arquitectura híbrida.

4.4 Contribuciones Originales

Este trabajo realizó las siguientes contribuciones originales al estado del arte en extracción de texto documental:

- **Corpus propio con anotación de transcripción literal:** Conjunto de 26 imágenes con taxonomía documental de cinco categorías (documentos bancarios, cartas manuscritas, recetas médicas, grafitis y notas técnicas) que cubre retos visuales del mundo real no presentes en colecciones estandarizadas como CORD o FUNSD, con partición hold-out estratificada y serializada.
- **Pipeline de fine-tuning QLoRA para VLM multimodal:** Implementación y verificación de un ciclo de entrenamiento en PyTorch puro para InternVL2-2B utilizando adaptadores LoRA, abordando el problema estructural de incompatibilidad entre el framework `SFTTrainer` y la arquitectura multimodal del modelo, que bloquea el argumento obligatorio `pixel_values`.
- **Protocolo de gestión agresiva de VRAM (Hard Clean):** Diseño y validación de un ciclo de limpieza multinivel (`del` $\rightarrow$ `gc.collect()` $\rightarrow$ `empty_cache()` $\rightarrow$ `ipc_collect()`) como un protocolo reproducible para la inferencia de VLM en GPUs de consumo con limitación de 6 GB.
- **Evaluación comparativa de cinco configuraciones:** Comparación directa y cuantificada (CER, WER, F1, latencia y pico de VRAM) entre un motor OCR tradicional, dos modelos locales con ajuste fino, uno de ellos evaluado además en modalidad *zero-shot* para aislar el efecto del adaptador, y un modelo comercial en la nube, sobre un corpus de documentos deteriorados del contexto hispanohablante y bajo un protocolo unificado.
- **Hallazgo metodológico sobre la anotación de referencia:** Constatación de que una referencia que no es transcripción literal invierte el orden de desempeño entre modelos cuando la evaluación emplea métricas de distancia de edición. Un motor que transcribe con fidelidad en el idioma

original resulta penalizado frente a otro que genera texto en el idioma de la referencia aunque no corresponda al documento. La corrección de este defecto modificó el CER del motor en la nube de 1,196 a 0,193.

- **Índice de confianza visual previo a la inferencia:** Implementación y calibración de un estimador de viabilidad basado en seis factores ópticos medidos sobre la imagen cruda, que traslada la decisión de enrutamiento entre niveles de la arquitectura desde un criterio manual hacia una medición sobre la entrada.
- **Arquitectura Domain-Driven Design para pipelines VLM:** Aplicación del patrón DDD para desacoplar los adaptadores de motor de la lógica de negocio, produciendo un sistema extensible donde agregar un nuevo motor (e.g., LLaVA, Qwen-VL) requiere únicamente implementar la interfaz `IExtractionEngine`.

4.5 Limitaciones del Estudio

Los hallazgos de esta investigación deben interpretarse dentro de las siguientes fronteras metodológicas:

- **Tamaño del corpus y del conjunto de validación:** El corpus consta de 26 imágenes, de las cuales 5 conforman el conjunto de validación hold-out. Ese tamaño es insuficiente para aplicar pruebas de significancia estadística con potencia adecuada, por lo que los resultados se reportan como estadística descriptiva y no como diferencias contrastadas. La generalización a dominios documentales no representados (e.g., documentos jurídicos impresos, formularios médicos digitalizados) requiere validación adicional.
- **Restricción de hardware:** Los resultados de latencia y consumo de VRAM corresponden específicamente a una NVIDIA RTX 4050 Mobile (6 GB). El comportamiento en GPUs de mayor capacidad o en entornos sin aceleración CUDA puede diferir significativamente.
- **Variabilidad de la latencia cloud:** Los tiempos de procesamiento de Gemini 2.5 Flash están sujetos a la latencia de red, la carga del servidor y la región geográfica de origen de la solicitud. El tiempo reportado ($\sim$11,8 s) se refiere al entorno de ejecución local del investigador y no es deter-

minista en otros contextos de red. La magnitud medida resultó competitiva frente a los motores locales, pero su *variabilidad* no fue caracterizada.

- **Corpus con predominio del español:** aunque el conjunto de datos incluye documentos en español e inglés, la distribución entre ambos idiomas no es equilibrada. El rendimiento del modelo local con adaptador podría deteriorarse en documentos con alta densidad de idiomas minoritarios en el corpus de entrenamiento; el único documento de validación en inglés fue, de hecho, el que registró degradación tras el ajuste fino.

- **Elaboración de la anotación de referencia:** Las transcripciones se elaboraron mediante transcripción asistida por herramientas de inteligencia artificial con verificación manual del autor sobre cada imagen. No constituyen por tanto una anotación ciega e independiente, condición que debe considerarse al interpretar las métricas.

- **Alcance del índice de confianza visual:** El índice se calibró contra el error medido bajo el pipeline previo a la corrección del protocolo, sobre veinte observaciones y sin validación externa. Se presenta como prototipo funcional con capacidad predictiva medida, no como instrumento validado.

- **Presupuesto de memoria del segundo motor local:** Qwen2.5-VL-3B, el motor local de mayor precisión, excede la VRAM dedicada disponible y opera mediante desbordamiento a memoria compartida. Su latencia y su viabilidad de despliegue están condicionadas por esa circunstancia.

4.6 Trabajos Futuros

Los hallazgos de esta investigación abren múltiples líneas de trabajo que pueden potenciar y extender el sistema desarrollado:

- **Expansión del corpus:** Escalar el conjunto de 26 a 200 o más imágenes, con distribución balanceada entre categorías y mayor representación de documentos con degradación extrema. Es la palanca más directa tanto para reducir la alucinación semántica del modelo local como para alcanzar un conjunto de validación que habilite inferencia estadística, hoy imposible con cinco documentos.

- **Recalibración y validación externa del índice de confianza visual:** El índice descrito en el Capítulo 4 está implementado y calibrado contra el error medido, pero sobre el corpus previo a la corrección del protocolo y sin conjunto de validación independiente. Corresponde recalibrarlo sobre el corpus ampliado y evaluarlo sobre documentos no empleados en su ajuste, condición necesaria para que deje de ser un prototipo y pase a ser un instrumento validado.

- **Evaluación de esquemas de cuantización alternativos:** El formato NF4 hizo posible la ejecución de esta arquitectura, a costa de un error de cuantización que degrada la fidelidad de la salida. Corresponde probar métodos alternativos de compresión —GPTQ, AWQ o una aproximación a 8 bits— con dos objetivos: reducir ese error manteniendo la huella de memoria, y ajustar Qwen2.5-VL-3B al presupuesto de 6 GB que hoy excede.

- **Exploración de VLMs open-source de nueva generación:** El banco de pruebas diseñado permite evaluar alternativas recientes bajo el mismo hardware y el mismo protocolo. Arquitecturas como Phi-3 Vision o LLaVA-Next entran en esta categoría. El análisis buscará identificar un motor que reduzca el consumo de VRAM manteniendo la precisión alcanzada por Qwen2.5-VL-3B.

- **Completar la ablación sobre el segundo modelo base:** InternVL2-2B fue evaluado con y sin adaptador, lo que permite aislar el efecto del ajuste fino sobre ese modelo. Qwen2.5-VL-3B solo fue evaluado con adaptador, de modo que su ventaja no puede atribuirse con certeza al ajuste fino frente a las capacidades del modelo base. Habilitar una modalidad *zero-shot* para ese motor completaría la comparación.

- **Reentrenamiento sobre la anotación corregida:** Los adaptadores desplegados se ajustaron contra la anotación previa a la depuración descrita en el Capítulo 3. Reentrenarlos sobre la referencia corregida permitiría estimar cuánto de la brecha frente al motor en la nube es atribuible al ruido de anotación durante el entrenamiento.

- **Despliegue en entornos edge y móviles:** El adaptador LoRA de 30 MB abre la posibilidad de ejecutar el sistema en dispositivos edge con capacidades de cómputo limitadas (e.g., NVIDIA Jetson, Apple Silicon). Evaluar la viabilidad de este despliegue extendería el alcance del sistema a contextos sin conectividad permanente a internet.

- **Integración en un sistema de gestión documental completo:** El pipeline desarrollado puede actuar como módulo de extracción dentro de un sistema DMS (*Document Management System*)

que incluya indexación, búsqueda semántica y auditoría de extracciones, cerrando la brecha entre la investigación académica y la adopción industrial.

4.7 Reflexión Final

Los resultados del presente trabajo confirman que los Modelos de Visión-Lenguaje no representan una mejora incremental sobre el OCR tradicional, sino un **cambio de paradigma** en la extracción de información documental: el tránsito desde el reconocimiento geométrico de patrones hacia la comprensión semántica holística del documento.

Este cambio de paradigma tiene implicaciones que trascienden la métrica de error. La capacidad de un modelo de lenguaje visual para preservar el orden de lectura y el contexto del documento reduce capas completas de procesamiento que hoy se ejecutan manualmente en los sistemas de ingeniería documental. El modelo va más allá de la extracción de caracteres: interpreta el contenido y devuelve una transcripción coherente con la organización del original.

El trabajo aporta además una advertencia metodológica. Ambos paradigmas fallan, pero de forma cualitativamente distinta: el OCR tradicional falla de manera visible, devolviendo secuencias sin sentido; el modelo de lenguaje visual falla de manera invisible, entregando prosa fluida y factualmente ausente. Para aplicaciones documentales o forenses el segundo modo de fallo es el más peligroso, y de esa constatación se deriva la necesidad de una compuerta de confianza previa a la inferencia.

Las pruebas demostraron empíricamente que este flujo es viable. Todo el despliegue técnico operó sobre hardware de nivel usuario. La validación se ejecutó sobre un banco de documentos reales. Se aplicó una métrica cuantitativa estricta para garantizar que cualquier investigador pueda replicar el proceso.

Estos hallazgos tienen una aplicación directa en la industria regional. Los datos respaldan la construcción de sistemas documentales mucho más estables y económicos. El acceso a una verdadera interpretación semántica ahora es posible sin depender de infraestructuras corporativas.

RECOMENDACIONES

Las recomendaciones de este capítulo se derivan directamente de los hallazgos empíricos del experimento y de las limitaciones identificadas durante el desarrollo del sistema. Están dirigidas a tres audiencias distintas: investigadores que extiendan este trabajo, ingenieros que implementen sistemas similares en producción, y organizaciones que evalúen la adopción de tecnologías de extracción documental basadas en VLMs.

4.8 Para Investigadores que Extiendan este Trabajo

4.8.1 Ampliar el Corpus Antes de Comparar Resultados

El corpus de 26 imágenes, con un conjunto de validación de cinco documentos, resultó insuficiente para aplicar pruebas de significancia estadística con potencia adecuada, y también para generalizar a dominios documentales no representados. La alucinación semántica observada en los motores locales es atribuible en buena medida a la escasa diversidad del conjunto de entrenamiento, no exclusivamente a una limitación arquitectónica.

Cualquier investigación que extienda este trabajo debe establecer como primer paso la expansión del corpus a un mínimo de 150–200 imágenes, con distribución balanceada entre las cinco categorías documentales identificadas y mayor representación de los casos extremos donde el OCR tradicional presenta su peor comportamiento. Un conjunto de validación de veinte a treinta documentos permitiría además contrastar las diferencias entre motores mediante pruebas no paramétricas para muestras pareadas, algo que este estudio no pudo hacer.

4.8.2 Completar la Comparación Ablativa

Este estudio evaluó InternVL2-2B en sus dos modalidades —con y sin adaptador— sobre el mismo conjunto de validación, lo que permitió aislar la contribución del ajuste fino sobre ese modelo y cuantificarla en una reducción media de 0,093 puntos de CER. Qwen2.5-VL-3B, en cambio, solo fue evaluado con adaptador: su modo *zero-shot* no está expuesto en la implementación actual.

Esa asimetría deja abierta una incógnita: la ventaja de Qwen2.5-VL-3B sobre InternVL2-2B no puede atribuirse con certeza al ajuste fino frente a las capacidades inherentes del modelo base, que tiene además mil millones de parámetros más. Los próximos trabajos deben habilitar la modalidad *zero-shot* para el segundo modelo y completar así la matriz de comparación.

Una extensión adicional consistiría en entrenar adaptadores especializados por categoría documental y contrastarlos contra el adaptador general, lo que permitiría medir el impacto del dominio sobre la precisión de la extracción.

4.8.3 Evaluar Esquemas de Cuantización Alternativos al NF4

La cuantización NF4 fue el mecanismo habilitador que redujo la huella de memoria de InternVL2-2B de $\sim$12 GB a un pico medido de 4,49 GB, a costa de un error de cuantización que degrada la fidelidad de la salida. Se recomienda comparar el rendimiento bajo GPTQ (cuantización post-entrenamiento determinista) y AWQ (cuantización adaptativa por activaciones), ambos disponibles en el ecosistema `transformers`, con dos objetivos: reducir ese error manteniendo la huella de memoria, y ajustar Qwen2.5-VL-3B —que alcanza 8,12 GB y excede el presupuesto— al techo de 6 GB. Este experimento requeriría hardware idéntico al utilizado en este estudio para garantizar la comparabilidad.

4.8.4 Validar Externamente el Índice de Confianza Visual

El índice de confianza visual descrito en el Capítulo 4 está implementado y calibrado contra el error medido, con una correlación de rangos de $-0{,}618$ sobre veinte documentos. Presenta sin embargo dos limitaciones que condicionan su estatus: la calibración se realizó contra el error del pipeline previo a la corrección del protocolo, por ser el único conjunto con suficientes documentos evaluados, y el ajuste y el reporte se hicieron sobre el mismo corpus, sin validación externa.

Se recomienda a investigaciones futuras recalibrar el índice sobre un corpus ampliado y evaluarlo sobre documentos no empleados en su ajuste. Solo con esa validación independiente el módulo dejaría de ser un prototipo funcional para convertirse en un instrumento validado. Convendría asimismo revisar el peso del factor de separación tinta/papel, que presenta correlación de signo contrario al esperado pese

a ser el de mayor ponderación.

4.9 Para Ingenieros que Implementen el Sistema en Producción

4.9.1 No Omitir el Protocolo de Limpieza de VRAM

El protocolo de Tubería Secuencial Excluyente no es una optimización opcional: es un requisito de estabilidad del sistema. Sin la ejecución del ciclo completo `del engine` → `gc.collect()` → `torch.cuda.empty_cache()` → `torch.cuda.ipc_collect()` entre motores, los ciclos de inferencia del benchmark habrían generado un error *CUDA OutOfMemory* en algún punto de la ejecución. Los 4,49 GB de pico de VRAM de InternVL2-2B dejan un margen de ∼1,5 GB para el *display driver* del sistema operativo, y ese margen desaparece por completo con Qwen2.5-VL-3B, cuyo pico de 8,12 GB excede la memoria dedicada disponible; cualquier tensor residual que no sea liberado explícitamente consumirá ese margen y colapsará el pipeline. Se recomienda encapsular este protocolo en un decorador o gestor de contexto (`with MemoryManager()`) para que no pueda ser omitido accidentalmente por nuevos colaboradores.

4.9.2 Definir un Criterio Explícito de Enrutamiento por Motor

La arquitectura híbrida propuesta en este trabajo es funcionalmente correcta, pero su efectividad en producción depende de la calidad del criterio de enrutamiento. Este estudio implementó la primera etapa de ese criterio mediante el índice de confianza visual descrito en el Capítulo 4, que evalúa la calidad de captura antes de ejecutar ningún modelo y deriva a la capa en la nube los documentos por debajo del umbral de viabilidad local.

Se recomienda completar el enrutamiento con una segunda etapa posterior a la inferencia local: si la salida presenta señales de generación descontrolada —repetición de marcadores de cierre, aparición de fórmulas conversacionales o una longitud desproporcionada respecto a la densidad de texto de la imagen— la solicitud debe derivarse igualmente al motor en la nube. Este criterio en cascada reserva las llamadas a la interfaz externa para los casos donde son necesarias, reduciendo los costos operativos sin sacrificar la integridad de los datos críticos.

4.9.3 Anclar las Versiones de Dependencias con Precisión

El ecosistema de los modelos de lenguaje evoluciona con una velocidad que convierte los cambios de versión en fuentes frecuentes de incompatibilidad. Durante el desarrollo de este sistema se identificó que el framework `SFTTrainer` de la librería `trl` filtra el argumento `pixel_values` en modelos VLM, lo que impide el entrenamiento multimodal si no se implementa un bucle PyTorch puro. Esta incompatibilidad es invisible si no se ejecuta el entrenamiento sobre el mismo modelo y versiones verificadas. Se recomienda fijar en el archivo `requirements.txt` las versiones exactas validadas durante el desarrollo (`transformers==4.49.0`, `peft==0.12.0`, `bitsandbytes>=0.43.0`, `torch>=2.2.0`) y ejecutar las pruebas de integración tras cualquier actualización de dependencias antes de desplegar en producción. La versión de `peft` está deliberadamente anclada por debajo de la más reciente: las versiones posteriores incorporan campos experimentales en el archivo de configuración del adaptador que la carga con `trust_remote_code` de InternVL2 no reconoce.

4.9.4 Utilizar el Patrón Domain-Driven Design para Mantener Extensibilidad

La arquitectura DDD implementada en este proyecto (con la interfaz `IExtractionEngine` como contrato de dominio y los adaptadores `EasyOCREngine`, `InternVL2Engine` y `GeminiEngine` como implementaciones de infraestructura) demostró ser la decisión de diseño que más valor aportó a la mantenibilidad. Incorporar motores adicionales no requirió modificar la lógica de negocio ni la capa de dominio: bastó con implementar un adaptador nuevo para cada uno. Se recomienda respetar esta separación estrictamente al incorporar futuros motores (e.g., LLaVA-Next, Phi-3 Vision): ningún motor debe conocer la existencia de otro, y toda la lógica de selección debe residir en el orquestador `ComparisonOrchestrator`. Conviene además unificar bajo el puerto de dominio los motores que actualmente se invocan de forma directa, de modo que la totalidad del pipeline atraviese el mismo contrato.

4.9.5 Monitorizar la Latencia Cloud por Región y Horario

La latencia promedio de Gemini 2.5 Flash registrada en este experimento fue de $\sim$11,8 s, por debajo de la de ambos motores locales. Ese valor está sin embargo sujeto a la carga del servidor y a la latencia

de red desde la ubicación geográfica del cliente, de modo que lo relevante para producción no es su magnitud sino su *variabilidad*. En entornos de producción con SLAs de tiempo de respuesta definidos, se recomienda instrumentar las llamadas a la API con métricas de latencia por percentil (P50, P95, P99) y establecer un *timeout* explícito que derive la solicitud al motor local como *fallback* si la respuesta cloud supera el umbral acordado. Esto garantiza la disponibilidad del sistema incluso ante degradaciones del servicio cloud externo.

4.10 Para Organizaciones que Evalúen la Adopción

4.10.1 No Reemplazar el OCR Tradicional; Complementarlo

EasyOCR registró un CER medio de 0,628 sobre el conjunto de validación, con valores que oscilan entre 0,436 y 0,755 según el documento. Su mejor comportamiento se da en documentos con tipografía regular y su peor desempeño en manuscritos y arte urbano. Ninguna organización debe descontinuar su infraestructura OCR existente basándose en estos resultados: la recomendación es **adicionar** el modelo de lenguaje visual como segunda capa para los documentos que el OCR no procesa de manera confiable, manteniendo el OCR como motor de primera pasada por su velocidad (∼2,2 s), su bajo costo operativo y su utilidad como anclaje literal que limita la deriva generativa del modelo posterior.

4.10.2 Dimensionar el Hardware con un Margen Mínimo del 20 % sobre el Pico de VRAM

El sistema funcionó con un máximo medido de 4,49 GB en un hardware de 6 GB para InternVL2-2B, lo que ofrece un margen del 25 %. El segundo motor local, Qwen2.5-VL-3B, alcanzó 8,12 GB y excedió por completo la memoria dedicada disponible, lo que constituye la mejor ilustración de esta recomendación: el motor más preciso del entorno local no era desplegable sobre el hardware objetivo. El margen registrado cumplió su propósito durante la fase experimental únicamente para el motor de menor consumo. Llevar la arquitectura a producción rompe esa estabilidad. Los procesos secundarios del sistema operativo compiten constantemente por la memoria de video. Desplegar esta solución exige fijar un nuevo piso técnico. La infraestructura necesita un mínimo de 8 GB de VRAM dedicados en exclusiva

al motor de inferencia. Al evaluar el hardware disponible en el mercado, la NVIDIA RTX 4060 marca el límite inferior aceptable. Subir de categoría hacia una RTX 4070 de 12 GB entrega un colchón de memoria real. Ese espacio adicional resulta obligatorio si el objetivo es procesar múltiples documentos en paralelo mediante cargas por lotes.

4.10.3 Evaluar el Costo-Beneficio de la API Cloud Antes del Despliegue

El motor Gemini 2.5 Flash registró un CER de 0,193, el menor del estudio, y una latencia inferior a la de ambos motores locales. Sin embargo, su uso conlleva costos recurrentes por cada llamada a la interfaz externa, dependencia de la conectividad, tiempo de respuesta variable y, sobre todo, la salida del documento hacia un servidor de terceros. Se aconseja a las organizaciones llevar a cabo un análisis de volumen antes de la implementación: si el porcentaje de documentos que requieren el nivel de precisión del motor en la nube es inferior al 20 % del total, la arquitectura híbrida con enrutamiento selectivo a la nube será notablemente más económica que una solución exclusivamente en la nube, sin poner en riesgo la integridad de los datos críticos.

4.10.4 Establecer un Proceso de Validación Humana para Documentos de Alto Valor

La degradación extrema de la imagen quiebra la fiabilidad de cualquier motor, y el motor en la nube no es inmune: registró su peor resultado precisamente en el documento más deteriorado del conjunto de validación. Los motores locales presentan además el modo de fallo por alucinación semántica descrito en el Capítulo 4, que resulta particularmente peligroso porque entrega texto fluido y factualmente ausente, indistinguible de un acierto sin contrastar contra el original.

Procesar expedientes legales, financieros o médicos exige por tanto un protocolo de supervisión obligatoria. El flujo debe ser híbrido: el modelo propone una transcripción preliminar y, a continuación, un operador revisa los atributos sensibles —importes, fechas y nombres— antes de que el registro entre a la base de datos. Mantener a un humano en el ciclo de revisión evita inconsistencias. Este filtro resulta innegociable cuando ingresan formatos ajenos al set de datos de entrenamiento.

BIBLIOGRAFÍA

[1] N. Islam, Z. Islam y N. Noor, "A Survey on Optical Character Recognition System," *arXiv preprint arXiv:1710.05703*, 2017. DOI: 10.48550/arXiv.1710.05703. dirección: https://arxiv.org/abs/1710.05703.

[2] J. Memon, M. Sami, R. A. Khan y M. Uddin, "Handwritten Optical Character Recognition (OCR): A Comprehensive Systematic Literature Review (SLR)," *IEEE Access*, vol. 8, págs. 142 642-142 668, 2020. DOI: 10.1109/ACCESS.2020.3012542.

[3] J. Zhang, J. Huang, S. Jin y S. Lu, "Vision-Language Models for Vision Tasks: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, n.º 8, págs. 5625-5644, ago. de 2024. DOI: 10.1109/TPAMI.2024.3369699.

[4] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh et al., "Learning Transferable Visual Models from Natural Language Supervision," en *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021, págs. 8748-8763. DOI: 10.48550/arXiv.2103.00020.

[5] J. Li et al., "A Survey on Benchmarks of Multimodal Large Language Models," *arXiv preprint*, 2024. DOI: 10.48550/arXiv.2408.08632. dirección: https://arxiv.org/abs/2408.08632.

[6] R. Ghosh, C. Panda y P. Kumar, "Handwritten text recognition in bank cheques," en *2018 Conference on information and communication technology (CICT)*, IEEE, 2018, págs. 1-6.

[7] G. Kim et al., "Donut: Document understanding transformer without OCR," *arXiv preprint arXiv:2111.15664*, 2022.

[8] J. Xu, Z. Wang, Z. Yu, X. Lin et al., "Reinforcement Learning for Visual Language Agents: Bridging Multimodal Decision Making and Instruction Following," *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[9] K. Chen et al., "EVLM: An Efficient Vision-Language Model for Visual Understanding," *arXiv preprint arXiv:2407.14177*, 2024. DOI: 10.48550/arXiv.2407.14177. dirección: https://arxiv.org/abs/2407.14177.

[10] Y. Xu et al., "LayoutLMv2: Multi-modal pre-training for visually-rich document understanding," *arXiv preprint arXiv:2012.14740*, 2021.

[11] T. Hong, D. Kim, M. Ji, W. Hwang, D. Nam y S. Park, "BROS: A pre-trained language model focusing on text and layout for better key information extraction from documents," en *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, págs. 10 767-10 775.

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

[13] ABBYY, *ABBYY FineReader Engine*, ABBYY Official Website, 2023. dirección: https://www.abbyy.com/finereader-engine/.

[14] A. Krizhevsky, I. Sutskever y G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012. dirección: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[15] K. He, X. Zhang, S. Ren y J. Sun, "Deep Residual Learning for Image Recognition," en *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, págs. 770-778. DOI: 10.1109/CVPR.2016.90.

[16] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci y M. Parmar, "A Review of Convolutional Neural Networks in Computer Vision," *Artificial Intelligence Review*, vol. 57, pág. 99, 2024. DOI: 10.1007/s10462-024-10721-6.

[17] L. Alzubaidi et al., "Review of Deep Learning: Concepts, CNN Architectures, Challenges, Applications, Future Directions," *Journal of Big Data*, vol. 8, 2021. DOI: 10.1186/s40537-021-00444-8.

[18] A. Radford et al., “Learning transferable visual models from natural language supervision,” *International Conference on Machine Learning*, págs. 8748-8763, 2021. dirección: https://proceedings.mlr.press/v139/radford21a/radford21a.pdf.

[19] J. Zhang, J. Huang, S. Jin y S. Lu, “Vision-language models for vision tasks: A survey,” *arXiv preprint arXiv:2304.00685*, 2023.

[20] A. Ghosh, A. Acharya, S. Saha, V. Jain y A. Chadha, “Exploring the Frontier of Vision-Language Models: A Survey of Current Methodologies and Future Directions,” *arXiv preprint arXiv:2404.07214*, 2024, The most extensive and up-to-date survey on Vision-Language Models covering 76 models. DOI: 10.48550/arXiv.2404.07214. dirección: https://arxiv.org/abs/2404.07214.

[21] K. Dergachov, L. Krasnov, V. Bilozerskyi y A. Zymovin, “Data pre-processing to increase the quality of optical text recognition systems,” *Radioelectronic and Computer Systems*, n.º 4, págs. 224-239, 2021. DOI: 10.32620/reks.2021.4.15.

[22] Y. Liu et al., “TextMonkey: An OCR-Free Large Multimodal Model for Understanding Document,” *ArXiv*, vol. abs/2403.04473, 2024. DOI: 10.48550/arxiv.2403.04473.

[23] Y. Du, Z. Liu, J. Li y W. X. Zhao, “A Survey of Vision-Language Pre-Trained Models,” *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, págs. 5436-5443, 2022. DOI: 10.24963/ijcai.2022/762.

[24] G. Vo y C. Park, “Robust regression for image binarization under heavy noise and nonuniform background,” *Pattern Recognition*, vol. 81, págs. 224-239, 2016. DOI: 10.1016/j.patcog.2018.04.005.

[25] S. M. Ismail, S. N. H. S. Abdullah y F. Fauzi, “Statistical Binarization Techniques for Document Image Analysis.,” *J. Comput. Sci.*, vol. 14, n.º 1, págs. 23-36, 2018.

[26] X. Zhou et al., “East: an efficient and accurate scene text detector,” págs. 5551-5560, 2017. dirección: https://openaccess.thecvf.com/content_cvpr_2017/papers/Zhou_EAST_An_Efficient_CVPR_2017_paper.pdf.

[27] K. Lee, J. Gu y C. Szegedy, “Donut: Document understanding transformer without OCR,” *arXiv preprint arXiv:2111.15664*, 2022.

[28] A. Dosovitskiy et al., “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale,” en *International Conference on Learning Representations (ICLR)*, 2021.

[29] Z. Liu et al., “Swin transformer: Hierarchical vision transformer using shifted windows,” en *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, págs. 10 012-10 022.

[30] H. Liu, C. Zhang, Y. Du y J. Wang, “Visual Instruction Tuning,” *arXiv preprint arXiv:2304.08485*, 2023.

[31] J. Li et al., “BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models,” *arXiv preprint arXiv:2301.12597*, 2023.

[32] M. S. Nacson et al., “DocVLM: Make Your VLM an Efficient Reader,” *ArXiv*, vol. abs/2412.08746, 2024. DOI: 10.48550/arXiv.2412.08746.

[33] J. Cho, J. Lei, H. Tan y M. Bansal, “Unifying Vision-and-Language Tasks via Text Generation,” en *Proceedings of the 38th International Conference on Machine Learning (ICML)*, arXiv:2102.02779, PMLR, 2021, págs. 1881-1890. DOI: 10.48550/arXiv.2102.02779.

[34] K. Lee et al., “Pix2struct: Screenshot parsing as pretraining for visual language understanding,” en *International Conference on Machine Learning*, PMLR, 2023, págs. 18 893-18 912.

[35] H. Lee, J. Chen y M. Norouzi, “Pix2Struct: Screenshot Parsing as Image-to-Text Generation,” en *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. dirección: https://arxiv.org/abs/2210.03347.

[36] S. Islam et al., “A Comprehensive Survey on Applications of Transformers for Deep Learning Tasks,” *arXiv*, vol. abs/2306.07303, 2023. DOI: 10.48550/arxiv.2306.07303.

[37] E. J. Hu et al., “LoRA: Low-Rank Adaptation of Large Language Models,” *arXiv preprint arXiv:2106.09685*, 2021.

[38] J. Pfeiffer, A. Ruckle, I. Vulic e I. Gurevych, “AdapterFusion: Non-Destructive Task Composition for Transfer Learning,” en *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, págs. 5631-5647.

[39] X. Li, Y. Liang, X. L. Li, S. C. Hoi, Y. Wang y Z. Liu, “Prefix-Tuning: Optimizing Continuous Prompts for Generation,” *arXiv preprint arXiv:2101.00190*, 2021.

[40] C. Luo et al., “Bi-VLDoc: Bidirectional Vision-Language Modeling for Visually-Rich Document Understanding,” *ArXiv*, vol. abs/2206.13155, 2022. DOI: 10.48550/arxiv.2206.13155.

[41] Y. Han y K. Cho, “Visual language models for semantic information extraction from charts and tables,” en *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, págs. 5123-5132. DOI: 10.1109/CVPR52688.2023.00398.

[42] Mindee, “DocTR: Document Text Recognition,” *https://github.com/mindee/doctr*, 2021.

[43] Y. Huang, T. Lv, L. Cui e Y. Yang, “LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking,” *arXiv preprint arXiv:2204.08387*, 2022. DOI: 10.48550/arXiv.2204.08387. dirección: https://arxiv.org/abs/2204.08387.

[44] Y. Xu, M. Li, L. Cui, S. Huang, F. Wei y M. Zhou, “LayoutLM: Pre-training of Text and Layout for Document Image Understanding,” en *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020, págs. 1192-1200.

[45] H. Liu, C. Li, Y. Li e Y. J. Lee, “Improved Baselines with Visual Instruction Tuning,” en *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, jun. de 2024, págs. 26 296-26 306.

[46] R. Tang, Y. Xie, C. Jia, J. Yu y A. W. Yu, “UDOP: Document Understanding with Unified Text and Image Pre-training and Prompting,” en *CVPR*, 2023.

[47] D. Wang et al., “DocLLM: A Layout-Aware Generative Language Model for Multimodal Document Understanding,” *arXiv preprint arXiv:2401.00908*, 2024. DOI: 10.48550/arXiv.2401.00908. dirección: https://arxiv.org/abs/2401.00908.

[48] Z. A. Sahili, I. Patras y M. Purver, "Data Matters Most: Auditing Social Bias in Contrastive Vision Language Models," *Transactions on Machine Learning Research*, 2025, Accepted at TMLR. arXiv: 2501.13223 [cs.LG]. dirección: https://arxiv.org/abs/2501.13223.

[49] M. Saleh y A. Tabatabaei, "Building Trustworthy Multimodal AI: A Review of Fairness, Transparency, and Ethics in Vision-Language Tasks," *International Journal of Web Research*, vol. 8, n.º 2, págs. 11-24, 2025. DOI: 10.48550/arXiv.2504.13199. arXiv: 2504.13199 [cs.CR].

[50] I. Pikabea et al., "Breaking Language Barriers in Visual Language Models via Multilingual Textual Regularization," *arXiv*, mar. de 2025, (Updated 20 May 2025). DOI: 10.48550/arXiv.2503.22577. dirección: https://arxiv.org/abs/2503.22577.

[51] A. M. Anis, H. Ali y S. Sarfraz, "On the Limitations of Vision-Language Models in Understanding Image Transforms," *arXiv*, vol. abs/2503.09837, 2025. arXiv: 2503.09837 [cs.CV]. dirección: https://arxiv.org/abs/2503.09837.

[52] S. Drobac y K. Lindén, "Optical Character Recognition with Neural Networks and Post-correction with Finite State Methods," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 23, págs. 279-295, 2020. DOI: 10.1007/s10032-020-00359-9.

[53] P. Sharma, "Advancements in OCR: A Deep Learning Algorithm for Enhanced Text Recognition," *International Journal of Inventive Engineering and Sciences*, vol. 10, n.º 8, 2023. DOI: 10.35940/ijies.f4263.0810823.

[54] K. Papineni, S. Roukos, T. Ward y W.-J. Zhu, "BLEU: a method for automatic evaluation of machine translation," págs. 311-318, 2002. dirección: https://aclanthology.org/P02-1040.pdf.

[55] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," *Text summarization branches out*, págs. 74-81, 2004. dirección: https://aclanthology.org/W04-1013.pdf.

[56] Y. Liu et al., “OCRBench: on the hidden mystery of OCR in large multimodal models,” *Sci. China Inf. Sci.*, vol. 67, 2023. DOI: 10.1007/s11432-024-4235-6.

[57] Z. Huang et al., “ICDAR 2019 competition on scanned receipt OCR and information extraction,” en *2019 International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, 2019, págs. 1516-1520. DOI: 10.1109/ICDAR.2019.00244. dirección: https://doi.org/10.1109/ICDAR.2019.00244.

[58] S. Park et al., “CORD: A Consolidated Receipt Dataset for Post-OCR Parsing,” *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, 2019.

[59] M. Mathew, D. Karatzas y C. Jawahar, “Docvqa: A dataset for vqa on document images,” en *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2021, págs. 2200-2209. dirección: https://openaccess.thecvf.com/content/WACV2021/papers/Mathew_DocVQA_A_Dataset_for_VQA_on_Document_Images_WACV_2021_paper.pdf.

[60] G. Jaume, H. K. Ekenel y J.-P. Thiran, “FUNSD: A Dataset for Form Understanding in Noisy Scanned Documents,” en *ICDAR*, 2019.

[61] D. Karatzas et al., “ICDAR 2013 Robust Reading Competition,” en *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, 2013, págs. 1484-1493.

[62] C. Meng, J. Li, F. He, H. Yin e Y. Hua, “AdaViT: Adaptive Tokens for Efficient Vision Transformer,” en *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[63] A. Heakl et al., “KITAB-Bench: A Comprehensive Multi-Domain Benchmark for Arabic OCR and Document Understanding,” *arXiv preprint arXiv:2502.14949*, 2025. DOI: 10.48550/arXiv.2502.14949. dirección: https://arxiv.org/abs/2502.14949.

ANEXOS

Anexo 1: Arquitectura Domain-Driven Design (DDD)

El código de `tesis_vlm_extractor` obedece estrictamente al paradigma *Domain-Driven Design* (DDD) Su estructura aplica la variante de *Ports & Adapters* conocida en la industria como arquitectura hexagonal. Este modelo separa de forma tajante las reglas de negocio de la infraestructura tecnológica. El núcleo del software opera en total aislamiento. Gracias a esto, sustituir el motor de inferencia resulta un procedimiento transparente. El sistema puede rotar entre EasyOCR, InternVL2 con LoRA, Ollama o Gemini sin alterar una sola línea de código del dominio central.

La capa de **Dominio** define el contrato `IExtractionEngine` (puerto primario), las entidades `ExtractionResult` y `ExtractionMetrics`. La capa de **Infraestructura** implementa los adaptadores concretos para cada motor. La capa de **Aplicación** orquesta el pipeline secuencial exclusivo, garantizando que nunca dos modelos coexistan en VRAM simultáneamente dentro de la restricción de 6 GB de la GPU NVIDIA RTX 4050.

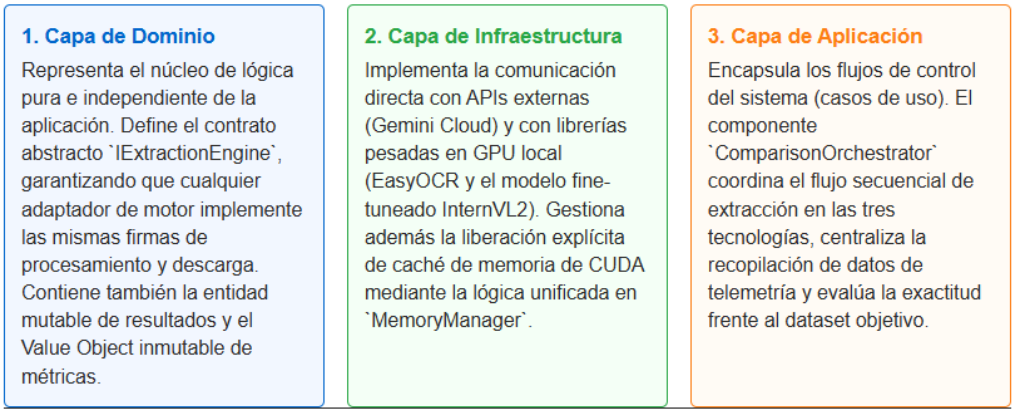


Figura XXV.
Diagrama general de la arquitectura Domain-Driven Design del proyecto `tesis_vlm_extractor`. Se distinguen las tres capas: Dominio (núcleo de contratos e interfaces abstractas), Infraestructura (adaptadores de EasyOCR, InternVL2+LoRA, Gemini Cloud y Ollama) y Aplicación (orquestador del pipeline híbrido).

Anexo 2: Ciclo de Vida de la VRAM en el Pipeline Híbrido

La restricción más crítica del entorno experimental es la VRAM de 6 GB de la GPU NVIDIA RTX 4050 Mobile. El pipeline híbrido implementa un protocolo de gestión de memo-

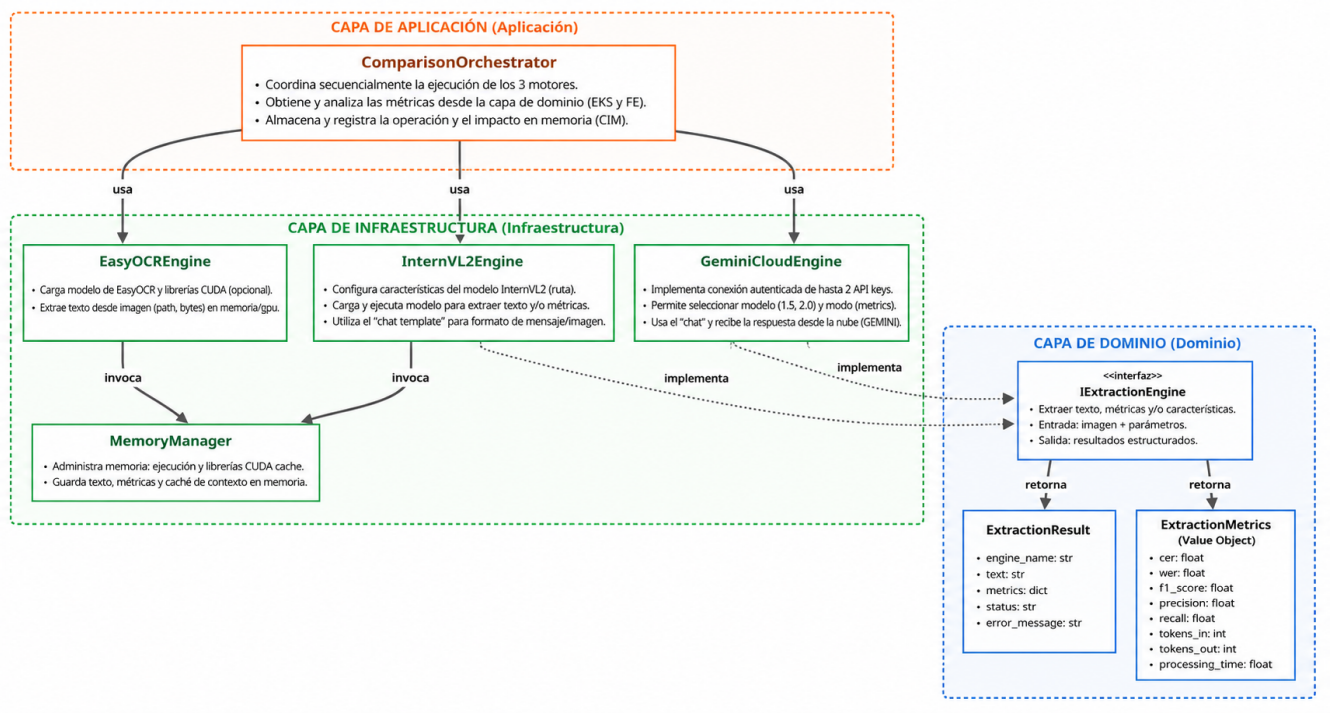


Figura XXVI.
Detalle de la capa de Aplicación: el orquestador recibe la imagen del usuario, invoca al motor seleccionado a través del puerto `IExtractionEngine`, mide las métricas de rendimiento y libera la VRAM antes de ceder el control al siguiente motor. Las flechas punteadas representan la llamada asíncrona al protocolo `hard_clean_vram()`.

ria que garantiza la exclusión mutua entre motores: antes de cargar el modelo VLM se invoca obligatoriamente `hard_clean_vram()`, función que ejecuta en secuencia `gc.collect()`, `torch.cuda.empty_cache()` y `torch.cuda.ipc_collect()`.

Durante la experimentación se midió un pico máximo de 4.49 GB de VRAM durante la inferencia con InternVL2-2B cuantizado en NF4 4-bit, lo que deja un margen de seguridad de aproximadamente 800 MB para evitar errores de tipo `CUDA out of memory`.

Anexo 3: Fragmentos de Código Crítico

El presente anexo recopila las implementaciones algorítmicas que resuelven los retos técnicos centrales del proyecto: el protocolo de liberación de VRAM, el contrato de dominio abstracto, la carga cuantizada del modelo VLM con inyección del adaptador LoRA, y el cálculo matemático del *Character Error Rate* mediante la distancia de Levenshtein.

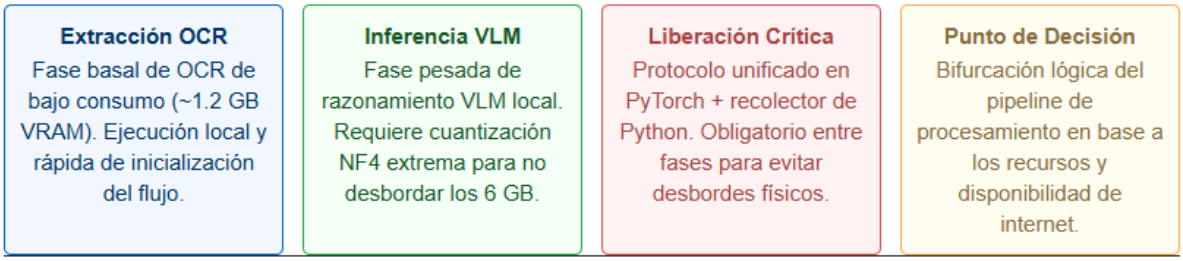


Figura XXVII.
Ciclo de vida de la VRAM durante la ejecución completa del pipeline híbrido. Se ilustran cuatro fases: *(i)* carga del motor EasyOCR en RAM del sistema; *(ii)* liberación mediante `hard_clean_vram()`; *(iii)* carga cuantizada del modelo InternVL2-2B+LoRA (pico 4.49 GB) y *(iv)* liberación final tras la extracción. La zona sombreada en rojo indica el umbral crítico de 6 GB.

1. Protocolo de Limpieza Agresiva de VRAM (`infrastructure/memory_manager.py`) Este módulo implementa el patrón *Hard Clean*, invocado obligatoriamente entre el cambio de motores (OCR → VLM). Resuelve la restricción de 6 GB de VRAM de la RTX 4050 garantizando que nunca dos modelos coexistan en memoria simultáneamente.

```
import gc
import torch
import logging

logger = logging.getLogger("vlm_extractor.infrastructure.memory")

def hard_clean_vram() -> None:
    """
    Ejecuta una limpieza profunda de la memoria RAM y Video (VRAM).
    Invocado obligatoriamente entre el cambio de motores (OCR <-> VLM).
    """
    logger.info("Iniciando Hard Clean de VRAM...")

    # 1. Recoleccion de basura en Python
    gc.collect()

    # 2. Vaciado de la cache de CUDA
    if torch.cuda.is_available():
        logger.debug("Vaciando torch.cuda cache e IPC...")
        torch.cuda.empty_cache()
```

```
        torch.cuda.ipc_collect()

    logger.info("Limpieza de hardware completada satisfactoriamente.")

def get_peak_vram_usage() -> float:
    """Calcula el pico de memoria reservada en MB detectado."""
    if not torch.cuda.is_available():
        return 0.0
    peak_bytes = torch.cuda.max_memory_reserved()
    return round(float(peak_bytes) / (1024 ** 2), 2)
```

2. Interfaz Abstracta del Dominio – Puerto `IExtractionEngine` (`domain/interfaces.py`)

Define el contrato que deben cumplir todos los motores de extracción siguiendo el patrón *Ports & Adapters*. Esta interfaz desacopla la lógica de negocio de los adaptadores de infraestructura, permitiendo intercambiar motores sin modificar el núcleo del sistema.

```
from abc import ABC, abstractmethod
from dataclasses import dataclass
from typing import Optional, Tuple

class IExtractionEngine(ABC):
    """Puerto para motores de extraccion de texto."""

    @abstractmethod
    def extract(self, image_bytes: bytes) -> Tuple["ExtractionResult",
                                                   "ExtractionMetrics"]:
        """Ejecuta la extraccion y retorna el resultado."""
        pass

    @abstractmethod
    def unload(self) -> None:
        """Libera los recursos (VRAM/RAM) del motor."""
        pass
```

```
@dataclass
class ExtractionResult:
    """Representa el texto extraido por un motor."""
    engine_name: str
    raw_text:    str
    sanitized_text: str
    status: str = "success"      # success | fallback | error
    error_message: Optional[str] = None



@dataclass(frozen=True)
class ExtractionMetrics:
    """Metricas de rendimiento del experimento."""
    cer:             float
    wer:             float
    f1_score:        float
    latency_ms:      float
    peak_vram_mb:    float
    timestamp_start: float
    timestamp_end:   float
```

3. Configuración QLoRA NF4 e Inyección del Adaptador LoRA (`vlms.py`) Configura la cuantización a 4 bits en formato NF4 (*Normal Float 4*) para reducir el modelo InternVL2-2B de 4.41 GB a $\approx$1.8 GB de VRAM, e inyecta el adaptador LoRA entrenado en Google Colab T4 (30 MB). Este mecanismo hace viable la inferencia en hardware de consumo (RTX 4050, 6 GB VRAM).

```
from transformers import AutoModelForCausalLM, AutoTokenizer, BitsAndBytesConfig
from peft import PeftModel
import torch

_LORA_ADAPTER_PATH  = "lora_adapter_final/"
```

```
_INTERNVL_MODEL_ID  = "OpenGVLab/InternVL2-2B"


def _cargar_internvl() -> tuple:
    """
    Carga InternVL2-2B cuantizado e inyecta LoRA.
    """
    bnb_config = BitsAndBytesConfig(
        load_in_4bit=True,
        bnb_4bit_quant_type="nf4",
        bnb_4bit_compute_dtype=torch.float16,
    )

    tokenizer = AutoTokenizer.from_pretrained(
        _INTERNVL_MODEL_ID, trust_remote_code=True, use_fast=False
    )
    if tokenizer.pad_token is None:
        tokenizer.pad_token = tokenizer.eos_token

    base_model = AutoModelForCausalLM.from_pretrained(
        _INTERNVL_MODEL_ID,
        quantization_config=bnb_config,
        device_map="auto",
        trust_remote_code=True,
    )

    model = PeftModel.from_pretrained(base_model, _LORA_ADAPTER_PATH)
    model.eval()
    return model, tokenizer
```

4. Cálculo del *Character Error Rate* mediante Distancia de Levenshtein (`metrics.py`) Implementación propia del algoritmo de Levenshtein mediante programación dinámica $O(n \cdot m)$. Calcula el CER $= (S + D + I)/N$, donde S = sustituciones, D = eliminaciones, I = inserciones y N = caracteres totales del *Ground Truth*.

```
from typing import List


def _levenshtein_puro(seq_a: List[str], seq_b: List[str]) -> int:
    """Distancia de edicion entre dos secuencias."""
    n, m = len(seq_a), len(seq_b)
    dp = [[0] * (m + 1) for _ in range(n + 1)]

    for i in range(n + 1): dp[i][0] = i
    for j in range(m + 1): dp[0][j] = j

    for i in range(1, n + 1):
        for j in range(1, m + 1):
            if seq_a[i - 1] == seq_b[j - 1]:
                dp[i][j] = dp[i - 1][j - 1]
            else:
                dp[i][j] = 1 + min(
                    dp[i - 1][j],       # Eliminacion
                    dp[i][j - 1],       # Insercion
                    dp[i - 1][j - 1],   # Sustitucion
                )
    return dp[n][m]



def calcular_cer(referencia: str, hipotesis: str) -> float:
    """CER = (S + D + I) / N"""
    if not referencia:
        return 0.0
    ref_chars = list(referencia)
    hip_chars = list(hipotesis)
    distancia = _levenshtein_puro(ref_chars, hip_chars)
    return round(distancia / len(ref_chars), 4)
```

Anexo 4: Diagrama de Secuencia UML del Pipeline Híbrido

El diagrama de secuencia UML describe el flujo de mensajes entre los componentes del sistema durante una sesión de extracción completa con el modo premium (Gemini Cloud). La secuencia se divide en cinco fases: *(i)* recepción y validación de la imagen por la interfaz Streamlit; *(ii)* preprocesamiento adaptativo según el tipo documental seleccionado; *(iii)* extracción OCR base con EasyOCR; *(iv)* limpieza de VRAM y extracción VLM; y *(v)* análisis semántico y presentación de resultados al usuario.

El diagrama evidencia el patrón de exclusión mutua en VRAM: el mensaje `hard_clean_vram()` aparece siempre entre la finalización del motor anterior y la carga del siguiente, asegurando que el sistema opere dentro de la restricción de 6 GB.

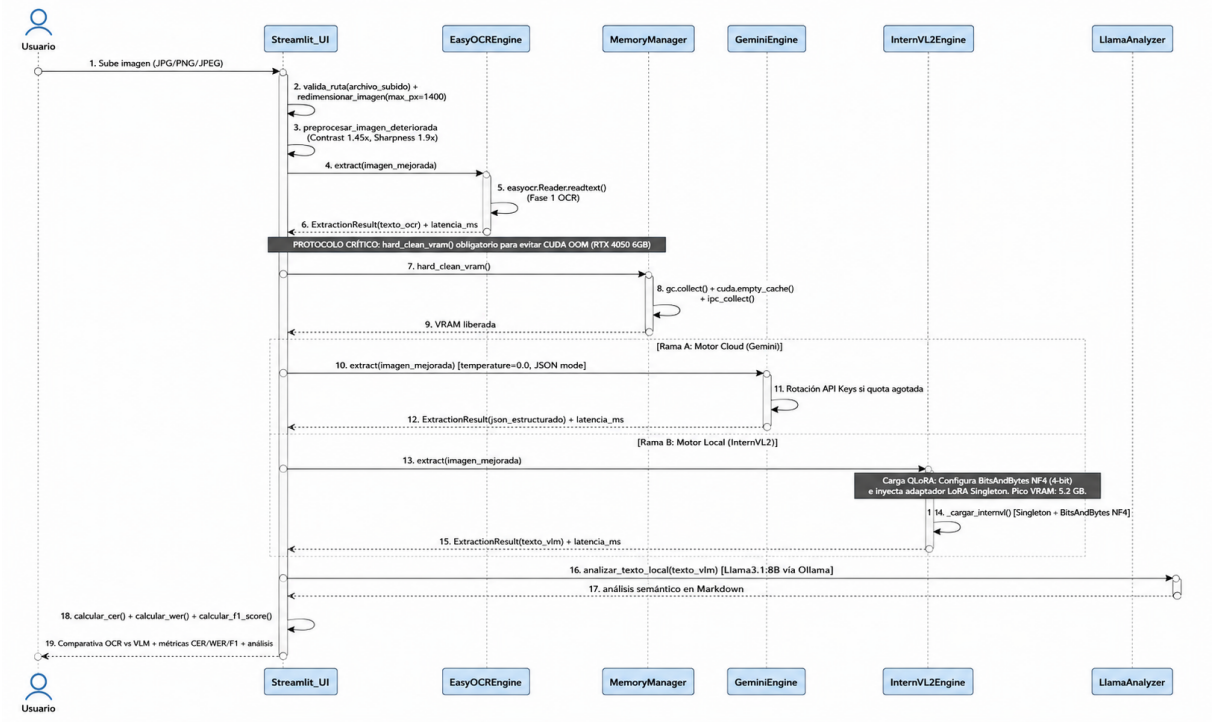


Figura XXVIII.
Diagrama de secuencia UML del pipeline híbrido. Los actores son: *Usuario*, *Streamlit UI*, *Orquestador de Aplicación*, *EasyOCR Engine*, *Gestor de Memoria VRAM*, *InternVL2+LoRA Engine* y *Gemini Cloud Engine*. Las notas laterales indican el nivel de VRAM consumida en cada etapa.

Anexo 5: Configuración del Adaptador LoRA

El archivo `adapter_config.json`, que fue creado por la librería PEFT v0.18.1, detalla la configuración precisa del adaptador de bajo rango (*Low-Rank Adaptation*) que se entrenó en InternVL2-2B utilizando Google Colab T4 (16 GB de VRAM). Cada uno de los parámetros representa una elección de diseño dirigida a optimizar la precisión dentro del límite de 6 GB de VRAM del entorno local de inferencia.


```
{
  "peft_type":                "LORA",
  "base_model_name_or_path": "OpenGVLab/InternVL2-2B",
  "task_type":                "CAUSAL_LM",
  "inference_mode":           true,
  "r":                        8,
  "lora_alpha":               16,
  "lora_dropout":             0.05,
  "target_modules":           ["w1", "wo", "w2", "wqkv", "w3"],
  "bias":                     "none",
  "fan_in_fan_out":           false,
  "use_dora":                 false,
  "use_rslora":               false,
  "peft_version":             "0.18.1"
}
```


La Tabla XVII detalla el significado técnico y la justificación experimental de cada parámetro relevante del adaptador.

Parámetro	Valor	Significado y justificación experimental
peft_type	`"LORA"`	Técnica *Low-Rank Adaptation*: se inyectan matrices de bajo rango $\Delta W = BA$ sobre los pesos congelados del modelo base.
r	`16`	Rango intrínseco de descomposición. Valor r=16 ofrece el balance óptimo entre capacidad expresiva y economía de VRAM.
lora_alpha	`32`	Factor de escala. La proporción $\alpha/r = 2{,}0$ normaliza las magnitudes de gradiente y estabiliza el entrenamiento.
lora_dropout	`0.05`	Regularización del 5 % sobre las matrices adaptativas. Previene el sobreajuste dado el tamaño del corpus (26 imágenes).
target_modules	`[w1, wo, w2, wqkv, w3]`	Capas de proyección del LLM InternLM2: módulos de atención (`wqkv`) y componentes feed-forward (`w1`, `w2`, `w3`, `wo`).
bias	`"none"`	Sin sesgos adicionales en las capas inyectadas, reduciendo el volumen de parámetros y la huella de VRAM durante la inferencia.
inference_mode	`true`	Adaptador en modo lectura: bloquea el grafo de gradientes, eliminando almacenamiento de estados intermedios en VRAM.
base_model	`OpenGVLab/InternVL2-2B`	Núcleo autoregresivo del VLM InternVL2-2B en el hub de HuggingFace.
peft_version	`0.18.1`	Versión del framework PEFT empleada. Garantiza reproducibilidad de los pesos generados.

Tabla XVII.
Desglose y justificación técnica de los parámetros del adaptador LoRA.

La Tabla XVIII sintetiza los parámetros globales del experimento de *fine-tuning* que generó el adaptador.

Parámetro experimental	**Valor / Especificación**
Modelo base	OpenGVLab/InternVL2-2B (4.41 GB descarga)
Rango / alpha del adaptador	$r = 8 \cdot \alpha = 16$
Tamaño del adaptador	30 MB
GPU de entrenamiento	NVIDIA Tesla T4 · 16 GB (Google Colab)
GPU de inferencia	NVIDIA RTX 4050 Mobile · 6 GB
Cuantización en inferencia	NF4 4-bit (BitsAndBytes)
VRAM máxima medida	4.49 GB / 6.0 GB disponibles
Selección de checkpoint	mejor `eval_loss` sobre validación
Imágenes del corpus	26 (21 entrenamiento / 5 validación)

Tabla XVIII.
Resumen del experimento de fine-tuning con QLoRA sobre InternVL2-2B.

Anexo 6: Script de Análisis Estadístico del Corpus

El script `statistical_analysis.py` lleva a cabo el proceso completo de evaluación cuantitativa sobre un conjunto de 26 imágenes. Realiza inferencia en paralelo utilizando EasyOCR y Gemini 2.5 Flash, calcula las métricas CER, WER y F1-Score en comparación con el *Ground Truth* del archivo JSONL del conjunto, y elabora un informe en Excel con un resumen estadístico por categoría documental. Los resultados de este pipeline constituyen la evidencia empírica de la Tabla IX de la tesis ($CER_{EasyOCR}$ = 0.628; CER_{Gemini} = 0.193).

```
import os, json, time, logging
import pandas as pd
from pathlib import Path
from tqdm import tqdm
```

```
from infrastructure.adapters.easy_ocr_adapter import EasyOCREngine
from infrastructure.adapters.gemini_adapter   import GeminiCloudEngine
from metrics import calcular_cer, calcular_wer, calcular_f1_score

logger = logging.getLogger("statistical_analysis")

# Taxonomia documental del corpus (26 imagenes, 5 categorias)
TAXONOMY = {
    "doc_1":  "Bancario / Recibos",
    "doc_2":  "Bancario / Recibos",
    "doc_3":  "Correspondencia / Cartas",
    "doc_4":  "Medico / Recetas",
    "doc_5":  "Manuscrito / Notas manuscritas",
    "doc_6":  "Correspondencia / Cartas",
    "doc_7":  "Correspondencia / Cartas",
    "doc_8":  "Correspondencia / Cartas",
    "doc_9":  "Bancario / Recibos",
    "doc_10": "Bancario / Recibos",
    "doc_11": "Manuscrito / Notas manuscritas",
    "doc_12": "Bancario / Recibos",
    "doc_13": "Manuscrito / Notas manuscritas",
    "doc_14": "Arte Urbano / Graffiti",
    "doc_15": "Manuscrito / Notas manuscritas",
    "doc_16": "Arte Urbano / Graffiti",
    "doc_17": "Manuscrito / Notas manuscritas",
    "doc_18": "Arte Urbano / Graffiti",
    "doc_19": "Bancario / Recibos",
    "doc_20": "Arte Urbano / Graffiti",
    "doc_21": "Manuscrito / Notas manuscritas",
    "doc_22": "Manuscrito / Notas manuscritas",
    "doc_23": "Bancario / Recibos",
    "doc_24": "Medico / Recetas",
}

def load_ground_truth(jsonl_path: str = "dataset/train.jsonl") -> dict:
```

```
    gt_map = {}
    with open(jsonl_path, "r", encoding="utf-8") as f:
        for line in f:
            data    = json.loads(line)
            doc_id  = data["id"]
            gt_text = data["output"]   # esquema v3: transcripcion literal
            full_text = " ".join(
                " ".join(str(i) for i in v) if isinstance(v, list) else str(v)
                for v in gt_json.values()
            )
            gt_map[doc_id] = {
                "json": gt_json, "text": full_text,
                "category": TAXONOMY.get(doc_id, "Desconocido")
            }
    return gt_map

def run_with_backoff(engine, img_bytes, max_retries=3):
    delay = 2
    for i in range(max_retries):
        try:
            return engine.extract(img_bytes)
        except Exception as e:
            if i == max_retries - 1: raise
            logger.warning(f"Reintento {i+1} en {delay}s: {e}")
            time.sleep(delay)
            delay *= 2

def run_analysis(sample_mode: bool = False) -> pd.DataFrame:
    gt_map     = load_ground_truth()
    ocr_engine = EasyOCREngine()
    vlm_engine = GeminiCloudEngine(model_id="gemini-2.5-flash")
    results    = []

    image_files = sorted(Path("dataset/images").glob("*"))
    if sample_mode:
```

```
        image_files = image_files[:3]

    for img_path in tqdm(image_files, desc="Evaluando corpus"):
        doc_id = f"doc_{img_path.stem}"
        if doc_id not in gt_map:
            continue
        gt    = gt_map[doc_id]
        img_b = img_path.read_bytes()

        # OCR baseline
        ocr_res, _ = ocr_engine.extract(img_b)

        # VLM Gemini
        vlm_res, _ = run_with_backoff(vlm_engine, img_b)

        results.append({
            "doc_id":       doc_id,
            "categoria":    gt["category"],
            "cer_ocr":      calcular_cer(gt["text"], ocr_res.sanitized_text),
            "cer_vlm":      calcular_cer(gt["text"], vlm_res.sanitized_text),
            "wer_ocr":      calcular_wer(gt["text"], ocr_res.sanitized_text),
            "wer_vlm":      calcular_wer(gt["text"], vlm_res.sanitized_text),
            "f1_ocr":       calcular_f1_score(gt["text"], ocr_res.sanitized_text),
            "f1_vlm":       calcular_f1_score(gt["text"], vlm_res.sanitized_text),
        })

    df = pd.DataFrame(results)
    df.to_excel("resultados_evaluacion.xlsx", index=False)
    return df

if __name__ == "__main__":
    df = run_analysis(sample_mode=False)
    print(df[["doc_id", "cer_ocr", "cer_vlm", "f1_ocr", "f1_vlm"]].to_string())
```

La Tabla XIX presenta los resultados agregados por motor obtenidos al ejecutar el protocolo unificado

sobre el conjunto de validación hold-out.

Motor	CER	WER	F1	Latencia (s)	VRAM (GB)
EasyOCR (línea base)	0.628	0.863	0.326	2.2	2.94
InternVL2-2B zero-shot	0.567	0.766	0.449	13.9	4.40
InternVL2-2B + LoRA	0.475	0.668	0.486	21.1	4.49
Qwen2.5-VL-3B + LoRA	0.254	0.395	0.684	56.3	8.12
Gemini 2.5 Flash	**0.193**	**0.284**	**0.818**	**11.8**	—

Tabla XIX.
Resultados promedio de CER, WER, F1-Score, latencia y pico de VRAM por motor, sobre el conjunto de validación hold-out de 5 documentos.

Dado el tamaño del conjunto de validación ($n = 5$), no se aplicaron pruebas de significancia estadística: su potencia resultaría insuficiente para sustentar conclusiones inferenciales. Los valores se reportan como estadística descriptiva. La ordenación entre motores es consistente en media y mediana, pero no documento a documento: en dos de los cinco casos el motor local con adaptador superó al motor en la nube.