Texto

Descripción generada automáticamente

**TÍTULO**

**UNIVERSIDAD NEBRIJA GRADO EN**

**INGENIERÍA INFORMÁTICA**

**MEMORIA TRABAJO FIN DE GRADO**

**Guillermo Martínez de Hurtado Aricha**

**07/2025**

Texto

Descripción generada automáticamente

**TITULO**

**UNIVERSIDAD NEBRIJA GRADO EN INGENIERÍA INFORMÁTICA**

**MEMORIA TRABAJO FIN DE GRADO**

**Guillermo Martínez de Hurtado Aricha**

**07/2025**

Christian Velsaco Gallego

Dn/Dña. Guillermo Martínez de Hurtado Aricha autoriza a que el presente trabajo se guarde y custodie en los repositorios de la Universidad Nebrija y además NO autoriza a su disposición en abierto.

*Dedicatoria (opcional).*

# Índice

[Índice 5](#_Toc204192470)

[Glosario de términos 7](#_Toc204192471)

[Índice de ilustraciones 8](#_Toc204192472)

[Índice de tablas 9](#_Toc204192474)

[Resumen 10](#_Toc204192475)

[Abstract 11](#_Toc204192476)

[1. Introducción 12](#_Toc204192477)

[1.1. Motivación 13](#_Toc204192478)

[1.2. Antecedentes 14](#_Toc204192479)

[1.2.1. Estado del arte 14](#_Toc204192480)

[1.2.2. Necesidad detectada 16](#_Toc204192481)

[1.3. Objetivos 17](#_Toc204192482)

[1.3.1. Objetivos específicos 17](#_Toc204192483)

[1.4. Requisitos técnicos 18](#_Toc204192484)

[1.4.1. Requisitos de software: 18](#_Toc204192485)

[1.4.2. Requisitos de hardware: 18](#_Toc204192486)

[1.4.3. Recursos de datos: 19](#_Toc204192487)

[1.4.4. Dependencias externas: 19](#_Toc204192488)

[2. Marco teórico 19](#_Toc204192489)

[2.1. Reconocimiento óptico de caracteres (OCR) 19](#_Toc204192490)

[2.2. Aprendizaje automático no supervisado 20](#_Toc204192491)

[2.3. Autoencoders 20](#_Toc204192492)

[2.4. Clustering con K-Means 20](#_Toc204192493)

[2.5. Clasificación supervisada 20](#_Toc204192494)

[3. Metodología 21](#_Toc204192495)

[3.1. Preprocesamiento de imágenes 21](#_Toc204192496)

[3.1.1. Escalado a escala de grises 21](#_Toc204192497)

[3.1.2. Umbralización y segmentación 22](#_Toc204192498)

[3.1.3. Redimensionado y normalización 22](#_Toc204192499)

[3.2. Entrenamiento no supervisado (Autoencoder) 22](#_Toc204192500)

[3.2.1. Codificador (encoder) 23](#_Toc204192501)

[3.2.2. Decodificador (decoder) 23](#_Toc204192502)

[3.3. Asignación manual de clústeres 23](#_Toc204192503)

[3.4. Entrenamiento supervisado (K-means) 24](#_Toc204192504)

[3.4.1. Vectorización del espacio latente 24](#_Toc204192505)

[3.4.2. Agrupamiento con K-Means 24](#_Toc204192506)

[3.4.3. Etiquetado para clasificación 24](#_Toc204192507)

[3.5. Reconstrucción del texto 25](#_Toc204192508)

[4. Proyecto 25](#_Toc204192509)

[4.1. Resumen de contribuciones y productos desarrollados 25](#_Toc204192510)

[4.1.1. Contribuciones principales 26](#_Toc204192511)

[4.1.2. Productos desarrollados 26](#_Toc204192512)

[4.2. Planificación temporal 29](#_Toc204192513)

[4.3. Recursos empleados 31](#_Toc204192514)

[4.3.1. Gestión y manipulación de datos 31](#_Toc204192515)

[4.3.2. Procesamiento y segmentación de imágenes 31](#_Toc204192516)

[4.3.3. Visualización 32](#_Toc204192517)

[4.3.4. Aprendizaje automático y modelos 32](#_Toc204192518)

[4.3.5. Control de flujo y utilidades 33](#_Toc204192519)

[4.3.6. Otras herramientas 33](#_Toc204192520)

[4.4. Trabajo desarrollado 33](#_Toc204192521)

[4.4.1. Cuaderno 1: Reconstrucción de frases con espacios 34](#_Toc204192522)

[4.4.2. Cuaderno 2: Textos antiguos 38](#_Toc204192523)

[5. Resultados y discusión 42](#_Toc204192524)

[5.1. Codificación latente y agrupamiento 43](#_Toc204192525)

[5.2. Etiquetado y clasificación 43](#_Toc204192526)

[5.3. Reconstrucción del texto 43](#_Toc204192527)

[5.4. Discusión y valoración general 44](#_Toc204192528)

[6. Conclusiones 44](#_Toc204192529)

[6.1. Valoración personal 45](#_Toc204192530)

[6.2. Líneas futuras 46](#_Toc204192531)

[Bibliografía 47](#_Toc204192532)

[ANEXOS 48](#_Toc204192533)

# Glosario de términos

→ **Autoencoder**: Red neuronal no supervisada diseñada para aprender una representación comprimida (codificación) de los datos de entrada y reconstruirlos desde esa codificación. Se compone de dos partes principales: codificador y decodificador.

→ **Codificador (Encoder)**: Parte del autoencoder encargada de transformar la entrada original en una representación de menor dimensión o latente.

→ **Decodificador (Decoder)**: Componente que toma la representación latente generada por el codificador y trata de reconstruir la entrada original.

→ **Embedding**: Técnica para representar caracteres (o palabras) como vectores en un espacio numérico continuo, donde la distancia entre ellos tiene significado semántico o estructural. En este trabajo, se aplica a nivel de caracteres.

→ **Reconstrucción**: Salida generada por el autoencoder a partir del vector comprimido. Se evalúa comparando la similitud entre la entrada original y la salida reconstruida.

→ **Red neuronal densa (Fully Connected Network)**: Tipo de red donde cada neurona de una capa está conectada con todas las neuronas de la capa siguiente. Utilizada en este proyecto para construir los autoencoders.

→ **Función de pérdida (Loss Function)**: Métrica que cuantifica la diferencia entre la salida reconstruida y la entrada original. En este caso, típicamente la pérdida por error cuadrático medio (*Mean Squared Error*).

→ **Época (Epoch)**: Una iteración completa sobre todo el conjunto de entrenamiento durante el proceso de aprendizaje del modelo.

→ **Sobreajuste (Overfitting)**: Situación en la que un modelo aprende demasiado bien los datos de entrenamiento, perdiendo capacidad de generalización a datos nuevos.

→ **Distancia de Levenshtein**: Métrica para medir la diferencia entre dos secuencias de caracteres, contabilizando inserciones, eliminaciones y sustituciones necesarias para transformar una en otra. Útil para evaluar la reconstrucción de texto.

→ **Tokenización carácter a carácter**: Proceso de convertir un texto en una secuencia de caracteres individuales, permitiendo un enfoque de aprendizaje más fino, pero más sensible al ruido.

→ **Generación de texto (Text generation)**: Tarea en la que un modelo predice o completa texto de manera autónoma. En este trabajo se propone como etapa posterior a la reconstrucción del autoencoder usando modelos como GPT.

# Índice de ilustraciones

# Índice de tablas

**No table of figures entries found.**

# Resumen

La digitalización de archivos históricos, como los del Archivo General de Indias, ha avanzado significativamente, pero sigue limitada por la falta de transcripciones textuales automatizadas. Los sistemas OCR tradicionales fallan ante caligrafías antiguas debido a su variabilidad, ruido y ausencia de datos etiquetados. Este trabajo propone un *pipeline* híbrido que combina *autoencoders* entrenados carácter a carácter con técnicas de *clustering* y etiquetado asistido, para recuperar y reconstruir textos manuscritos sin depender de datos anotados. Se han desarrollado dos casos de estudio diferenciados: uno basado en frases modernas estructuradas y otro que simula condiciones propias de documentos antiguos sin segmentación. En ambos escenarios se ha logrado reconstruir texto a partir de secuencias de caracteres procesadas mediante aprendizaje no supervisado, obteniendo resultados parciales que reflejan la capacidad del modelo para identificar patrones caligráficos básicos y estructurar fragmentos de texto, lo que supone un paso inicial funcional hacia la automatización de la transcripción de documentos históricos.

# Abstract

The digitization of historical archives, such as those of the General Archive of the Indies, has made noteworthy progress but remains limited by the lack of automated textual transcriptions. Traditional OCR systems fail when faced with ancient handwriting due to its variability, noise, and the absence of labeled data. This work proposes a hybrid pipeline that combines character-level autoencoders with clustering techniques and assisted labeling to recover and reconstruct handwritten texts without relying on annotated datasets. Two distinct case studies were developed: one based on structured modern sentences, and another simulating the conditions of historical documents with no segmentation. In both scenarios, it was possible to reconstruct text from character sequences processed through unsupervised learning, obtaining partial results that reflect the model’s ability to identify basic handwriting patterns and structure text fragments, representing a functional first step towards the automation of historical document transcription.

# Introducción

La digitalización del patrimonio documental se ha convertido en una prioridad en el ámbito archivístico y cultural, impulsada por la necesidad de preservar y democratizar el acceso a fuentes históricas de gran valor. En España, uno de los proyectos más ambiciosos en esta línea ha sido la digitalización del Archivo General de Indias, cuya colección incluye miles de documentos manuscritos relacionados con la administración del imperio español en América y Asia. Estos documentos, fundamentales para la investigación histórica y social, se encuentran disponibles en formato imagen a través del Portal de Archivos Españoles (PARES) del Ministerio de Cultura y Deporte ([pares.culturaydeporte.gob.es](https://pares.culturaydeporte.gob.es)).

No obstante, el acceso real al contenido de estos documentos sigue siendo limitado, ya que la mayoría carece de una transcripción textual estructurada. Frente a esta realidad, el Reconocimiento Óptico de Caracteres (OCR) aparece como una solución tecnológica imprescindible. Sin embargo, los sistemas OCR convencionales, diseñados para textos impresos y tipografías homogéneas, muestran un rendimiento deficiente cuando se enfrentan a documentos manuscritos antiguos. Factores como la caligrafía variable, el deterioro del soporte físico, la tinta desvanecida o la ausencia de puntuación estándar suponen obstáculos importantes. A ello se suma la escasez de conjuntos de datos anotados, lo que dificulta el uso de enfoques supervisados.

Este trabajo se enmarca en esa problemática, proponiendo un sistema basado en técnicas de aprendizaje no supervisado para el tratamiento y reconstrucción de texto a nivel de carácter. Concretamente, se explora el uso de autoencoders convolucionales, entrenados carácter a carácter, capaces de aprender representaciones latentes compactas de glifos manuscritos y generar versiones reconstruidas de las entradas. La arquitectura propuesta permite recuperar frases parcialmente degradadas, ya sea por pérdida de segmentación, eliminación de espacios o errores de transcripción.

Se han desarrollado dos variantes de modelos:

1. Una centrada en frases modernas con espacios y puntuación, como representación de textos contemporáneos correctamente estructurados.
2. Otra que simula la entrada de textos antiguos sin espacios, para recrear el efecto de documentos históricos que llegan digitalizados sin segmentación clara.

Ambos enfoques buscan evaluar la capacidad del modelo para identificar estructuras internas del lenguaje desde un nivel bajo, sin información semántica previa. En el caso del primer escenario, basado en frases modernas con puntuación y espacios, cuando la reconstrucción realizada por el autoencoder no alcanza a formar una frase completa o sintácticamente coherente, se emplea un modelo de lenguaje generativo, como GPT, para sugerir una finalización adecuada a partir del fragmento generado.

Este enfoque mixto, que parte de una codificación sin supervisión y puede derivar hacia generación contextual, representa un avance técnico con posibles aplicaciones en la transcripción automática de archivos históricos. Además, contribuye al desarrollo de soluciones OCR más robustas, modulares y escalables, capaces de adaptarse a la diversidad de documentos del patrimonio cultural.

## Motivación

La idea de preservar el pasado mediante herramientas tecnológicas ha sido una constante motivación personal a lo largo de mi formación académica. El patrimonio documental no solo representa una fuente inestimable de conocimiento histórico, sino también un legado cultural que debe ser protegido, comprendido y transmitido a futuras generaciones. Desde esta perspectiva, la posibilidad de aplicar conocimientos técnicos en inteligencia artificial al servicio de la conservación y difusión del patrimonio me resultó especialmente atractiva.

En particular, el Archivo General de Indias constituye uno de los testimonios documentales más importantes del periodo colonial español, tanto por su volumen como por la riqueza de la información que contiene. A través de cartas, informes, censos, registros de comercio y documentos administrativos, se puede reconstruir el funcionamiento institucional, económico y social de un vasto territorio bajo control imperial. El acceso real a este conocimiento sigue limitado porque la mayoría de los textos conservados están en formato imagen. Aunque han sido digitalizados por el Ministerio de Cultura y están disponibles en el portal PARES, su utilidad práctica es restringida si no se dispone de una transcripción accesible y automatizable.

Es precisamente esta situación la que despertó mi interés por desarrollar un sistema de Reconocimiento Óptico de Caracteres (OCR) capaz de operar sobre este tipo de documentos. No es solo un ejercicio técnico, sino un intento por conectar el conocimiento computacional con la comprensión del pasado, de hacer que las tecnologías de aprendizaje automático tengan un impacto tangible en las humanidades. La idea de que una herramienta desarrollada en este trabajo pueda contribuir, aunque sea parcialmente, a facilitar la lectura, transcripción o indexación de documentos del Archivo General de Indias supone una motivación profunda y persistente.

Desde el punto de vista técnico, me resulta especialmente estimulante enfrentarme a un problema abierto, en el que los métodos estándar no ofrecen resultados satisfactorios. La naturaleza irregular y compleja de los manuscritos históricos plantea retos que no se resuelven simplemente con el uso de modelos preentrenados, lo que exige desarrollar soluciones adaptadas y creativas. La idea de construir un OCR modular, basado en codificación no supervisada y en etiquetado humano mínimo, responde a una necesidad técnica real y a una inquietud personal: la de generar conocimiento útil que permita abrir las puertas del pasado con las herramientas del presente.

Este proyecto no se entiende como un desarrollo de software, sino como una convergencia entre dos mundos: la tecnología y la historia. La posibilidad de facilitar la digitalización de estos documentos no es solo una cuestión de eficiencia, sino una forma de acercar el pasado a quienes, como yo, creemos que su estudio es esencial para entender el presente y construir el futuro.

## Antecedentes

El avance de las tecnologías digitales ha facilitado en las últimas décadas la preservación y difusión del patrimonio documental mediante grandes iniciativas de digitalización. Archivos históricos de diversa naturaleza han comenzado a ser parte de plataformas accesibles al público, permitiendo a investigadores, docentes y ciudadanos consultar fuentes limitadas por barreras físicas, geográficas o de conservación.

En las últimas décadas, numerosos archivos históricos han emprendido procesos de digitalización masiva con el objetivo de preservar y democratizar el acceso a su patrimonio documental. Esta transformación ha permitido que millones de páginas manuscritas estén disponibles en formato digital, facilitando su consulta remota y su conservación a largo plazo. Pero esta accesibilidad no se ha traducido en una mejora efectiva de su aprovechamiento académico o computacional.

La mayoría de estos documentos históricos están digitalizados exclusivamente en formato imagen, sin contar con transcripciones asociadas, metadatos estructurados ni mecanismos de extracción automática del contenido textual. Este hecho limita enormemente su explotación por medios informáticos, ya que imposibilita búsquedas semánticas, análisis lingüísticos, tratamiento estadístico o traducción automatizada.

El principal obstáculo radica en la naturaleza del texto manuscrito. A diferencia del texto impreso, los manuscritos presentan una alta variabilidad caligráfica, estilos gráficos personalizados, uso de formas arcaicas del lenguaje y, con frecuencia, degradación física del soporte. Estos factores dificultan considerablemente su tratamiento mediante técnicas convencionales de Reconocimiento Óptico de Caracteres, las cuales han sido diseñadas y optimizadas para fuentes tipográficas modernas, regulares y alineadas.

Pese a los avances recientes en modelos de reconocimiento de texto basados en aprendizaje profundo, la aplicación efectiva del OCR a documentos históricos sigue enfrentándose a limitaciones importantes. Entre las principales barreras se encuentran la escasez de corpus manuscritos anotados para entrenamiento supervisado, la falta de modelos generalizables a distintos estilos caligráficos, y la necesidad de adaptación específica por colección, lo que encarece y ralentiza su despliegue.

Ante esto, hay que explorar enfoques alternativos más flexibles, que prescindan del etiquetado exhaustivo y puedan adaptarse progresivamente a las características visuales de cada fondo documental. Este proyecto se inscribe en dicha línea de trabajo, proponiendo un sistema OCR modular y no supervisado, diseñado específicamente para abordar los retos que plantean los documentos manuscritos históricos en entornos patrimoniales digitalizados.

### Estado del arte

La digitalización y reconocimiento automático de textos manuscritos ha sido durante décadas un reto tecnológico de primer orden. A diferencia del reconocimiento óptico de caracteres (OCR) aplicado a texto impreso, donde los modelos comerciales han alcanzado altos niveles de precisión, los manuscritos presentan una amplia variabilidad caligráfica, deformaciones, ruido visual, y escasez de datos etiquetados. En este contexto, tanto la comunidad científica como el sector tecnológico han propuesto soluciones diversas, desde arquitecturas entrenadas en dominios específicos hasta modelos no supervisados que permiten un reconocimiento más flexible. Este apartado presenta una revisión crítica de las principales aportaciones académicas y tecnológicas recientes, agrupadas por líneas temáticas.

**Digitalización de textos antiguos**

Diversos trabajos han abordado el problema de transcribir documentos históricos a partir de imágenes escaneadas. En el contexto español, iniciativas como el **Portal de Archivos Españoles (PARES)** han promovido la digitalización masiva de archivos como el **Archivo General de Indias**, aunque en muchos casos, sin acompañamiento de herramientas de transcripción automatizada. **Romera (2016)** ya señalaba la falta de soluciones OCR adaptadas a caligrafías antiguas dentro del marco institucional español.

A nivel internacional, plataformas como **Transkribus** (Muehlberger et al., 2019) han revolucionado el campo mediante modelos entrenados sobre colecciones específicas de manuscritos europeos, combinando segmentación automática, reconocimiento de texto y edición colaborativa. Otros estudios como los de **Springmann y Lüdeling (2016)** o **Kahle et al. (2017)** han propuesto modelos que combinan redes convolucionales con componentes recurrentes para trabajar con textos medievales o escritura gótica. A pesar de sus logros, estos sistemas suelen requerir grandes volúmenes de datos etiquetados y entrenamiento específico por colección, lo que limita su escalabilidad a nuevos contextos.

**Avances recientes en OCR mediante aprendizaje profundo**

La evolución de los sistemas OCR se ha visto muy marcada por la aparición de arquitecturas basadas en redes neuronales profundas. El modelo CTC (Connectionist Temporal Classification) propuesto por *Graves et al.* (2009) permitió por primera vez entrenar sistemas de reconocimiento sin segmentación explícita carácter a carácter, lo que supuso un avance significativo frente a los enfoques tradicionales. No obstante, este modelo presenta ciertas limitaciones: su rendimiento depende en gran medida de la calidad del alineamiento implícito entre la entrada visual y la secuencia textual, y requiere conjuntos de entrenamiento bien estructurados. Además, su capacidad para manejar textos con gran variabilidad caligráfica o con estructuras altamente deformadas es limitada, ya que el modelo no dispone de mecanismos internos para adaptarse a cambios visuales drásticos o ruido no estructurado.

Esta innovación fue ampliada por modelos como CRNN (Convolutional Recurrent Neural Network) (*Shi et al.*, 2016), que combinan extracción convolucional con predicción secuencial mediante redes recurrentes. Aunque CRNN mejora la precisión y estabilidad del reconocimiento en textos impresos y lineales, también presenta restricciones importantes. Por un lado, su arquitectura requiere que las secuencias de entrada mantengan una coherencia visual en la dirección de lectura, lo que dificulta su uso con manuscritos no alineados o con escritura curva. Por otro, su dependencia de datos etiquetados sigue siendo elevada, y la generalización a escrituras históricas o caligrafías no normalizadas resulta poco fiable sin reentrenamiento específico.

En trabajos posteriores, *Baek et al.* (2019) y *Wigington et al.* (2018) exploraron arquitecturas más complejas con módulos de atención, deformación geométrica y decodificación lingüística, obteniendo excelentes resultados sobre texto impreso, curvo o degradado. Sin embargo, su rendimiento sobre manuscritos sin anotación previa sigue siendo limitado.

Estudios como los de *Breuel* (2017) y *Clanuwat et al.* (2018) han demostrado que incluso los modelos más avanzados pierden precisión si no están específicamente adaptados al dominio visual del manuscrito, y que los resultados dependen en gran medida de la disponibilidad de corpus etiquetados de forma fiable.

**Autoencoders y codificación latente no supervisada**

Ante la dificultad de contar con grandes volúmenes de anotaciones, ha ganado interés el uso de técnicas no supervisadas, como los **autoencoders**, que permiten generar representaciones comprimidas de caracteres a partir de sus rasgos visuales.

Trabajos como los de **Gogna y Majumdar (2019)** y **Tang et al. (2020)** han demostrado que estas representaciones pueden utilizarse como entrada para clasificadores simples o procesos de agrupamiento, facilitando la detección de similitudes estructurales entre caracteres sin etiquetado explícito. Asimismo, las variantes **denoising** y **variational** introducidas por Vincent et al. (2008) y revisadas por Rastogi (2024) permiten trabajar sobre imágenes ruidosas y conservar información semántica latente.

Estas técnicas han sido aplicadas con éxito en tareas de preprocesamiento y codificación visual, y constituyen la base metodológica del sistema desarrollado en este proyecto, permitiendo abordar el reconocimiento sin corpus anotados y adaptarse progresivamente a nuevos dominios caligráficos.

Modelos industriales y multimodales actuales

En paralelo a la evolución académica, grandes empresas tecnológicas han desarrollado soluciones avanzadas para OCR integradas en modelos multimodales y generativos. **Google**, mediante sus APIs de Vision AI y Document AI, ofrece herramientas de OCR basadas en redes neuronales profundas entrenadas sobre millones de documentos.

**OpenAI**, con modelos como **GPT-4 Vision**, ha introducido capacidades de OCR integradas en sistemas de comprensión visual. Estos modelos no sólo extraen texto, sino que lo interpretan en contexto, combinando visión por computador con lenguaje natural.

**Meta (Facebook)** ha trabajado en el sistema **OCR++** y en modelos multilingües como **No Language Left Behind**, capaces de transcribir texto manuscrito en más de cien lenguas con modelos centralizados y adaptativos.

Por su parte, **Mistral** ha impulsado modelos de lenguaje ligeros y open-source que pueden combinarse con módulos visuales en tareas de OCR locales o embebidas.

Estos avances han ampliado el alcance del OCR más allá del reconocimiento básico, hacia sistemas de comprensión visual completa. No obstante, su aplicabilidad a documentos históricos sigue siendo limitada si no se adaptan a las condiciones particulares de estos textos.

### Necesidad detectada

La creciente digitalización de archivos históricos ha permitido que millones de documentos estén disponibles públicamente en formato imagen. Sin embargo, la accesibilidad real a estos fondos documentales sigue siendo limitada, ya que la mayoría carece de transcripciones textuales estructuradas. Este desfase entre digitalización visual y procesamiento textual crea una barrera importante para la búsqueda, análisis y reutilización de información contenida en dichos documentos.

Un caso emblemático es el del Archivo General de Indias, que custodia una de las colecciones más valiosas sobre la administración del Imperio español. Aunque sus manuscritos han sido escaneados y publicados en portales como PARES, el contenido sigue siendo en gran parte inexplorable para métodos computacionales debido a la falta de texto editable. Esta situación impide que historiadores, lingüistas o investigadores digitales puedan aplicar técnicas modernas de análisis textual o minería de datos.

El problema se agrava por la ineficacia de los sistemas OCR tradicionales frente a documentos manuscritos antiguos. Estas herramientas están diseñadas para textos impresos con tipografías limpias, y fallan cuando se enfrentan a caligrafías variables, espaciados irregulares, símbolos arcaicos o soportes degradados. Además, la mayoría de los modelos modernos basados en aprendizaje supervisado requieren grandes volúmenes de datos anotados, lo cual es inviable en muchos archivos históricos donde no existen transcripciones manuales previas.

Frente a este panorama, surge la necesidad de desarrollar métodos alternativos, no supervisados y adaptativos que permitan recuperar estructuras textuales sin depender de corpus etiquetados. En este sentido, el uso de autoencoders entrenados carácter a carácter se presenta como una solución viable: estos modelos pueden aprender patrones estructurales directamente desde los datos, sin etiquetas, y generar reconstrucciones que aproximan la forma y estructura original del texto.

Este enfoque no solo tiene implicaciones prácticas para la transcripción automatizada, sino también metodológicas: permite repensar el tratamiento computacional del texto desde una perspectiva más robusta, que tolera ruido, deformaciones y ambigüedad. Además, al permitir una posterior integración con modelos generativos como GPT, se abre la posibilidad de crear sistemas híbridos capaces de recuperar y completar contenido textual histórico con mínima intervención humana.

Por tanto, el presente trabajo no solo responde a una necesidad técnica, sino también cultural y científica: la de hacer accesible, analizable y reutilizable un patrimonio documental que, aunque digitalizado, sigue siendo en muchos casos opaco para las herramientas digitales modernas.

## Objetivos

El objetivo general de este Trabajo de Fin de Grado es diseñar, desarrollar y evaluar un sistema de Reconocimiento Óptico de Caracteres (OCR) adaptado a documentos manuscritos históricos, específicamente del fondo documental del Archivo General de Indias, que funcione sin necesidad de datos etiquetados previos y que permita reconstruir texto legible a partir de imágenes escaneadas. Para ello, se plantea una solución basada en técnicas de aprendizaje no supervisado, agrupamiento automático y clasificación supervisada asistida por etiquetado semántico mínimo.

Este objetivo responde a la necesidad de mejorar el acceso estructurado al contenido textual de archivos digitalizados sin transcripción, superando las limitaciones de los OCR tradicionales en contextos paleográficos y facilitando la posterior explotación automatizada del texto mediante búsquedas o análisis computacionales.

### Objetivos específicos

* Implementar un sistema de segmentación automática de caracteres individuales a partir de imágenes completas de documentos históricos.
* Construir un modelo de autoencoder convolucional capaz de generar representaciones latentes comprimidas de los caracteres, preservando sus características visuales esenciales.
* Aplicar un algoritmo de clustering no supervisado (K-Means) sobre los vectores latentes para identificar patrones visuales y agrupar caracteres similares.
* Desarrollar una herramienta de etiquetado asistido que permita asignar etiquetas semánticas a los clústeres sin necesidad de anotar cada carácter individualmente.
* Entrenar un clasificador supervisado sobre el espacio latente etiquetado para predecir automáticamente los caracteres presentes en nuevas imágenes.
* Reconstruir líneas completas de texto a partir de la secuencia de predicciones realizadas sobre los caracteres segmentados.

## Requisitos técnicos

El desarrollo del sistema OCR propuesto requiere una serie de recursos técnicos tanto a nivel de software como de hardware. Estos requisitos se han definido en función de las necesidades del procesamiento de imágenes, el entrenamiento de modelos de aprendizaje automático y la manipulación de datos asociados a documentos históricos digitalizados.

### Requisitos de software:

* Lenguaje de programación: Python 3.12
* Entorno de desarrollo: Visual Studio Code + Jupyter Notebooks.
* Entornos y librerías principales:
  + PyTorch 2.0.1 (entrenamiento del autoencoder y preprocesamiento de tensores)
  + TensorFlow/Keras 2.12 (clasificador supervisado)
  + scikit-learn 1.1.3 (clustering con K-Means y métricas)
  + OpenCV 4.5.5.62 y PIL 9.2.0 (segmentación y procesamiento de imágenes)
  + Matplotlib 3.6.3 y torchvision 0.15.2 (visualización y etiquetado asistido)
  + NumPy 1.24.0 y pandas 1.5.3 (manipulación de datos)
  + JSON y Pickle (almacenamiento de mapeos y configuraciones)

### Requisitos de hardware:

* CPU con múltiples núcleos (mínimo 4 núcleos)
* 8 GB de memoria RAM (mínimo recomendado)
* GPU compatible con CUDA (opcional, pero recomendable para acelerar el entrenamiento del autoencoder)
* Almacenamiento disponible: ~2 GB para conjuntos de imágenes y salidas del modelo

### Recursos de datos:

* Imágenes escaneadas del Archivo General de Indias, descargadas directamente desde el portal PARES, en formato .jpg o .png.
* Segmentación automática en caracteres individuales almacenados localmente para su uso durante el entrenamiento.

### Dependencias externas:

* Sistemas operativos compatibles: Linux, Windows o macOS.
* Conexión a internet (únicamente necesaria para la descarga inicial de datos).

# Marco teórico

El Reconocimiento Óptico de Caracteres (OCR, por sus siglas en inglés, *Optical Character Recognition*) es una disciplina dentro del procesamiento de imágenes y la visión por computador que tiene como objetivo la conversión automática de texto presente en imágenes (ya sea impreso, mecanografiado o manuscrito) en información digital editable y procesable por máquina. Esta tecnología permite automatizar la digitalización documental, facilitando tareas como el archivado, la búsqueda semántica o la traducción de textos a formatos accesibles.

### 2.1 Fundamentos del OCR

Un sistema OCR tradicional consta de varias etapas estructuradas:

1. **Preprocesamiento de la imagen**: incluye escalado, binarización, corrección de ruido y alineación.
2. **Segmentación**: se divide el contenido en regiones de texto, líneas, palabras y finalmente caracteres individuales.
3. **Extracción de características**: cada carácter se representa mediante descriptores visuales (bordes, perfiles, contornos).
4. **Clasificación**: se asigna una etiqueta al patrón visual mediante técnicas estadísticas o modelos entrenados.

Este pipeline ha sido ampliamente empleado desde mediados del siglo XX en aplicaciones como la lectura automática de cheques, formularios o documentos administrativos. Sin embargo, su rendimiento se ha visto históricamente limitado a textos con alta regularidad tipográfica, bien alineados y escaneados en condiciones óptimas.

### Tipos de OCR

Existen distintas clasificaciones de los sistemas OCR según criterios como la naturaleza del texto, el tipo de escritura o la estrategia de reconocimiento empleada:

* **Según el tipo de escritura**:
  + OCR impreso: orientado a textos mecanografiados o impresos con fuentes conocidas.
  + ICR (Intelligent Character Recognition): dirigido al reconocimiento de texto manuscrito, generalmente estructurado y con restricciones (por ejemplo, formularios).
  + HTR (Handwritten Text Recognition): enfocado en la escritura cursiva o libre, como la presente en cartas, notas o documentos históricos.
* **Según la segmentación**:
  + OCR segmentado: cada carácter se identifica de forma aislada.
  + OCR de línea completa (CTC-based): se reconoce el texto sin segmentación explícita, como en CRNNs o modelos transformer.
* **Según el entrenamiento**:
  + OCR basado en reglas o plantillas: compara cada carácter con un catálogo predefinido.
  + OCR entrenado por aprendizaje automático: aprende a clasificar caracteres a partir de ejemplos anotados.

### Evolución tecnológica

Durante décadas, los sistemas OCR se basaron en reglas heurísticas y correspondencia de patrones, lo que los hacía muy sensibles a pequeñas deformaciones o variaciones tipográficas. Con la aparición del aprendizaje automático, comenzaron a utilizarse modelos estadísticos como *Support Vector Machines (SVM)* o *Multilayer Perceptrons (MLP)*, que permitieron mejorar la generalización del reconocimiento.

El avance más importante llegó con el aprendizaje profundo. Modelos como las Redes Neuronales Convolucionales (CNN) automatizaron la extracción de características visuales. Posteriormente, modelos como CRNN (Shi et al., 2016) y aquellos basados en CTC (Graves et al., 2009) eliminaron la necesidad de segmentar los caracteres manualmente, tratando la imagen como una secuencia visual completa.

Más recientemente, se han explorado arquitecturas más complejas con atención visual (*attention-based OCR*) y modelos multimodales que combinan visión con lenguaje, como en GPT-4 Vision (OpenAI), TrOCR (Microsoft) o Donut (Naver AI), capaces de interpretar texto dentro de contexto semántico o estructural.

### OCR manuscrito y sus desafíos

El reconocimiento de escritura manuscrita representa uno de los mayores retos del OCR moderno. A diferencia de los textos impresos, los manuscritos presentan:

* Alta variabilidad caligráfica, tanto entre autores como entre muestras de un mismo texto.
* Problemas de degradación visual, como manchas, papel envejecido o tinta desvanecida.
* Formas irregulares y superposición de trazos, especialmente en escritura cursiva.
* Ausencia de anotación estructurada que permita el entrenamiento supervisado.

Por estas razones, la eficacia de los OCR comerciales y de los modelos preentrenados se ve reducida significativamente cuando se aplican sobre documentos históricos o manuscritos no normalizados.

### Hacia enfoques no supervisados y adaptativos

Para sortear la dependencia de corpus etiquetados y aumentar la flexibilidad del reconocimiento, han surgido enfoques alternativos que combinan representación no supervisada, agrupamiento automático y clasificación ligera.

Estos sistemas no buscan predecir directamente los caracteres, sino que primero codifican visualmente sus rasgos mediante modelos como autoencoders o transformers auto-regresivos, agrupan caracteres similares con algoritmos como K-Means, y permiten al usuario asignar etiquetas a grupos completos. Posteriormente, un clasificador entrenado sobre ese espacio comprimido permite generalizar a nuevas muestras.

Esta filosofía de diseño modular, adaptativa y con supervisión mínima es la base metodológica sobre la que se apoya el presente proyecto.

# Metodología

La metodología seguida para el desarrollo del sistema OCR se ha estructurado en torno a un flujo de trabajo modular y reproducible, diseñado para operar sin necesidad de datos etiquetados previamente. Este enfoque se implementó de forma coherente en los dos cuadernos de trabajo del núcleo experimental del proyecto.

La Figura 1 muestra un esquema simplificado del proceso general aplicado. Cada una de las fases representa una etapa clave en el pipeline de digitalización y transcripción automática de texto manuscrito, permitiendo transformar imágenes escaneadas en texto legible mediante una combinación de aprendizaje no supervisado, etiquetado asistido y clasificación supervisada.

Diagrama

El contenido generado por IA puede ser incorrecto.

**Ilustración 1.** Metodología aplicada a cada libro*. Fuente (Guillermo Martínez, 2025)*

1. **Preprocesamiento de imágenes**: las imágenes manuscritas son transformadas mediante técnicas de escalado a grises, binarización y segmentación para extraer caracteres individuales de forma normalizada.
2. **Entrenamiento no supervisado**: se entrena un autoencoder convolucional para obtener una representación latente comprimida de cada carácter, capturando patrones visuales relevantes sin etiquetas.
3. **Asignación manual de clústeres**: los vectores latentes se agrupan mediante clustering (K-Means), y se asignan etiquetas manuales a los clústeres completos en lugar de a ejemplos individuales, reduciendo el esfuerzo de anotación.
4. **Entrenamiento supervisado**: utilizando las etiquetas generadas, se entrena un clasificador ligero que permite generalizar la predicción de caracteres a nuevas muestras no vistas.
5. **Reconstrucción del texto**: los caracteres predichos se reordenan según su posición espacial original en la imagen para formar frases completas en formato digital.

Esta metodología ha sido aplicada con éxito tanto en casos con escritura clara como en contextos históricos más complejos, validando su flexibilidad y robustez. En estas secciones se detallan cada fase con mayor profundidad, al referirse a su implementación práctica en los cuadernos de trabajo.

## Preprocesamiento de imágenes

El primer paso del sistema consiste en transformar imágenes completas de texto en representaciones normalizadas de caracteres individuales. Esta etapa es fundamental, ya que condiciona la calidad de las representaciones latentes posteriores.

### Escalado a escala de grises

Todas las imágenes de entrada, independientemente de su formato original, se convierten a escala de grises. Esta transformación reduce la dimensionalidad del canal de color sin pérdida significativa de información visual relevante para el reconocimiento de caracteres. Para ello, se utiliza la conversión ponderada estándar basada en luminancia perceptiva:

donde R, G, B son las intensidades de cada canal de color en el píxel (x,y). El resultado es una imagen , en un solo canal.

### Umbralización y segmentación

Para aislar los caracteres individuales, se aplica una binarización mediante umbral estático, invirtiendo la intensidad para que el fondo sea negro y el trazo blanco. A partir de la imagen binaria resultante, se detectan contornos externos mediante algoritmos de seguimiento de bordes. Cada contorno se asocia a una región candidata a carácter.

Una vez identificadas estas regiones, se extraen las cajas delimitadoras correspondientes y se ordenan en sentido de lectura, priorizando primero el eje vertical (líneas) y luego el eje horizontal (caracteres dentro de la línea). Esta ordenación garantiza que la secuencia resultante preserve el flujo del texto original.

### Redimensionado y normalización

Cada fragmento segmentado se recorta con base en su contorno y se redimensiona a una matriz cuadrada de 28×28 píxeles. Esta transformación asegura uniformidad dimensional en las entradas al sistema, independientemente del tamaño original del carácter. Además, se aplica una normalización lineal al rango dividiendo por 255, lo cual mejora la estabilidad numérica durante el aprendizaje posterior.

El resultado de esta fase es un conjunto de imágenes individuales de caracteres, uniformemente escaladas y preprocesadas, listas para su proyección al espacio latente.

## Entrenamiento no supervisado (Autoencoder)

El sistema de reconocimiento parte de la necesidad de extraer representaciones compactas y significativas de las imágenes de caracteres sin recurrir a etiquetas. Para ello, se entrena un autoencoder convolucional, una arquitectura que aprende a proyectar datos a un espacio latente de menor dimensión de manera no supervisada, preservando la información estructural necesaria para su reconstrucción.

Un autoencoder está compuesto por dos funciones parametrizadas:

* Un codificador
* Un decodificador

donde es la dimensión de la imagen de entrada, y es la dimensión del espacio latente.

### Codificador (encoder)

El encoder aplica una secuencia de convoluciones con funciones de activación no lineales que transforman progresivamente la imagen de entrada en un tensor de baja resolución, pero alta profundidad. A cada imagen se le aplica una transformación compuesta:

donde cada representa una operación convolucional seguida de una función de activación (ReLU) y, en algunos casos, de reducción espacial mediante stride. El resultado es un tensor con canales y resolución reducida, que se reinterpreta como un vector con .

Este vector latente representa una codificación comprimida del carácter original, en la que se espera que se conserven las características morfológicas esenciales, mientras se filtra la redundancia visual.

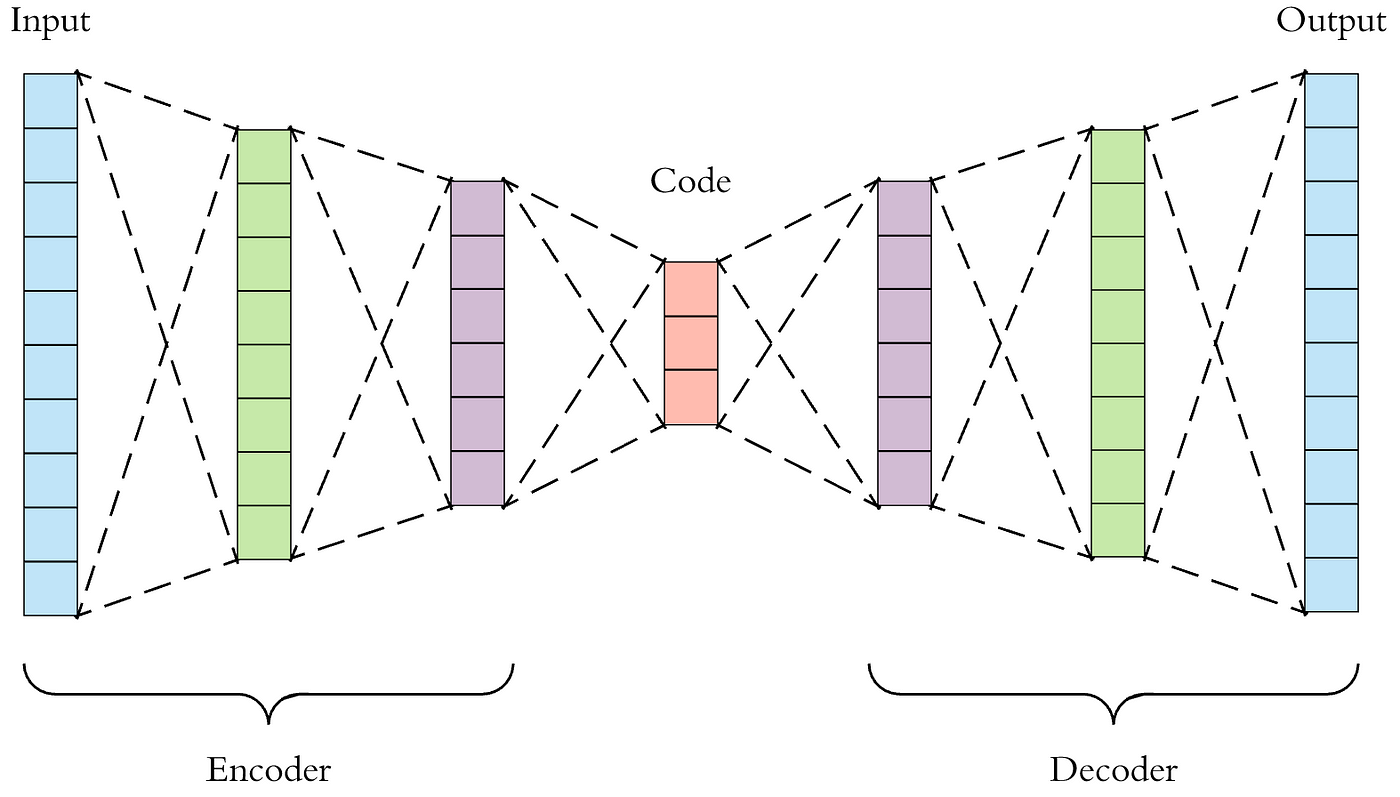
### Decodificador (decoder)

El decoder aplica una secuencia inversa de operaciones, convoluciones transpuestas, que proyectan nuevamente el vector latente al espacio de imagen original. Formalmente:

donde cada es una capa de expansión espacial que invierte progresivamente la reducción del encoder. La salida final es una reconstrucción que intenta aproximar a la imagen original.

El objetivo del entrenamiento consiste en minimizar la pérdida de reconstrucción entre y . Dado que los valores de los píxeles están en el intervalo , se emplea la entropía cruzada binaria como función de pérdida:

Este criterio penaliza fuertemente las diferencias en los píxeles reconstruidos, incentivando una codificación que preserve tanto el trazo como el fondo del carácter.



**Ilustración 2.** Estructura de un autoencoder*. Fuente (Guillermo Martínez, 2025)*

## Asignación manual de clústeres

Una vez agrupadas las representaciones latentes mediante un algoritmo no supervisado, es necesario establecer una correspondencia semántica entre cada clúster y el carácter que representa. Este proceso se realiza de forma manual, pero asistida por visualización.

Para ello, se extrae una muestra representativa de elementos de cada clúster y se presentan de manera conjunta. El observador humano asigna a cada grupo una etiqueta simbólica, por ejemplo, una letra, número o símbolo de puntuación en función de la regularidad visual observada.

Este paso permite vincular los identificadores internos de los clústeres a una etiqueta de carácter explícita, transformando la salida del sistema no supervisado en un conjunto parcialmente anotado listo para su uso en tareas de clasificación supervisada.

## Entrenamiento supervisado (K-means)

Una vez obtenidas las representaciones latentes de los caracteres mediante el autoencoder, el siguiente paso consiste en agrupar dichos vectores en categorías visuales homogéneas utilizando el algoritmo de clustering K-Means. Este procedimiento no supervisado permite identificar patrones estructurales sin necesidad de etiquetas previas.

### Vectorización del espacio latente

Cada imagen de carácter procesada por el encoder se transforma en un tensor latente tridimensional . Este tensor se aplana en un vector , donde , para facilitar su tratamiento con algoritmos clásicos de agrupamiento.

El conjunto de todos los vectores latentes ​ constituye el espacio de entrada para el algoritmo K-Means.

### Agrupamiento con K-Means

El algoritmo K-Means tiene como objetivo particionar el conjunto de vectores en kkk grupos, minimizando la varianza intra-clúster. Formalmente, resuelve el siguiente problema de optimización:

donde es el centroide del clúster ​. El número de clústeres se establece previamente y se corresponde con una estimación del número de clases distintas que se espera encontrar en los datos (por ejemplo, letras del alfabeto más el espacio).

### Etiquetado para clasificación

Tras aplicar K-Means sobre el conjunto de vectores latentes, cada muestra queda etiquetada con el identificador de su clúster. Estas etiquetas se cruzan posteriormente con el mapeo semántico realizado manualmente (véase 3.3), generando un conjunto de datos etiquetado útil para tareas de clasificación.

El conjunto resultante se divide en subconjuntos de entrenamiento y prueba, manteniendo la correspondencia entre vectores y etiquetas semánticas. De este modo, el espacio latente generado por el autoencoder se convierte en la base para un sistema de clasificación supervisada.

## Reconstrucción del texto

Una vez entrenado el clasificador sobre el espacio latente etiquetado, el sistema puede inferir sobre nuevas imágenes. Este proceso consiste en aplicar de forma secuencial todas las transformaciones previamente definidas: segmentación, codificación, clasificación y reconstrucción textual.

Dado un conjunto de caracteres extraídos de una imagen de entrada, cada uno se transforma en un vector latente a través del encoder entrenado. Estos vectores se introducen en el clasificador supervisado, que predice para cada uno una etiqueta alfabética correspondiente. Las predicciones individuales se ordenan de acuerdo con la disposición espacial original de los caracteres.

La secuencia final de etiquetas constituye el texto reconstruido por el sistema. Este texto puede almacenarse, mostrarse o utilizarse como entrada para tareas posteriores. La calidad de esta reconstrucción depende directamente de la fidelidad del proceso de segmentación, la expresividad del espacio latente y la precisión del clasificador entrenado.

# Proyecto

## Resumen de contribuciones y productos desarrollados

A lo largo de este Trabajo de Fin de Grado se ha diseñado e implementado un sistema completo de Reconocimiento Óptico de Caracteres (OCR) orientado al procesamiento de escritura manuscrita, especialmente en el contexto de documentos históricos. El sistema desarrollado permite reconocer caracteres sin necesidad de un conjunto de datos previamente etiquetado, combinando técnicas de aprendizaje no supervisado y supervisado de manera eficiente y modular.

El resultado final se materializa en dos notebooks funcionales, que implementan el pipeline completo y abordan distintos escenarios de prueba: uno enfocado en frases modernas manuscritas con separación de palabras, y otro orientado al análisis de textos históricos con grafías irregulares y condiciones reales de escaneo.

### Contribuciones principales

* **Desarrollo de un sistema OCR no supervisado basado en autoencoders y clustering:**  
  El sistema propuesto evita la dependencia de conjuntos de datos etiquetados, extrayendo patrones visuales directamente del espacio latente aprendido por un autoencoder convolucional.
* **Agrupamiento visual automático mediante K-Means:**  
  Los vectores latentes generados por el encoder son organizados en clústeres que agrupan caracteres similares sin intervención humana directa.
* **Procedimiento de etiquetado semiautomático por clústeres:**  
  El sistema permite asignar etiquetas a clústeres completos tras visualizar una pequeña muestra representativa, facilitando así la creación de un dataset etiquetado sin necesidad de etiquetar carácter por carácter.
* **Entrenamiento de un clasificador sobre el espacio latente comprimido:**  
  Una vez etiquetados los clústeres, se entrena un modelo supervisado que aprende a asociar vectores latentes con su clase correspondiente, permitiendo la inferencia directa sobre nuevos caracteres.
* **Reconstrucción de frases manuscritas a partir de predicciones individuales:**  
  Los caracteres clasificados son ordenados según su posición original en la imagen para reconstruir automáticamente frases completas con sentido, respetando espacios y estructuras léxicas.

### Productos desarrollados

* **Notebook 1: Autoencoder\_Frases\_Con\_Espacios.ipynb**  
  Implementa el pipeline completo sobre frases manuscritas modernas. Se trabaja con EMNIST para entrenamiento, y se prueba sobre imágenes reales que contienen palabras con espacios. Incluye preprocesamiento, entrenamiento del autoencoder, clustering, etiquetado, clasificación y reconstrucción final del texto.
* **Notebook 2: Autoencoder\_Textos\_Antiguos.ipynb**  
  Aplica el sistema sobre ejemplos reales de documentos con estilo antiguo o histórico. Se adapta el preprocesamiento a condiciones más complejas de imagen (ruido, inclinaciones, deformación), permitiendo evaluar la robustez del sistema en contextos de archivo.
* **Modelo de autoencoder entrenado y reutilizable:**  
  Capaz de codificar caracteres manuscritos en vectores latentes compactos, entrenado inicialmente sobre el dataset EMNIST.
* **Modelo de clasificación entrenado sobre espacio latente:**  
  Una red supervisada que aprende a asignar etiquetas a nuevos vectores codificados según la estructura aprendida durante la agrupación.
* **Diccionario de mapeo clúster-etiqueta (cluster\_label\_mapping.json):**  
  Estructura clave que enlaza los clústeres obtenidos automáticamente con la etiqueta semántica asignada por el usuario.
* **Dataset propio segmentado a partir de imágenes reales:**  
  Conjunto de caracteres extraídos de frases manuscritas escritas a mano y escaneadas para entrenamiento y validación del modelo.

Tabla comparativa de notebooks desarrollados y OCR tradicionales

|  |  |  |  |
| --- | --- | --- | --- |
| Característica / Fase | Autoencoder Textos Modernos | Autoencoder Textos Antiguos | OCRs tradicionales (p.ej. Tesseract) |
| Tipo de entrada | Frases modernas manuscritas con separación clara | Palabras y líneas de estilo antiguo, escritura más densa | Texto impreso o digital escaneado |
| Dataset de entrenamiento | EMNIST Letters (letras segmentadas, limpias) | EMNIST + ejemplos propios segmentados de textos históricos | Tipografías modernas preentrenadas |
| Arquitectura de autoencoder | Convolucional, simétrica, dimensión latente media | Convolucional, con ajuste más profundo para capturar ruido | No aplica |
| Segmentación | Contornos en blanco y negro, limpieza mínima | Más filtros morfológicos, ajustes de posición y proporción | Integrada parcialmente, sensible a ruido |
| Clustering (K-Means) | 63 clústeres, buena separación por forma | 80+ clústeres, agrupamiento más disperso | No utilizado; requiere reconocimiento directo |
| Etiquetado | Asistido visualmente por clúster (manual con apoyo gráfico) | Asistido, pero requiere más revisión por ruido e inclinación | No hay; el sistema ya está pre-entrenado y cerrado |
| Clasificación final | Red densa sobre espacio latente, buena precisión local | Clasificador más robusto, adaptado a distorsiones | Basado en patrones OCR fijos, débil ante escritura manual |
| Reconstrucción de texto | Ordenamiento por coordenada X, frase coherente con espacios | Más difícil por densidad de caracteres | Reconstrucción directa, muy débil con manuscritos |
| Flexibilidad y adaptabilidad | Alta: entrenamiento rápido sobre nuevos estilos | Alta, aunque requiere más ajuste en preprocesado | Muy baja sin retraining o fine-tuning manual |
| Dependencia de etiquetas previas | Ninguna: etiquetas generadas manualmente por clústeres | Ninguna: mismo método adaptado | Requiere corpus etiquetado previamente |
| Ventajas | Interpretabilidad, control total del pipeline | Aplicabilidad realista sobre textos históricos | Precisión alta en impresos modernos |
| Limitaciones | Menor robustez en grafías complejas | Segmentación difícil, más ruido visual | Incapacidad para manuscritos os documentos degradados |

## Planificación temporal

El desarrollo del sistema OCR se ha organizado en una serie de etapas consecutivas, siguiendo una lógica de experimentación progresiva. Cada fase ha permitido identificar las limitaciones de los enfoques anteriores y tomar decisiones informadas sobre la dirección del proyecto. A continuación, se describen los principales pasos realizados durante el trabajo:

Fase 1: Evaluación de librerías OCR existentes

* Objetivo: Explorar soluciones OCR ya implementadas en Python para comprobar su aplicabilidad directa sobre escritura manuscrita.
* Herramientas evaluadas:
  + pytesseract (basado en Tesseract OCR de Google)
  + spaCy (aunque orientado a NLP, se exploró su uso conjunto para procesar textos OCR extraídos)
* Resultado:  
  Las pruebas realizadas mostraron un rendimiento muy pobre sobre documentos manuscritos. pytesseract, aunque eficaz en textos impresos, no fue capaz de segmentar ni reconocer caracteres manuscritos de forma fiable. Esto motivó el abandono del enfoque basado en OCR comercial y el paso a una solución personalizada.

Fase 2: Agrupamiento de caracteres con K-Means

* Objetivo: Construir un pipeline sin supervisión para organizar caracteres visualmente similares, partiendo de imágenes segmentadas manual o automáticamente.
* Acciones realizadas:
  + Preprocesamiento de caracteres (escala de grises, normalización, redimensionado).
  + Aplanado de las imágenes en vectores de características de alta dimensión.
  + Aplicación de K-Means sobre estos vectores.
* Resultado:  
  El clustering basado en píxeles crudos generaba agrupaciones inconsistentes, sensibles a pequeñas variaciones. Si bien algunos clústeres mostraban cohesión, muchos agrupaban caracteres diferentes o separaban formas idénticas con pequeñas deformaciones. Esto evidenció la necesidad de una representación más robusta y comprimida del contenido visual.

Fase 3: Codificación latente con autoencoder

* Objetivo: Diseñar una arquitectura de autoencoder convolucional para proyectar los caracteres a un espacio latente más representativo, sobre el cual aplicar el clustering.
* Acciones realizadas:
  + Entrenamiento de un autoencoder sobre caracteres segmentados.
  + Extracción de vectores latentes comprimidos desde la capa intermedia del encoder.-
  + Aplicación de K-Means sobre el espacio latente, obteniendo clústeres visualmente coherentes.
  + Etiquetado manual por clústeres y entrenamiento del clasificador final.
* Resultado:  
  El uso del autoencoder permitió reducir ruido y capturar patrones visuales estables en el espacio latente. Las agrupaciones mejoraron notablemente, y el clasificador entrenado sobre vectores latentes etiquetados logró resultados precisos sobre nuevas muestras. Esta fase definió la estructura final del sistema OCR implementado.

## Recursos empleados

El sistema OCR desarrollado en este trabajo se apoya en un conjunto de herramientas y librerías de software libre que permiten abordar de forma modular y reproducible todas las fases del procesamiento: desde la manipulación de imágenes hasta el entrenamiento de modelos de aprendizaje automático y la visualización de resultados. A continuación, se describen los recursos utilizados, clasificados por su propósito funcional dentro del sistema.

### Gestión y manipulación de datos

* NumPy:  
  Base para el trabajo con arrays multidimensionales. Utilizado para representar y transformar imágenes en formato matricial, operar sobre lotes de datos, codificar tensores intermedios y calcular distancias entre vectores latentes.
* Pandas:  
  Se ha utilizado para la gestión de etiquetas, creación de estructuras tabulares durante el entrenamiento y evaluación del modelo, y análisis de agrupaciones.
* JSON y Pickle:  
  Módulos estándar utilizados para guardar y cargar estructuras persistentes del sistema: mapeos de clúster a etiqueta, configuraciones de entrenamiento, vectores codificados y salidas de inferencia.

### Procesamiento y segmentación de imágenes

* OpenCV (cv2):  
  Herramienta clave para el preprocesamiento. Se ha utilizado para convertir imágenes a escala de grises, aplicar umbralización, detectar contornos, extraer regiones individuales (bounding boxes) y segmentar caracteres. También permite invertir colores y ordenar caracteres en base a coordenadas.
* PIL (Python Imaging Library):  
  Complemento de OpenCV para la carga, redimensionado y conversión de imágenes entre formatos compatibles con TensorFlow y PyTorch. Ha sido útil especialmente en las fases de inspección visual y en la creación de imágenes normalizadas.
* glob, os, pathlib, shutil:  
  Módulos para gestionar rutas, crear carpetas, recorrer directorios y manipular archivos de imágenes segmentadas y etiquetas asociadas.

### Visualización

* Matplotlib:  
  Utilizada en la mayoría de las etapas del proyecto para representar imágenes, visualizar matrices de confusión, trazar curvas de pérdida durante el entrenamiento y mostrar agrupaciones de caracteres en el espacio latente.
* Seaborn:  
  Empleada para generar gráficos más expresivos, como histogramas de distribución de clústeres, análisis de métricas y apoyo a la interpretación de resultados.
* torchvision.utils (make\_grid):  
  Utilidad muy útil para mostrar múltiples imágenes en una sola cuadrícula, lo que facilita la inspección rápida de ejemplos por clúster durante el etiquetado manual.

### Aprendizaje automático y modelos

* TensorFlow / Keras:  
  Plataforma principal para el entrenamiento de modelos. Se ha utilizado para definir y entrenar el autoencoder convolucional y el clasificador denso final. Permite trabajar con capas convolucionales, activaciones no lineales, funciones de pérdida, optimizadores (Adam) y callbacks de entrenamiento como EarlyStopping.
* scikit-learn:  
  Utilizada principalmente para implementar el algoritmo de clustering K-Means sobre los vectores latentes generados por el autoencoder. También se ha empleado para la división del conjunto de datos (train/test split), cálculo de métricas (accuracy, classification report), y análisis de agrupamiento.

### Control de flujo y utilidades

* random:  
  Utilizado para la selección aleatoria de muestras dentro de cada clúster con el fin de mostrar ejemplos representativos durante el proceso de etiquetado asistido.
* time:  
  Aplicado de forma auxiliar para medir tiempos de ejecución durante el entrenamiento o la segmentación, ayudando a optimizar procesos iterativos.

### Otras herramientas

* Jupyter Notebook (VS Code):  
  Entorno de desarrollo interactivo donde se ha integrado todo el código del proyecto, permitiendo una ejecución modular, visualización inmediata de resultados y organización clara del pipeline.

## Trabajo desarrollado

El sistema OCR implementado en este Trabajo de Fin de Grado se ha estructurado en torno a dos notebooks funcionales, diseñados para abordar y evaluar distintos escenarios de entrada manuscrita. Esta organización responde a la necesidad de verificar la robustez, adaptabilidad y precisión del modelo en situaciones reales, desde frases claras y bien segmentadas hasta ejemplos más complejos procedentes de manuscritos antiguos.

Ambos notebooks implementan el pipeline completo del sistema OCR, que incluye:

* Entrenamiento de un autoencoder convolucional para la extracción de representaciones latentes de caracteres manuscritos.
* Agrupamiento no supervisado de dichos vectores mediante K-Means.
* Etiquetado asistido de los clústeres generados, asignando manualmente una clase por grupo.
* Entrenamiento de un clasificador supervisado sobre el espacio latente comprimido.
* Segmentación y preprocesamiento de nuevas imágenes.
* Clasificación de caracteres y reconstrucción automática del texto a partir del orden espacial de los segmentos.

Esta estructura común permite comparar el rendimiento del sistema bajo distintas condiciones, y facilita su extensión o adaptación a nuevos tipos de documentos. Los notebooks desarrollados son los siguientes:

* **Notebook 1: Textos modernos**  
  Aplicación del sistema sobre frases manuscritas limpias y estructuradas, con separación clara entre palabras. Sirve como base de validación del pipeline completo.
* **Notebook 2: Textos antiguos**  
  Evaluación del sistema sobre textos históricos con escritura manuscrita irregular, orientado a simular condiciones reales del Archivo General de Indias.

Cada cuaderno se describe en las siguientes secciones de forma individual, siguiendo una estructura homogénea: parametrización del autoencoder, agrupamiento y etiquetado, preprocesamiento de la imagen a analizar, asignación de clústeres y reconstrucción final del texto.

### Cuaderno 1: Reconstrucción de frases modernas

Este cuaderno implementa el flujo completo del sistema OCR aplicado a una imagen manuscrita que contiene una frase con palabras separadas. El objetivo principal es comprobar que el sistema es capaz de identificar los caracteres individualmente y reconstruir correctamente la secuencia completa del texto, incluyendo los espacios entre palabras.

**1. Entrenamiento y parametrización del autoencoder**

El entrenamiento del autoencoder se realiza utilizando el conjunto de datos EMNIST Balanced, una extensión del clásico MNIST que incluye letras mayúsculas, minúsculas, dígitos y algunos símbolos manuscritos. Este dataset proporciona ejemplos ya segmentados y normalizados de caracteres, permitiendo entrenar el modelo sin segmentación manual previa.

Cada imagen se transforma en un vector plano de 784 valores (28×28 píxeles) y se normaliza al rango [0,1]. Este preprocesamiento se implementa mediante funciones de mapeo sobre los datasets train y test combinados desde tensorflow\_datasets.

El modelo definido es un autoencoder completamente denso, con arquitectura simétrica: cuatro capas densas en el encoder, que reducen la entrada a un vector latente de 64 dimensiones, y otras cuatro en el decoder, que reconstruyen la entrada original. El modelo se entrena durante 50 épocas usando MSE como función de pérdida y el optimizador Adam con tasa de aprendizaje 0.001.

*input\_img = keras.Input(shape= (784,))*

*encoded = layers.Dense(500, activation='relu') (input\_img)*

*encoded = layers.Dense(500, activation='relu') (encoded)*

*encoded = layers.Dense(2000, activation='relu') (encoded)*

*encoded = layers.Dense(64, activation=None)(encoded)*

*decoded = layers.Dense(2000, activation='relu') (encoded)*

*decoded = layers.Dense(500, activation='relu') (decoded)*

*decoded = layers.Dense(500, activation='relu') (decoded)*

*decoded = layers.Dense(784, activation='sigmoid') (decoded)*

*autoencoder = keras.Model(input\_img, decoded)*

*encoder = keras.Model(input\_img, encoded)*

*autoencoder.compile(optimizer=keras.optimizers.Adam(0.001), loss='mse')*

*autoencoder.fit(X, X, epochs=50, batch\_size=256)*

Mapa

El contenido generado por IA puede ser incorrecto.

**Ilustración 3.** *Titulo. Fuente (Guillermo Martínez, 2025)*

Cada punto en el gráfico representa un carácter codificado, y los colores indican la asignación a clúster. La figura muestra que el modelo ha aprendido a separar grupos visualmente consistentes de forma clara.

**2. Agrupamiento y etiquetado manual**

Una vez entrenado el autoencoder, se obtienen los vectores latentes de todos los caracteres procesados aplicando el encoder sobre los datos de entrada. Estos vectores comprimidos son la base para aplicar el algoritmo K-Means, seleccionando 47 clústeres para reflejar las clases potenciales: letras (mayúsculas y minúsculas), el espacio y ruido visual.

La calidad del agrupamiento se evalúa visualmente con UMAP, una técnica de reducción de dimensionalidad que proyecta los vectores latentes en dos dimensiones. Esto permite inspeccionar la separación entre clústeres y comprobar que el modelo ha aprendido una representación estructurada.

El etiquetado de los clústeres se realiza de forma semiautomática: se seleccionan ejemplos representativos de cada grupo y se asigna manualmente una etiqueta textual (por ejemplo, 'A', 'B', ' ', etc.). Esta información se guarda en un archivo cluster\_label\_mapping.json, que será reutilizado en la fase de clasificación.

*encoded\_imgs = encoder.predict(X) kmeans = KMeans(n\_clusters=47).fit(encoded\_imgs)*

*embedding = umap.UMAP(n\_components=2).fit\_transform(encoded\_imgs) plt.scatter(embedding**[:, 0], embedding**[:, 1], c=kmeans.labels\_, cmap='Spectral')*

*with open('cluster\_label\_mapping.json') as f:*

*cluster\_label\_mapping = json.load(f)*

Texto

El contenido generado por IA puede ser incorrecto.**Ilustración 4.** *Titulo. Fuente (Guillermo Martínez, 2025)*

Este proceso produce un diccionario de correspondencias clúster → carácter, que se utilizará posteriormente para clasificar nuevos caracteres en función de su pertenencia a un clúster ya etiquetado.

**3. Preprocesamiento de imagen con frase manuscrita**

La imagen de prueba contiene la frase manuscrita **"EL OCR DE ESTE TFG DETECTA FRASES CON PRECISION"**, escrita con buena legibilidad y fondo claro. El procesamiento aplicado sigue los siguientes pasos:

1. Conversión a escala de grises.
2. Umbralización binaria estática (invirtiendo el fondo).
3. Detección de contornos externos con OpenCV.
4. Ordenación de los caracteres detectados por coordenada horizontal (x).
5. Recorte, redimensionado a 28×28 píxeles y normalización al rango [0,1].

Cada carácter segmentado se convierte en un vector plano, como en el entrenamiento original.

*img = cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE) \_, thresh = cv2.threshold(img, 127, 255, cv2.THRESH\_BINARY\_INV) contours, \_ = cv2.findContours(thresh, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)*

*segments = [] for cnt in sorted(contours, key=lambda c: cv2.boundingRect(c)[0]): x, y, w, h = cv2.boundingRect(cnt) char\_img = cv2.resize(thresh[y:y+h, x:x+w], (28, 28)) segments.append((x, char\_img / 255.0))*

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Imagen de la pantalla de un celular con texto e imagen

El contenido generado por IA puede ser incorrecto.

**Ilustración 5.** *Titulo. Fuente (Guillermo Martínez, 2025)*

**4. Asignación de clústeres a los segmentos**

Para clasificar los caracteres, cada imagen segmentada es convertida en un vector y pasada por el encoder, obteniendo su representación latente. Se utiliza el modelo de clustering K-Means previamente entrenado para determinar a qué clúster pertenece. Luego, se consulta el diccionario de etiquetas para traducir ese clúster a un carácter textual.

*predictions = [] for x, segment in segments: latent = encoder.predict(segment.reshape(1, -1)) cluster = kmeans.predict(latent) [0] char = cluster\_label\_mapping.get(str(cluster), '?') predictions.append({'x': x, 'char': char})*

Una vez conocido el clúster, se consulta el diccionario de etiquetas manuales para determinar la clase textual asociada al carácter.

Imagen que contiene Código QR

El contenido generado por IA puede ser incorrecto.

**Ilustración 6.** *Titulo. Fuente (Guillermo Martínez, 2025)*

**5. Reconstrucción final del texto**

Una vez obtenidos todos los caracteres clasificados, estos se ordenan por su posición horizontal original en la imagen. Finalmente, se concatenan sus etiquetas para reconstruir automáticamente la frase manuscrita.

*sorted\_chars = sorted(predictions, key=lambda x: x['x']) final\_text = ''.join([p['char']*

*for p in sorted\_chars])*

*print ("Texto detectado:", final\_text)*

Aunque el sistema puede presentar algunas confusiones entre caracteres similares (por ejemplo, ‘I’ y ‘l’, o ‘O’ y ‘0’), la reconstrucción es en general correcta y coherente. Este cuaderno valida con éxito la funcionalidad del sistema OCR completo sobre escritura manuscrita clara, cumpliendo el objetivo de segmentar, clasificar y reconstruir texto sin necesidad de datos etiquetados previos.

### Cuaderno 2: Textos antiguos

Este cuaderno está diseñado para evaluar el rendimiento del sistema OCR frente a un escenario más exigente: escritura manuscrita de estilo antiguo, con mayor variabilidad en los trazos, ruido visual y condiciones de escaneo menos controladas. El objetivo es validar la capacidad del sistema para adaptarse a textos históricos similares a los encontrados en el Archivo General de Indias.

**1. Entrenamiento y parametrización del autoencoder**

Para esta fase se reutiliza el dataset EMNIST Letters como base de entrenamiento. Aunque su estilo no refleja fielmente la escritura antigua, proporciona ejemplos normalizados que permiten al autoencoder aprender patrones visuales consistentes.

Las imágenes se cargan y preprocesan con tensorflow\_datasets, transformándolas a vectores de 784 elementos (28x28 píxeles aplanados) y normalizándolos. La arquitectura del autoencoder es idéntica a la empleada en el cuaderno anterior: cuatro capas densas en el encoder y otras cuatro en el decoder, con un espacio latente de 64 dimensiones.

*input\_img = keras.Input(shape= (784,))*

*encoded = layers.Dense(500, activation='relu') (input\_img)*

*encoded = layers.Dense(500, activation='relu') (encoded)*

*encoded = layers.Dense(2000, activation='relu') (encoded)*

*encoded = layers.Dense(64, activation=None)(encoded)*

*decoded = layers.Dense(2000, activation='relu') (encoded)*

*decoded = layers.Dense(500, activation='relu') (decoded)*

*decoded = layers.Dense(500, activation='relu') (decoded)*

*decoded = layers.Dense(784, activation='sigmoid') (decoded)*

*autoencoder = keras.Model(input\_img, decoded)*

*encoder = keras.Model(input\_img, encoded)*

*autoencoder.compile(optimizer=keras.optimizers.Adam(0.001), loss='mse')*

El entrenamiento se realiza con los datos aplanados y normalizados. Una vez entrenado, se generan los vectores latentes, que serán utilizados para la fase de clustering.

Imagen que contiene Texto

El contenido generado por IA puede ser incorrecto.**Ilustración 7.** *Titulo. Fuente (Guillermo Martínez, 2025)*

**2. Agrupamiento y etiquetado manual**

Con los vectores latentes obtenidos del autoencoder, se aplica K-Means con 80 clústeres para capturar mejor la diversidad visual de los caracteres antiguos. Este número se elige como compromiso entre granularidad y manejabilidad del etiquetado.

encoded\_imgs = encoder.predict(X)

kmeans = KMeans(n\_clusters=80)

kmeans.fit(encoded\_imgs)

Los vectores se visualizan en 2D mediante UMAP, lo que permite observar una estructura más dispersa que en el caso anterior, debido a la mayor complejidad visual de los caracteres.

Para cada clúster, se presentan muestras representativas y se procede al etiquetado manual asistido. Algunos clústeres se descartan como "ruido" cuando no representan formas reconocibles.

*embedding = umap.UMAP(n\_components=2).fit\_transform(encoded\_imgs) plt.scatter(embedding**[:, 0], embedding**[:, 1], c=kmeans.labels\_, cmap='Spectral')*

Las etiquetas se almacenan en cluster\_label\_mapping.json, y posteriormente se cargan con:

*with open('cluster\_label\_mapping.json') as f:*

*cluster\_label\_mapping = json.load(f)*

Texto

El contenido generado por IA puede ser incorrecto.

**Ilustración 8.** *Titulo. Fuente (Guillermo Martínez, 2025)*

**3. Preprocesamiento de imagen manuscrita**

La imagen de entrada pertenece a una colección de textos antiguos con trazos irregulares. Para extraer los caracteres, se emplea una función personalizada de OpenCV que:

* Convierte a escala de grises.
* Aplica **umbralización adaptativa** con THRESH\_OTSU.
* Filtra contornos por tamaño y proporción de aspecto para descartar ruido.
* Centra cada carácter detectado en un lienzo de 28×28 píxeles, reescalándolo proporcionalmente.

def extract\_characters\_from\_image(image\_path): img = cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE) \_, thresh = cv2.threshold(img, 0, 255, cv2.THRESH\_BINARY\_INV + cv2.THRESH\_OTSU) contours, \_ = cv2.findContours(thresh, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE) ... return chars

Se aplica esta función sobre todas las imágenes de una carpeta (Imagenes\_Pares) para construir el conjunto de caracteres a analizar.

carpeta = "Imagenes\_Pares" chars = extract\_characters\_from\_folder(carpeta) X = np.array([char.flatten().astype("float32") / 255.0 for char in chars])

carpeta = "Imagenes\_Pares" chars = extract\_characters\_from\_folder(carpeta) X = np.array([char.flatten().astype("float32") / 255.0 for char in chars])

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**Ilustración 9.** *Titulo. Fuente (Guillermo Martínez, 2025)*

**4. Asignación de clústeres a los segmentos**

Cada carácter segmentado es proyectado al espacio latente utilizando el encoder del autoencoder previamente entrenado. Posteriormente, se asigna al clúster más cercano utilizando el modelo de K-Means y se consulta el diccionario de etiquetas.

latent = encoder.predict(X) preds = kmeans.predict(latent) text = ''.join([cluster\_label\_mapping.get(str(c), '?') for c in preds])

Debido a la complejidad de los caracteres antiguos, algunos vectores latentes caen en clústeres indefinidos o de baja confianza, lo que repercute en el rendimiento del sistema. Aun así, se consigue una predicción útil en muchos casos.

Código QR

El contenido generado por IA puede ser incorrecto.

**Ilustración 10.** *Titulo. Fuente (Guillermo Martínez, 2025)*

**5. Reconstrucción final del texto**

Finalmente, se reconstruye el texto a partir de las etiquetas asignadas. El proceso de ordenación por posición horizontal se mantiene, aunque en este caso algunas letras pueden estar incompletas o superpuestas en la imagen original, afectando ligeramente el orden final.

El texto resultante conserva en gran parte el contenido semántico, aunque se detectan errores en la clasificación de letras con trazos similares o poco definidos.

(imagen)

Este segundo caso de uso demuestra que el sistema, aunque entrenado sobre un dataset estándar como EMNIST, es capaz de adaptarse parcialmente a condiciones reales de manuscrito histórico, gracias a la segmentación personalizada, el uso del espacio latente comprimido y el etiquetado asistido por agrupamiento.

Aunque el rendimiento decrece respecto a casos más limpios, el sistema sigue produciendo resultados funcionales y representa una base sólida para futuras extensiones orientadas a la paleografía y la transcripción de archivos históricos.

# Resultados y discusión

Los resultados obtenidos en el proyecto se analizaron según las principales fases del sistema OCR implementado. Esta evaluación permite identificar con claridad las fortalezas del enfoque propuesto, así como las limitaciones detectadas en escenarios más exigentes. A continuación, se detallan los principales hallazgos por etapa del sistema.

## Codificación latente y agrupamiento

La fase de entrenamiento del autoencoder ha demostrado ser efectiva para extraer representaciones latentes significativas a partir de imágenes de caracteres manuscritos. Estas representaciones han facilitado una buena separación en el espacio latente, lo que ha permitido una agrupación coherente de caracteres similares mediante clustering con K-Means.

Esta estructura ha funcionado de manera sólida tanto en casos con escritura clara como en entornos con ruido visual o trazos deformados, aunque con menor cohesión en estos últimos. Aun así, se confirmó que el modelo puede capturar características visuales esenciales sin etiquetas previas.

## Etiquetado y clasificación

La estrategia de etiquetado asistido por clústeres ha resultado eficiente y ha permitido reducir significativamente el esfuerzo de anotación manual. En entornos controlados, este método ha generado conjuntos etiquetados consistentes y útiles para el entrenamiento del clasificador final.

Sin embargo, en escenarios donde la caligrafía manuscrita presenta alta variabilidad o estilos poco representados en el conjunto de entrenamiento, el rendimiento del clasificador se ha visto afectado. En estos casos, el sistema ha mostrado dificultades para generalizar, produciendo predicciones inconsistentes o incorrectas, incluso cuando el agrupamiento era visualmente razonable.

## Reconstrucción del texto

La reconstrucción del texto a partir de las predicciones de caracteres ha sido satisfactoria en casos donde el sistema ha podido clasificar correctamente la mayoría de los segmentos. Se ha logrado mantener el orden y la estructura de las frases originales, incluyendo los espacios.

No obstante, cuando la clasificación ha fallado, la reconstrucción también se ha visto comprometida. Esto ha sido especialmente evidente en entornos con escritura histórica o caligráficamente compleja, donde la salida del sistema no ha logrado formar secuencias comprensibles ni palabras coherentes.

## Discusión y valoración general

Los resultados permiten afirmar que se han alcanzado con éxito los objetivos principales del proyecto en lo que respecta a la validación del enfoque: se ha demostrado que es posible desarrollar un sistema OCR funcional y modular que no depende de datos etiquetados previamente. El sistema ha sido capaz de segmentar, codificar, agrupar, etiquetar y reconstruir texto manuscrito en condiciones controladas.

Asimismo, se ha dado respuesta a la necesidad detectada inicialmente: ofrecer una alternativa viable para digitalizar escritura manuscrita sin grandes esfuerzos de anotación. El enfoque basado en agrupamiento latente y etiquetado semiautomático permite una implementación flexible y extensible.

Sin embargo, también se ha identificado una limitación clave: el sistema pierde capacidad de generalización cuando se enfrenta a caligrafías nuevas o estilos no contemplados en el conjunto inicial. Esto impide, por el momento, su aplicación directa sobre documentos históricos reales sin una fase previa de entrenamiento adaptativo.

En conclusión, los resultados son prometedores en condiciones controladas y sientan las bases para extender el sistema a contextos más complejos. No obstante, será necesario reforzar el modelo con ejemplos caligráficos variados o incorporar técnicas más avanzadas de aprendizaje para abordar de forma efectiva la transcripción de documentos antiguos.

# Conclusiones

El presente Trabajo de Fin de Grado ha abordado con éxito el diseño e implementación de un sistema OCR orientado al reconocimiento de caracteres manuscritos sin necesidad de datos previamente etiquetados. A partir de la combinación de técnicas de codificación latente, agrupamiento automático y clasificación supervisada, se ha construido una arquitectura modular capaz de segmentar imágenes manuscritas, identificar caracteres individuales y reconstruir frases completas en formato digital.

Entre los principales logros cabe destacar la implementación de un pipeline funcional que:

* Evita la dependencia de corpus anotados.
* Permite una segmentación flexible de caracteres a partir de imágenes reales.
* Utiliza agrupamiento en el espacio latente como herramienta para reducir la complejidad del etiquetado.
* Reconstruye texto a partir de secuencias de predicciones ordenadas espacialmente.

Los resultados obtenidos confirman la validez del enfoque en contextos controlados, especialmente cuando el estilo caligráfico de entrada es consistente con el conjunto de entrenamiento. En estas condiciones, el sistema ha mostrado una buena precisión en la clasificación y una reconstrucción coherente del contenido textual.

Sin embargo, también se ha identificado una limitación relevante: la pérdida de precisión en la clasificación cuando el sistema se enfrenta a caligrafías con gran variabilidad o con trazos que no han sido representados en el entrenamiento. Esta limitación reduce la aplicabilidad directa del modelo sobre documentos históricos reales, como los del Archivo General de Indias, y pone de manifiesto la necesidad de extender el sistema para adaptarse a una mayor diversidad caligráfica.

En conjunto, los objetivos definidos al inicio del trabajo han sido alcanzados en su mayoría. Se ha demostrado la viabilidad técnica del sistema propuesto, se ha validado su funcionamiento en distintos escenarios, y se ha proporcionado una solución que responde a la necesidad de digitalizar textos manuscritos sin transcripción previa. No obstante, se abre una línea clara de mejora para garantizar su robustez en condiciones más exigentes.

## Valoración personal

Desde una perspectiva personal, el desarrollo de este proyecto ha representado un reto intelectual y técnico muy enriquecedor. El planteamiento inicial de construir un OCR sin supervisión directa supuso una dificultad añadida respecto a enfoques tradicionales, pero permitió profundizar en conceptos como representación latente, agrupamiento, codificación visual y procesamiento de imágenes a bajo nivel.

La implementación progresiva de las distintas fases, el análisis de errores, y la validación de hipótesis en entornos reales han permitido una comprensión más profunda del aprendizaje automático aplicado a datos visuales. Además, el hecho de trabajar con imágenes manuscritas reales ha acercado el proyecto a un contexto histórico y patrimonial que le otorga un valor social y cultural adicional.

Aunque el sistema no alcanza aún el nivel de precisión deseado en contextos históricos, considero que se han sentado unas bases sólidas para seguir mejorándolo. El enfoque modular, replicable y extensible del sistema facilita su evolución futura, tanto en términos técnicos como aplicados.

## Líneas futuras

A partir de las conclusiones extraídas y de las limitaciones observadas, se identifican varias líneas futuras de desarrollo que podrían mejorar notablemente el rendimiento y aplicabilidad del sistema:

* **Ampliación del conjunto de entrenamiento con múltiples estilos caligráficos:**  
  Incluir ejemplos de distintos autores, épocas y condiciones de escritura permitiría al modelo aprender patrones más diversos y mejorar su capacidad de generalización.
* **Aplicación de técnicas de aprendizaje auto-supervisado o contrastivo:**  
  En lugar de basarse exclusivamente en agrupamientos manuales, podrían emplearse métodos más avanzados para extraer relaciones visuales entre caracteres sin etiquetas explícitas.
* **Entrenamiento incremental adaptado al documento:**  
  Permitir al sistema ajustar su clasificación sobre la marcha a partir de correcciones o ejemplos adicionales tomados del mismo manuscrito.
* **Integración de modelos de lenguaje:**  
  Incorporar un modelo lingüístico como posprocesamiento ayudaría a corregir predicciones erróneas y reconstruir frases más coherentes desde el punto de vista semántico.

# Bibliografía

1. Ministerio de Cultura. (s.f.). Portal de Archivos Españoles (PARES). [https://pares.culturaydeporte.gob.es](https://pares.culturaydeporte.gob.es/)
2. Mistral AI. (2024, abril 12). Mistral OCR: advancing handwritten text recognition. <https://mistral.ai/news/mistral-ocr>
3. OpenAI. (s.f.). Images and vision. <https://platform.openai.com/docs/guides/images-vision?api-mode=responses>
4. Colutto, S., Kahle, P., Hackl, G., & otros. (2019). Transkribus: A platform for automated text recognition and searching of historical documents. En 2019 15th International Conference on eScience (eScience) (pp. 463–466). IEEE. <https://doi.org/10.1109/eScience.2019.00060>
5. Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. arXiv. <https://arxiv.org/abs/1606.05157>
6. Kahle, P., Colutto, S., Hackl, G., & Mühlberger, G. (2017). Transkribus: A service platform for transcription, recognition and retrieval of historical documents. En 14th IAPR International Conference on Document Analysis and Recognition (ICDAR) (pp. 19–24). IEEE. <https://doi.org/10.1109/ICDAR.2017.307>
7. Graves, A. (2012). Connectionist temporal classification. En Supervised Sequence Labelling with Recurrent Neural Networks (pp. 61–93). Springer, Berlin, Heidelberg. <https://doi.org/10.1007/978-3-642-24797-2_7>

# ANEXOS

[Se deben indicar aspectos que no se han podido incluir en la memoria principal dadas sus dimensiones. Por ejemplo, en anexos de debería incluir el código implementado, manuales de usuario, manuales de instalación, resultados adicionales, etc.]