ISSN 1688-2806


Universidad de la República
Facultad de Ingeniería


# Clasificación del Archivo Berrutti

Tesis presentada a la Facultad de Ingeniería de la
Universidad de la República por

Damián Pintos

en cumplimiento parcial de los requerimientos
para la obtención del título de
Maestría en Ciencia de Datos y Aprendizaje Automático.

Directores de Tesis

Ignacio Ramírez ..... Universidad de la República
Gregory Randall ..... Universidad de la República

Tribunal

Lorena Etcheverry ..... Universidad de la República
Javier Preciozzi ..... Universidad de la República
Elina Gómez ..... Universidad de la República

Director Académico

Javier Baliosián ..... Universidad de la República

Montevideo
July 2025

*Clasificación del Archivo Berrutti*, Damián Pintos.

ISSN 1688-2806

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 74 páginas.
Compilada el martes 25 noviembre, 2025.
http://iie.fing.edu.uy/

Programa de Posgrado en Ciencia de Datos y Aprendizaje Automático
Facultad de Ingeniería
Universidad de la República
Montevideo – Uruguay

Esta página ha sido intencionalmente dejada en blanco.

# Agradecimientos

A Gaby, Rosita y Nelson, por estar siempre.

A todas las personas que forman parte del grupo Cruzar, por su compromiso, su trabajo colectivo y su convicción en la construcción de memoria.

A quienes, desde distintos espacios, sostienen las banderas de verdad y justicia, y mantienen viva la búsqueda que da sentido a este trabajo.

Finalmente, a la Universidad de la República, por ser el espacio público que hace posible estudiar, investigar y aportar al conocimiento con libertad y compromiso con la sociedad a la que pertenece.

Esta página ha sido intencionalmente dejada en blanco.

# Resumen

Este trabajo presenta una investigación sobre la clasificación automática de documentos pertenecientes al llamado Archivo Berrutti, del cual tenemos un conjunto de aproximadamente 2,2 millones de imágenes escaneadas de microfilmaciones producidas por organismos represivos del Estado uruguayo en un periodo que incluye la dictadura cívico-militar (1973–1985). El objetivo principal es evaluar distintas metodologías de aprendizaje automático aplicadas a este corpus, con énfasis en modelos basados en texto, en imágenes y en enfoques híbridos que combinan ambas modalidades. Para ello, se emplean arquitecturas modernas como *BERT* (Bidirectional Encoder Representations from Transformers) y *EfficientNet*, adaptadas a las particularidades del dominio y entrenadas sobre subconjuntos representativos.

Los resultados muestran que los modelos híbridos logran un desempeño superior en comparación con los enfoques unimodales, alcanzando precisiones cercanas al 99 % en tareas de clasificación multiclase, con los conjuntos de datos utilizados para esta evaluación. Estos hallazgos aportan evidencia sobre la utilidad de los modelos multimodales en escenarios donde los datos presentan variabilidad en calidad y estructura. Finalmente, se discuten las implicancias de esta investigación en el marco de los estudios de derechos humanos, destacando el potencial de las técnicas de aprendizaje automático para facilitar la organización y el análisis de archivos históricos complejos.

Esta página ha sido intencionalmente dejada en blanco.

# Tabla de contenidos

Tabla de contenidos

# Índice de figuras

Esta página ha sido intencionalmente dejada en blanco.

# Índice de tablas

Esta página ha sido intencionalmente dejada en blanco.

# Capítulo 1

# Introducción

En 1973 inició en Uruguay una dictadura cívico-militar que articuló con otras dictaduras de la región en el marco del Plan Cóndor. La dictadura en Uruguay terminó en 1985 y durante ese período los órganos represivos del Estado cometieron violaciones a los derechos humanos tales como secuestros, torturas, violaciones, asesinatos y desapariciones forzadas. La complicidad y el silencio de militares y civiles se mantienen hasta hoy, lo que dificulta la investigación de los crímenes y la búsqueda de las personas desaparecidas [1].

Antes (particularmente desde 1968), durante y después de la dictadura, los órganos represivos del Estado generaron documentación relacionada tanto con sus procedimientos internos de trabajo como con la investigación de civiles. Parte de esa documentación fue incautada y se la conoce como el Archivo Berrutti.

El llamado Archivo Berrutti es una colección heterogénea compuesta por cerca de 1500 rollos de microfilm, con cientos de imágenes cada uno, que contienen documentos de muy diverso tipo generados por distintos órganos represivos del Estado durante y después de la dictadura. A pesar de su denominación, no cuenta con una organización archivística formal al estilo de un archivo institucional, sino que está en proceso de sistematización y digitalización, lo que genera desafíos para su ordenamiento, descripción y búsqueda eficiente [2]. Se trata de alrededor de 2,2 millones de imágenes escaneadas de microfilmaciones [1]. Las imágenes representan documentos de diversas fuentes dentro de los organismos represores a lo largo de todo el período, y su calidad también es heterogénea: se pueden encontrar imágenes de muy buena calidad y otras directamente ilegibles. Para cada documento se tiene, además de la imagen, una aproximación al texto que contiene a partir de la utilización del OCR *Tesseract* [3] y/o del OCR *Calamari* [4] sobre la imagen [5].

Lograr categorizar los documentos de la colección es importante tanto para su uso directo como para facilitar otros trabajos de digitalización. Por un lado, una buena categorización facilita la búsqueda de documentos a partir de filtros. La cantidad de documentos es muy grande y para una investigación puede ser de interés, por ejemplo, limitar la búsqueda a los recortes de prensa o las fichas personales. A su vez, en otros trabajos de digitalización resulta relevante realizar una separación preliminar de los documentos. Por ejemplo, es de interés entrenar un OCR de manera diferenciada para aquellos organizados en una sola columna y para los que presentan dos o más columnas. En el caso particular de las fichas, dentro del proyecto ya se han obtenido resultados alentadores para su procesamiento, siempre que las hojas hayan sido previamente identificadas como tales [6]. La clasificación también es importante para la discusión sobre la divulgación de los documentos: no es lo mismo publicar sin un proceso cuidado

de anonimización las actas de interrogatorio que hacerlo para los recortes de prensa.

El objetivo de este trabajo es encontrar una metodología eficiente para clasificar documentos. La categorización total de la colección es un problema no resuelto todavía y los usos de una buena clasificación son amplios. Entonces la metodología intentará abarcar tanto una clasificación binaria (determinar si un documento es o no de un determinado tipo) como una clasificación múltiple donde se determinará si el documento pertenece a una entre varias clases o a ninguna de ellas.

## 1.1. Trabajos previos

En la literatura reciente existen varios trabajos que abordan la clasificación automática de documentos escaneados utilizando enfoques unimodales y multimodales. Por ejemplo, Harley et al. [7] aplicaron redes neuronales convolucionales (CNN, por sus siglas en inglés) para clasificar imágenes de documentos en el conjunto RVL-CDIP [8], mostrando que la estructura visual por sí sola permite alcanzar desempeños competitivos.

En el plano textual, Chen et al. [12] y Shao y Wang [13] evaluaron variantes de BERT para clasificación de documentos, destacando que los modelos preentrenados en corpus generales pueden transferirse eficazmente a colecciones específicas. Por su parte, Xu et al. [14] introdujeron LayoutLM, un modelo que integra texto y la disposición espacial (layout) de los documentos, alcanzando resultados de estado del arte en múltiples benchmarks de clasificación y comprensión documental. Posteriormente, se propusieron variantes como LayoutLMv2 [15] y LayoutLMv3 [16], que integran de manera más profunda la información textual y su disposición espacial, consolidando este paradigma para tareas de clasificación.

Finalmente, los métodos multimodales combinan texto e imagen. Existen dos estrategias principales: la fusión tardía (late fusion), donde se entrenan modelos independientes y luego se combinan sus salidas [17]; y la fusión temprana (early fusion), que integra las representaciones de texto e imagen antes de la clasificación [18]. Ambas variantes se ilustran en la Figura 1.1 y la Figura 1.2. También se han propuesto arquitecturas como BERTgrid [10], que proyecta embeddings de texto sobre una grilla espacial para alinear información textual y visual, y DocFormer [19], que combina ambas modalidades mediante mecanismos de atención cruzada. Afzal et al. [9] extendieron esta idea incorporando características textuales junto con las visuales, logrando mejoras consistentes en escenarios multimodales. Más recientemente, Denk y Reisswig [10] exploraron arquitecturas multimodales con fusión temprana para tareas de clasificación y extracción de información en documentos, confirmando la complementariedad entre texto e imagen. Además, Adhikari et al. [11] presentaron *DocBERT*, una variante de BERT (Bidirectional Encoder Representations from Transformers) adaptada específicamente a la clasificación documental, mostrando que el ajuste fino de modelos de lenguaje general sobre colecciones de documentos escaneados puede superar ampliamente a representaciones tradicionales.

Posteriormente a este trabajo, surgieron modelos que ampliaron el estado del arte en clasificación documental multimodal. DocFormerv2 [20] introdujo mejoras en la atención multimodal respecto a su predecesor, con un diseño más eficiente para capturar relaciones cruzadas entre texto e imagen, lo que se traduce en un mejor desempeño en tareas de clasificación y comprensión documental. DocLLM [21] propuso unificar texto e imagen en un modelo de lenguaje entrenado específicamente para documentos, integrando la semántica textual con la información visual en un mismo espacio representacional, lo que permite abordar tanto clasificación como recuperación y preguntas-respuestas en documentos escaneados. Más recientemente, DocLayLLM [22] presentó una extensión multimodal eficiente de los modelos de lenguaje de gran ta-

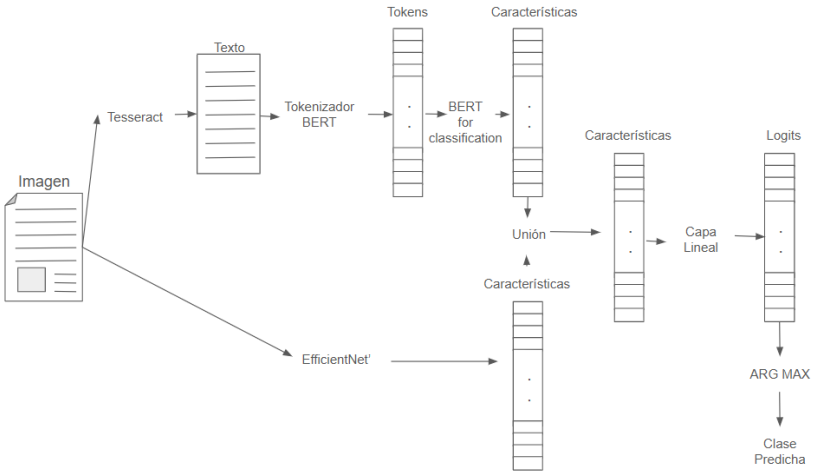


Figura 1.1: Esquema de fusión temprana. Las representaciones de BERT (texto) y EfficientNet (imagen) se concatenan en un único vector, que alimenta una capa de clasificación final. Este diseño busca capturar interacciones entre modalidades, aunque requiere mayor volumen de datos y presenta desafíos de dimensionalidad.

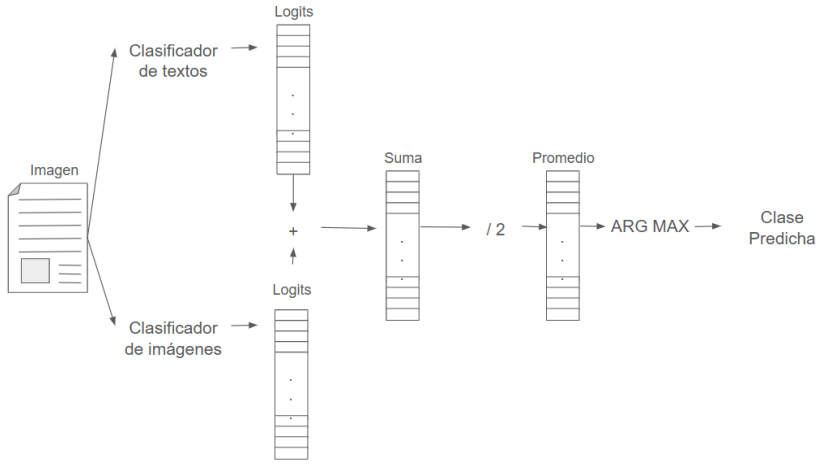


Figura 1.2: Esquema de fusión tardía. Se entrenan de manera independiente un modelo textual (BERT) y uno visual (EfficientNet). Las probabilidades de salida de ambos modelos se combinan mediante un promedio simple para generar la predicción final. Esta estrategia resultó la más efectiva en este trabajo.

maño, optimizada para procesar documentos con estructuras complejas que combinan texto, imágenes, tablas y elementos gráficos. Finalmente, Docopilot [23] exploró la comprensión documental a nivel de página completa, proponiendo una arquitectura orientada a mejorar la coordinación entre modalidades mediante mecanismos avanzados de atención cruzada. Estos modelos no fueron implementados en esta tesis, pero ilustran con claridad la evolución más reciente del campo y marcan el camino hacia sistemas más robustos y generalistas para el análisis automático de grandes colecciones documentales.

## 1.2. Contribuciones de este trabajo

Esta tesis aporta al estudio de la clasificación automática de documentos históricos escaneados a partir de una exploración sistemática de enfoques unimodales y multimodales. Se evaluaron modelos basados exclusivamente en texto, empleando BERT para aprovechar la semántica de los documentos, y modelos basados en imágenes con EfficientNet [29], capaces de extraer patrones visuales como tipografía y diagramación. A partir de estos experimentos también se construyeron modelos híbridos mediante fusión temprana y fusión tardía, lo que permitió contrastar la capacidad de cada modalidad y analizar sus ventajas en combinación.

Una contribución central de este trabajo es la adaptación de estas metodologías al Archivo Berrutti, una colección de más de dos millones de imágenes digitalizadas que documenta las violaciones a los derechos humanos cometidas durante la dictadura uruguaya. El archivo carece de una organización sistemática y presenta una calidad heterogénea en sus documentos, lo que lo convierte en un caso de estudio particularmente desafiante. La aplicación de modelos de aprendizaje profundo en este contexto muestra la viabilidad técnica de abordar colecciones históricas de gran escala con técnicas contemporáneas de inteligencia artificial.

Otro aporte se encuentra en la evaluación comparativa de distintas estrategias de clasificación. Se experimentó con configuraciones binarias para determinar si un documento pertenece o no a una categoría específica, con modelos multiclase y con una variante que introduce una clase "sin categoría" para manejar casos incompletos. Asimismo, se analizó el comportamiento de los modelos híbridos, confirmando que la fusión tardía mediante promedio de salidas resulta especialmente robusta frente a errores unimodales.

El trabajo también discute el rol del tokenizador en contextos históricos. Se probó un reentrenamiento del tokenizador de BERT con documentos del Archivo Berrutti, observándose que en algunos casos el tokenizador original, entrenado sobre corpus generales y mucho más extensos, alcanza un rendimiento superior. Esta observación contribuye a la reflexión metodológica sobre cuándo conviene especializar un modelo y cuándo es suficiente aprovechar recursos preentrenados disponibles.

Desde el punto de vista experimental, se desarrolló un entorno controlado en el Cluster-UY [24], con código modular en PyTorch Lightning y fijación de semillas aleatorias en las librerías relevantes, lo que asegura la reproducibilidad de los experimentos. Este entorno constituye una base reutilizable para investigaciones posteriores en clasificación documental multimodal.

Finalmente, esta tesis contribuye no solo al ámbito académico sino también al social. El trabajo se enmarca en el proyecto de digitalización del Archivo Berrutti, cuyo valor trasciende lo técnico y se vincula directamente con la preservación de la memoria histórica y con los procesos de verdad y justicia en Uruguay. La aplicación de técnicas de inteligencia artificial a esta colección demuestra que la investigación en aprendizaje automático puede ofrecer herramientas concretas para el análisis de archivos vinculados al terrorismo de Estado, aportando a la vez rigor metodológico y utilidad social. Además, la metodología propuesta es extensible: aplicando el mismo enfoque, es posible incorporar progresivamente nuevas categorías a partir de conjuntos de documentos etiquetados, con el objetivo de alcanzar en el futuro la clasificación integral del archivo y asignar a cada documento su categoría correspondiente.

## 1.3. Organización del documento

Este documento se organiza en siete capítulos.

El **Capítulo 1, Introducción**, enmarca el problema, presenta el contexto histórico y social del Archivo Berrutti y plantea los objetivos de la tesis. Se describen las motivaciones del proyecto de digitalización y el valor de una clasificación automatizada para la investigación, la preservación y la accesibilidad del acervo.

El **Capítulo 2, Introducción general sobre clasificación**, ofrece los conceptos de base para el resto del trabajo. Se repasan nociones fundamentales de clasificación supervisada, las diferencias entre enfoques binarios y multiclase, consideraciones sobre métricas (accuracy, matrices de confusión, etc.) y el rol de las decisiones de preprocesamiento, incluyendo la tokenización y la preparación de los datos antes de su utilización en modelos de aprendizaje automático.

El **Capítulo 3, Conjuntos de Datos**, describe las colecciones utilizadas. Se detalla el corpus estándar *Tobacco-3482* (categorías, características y calidad de las imágenes) y el *Archivo Berrutti* (origen, heterogeneidad, OCR disponible y particularidades de su organización). Se explican los criterios de partición en entrenamiento/-validación/test y las implicancias de balanceo y representatividad.

El **Capítulo 4, Evaluación de Métodos de Clasificación**, presenta la metodología y los resultados sobre Tobacco-3482. Se analizan enfoques unimodales: clasificación a partir de textos con *BERT* y clasificación a partir de imágenes con *EfficientNet*. Posteriormente se exploran los enfoques multimodales, comparando la fusión temprana y la fusión tardía (promedio). Se discuten las curvas y la elección de tasas de aprendizaje, la longitud de la secuencia de tokens, el impacto del tokenizador y el desempeño relativo de cada familia de métodos con apoyo en tablas y figuras. Asimismo, se incluye un análisis específico de la categoría `Advertising`, que permitió estudiar en detalle las fortalezas y limitaciones de cada modalidad.

El **Capítulo 5, Clasificación del Archivo Berrutti**, traslada la metodología al caso de estudio local. Se documentan los experimentos binarios por clase, las pruebas multiclase y la variante con "sin categoría" para manejar clasificación incompleta. Se destacan las particularidades de este corpus y los desafíos de aplicar técnicas de clasificación en un archivo heterogéneo, en contraste con los resultados obtenidos en Tobacco-3482.

El **Capítulo 6, Conclusiones y trabajos futuros**, sintetiza los hallazgos principales, extrae lecciones metodológicas (incluida la discusión sobre tokenización) y describe líneas de continuidad: extender la taxonomía incorporando nuevas clases, consolidar la clasificación integral del archivo y robustecer los modelos multimodales en escenarios de calidad variable.

Finalmente, el **Anexo 8** reúne material de apoyo: definiciones y ejemplos de categorías del Archivo Berrutti y detalles complementarios que facilitan la interpretación de resultados y la replicabilidad.

Esta página ha sido intencionalmente dejada en blanco.

# Capítulo 2

# Introducción general sobre clasificación

La clasificación automática de documentos ha sido un área de investigación activa durante varias décadas, motivada tanto por la necesidad de organizar grandes colecciones digitales como por aplicaciones específicas en bibliotecas, archivos y entornos corporativos. Los enfoques propuestos en la literatura abarcan desde técnicas clásicas de representación textual hasta arquitecturas profundas capaces de integrar información multimodal. En esta subsección se presenta una síntesis de esa evolución, destacando los avances más relevantes en métodos unimodales (texto o imagen) y multimodales, así como su relación con los experimentos desarrollados en este trabajo.

Los primeros intentos de clasificación documental se apoyaron en representaciones simples como *bag-of-words* y TF-IDF [25]. Aunque fáciles de implementar, estas técnicas no capturan la semántica ni el orden de las palabras.

Posteriormente, los *word embeddings* como Word2Vec [26] y GloVe [27] representaron un avance al mapear palabras en espacios vectoriales densos, reflejando similitudes semánticas. Sin embargo, estos embeddings son estáticos: una palabra ambigua mantiene la misma representación sin importar el contexto.

El gran salto se dio con los modelos *Transformer*, en particular BERT [28], que captura dependencias contextuales bidireccionales mediante mecanismos de atención. BERT se preentrena en grandes corpus y luego se ajusta (*fine-tuning*) para tareas específicas, alcanzando resultados de estado del arte en clasificación textual. Esta arquitectura se esquematiza en la Figura 4.1, tal como se implementó en este trabajo.

En el plano visual, los documentos contienen estructura gráfica (formato, tipografía, diagramación). Las redes neuronales convolucionales (CNN, por sus siglas en inglés) constituyen un tipo de arquitectura especializada en procesar datos con estructura espacial, como imágenes. Su principio básico consiste en aplicar filtros (convoluciones) que detectan patrones locales —por ejemplo, bordes o texturas— y combinarlos jerárquicamente para capturar representaciones de mayor nivel.

EfficientNet [29] se distingue entre las CNN modernas por su estrategia de escalamiento compuesto (profundidad, ancho y resolución), logrando alta precisión con menor número de parámetros. Su esquema general se muestra en la Figura 4.4.

Además de los modelos empleados en este trabajo, existen variantes ampliamente utilizadas en la literatura. En visión por computadora, arquitecturas como ResNet [30] o Inception [31] han sido referencia durante años por su capacidad de aprendizaje jerárquico de características. En procesamiento de lenguaje natural, modelos derivados de BERT como RoBERTa [32] o DistilBERT [33] han demostrado que ajustes en

el preentrenamiento o simplificaciones arquitectónicas pueden mejorar la eficiencia manteniendo un rendimiento competitivo. En pruebas preliminares realizadas con el conjunto de datos del Archivo Berrutti, se experimentó con distintas arquitecturas convolucionales y el modelo que mostró mejor desempeño fue EfficientNet, lo que motivó su selección definitiva para los experimentos desarrollados en esta tesis. Aunque no se implementaron todas estas variantes en detalle, su mención resulta relevante para situar este trabajo en el panorama actual de investigación.

Estos antecedentes muestran que la clasificación documental ha evolucionado desde enfoques unimodales hasta modelos multimodales que integran de forma más estrecha las modalidades disponibles. El presente trabajo se enmarca en esa línea: utiliza BERT y EfficientNet como representaciones unimodales robustas y explora tanto esquemas de fusión tardía como temprana para evaluar su efectividad en la colección Tobacco-3482 y en el Archivo Berrutti.

# Capítulo 3

# Conjuntos de Datos

En este trabajo se utilizaron dos conjuntos de datos principales. Por un lado, el *Archivo Berrutti*, que constituye el corpus documental de interés específico para esta investigación. Por otro lado, se empleó el conjunto *Tobacco-3482* [34] como base de referencia para validar metodologías de clasificación y obtener métricas comparables. La utilización del conjunto Tobacco-3482 precedió al trabajo con el *Archivo Berrutti*, permitiendo revisar la bibliografía relevante e implementar versiones preliminares de los métodos de clasificación sobre un conjunto conocido, con resultados esperables y previamente reportados.

## 3.1. Tobacco-3482

El conjunto de datos *Tobacco-3482* [34] está compuesto por 3.482 documentos digitalizados y etiquetados pertenecientes a la industria tabacalera. Su principal valor radica en que constituye un corpus ampliamente utilizado en la literatura como referencia para la clasificación automática de documentos, lo que permite comparar el desempeño de distintos enfoques con resultados previamente reportados. Para este trabajo, se optó por dividir el conjunto en tres particiones: 80 % para entrenamiento, 10 % para validación y 10 % para test, siguiendo la práctica común en los estudios previos.

Una característica relevante del Tobacco-3482 es que sus documentos están organizados en diez categorías, las cuales corresponden a una clasificación definida de acuerdo con el contenido y la función del texto en el ámbito corporativo, según se muestra en la tabla 3.1. Estas categorías representan tipos de documentos reconocibles (cartas, informes, formularios, correos electrónicos, etc.), que resultan comparables con las tipologías presentes en archivos institucionales más amplios, como el *Archivo Berrutti*. En este sentido, aunque los temas específicos difieren, la estructura de los documentos comparte similitudes con los documentos que integran nuestro corpus principal, lo que hace de Tobacco-3482 una base válida para probar metodologías de clasificación. La posibilidad de replicar experimentos sobre un conjunto bien caracterizado y contrastar los resultados con la bibliografía existente aporta robustez al análisis y permite evaluar la transferibilidad de los métodos hacia el caso de estudio.

### 3.1.1. Descripción de las Categorías

Las categorías fueron definidas originalmente por los compiladores del conjunto a partir de los tipos de documentos más frecuentes en los archivos corporativos de

Tabla 3.1: Distribución de documentos por categoría en el conjunto de datos Tobacco-3482.

| Categoría | Nombre | Cantidad |
|---|---|---|
| 0 | Advertising | 230 |
| 1 | Email | 599 |
| 2 | Form | 431 |
| 3 | Letter | 567 |
| 4 | Memorandum | 620 |
| 5 | News | 188 |
| 6 | Note | 201 |
| 7 | Report | 265 |
| 8 | Resume | 120 |
| 9 | Scientific | 261 |

la industria tabacalera. A continuación, se presentan las principales características de cada una:

- **Advertising**: documentos relacionados con campañas publicitarias, incluyendo anuncios, planes de marketing y estrategias de promoción.
- **Email**: correspondencia electrónica interna de la industria tabacalera.
- **Form**: formularios utilizados en los procesos administrativos y comerciales.
- **Letter**: cartas formales e informales, tanto internas como externas.
- **Memorandum**: memorandos que documentan comunicaciones internas y registros de gestión.
- **News**: artículos periodísticos, recortes de prensa y otros documentos de carácter informativo.
- **Note**: anotaciones breves, comentarios o apuntes generados en el marco del trabajo diario.
- **Report**: informes internos u oficiales relacionados con actividades, investigaciones o análisis.
- **Resume**: documentos que recopilan antecedentes laborales o perfiles profesionales.
- **Scientific**: publicaciones o informes técnicos de carácter científico.

La Figura 3.1 muestra un ejemplo visual de cada una de las categorías, ilustrando la diversidad de estructuras y estilos presentes en el conjunto Tobacco-3482.

## 3.2. Archivo Berrutti

Para entrenar modelos de clasificación documental es necesario contar con imágenes, texto y etiquetas correspondientes a los diferentes tipos de documentos que conforman la colección. En una primera etapa se trabajó con un conjunto de 51.726 documentos previamente etiquetados en la Facultad de Información y Comunicación mediante el software *labelme* [5]. Este conjunto incluía 19 categorías y 73 subcategorías; sin embargo, presentaba limitaciones significativas, tales como una cobertura incompleta de algunos tipos de documentos (por ejemplo, ausencia de etiquetas para hojas en blanco) y una alta tasa de error de etiquetado, lo que motivó su descarte para las etapas posteriores del análisis.

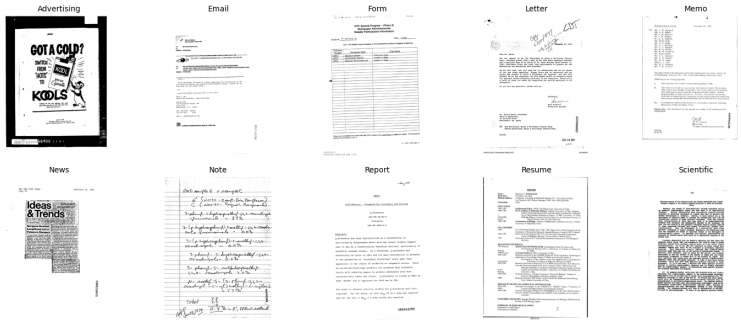


Figura 3.1: Ejemplos representativos de las diez categorías del conjunto Tobacco-3482: Advertising, Email, Form, Letter, Memorandum, News, Note, Report, Resume y Scientific.

El llamado Archivo Berrutti está formado por rollos numerados, almacenados en carpetas que contienen las imágenes correspondientes. En general, un mismo rollo puede incluir documentos de dos o más tipos, lo que dificulta su utilización directa para el entrenamiento de modelos de clasificación. Para esta tesis se optó por seleccionar únicamente aquellos rollos en los que predominaba un tipo de documento, de modo de facilitar el proceso de curado y la construcción de subconjuntos consistentes. Una vez seleccionados, los rollos fueron depurados manualmente para eliminar carátulas, hojas en blanco y documentos que no correspondieran a la categoría mayoritaria. Este procedimiento permitió conformar cuatro subconjuntos que fueron utilizados para entrenar modelos de clasificación, los cuales se describen brevemente a continuación.

### 3.2.1. Fichas de exfuncionarios de AFE

Entre los rollos 930 y 935 se identificaron 12.925 documentos correspondientes a fichas de exfuncionarios de la Administración de Ferrocarriles del Estado (AFE). Estos documentos se dividieron en 10.322 para entrenamiento, 1.305 para prueba y 1.298 para validación. El interés de esta categoría radica en su potencial para continuar el proceso de sistematización de la información contenida en fichas, dado que en el Archivo Berrutti existen numerosas secciones formadas por fichas de naturaleza similar pero con características específicas según el organismo que las coleccionó. El trabajo con las fichas es un subproblema específico que fue abordado para el caso de las fichas de la Oficina Coordinadora de Operaciones Antisubversivas (OCOA) en otro trabajo [6].

Las fichas presentan una estructura visual regular que facilita su identificación tanto manual como automatizada. En la Figura 3.2 se muestra un ejemplo, donde se observa un diseño estandarizado y un nivel aceptable de legibilidad del texto extraído mediante técnicas de reconocimiento óptico de caracteres (OCR). Estas características las convierten en un candidato adecuado para pruebas de clasificación automática basadas en una combinación de características visuales y texto.

### 3.2.2. Actas de interrogatorios de la OCOA

En los rollos 724, 808, 815, 834 y 841 se encontraron 12.308 actas correspondientes a interrogatorios llevados a cabo por la OCOA. Estos documentos fueron divididos en 9.838 para entrenamiento, 1.238 para prueba y 1.238 para validación.

Dada su naturaleza sensible, la correcta identificación de este tipo de documento resulta prioritaria, tanto para procesos eventuales de anonimización de datos de víctimas como para el desarrollo de herramientas automáticas que permitan su clasificación

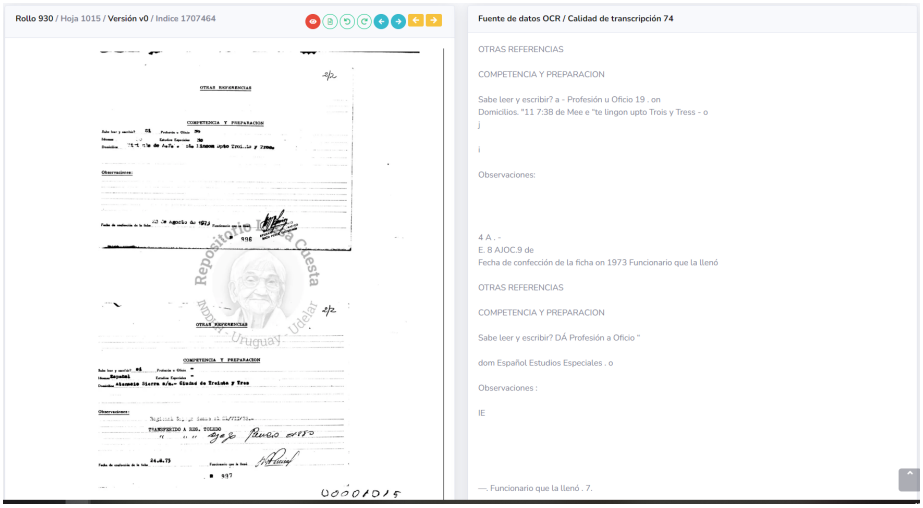


Figura 3.2: Ejemplo de ficha correspondiente a exfuncionarios de AFE El documento presenta un formato estructurado que favorece su reconocimiento a partir de la imagen. El texto extraído mediante OCR resulta en general legible, y se identifican patrones visuales consistentes que permiten discriminar este tipo de documento de otros presentes en la colección.

confiable. A diferencia de otras categorías, las actas de interrogatorio presentan una apariencia visual similar a la de cartas u otros documentos administrativos, lo que dificulta su detección basada exclusivamente en características visuales. Sin embargo, su contenido textual ofrece indicios relevantes para su identificación. En la Figura 3.3 se presenta un ejemplo representativo de este tipo de documento.

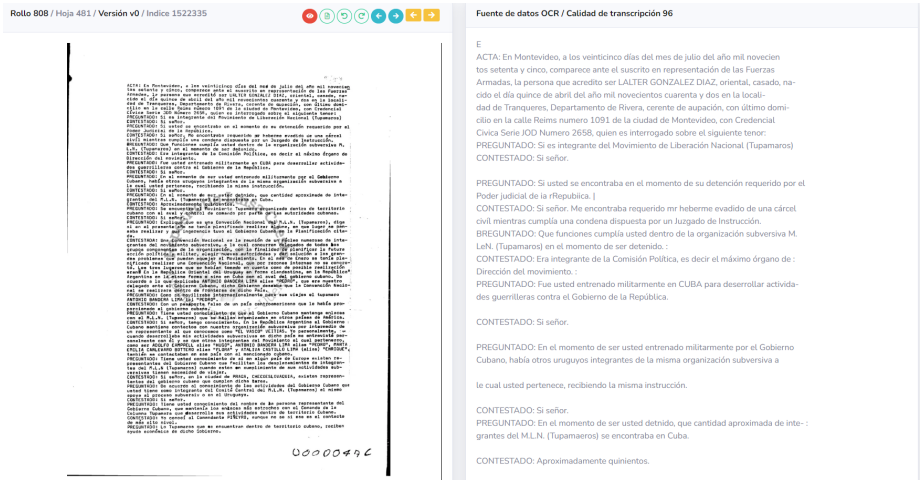


Figura 3.3: Ejemplo de acta de la OCOA Desde el punto de vista visual, este tipo de documento puede confundirse con cartas u otros textos administrativos. No obstante, su contenido textual contiene indicios clave —como formas de hacer referencia a quien pregunta y a quien responde— que permiten su identificación como acta.

### 3.2.3. Fichas de docentes del Consejo Nacional de Educación

En los rollos 1052 al 1055, 1058 y 1059 se identificaron 14.708 documentos correspondientes a fichas de docentes del Consejo Nacional de Educación. Estos documentos se dividieron en 11.762 para entrenamiento, 1.472 para prueba y 1.474 para validación.

Estas fichas presentan un formato preimpreso que facilita su detección a través de métodos basados en imagen. Además, el texto extraído mediante OCR es de buena calidad, lo que permite identificar este tipo de documento también a partir del contenido textual. Un ejemplo puede observarse en la Figura 3.4.

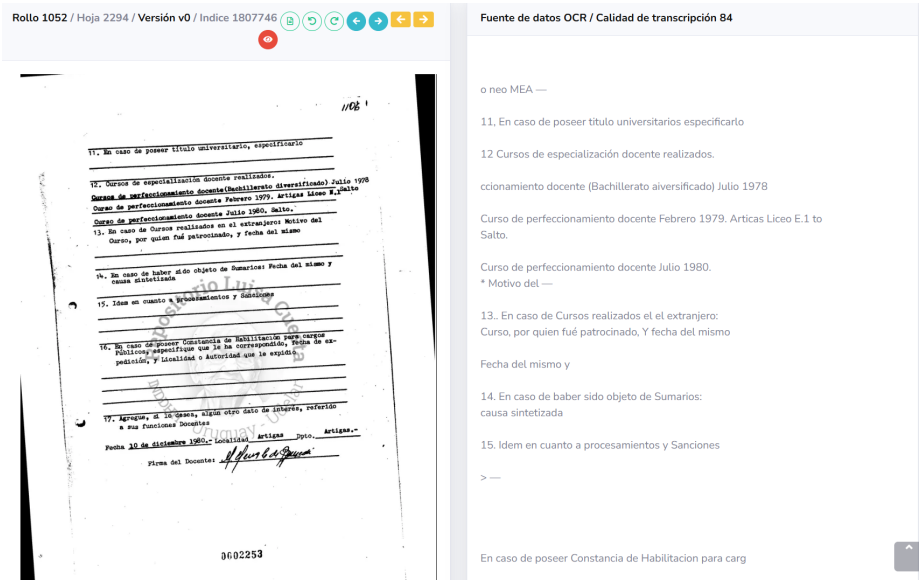


Figura 3.4: Ejemplo de ficha correspondiente a docentes del Consejo Nacional de Educación. Este tipo de documento presenta un formato preimpreso que favorece su reconocimiento visual, y el texto extraído mediante OCR es suficientemente claro como para permitir su clasificación basándose únicamente en el contenido textual.

### 3.2.4. Fichas de la Unión de Jóvenes Comunistas (U.J.C.)

Finalmente, en los rollos 101 al 103 se identificaron 3.422 documentos correspondientes a fichas de la Unión de Jóvenes Comunistas (U.J.C.), que dividimos en 2.732 para entrenamiento, 346 para prueba y 340 para validación.

Este tipo de documento presenta particularidades que dificultan su procesamiento automático. En primer lugar, las imágenes están rotadas 90 grados, lo que en ocasiones dificulta obtener una salida significativa mediante OCR. Además, muchos de los documentos se encuentran en mal estado de conservación. No obstante, al tratarse de formularios basados en un diseño preimpreso, es posible ser optimista respecto a su clasificación utilizando exclusivamente características visuales. Un ejemplo representativo se muestra en la Figura 3.5.

### 3.2.5. Resumen de las categorías utilizadas

Como se muestra en la Tabla 3.2, las cuatro categorías seleccionadas del Archivo Berrutti presentan tamaños desiguales, lo cual constituye un desafío metodológico

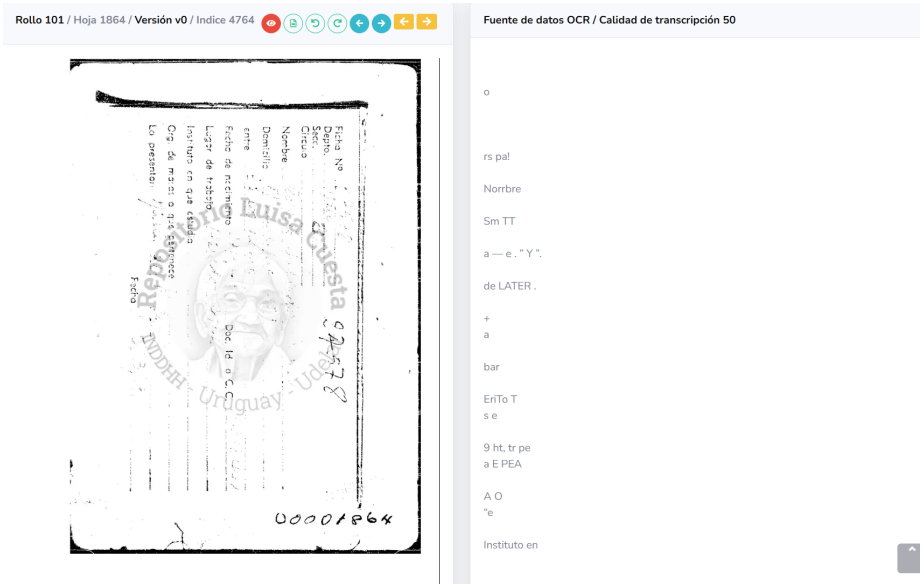


Figura 3.5: Ejemplo de ficha correspondiente a la Unión de Jóvenes Comunistas (U.J.C.). La imagen se encuentra rotada y el estado del documento es deficiente, lo que dificulta obtener una salida útil mediante OCR. Sin embargo, su formato preimpreso regular permite considerar la viabilidad de su clasificación utilizando únicamente la información visual.

relevante. La partición en conjuntos de entrenamiento, validación y prueba fue definida manualmente, a partir de una selección cuidadosa de documentos, con el objetivo de garantizar la calidad de las etiquetas y evitar solapamientos entre particiones.

Esta tabla cumple además el propósito de documentar de forma explícita el corpus empleado, facilitando la reproducibilidad de los experimentos y la comparación con trabajos futuros.

Tabla 3.2: Resumen de categorías y particiones del Archivo Berrutti

| **Categoría** | **Entrenamiento** | **Validación** | **Prueba** |
|---|---|---|---|
| Fichas de A.F.E. | 10.322 | 1.298 | 1.305 |
| Actas de O.C.O.A. | 9.838 | 1.238 | 1.238 |
| Fichas de docentes | 11.762 | 1.474 | 1.472 |
| Fichas de la UJC | 2.732 | 340 | 346 |

# Capítulo 4

# Evaluación de Métodos de Clasificación

En este capítulo se presentan las condiciones bajo las cuales se realizaron los experimentos, un repaso del estado del arte en clasificación de documentos y, posteriormente, la implementación y evaluación de diferentes metodologías aplicadas al conjunto de datos Tobacco-3482. Se incluyen tanto métodos basados en texto como en imágenes, así como enfoques híbridos que combinan ambas modalidades. Finalmente, se discuten los resultados obtenidos.

## 4.1. Entorno

Los experimentos fueron implementados en el lenguaje Python, utilizando como marco principal la librería PyTorch. Para facilitar la organización del código y minimizar la reescritura entre diferentes modelos, se empleó *PyTorch Lightning*, que provee una estructura modular y separa de manera clara las etapas de definición del modelo, entrenamiento y validación.

Todos los modelos se entrenaron y evaluaron en un mismo entorno de trabajo, lo que permitió mantener condiciones controladas y comparables entre experimentos. La infraestructura de hardware utilizada incluyó recursos de *Cluster-UY* [24], el clúster nacional de supercómputo de Uruguay, empleando nodos con GPU NVIDIA y soporte CUDA 12.1. El entorno de cómputo fue configurado con versiones compatibles de los principales paquetes utilizados (*transformers*, *torchmetrics*, entre otros).

En relación con los datos, se trabajó con el conjunto Tobacco-3482, un corpus estándar en tareas de clasificación documental. Este conjunto contiene 10 categorías de documentos escaneados, lo que lo convierte en un recurso adecuado tanto para entrenar como para evaluar la capacidad de generalización de los modelos.

La reproducibilidad de los experimentos se aseguró mediante la fijación de semillas aleatorias en todas las librerías relevantes y el control de versiones del código. Este entorno común constituye la base sobre la cual se compararon las metodologías presentadas en las siguientes subsecciones.

## 4.2. Gestión de datos

En el procesamiento de texto, un *tokenizador* es la herramienta que divide una secuencia de caracteres en unidades mínimas llamadas *tokens*, que pueden corresponder a palabras completas, fragmentos de palabras o símbolos. Este paso es fundamental porque los modelos de lenguaje trabajan con secuencias numéricas que representan tokens y no directamente con texto en bruto.

Los documentos de Tobacco-3482 se procesaron en dos modalidades. Primero, se aplicó OCR con *Tesseract* para extraer el contenido textual. Este texto fue convertido en tokens mediante el tokenizador preentrenado de BERT (longitud fija de 512 tokens, truncado o *padding* con ceros). Por ejemplo:

```
This is an example sentence for tokenization.
```

se transforma en la secuencia `[101, 2023, 2003, 2019, 2742, 6251, 2005, 19204, 1012, 102]`, donde `[101]` y `[102]` marcan el inicio y fin.

Además, se procesaron las imágenes escaneadas, redimensionadas a la resolución de 800 por 800 píxeles requerida por EfficientNet. La estrategia general adoptada es explícitamente multimodal: texto e imagen se utilizan de forma complementaria en los experimentos.

## 4.3. Métodos basados en texto

### Aspectos conceptuales y arquitectura

Una red neuronal se compone de capas, que son funciones que transforman una entrada en salidas intermedias. Una **capa completamente conectada** conecta cada entrada con cada salida mediante un conjunto de pesos.

En tareas de clasificación, la última capa suele generar probabilidades sobre las clases. Para ello se utiliza la función *softmax*, que es una generalización de la sigmoide para múltiples clases. Esta función toma como entrada un vector de valores reales $z = (z_1, z_2, \ldots, z_{|\mathcal{Y}|})$, conocidos como *logits*, y lo transforma en una distribución de probabilidad:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{|\mathcal{Y}|} e^{z_j}}.$$

Aquí, $e^{z_i}$ representa la exponenciación del valor $z_i$, lo que garantiza que todos los términos sean positivos. El denominador corresponde a la suma de todas las exponenciales de los logits, asegurando que la salida esté normalizada y que las probabilidades resultantes sumen 1. Como consecuencia, cada componente $\text{softmax}(z_i)$ se encuentra en el rango $[0, 1]$ y puede interpretarse como la probabilidad de que la instancia pertenezca a la clase $i$.

En BERT, el token especial `[CLS]` actúa como un resumen global del documento, ya que su representación final condensa información del resto de la secuencia. Denotamos dicha representación como un vector $\mathbf{h}_{\texttt{[CLS]}} \in \mathbb{R}^d$, donde $d$ es la dimensión del espacio latente generado por el modelo.

Para obtener la predicción de clase, este vector se proyecta mediante una capa lineal parametrizada por una matriz de pesos $\mathbf{W} \in \mathbb{R}^{|\mathcal{Y}| \times d}$ y un vector de sesgo $\mathbf{b} \in \mathbb{R}^{|\mathcal{Y}|}$, donde $|\mathcal{Y}|$ corresponde al número total de clases. La operación se expresa como:

$$\mathbf{z} = \mathbf{W}\,\mathbf{h}_{\texttt{[CLS]}} + \mathbf{b}.$$

El vector resultante $\mathbf{z} \in \mathbb{R}^{|\mathcal{Y}|}$ contiene un valor (o *logit*) para cada clase posible. Finalmente, al aplicar la función *softmax* sobre $\mathbf{z}$, se obtiene una distribución de probabilidad:

$$p(y = i \mid d) = \frac{e^{z_i}}{\sum_{j=1}^{|\mathcal{Y}|} e^{z_j}},$$

donde $p(y = i \mid d)$ representa la probabilidad de que el documento $d$ pertenezca a la clase $i$.

La arquitectura de clasificación basada en BERT utilizada en este trabajo se esquematiza en la Figura 4.1.

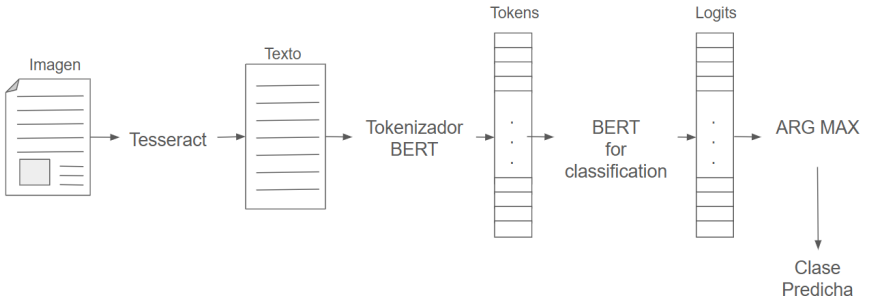


Figura 4.1: Esquema del modelo de clasificación basado en texto utilizando BERT en el conjunto Tobacco-3482.

Durante el entrenamiento se utiliza una **función de pérdida** que mide la discrepancia entre predicción y etiqueta real. En este trabajo se empleó la entropía cruzada:

$$\mathcal{L}(y, \hat{y}) = -\sum_{i=1}^{|\mathcal{Y}|} y_i \log(\hat{y}_i),$$

minimizada mediante retropropagación.

## Implementación y resultados

En los experimentos con el modelo basado en texto se exploraron distintas tasas de aprendizaje: inicialmente se probaron los valores 0,1, 0,01, 0,001 y 0,0001. Como la tasa que mejor desempeño mostró fue 0,001, se realizó un ajuste fino probando valores más cercanos, específicamente 0,0005 y 0,005. En esas pruebas, la tasa 0,001 siguió siendo la óptima. En la Figura 4.2 se muestran los resultados de la primera etapa de exploración, y en la Figura 4.3 el ajuste fino alrededor de esa mejor configuración.

La mejor *accuracy* alcanzada sobre el conjunto de validación fue de 0,796. Esta cifra sirve como línea base razonable para el corpus, pues aunque no captura toda la variabilidad del conjunto, está dentro del orden de magnitud de los resultados reportados en la literatura para Tobacco-3482. Por ejemplo, Kölsch et al. (2017) reportan una *accuracy* de 83.24 % utilizando CNN + ELMs sobre Tobacco-3482 como comparativa [35]. Además, en trabajos recientes se han reportado modelos multimodales o mejoras visuales que superan ampliamente ese umbral; por ejemplo, WordVIS alcanzó 91.14 % de *accuracy* sobre Tobacco-3482 en una publicación de 2024 [36].

Aunque el valor de 0.796 no es competitivo frente a las implementaciones más recientes, representa una línea de partida confiable dado que se obtuvo con un modelo

exclusivamente textual sin ajustes complejos ni soporte multimodal y permite medir la ganancia que trae incorporar modalidades visuales y estrategias de fusión en etapas posteriores del trabajo.

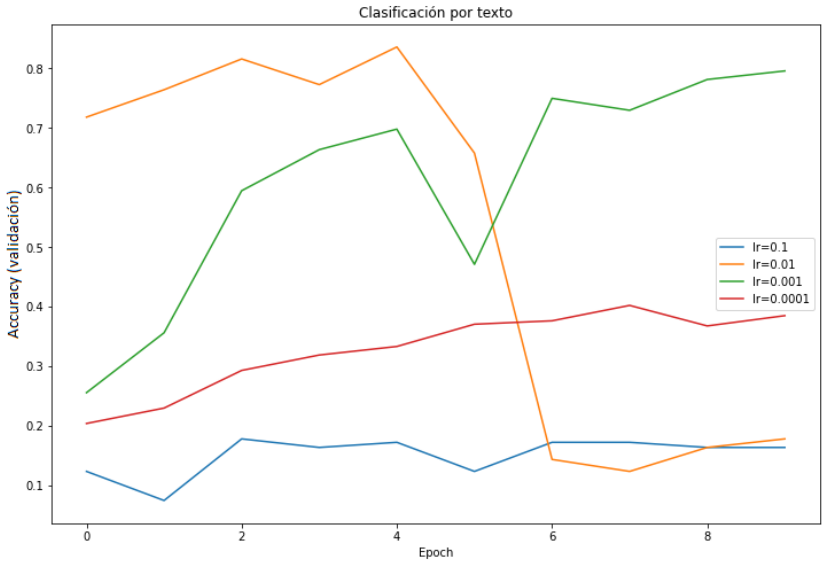


Figura 4.2: Curvas de entrenamiento y validación para las tasas de aprendizaje $\{0{,}1, 0{,}01, 0{,}001, 0{,}0001\}$ en el modelo textual. Se observa que $\alpha = 0{,}001$ logra un balance entre estabilidad y precisión, mientras que tasas mayores producen sobreajuste temprano o inestabilidad.

## 4.4. Métodos basados en imagen

### Aspectos conceptuales y arquitectura

Las redes neuronales convolucionales (CNN, por sus siglas en inglés) son arquitecturas diseñadas específicamente para procesar datos con estructura espacial, como imágenes. Su elemento central es la **operación de convolución**, que aplica un filtro o kernel sobre regiones locales de la imagen. Cada filtro se desplaza por la matriz de píxeles y calcula combinaciones lineales que permiten detectar patrones básicos, como bordes o texturas. Al apilar varias capas convolucionales, se construye una jerarquía de representaciones: las capas iniciales identifican características simples, mientras que las más profundas capturan estructuras complejas y patrones semánticos.

Además de las capas convolucionales, las CNNs suelen incorporar operaciones de *pooling*, que reducen la dimensionalidad agregando información local (por ejemplo, tomando máximos o promedios). Esto ayuda a resumir características relevantes y aporta invarianza ante traslaciones o pequeñas deformaciones en la imagen.

### Implementación y resultados

En este trabajo se empleó EfficientNet-B0 [29], una arquitectura moderna de redes CNN que se distingue por su estrategia de *escalamiento compuesto*. A diferencia de los enfoques tradicionales que aumentan profundidad, ancho o resolución de manera independiente, EfficientNet combina los tres factores de forma balanceada, logrando

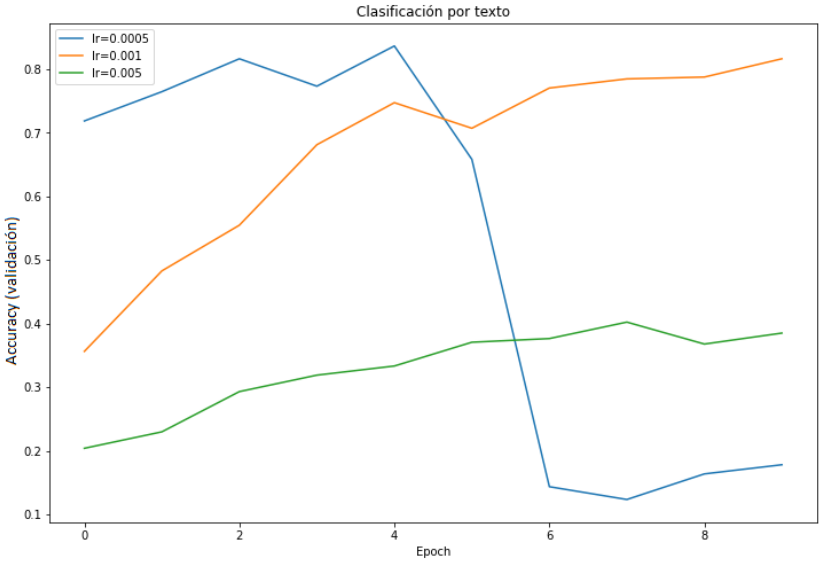


Figura 4.3: Exploración de valores cercanos a la tasa de aprendizaje seleccionada (0,001). No se observaron mejoras significativas frente a la configuración base, confirmando a 0,001 como la opción más estable.

un mejor compromiso entre precisión y costo computacional. El esquema general de la arquitectura utilizada se muestra en la Figura 4.4.

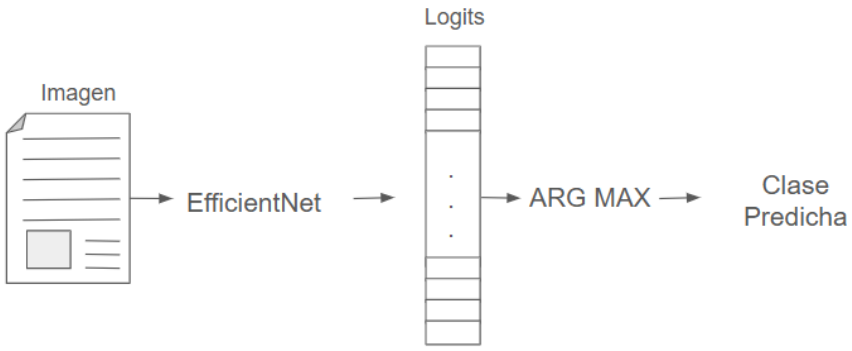


Figura 4.4: Esquema del modelo visual. Las imágenes escaneadas de documentos se redimensionan y procesan mediante EfficientNet-B0. Las capas convolucionales extraen características jerárquicas y, tras anular la capa de clasificación original, se añade una nueva capa *softmax* adaptada a las categorías de Tobacco-3482.

El hiperparámetro principal a ajustar fue la tasa de aprendizaje. En una primera etapa de exploración se ensayaron los valores 0,1, 0,01, 0,001 y 0,0001. Como la tasa más prometedora fue 0,01, en una segunda etapa se probó con valores intermedios cercanos, específicamente 0,05 y 0,075. De esta manera, se observó que 0,075 ofrecía el mejor equilibrio entre estabilidad y desempeño. En la Figura 4.5 se muestran los

resultados iniciales, donde algunas configuraciones presentaron falta de convergencia, mientras que en la Figura 4.6 se evidencia la mejora obtenida en la etapa de ajuste.

Los resultados confirman que el enfoque visual supera consistentemente al textual en este corpus, alcanzando valores de *accuracy* más altos y una convergencia más estable. Este comportamiento resulta coherente con la naturaleza del conjunto Tobacco-3482, donde los rasgos visuales como la diagramación, el tipo de letra y la disposición de los elementos aportan información discriminante que no siempre queda reflejada en el texto extraído por OCR.

En la literatura, otros trabajos también destacan la potencia de modelos basados en imágenes para este conjunto de datos. Por ejemplo, Harley et al. (2015) reportaron resultados competitivos utilizando únicamente CNN sobre Tobacco-3482 [7], y posteriores mejoras con arquitecturas más profundas como ResNet han mostrado incrementos adicionales [30]. En este contexto, los resultados obtenidos con EfficientNet-B0 en esta tesis pueden considerarse una línea de referencia sólida, equilibrando precisión y eficiencia computacional, y marcando un punto de partida para posteriores exploraciones multimodales.

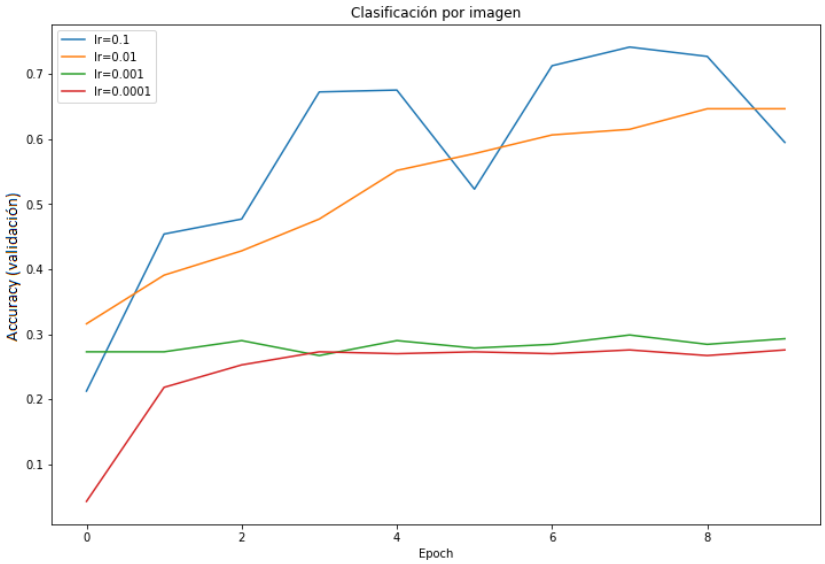


Figura 4.5: Exploración inicial de tasas de aprendizaje para EfficientNet-B0. Algunas configuraciones muestran inestabilidad o falta de convergencia. Se seleccionan tasas intermedias para una segunda etapa de ajuste fino.

## 4.5. Métodos híbridos

### Aspectos conceptuales y arquitecturas de fusión

Los métodos híbridos integran simultáneamente información textual y visual con el objetivo de aprovechar las ventajas de ambas modalidades. Existen dos estrategias principales [37]:

- **Fusión temprana (early fusion):** consiste en diseñar un único modelo multimodal que procese de manera conjunta texto e imagen. En este trabajo se

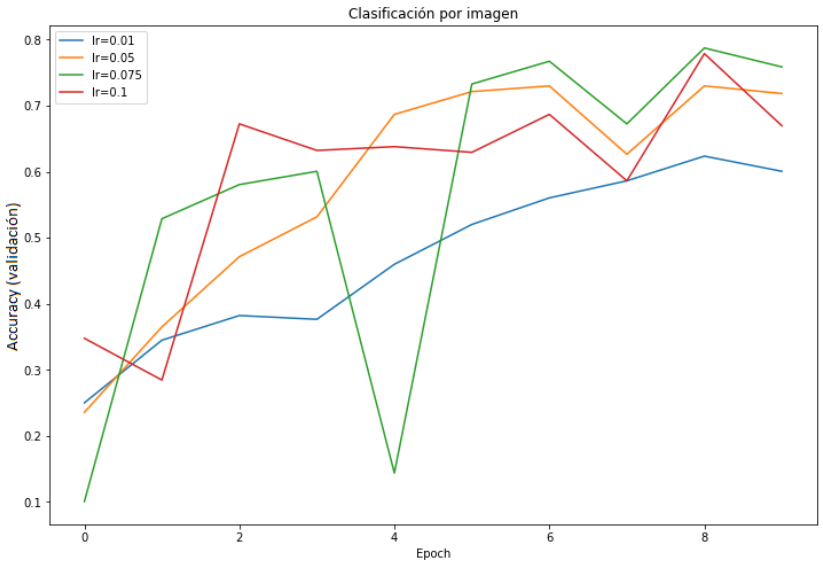


Figura 4.6: Ajuste fino de tasas en torno a la mejor configuración identificada en la etapa 1. Se observa mayor estabilidad en las curvas de validación y mejora sostenida frente a configuraciones extremas.

construyó un modelo que incluye como componentes internos a BERT y a EfficientNet. La lista de tokens extraída mediante OCR se procesa con BERT para obtener representaciones semánticas, mientras que la imagen escaneada se procesa con EfficientNet para extraer características visuales. Ambas representaciones se concatenan y se pasan a través de una capa lineal, que produce los *logits* correspondientes a cada clase. Este enfoque permite capturar interacciones entre modalidades, aunque presenta desafíos de alta dimensionalidad y requiere un mayor volumen de datos para alcanzar su máximo potencial.

- **Fusión tardía (late fusion):** consiste en entrenar por separado un modelo textual y uno visual, y luego combinar sus salidas de probabilidad. En este trabajo se utilizó la variante más simple: el promedio aritmético de las distribuciones de probabilidad generadas por cada modelo. Este enfoque es más sencillo de implementar y, aunque no captura interacciones profundas entre modalidades, suele ofrecer mejoras de desempeño al aprovechar la robustez de cada modelo unimodal.

Ambas variantes se ilustran en las Figuras 1.1 y 1.2.

## Implementación y resultados

En el caso de la clasificación combinada se exploraron diferentes tasas de aprendizaje en dos etapas. En la primera, se ensayaron los valores 0,1, 0,01, 0,001 y 0,0001. El mejor desempeño se alcanzó con una tasa de 0,1, que mostró una convergencia más rápida y estable que las configuraciones alternativas. Los resultados de esta etapa se presentan en la Figura 4.7.

A partir de este resultado, se realizaron experimentos adicionales manteniendo la tasa en 0,1 pero comparando el uso de transformaciones de aumento de datos en las imágenes frente a su ausencia. Los resultados fueron similares en ambos casos, lo

que indica que la robustez del modelo combinado no depende fuertemente de dichas transformaciones, posiblemente porque la integración de información textual y visual ya introduce un grado de variabilidad suficiente. La Figura 4.8 muestra la comparación entre ambas configuraciones.

El esquema de fusión tardía, implementado mediante el promedio de las salidas de los modelos unimodales, resultó más efectivo, alcanzando un *accuracy* de validación de 0,908. La fusión temprana, en cambio, evidenció limitaciones atribuibles a la alta dimensionalidad y a la necesidad de mayor volumen de datos para entrenar representaciones conjuntas robustas, lo que restringió su rendimiento en comparación con el enfoque de promedio.

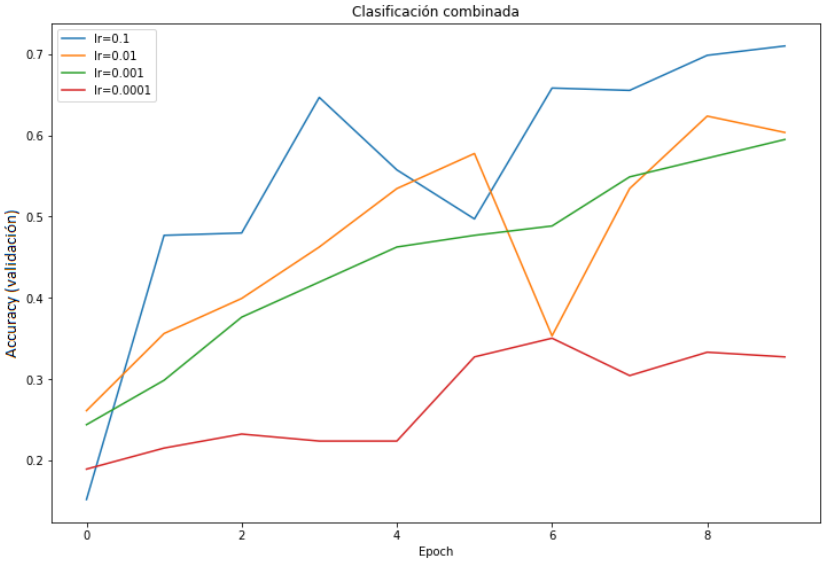


Figura 4.7: Exploración inicial de tasas de aprendizaje para el esquema combinado. Se observa que algunas configuraciones generan sobreajuste temprano, mientras que otras mantienen un aprendizaje más estable.

## 4.6. Clasificación incompleta

En este trabajo se exploró un escenario de clasificación incompleta, en el que no se dispone de un conjunto exhaustivo de categorías para etiquetar la totalidad de los documentos. En tales casos resulta útil incorporar una clase adicional que agrupe aquellos documentos cuya categoría específica no ha sido determinada. A esta clase la denominamos *sin categoría*.

Para simular esta situación, se utilizó el conjunto de datos Tobacco-3482 con un esquema de etiquetas modificado. En particular, se conservaron las cinco primeras categorías originales, mientras que los documentos pertenecientes a las cinco categorías restantes fueron recategorizados bajo la nueva clase *sin categoría*. De este modo, se reproduce un contexto en el que solo una fracción del corpus está completamente clasificada, mientras que el resto se concentra en una clase residual.

A partir de este conjunto modificado, se entrenaron de forma independiente dos modelos: uno basado en el contenido textual obtenido mediante OCR y otro utilizando representaciones visuales extraídas de las imágenes de documentos. En ambos casos

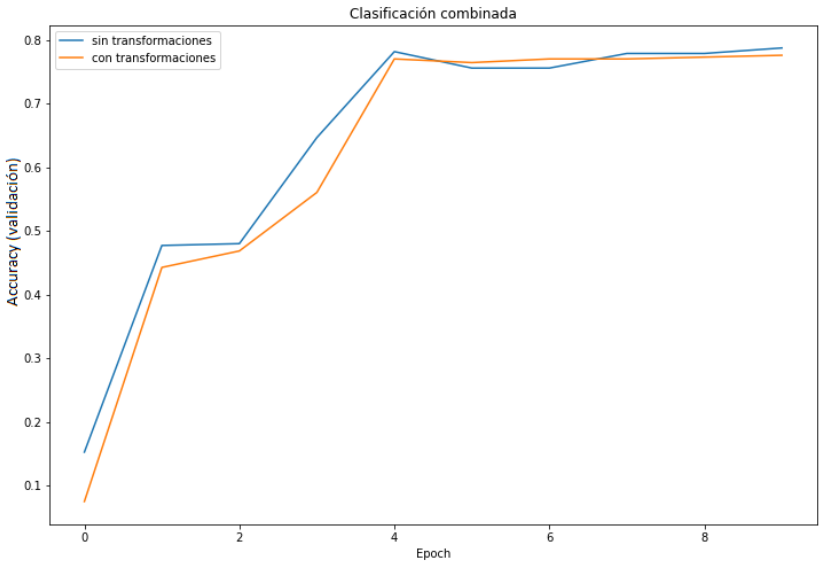


Figura 4.8: Ajuste fino en torno a la mejor tasa de aprendizaje identificada en la primera etapa. Se confirma que la configuración seleccionada mantiene un desempeño consistente y generaliza mejor en validación.

se mantuvieron las arquitecturas y los hiperparámetros previamente descritos para los experimentos unimodales, sin realizar ajustes adicionales. Finalmente, se combinó la salida de ambos modelos mediante un promedio simple de probabilidades.

El resultado sobre el conjunto de validación mostró un *accuracy* de 0.914, lo que indica que la estrategia es adecuada para contextos en los que el esquema de categorías es incompleto. Si bien la clase *sin categoría* constituye una simplificación del problema real, este experimento sugiere que los modelos pueden adaptarse a escenarios con taxonomías parciales y mantener un nivel elevado de desempeño.

## 4.7. Análisis por categoría: Advertising

En la categoría `Advertising` se analizaron en detalle las probabilidades asignadas por los modelos textuales, visuales y multimodales. La lógica de este análisis es sencilla: un documento de tipo `Advertising` debería recibir una probabilidad alta para dicha clase; en cambio, valores bajos indicarían que el modelo considera que pertenece a otra categoría.

Los resultados se presentan en las Figuras 4.9, 4.10 y 4.11. Cada punto en los gráficos corresponde a un documento de la categoría, donde el color indica si fue correctamente clasificado (negro), clasificado de forma errónea (rojo) o corregido mediante la combinación multimodal (azul).

En el caso del modelo visual (Figura 4.9), se observa que la mayoría de los documentos mal clasificados presentan probabilidades altas para `Advertising`, pero fueron desplazados hacia otra clase con una probabilidad aún mayor. Es decir, el error no surge de una incapacidad del modelo para reconocer patrones visuales de la categoría, sino de confusiones con características visuales compartidas con otras clases. Solo un grupo reducido de documentos muestra valores bajos (menores a 0.5), lo que indicaría que efectivamente el modelo no los asocia visualmente con la categoría.

El modelo textual (Figura 4.10) exhibe un comportamiento similar: la mayoría de los errores no se deben a que el modelo haya asignado una baja probabilidad a `Advertising`, sino a que simultáneamente otorgó una probabilidad mayor a otra clase. Esto refuerza la idea de que el texto disponible contiene señales útiles para la clasificación, pero que esas señales no siempre son discriminativas frente a clases cercanas. Solo un caso aparece con probabilidad realmente baja, lo cual marca un error más profundo del modelo.

Finalmente, la combinación multimodal mediante promedio de probabilidades (Figura 4.11) reduce de manera significativa la cantidad de documentos mal clasificados: de todos los casos erróneos, solo cuatro permanecen incorrectos y únicamente uno de ellos presenta un valor claramente bajo para `Advertising`. Esto indica que los errores cometidos por los modelos unimodales no siempre coinciden, y que la fusión tardía aprovecha esa complementariedad. La aparición de puntos azules —documentos corregidos gracias al promedio— demuestra que el esquema multimodal logra rescatar predicciones que individualmente habrían fallado.

En conjunto, este análisis evidencia dos aspectos importantes. Primero, tanto las modalidades de texto como de imagen capturan información relevante y consistente con la categoría, aunque con limitaciones específicas. Segundo, la integración de ambas modalidades, incluso mediante un mecanismo simple como el promedio, puede mejorar sustancialmente la robustez de la clasificación. Esto sugiere que para categorías complejas o heterogéneas como `Advertising`, los enfoques multimodales no solo son deseables sino probablemente necesarios.

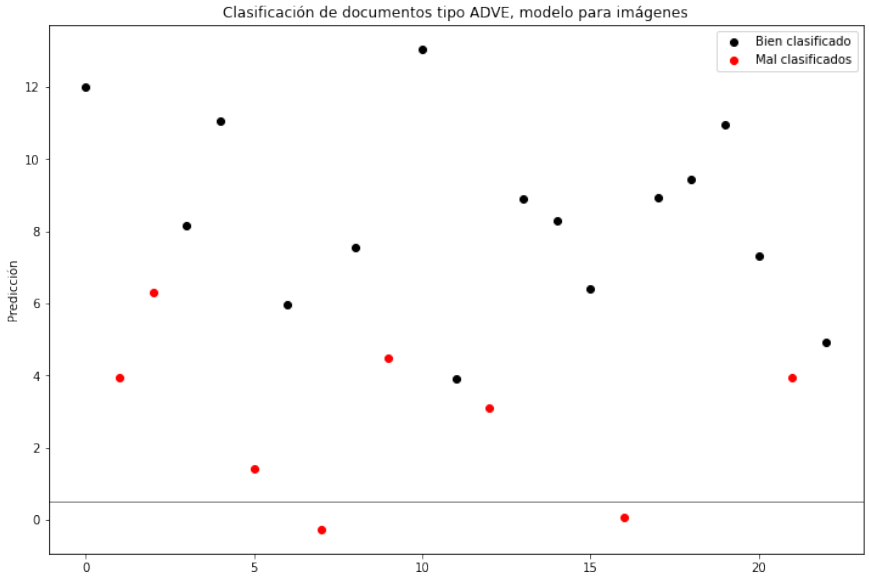


Figura 4.9: Documentos de la categoría `ADVE` clasificados con el modelo visual. Puntos negros: documentos correctamente clasificados; puntos rojos: documentos mal clasificados. La mayoría de los errores corresponden a documentos con alta probabilidad asignada a `ADVE`, pero con otra clase aún más probable. Solo unos pocos errores muestran probabilidad inferior a 0.5.

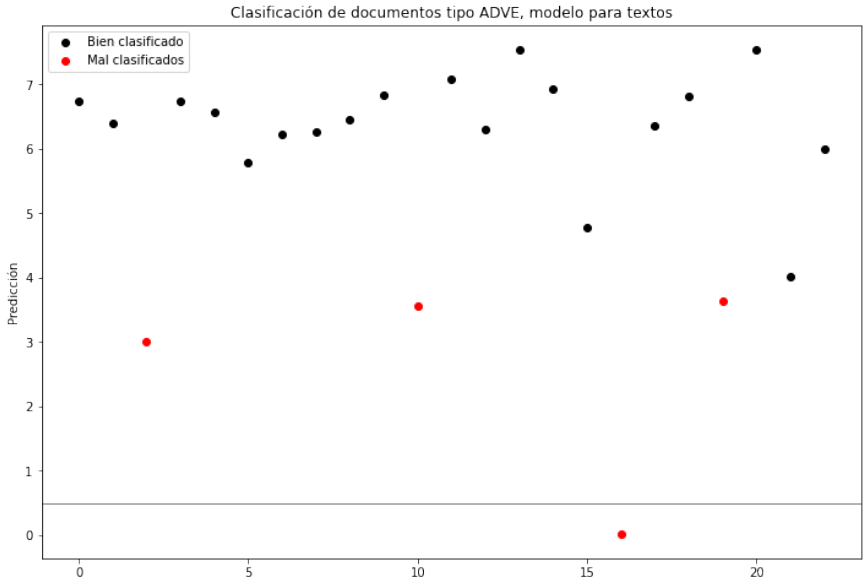


Figura 4.10: Resultados del modelo textual en la categoría ADVE. De los documentos mal clasificados, la mayoría obtuvieron valores altos para ADVE, lo que indica que el error provino de otra clase con probabilidad aún mayor. Solo un documento mal clasificado presentó probabilidad menor a 0.5.

## 4.8. Discusión general

Como se puede observar en la Tabla 4.1, la evaluación sobre Tobacco-3482 muestra un patrón claro:

- Los métodos textuales (BERT) logran un desempeño razonable pero limitado frente a la heterogeneidad del corpus.
- Los enfoques basados en imagen (EfficientNet) superan a los textuales cuando la estructura visual es informativa o el OCR es ruidoso.
- La fusión multimodal, particularmente la *tardía*, ofrece la mejor performance global en esta etapa (validación: 0,908).
- La clasificación incompleta añade un mecanismo útil para simular escenarios con taxonomías parciales.

Como se muestra en la Tabla 4.1, la comparación entre los distintos enfoques revela que los métodos multimodales, y en particular la fusión tardía, alcanzan el mayor rendimiento en accuracy al combinar la información textual y visual.

En conjunto, la integración de modalidades constituye un camino prometedor para la clasificación documental. Si bien la fusión temprana exige mayor diseño y más datos, los resultados respaldan que combinar texto e imagen mejora la robustez frente a la variabilidad de los documentos.

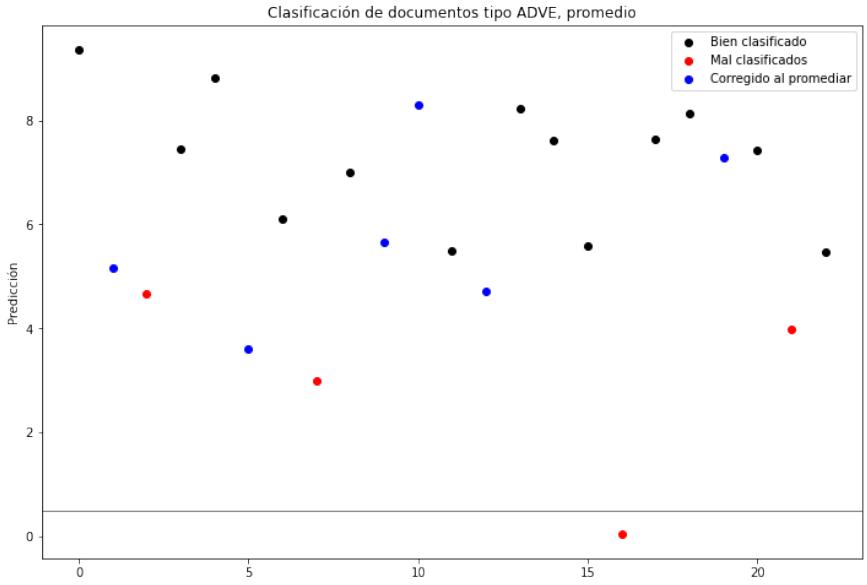


Figura 4.11: Clasificación multimodal mediante promedio de probabilidades. Se reducen los errores a solo cuatro documentos, de los cuales solo uno tiene una probabilidad realmente baja para `ADVE`. Esto muestra la complementariedad de las modalidades y la capacidad del promedio de mejorar la robustez de las predicciones.

Tabla 4.1: Comparación de métodos de clasificación documental en Tobacco-3482

| **Método** | **Ventajas** | **Desventajas** | **Accuracy (validación)** |
|---|---|---|---|
| **Texto (BERT)** | Captura relaciones contextuales bidireccionales; robusto ante variaciones semánticas; ampliamente probado en NLP. | Sensibilidad al ruido del OCR; requiere corpus grandes para *fine-tuning*. | 0.796 |
| **Imagen (EfficientNet-B0)** | Extrae directamente estructura visual; eficiente en parámetros; aprovecha tipografía, diagramación y layout. | Pierde información semántica explícita del texto; requiere imágenes de buena calidad. | 0.861 |
| **Multimodal (fusión temprana)** | Permite capturar interacciones entre texto e imagen en un único espacio de representación. | Alta dimensionalidad; necesita mayor volumen de datos; entrenamiento complejo. | 0.842 |
| **Multimodal (fusión tardía)** | Combina fortalezas de modelos unimodales; sencillo de implementar; mayor robustez frente al ruido. | Ignora interacciones profundas entre modalidades; depende del desempeño unimodal. | 0.908 |
| **Clasificación incompleta** | Permite manejar taxonomías parciales; evita decisiones forzadas; mantiene buen desempeño global. | Clase residual puede ocultar patrones finos; simplificación del problema real. | 0.914 |

Esta página ha sido intencionalmente dejada en blanco.

# Capítulo 5

# Clasificación del Archivo Berrutti

Los modelos desarrollados y evaluados sobre el conjunto Tobacco-3482, con algunas adaptaciones, fueron posteriormente aplicados a un subconjunto de documentos provenientes del Archivo Berrutti.

## 5.1. Modelos binarios

El volumen de datos disponible en el Archivo Berrutti es considerablemente mayor que en el conjunto Tobacco-3482, lo que conlleva un aumento en los tiempos de entrenamiento. Además, en etapas preliminares se detectaron problemas de etiquetado en los documentos anotados con LabelMe, lo cual afectó negativamente la calidad de algunos ensayos iniciales. Por este motivo, antes de abordar un escenario multiclase, se optó por implementar modelos de clasificación binaria, cuya tarea consiste en determinar si un documento pertenece o no a una determinada categoría. Una vez obtenidos resultados satisfactorios en este esquema más simple, se avanzó hacia modelos más complejos.

### 5.1.1. Textos

Siguiendo la estrategia adoptada en la subsección de modelos binarios, se comenzó evaluando el desempeño del modelo basado únicamente en texto. Para ello se empleó BERT preentrenado en español, utilizando como entrada el contenido de los documentos del Archivo Berrutti extraído mediante OCR.

El modelo recibe como entrada la secuencia tokenizada y la procesa mediante BERT configurado para generar una representación vectorial de 1.000 dimensiones. Esta salida se conecta a una capa lineal que transforma el vector de características en un escalar, entrenado en configuración binaria para distinguir entre pertenencia o no a una clase específica. Durante la inferencia, se considera que un documento pertenece a la clase cuando la salida correspondiente del modelo es mayor o igual a 0.5.

- **Entrada:** Secuencia tokenizada mediante BERT.
- **BERT para clasificación:** Configurado para producir una salida de dimensión 1.000.
- **Capa lineal:** Proyecta el vector en un valor escalar para la clasificación binaria.

Se entrenaron dos instancias del modelo: una para clasificar las Fichas de AFE y otra para las Actas de la OCOA En cada caso, se construyó un conjunto de entrenamiento balanceado compuesto por documentos de la categoría objetivo y una

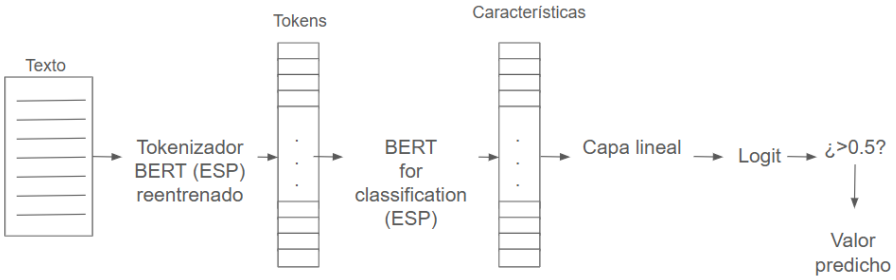


Figura 5.1: El modelo toma como entrada el texto de cada documento, genera una secuencia de tokens mediante el tokenizador BERT y obtiene un vector de 1.000 dimensiones utilizando *BERT for Classification*. Este vector pasa por una capa lineal que produce un valor escalar. Se espera que este valor sea bajo (menor a 0.5) cuando el documento no pertenece a la categoría objetivo, y alto (mayor a 0.5) en caso contrario.

cantidad equivalente de documentos de cualquier otra categoría anotados mediante LabelMe. Por ejemplo, para las fichas de AFE se emplearon 10.322 documentos por clase, mientras que para las actas de la OCOA se utilizaron 9.838 en cada grupo.

Se entrenó de forma progresiva con distintos largos de secuencia de tokens. Los resultados obtenidos se presentan en la Tabla 5.1, donde se observa que el modelo alcanza una *accuracy* elevada para la clasificación de fichas de AFE incluso con secuencias relativamente cortas, estabilizándose en torno al 99.5 % al aumentar la longitud de entrada. En el caso de las actas de la OCOA, el rendimiento es inferior, aunque mejora al aumentar la cantidad de tokens procesados.

Tabla 5.1: *Accuracy* para distintos largos (en cantidad de tokens)

| Cantidad de tokens | *Accuracy* Fichas AFE | *Accuracy* Actas OCOA |
|---|---|---|
| 16 | 0.990 | 0.660 |
| 32 | 0.991 | 0.593 |
| 64 | 0.993 | 0.681 |
| 128 | 0.993 | 0.662 |
| 256 | 0.993 | 0.762 |
| 512 | 0.995 | 0.747 |

Posteriormente, se repitieron los entrenamientos empleando la tasa de aprendizaje óptima determinada automáticamente mediante la funcionalidad disponible en la biblioteca Torch. Los resultados con la tasa óptima aplicada en cada caso se presentan en las Tablas 5.2 y 5.3.

Estos resultados indican que la búsqueda automática de la tasa de aprendizaje no garantiza una mejora consistente: en el caso de OCOA algunos entrenamientos no convergen, mientras que para AFE los resultados son estables y robustos. La mejora observada en AFE muestra la importancia del ajuste de hiperparámetros, mientras que la variabilidad en OCOA sugiere que factores como la calidad del OCR y la complejidad estilística limitan el rendimiento del modelo.

Finalmente, se entrenó el modelo con secuencias de 512 tokens utilizando el tokenizador original de BERT, sin reentrenarlo con documentos del Archivo Berrutti. En este caso, los resultados fueron incluso superiores:

Tabla 5.2: Fichas AFE, *Accuracy* para distintos largos (en cantidad de tokens)

| Cantidad de tokens | Tasa de aprendizaje | *Accuracy* |
|---|---|---|
| 16 | 0.125 | 0.992 |
| 32 | 0.009 | 0.992 |
| 64 | 0.008 | 0.994 |
| 128 | 0.003 | 0.994 |
| 256 | 0.009 | 0.996 |
| 512 | 0.0025 | 0.995 |

Tabla 5.3: Actas OCOA, *Accuracy* para distintos largos (en cantidad de tokens)

| Cantidad de tokens | Tasa de aprendizaje | *Accuracy* |
|---|---|---|
| 16 | 0.074 | 0.665 |
| 32 | 0.088 | 0.643 |
| 64 | 0.035 | 0.676 |
| 128 | 0.192 | 0.500 |
| 256 | 0.025 | 0.747 |
| 512 | 0.039 | 0.589 |

- **Fichas AFE:** *Accuracy* 0.995
- **Actas OCOA:** *Accuracy* 0.883

Este hallazgo es relevante porque muestra que el tokenizador preentrenado, construido a partir de un corpus mucho más amplio y diverso, resulta más adecuado que uno reentrenado con un conjunto limitado de textos. En particular, en OCOA el salto de 0.747 a 0.883 confirma que el reentrenamiento del tokenizador no aporta beneficios y puede ser contraproducente.

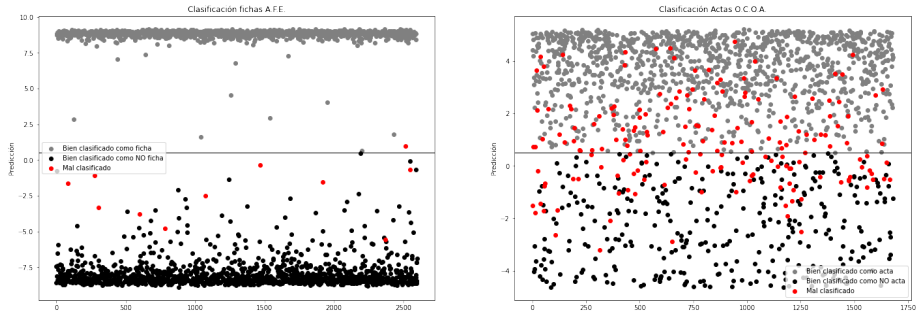


Figura 5.2: Comparación del rendimiento del modelo de clasificación textual en las dos clases analizadas. En el caso de las fichas de AFE, los valores de salida se alejan claramente del umbral de decisión (0.5), lo que facilita una separación nítida entre clases. Por el contrario, las actas de la OCOA presentan una distribución más difusa, con mayor solapamiento entre clases.

La Figura 5.2 permite observar que, en el caso de las fichas de AFE, el modelo separa claramente la clase objetivo del resto, mientras que en las actas de la OCOA la frontera es más difusa, lo que ilustra las dificultades de clasificación observadas en las métricas.

### 5.1.2. Imágenes

Los documentos del Archivo Berrutti están digitalizados en formato binario, donde los píxeles blancos se codifican como ceros y los negros como unos. Para el entrenamiento, las imágenes se escalaron a una resolución de 400×400 píxeles y se convirtieron a escala de grises.

Tras realizar pruebas comparativas preliminares entre modelos conocidos como LeNet-5 [38], ResNet18 [30] y EfficientNet [29], se concluyó que EfficientNet proporciona los mejores resultados para este tipo de documentos, por lo que se adoptó como arquitectura principal para esta etapa. La Figura 5.3 ilustra el procedimiento adoptado:

- **Entrada:** Imagen de 400×400 píxeles en escala de grises.
- **EfficientNet:** configurado para producir una representación vectorial de dimensión 1.000.
- **Capa lineal:** Transforma el vector de características a una dimensión igual al número de clases.

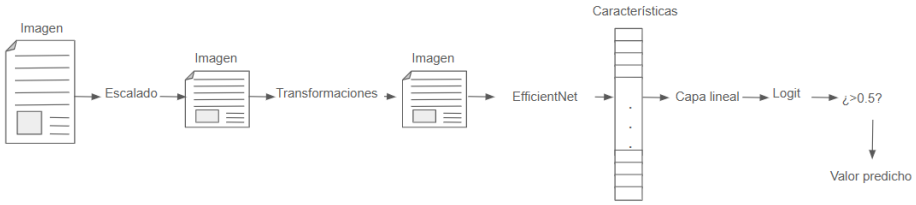


Figura 5.3: El modelo recibe como entrada una imagen procesada del documento. Esta se somete a transformaciones aleatorias y se procesa mediante EfficientNet, obteniéndose un vector de 1.000 características. Luego, una capa lineal transforma ese vector en una salida escalar. Se espera que esta salida sea baja para documentos fuera de la clase objetivo y alta en caso contrario.

El entrenamiento se realizó durante cinco épocas por cada clase, con un tamaño de lote de 32. Para mejorar la capacidad de generalización y prevenir el sobreajuste, se aplicaron transformaciones aleatorias a las imágenes durante el entrenamiento:

- Variaciones de brillo, contraste, saturación y tono (±10 %).
- Rotaciones aleatorias de hasta 8 grados.
- Traslaciones horizontales y verticales (hasta 10 %).
- Ruido gaussiano (1 % de probabilidad).

Con una tasa de aprendizaje arbitraria fija de 0.05, los resultados obtenidos fueron:

- **Fichas AFE:** *Accuracy* 0.989
- **Actas OCOA:** *Accuracy* 0.956

Estos valores confirman que, en el caso de AFE, la información visual resulta tan discriminativa como el texto, alcanzando desempeños muy similares a los del modelo textual. En cambio, para OCOA el resultado visual es muy superior al textual (0.956 vs. 0.883), lo que indica que la estructura gráfica de las actas ofrece pistas más fiables que el texto ruidoso extraído por OCR. Este hallazgo coincide con la intuición de que documentos con diagramación estable y formatos repetitivos favorecen el aprendizaje visual.

Tabla 5.4: *Accuracy* con tasa de aprendizaje óptima

| Clase | Tasa de aprendizaje | *Accuracy* |
|---|---|---|
| Fichas AFE | 0.166 | 0.998 |
| Actas OCOA | 0.434 | 0.977 |

Al repetir el entrenamiento utilizando la tasa de aprendizaje óptima calculada automáticamente, los resultados mejoraron aún más:

La mejora alcanzada tras optimizar la tasa de aprendizaje confirma la importancia de una búsqueda sistemática de la tasa de aprendizaje. En particular, en fichas AFE el modelo logra un desempeño prácticamente perfecto (0.998), mientras que en OCOA la ganancia es más moderada, lo que sugiere que el límite de rendimiento está dado no tanto por la arquitectura como por la calidad intrínseca de los datos. En cualquier caso, estos resultados muestran que la modalidad visual por sí sola resulta altamente competitiva en la clasificación de documentos del Archivo Berrutti.

### 5.1.3. Combinado: Promedio de modelos

A partir de los modelos entrenados individualmente para texto e imagen, se implementó un esquema de combinación por promedio que se ilustra en la Figura 5.4. Para cada documento, se calcularon las salidas de ambos clasificadores y se promediaron. Si el valor promedio superaba el umbral de 0.5, el documento se consideraba perteneciente a la clase objetivo.

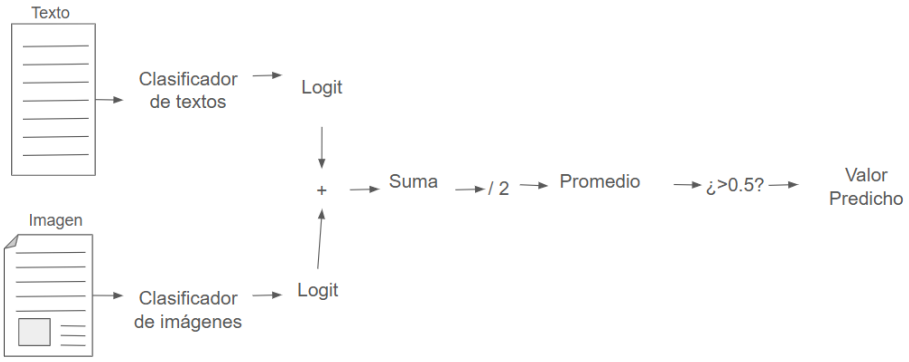


Figura 5.4: Esquema combinado: cada documento se procesa en paralelo por los modelos de texto e imagen. Las salidas numéricas generadas por ambos clasificadores se promedian para obtener un valor final.

Los resultados obtenidos demuestran una mejora respecto a los modelos individuales, especialmente en los documentos más difíciles de clasificar, como las actas de la OCOA. En la Tabla 5.5 se observa que el modelo combinado alcanza un *accuracy* de 0.999 para fichas AFE y de 0.983 para OCOA, valores que superan a los unimodales.

En la Figura 5.5 se representa, para cada documento, el valor asignado por el modelo, recordando que valores superiores a 0.5 indican la pertenencia a la categoría objetivo. En el caso de las fichas de A.F.E., se observa una separación nítida entre los documentos correctamente clasificados como fichas y aquellos correctamente identificados como no fichas; los errores de clasificación se concentran en valores cercanos

al umbral de decisión, lo que sugiere incertidumbre del modelo únicamente en casos límite.

En cambio, para las actas de la O.C.O.A. la separación entre clases es menos marcada. Se identifican documentos mal clasificados con valores alejados del umbral, incluyendo actas rechazadas por el modelo con una puntuación inferior a la asignada a documentos que efectivamente no pertenecen a esa categoría. Este comportamiento indica que, en esta clase, el modelo no solo duda en los casos limítrofes, sino que confunde ciertos patrones propios del tipo documental.

Tabla 5.5: *Accuracy* del modelo combinado por promedio

| Clase | *Accuracy* |
|---|---|
| Fichas AFE | 0.999 |
| Actas OCOA | 0.983 |

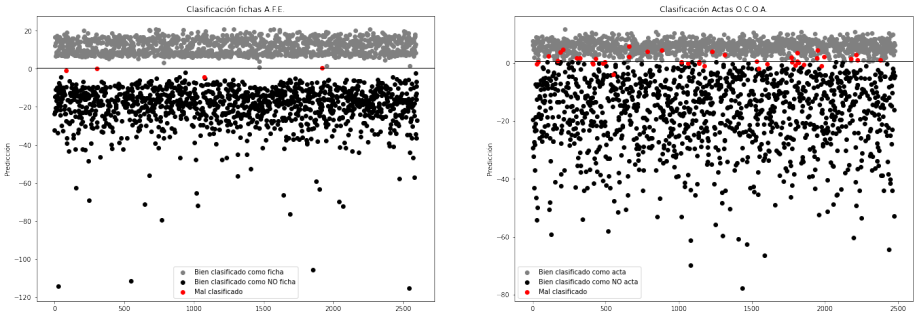


Figura 5.5: La clasificación combinada permite una mejor separación de clases en ambos casos. Se observa una mayor consistencia en la clasificación de fichas de AFE, mientras que la nube de puntos correspondiente a las actas de la OCOA continúa mostrando mayor dispersión.

En conjunto, este enfoque de combinación simple demuestra que integrar la información proveniente de distintas modalidades (texto e imagen) mejora significativamente el desempeño de los clasificadores. La ganancia es marginal en AFE, donde los modelos unimodales ya eran casi perfectos, pero en OCOA el efecto es muy claro: la fusión compensa las debilidades del texto (afectado por ruido de OCR) con la fortaleza de la modalidad visual. Aunque el método de promedio no explota interacciones profundas entre modalidades, ofrece una solución práctica, robusta y de bajo costo computacional, con resultados comparables a los de un modelo híbrido más complejo.

## 5.2. Modelos multiclase

Luego de los experimentos con modelos binarios, se abordó la clasificación multiclase en el Archivo Berrutti. El objetivo fue entrenar un único modelo capaz de distinguir entre varias categorías de documentos de manera simultánea. Para ello se adaptaron las arquitecturas previamente utilizadas en configuración binaria, ajustando la capa de salida al número total de clases y aplicando la función *softmax* para generar distribuciones de probabilidad.

### 5.2.1. Textos

El modelo BERT en español se aplicó directamente sobre el contenido textual extraído mediante OCR, manteniendo la configuración de secuencias de 512 tokens. El entrenamiento se realizó sobre un conjunto balanceado de documentos representativos de cada clase.

Tabla 5.6: *Accuracy* del modelo de textos para distintas tasas de aprendizaje

| Tasa de aprendizaje | Entrenamiento | Validación |
|---|---|---|
| 0.0005 | 0.930 | 0.926 |
| 0.001 | 0.940 | 0.933 |
| 0.005 | 0.963 | 0.948 |
| 0.01 | 0.962 | 0.950 |
| 0.05 | 0.260 | 0.260 |
| 0.1 | 0.260 | 0.260 |

Como se muestra en la Tabla 5.6, el modelo presenta un comportamiento claramente dependiente de la tasa de aprendizaje. Para valores elevados (0,05 y 0,1), el entrenamiento no converge y el *accuracy* cae abruptamente, lo que evidencia inestabilidad en la optimización. En cambio, tasas comprendidas entre 0,0005 y 0,01 permiten alcanzar desempeños estables y consistentes tanto en entrenamiento como en validación. Entre ellas, la tasa de 0,01 resulta la más adecuada, ya que combina un *accuracy* alto en entrenamiento con una buena capacidad de generalización en validación.

Los resultados mostraron que, aunque el modelo textual captura buena parte de la información semántica, el ruido introducido por el OCR afecta el desempeño en categorías con mayor variabilidad estilística. El valor de *accuracy* alcanzado fue de 0.950, constituyendo una base sólida pero inferior al obtenido con información visual. En la Figura 5.7 se presenta un ejemplo de clasificación sobre las fichas de AFE para el modelo entrenado con tasa 0.01.

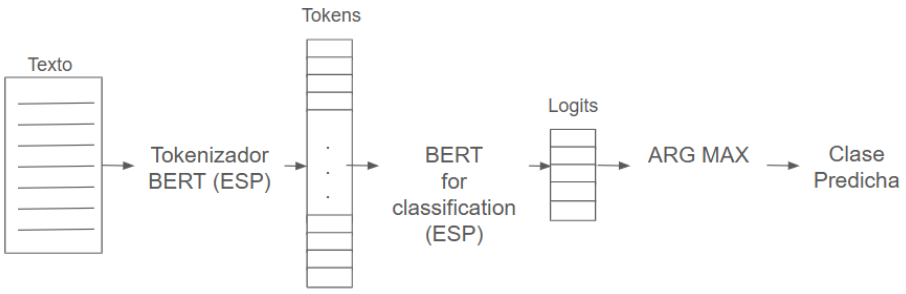


Figura 5.6: Berrutti Modelo Textos Multiclase. El modelo toma como entrada el texto de cada documento, genera una secuencia de tokens mediante el tokenizador BERT y obtiene un vector de 1.000 dimensiones utilizando BERT for Classification. Este vector pasa por una capa lineal que produce un *score* para cada clase.

### 5.2.2. Imágenes

Para la modalidad visual se utilizó EfficientNet-B0, ajustada a la tarea multiclase. Las imágenes se procesaron en escala de grises y redimensionadas a la resolución

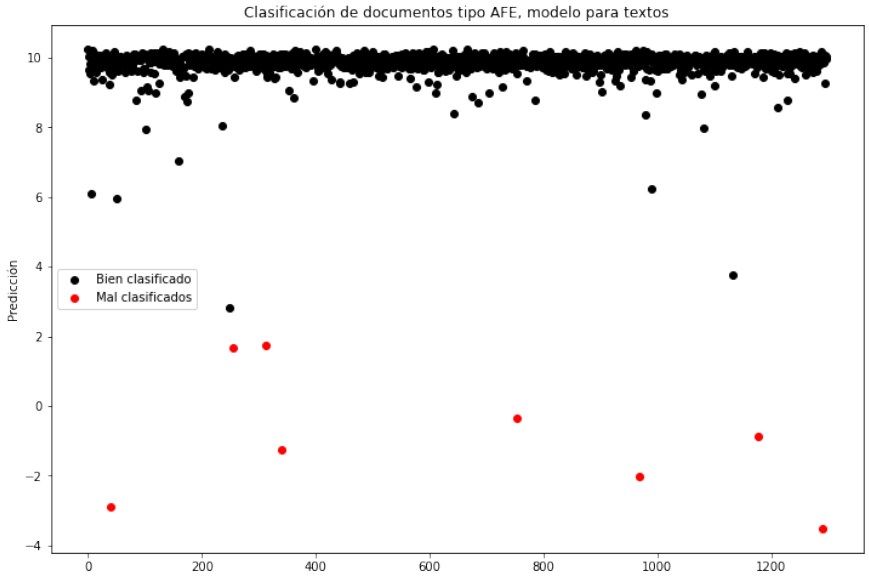


Figura 5.7: Modelo multiclase, para textos, clasificación de fichas de AFE

de entrada del modelo (400 píxeles de ancho y 400 píxeles de alto). Se aplicaron técnicas de *data augmentation* durante el entrenamiento para mejorar la capacidad de generalización.

Tabla 5.7: *Accuracy* del modelo de imágenes para distintas tasas de aprendizaje

| Tasa de aprendizaje | Entrenamiento | Validación |
|---|---|---|
| 0.0005 | 0.645 | 0.618 |
| 0.001 | 0.723 | 0.714 |
| 0.005 | 0.927 | 0.907 |
| 0.01 | 0.949 | 0.725 |
| 0.05 | 0.981 | 0.980 |
| 0.1 | 0.985 | 0.889 |
| 0.25 | 0.982 | 0.968 |
| 0.5 | 0.980 | 0.985 |
| 0.75 | 0.980 | 0.988 |
| 0.9 | 0.974 | 0.978 |
| 1 | 0.977 | 0.987 |

Como se muestra en la Tabla 5.7, el modelo visual es más robusto frente a variaciones en la tasa de aprendizaje que el modelo textual, alcanzando desempeños razonables en un rango amplio de valores. Mientras que las tasas muy bajas (0,0005 y 0,001) generan convergencia lenta y un *accuracy* insuficiente, los valores comprendidos entre 0,005 y 1,0 permiten obtener resultados estables y altos. El mejor rendimiento se observa en torno a 0,75, donde el modelo alcanza su máximo *accuracy*, lo que indica que la arquitectura EfficientNet puede aprovechar tasas más agresivas sin perder capacidad de generalización.

El modelo visual alcanzó un *accuracy* de 0.988, superando ampliamente al modelo textual. Este resultado confirma que, en el Archivo Berrutti, la estructura gráfica de los documentos constituye una fuente de información más estable y menos ruidosa que el texto extraído por OCR.

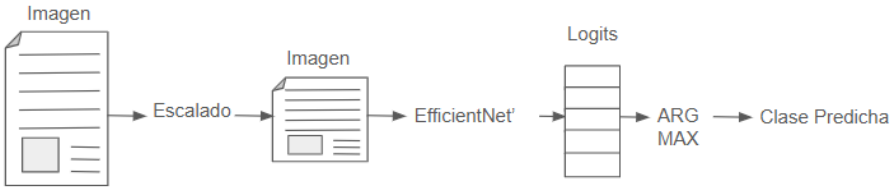


Figura 5.8: Berrutti Modelo Imágenes Multiclase. El modelo recibe como entrada una imagen procesada del documento. Esta se procesa mediante EfficientNet, obteniéndose un vector de 1.000 características. Este vector pasa por una capa lineal que produce un *score* para cada clase.

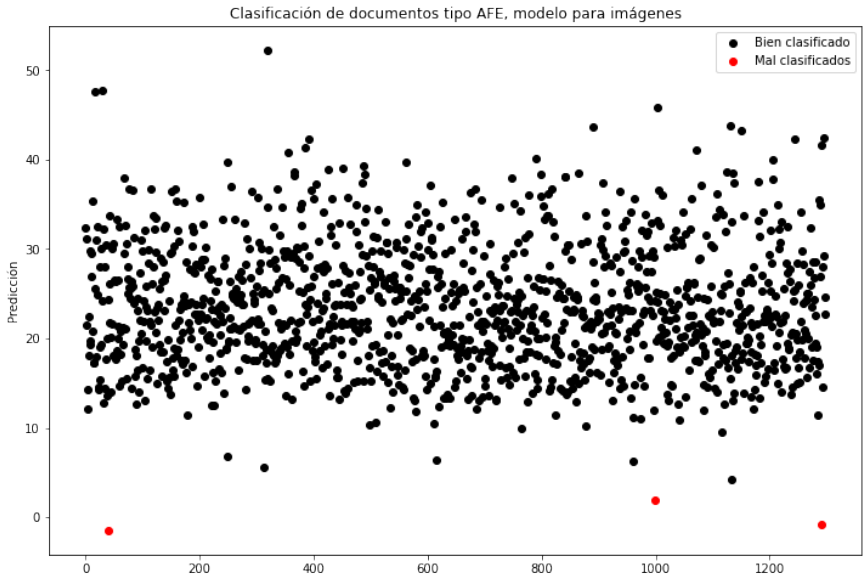


Figura 5.9: Modelo multiclase, para imágenes, clasificación de fichas de AFE

En la Figura 5.9 se muestra la salida del modelo sobre documentos del conjunto de validación pertenecientes a la clase fichas de AFE, empleando la mejor configuración encontrada.

### 5.2.3. Combinado: Promedio de modelos

Como se muestra en la Figura 5.10, a partir de los modelos unimodales entrenados, se implementó un esquema de fusión tardía basado en el promedio simple de las salidas de probabilidad. Para cada documento, se calcularon las predicciones de los modelos de texto e imagen, y se promediaron para obtener una salida final.

El promedio logró mejorar ligeramente el desempeño respecto a cada modalidad por separado, alcanzando un *accuracy* de 0.990. Este resultado muestra que incluso

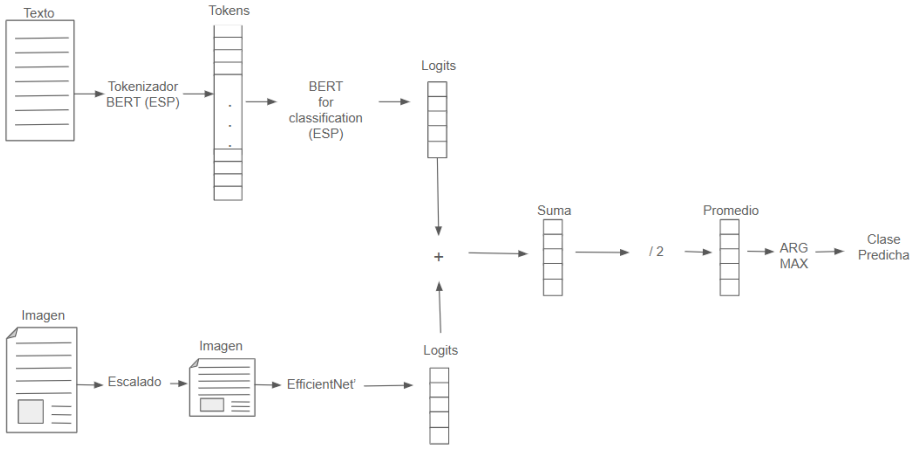


Figura 5.10: Berrutti Modelo Promedio Multiclase. Cada documento se procesa en paralelo por los modelos de texto e imagen. Las salidas numéricas generadas por ambos clasificadores se promedian para obtener un *score* final.

un método de combinación sencillo puede aportar robustez adicional, especialmente en clases donde el texto y la imagen capturan aspectos complementarios.

En la Figura 5.11 se presenta la matriz de confusión correspondiente, donde se observa un desempeño general satisfactorio. La mayoría de los documentos mal clasificados (29) son identificados como actas de interrogatorios cuando en realidad son documentos sin asignar a ninguna clase. A su vez, 13 de las actas de interrogatorio no son identificadas y se las clasifica como sin clase. Identificar las actas de interrogatorio es el problema más desafiante entre las clases utilizadas. Por otro lado, el modelo logra distinguir adecuadamente entre las distintas clases sin mostrar una penalización evidente hacia aquellas con menor cantidad de ejemplos.

### 5.2.4. Combinado: Modelo híbrido

Finalmente, se entrenó un modelo híbrido en configuración de fusión temprana. El texto fue procesado con BERT y las imágenes con EfficientNet, concatenando ambas representaciones en una capa lineal para obtener los *logits* de salida (Figura 5.12).

Como se observa en la Tabla 5.8, el modelo híbrido muestra un comportamiento estable para tasas de aprendizaje comprendidas entre 0,0005 y 0,1, mientras que valores superiores (0,5) impiden la convergencia del entrenamiento. Dentro del rango efectivo, la tasa de 0,1 resulta especialmente adecuada, alcanzando un *accuracy* de 0,984 en entrenamiento y manteniendo una capacidad de generalización elevada en validación (0,991). Este resultado confirma que la combinación de modalidades, cuando se optimiza correctamente, puede superar el desempeño de los modelos unimodales y del modelo basado en el promedio.

El híbrido alcanzó un *accuracy* de 0.991, el mejor desempeño global entre los experimentos multiclase. Sin embargo, la diferencia respecto al promedio es mínima, lo que sugiere que con el volumen de datos disponible la fusión tardía es suficiente para obtener un resultado competitivo. El híbrido tiene mayor potencial en contextos con más datos, donde puede explotar mejor las interacciones entre modalidades.

La matriz de confusión para el conjunto de validación (Figura 5.13) muestra un alto nivel de *accuracy* en todas las clases, confirmando la capacidad del modelo híbrido

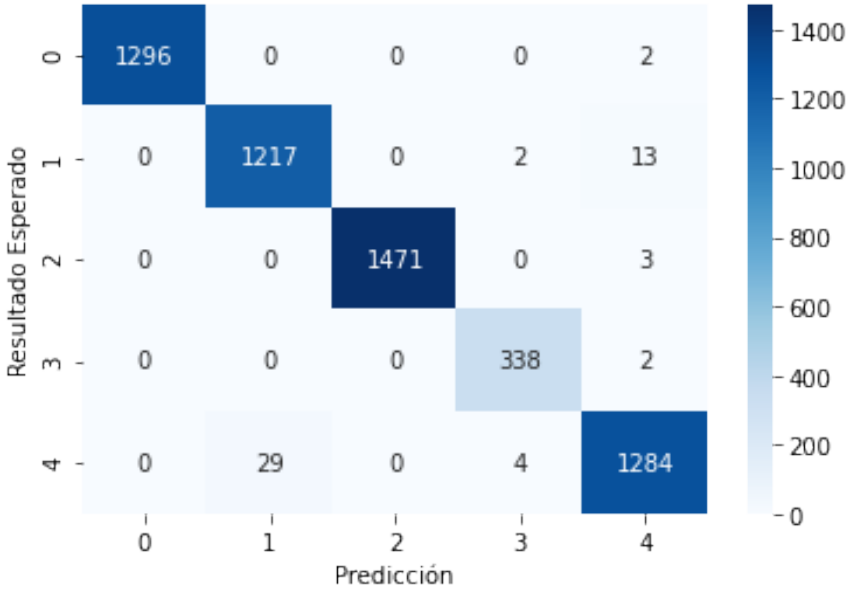


Figura 5.11: Matriz de confusión, clasificación por promedios. La matriz muestra un buen desempeño para todas las clases.

Tabla 5.8: *Accuracy* del modelo híbrido para distintas tasas de aprendizaje

| Tasa de aprendizaje | Entrenamiento | Validación |
|---|---|---|
| 0.0005 | 0.912 | 0.915 |
| 0.001 | 0.937 | 0.929 |
| 0.005 | 0.965 | 0.948 |
| 0.01 | 0.966 | 0.914 |
| 0.05 | 0.981 | 0.975 |
| 0.1 | 0.984 | 0.991 |
| 0.5 | 0.229 | 0.229 |

para aprovechar información complementaria proveniente del texto y de la imagen. El error para las actas de interrogatorio es similar al del modelo por promedio, donde los mismos 29 documentos son confundidos con actas cuando no lo son, pero mejoran de 13 a 10 las actas identificadas de forma errónea como documentos sin clase.

### 5.2.5. Resumen y discusión

En conjunto, los experimentos multiclase confirman la ventaja de los enfoques multimodales. El modelo textual alcanzó 0.950 de *accuracy*, mientras que el visual llegó a 0.988. Al combinar ambos, el promedio alcanzó 0.990 y el híbrido 0.991. Aunque las mejoras absolutas parecen pequeñas, son significativas considerando que los unimodales ya tenían un desempeño muy alto.

Estos resultados muestran que, en el Archivo Berrutti, la información visual suele ser más estable que la textual, pero que la combinación de modalidades aporta un

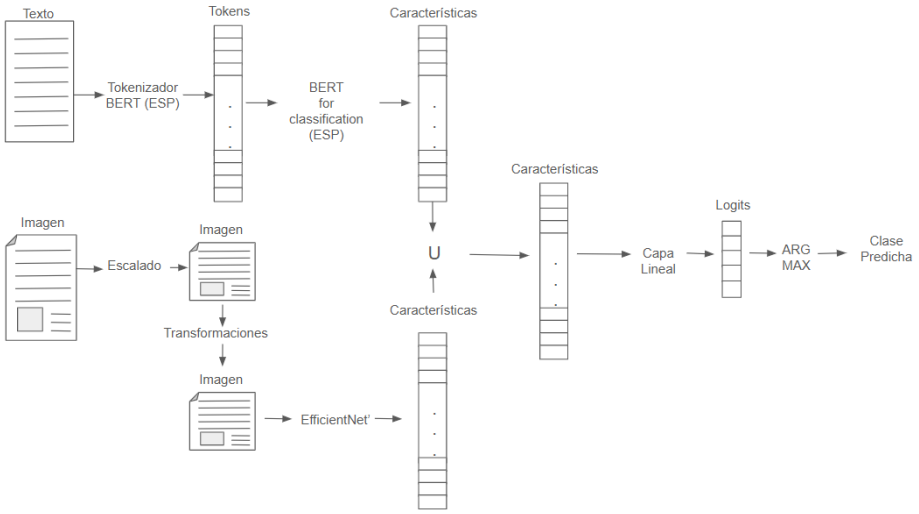


Figura 5.12: Berrutti Modelo Híbrido Multiclase: se extraen representaciones textuales utilizando *BERT for classification* y representaciones visuales a partir de una versión modificada de *EfficientNet*. Ambas representaciones se concatenan y se utilizan como entrada para una capa lineal de clasificación.

beneficio adicional. El promedio se destaca como una alternativa simple y eficaz, mientras que el híbrido plantea un camino prometedor en caso de contar con un volumen mayor de datos.

Con base en estos resultados, se entrenó nuevamente el modelo híbrido utilizando tanto los conjuntos de entrenamiento como los de validación y se evaluó sobre el conjunto de test. El *accuracy* alcanzado fue de 0.992, lo que indica que el modelo no solo es el más preciso, sino que también generaliza adecuadamente.

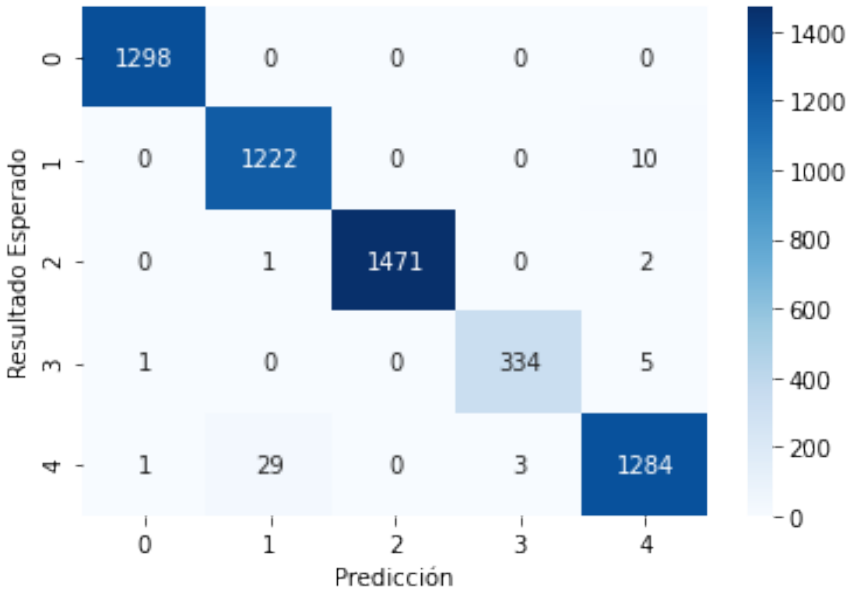


Figura 5.13: Matriz de confusión, modelo híbrido. La matriz muestra un buen desempeño para todas las clases.

Esta página ha sido intencionalmente dejada en blanco.

# Capítulo 6

# Conclusiones y trabajos futuros

El trabajo desarrollado en esta tesis abordó el problema de la clasificación automática de documentos en el contexto del Archivo Berrutti, explorando tanto métodos unimodales (texto o imagen) como multimodales (texto e imagen). Desde el inicio, la motivación principal no fue únicamente académica, sino práctica: aportar soluciones concretas al proyecto de digitalización del archivo, vinculado a la documentación del terrorismo de Estado en Uruguay. Estos documentos no solo tienen un valor histórico, sino también jurídico, político y social, como insumo para la memoria colectiva y las luchas por la verdad y la justicia.

En este marco, los experimentos mostraron que, si bien cada modalidad por separado alcanza desempeños elevados, la integración de ambas constituye la estrategia más robusta, en particular para categorías complejas y heterogéneas como las actas de la OCOA Entre los principales aportes de este trabajo se destacan: la evaluación comparativa de modelos basados en texto, imagen y combinaciones multimodales; la validación de que los tokenizadores preentrenados resultan más efectivos que los reentrenados desde cero en contextos de datos limitados; y la confirmación de que arquitecturas modernas como BERT y EfficientNet son adecuadas para la clasificación de documentos históricos de esta naturaleza.

Además, la experimentación con distintos esquemas de fusión mostró que incluso métodos sencillos, como el promedio de salidas, pueden superar a los modelos unimodales y que los modelos híbridos presentan un potencial significativo cuando se dispone de un mayor volumen de datos. Los resultados alcanzados fueron satisfactorios: se obtuvieron valores de *accuracy* cercanos al 0.99 en los mejores experimentos, lo que confirma la viabilidad de aplicar enfoques de aprendizaje profundo en la organización y análisis de este acervo documental. Más allá de las métricas, la experiencia acumulada permitió identificar ventajas y limitaciones de cada enfoque, aportando insumos valiosos para futuros desarrollos en el área.

## Trabajos futuros

Si bien los avances obtenidos son alentadores, aún quedan abiertas varias líneas de investigación. Entre las más relevantes se destacan:

- Extender los experimentos a un conjunto más amplio y diverso de categorías documentales del Archivo Berrutti, para validar la escalabilidad de los modelos.
- Ampliar progresivamente el número de clases de documentos con el objetivo de tender hacia la clasificación integral de todo el archivo, avanzando hacia un

sistema automatizado que organice de manera exhaustiva el acervo.

- Incorporar modelos de lenguaje más recientes, como RoBERTa o DistilBERT, y arquitecturas visuales avanzadas como ResNet o Vision Transformers, para comparar su desempeño con BERT y EfficientNet.
- Profundizar en métodos híbridos que integren no solo texto e imagen, sino también metadatos documentales (fechas, firmas, sellos) como fuente adicional de información.
- Desarrollar sistemas que permitan interpretar las decisiones de los modelos, facilitando su uso en investigaciones históricas y jurídicas.

En síntesis, este trabajo no solo aporta resultados concretos en la clasificación documental, sino que también se inscribe en un esfuerzo colectivo por organizar y hacer accesibles los archivos del terrorismo de Estado. En este sentido, los avances aquí presentados constituyen un paso inicial hacia sistemas más completos y escalables, que podrán contribuir a la memoria, la investigación académica y la construcción de verdad y justicia en Uruguay.

# Referencias


[1] Cruzar – Archivos del pasado reciente. (s.f.). Consultado el 5 de julio de 2023, desde https://cruzar.edu.uy/

[2] Sitios de Memoria Uruguay. Archivo Berrutti. Disponible en: `https://sitiosdememoria.uy/origen/archivo-berrutti`. Accedido: septiembre 2025.

[3] Smith, R. (2007). An overview of the Tesseract OCR engine. Consultado el 3 de septiembre de 2025, desde https://research.google.com/pubs/archive/33418.pdf

[4] Wick, C., Reul, C., Puppe, F. (2020). *Calamari – A High-Performance Tensorflow-based Deep Learning Package for Optical Character Recognition*. Digital Humanities Quarterly, 14(1). Consultado el 3 de septiembre de 2025, desde https://digitalhumanities.org/dhq/ (o desde el enlace PDF correspondiente)

[5] Etcheverry, L, Agorio, L, Bacigalupe, V, Barreiro, S, Bing, E, Blixen, S, Calegari, D, Cardozo, L, Carpani, F, Chavat, F, Garat, D, Gómez, A, Fernández, E, Fioritto, F, Hernánez, F, Laguna, R, Marabotto, V, Moncecchi, G, Ramírez, I, Rosa, A, Stabile, J, Tiscornia, J, Patiño, N, Rivero, L, Wonsever, D, Zorron, G y Randall, G. (2021.). A computational framework for the analysis of the Uruguayan dictatorship archives. Qurator 2021 - Conference on Digital Curation Technologies.

[6] Nogueira, M. (2023). Construcción de herramientas para contribuir al análisis de los archivos de la OCOA.

[7] Harley, A. W., Ufkes, A., & Derpanis, K. G. (2015). Evaluation of deep convolutional nets for document image classification and retrieval. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)* (pp. 991–995). IEEE.

[8] S. Harley, A. Ufkes y K. Derpanis. *Evaluation of Deep Convolutional Networks for Document Image Classification.* Proceedings of the International Conference on Document Analysis and Recognition (ICDAR), 2015. Disponible en: `https://www.cs.cmu.edu/~aharley/rvl-cdip/`

[9] Afzal, M. Z., Kölsch, A., Ahmed, S., & Liwicki, M. (2017). Cutting the error by half: Investigation of very deep CNN and advanced training strategies for document image classification. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)* (Vol. 1, pp. 883–888). IEEE.

[10] Denk, T., & Reisswig, C. (2019). BERTgrid: Contextualized embedding for 2D document representation and understanding. In *NeurIPS Workshop on Document Intelligence*.

[11] Adhikari, A., Ram, A., Tang, R., & Lin, J. (2019). DocBERT: BERT for Document Classification. *arXiv preprint arXiv:1904.08398*.

[12] Chen, Q., Wang, W., & Chang, B. (2019). Investigating BERT for text classification. *arXiv preprint arXiv:1905.05583*.

[13] Shao, Y., & Wang, R. (2021). Document classification with pre-trained language models. *Journal of Information Science*, 47(1), 56–68.

[14] Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., & Zhou, M. (2020). LayoutLM: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 1192–1200).

[15] Xu, Y., Xu, Y., Lv, T., Cui, L., Wei, F., Wang, G., ... & Zhou, M. (2021). LayoutLMv2: Multi-modal Pre-training for Visually-rich Document Understanding. *Proceedings of ACL*.

[16] Huang, Y., Zhang, Y., Xu, Y., Xu, Y., Cui, L., Wei, F., ... & Zhou, M. (2022). LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking. *Proceedings of ACL*.

[17] Javier Ferrando, Juan Luis Dominguez, Jordi Torres, Raul Garcia, David Garcia, Daniel Garrido, Jordi Cortada, Mateo Valero (2020). Improving accuracy and speeding up Document Image Classification through parallel systems.

[18] Audebert, N., Herold, C., Slimani, K., & Vidal, C. (2020). Multimodal deep networks for text and image-based document classification. In Machine Learning and Knowledge Discovery in Databases: International Workshops of ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I (pp. 427-443). Springer International Publishing.

[19] Appalaraju, S., Jasani, B., Krishnan, R., & Manmatha, R. (2021). DocFormer: End-to-End Transformer for Document Understanding. *Proceedings of ICCV*.

[20] Appalaraju, S., Krishnan, R., Manmatha, R., & Choudhury, S. (2023). DocFormerv2: Local-Global Document Transformer for Efficient Pre-training on Long Documents. *arXiv preprint arXiv:2306.01733*.

[21] Liu, S., Chen, Z., Zhang, H., & Zhang, Y. (2023). DocLLM: A Multimodal Language Model for Document Understanding. *arXiv preprint arXiv:2401.00908*.

[22] Liao, Y., Xu, H., Li, B., ... & Sun, X. (2025). DocLayLLM: An Efficient Multimodal Extension of Large Language Models for Document Understanding. *Proceedings of CVPR 2025*.

[23] Duan, Y., Wang, J., Chen, H., ... & Zhang, L. (2025). Docopilot: Improving Multimodal Models for Document-Level Understanding. *Proceedings of CVPR 2025*.

[24] Nesmachnow, S., y Iturriaga, S. (2019). Cluster-UY: Collaborative scientific high performance computing in uruguay. En M. Torres y J. Klapp (Eds.), Supercomputing (pp. 188-202). Springer International Publishing.

[25] G. Salton, *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*, Addison-Wesley, Reading, MA, 1988.

[26] T. Mikolov, K. Chen, G. Corrado, y J. Dean, "Efficient Estimation of Word Representations in Vector Space," en *Proceedings of Workshop at ICLR*, 2013. Disponible en: `https://arxiv.org/abs/1301.3781`

[27] J. Pennington, R. Socher, y C. D. Manning, "GloVe: Global Vectors for Word Representation," en *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014. DOI: 10.3115/v1/D14-1162.

[28] J. Devlin, M.-W. Chang, K. Lee, y K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," en *Proceedings of NAACL-HLT*, pp. 4171–4186, 2019. Disponible en: `https://arxiv.org/abs/1810.04805`

[29] M. Tan y Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," en *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pp. 6105–6114, 2019. Disponible en: `https://arxiv.org/abs/1905.11946`

[30] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

[31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, y A. Rabinovich, "Going deeper with convolutions," en *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2015. DOI: 10.1109/CVPR.2015.7298594

[32] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, y V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," *arXiv preprint*, 2019. Disponible en: `https://arxiv.org/abs/1907.11692`

[33] V. Sanh, L. Debut, J. Chaumond, y T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," en *Proceedings of NeurIPS EMC2 Workshop*, 2019. Disponible en: `https://arxiv.org/abs/1910.01108`

[34] Kumar, J., Ye, P., y Doermann, D. (2014). Learning document structure for retrieval and classification. En *2014 11th IAPR International Workshop on Document Analysis Systems* (pp. 54-58). IEEE. https://doi.org/10.1109/DAS.2014.21

[35] Koelsch, F., Ebbecke, M., Ewerth, R. (2017). Towards Layout-Aware Document Categorization. *Proceedings of the 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 383–388.

[36] Wang, Z., Huang, Y., & Li, J. (2024). WordVIS: Multimodal Document Classification with Visual and Textual Features. *Pattern Recognition*, 150, 110345.

[37] BJ, B. N., & Yadhukrishnan, S. (2023, August). A Comparative Study on Document Images Classification Using Logistic Regression and Multiple Linear Regressions. In 2023 Second International Conference on Augmented Intelligence and Sustainable Systems (ICAISS) (pp. 1096-1104). IEEE.

[38] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.

[39] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, 30.

[40] Adhikari, A., Ram, A., Tang, R., & Lin, J. (2019). Docbert: Bert for document classification. arXiv preprint arXiv:1904.08398.

[41] Ding, M., Zhou, C., Yang, H., & Tang, J. (2020). Cogltx: Applying bert to long texts. Advances in Neural Information Processing Systems, 33, 12792-12804.

Esta página ha sido intencionalmente dejada en blanco.

# Capítulo 7

# Script de Entrenamiento

## 7.1. Constantes y configuración inicial

Este bloque define las importaciones, constantes de rutas para el almacenamiento temporal y para la carpeta de imagenes redimensionadas a 400×400, además de dependencias auxiliares utilizadas en el flujo de trabajo.

```
#!/usr/bin/env python3

from __future__ import annotations

# ===== Standard imports =====
import os
import os.path as op
import shutil
from typing import List, Tuple, Any

# ===== Third-party =====
import numpy as np
import pandas as pd
import torch
import torch.nn as nn
import torch.nn.functional as F
import torchmetrics
import timm
from PIL import Image
from tqdm import tqdm
from torch.utils.data import DataLoader, Dataset

import pytorch_lightning as L
from pytorch_lightning.loggers import CSVLogger
from torch_lr_finder import LRFinder

from transformers import AutoTokenizer
from transformers import AutoModelForSequenceClassification

import matplotlib.pyplot as plt  # LR Finder curve
```

```
from torchvision import transforms

TMP: str = "/scratch/juan.damian.pintos"
TMP_IMG_REDIMENSIONADAS: str = op.join(TMP, "400x400")

# (Optional) Folder with tokenized tensors (.pt)
DATA_TOKENIZADO: str = "datos/calamaritokenizado"
```

## 7.2. Definición del modelo (`luisamodel`)

El modelo admite tres configuraciones: solo texto (BERT), solo imagen (EfficientNet-B0 adaptada a escala de grises) o combinación (concatenación de logits de texto y *features* de imagen).

```
class luisamodel(torch.nn.Module):
    """
    Modes:
      - Text only (BERT) when TXT=True, IMG=False
      - Image only (EfficientNet-B0, 1 channel) when TXT=False
        , IMG=True
      - Combined (text logits + image features) when TXT=True,
         IMG=True
    """

    def __init__(self, num_classes: int):
        super().__init__()
        if TXT and not IMG:
            self.bert = AutoModelForSequenceClassification.
                from_pretrained(
                "dccuchile/bert-base-spanish-wwm-uncased",
                    num_labels=num_classes
                )
        elif not TXT and IMG:
            self.efficientnet = timm.create_model("
                efficientnet_b0", pretrained=False)
            self.efficientnet.conv_stem = torch.nn.Conv2d(
                1, 32, kernel_size=3, stride=2, padding=1,
                    bias=False
                )
            num_features = self.efficientnet.classifier.
                in_features
            self.efficientnet.classifier = nn.Sequential(
                nn.Dropout(p=0.2, inplace=True), nn.Linear(
                    num_features, num_classes)
                )
        else:
            self.bert = AutoModelForSequenceClassification.
                from_pretrained(
                "dccuchile/bert-base-spanish-wwm-uncased",
                    num_labels=128
```

```
            )
            self.efficientnet = timm.create_model("
                efficientnet_b0", pretrained=False)
            self.efficientnet.conv_stem = torch.nn.Conv2d(
                1, 32, kernel_size=3, stride=2, padding=1,
                    bias=False
            )
            self.efficientnet.classifier = nn.Identity()
            self.efficientnet_out_features = self.efficientnet
                .num_features
            self.combined_features_size = 128 + self.
                efficientnet_out_features
            self.classifier = nn.Linear(self.
                combined_features_size, num_classes)

    def forward(self, entrada: List[torch.Tensor]) -> torch.
        Tensor:
        input_ids = entrada[0]
        attention_mask = entrada[1]
        img = entrada[2]

        if TXT and not IMG:
            outputsb = self.bert(input_ids=input_ids,
                attention_mask=attention_mask)
            return outputsb.logits
        elif not TXT and IMG:
            logits = self.efficientnet(img)
            return logits
        else:
            text_outputs = self.bert(input_ids=input_ids,
                attention_mask=attention_mask)
            img_features = self.efficientnet(img)
            combined_features = torch.cat((text_outputs.logits
                , img_features), dim=1)
            logits = self.classifier(combined_features)
            return logits
```

## 7.3. *Dataset* y *DataModule*

Se definen utilidades para generar rutas, copiar archivos al *scratch*, redimensionar imagenes y producir lotes. El `DataModule` aplica *augmentations* leves en entrenamiento y normalización en validación/prueba.

```
def cambiar_extension(nombre_archivo: str, nueva_extension:
    str = ".pt") -> str:
    nombre_base, _ = os.path.splitext(nombre_archivo)
    return f"{nombre_base}_512{nueva_extension}"


class HojasDataset(Dataset):
```

```
    """
    Reads a CSV with columns: carpeta, imagen, categoria.
    Creates .pt column with cambiar_extension().
    Copies tensors/images to scratch and resizes images to 400
        x400.
    """

    def __init__(self, csv_path: str, transform=None, scratch:
        bool = True, achicar: bool = True):
        df = pd.read_csv(csv_path)
        df["pt"] = df["imagen"].apply(cambiar_extension)
        self.transform = transform
        self.carpeta = df["carpeta"]
        self.img_names = df["imagen"]
        self.pt = df["pt"]
        self.labels = df["categoria"]
        self.scratch = scratch
        self.isachicar = achicar
        if self.scratch:
            self.toscratchall()
            if achicar:
                self.achicarall()

    @staticmethod
    def crearcarpeta(directory_path: str) -> None:
        os.makedirs(directory_path, exist_ok=True)

    @staticmethod
    def achicar(carpeta_origen: str, carpeta_destino: str,
        archivo: str, nuevo_ancho: int, nuevo_alto: int) ->
        None:
        destino = op.join(carpeta_destino, archivo)
        if not os.path.isfile(destino):
            imagen_tif = Image.open(op.join(carpeta_origen,
                archivo)).convert("L")
            nuevo_tamano = (nuevo_ancho, nuevo_alto)
            imagen_redimensionada = imagen_tif.resize(
                nuevo_tamano, Image.Resampling.LANCZOS)
            imagen_redimensionada.save(destino)

    def achicarall(self) -> None:
        self.crearcarpeta(TMP_IMG_REDIMENSIONADAS)
        for _, imagen in tqdm(enumerate(self.img_names), desc=
            "Redimensionando", total=len(self.img_names)):
            self.achicar(TMP, TMP_IMG_REDIMENSIONADAS, imagen,
                400, 400)

    @staticmethod
    def toscratch(carpeta: str, imagen: str) -> None:
        origen = op.join(carpeta, imagen)
        destino = op.join(TMP, imagen)
        if os.path.exists(origen) and not os.path.exists(
```

```
            destino):
            shutil.copyfile(origen, destino)

    def toscratchall(self) -> None:
        for _, ruta in tqdm(enumerate(self.pt), desc="Copiando
            .pt", total=len(self.pt)):
            self.toscratch(DATA_TOKENIZADO, ruta)
        for i, _ in tqdm(enumerate(self.carpeta), desc="
            Copiando imagenes", total=len(self.carpeta)):
            self.toscratch(self.carpeta[i], self.img_names[i])

    def __getitem__(self, index: int) -> Tuple[List[torch.
        Tensor], Any]:
        if self.scratch:
            contenido = torch.load(op.join(TMP, self.pt[index
                ]))
        else:
            contenido = torch.load(op.join(DATA_TOKENIZADO,
                self.pt[index]))

        label = self.labels[index]
        input_ids = contenido["input_ids"].squeeze()
        attention_mask = contenido["attention_mask"].squeeze()

        if self.scratch:
            if self.isachicar:
                img_path = op.join(TMP_IMG_REDIMENSIONADAS,
                    self.img_names[index])
            else:
                img_path = op.join(TMP, self.img_names[index])
        else:
            img_path = op.join(self.carpeta[index], self.
                img_names[index])

        img = Image.open(img_path)
        if self.transform is not None:
            img = self.transform(img)

        return [input_ids, attention_mask, img], label

    def __len__(self) -> int:
        return self.labels.shape[0]


class HojasDataModule(L.LightningDataModule):
    def __init__(self):
        super().__init__()
        self.data_transforms = {
            "train": transforms.Compose([
                transforms.ColorJitter(brightness=0.1,
                    contrast=0.1, saturation=0.1, hue=0.1),
                transforms.RandomAffine(degrees=8, translate
```

```
                    =(0.1, 0.1), scale=(0.9, 1.1)),
                transforms.ToTensor(),
                transforms.Lambda(lambda x: x + torch.
                    randn_like(x) * 0.1),
                transforms.Normalize((0.5,), (0.5,)),
            ]),
            "test": transforms.Compose([
                transforms.ToTensor(),
                transforms.Normalize((0.5,), (0.5,)),
            ]),
        }

    def prepare_data(self):
        return

    def setup(self, stage=None):
        return

    def train_dataloader(self) -> DataLoader:
        train_dataset = HojasDataset(
            csv_path="datos/m_entrenamientoyval.csv",
            transform=self.data_transforms["train"],
        )
        return DataLoader(dataset=train_dataset, batch_size
            =16, shuffle=True, num_workers=16)

    def val_dataloader(self) -> DataLoader:
        val_dataset = HojasDataset(csv_path="datos/m_test.csv"
            , transform=self.data_transforms["test"])
        return DataLoader(dataset=val_dataset, batch_size=16,
            shuffle=False, num_workers=16)

    def test_dataloader(self) -> DataLoader:
        test_dataset = HojasDataset(csv_path="datos/m_test.csv
            ", transform=self.data_transforms["test"])
        return DataLoader(dataset=test_dataset, batch_size=16,
             shuffle=False, num_workers=16)
```

## 7.4. *LightningModule* y utilidades

El módulo Lightning centraliza perdidas, metricas y el ciclo de entrenamiento/validación/prueba. Se incluye una utilitaria para ejecutar y graficar el *LR Finder*.

```
class LightningModel(L.LightningModule):
    def __init__(self, model: torch.nn.Module, learning_rate:
        float, num_classes: int):
        super().__init__()
        self.learning_rate = learning_rate
        self.model = model
        self.save_hyperparameters(ignore=["model"])
```

```
        self.num_classes = num_classes
        if num_classes == 1:
            self.train_acc = torchmetrics.Accuracy(task="
                binary")
            self.val_acc = torchmetrics.Accuracy(task="binary"
                )
            self.test_acc = torchmetrics.Accuracy(task="binary
                ")
        else:
            self.train_acc = torchmetrics.Accuracy(task="
                multiclass", num_classes=num_classes)
            self.val_acc = torchmetrics.Accuracy(task="
                multiclass", num_classes=num_classes)
            self.test_acc = torchmetrics.Accuracy(task="
                multiclass", num_classes=num_classes)

    def loss_fn(self, logits: torch.Tensor, true_labels: torch
        .Tensor) -> torch.Tensor:
        if self.num_classes == 1:
            return F.binary_cross_entropy_with_logits(logits,
                true_labels.float())
        return F.cross_entropy(logits, true_labels)

    def plot_and_save(self, lr_finder: LRFinder, file_path:
        str, skip_start: int = 10, skip_end: int = 5,
                      log_lr: bool = True, show_lr: float |
                          None = None, suggest_lr: bool = True
                          ):
        fig, ax = plt.subplots()
        ax, suggested_lr = lr_finder.plot(
            skip_start=skip_start,
            skip_end=skip_end,
            log_lr=log_lr,
            show_lr=show_lr,
            suggest_lr=suggest_lr,
            ax=ax,
        )
        fig.savefig(file_path)
        plt.close(fig)
        return suggested_lr

    def find_learning_rate(self, datamodule: L.
        LightningDataModule, min_lr: float = 1e-7, max_lr:
        float = 10):
        optimizer = torch.optim.SGD(self.parameters(), lr=
            min_lr)
        lr_finder = LRFinder(self, optimizer, self.loss_fn,
            device=self.device)
        lr_finder.range_test(datamodule.train_dataloader(),
            end_lr=max_lr, num_iter=100)
        file_path = "lr_finder_plot.png"
        suggested_lr = self.plot_and_save(lr_finder, file_path
```

```
            )
        print(f"Suggested learning rate: {suggested_lr}")
        return suggested_lr

    def forward(self, entrada: List[torch.Tensor]) -> torch.
        Tensor:
        return self.model(entrada)

    def _shared_step(self, batch: Tuple[List[torch.Tensor],
        torch.Tensor]):
        entrada, true_labels = batch
        logits = self(entrada)
        if self.num_classes == 1:
            true_labels = true_labels.unsqueeze(1)
            predicted_labels = (logits > 0.5).float()
        else:
            predicted_labels = torch.argmax(logits, dim=1)
        loss = self.loss_fn(logits, true_labels)
        return loss, true_labels, predicted_labels

    def training_step(self, batch, batch_idx):
        loss, true_labels, predicted_labels = self.
            _shared_step(batch)
        self.log("train_loss", loss)
        self.train_acc(predicted_labels, true_labels)
        self.log("train_acc", self.train_acc, prog_bar=True,
            on_epoch=True, on_step=False)
        return loss

    def validation_step(self, batch, batch_idx):
        loss, true_labels, predicted_labels = self.
            _shared_step(batch)
        self.log("val_loss", loss, prog_bar=True)
        self.val_acc(predicted_labels, true_labels)
        self.log("val_acc", self.val_acc, prog_bar=True)

    def test_step(self, batch, batch_idx):
        loss, true_labels, predicted_labels = self.
            _shared_step(batch)
        self.test_acc(predicted_labels, true_labels)
        self.log("test_acc", self.test_acc)

    def configure_optimizers(self):
        optimizer = torch.optim.SGD(self.parameters(), lr=self
            .learning_rate)
        return optimizer
```

## 7.5. Bloque principal de entrenamiento

Ejecución reproducible del flujo: selección de modalidad (`IMG/TXT`), *seed*, tasa de aprendizaje, *logger*, entrenamiento y guardado de pesos.

```
IMG: bool = True
TXT: bool = True

if __name__ == "__main__":
    print(f"IMG {IMG} TXT {TXT}")

    infijo = "combinado"
    if not IMG:
        infijo = "txt"
    if not TXT:
        infijo = "img"

    L.seed_everything(22)
    lr = 0.1
    logs = f"m_{infijo}_todo_{lr}"

    model = luisamodel(5)
    dm = HojasDataModule()

    lightning_model = LightningModel(model=model,
        learning_rate=lr, num_classes=5)

    trainer = L.Trainer(
        max_epochs=5,
        accelerator="gpu",
        precision="bf16",
        devices="auto",
        logger=CSVLogger(save_dir="logs/", name=logs),
        deterministic=True,
    )

    trainer.fit(model=lightning_model, datamodule=dm)

    # Save weights
    os.makedirs("modelos", exist_ok=True)
    torch.save(lightning_model.state_dict(), f"modelos/{logs}"
        )
```

Esta página ha sido intencionalmente dejada en blanco.

# Capítulo 8

# Anexo: Categorías del Archivo Berrutti

Tabla 8.1: Categorías y Subcategorías de Documentos etiquetados en Facultad de Información y Comunicación

| Categoría | Subcategoría | Cantidad |
|---|---|---|
| Comunicado | - | 216 |
| Comunicado | Comunicado de prensa | 20 |
| Comunicado | Comunicado de prensa de las FFCC | 39 |
| Comunicado | Comunicado especial de las FFCC | 22 |
| Documentos tipo 2 | - | 2796 |
| Documentos tipo 2 | Expediente | 446 |
| Documentos tipo 2 | Ficha decidactilardecadactilar | 106 |
| Documentos tipo 2 | Ficha personal Patronímica | 3189 |
| Documentos tipo 2 | Folleto | 305 |
| Documentos tipo 2 | Formulario Solicitud de Beca | 2609 |
| Documentos tipo 2 | Fotografía | 66 |
| Documentos tipo 2 | Legajo | 13 |
| Documentos tipo 2 | Libro | 532 |
| Documentos tipo 2 | Lista de votación | 233 |
| Documentos tipo 2 | Normas, leyes y procedimientos | 148 |
| Documentos tipo 2 | Organigrama | 78 |
| Documentos tipo 2 | Plan | 175 |
| Documentos tipo 2 | Postal | 8 |
| Documentos tipo 2 | Prensa | 2089 |
| Documentos tipo 2 | Prontuario | 101 |
| Documentos tipo 2 | Revista | 836 |
| Documentos tipo 2 | Solicitud de constancia habilitación cargos públicos | 2102 |
| Documentos tipo 2 | Tarjeta | 18 |
| Documentos tipo 2 | Telegrama | 165 |
| Documentos tipo 2 | Texto | 1833 |
| Documentos tipo 2 | Traducción de documento | 130 |
| Documentos tipo 2 | Transcripción | 689 |

Continúa en la siguiente página

| Categoría | Subcategoría | Cantidad |
|---|---|---|
| Documentos tipo 2 | Transcripción de audición | 1080 |
| Documentos tipo 2 | Volante | 253 |
| Informe | - | 2660 |
| Informe | Informe de inteligencia | 1357 |
| Informe | Informe especial | 193 |
| Informe | Informe mensual o memoria | 75 |
| Memorándum | - | 2392 |
| Memorándum | Memo de anotaciones | 710 |
| Memorándum | Memo de antecedentes | 472 |
| Memorándum | Memo de información | 703 |
| Memorándum | Memo especial de información | 17 |
| Memorándum | Memo operacional | 42 |
| Nómina de personas | - | 1188 |
| Nómina de personas | Índice | 350 |
| Nómina de personas | Listado | 3312 |
| Nómina de personas | Relación | 434 |
| Parte de información | - | 1709 |
| Parte de información | Parte | 328 |
| Parte de información | Parte de novedades diario | 3262 |
| Parte de información | Parte especial de información | 4673 |
| Parte de información | Parte periódico de información | 1150 |
| Pedido | - | 41 |
| Pedido | Pedido de información | 179 |
| Pedido | Pedido de informes | 104 |
| Resumen | - | 179 |
| Resumen | Resumen de antecedentes | 916 |
| Resumen | Resumen de información | 640 |
| Solicitud | - | 277 |
| Solicitud | Solicitud de captura | 146 |
| Solicitud | Solicitud de información | 356 |
| Sumario | - | 673 |
| Sumario | Sumario de operaciones | 55 |
| Acta | - | 507 |
| Acta | Acta de declaración | 132 |
| Acta | Acta de interrogatorio | 1326 |
| Acta | Acta de la Justicia Militar | 7 |
| Carta | - | 3083 |
| Carta | Carta anónima | 65 |
| Carta | Total | 3148 |
| Circular | - | 238 |
| Diligenciado | - | 1261 |
| Hoja de trámite | - | 656 |
| Nota | - | 1833 |
| Oficio | - | 5169 |
| Requisitoria | - | 238 |
| Resolución | - | 455 |