UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ACATLÁN

USO DE APRENDIZAJE DE MÁQUINA PARA EL ANÁLISIS DE LOS FICHEROS DE LA DIRECCIÓN FEDERAL DE SEGURIDAD (DFS).

T E S I S

QUE PARA OPTAR POR EL GRADO DE:

LICENCIADO EN MATEMÁTICAS APLICADAS Y COMPUTACIÓN

PRESENTA:

JUAN CARLOS GONZÁLEZ AGUILAR

TUTOR PRINCIPAL:
DRA. MARIANA ESTHER MARTÍNEZ SÁNCHEZ
INSTITUTO NACIONAL DE ENFERMEDADES RESPIRATORIAS
"ISMAEL COSÍO VILLEGAS"

ESTADO DE MÉXICO, MÉXICO, 2022

El aspecto más triste de la vida en este preciso momento
es que la ciencia reúne el conocimiento más
rápido de lo que la sociedad reúne la sabiduría.

— Issac Asimov

AGRADECIMIENTOS

Primero, debo agradecer a mis padres: Emilia Aguilar y Nestor Enrique González. Juntos moldearon un hogar lleno de amor y valores, me dieron las herramientas que hoy me permiten resolver los retos que la vida me enfrenta. Gracias por todo lo que nos dieron. Me siento orgullo de poder decir que fui criado por personas tan maravillosas como ustedes. Este trabajo se los dedico a ustedes.

A mis hermanos Luis Alberto González Aguilar, Victor Enrique González Aguilar y Josué Brandon González Aguilar. Quienes no dudaron en levantarme y aconsejarme cuando fue necesario. Estoy sumamente orgulloso de ser su hermano.

A mi tutora de tesis, Doctora Mariana Esther Martínez Sánchez porque fue gracias a tu asesoramiento y experiencia que pude sortear los momentos más complicados durante la realización de este trabajo. Gracias por tu tiempo y confianza. Me hiciste recobrar la motivación cuando parecía lejana la culminación de este trabajo.

Agradezco a la Comisión Nacional de Búsqueda (CNB) y al proyecto Archivos de la Represión (AR) por gestar tan loables proyectos cuyo conocimiento fue vital para el desarrollo de esta tesis. Además hago un especial agradecimiento al Centro de Investigación en Matemáticas (CIMAT) por su apoyo brindado mediante sus docentes, estudiantes y recursos en la realización del presente trabajo.

A la Universidad Nacional Autónoma de México (UNAM) por brindarme la posibilidad de cursar estudios de preparatoria y universidad cuya calidad no me hubiera sido posible obtener de otra forma. A esta institución le debo todo lo que soy hoy profesionalmente y siempre le estaré en deuda.

A la Facultad de Estudios Superiores Acatlán, así como a sus profesores y administrativos, quienes me apoyaron todos estos años y a quienes debemos que la Facultad siga vigente.

Al Doctor Víctor Mireles y al Doctor Víctor Muñiz, porque me apoyaron con ideas, dedicaron parte de su tiempo en conocer lo que

estábamos haciendo y por sus valiosos comentarios.

Además, debo agradecer a todas estas personas que me apoyaron emocionalmente y a quienes han guardado un sentimiento de estima hacia mí. Valoro mucho su presencia y aunque no los mencione son parte importante de este logro.

Finalmente, a ti, por tomarte el tiempo de leer mi Tesis. Muchas gracias.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CÓDIGOS

ACRÓNIMOS

DFS Dirección Federal de Seguridad

DIPS Departamento de Investigación Política y Social

DGIPS Dirección General de Investigaciones Políticas y Sociales

DGISN Dirección General de Investigación y Seguridad Nacional

CISEN Centro de Investigación y Seguridad Nacional

CNI Centro Nacional de Inteligencia

AGN Archivo General de la Nación

AR Archivos de la represión

CNDH Comisión Nacional de Derechos Humanos

FEMOSPP Fiscalía Especial para Movimientos Sociales y Políticos del Pasado

COMVERDAD Comisión de la Verdad del Estado de Guerrero

CNB Comisión Nacional de Búsqueda

OCR Optical character recognition

INTRODUCCIÓN

El presente trabajo, tiene como objetivo desarrollar una metodología basada en técnicas de análisis de imágenes y texto por computadora. Dicha metodología extrae la clave del expediente de las fotografías de los ficheros de la DFS que se encuentran resguardados públicamente en Archivos de la represión (AR). La importancia de estudiar este tema en particular radica en que permite explorar fuentes de información físicas con alta relevancia histórica y social para auxiliar en la búsqueda de personas desaparecidas por la DFS durante la Guerra Sucia en México (ver sección 1.1). Como consecuencia, vuelve eficiente la transcripción de los documentos y con ello acerca a sus usuarios en la búsqueda de la verdad.

Un elemento importante para entender la realización de este trabajo es el papel de la ciencia como herramienta para la búsqueda de la verdad. En la página oficial de la UNESCO podemos leer lo siguiente sobre la ciencia:

> "La ciencia ofrece soluciones para los desafíos de la vida cotidiana y nos ayuda a responder a los grandes misterios de la humanidad. En otras palabras, es una de las vías más importantes de acceso al conocimiento. Tiene un papel fundamental del cual se beneficia el conjunto de la sociedad: genera nuevos conocimientos, mejora la educación y aumenta nuestra calidad de vida.".
>
> (*UNESCO*, 2015)

Así como los avances en el ADN y las huellas dactilares se convirtieron en herramientas de apoyo contra la impunidad, vale la pena discutir el papel de los nuevos avances en el campo de la inteligencia artificial y su potencial progreso en el esclarecimiento de crímenes no solo actuales, sino del pasado. Por lo que es sustancial y natural discutir la necesidad de la realización de este tipo de trabajos.

En cuanto a los esfuerzos para aplicar estas técnicas a los ficheros de la DFS, el presente trabajo pretende ser un parteaguas que aborde la utilización de técnicas de aprendizaje de máquina. Sin embargo, cabe destacar el esfuerzo de historiadores, familiares de víctimas y activistas por esclarecer la verdad de los desaparecidos de la guerra sucia. Trabajos que de no existir, no hubieran hecho posible la correcta investigación de los ficheros de la DFS.

El objetivo principal de la presente tesis, es el de sugerir una metodología que permita la explotación de los ficheros de la DFS. Se trabajará sobre la hipótesis de que es posible la extracción de la clave de expediente de forma automatizada para las fotografías de los ficheros de la DFS usando una metodología basada en técnicas de Aprendizaje de máquina. Adicionalmente, se supondrá que una serie de pre-procesamientos específicos a este conjunto de datos, mejorarán la calidad de la extracción de la clave de expediente.

De igual forma, se describirán los principales retos encontrados y se discutirá la importancia de cada una de las soluciones y sus siguientes pasos.

En el capítulo 1 se abordarán las cuestiones teóricas que sustentan el planteamiento del trabajo. Se hace un resumen del contexto histórico que acompaña la realización de los ficheros, desde la creación de las organizaciones de inteligencia en México y su relación con la Guerra Sucia (principal tema abordado en la investigación de este tipo de documentos). Finalmente, se hará un breve repaso sobre redes neuronales, en particular la teoría relacionada a las redes convolucionales, cuya estructura fundamenta el desarrollo de las últimas arquitecturas. Mencionamos la arquitectura Yolov5 implementada durante el desarrollo de este trabajo.

En el capítulo 2 se abordan los métodos desarrollados para la obtención, preparación y explotación de los ficheros. Entre los métodos que se explican, se encuentran métodos desarrollados, entrenados y aplicados; relacionados con descarga de archivos, etiquetado de imágenes, limpieza de imágenes, reconocimiento de objetos, reconocimiento óptico de imágenes y estimación de la calidad del texto. Asimismo, se identificarán y generarán arquitecturas basadas en modelos de aprendizaje de máquina para resolver subproblemas encontrados dentro de la obtención del texto en los ficheros de la DFS. El capítulo finaliza explicando la metodología desarrollada para la explotación de los ficheros de la DFS (ver sección 2.7).

En el capítulo 3 serán discutidos los resultados obtenidos de los métodos desarrollados en el capítulo 2. Se medirá la eficacia de su utilización y con la finalidad de probar la hipótesis se hará un contraste de los resultados del flujo propuesto por la librería Tesseract-OCR (ver sección 2.5) y la metodología propuesta. Los resultados contenidos en este capítulo justifican la división de tareas propuestas y el desarrollo de técnicas personalizadas para los ficheros de la DFS.

1

MARCO TEÓRICO

1.1 LA GUERRA SUCIA

El periodo conocido como "Guerra Sucia", es un momento en la historia de México que abarca entre 1943-1981. Se caracteriza por el aplastamiento, hostigamiento y desaparición de personas relacionadas con el activismo político y con ideologías comunistas/socialistas. Si bien, la revolución mexicana luchó y consiguió mayores derechos para los agricultores mexicanos, en el transcurso de unos años se repiten los mismos atropellos con la complicidad del gobierno estatal y federal. En el marco de estas arbitrariedades, líderes agrarios regresan a las armas, hartos de la imposición de líderes corruptos y de la maquinaria de sabotaje enfocada en beneficiar, con los apoyos gubernamentales, solo a un grupo selecto de empresarios y funcionarios. Los agricultores oprimidos se quedan con tierras, pero sin dinero ni mecanismos para sembrarlas. De esta manera, se ponen en bandeja de plata a los especuladores y oportunistas (Castellanos, 2013, p.43-51).

Figura 1.1: Imagen tomada del documental Guerrero: Memoria y verdad (CDHDF, 2016).

Al término de la Segunda Guerra Mundial y en el contexto de la Guerra Fría, en México se desencadena toda una maquinaria de comunicación y política para desprestigiar las ideologías y a personajes afines al socialismo y al comunismo. La agenda es impulsada para

prevenir el costo político y social que implica el llevar un sistema político y económico diferente. (Castellanos, 2013, p.33-43).

Con el pretexto del combate a los enemigos del estado, las personas en el poder durante la guerra sucia usaron maquinaria militar y de inteligencia para desaparecer a quienes consideraban enemigos del estado. Es importante señalar que a lo largo de la historia, los gobiernos siempre han contado con instituciones encargadas de recabar información oportuna y veraz con la mayor discreción posible. Dicha información es analizada, evaluada y posteriormente entregada con el nombre de "inteligencia" a los gobernantes y es empleada para tomar decisiones. La organización de este tipo de instituciones suele ser parecida. En primer lugar, el área de investigación obtiene datos de fuentes abiertas o encubiertas (para estas últimas se valen de informantes voluntarios o pagados). Una división fundamental es la encargada de analizar y procesar la información. Una función sumamente valiosa en la recopilación de estos datos es evitar que enemigos de la seguridad nacional accedan a información vital que, dependiendo del régimen, pueden recibir el encargo de realizar operaciones en contra de los enemigos del Estado (Quezada, 2014, p.364).

Las instituciones de inteligencia deberían de ser altamente capacitadas, profesionalizadas y que trabajen de acuerdo a los intereses del Estado, con plena independencia a las disputas por el poder de grupos o partidos. Situación que lamentablemente no se cumple en instituciones como la DFS. Como veremos en los relatos del académico Sergio Aguayo Quezada, autor de *La Charola* (Quezada, 2014), su labor fue la de responder en beneficio de las personas cercanas a la vida política mexicana. Información confirmada por las fichas y por varias de las tarjetas de la DFS en poder de políticos y empresarios mexicanos pertenecientes al status-quo del México de la época:

> "Tenían credencial de la DFS el maquinista del tren presidencial, el magnate de la comunicación Emilio Azcárraga, mandos de diversas corporaciones y el periodista Manuel Buendía Tellezgirón".
>
> (Quezada, 2014, p. 1,140)

En 1942, durante la intervención de México en la Segunda Guerra Mundial, se crea el Departamento de Investigación Política y Social (DIPS). Este se encargaba principalmente de asuntos de orden político internos y después de la intervención de México en la guerra, se le agregaron facultades para poder realizar labores de inteligencia en personas extranjeras. En 1967 se transformó en Dirección General de Investigaciones Políticas y Sociales (DGIPS), cuyo objetivo era efectuar estudios de orden político y social del país, además de llevar a cabo

encuestas de opinión pública sobre asuntos de relevancia nacional (CNI, 2019).

En cuanto a la DFS, esta fue una agencia de inteligencia mexicana cuya creación se sitúa entre diciembre de 1946 y enero de 1947. En sus inicios, la DFS dependía directamente de la presidencia de la república, pero fue durante el mandato de Adolfo Ruiz Cortines (1952 a 1958) que en miras de desaparecer la corporación, finalmente se decidió enviarla bajo la supervisión de la Secretaria de Gobernación, lugar donde permaneció hasta el año 1985, fecha de su desaparición (Quezada, 2014, p.827). Su principal objetivo era la investigación de delitos del fuero federal; es decir, cualquiera que atentara con la seguridad de la nación y en su caso tenía la responsabilidad de informarlo al Ministerio Público Federal (CNI, 2019).

Con el objetivo de evitar la duplicidad de actividades y replantear el papel de los servicios de inteligencia en México, se creó la Dirección General de Investigación y Seguridad Nacional (DGISN) en el año 1985, institución que fusionó las labores de la DGIPS Y DFS. De 1989 a 2018 pasó a ser nombrado como el Centro de Investigación y Seguridad Nacional (CISEN). En 2018 la institución fue renombrada como el Centro Nacional de Inteligencia (CNI), nombre que ostenta actualmente (2022) (CNI, 2019).

El presente trabajo se enfoca principalmente en la DFS. Sin embargo, es importante puntualizar que a pesar del parecido en funciones y que eran mayormente contemporáneas, la DGIPS y la DFS fueron instituciones separadas y distintas. Como lo han anotado investigadores del tema, es fundamental que los documentos y en general el estudio de ambas se separe y no se traten como lo mismo (Proceso, 2020).

Entre las actividades de interés para víctimas e instituciones que estudian a la DFS, se encuentran las desapariciones forzadas. Con base en la jurisprudencia internacional, Vicente Ovalle la define como: "La privación de la libertad de una persona o grupo de personas por parte de un servidor público o con la aquiescencia del Estado, acompañada de la falta o negativa de información sobre el paradero de la persona, sustrayéndola de los efectos de la ley". La primera de ellas, la desaparición forzada, es de carácter permanente y su principal objetivo es la eliminación de las víctimas. Sin embargo, existe otro fenómeno poco estudiado, pero presente a lo largo del período histórico, que es el de la desaparición forzada transitoria. Esta es una categoría que define a las personas que sobrevivieron a la desaparición (Vicente Ovalle, 2019, p.20).

Consultando las cifras recolectadas en el libro "Tiempo suspendido: una historia de la desaparición forzada en México, 1940-1980," se sabe que según un informe de la Comisión Nacional de Derechos Humanos (CNDH) en 2021 se reportaban 532 casos de desaparición forzada. De ellos, 181 en zonas urbanas y 351 en zonas rurales. Esta misma acredita 275 casos, en 97 se tienen indicios y 160 son no acreditados. La Fiscalía Especial para Movimientos Sociales y Políticos del Pasado (FEMOSPP) reporta 787 casos entre 1960 y 1980, 436 acreditados, 207 con presunción fundada y 145 carecen de información. La Comisión de la Verdad del Estado de Guerrero (COMVERDAD) cuyo origen se explica en la sección 1.1.3, a diferencia de los anteriores, nos informa sobre las desapariciones forzadas transitorias con 205 personas detenidas y/o desaparecidas y que sobrevivieron (Vicente Ovalle, 2019).

1.1.1 *Sistema de archivos de la DFS*

El sistema de archivos de la DFS se desarrolló antes de la creación de la institución (1944-1945). Este archivo incluye fichas biográficas de personas, expedientes, estudios especiales, recortes de periódicos, etc. El acervo comenzó a formarse en 1924 cuando llegó un archivista de la Secretaría de Industria y Comercio, Luis Vargas González, quien elaboró la "nomenclatura que llevan los expedientes con el tarjetero respectivo". En 1934, ese archivo tenía "millares de expedientes", los cuales eran muy vulnerables a los vaivenes políticos (Quezada, 2014, p.602). Estos expedientes eran navegables usando el sistema de ficheros que se estudia en este trabajo.

El archivo original comprende entre los años 1947 y 1991, las fichas siguieron empleándose posterior a la desintegración de la DFS (1985), a partir de 1990-1991 se comenzaron a usar archivos electrónicos. En este periodo, se acumularon de 60 a 80 millones de fichas, con un total de 3 a 4 millones de personas o instituciones registradas. Además de las fichas, se cuentan con 26 mil vídeos y más de 250 mil fotografías (Quezada, 2014, p.202).

En cuanto a la historia resguardada en el archivo, se encuentran acontecimientos importantes del México contemporáneo. Uno de ellos es el asalto al Cuartel de Madera el 23 de septiembre de 1968, que posteriormente influenciaría la creación de La Liga Comunista "23 de septiembre" en honor a dicho evento el 15 de marzo de 1973. Se encuentra también evidencia de la participación de la DFS en los sucesos ocurridos el 2 de octubre de 1968 en Tlatelolco (Quezada, 2014, p.1485), además de informes del involucramiento de altos mandos del Estado mexicano en el aplastamiento del movimiento estudiantil el 10 de junio de 1971, el llamado "Halconazo" (Quezada, 2014, p.2119). To-

dos estos sucesos se caracterizan por la presencia infiltrada de miembros activos de la DFS en organizaciones civiles opuestas al régimen y de un uso desmesurado de la fuerza del estado en el aplastamiento de los movimientos insurgentes (Quezada, 2014, p.1884-2119).

1.1.2 *Cómo llegaron los ficheros al AGN*

El Archivo General de la Nación (AGN) es la institución responsable de resguardar y difundir el patrimonio documental de México. Dicho patrimonio da cuenta del desarrollo histórico del país, aportando evidencias de los sucesos trascendentes que han marcado a la sociedad Mexicana (AGN, Fecha de consulta: 2022).

Los ficheros terminaron en el AGN para su resguardo en enero de 1982. Fecha en la que el AGN se encontraba estrenando instalaciones en el palacio de Lecumberri. Bajo resguardo quedaron poco más de 2,900 cajas. El acervo se colocó en la Galería 7. A diferencia de documentos generados por instituciones como la Secretaria de gobernación o puestos a consulta pública desde fecha temprana, los ficheros de la AGN tuvieron un tratamiento especial debido a que contenían información considerada confidencial y de seguridad nacional. Dicha información era producida por la DGIPS, DFS y la Secretaría Particular (Salazar Anaya, 2006).

Esta gama de documentos, que testifica la actividad de las dependencias encargadas de llevar a cabo la inteligencia política en el país entre las décadas de los años de 1920 a 1980, se consideraba de acceso restringido para las normas vigentes, en virtud de que en el Reglamento del AGN, las secretarías de Estado podían señalar cuáles eran los documentos que debían ser tenidos como reservados y a la vez, indicar la fecha en que finalizaba esta restricción. Fue hasta agosto de 1998, según el Reglamento Interior de la Secretaría de Gobernación, cuando se estipuló que la consulta podría realizarse "hasta después de 30 años contados a partir de la fecha de generación del documento, en tanto no afecte la seguridad nacional o la vida privada de las personas". De tal forma que, a partir de ese momento, los papeles de la DGIPS paulatinamente fueron puestos a consulta pública en la galería.

1.1.3 *Esfuerzos por esclarecer los acontecimientos sucedidos durante la Guerra Sucia*

En el año 2000, como parte del reconocimiento del Estado en su participación en los hechos del 2 de octubre de 1968, el presidente Vicente Fox dejó un importante precedente durante una visita a París, donde

se refirió a este suceso de la siguiente manera: "Su sacrificio no fue en vano... sembraron una voluntad de cambio que hoy está dando los primeros frutos tangibles" (La Jornada, 3 de Octubre del 2000). La importancia (como se discute en Montaño,2017) era poder esclarecer el pasado para asegurar el futuro de la democracia. Por decreto oficial, Fox crea la Fiscalía Especial para Movimientos Sociales y Políticos del Pasado (FEMOSPP) el 27 de Noviembre de 2001. El objetivo era llevar justicia Jurídica y esclarecimiento histórico de la represión del régimen autoritario en contra de integrantes de movimientos opositores (Montaño, 2017).

En noviembre de 2006, al término del mandato del presidente Vicente Fox, se dieron por finalizadas las labores de la FEMOSPP junto con un extenso reporte (FEMOSPP, 2006). En él, se concluía que el Estado Mexicano había incurrido en graves violaciones a los derechos humanos, entre las que estaban: homicidios, desapariciones forzadas y tortura. Aunque se abrieron juicios para los eventos ocurridos el 2 de Octubre de 1968 y el 10 de Junio de 1971, según este informe, la investigación no concluyó en ninguna sentencia condenatoria para los culpables. No obstante, lo anterior es desmentido en El Canto del cisne de la FEMOSPP(Yankelevich, 2018). En este documento se menciona a Esteban Guzmán Salgado, exagente de la DFS, que en septiembre de 2009 fue condenado por la desaparición forzada de Miguel Ángel Hernández Valerio, quién desapareció en 1977 en Mazatlán y hasta la fecha no se sabe de su paradero. El juicio de Guzmán Salgado comenzó en 2006.

Otro esfuerzo notable en beneficio del esclarecimiento de la suerte de los desaparecidos fue el de la COMVERDAD. Comisión formada oficialmente el 23 de diciembre de 2011, según el decreto publicado en el Diario Oficial del Gobierno del Estado. Unos meses antes, el Pleno de la Cámara de Diputados del Congreso del Estado había aprobado por unanimidad la “Ley por la que se crea la Comisión de la Verdad para la Investigación de las Violaciones a los Derechos Humanos Durante la Guerra Sucia de los años Sesenta y Setenta del Estado de Guerrero” en la sesión celebrada el 19 de diciembre de ese año. Las actividades de la COMVERDAD finalizaron en 2014 con el “Informe final de actividades de la Comisión de la Verdad para la investigación de las violaciones a los Derechos Humanos durante la Guerra Sucia de los años sesenta y setenta del estado de Guerrero.” Estos archivos fueron fotografiados y existe una versión pública que se puede encontrar en la página https://archivosdelarepresion.org/ (Artículo 19 , COMVERDAD, 2020).

La COMVERDAD encontró evidencia sobre:

- Ejecuciones arbitrarias.
- Desaparición forzada.

- Desaparición forzada transitoria.
- Desplazamiento forzado.
- Violación generalizada o sistemática de los derechos humanos.
- Crímenes de lesa humanidad en la guerra sucia de Guerrero.

Actualmente (2022) existen varias instituciones trabajando en el tema. La Comisión Nacional de Búsqueda (CNB) tiene una Unidad para personas desaparecidas víctimas del terrorismo de Estado durante la "Guerra sucia", con quien se colaboró para la realización de este trabajo. La CNDH ha puesto en operación una Oficina Especial para Investigar la Represión y Desapariciones Forzadas por Violencia Política del Estado Durante el Pasado Reciente. Finalmente, en 2021, el Presidente Andrés Manuel López Obrador, formó una Comisión por el Acceso a la Verdad y el Esclarecimiento Histórico y Justicia a las Violaciones graves a los Derechos Humanos, cometidos entre los años 1965-1990.

1.2 INTRODUCCIÓN A LAS REDES NEURONALES

Los seres humanos almacenamos el aprendizaje y la memoria mediante estructuras neuronales. Algunas de estas estructuras las obtenemos al nacer y otras se van formando con la edad y las experiencias. El objetivo de generar redes neuronales en un computador es tratar de aproximar la capacidad que tiene su símil biológico para poder resolver tareas que necesitan de un proceso de aprendizaje y explotación más complejo. Algunas de las aplicaciones más conocidas son:

- En la fabricación de automóviles con conducción automática.
- En la industria de la salud, con herramientas de diagnóstico para enfermedades como cáncer de mama y de piel.
- En la conversión de voz a texto y viceversa.
- En reconocimiento de imágenes: librerías como opencv cuentan con clasificadores muy potentes.
- En traducción automática de textos.

Respecto a la capacidad de las redes neuronales para realizar poderosas aproximaciones a soluciones de aprendizaje automático, existen trabajos como el de George Cybenko, el cual demuestra que una red neuronal multicapa es capaz de representar cualquier función continua con soporte en el conjunto hipercubo $[0,1]^n$. Este resultado es usado para explicar la red neuronal como estructura de aprendizaje eficaz para la resolución de problemas de aprendizaje de máquina (Cybenko, 1989).

1.2.1 *Perceptrón*

El perceptrón es considerado el modelo más simple de una red neuronal, al ser eficaz para resolver problemas que son linealmente separables, es decir, problemas a cuya solución se pueda llegar usando un corte lineal en las variables. El diagrama de su funcionamiento puede verse en la figura 1.2.

El perceptrón asume que y es combinación lineal de las variables $x \in X$ están, es decir:

$$y = \sigma(w_1 x_1 + w_2 x_2 + ... + w_n x_n + b) \tag{1.1}$$

Donde σ es la función de activación y b el sesgo del modelo.

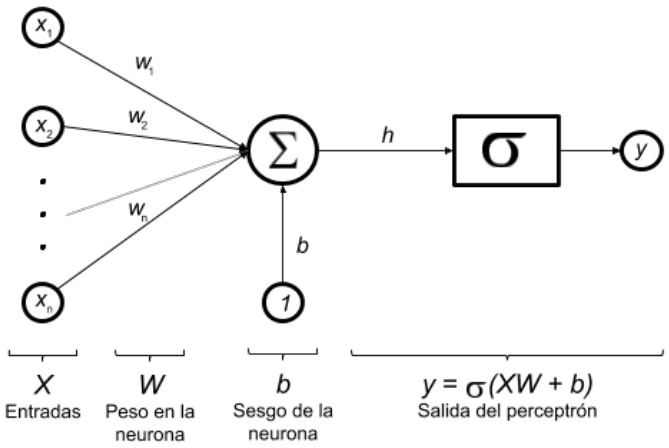


Figura 1.2: Modelo perceptrón.

En inferencia estadística, el sesgo o *bias* nos indica qué tan bien nuestro modelo se ajusta al fenómeno a predecir, limitado por la cantidad de observaciones y las variables recolectadas durante la elección del conjunto de entrenamiento. Si el valor del sesgo es cercano a 0, el error del modelo tiende a 0, este valor es calculado en el perceptrón para corregir el error del modelo.(Mehta et al., 2019).

Entre las desventajas se encuentra su incapacidad para resolver problemas no lineales. Pocos de los problemas que se resuelven actualmente con redes neuronales se pueden solucionar con el perceptrón (Goodfellow et al. 2016, p.240). En este sentido, una solución más robusta que resuelve la limitante del perceptrón en separabilidad es una red completamente conectada.

1.2.2 *Función de activación*

La función de activación tiene como objetivo mapear el resultado de aplicar los pesos y el *bias* a las variables de entrada al espacio de los valores posibles de y. El propósito de mostrar las siguientes funciones de activación es el de explicar la forma en la que se le dota a la red neuronal la capacidad de ajustarse a y, parte que se considera relevante durante la definición del marco teórico. Algunas de las funciones de activación más usadas son:

1.2.2.1 *Paso Binario*

Esta función mapea su entrada al conjunto $\{0, 1\}$:

$$\sigma(x) = \begin{cases} 0 & for \quad x \leq 0 \\ 1 & for \quad x > 0 \end{cases} \tag{1.2}$$

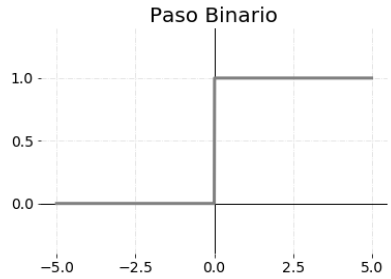


Figura 1.3: Función de activación: Paso Binario.

1.2.2.2 *Rectificador Lineal*

El Rectificador Lineal o ReLU es la función de activación más usada y recomendada en la mayoría de los casos, ya que permite tener una transformación no-lineal pero conservando características que le permiten a la red aplicar elementos como la propagación hacia atrás, necesarios para el proceso de entrenamiento (Goodfellow et al., 2016, p.190) . Esta función mapea la entrada al conjunto de los enteros positivos:

$$\sigma(x) = \begin{cases} 0 & for \quad x \leq 0 \\ x & for \quad x > 0 \end{cases} \tag{1.3}$$

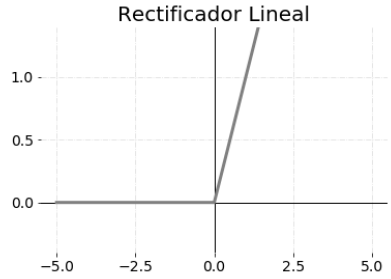


Figura 1.4: Función de activación: Rectificador Lineal(ReLU).

1.2.2.3 *Arcotangente*

Esta función aplica la función arcotangente a la entrada:

$$\sigma(x) = tan^{-1}(x) \tag{1.4}$$

1.2.2.4 *Sigmoide*

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1.5}$$

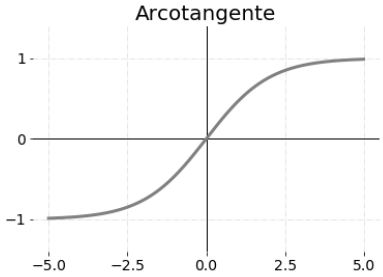


Figura 1.5: Función de activación: Arcotangente.

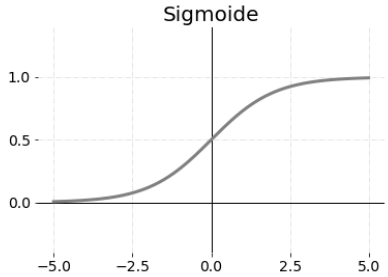


Figura 1.6: Función de activación: Sigmoide.

1.2.2.5 *Sigmoidal Bipolar*

$$\sigma(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \tag{1.6}$$

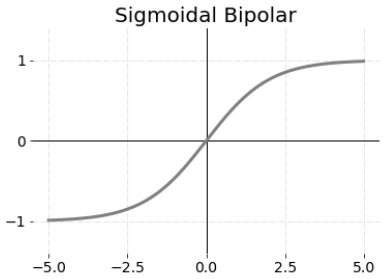


Figura 1.7: Función de activación: Sigmoidal Bipolar.

1.2.3 *Redes neuronales completamente conectadas de una capa*

Una red completamente conectada es una red donde tanto las neuronas de entrada como las de salida están fuertemente conectadas. De este modo, la salida es un vector y con n elementos.

Usando como punto de partida la ecuación 1.1, sea $x \in \mathbb{R}^m$ el conjunto de las entradas y sea $y_i \in \mathbb{R}^n$ los elementos del vector y de salidas:

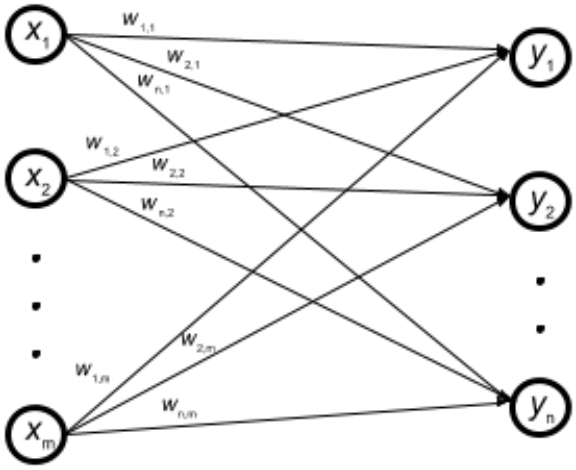


Figura 1.8: Modelo de una red completamente conectada.

$$y = \begin{pmatrix} \sigma(w_{1,1}x_1 + \cdots + w_{1,m}x_m) \\ \vdots \\ \sigma(w_{n,1}x_1 + \cdots + w_{n,m}x_m) \end{pmatrix} \tag{1.7}$$

1.2.4 *Redes neuronales completamente conectadas multicapa*

Una generalización del modelo de red neuronal completamente conectada de una capa es el de múltiples capas. De esta manera, es posible para la red aprender patrones más complejos y tener áreas *especializadas* que ayuden a generalizar de una mejor manera el aprendizaje de la red.

Esto es explicado en el libro *DeepLearning Book* (Goodfellow et al., 2016, p.20), en el que el autor usa de punto de partida a las capas como un estado en memoria, donde cada estado es el resultado de aplicar instrucciones en paralelo (perceptrones). De este modo, redes neuronales con muchas capas pueden ejecutar una mayor cantidad de instrucciones en secuencia (Goodfellow et al., 2016, p.20).

Expresar mediante una fórmula las redes neuronales multicapa se vuelve más complicado. De ahora en adelante se empleará θ para hacer referencia a los pesos de la red y X para el conjunto de covariables que explican a y. De esta forma expresaremos la red como una función en términos de X y θ:

$$\hat{y} = f(X, \theta) \tag{1.8}$$

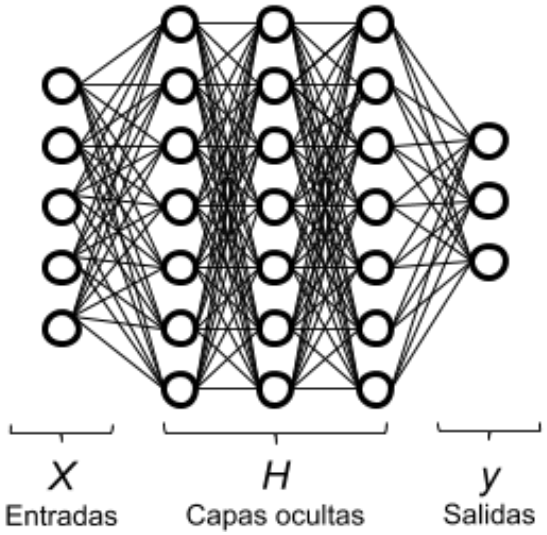


Figura 1.9: Modelo de una red completamente conectada multicapa.

1.3 REDES PROFUNDAS CONVOLUCIONALES

Las redes neuronales convolucionales dotan a las redes neuronales profundas la capacidad de aprender patrones espaciales como los que hay en imágenes, vídeos y textos. Gran parte del progreso en campos como el reconocimiento de imágenes se debe a este tipo de redes y al avance en el hardware necesario para procesar el entrenamiento de estas redes.

Las redes neuronales convolucionales se basan principalmente en dos operaciones:

- Agrupación
- Convolución

La *Convolución* aplica un filtro a los elementos de la entrada. Este filtro está definido por el *kernel*. El filtro aplica el producto a escalar entre el *kernel* y la entrada (ver figura 1.10). Otro elemento a considerar en la etapa de convolución es el *desplazamiento*. El desplazamiento define cada cuántos elementos se moverá el kernel. Además, el valor de éste y el tamaño del kernel definen la forma de la salida.

En la figura 1.10 podemos observar una convolución operando a una imagen de entrada de 8x8, un **kernel** de tamaño 3x3 con una **ventana de desplazamiento** de 1, dado que en cada iteración nuestro kernel se desplaza 1 casilla, tenemos que dentro de 8 posiciones, nuestra ventana puede desplazarse 5 veces a lo ancho (8 por el ancho original de la imagen, menos 3 del ancho del kernel), a su vez sucede lo mismo de forma vertical, dando como resultado un arreglo de 5x5

elementos, como se muestra a continuación en la figura 1.10.

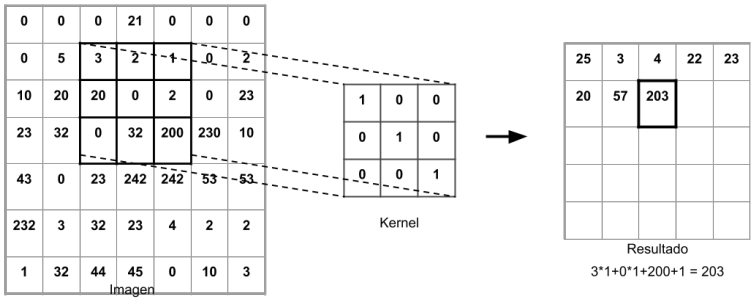


Figura 1.10: Diagrama de una convolución con ventana de desplazamiento de 1. Cada uno de los elementos del arreglo resultante es el resultado de aplicar el producto punto del kernel en las distintas regiones de la imagen.

Una operación relevante para reducir el tamaño de la entrada y extraer las características más importantes es aplicar una **agrupación**. Igual que la convolución, la agrupación también cuenta con el parámetro de ventana de desplazamiento y dependiendo de ésta será el tamaño de la salida. En la figura 1.11, la ventana de desplazamiento es de 2x2 y la función que se aplicó es una agrupación máxima. Esta transformación conserva el valor más grande de cada iteración en la ventana de desplazamiento.

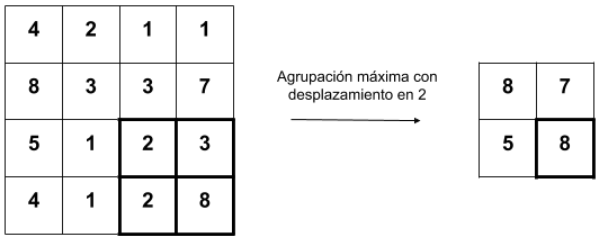


Figura 1.11: Diagrama de una agrupación máxima con una ventana de desplazamiento de 2, cada elemento corresponde al valor máximo de la ventana por transición.

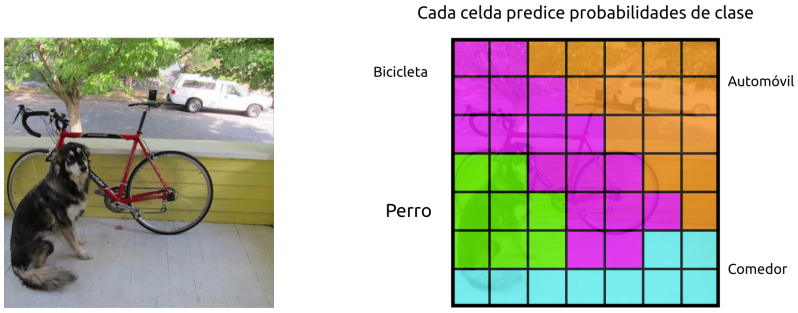
La convolución es una transformación en la que se aplica un cambio a la entrada mediante un filtro definido a través de una matriz K llamada *kernel*. La agrupación es una forma de muestrear sobre los datos y conservar solo los más relevantes. Esta estructura es la base de grandes algoritmos del estado del arte para tareas de detección automática de objetos y reconocimiento óptico de caracteres.

1.4 LA ARQUITECTURA YOLOV5

Uno de los más grandes problemas en el área de visión por computadoras es la detección automática de objetos. Los modelos basados en Redes Neuronales Convolucionales suelen tener inconvenientes cuando se necesita rapidez y precisión. Esto se debe a una gran y necesaria cantidad de información para generalizar y a amplios e indispensables parámetros para su entrenamiento. Además, por si solos suelen tener inconvenientes para solucionar más de un problema a la vez (como lo puede ser detectar entre animales y objetos dentro de la misma arquitectura). Dentro del grupo de las arquitecturas diseñadas para solucionar este problema se encuentran los modelos descendientes del YOLOV.

1.4.1 *Historia*

En el año 2016, Joseph Readmon desarrolló y publicó un trabajo llamado: "You Only Look Once: Unified, Real-Time Object Detection". En este trabajo describe la primera de 3 arquitecturas con el prefijo Yolov. Readmon propone generar una arquitectura que le permita a la red identificar entre varias clases y determinar áreas completas cuyo conjunto identifiquen objetos. Además, propone a Yolov como una arquitectura rápida y certera comparada con los modelos de su tiempo.



(a) Imagen ejemplo (b) Imagen segmentada

Figura 1.12: Gráfico que representa cómo funciona el modelo Yolov

Readmon estuvo involucrado en las versiones Yolov2 (2017), Yolov3 (2018) y en 2020 anunció su retiro de la investigación en el campo de visión por computadora. En abril 23 del 2020, Alexey Bochkovskiy desarrolla la implementación Yolov4.

Finalmente, en junio 9 del 2020, Glenn Jocher publicó el modelo Yolov5 que, entre otras cosas, destaca por su rapidez y por ser 90% menos pesado que su antecesor Yolov4.

Hasta el momento Yolov5 no cuenta con algún documento publicado, pero sí con un repositorio público y con el código necesario para

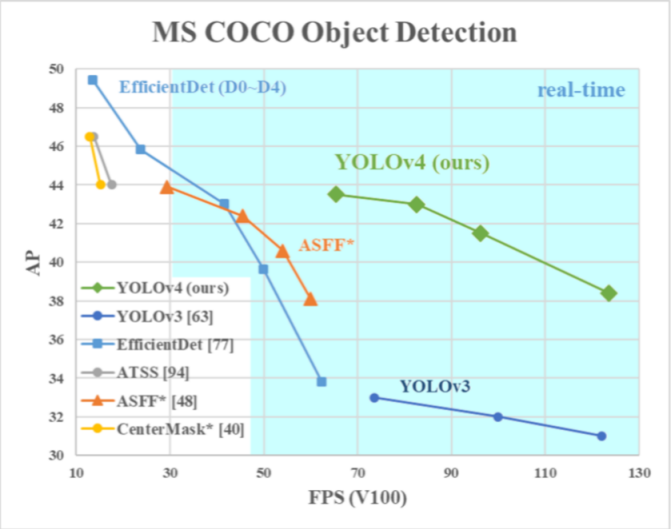


Figura 1.13: Comparación de yolov4 vs modelos similares.

realizar el entrenamiento de nuevas tareas de detección por computadora usando las técnicas de data-augmentation y la estructura de la red implementada por el equipo de ultralytics.

1.4.2 *Implementación*

Yolov5 implementa cuatro distintas arquitecturas. Sus implementaciones se basan en limitar el número de parámetros entrenables, sacrificando poder predictivo, pero obteniendo rapidez de evaluación y entrenamiento (ver 1.1).

Model	APval	APtest	AP50	SpeedGPU	FPSGPU	params	FLOPS
YOLOv5s	36.1	36.1	55.3	2.1ms	476	7.5M	13.2B
YOLOv5m	43.5	43.5	62.5	3.0ms	333	21.8M	39.4B
YOLOv5l	47.0	47.1	65.6	3.9ms	256	47.8M	88.1B
YOLOv5x	49.0	49.0	67.4	6.1ms	164	89.0M	166.4B

Tabla 1.1: Arquitecturas implementadas en Yolov5 junto con su desempeño. Las arquitecturas con el mayor número de parámetros sacrifican tiempo de procesamiento (representado en las columnas SpeedGPU y FPSGPU) pero ganan precisión (APval y APtest). (Jocher et al., 2020),

1.4.3 *Re-entrenamiento*

Los creadores de la arquitectura recomiendan re-entrenar mediante GPU. Asimismo, proveen de un conjunto de herramientas en el repo-

sitorio que permiten re-entrenar cualesquiera de las redes disponibles para realizar una nueva tarea (ver Figura 1.14).

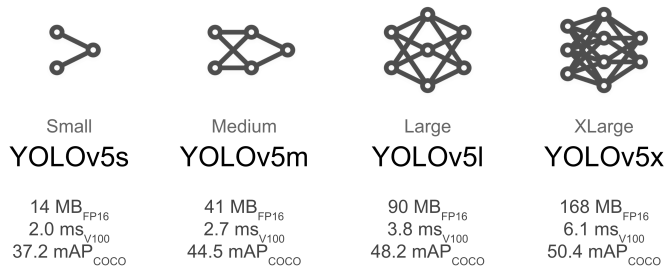


Figura 1.14: Marco teórico: Diagrama de las distintas arquitecturas Yolov5 y sus características (Jocher et al., 2020).

Entre los principales parámetros disponibles para manipular se encuentran:

- Batch: el número de imágenes que evaluará la red por cada paso de entrenamiento y re-calibración.
- Epochs: cuántas veces se entrenará con toda la muestra de entrenamiento.
- Data: el conjunto de entrenamiento y validación con los que se calibrará y evaluará el desempeño del nuevo clasificador.
- Weights: los pesos que usará inicialmente el modelo del cual partirá el entrenamiento. Los pesos disponibles son las arquitecturas antes mencionadas (en la Tabla 1.1 y Figura 1.14). Los pesos son el resultado de 300 épocas (*Epochs*) de entrenamiento. El valor del parámetro se asigna mediante el nombre de la arquitectura, por ejemplo: yolov5s.

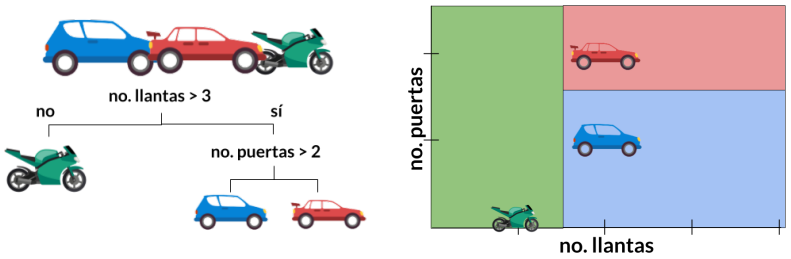
El framework necesita ejecutarse en python, en una versión igual o mayor a la 3.8. Las dependencias base necesarias para re-entrenar y ejecutar los modelos del proyecto son las siguientes:

- cython
- matplotlib
- numpy
- opencv-python
- pillow
- PyYAML
- scipy
- tensorboard
- torch
- torchvision
- tqdm

1.5 MODELOS BASADOS EN ÁRBOLES

1.5.1 *Introducción a los modelos basados en árboles*

Una de las implementaciones más usadas en el ámbito de Aprendizaje de Máquina es la de los modelos basados en árboles. Los modelos basados en árboles son un tipo de modelos predictivos que utilizan cortes en las variables observadas para segmentar en áreas simples la variable objetivo y de esta forma poder asociar un valor a observaciones dependiendo de su región. Estas "reglas" pueden tomarse y sumarizarse en una estructura de árbol, a esta implementación se le conoce como *Árbol de Decisión* (Gareth et al., 2021, p.327).



(a) Diagrama de cortes en Árbol de Decisión

(b) Segmentación del espacio vectorial de las observaciones

Figura 1.15: Diagramas ejemplo de árbol y área de decisión

El corte realizado por las variables puede ser definido de distintas formas, sin embargo, para el caso de clasificadores es bastante popular y usual tomar el índice "Gini" como métrica para obtener la variable objetivo.

La medida Gini introducida por el estadístico Italiano Corrado Gini, es una medida usada habitualmente para medir la desigualdad en ingresos. El coeficiente toma valores entre cero (perfectamente equilibrado, por ejemplo: todos cuentan con el mismo ingreso) y uno (perfectamente desequilibrado, por ejemplo: todos los ingresos se concentran en una persona) (Luebker, 2010).

El índice Gini (G) esta definido como:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{1.9}$$

En la ecuación 1.9 $\hat{p}_{mk}$ representa la porción de registros de la región m que pertenecen a la clase k. Hay que notar que es una medida de varianza entre clases; cuando los $\hat{p}$ contienen valores extremos entre 0 y 1 la métrica gini toma valores pequeños. Por esta

cualidad la métrica gini se asume como una medida de impureza de los cortes, lo cual permite identificar que, con valores pequeños, se obtendrán regiones con una clase predominante (Gareth et al., 2021, p.336).

Sus hiper-parámetros más comunes son: profundidad, cantidad mínima para considerar un corte nuevo, número máximo de hojas y número de registros mínimos dentro de una hoja durante él entrenamiento.

Las principales ventajas del uso del Árbol de Decisión son por un lado, su fácil interpretación; es relativamente sencillo explicar la toma de decisiones de los modelos basados en árboles, se pueden tomar las reglas definidas para explicar cómo se determina la variable objetivo. Árboles sencillos pueden ser explicados inclusive por personas no-expertas. Por otro lado, no necesita un tratamiento previo para reescalar las variables o transformar en variables "dummies", además de que se puede trabajar con valores nulos.

Las desventajas son su bajo poder predictivo comparado con su ensamble, el Bosque Aleatorio, además de no ser un modelo robusto (cambios en las variables de entrada pueden impactar el modelo resultante) (Gareth et al., 2021, p.339).

En cuanto a las tareas, el Árbol de Decisión puede ser usado tanto para clasificar como para problemas de regresión. Para este trabajo se vuelve relevante abordar los árboles de clasificación, en particular el ensamble de *Bosques Aleatorios*, el cual cuenta con una implementación en modo clasificador.

1.5.2 *Bosques Aleatorios*

Un Bosque Aleatorio es un tipo de ensamble de modelos que entrena dos o más Árboles de Decisión mediante una estrategia que decorrelaciona los árboles. Esto lo logra realizando un muestreo aleatorio de las variables predictoras (Gareth et al., 2021, p.343-345).

Un ensamble es un método que combina varios modelos simples con el objetivo de obtener un solo y potente modelo. Los modelos que lo conforman son llamados modelos débiles. El adjetivo "débil" es usado porque generalmente estos modelos suelen tener predicciones mediocres por si solas (Gareth et al., 2021, p.340).

El ensamble Bosque Aleatorio está conformado por B árboles de decisión (se les llama estimadores). Cada uno de ellos entrena con una muestra y conjunto de variables. Cuando se busca entrenar a los

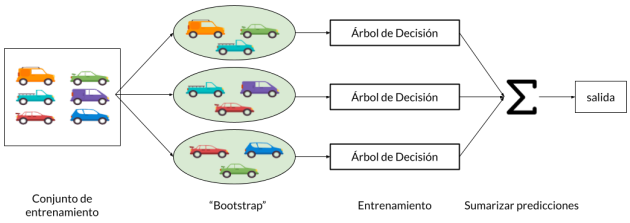


Figura 1.16: Diagrama del proceso de entrenamiento de un Árbol Aleatorio.

árboles con distintos conjuntos de observaciones se realiza la estrategia "Bootstrap". Bootstrap es una estrategia que permite muestrear B conjuntos de observaciones de tamaño M. El muestreo suele ser con remplazamiento (es decir, pueden repetirse observaciones entre los conjuntos) (ver 1.16).

Entre los hiper-parámetros disponibles se encuentran: número de estimadores (árboles de decisión a entrenar), estrategia de muestreo, cantidad máxima de predictores por árbol (Breiman recomienda asignar $\sqrt{total\_predictores}$ para evitar sobre-entrenamiento) y los parámetros disponibles en los árboles de decisión (profundidad, número máximo de hojas, etc.) (Breiman, 2001). Se dice que un modelo ha sido sobre-entrenado cuando su desempeño fuera del conjunto de entrenamiento decae abruptamente. Es un efecto que puede ser causado por una pobre definición del conjunto de datos con los que se entra o por usar modelos en exceso complejos con conjuntos de entrada ruidosos o pequeños. Una forma de detectar sobre-entrenamiento es validar mediante datos fuera del conjunto de entrenamiento, buscando diferencias significativas en las métricas de desempeño del modelo a evaluar (Gareth et al., 2021, p.32).

Para los problemas de regresión se promedian las salidas de los árboles (Bagging). Para los de clasificación se emplea un sistema de votación (la salida es la clase más votada). Durante estas votaciones se promedian las respuestas de los arboles entrenados durante el proceso de "Bootstrap", de esta forma obtenemos la respuesta promedio de cada uno de los estimadores (corresponde a la etapa "Sumarizar predicciones" en la figura 1.16).

Las ventajas del ensamble son: buenos resultados sin necesidad de ajuste de hiper-parámetros, se mantiene estable con muestras (se muestrea para cada árbol y los resultados son promediados), al aumentar el número de árboles se previene el riesgo de sobre-entrenamiento, interpretación (es posible conocer el peso de las variables tomando la importancia de variable, medida calculada con el promedio del gini que aporta cada variable, el proceso es distinto al del

árbol de decisión donde basta con obtener la estructura de los cortes).

Las desventajas identificadas son: aumenta el costo computacional con respecto al árbol de decisión y el modelo Gradient Boosting Machine (GBM) parece tener mejor desempeño en condiciones similares (Hastie y Friedman, 2009, p.587-604).

2

MÉTODOS

2.1 OBTENCIÓN DE ARCHIVOS

En la página https://archivosdelarepresion.org/ hay una sección de archivos en donde se encuentra un link hacia el Archivo sin catalogar(ver Figura 2.1).

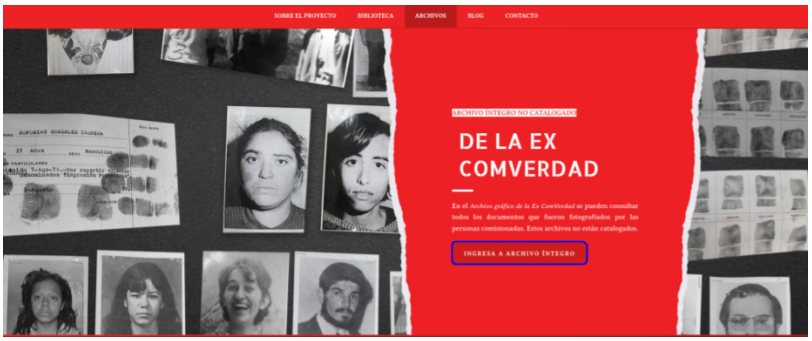


Figura 2.1: Sección Archivo en web: https://archivosdelarepresion.org/

Una vez dentro de la ruta https://archivo.archivosdelarepresion.org/, es posible ver los archivos intactos sin clasificar. Archivos que las personas comisionadas fotografiaron, entre ellos los ficheros (ver Figura 2.2).

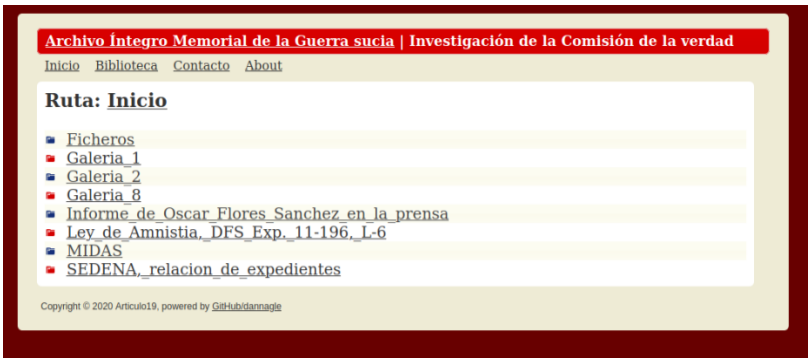


Figura 2.2: Directorio en: https://archivo.archivosdelarepresion.org/

Para descargar los ficheros, se pueden bajar una por una las fotografías. Sin embargo, esto sería muy tardado, tomando en cuenta que el número total de imágenes en toda la carpeta de ficheros es de 18,008. Además, la estructura del directorio también contiene información relevante. Es por eso que se decidió desarrollar una herramienta que, mediante técnicas de web-scraping, pueda descargar la totalidad de los documentos manteniendo la estructura original.

Se le llama web-scraping al proceso de extraer información de una página web o pantalla de computadora y depositarlo dentro de un documento ordenado o en una computadora. Frecuentemente, es usado en proyectos de Ciencia de Datos.

Para descargar las imágenes mediante web-scraping, primero se ingresa con el explorador de archivos del servidor, agregando */Ficheros* a la ruta, obteniendo la ruta: https://archivo.archivosdelarepresion.org/Ficheros/. Una vez ahí, se analizará con la herramienta del desarrollador del navegador la estructura del html (ver figura 2.3).

Figura 2.3: Directorio de los Ficheros vista desde herramientas de desarrollador: https://archivo.archivosdelarepresion.org/Ficheros/

Se encontró que el servidor devuelve la ruta, nombre de los ficheros y carpetas de la misma forma, usando las etiquetas *table*, *tr*, *td* y *a*, la implementación del código usando el lenguaje de programación python y la explicación del proceso puede encontrarse en el Anexo A.1

2.2 RECONOCIMIENTO ÓPTICO DE CARACTERES (OCR)

En el ámbito del procesamiento de imágenes, se le conoce como Reconocimiento Óptico de Caracteres (Optical character recognition (OCR) por sus siglas en ingles) al proceso de extraer electrónicamente el texto (escrito a mano, con máquina o publicado en sitios web) mediante imágenes. Entre sus aplicaciones se encuentran el reconocimiento de escritura a mano, la industria legal, la banca, la medicina, el procesamiento de Captcha, sistemas de repositorios y librerías digitales.

Su uso en librerías digitales es de particular interés para este proyecto. Parafraseando un poco la descripción que se da en el documento de IJMLC sobre la utilización del OCR en los repositorios institucionales: el principal objetivo de los repositorios institucionales y de las librerías digitales, es poner a disposición y facilitar el intercambio de conocimiento; es decir, proporciona literatura de libre acceso. Bajo este enfoque, el OCR funciona como herramienta para escanear do-

cumentos físicos y coleccionar la información contenida en estos de forma digital para su posterior explotación.(Singh et al., 2012)

En este trabajo se utilizará Tesseract (Tesseract-ocr,s.f) como librería OCR. Este paquete contiene una implementación basada en redes neuronales y es capaz de evaluar imágenes de forma local. Otro aspecto considerado durante la elección de la tecnología a usar para el OCR fue que Tesseract es una librería de código abierto y esto permite implementarlo sin necesidad de costes en licencias.

2.2.1 *Limpieza*

Para la preparación de las imágenes, Tesseract recomienda el siguiente flujo (Tesseract-OCR, Consultado en 2022):

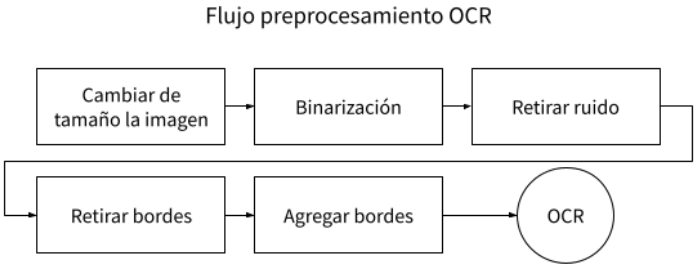


Figura 2.4: Flujo de pre-procesamiento sugerido por tesseract

.

Se eligió este flujo para aplicar en los Ficheros del AR(ver Figura 2.5).

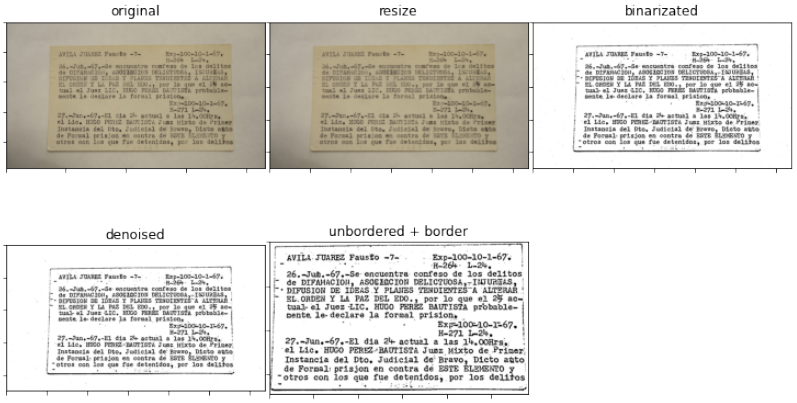


Figura 2.5: Flujo de pre-procesamiento aplicado a una ficha de la DFS.

A continuación se describen los pasos junto con piezas de código desarrolladas para aplicarlos a los ficheros de la DFS. La clase desarrol-

lada para implementar en su totalidad la etapa de pre-procesamiento de la imagen puede ser consultada en el anexo A.3

2.2.2 *Re-configuración del tamaño de la imagen*

Tesseract OCR discute la influencia de la resolución de la imagen en la calidad de la traducción. Como primera recomendación, sugiere que la imagen tenga por lo menos 300 dpis(300px X 300px). Asimismo, proporciona la información de la figura 2.6[1].

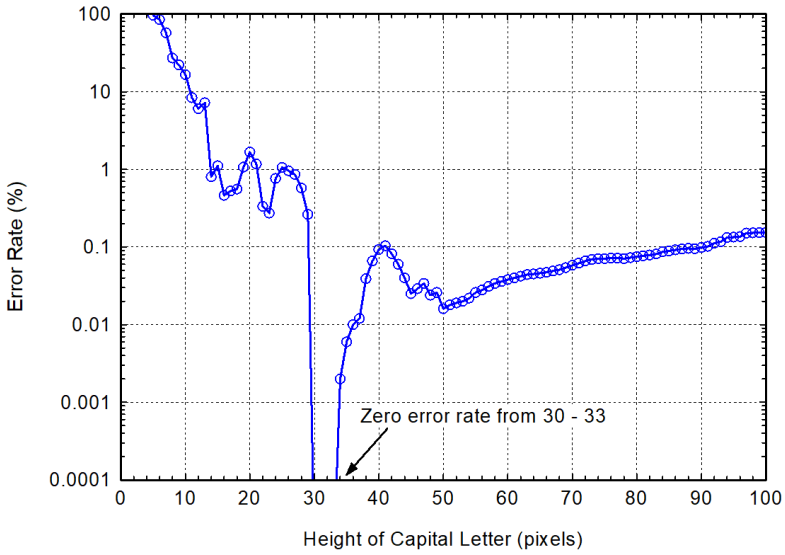


Figura 2.6: Relación entre el tamaño de la fuente en la imagen y la calidad de la traducción.

Partiendo de la discusión anterior, se puede deducir que uno de los procesos a realizar será el de definir los parámetros para el cambio de tamaño de las imágenes. Esto antes de procesarlas con el OCR y con los cuales tengamos mejor evidencia de desempeño.

De cualquier forma, para Python la función sugerida para realizar esta etapa es la siguiente:

Código 2.1: Función para cambiar de tamaño la imagen.

```
def resizing_img(img, height=900):
        return imutils.resize(img, height = height)
```

1 Gráfica tomada de: https://groups.google.com/g/tesseract-ocr/c/Wdh_JJwnw94/m/24JHDYQbBQAJ

2.2.2.1 *Binarización*

En esta etapa, se busca convertir la imagen que suele ir del rango RGB en valores entre el 0 y el 255 a una imagen que contenga solo blanco o negro. Aquí nos basamos en el método sugerido por opencv[2] (ver 2.2).

Código 2.2: Función para aplicar binarización a una imagen.

```
def binarization_img(img):
    """
        Based on: https://docs.opencv.org/master/d7/d4d/
            tutorial_py_thresholding.html
    """
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY);
    img = cv2.medianBlur(img,5)
    thld = cv2.adaptiveThreshold(img,255,cv2.
        ADAPTIVE_THRESH_GAUSSIAN_C,
                cv2.THRESH_BINARY,11,2)
    return thld
```

Tesseract realiza binarización a las imágenes internamente (su implementación funciona con imágenes blanco y negro), no obstante, se sugiere que se use una implementación propia del algoritmo ya que la implementación por defecto es genérica y puede tener resultados subóptimos(Tesseract-OCR, Consultado en 2022).

2.2.2.2 *Retirar ruido*

El ruido es una variable aleatoria que suele estar presente en todas las imágenes debido a factores como el brillo o la luz. La etapa de binarización no retira ciertos tipos de ruido. Tener este ruido puede dificultar el funcionamiento de los métodos OCR, es por eso que mejorar el proceso que retira el ruido mejora la lectura de las imágenes.

Código 2.3: Función para retirar ruido en imagenes.

```
def denoise_img(img):
    """
        Based on: https://opencv24-python-tutorials.
            readthedocs.io/en/latest/py_tutorials/py_photo/
            py_non_local_means/py_non_local_means.html
    return cv2.fastNlMeansDenoising(img, None, 20,7,21)
```

El método es una implementación del trabajo: *Non-Local Means Denoising* (Buades et al., 2011). Parafraseando la explicación del método,

2 Método basado en: https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html

el proceso se basa en la suposición de el ruido (n) como variable aleatoria de media 0.

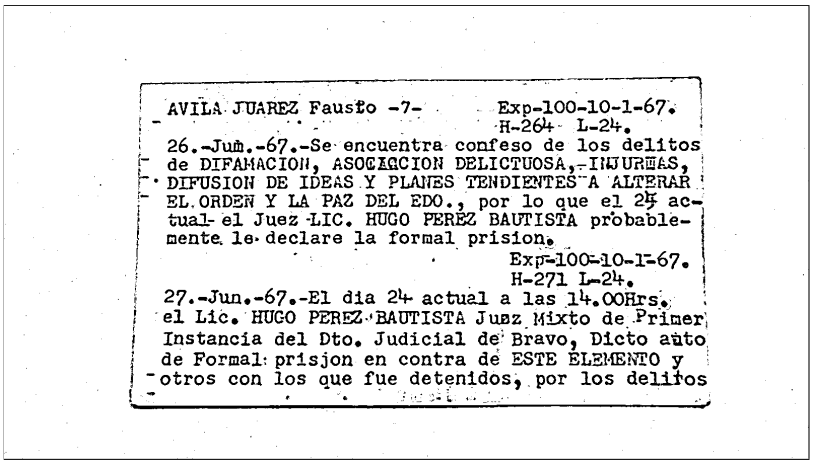


Figura 2.7: Ficha ejemplo de la figura 2.4 evaluada en la etapa de binarización y retirado de ruido.

El proceso toma regiones de píxeles parecidas dentro de la imagen asumiendo que se tratan de la misma imagen pero con ruido n. Por lo tanto el pixel $p = p_0 + n$, donde p_0 es el valor del pixel real y p es su valor con ruido. Para retirar el ruido el método toma todos estos recuadros y los promedia, como el promedio del ruido es n (basados en la suposición), el valor del mismo pixel promediado en todas las imágenes, se espera que el valor del pixel tienda a su valor real.

2.2.2.3 *Retirar y agregar bordes*

En esta etapa se sugiere retirar los bordes que no pertenecen al área de texto. Entre mayor sea el área de la imagen distinta a la superficie del texto, existe la posibilidad de leer caracteres dentro de los cambios de tonalidad en la frontera del área del texto (Tesseract-OCR, Consultado en 2022). Además, se recomienda tener un borde para facilitar la lectura de los caracteres que se encuentran en ellos. Para esto, se propone agregar un borde blanco. No obstante, dado que las imágenes suelen tener ya un borde pre-establecido, se decidió agregar píxeles de la imagen original (ver la figura 2.8). El no agregar el borde repercute directamente con las letras en la frontera de la imagen.

2.2.2.4 *Aplicado del algoritmo*

Para aplicar el algoritmo basta con instalar los binarios y la librería pytesseract, el conector de python para tesseract (ver código 2.4).

Para su uso tan solo se necesita llamar a la función **image_to_string**. La función recibe como parámetros: la imagen a analizar y configuración adicional.

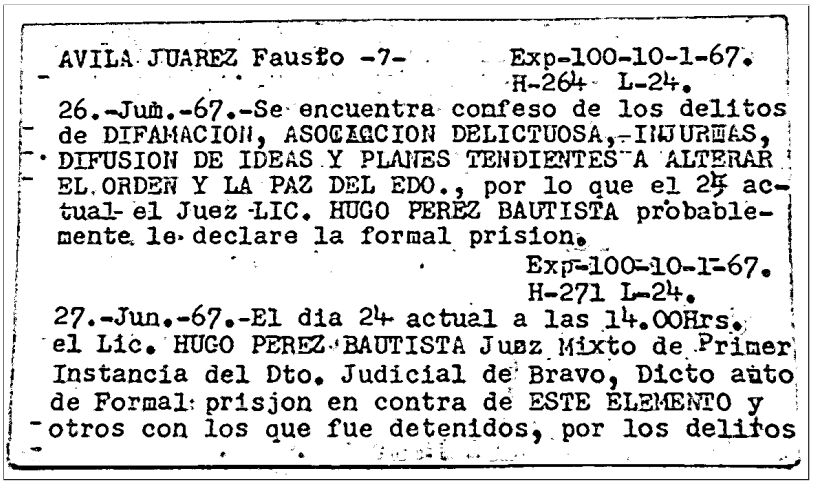

AVILA JUAREZ Fausto -7- Exp-100-10-1-67.
H-264 L-24.
26.-Jun.-67.-Se encuentra confeso de los delitos de DIFAMACION, ASOCIACION DELICTUOSA, INJURIAS, DIFUSION DE IDEAS Y PLANES TENDIENTES A ALTERAR EL ORDEN Y LA PAZ DEL EDO., por lo que el 25 actual el Juez LIC. HUGO PEREZ BAUTISTA probablemente le declare la formal prision.
Exp-100-10-1-67.
H-271 L-24.
27.-Jun.-67.-El dia 24 actual a las 14.00Hrs. el Lic. HUGO PEREZ BAUTISTA Juez Mixto de Primer Instancia del Dto. Judicial de Bravo, Dicto auto de Formal prisjon en contra de ESTE ELEMENTO y otros con los que fue detenidos, por los delitos

Figura 2.8: Ficha ejemplo de la figura 2.4 evaluada en la etapa de retirar y agregar bordes. Retirar los pixeles que no corresponden al fichero nos ayudara a evitar tener falsas lecturas.

Código 2.4: Comandos para instalar tesseract en Ubuntu y Python.

```
# instalaci n en ubuntu
$ sudo apt install tesseract-ocr
$ sudo apt install libtesseract-dev

# instalaci n de la librer a usando el gestor de packetes de
    python
$ pip install pytesseract
```

2.3 BÚSQUEDA DE HIPER-PARÁMETROS

Parafraseado de *Stack Exchange*, en el contexto estadístico, los hiper-parámetros son aquellos que se proporcionan al modelo, por ejemplo en una Red Neuronal: número de nodos, capas, variables de entrada, ratio de aprendizaje, función de activación, etc. Por otra parte, los parámetros son aquellos que pueden ser aprendidos mediante algún proceso estadístico, por ejemplo en una Red Neuronal: Los pesos de la red (Prasad, 2018).

En el método propuesto encontramos los siguientes hiper-parámetros:

- Height: altura de la imagen resultante en el paso de re-escalamiento (Es un parámetro entero, positivo y con rango de 0 a *altura_imagen*).
- Border_size: cantidad de píxeles que dejaremos al borde de la imagen al recortar la ficha (Es un parámetro entero, positivo, el proceso tomara el mínimo entre la cantidad restante de la imagen y el valor proporcionado).

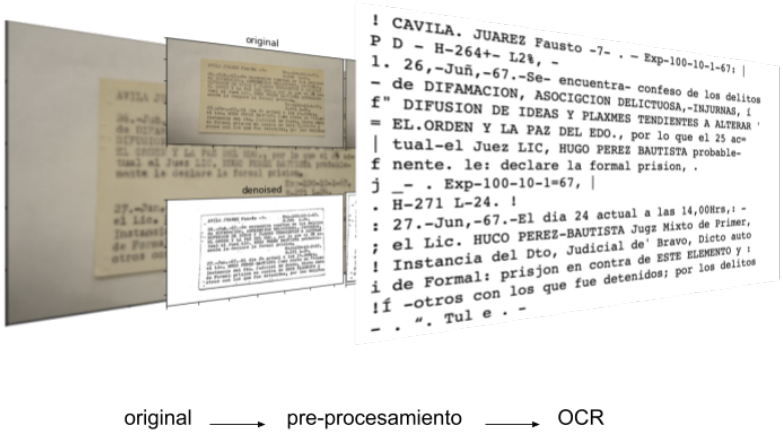
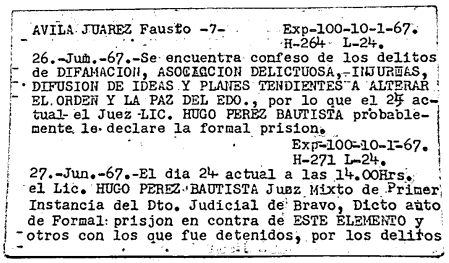


Figura 2.9: Flujo aplicado a un fichero de los AR.



(a) Fichero ejemplo después de pasar por el flujo sugerido (ver Figura 2.4).

(b) Texto obtenido con el flujo propuesto

Figura 2.10: Fichero ejemplo procesado en la herramienta OCR.

- Sigma: parámetro que modifica el umbral de detección de bordes (el parámetro acepta valores de punto flotante entre el 0.0 y el 1.0).

Alguna variación en los valores de uno de los parámetros puede ayudar o entorpecer el desempeño de la lectura. Dado que el objetivo es obtener la mayor cantidad de expedientes posibles, se usará como medida de "desempeño" la cantidad de expedientes obtenidos por cada combinación de parámetros y fichas.

Se seleccionó una muestra aleatoria de 500 fichas junto con los siguientes posibles valores:

- height: [900, 1100, 1200, 1300, 1700].
- border_size: [15,20,25].
- sigma: [0.33, 0.01].

Se eligieron los parámetros de esta forma en el siguiente orden: para el parámetro *height* se tomó el valor más pequeño de las imá-

genes de nuestro repositorio (1,728), a partir de ahí se centraron los otros 4 valores en intervalos de 300, centrados en 1,000.

Para el parámetro *border_size* se decidió usar los valores 15, 20, 25. No fue posible encontrar mucha documentación para este paso, así que se determinó buscar dentro de este conjunto.

En cuanto el parámetro *sigma*, este trabajo va guiado por las recomendaciones hechas por el autor del método usado (Rosebrock, 2015).

Tomando en cuenta 500 fichas, 5 posibles valores para *height*, 3 valores posibles para *border_size* y 2 parámetros para el parámetro *sigma*, el total de evaluaciones a realizar es de: $500 * 5 * 3 * 2 = 1,500$. Para generar la rejilla de evaluación, se hizo uso de la librería scikit-learn con la función ParameterGrid (Buitinck et al., 2013).

Para evaluar la "calidad" de la lectura en este punto del trabajo, se empleó una búsqueda RegEx. También se buscaron patrones parecidos a los que tiene un expediente (la clave de expediente): dígitos seguidos por guiones (ver código 2.5).

Código 2.5: Funciones que detectan expedientes, fechas y evalúan fichas dependiendo de los parámetros de entrada.

```
import re
is_exp = lambda txt: re.search("(\d{1,3})(-)(\d{1,3})(-)(\w{1,3})
    ", txt) if txt is not None else None
is_date = lambda txt: re.search("(\d{2})-([a-zA-Z]{3})-(\d{2})",
    txt) if txt is not None else None
```

En este punto no fueron consideradas todas las posibilidades, como que el OCR detecte dobles guiones o caracteres especiales parecidos. No se tomaron en cuenta porque se necesitaba delimitar el alcance de la búsqueda RegEx y el objetivo en esta parte del trabajo era obtener los hiper-parámetros que mejor funcionaban para obtener la clave de expediente de forma integra por el OCR.

De cualquier forma, se asumió que basta con la búsqueda antes mencionada para generar los mejores hiper-parámetros. Por último, se desarrolló una función que, dependiendo del tamaño, devuelve los mejores parámetros encontrados durante la búsqueda de hiper-parámetros. Para este proceso se usa la distancia euclidiana (ver código 2.6).

Código 2.6: Función que sugiere la mejor configuración de hiper-parámetros dado el tamaño de la imagen a evaluar.

```
configs = [{"h":1728, "w":2304, "sigma":0.33, "pixels":1200, "
    border":15},
{"h":2432, "w":4320, "sigma":0.01, "pixels":900, "border":20},
{"h":3456, "w":4608, "sigma":0.01, "pixels":1300, "border":25},
{"h":2304, "w":1728, "sigma":0.33, "pixels":1300, "border":20},
{"h":1944, "w":2592, "sigma":0.33, "pixels":1300, "border":15},
{"h":4608, "w":3456, "sigma":0.33, "pixels":1200, "border":15},
{"h":3264, "w":2448, "sigma":0.33, "pixels":1300, "border":20},
{"h":2448, "w":3264, "sigma":0.33, "pixels":1200, "border":20}]
def get_nearest_conf(shape):
    """
        Function that returns best parameters found given a shape
            (height, width), for each possible configuration it
            uses euclidean distance to determinate which one fits
             better, minimezing the distance.
    """
    return min(configs,key=lambda cnf: np.sqrt(((cnf["h"]-shape
        [0])**2)+((cnf["w"]-shape[1])**2)))
```

Con la intención de tener la mayor eficiencia (mejores tiempos de evaluación) en el uso de los recursos destinados a la lectura y procesamiento de las fichas, se encontró que definir la variable de entorno **OMP_THREAD_LIMIT** en **1** y usando procesamiento en hilos puede haber una mejora sustancial (se obtuvo un tiempo de ejecución 124% mejor usando la variable de entorno en 1, evaluando el flujo en 100 ficheros).

El resultado de la búsqueda de los mejores hiper-parámetros fue almacenado en formato csv.

2.4 PRIMERA EJECUCIÓN

Una vez obtenidos los mejores parámetros, se ejecutó la totalidad de las fichas con las mejores configuraciones encontradas para cada resolución (ver Código 2.6)

Se empleó la clase **PipelineImg**. Esta clase se desarrolló a partir de los hallazgos en la sección 2.2. El contenido de la clase se encuentra en el anexo A.2.1, y su funcionamiento es el del flujo presentado en la figura 2.4.

2.5 EJECUCIÓN OBTENIENDO INFORMACIÓN DE LA TRANSFORMACIÓN Y LOS DATOS DE LA LECTURA

Para obtener la mayor cantidad de información posible no basta con extraer el texto, como se realizó en la primera ejecución (ver 2.4). Se tiene que extraer la confianza de la lectura de cada palabra y su posición en los píxeles de la imagen.

En la segunda ejecución, se modificó la clase **PipelineImg** para salvar el área de recorte. En la evaluación se cambió la función **image_to_string** (forma en la que obteníamos el texto con anterioridad) por la función **image_to_data**. Esto se hizo para poder tener la meta-data de cada palabra encontrada (ver tabla 2.1). Por último, se cambió el uso del algoritmo que ajusta la rotación en imágenes con texto rotado por una rotación de 90 grados cuando la imagen fue tomada de forma vertical.

2.6 DETECCIÓN DEL TAMAÑO DE LA FUENTE

Después de la segunda ejecución y habiendo obtenido el dato de las palabras, se decidió analizar el impacto del tamaño de la fuente con la cantidad de expedientes obtenidos. Para esto se filtró siguiendo el flujo representado en la figura 2.11.

1. Lectura de la ficha en modo image_to_data (ver tabla 2.1).
2. Filtro para mantener las palabras que tengan una confianza mayor a 70 (se usa el atributo *conf*).
3. Filtrar palabras con por lo menos una letra o número.
4. Filtro para retirar palabras con tamaño de fuente atípica.

En el primer paso, la apariencia de la meta-data obtenida fue acomodada de forma tabular para mayor comprensión (ver 2.1).

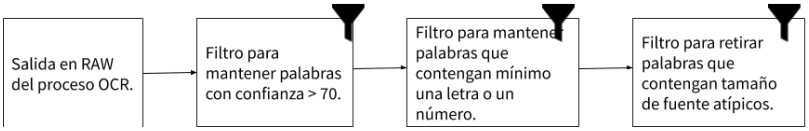


Figura 2.11: Flujo para obtener la fuente de las fichas utilizando la salida del OCR.La medida de confianza es devuelta por Tesseract-OCR durante la lectura en modo **image_to_data**. La figura 2.12 contiene un ejemplo de una ficha evaluada en el flujo.

Se observa que inicialmente el proceso OCR arroja bastantes lecturas, pero no todas son confiables. Del ruido que podrían causar, se debe tomar en cuenta el tamaño de la fuente de palabras que no existen en la imagen, que no son parte de la ficha o que corresponden a

left	top	width	height	conf	text
2071	779	206	87	14.507744	-2-
934	640	3083	128	-1	
1193	1064	123	62	95.452759	se
3377	2185	1052	120	91.110443	interrogarsele
2769	2470	205	44	72.269279	Omo
2511	1777	134	93	95.188423	la
2233	2155	150	19	32.839806	ave.
3795	1815	144	69	93.121239	en

Tabla 2.1: Muestra de 8 palabras y los atributos devueltos durante la lectura de la ficha de la figura 2.12.

signos de puntuación (ver figura 2.12), el código que fue desarrollado para implementar este flujo de limpieza se encuentra en el código 2.7.

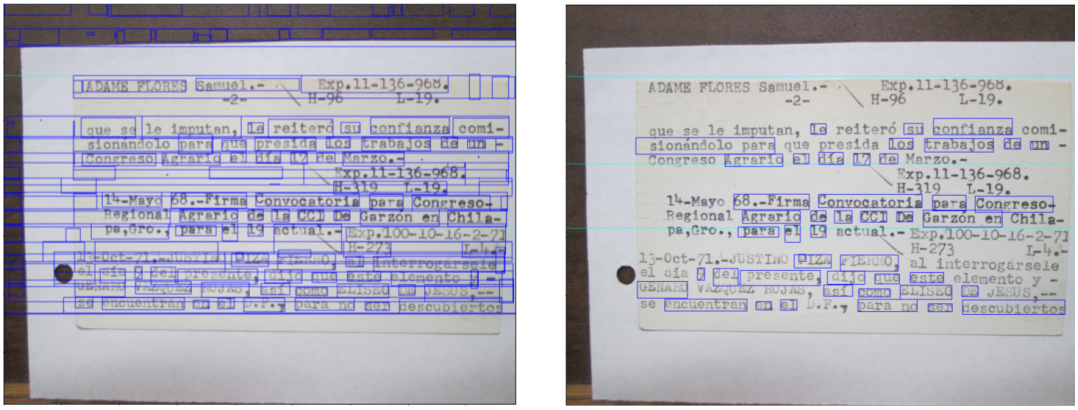


Figura 2.12: Ambas imágenes corresponden a la misma ficha. La de lado izquierdo tiene seleccionadas todas las palabras encontradas por el OCR, la de lado derecho solo aquellas que cumplen con los filtros para el tamaño de fuente.

Por último, después de la depuración de las palabras de las fichas, se utilizó el promedio del tamaño de las palabras como una forma de representar el tamaño de la fuente de las mismas (ver figura 3.1).

Código 2.7: Función que limpia la salida de Tesseract-OCR usando el flujo definido en la figura 2.11

```
import re
def filter_words(feature_dataframe):
    """
        This function filters the ocr reading according to the
            font size filtering stream.
        Confidence filter > 70
        Word with atleast one number or letter filter
        Words with font between -2 or +2 standard deviation from
            its median filter

        @param: dataframe
        :return dataframe
    """
    #This line defines the pattern which detect words with
        atleast one letter or number
    regex_pattern = "[A-Za-z0-9]"
    #This function seachs previus pattern and returns True if it
        detect the pattern
    filter_words_fun = lambda txt: (re.search(regex_pattern,txt)
        is None) == False
    #This line applies confidence filter
    conf_filter_dataframe = feature_dataframe[(feature_dataframe[
        "conf"].apply(lambda el: el.split(".")[0]).astype(int)
        >70)]
    #This line applies word content filter
    word_filter_dataframe = conf_filter_dataframe[
        conf_filter_dataframe["text"].apply(filter_words_fun)]

    #Mean and std are used by outlayer filter
    mean_value =  np.mean(word_filter_dataframe["height"])
    std_value = np.std(word_filter_dataframe["height"])
    #This line applies outlater filter
    outlayer_filter_dataframe = word_filter_dataframe[
                                        abs(word_filter_dataframe
                                            ["height"] -
                                            mean_value) < 2 *
                                            std_value
                                ]
    return outlayer_filter_dataframe
```

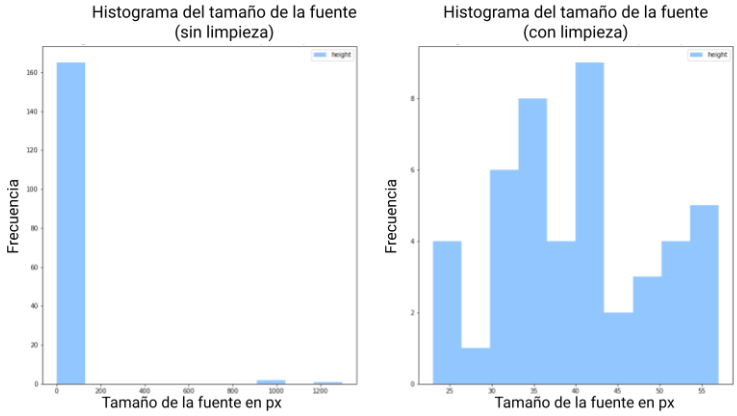


Figura 2.13: Ambos histogramas corresponden a la misma ficha. La de lado izquierdo tiene seleccionadas todas las palabras encontradas por el OCR. La de la derecha solo aquellas que cumplen con los filtros para el tamaño de fuente.

2.7 METODOLOGÍA PROPUESTA BASADA EN TÉCNICAS DE APRENDIZAJE DE MÁQUINA PARA LA EXTRACCIÓN DEL TEXTO EN LOS FICHEROS DE LA DFS

En las secciones anteriores ha sido presentada la metodología en el orden en el que fue desarrollada. Equipos trabajando en documentos relacionados con graves violaciones a derechos humanos pueden tener problemas similares, por lo cual es de interés lo siguiente:

- Mostrar la lógica detrás del proceso.
- Demostrar que el ajuste de los hiper-parámetros y pruebas de validación son necesarias.
- Mostrar ideas para diagnosticar y solucionar problemas.

Durante la lectura del OCR en las fotografías de la DFS, se identificaron los siguientes subproblemas:

- Delimitación del área correspondiente al texto de la ficha.
- Identificar cuando la imagen corresponde a la fotografía de una ficha.
- Identificar la rotación correcta de la imagen.
- Obtener una medida de calidad de la lectura.

Por los puntos anteriores se identificó la necesidad de etiquetar una parte de la muestra para generar modelos predictivos supervisados. Para esto se desarrolló una aplicación web en Flask que permite anotar en una base de datos (con una sola tabla cuyo diagrama y diccionario de datos puede ser consultado en el anexo A.3.1) la rotación de la imagen, las coordenadas de la ficha en la imagen, si la imagen contiene fichas de la DFS, etc. (ver sección 2.14).

Flask es un marco de trabajo ligero para aplicaciones web, que ofrece sugerencias, pero no impone dependencias ni diseño para el proyecto. Está diseñado para que comenzar sea ligero y rápido. Depende del desarrollador elegir las herramientas y bibliotecas que desea utilizar(Flask, Consultado en 2022). Por lo anterior decidimos implementar el anotador en Flask.

Las meta-data disponibles para anotar en la base de datos mediante la aplicación fueron las siguientes:

- **is_expediente**: 1 en caso de que la imagen contenga un fichero o en caso contrario.
- **has_exp**: bandera identificando si el fichero contiene un expediente anotado.

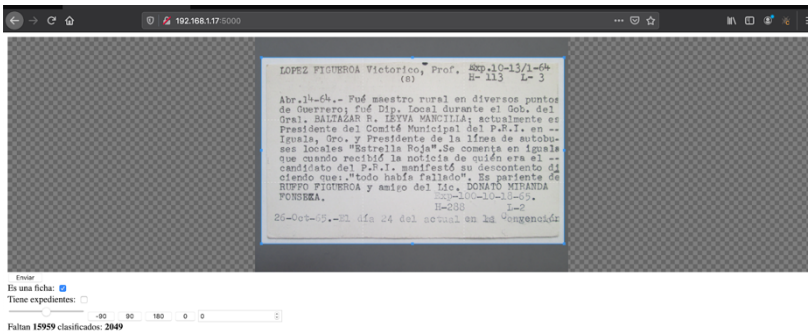


Figura 2.14: Aplicación usada para etiquetar un conjunto de las fichas.

- **angle**: ángulo necesario para alinear correctamente el texto en la imagen.

La metodología propuesta se encuentra reflejada en los siguientes pasos (ver figura 2.15):

Detector de fichas: se identifica la ficha y el área correspondiente en píxeles de la ficha dentro de la imagen. El proceso se realiza rotando la imagen en los ángulos 0, 90, 180, 270. Además de servir para delimitar el área correspondiente a la ficha, funciona para decidir si la imagen corresponde a una de ellas.

Detector de rotación y Detector de calidad OCR: Se ejecuta una lectura OCR para cada rotación y se evalúa en un modelo que identifica cuál texto corresponde a la rotación correcta de la imagen. Se usa la propensión para determinar la calidad de la lectura.

2.7.1 *Construcción del anotador web*

Algunas de las soluciones gratuitas en el mercado permiten registrar imágenes con recuadros y anotar sus clases. Sin embargo, era necesario anotar información relevante como el ángulo de la imagen y si la imagen corresponde a una ficha. Lo anterior hizo factible construir un anotador a la medida.

La aplicación fue construida en Python mediante la librería Flask. El flujo de la aplicación está representado en el siguiente diagrama:

2.7.2 *Detector de fichas usando la arquitectura YOLOV5*

Para la delimitación de los ficheros y clasificación de las imágenes se anotaron un total de 2,000 imágenes, de las cuales se usaron 1,000 para el entrenamiento y 1,000 para la prueba (la partición fue grande porque 1,000 es la cantidad máxima que la plataforma de aumento de datos nos permitía subir, ver sección 2.7.2.1).

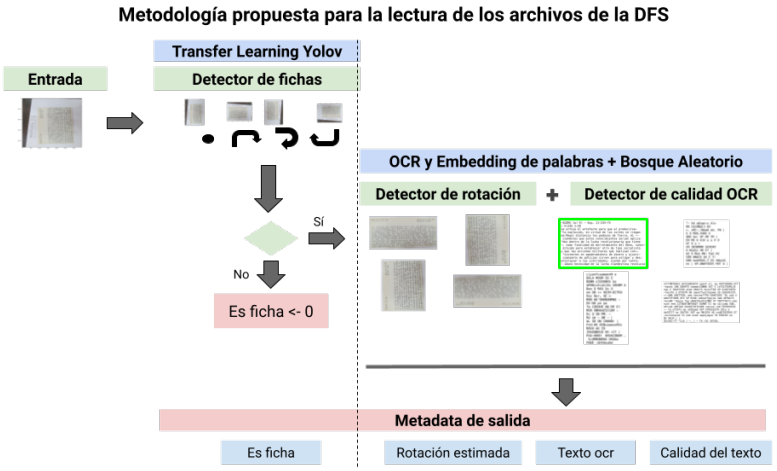


Figura 2.15: Flujo usado en la metodología para analizar los ficheros de la DFS. El flujo comienza por identificar si la imagen contiene un fichero de la DFS, se evalúa la imagen en 4 ángulos distintos (0, 90, 180, 270). Si el modelo Yolov5 encuentra una ficha en todos los ángulos, entonces suponemos que la imagen es una ficha (ver sección 2.7.3). El segundo paso es leer mediante OCR la ficha en las 4 rotaciones, el texto resultante se evalúa en un modelo Bosque Aleatorio para determinar cuál texto corresponde a la rotación correcta (ver sección 2.7.4). Finalmente, almacenamos el texto obtenido mediante la evaluación en la rotación correcta, y usamos la probabilidad obtenida del modelo como medida de calidad del texto (ver sección 2.7.5).

2.7.2.1 *Aumento de datos*

Con el propósito de que el modelo generalizara de forma correcta, se decidió realizar aumento de datos. Se rotaron las imágenes del entrenamiento en los ángulos 90, -90, 180 y 0 grados. El resultado del aumento de datos es pasar de 1,000 imágenes a 2,300 imágenes, cantidad determinada por la versión gratuita de la herramienta RoboFlow.

Roblow es una herramienta para transformar imágenes en información, de forma que pueda ser utilizada en flujos de aprendizaje automático (Dwyer, 2021). Entre sus funcionalidades relevantes para este trabajo están:

- Almacenar y organizar datos de imagen (se usó esta funcionalidad).
- Anotar imágenes (no utilizada porque era necesario anotar datos particulares de los archivos, datos mencionados en la sección 2.7).
- Procesar y aumentar las imágenes (se usó esta funcionalidad).

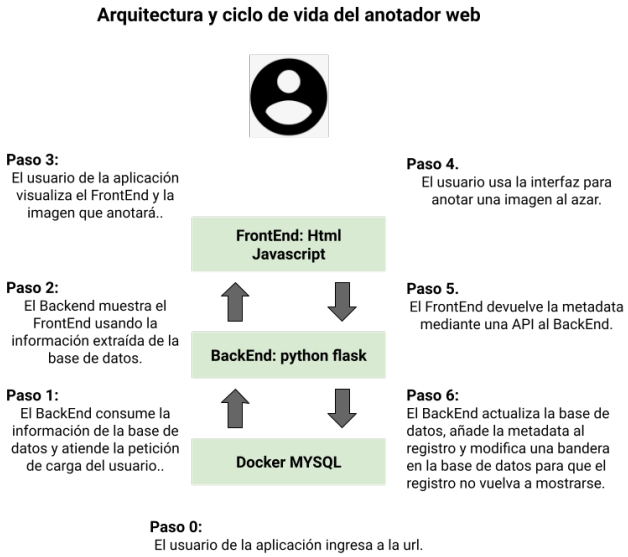


Figura 2.16: Arquitectura y ciclo de vida del anotador web. En el anexo A.3.1, se encuentran descritos la API, la url donde se desplegó la herramienta y la base de datos, tanto el FrontEnd y el BackEnd fueron desarrollados durante la realización de esta Tesis.

2.7.2.2 *Entrenamiento*

Para entrenar se usó la técnica llamada re-entrenamiento. Esta técnica consiste en congelar capas profundas de una arquitectura previamente entrenada para llevar a cabo alguna tarea, en este caso el reconocimiento de objetos en imágenes. Se agregan o re-entrenan las últimas capas de la arquitectura. De esta manera se ahorra tiempo de computación enfocado en representar en la arquitectura la detección de patrones comunes en tareas parecidas.

Dispositivo	Descripción
cpu	Intel Core i7-7700HQ
gpu	GTX 1050 4GB RAM
memoria ram	16GB

Tabla 2.2: Características de la computadora utilizada para el entrenamiento del modelo Yolov5.Las caracteristicas corresponden a una computadora ASUS VivoBook M580VD.

Yolov5 cuenta con esta característica, actualmente implementada en su repositorio oficial. En la sección 1.4.3 se habla de esta característica y los parámetros que pueden utilizarse para modificar la velocidad y el desempeño de la herramienta.

En el caso de los ficheros se re-entrenará con los parámetros (ver figura 2.8):

- Img, 640: la resolución de las imágenes de entrada.
- Batch, 16: se evaluarán 16 imágenes por cada paso de entrenamiento.
- Epochs,10: se entrenarán y evaluarán la totalidad de las imágenes 10 veces.
- Data: los datos del modelo son 2,301 imágenes como el conjunto de entrenamiento y 1,000 imágenes como conjunto de validación del modelo.
- Weights,yolov5s.pt: se entrenará con la arquitectura más pequeña por limitaciones en hardware (ver tabla 2.2), el desarrollador de la arquitectura no da pautas para el hardware recomendado, por otra parte, sí hace recomendaciones sobre el monitoreo de la memoria RAM del GPU (Jocher, 2021). En este caso el uso de memoria se desbordaba con las arquitecturas restantes, con esta arquitectura el uso de memoria fue del 97% considerando una memoria GPU de 4GB (ver tabla 2.2).

Código 2.8: Comando para entrenar con los ficheros de la DFS un clasificador Yolov5s.

```
train.py --hyp data/hyp.scratch.yaml --img 640 --batch 16 --
    epochs 10 --data ficheros.yaml --weights yolov5s.pt
```

2.7.3 *Detector de fichas*

Una vez obtenido el modelo identificador de fichas, se evalúa en la imagen rotada de 4 formas distintas: 0,90,180, 270 y se realiza una votación. Si la imagen evaluada en las 4 rotaciones contiene un fichero, entonces decidimos que la imagen corresponde a una ficha; en caso contrario es otro tipo de imagen.

Una forma de ver el funcionamiento del criterio usado para determinar si una imagen es una ficha, es pensar en el modelo Yolov5 como un juez, bajo esta analogía las imágenes serían los participantes del juicio. Si el mismo juez determina que la imagen tiene una ficha viéndolo desde perspectivas distintas (las 4 rotaciones) se puede determinar que la evaluación del juez tiene mucho más certeza. Medimos esta certeza más adelante y mostramos los resultados en la tabla 3.11.

2.7.4 *Detector de rotación*

En cuanto se obtenga el área correspondiente a la ficha en sus cuatro ángulos, se evalúa el área con OCR (ver 2.2). Para identificar analíticamente la rotación de la imagen, se decidió entrenar un modelo de clasificación binario basado en árboles; es decir, un RandomForest. Para entrenar el modelo se usó la muestra de 2,000 mil imágenes anotadas con anterioridad con ayuda de la aplicación mostrada en la sección 2.14, filtrando aquellas que eran ficheros.

La variable objetivo del modelo es 1 si el texto corresponde a la lectura OCR de la imagen rotada correctamente, 0 en otro caso (ver figura 2.17).

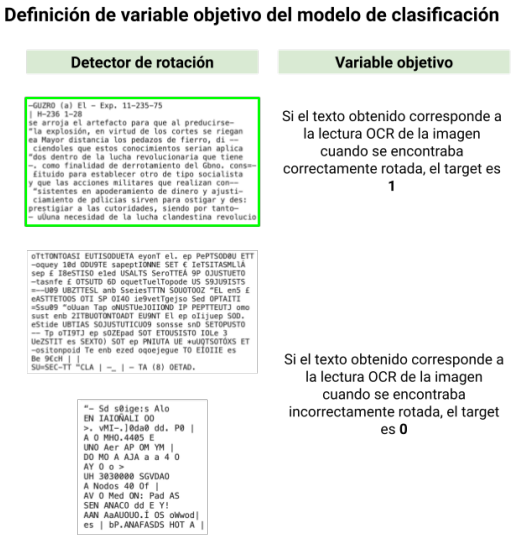


Figura 2.17: Definición de variable objetivo del modelo detector de rotación.

Lo anterior dio como resultado una muestra de 5,481 registros, desbalanceada con 1,366 registros de la clase 1 y 4,115 registros de la clase 0 (ver 2.18).

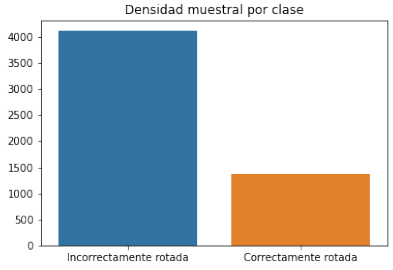


Figura 2.18: Densidad original en la muestra para modelo detector de rotación.

Se identificó que es un problema supervisado des-balanceado. Por lo tanto, se decidió muestrear la clase mayoritaria para obtener la misma cantidad de 1 como de 0 en la muestra y evitar un problema de sobre-entrenamiento.

Para explicar la relación del sobre-entrenamiento y él des-balanceo de clases, se puede pensar que existe una base donde el 99% de las imágenes están rotadas correctamente y solo 1% de ellas no lo están. En este sesgo de captura sería válido decir que todas las fichas siempre están rotadas correctamente, lo cual no necesariamente es cierto. Si estuviéramos seguros de esta predominancia se buscaría representarla en el conjunto de entrenamiento, no obstante, se busca que el modelo sea capaz de detectar la correcta rotación, aún sí la imagen a evaluar está rotada correctamente con probabilidad 0.5 (azar). De este modo, si se busca que el modelo sea capaz de detectar la rotación correcta sin importar la rotación con la que se obtienen los documentos se vuelve clave balancear y forzar al modelo a buscar la relación entre las co-variables y la variable respuesta.

Para usar el texto como co-variables del modelo se entrenó un word embedding basado en el conteo de palabras. En este caso, se limitó el vocabulario a 1,500 atributos y se evitaron palabras que se encuentran en más del 70% de la muestra de entrenamiento, se retiran como una forma de evitar que el modelo tome decisiones con atributos poco valiosos (ver código 2.9). De esta forma, se obtiene una representación vectorial del texto que podemos utilizar para construir un modelo de clasificación binario.

Código 2.9: Word Embedding OCR.

```
from sklearn.feature_extraction.text import CountVectorize
vectorizer = CountVectorizer(max_features=1500,
                min_df=5, max_df=0.7)
```

Se propone un modelo de clasificación binaria donde se etiqueta como 1 el texto correspondiente a la imagen cuando está rotada correctamente y 0 en caso contrario. El modelo detector de rotación queda representado como: $Pr(img, \theta)$, donde Pr es la propensión de que la imagen (img) esté rotada sobre el ángulo θ. Finalmente, para predecir el ángulo se toma el que tenga el máximo puntaje.

Los pasos son los siguientes:

1. Se definen los ángulos a evaluar, en este caso $\theta = \{0, 90, 180, 270\}$.
2. Para cada θ_i se evalúa la imagen img rotada θ_i ángulos en el pipeline de OCR.

3. El texto obtenido es mapeado a un vector. Un ejemplo de esta clase de transformación sería transformar "una casa azul" al vector (1,0,0,1,1,0,0,0,0...).
4. Se evalúa el texto en forma vectorial en un modelo clasificador.
5. Se obtiene el ángulo θ_i con mayor propensión Pr.

$$Angulo(img) = \underset{\theta=\{0,90,180,270\}}{\text{argmax}} \; Pr(img, \theta) \tag{2.1}$$

2.7.5 *Modelo identificador de la calidad del texto*

Para identificar la calidad del texto se tomará:

$$Ct(img) = \max_{\theta=\{0,90,180,270\}} Pr(img, \theta) \tag{2.2}$$

Se planeó poner el criterio de obtención de la calidad del texto en fórmula como una forma de facilitar la lectura, en palabras, el proceso de obtener la calidad del texto corresponde al de tomar la probabilidad del ángulo que contiene el valor máximo en el modelo detector de rotación.

3 RESULTADOS

3.1 CARACTERÍSTICAS DE LOS ARCHIVOS

Después de la descarga de archivos (ver sección 2.1) los 18,008 ficheros descargados se distribuyen en 9 resoluciones (ver tabla 3.1). De ese total, el 76% (13,826) de las imágenes son tomadas horizontalmente.

ALTURA	ANCHURA	CONTEO	PORCENTAJE(%)
2304	1728	792	4.40
2592	1944	6	0.03
1728	2304	1876	10.42
3264	2448	722	4.01
1944	2592	2637	14.64
2448	3264	413	2.29
4608	3456	2662	14.78
2432	4320	188	1.04
3456	4608	8712	48.38

Tabla 3.1: Resolución de los Ficheros.

La profundidad de las imágenes se distribuye en 3,699 con solo un nivel de profundidad, 12,797 imágenes con solo dos niveles de profundidad y 1,512 con tres niveles de profundidad. La totalidad de los archivos y carpetas representan 54.8 GB de información.

3.2 LOS MEJORES HIPER-PARÁMETROS ENCONTRADOS

En cuanto a los mejores hiper-parámetros encontrados en la sección 2.3, pueden ser consultados en la tabla 3.2.

altura	ancho	sigma	pixels	border
1728	2304	0.33	1200	15
1944	2592	0.33	1300	15
2304	1728	0.33	1300	20
2432	4320	0.01	900	20
2448	3264	0.33	1200	20
3264	2448	0.33	1300	20
3456	4608	0.01	1300	25
4608	3456	0.33	1200	15

Tabla 3.2: Mejores hiper-parámetros para las resoluciones de los Ficheros.

Las resoluciones que mejor ajustaron a las distintas resoluciones fueron: 1200-1300 (la configuración que obtuvo 900 solo está en el 1% de los ficheros), por otro lado, el sigma que mejor ajusta es el valor 0.33 y finalmente el tamaño del borde varía dependiendo de la resolución de la imagen. Todo este paso tardó 5 horas, 4 minutos y 1 segundo.

3.3 RESULTADOS DEL PRIMER PROCESAMIENTO

El primer procesamiento consistió en evaluar la totalidad de los ficheros (18,008 imágenes) con el flujo desarrollado en la sección 2.2, después de obtener los mejores hiper-parámetros para nuestro conjunto de fichas en la sección 3.2. Los resultados del primer procesamiento (ver sección 2.4) fueron los siguientes:

Del total de las 18,008 fichas con el primer procesamiento se logró detectar lo siguiente:

Patrón de expediente detectado	Total
1	9,011
0	8,997

Tabla 3.3: Patrón de expediente detectados en primer procesamiento (1 sí, 0 no).

Analizando al comportamiento que tuvo el algoritmo por resolución, hay evidencia de que cuando la imagen es tomada verticalmente el algoritmo pierde efectividad (ver tabla 3.4). La peor efectividad en imágenes con resolución horizontal (0.58) es 1.52 veces más grande que la mejor resolución en imágenes con resolución vertical (0.23). En estos casos se usa el método de segmentación 1 (Automatic page segmentation con OSD, método usado con imágenes que no están rotadas correctamente) el cual parece no mejorar la lectura, por lo cual buscar alternativas para rotar la imagen correctamente se vuelve un factor a considerar.

ALTURA	ANCHO	HORIZONTAL	F. TOTALES	F. PATRÓN	EFECTIVIDAD
3456	4608	1	8712	5021	0.58
2448	3264	1	413	244	0.59
2432	4320	1	188	144	0.77
1944	2592	1	2637	1854	0.70
1728	2304	1	1876	1177	0.63
4608	3456	0	2662	333	0.13
3264	2448	0	722	53	0.07
2592	1944	0	6	0	0.00
2304	1728	0	792	185	0.23

Tabla 3.4: Patrón de expediente detectados en primer procesamiento dividido por resolución.

De las 9,011 fichas detectadas, la mayoría se concentran en el grupo de 3-4 dígitos (ver tabla 3.5).

Tipo de patrón	Fichas detectadas
Expedientes con 5 digitos	1,584
Expedientes con 4 digitos	4,793
Expedientes con 3 digitos	9,011

Tabla 3.5: Fichas detectadas por tipo de patrón.

Todo el primer procesamiento tardó 6 horas, 45 minutos y 44 segundos.

3.4 RESULTADOS DEL SEGUNDO PROCESAMIENTO

Después de obtener los resultados del primer procesamiento (ver sección 3.3), algunos de los hallazgos fueron que era una desventaja no tener la información de la confianza de las palabras y su posición en la imagen. Además de que había que buscar un nuevo enfoque en el caso de las imágenes en vertical.

Del total de las 18,008 fichas con el segundo procesamiento se pudo detectar lo siguiente:

Patrón de expediente detectado	Total
1	9,238
0	8,770

Tabla 3.6: Patrón de expediente detectados en segundo procesamiento (1 sí, 0 no).

Poniendo atención al comportamiento que tuvo el algoritmo por resolución, se sigue observando que el hecho de que la imagen haya sido tomada de forma vertical afecta significativamente la efectividad del algoritmo (ver tabla 3.7).

Girar la ficha 90 grados cuando se tiene una imagen vertical, mejoró solo en 227 nuevas fichas con expedientes detectados. Se obtuvieron 174 fichas más en la resolución 4608x3456, 39 en 2304x1728, 9 en 3264x2448, 3 en 3456x4608, 1 en 1944x2592 y 1 en 1728x2304 (ver tabla 3.7).

ALTURA	ANCHO	HORIZONTAL	F. TOTALES	F. PATRÓN	EFECTIVIDAD
3456	4608	1	8712	5024	0.58
2448	3264	1	413	244	0.59
2432	4320	1	188	144	0.77
1944	2592	1	2637	1855	0.70
1728	2304	1	1876	1178	0.63
4608	3456	0	2662	507	0.19
3264	2448	0	722	62	0.09
2592	1944	0	6	0	0.00
2304	1728	0	792	224	0.28

Tabla 3.7: Patrón de expediente detectados en segundo procesamiento dividido por resolución.

De las 9,238 fichas detectadas, la mayoría se concentran en el grupo de 3-4 dígitos (ver tabla 3.9).

Tipo de patrón	Fichas detectadas
Expedientes con 5 digitos	1607
Expedientes con 4 digitos	4873
Expedientes con 3 digitos	9238

Tabla 3.8: Fichas detectadas por tipo de patrón en segundo procesamiento.

Todo el segundo procesamiento tardó 7 horas, 46 minutos y 56 segundos.

3.4.1 *Las fuentes y su distribución en la salida del segundo procesamiento*

En el segundo procesamiento debido a que se guardaron los metadatos del OCR, se pudo programar un proceso para detectar el tamaño de la fuente de la ficha al momento de la lectura. El flujo para hacerlo se encuentra en la figura 2.11.

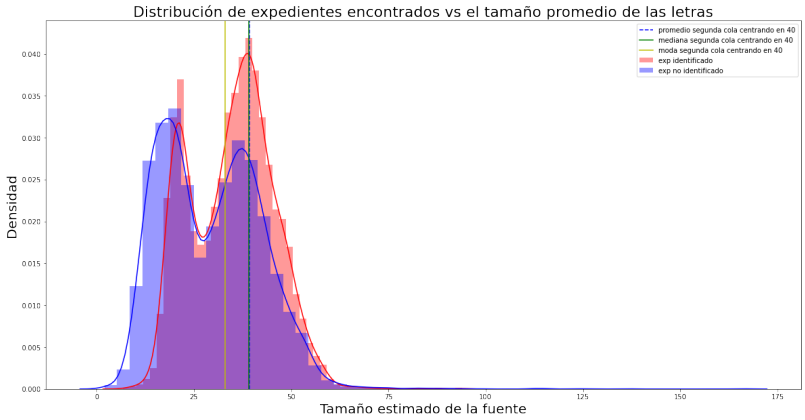


Figura 3.1: Distribución del tamaño de la fuente en fichas con patrones de expedientes detectados.La distribución roja corresponde a

Uno de los hallazgos, cuando el tamaño del texto fue detectado con el método propuesto, es que existe una relación entre el tamaño de la fuente y la calidad de la lectura. Información consistente con foros sobre reconocimiento óptico de texto (Tesseract Community, 2018).

Además, al observar los extremos de la distribución de las fichas (ver 3.1), se puede notar que suele representar fotografías de fichas vacías, con enmendaduras o documentos que no son fichas.

3.4.1.1 *Resultado de la detección de más de un expediente por ficha*

En cuanto a los expedientes que se pudieron detectar en una ficha, se encontró que:

Entre los casos peculiares se encuentran aquellos casos donde encontramos 8 expedientes en una sola imagen (ver figuras 3.4 y 3.5).

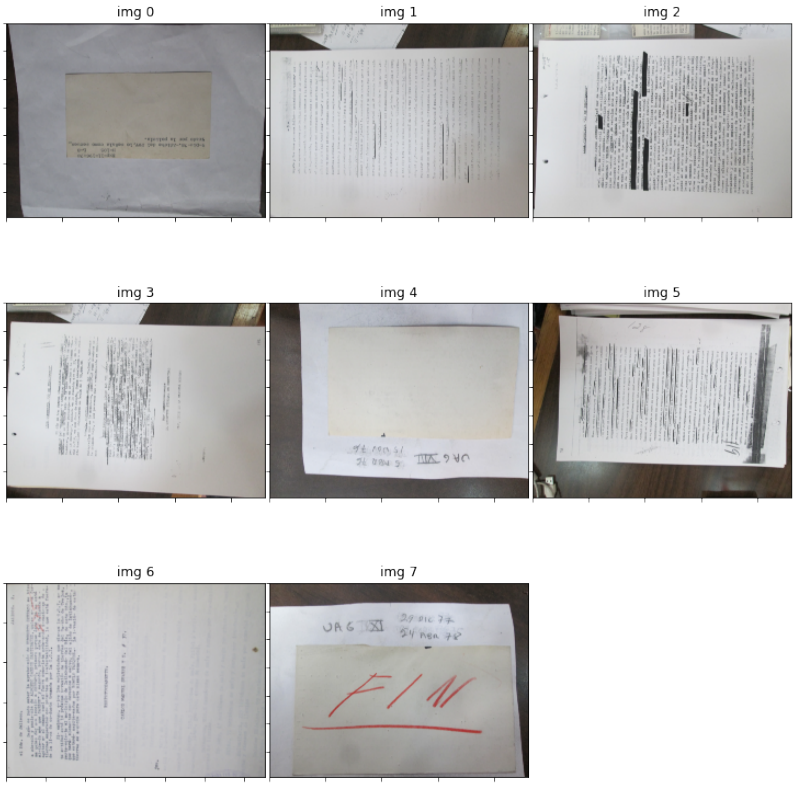


Figura 3.2: Muestra de imágenes clasificadas con fuentes menores a 10. Algunas de las imágenes de esta muestra no son consideradas un fichero de la DFS, a pesar de que, estas imágenes fueron clasificadas como ficheros en nuestra fuente de información. El contenido de los documentos no es relevante, pero si la apariencia.

Número de expedientes por ficha	Fichas
8	2
7	10
6	22
5	47
4	166
3	612
2	2552
1	5827

Tabla 3.9: Número de expedientes detectados por Ficha.

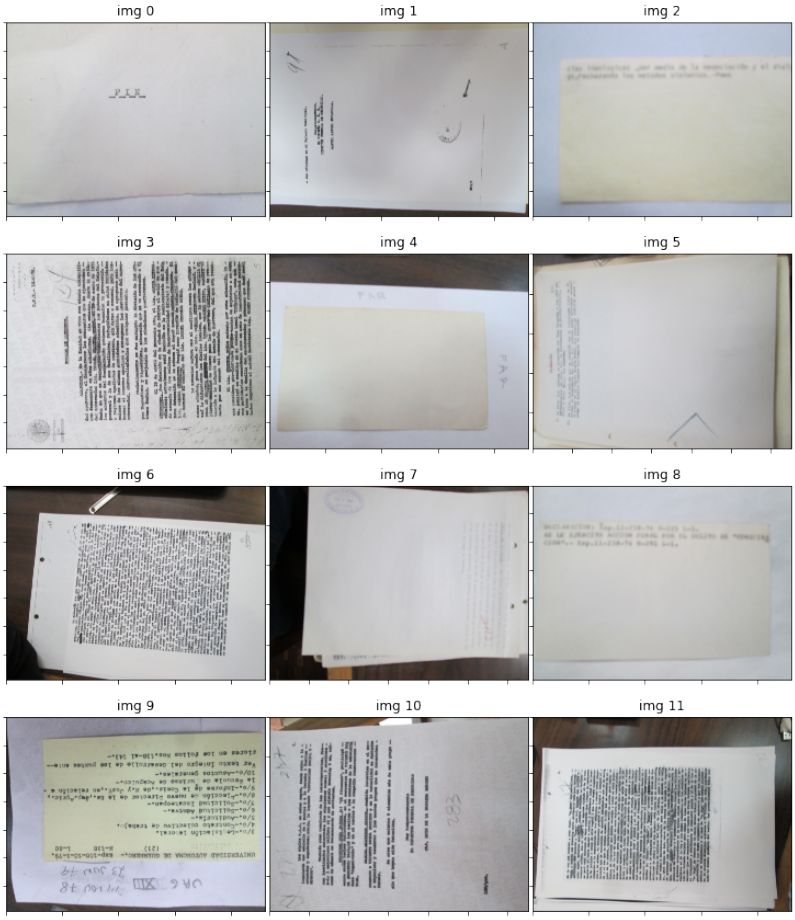


Figura 3.3: Muestra de imágenes clasificadas con fuentes mayores a 100. Visualmente ninguno de los documentos de esta imagen parecen tener fuentes grandes. Aquí podemos observar que el proceso de lectura OCR no es un proceso perfecto y que suele estar sujeto a errores de lectura.

Figura 3.4: Imagen ejemplo 1 donde fueron detectados 8 expedientes. En este caso los ficheros se encuentran en bolsas plásticas y acomodados en forma de cuadrícula.

Figura 3.5: Imagen ejemplo 2 donde fueron detectados 8 expedientes. En este caso los ficheros se encuentran en bolsas plásticas y acomodados en forma de cuadrícula.

3.5 RESULTADOS DE LA METODOLOGÍA PROPUESTA BASADA EN TÉCNICAS DE APRENDIZAJE DE MÁQUINA PARA LA EXTRACCIÓN DEL TEXTO EN LOS FICHEROS DE LA DFS

3.5.1 *Etiquetado de imágenes*

En total se anotaron 2,000 imágenes. Se tomaron las imágenes con las que no se había podido tomar un expediente mediante el RegEx.

Sus características fueron las siguientes:

Es un fichero	Es un expediente	total
0	0	618
	1	1
1	0	640
	1	741

Tabla 3.10: Número de imágenes en cada grupo etiquetado. El valor 1 en el caso del campo "Es un fichero" corresponde a sí, 0 en caso contrario. En el caso de "Es un expediente" el caso es el mismo, 1 es sí, 0 corresponde a no.

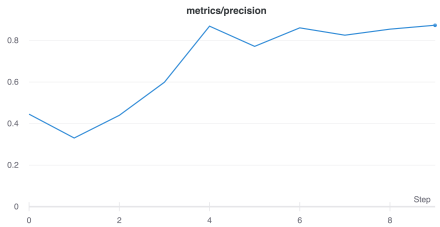
Si bien, había más fichas en el conjunto, existía la proporción suficiente para esperar una buena generalización sin necesidad de hacer oversampling.

3.5.2 *Eficacia del modelo detector de fichas*

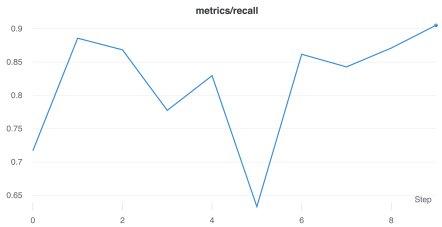
Como se discutió en la sección 2.7.2, se usó la plataforma Roboflow para realizar el proceso de "data augmentation". Posteriormente, se utilizó el framework de entrenamiento proporcionado por los creadores de la arquitectura Yolov5 para generar un modelo especializado en detectar fichas de la DFS dentro de imágenes.

Después de 10 épocas (existe un probable rango de mejora en el modelo, sin embargo, hay que considerar dejar el modelo en sus 10 épocas después del mAP encontrado) el modelo aprendió a generalizar e identificar con un mAP entre los quantiles 0.5 y 0.95 arriba del 95% (ver figura 3.6) en el conjunto de prueba. Además, la precisión se encontró en 0.87 y el recall 0.9 las cuales son buenas métricas tomando en cuenta que son métricas enfocadas en identificar objetos y su contorno.

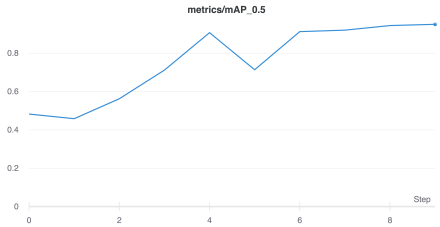
Adicional de las métricas obtenidas durante el entrenamiento del modelo (ver anexo A.5a), se añadieron las métricas obtenidas de identificar si la imagen contiene una ficha de la DFS (la tabla 3.11 se cons-



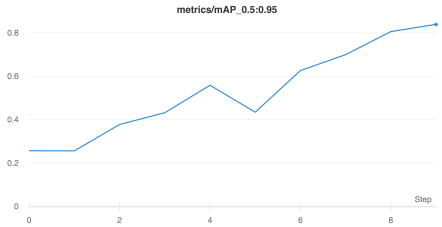
(a) Métrica precisión del conjunto de validación



(b) Métrica recall del conjunto de validación



(c) Métrica mAP arriba del percentil 5



(d) Métrica mAP del percentile 0.5 - 0.95

Figura 3.6: Métricas de desempeño del modelo yolov5. Las dos gráficas relacionadas al mAP nos dicen que el modelo es capaz de detectar el área de la ficha hasta en un 0.83 para las fichas dentro del percentile 0.5 y 0.95 (mAP_0.05:0.95) y del 0.95 para el percentile arriba de la mediana (mAP_0.5). Por otro lado, la precisión y el recall en la tarea de clasificación nos interesa porque por un lado la presición del 0.87 nos dice que con el corte de probabilidad 0.5 acertaremos el 87% de las veces en encontrar las fichas y con ese corte esperamos obtener el 90% de las fichas en la muestra.

truye de las fórmulas 3.1 y 3.2).

La primera de estas medidas fue calcular una métrica de tipo votación. La métrica a evaluar es si la imagen contiene una ficha (dato que se registró con el anotador web a un conjunto de imágenes ver 2.7.1). En este caso la imagen evaluada en diferentes rotaciones hace el papel de votación. La medida está definida de la siguiente forma:

$$SF(img) = \sum_{\theta=\{0,90,180,270\}} FF(img, \theta) \tag{3.1}$$

$$\text{FF(img, } \theta) = \begin{cases} 1, & \text{Si } pf(img, \theta) > 0.4 \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

Donde SF nos dice en cuantas rotaciones la imagen fue detectada como fichero de la DFS. Por otra parte, la función FF identifica si una imagen en la rotación θ es una ficha según su evaluación.

Donde pf es la propensión de la imagen img rotada en ángulo θ de contener una ficha.

Como resultado, en las 2000 imágenes anotadas obtuvimos que en la muestra etiquetada hay un porcentaje de ficheros del: 69.05% (resultado de dividir las fichas encontradas en la tabla 3.10 durante el etiquetado), le llamaremos precisión teórica, si se asume que todas las imágenes son ficheros se acertará el 69.05% de las veces, en contraste con el 97.98 % del mejor corte en la función FF, si se etiquetan como ficheros solo con el valor 4.0 de la función FF se acertará el 98% de las veces con una perdida de 1.3% fichas no detectadas. Lo cual permite demostrar en el conjunto de prueba la efectividad de usar el modelo para identificar si una imagen pertenece a una ficha de la DFS en contraste con no usarlo (precisión teórica).

	Eventos		Precisión del grupo	Recall del grupo
Fichero SF	Es fichero	No es fichero		
0.0	4	402	0.009852	0.002896
1.0	3	94	0.030928	0.002172
2.0	6	59	0.092308	0.004345
3.0	6	36	0.142857	0.004345
4.0	1362	28	0.979856	0.986242

Tabla 3.11: Precisión y Recall del detector de ficheros. El grupo con el valor de SF mas grande (4.0) tiene la mayor y precisión en nuestro conjunto de prueba. Al usar ese corte en el conjunto de prueba se escapan 19 ficheros (el 1.3% de la muestra).

3.5.3 *Resultados del modelo detector de rotación*

Como resultado del entrenamiento propuesto en la sección 2.7.4, las métricas de desempeño del modelo detector de correcta rotación son los siguientes:

La implementación del modelo mejora 4 veces la precisión de una decisión aleatoria, pasa de tener 0.2479 a 0.9924 de efectividad. Comparando con la precisión de una decisión aleatoria, se tiene evidencia para decir que el modelo propuesto mejora la tarea de encontrar la rotación adecuada y que el error es aceptable.

3.5.4 *Resultados del OCR*

Dentro de la carpeta analizada se encontró que 15,255 eran detectadas como imágenes con Ficheros de la DFS (ejecutando el modelo detector de fichas visto en la sección 2.7.2 en las 18,008 imágenes). En cuanto a la calidad del texto, los números pueden ser consultados en la tabla 3.13.

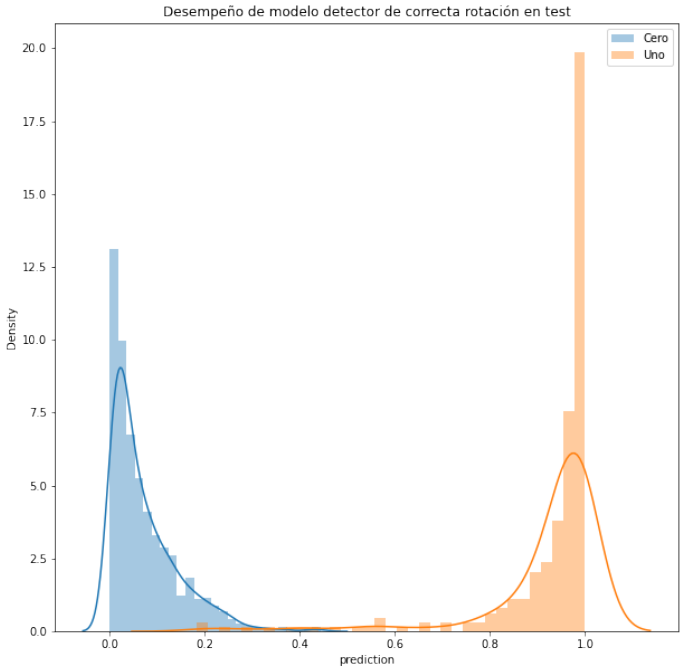


Figura 3.7: Desempeño del modelo detector de correcta rotación en él conjunto de prueba. El histograma azul corresponde a las probabilidades asignadas a imágenes que no son ficheros, en contraste con el histograma naranja, el cual corresponde a probabilidades asignadas a imágenes de ficheros. En esta gráfica se espera que un buen modelo de clasificación asigne probabilidad baja (cercana 0) a imágenes que no son ficheros y probabilidades cercanas al 1.0 a las imágenes que son ficheros.

Otro aspecto de importancia es la rotación del texto. En este paso la rotación más común fue la de 0 grados, sin embargo, 2,898 de las imágenes (aproximadamente el 20% de la muestra) se encuentran en otra rotación (ver tabla 3.14).

En cuanto el dato con mayor relevancia para este trabajo (la clave de expediente) pudimos encontrar los expedientes de cerca del 70% de las fichas detectadas, una métrica considerada aceptable tomando en cuenta que sabemos que no todos los ficheros contienen expedientes (confirmado durante el etiquetado de las imágenes en la sección 2.7.1).

Los 10,678 expedientes (ver Tabla 3.15) representan una mejora del 16%, el cociente de dividir 10,678 ficheros con clave de expediente obtenidos con la metodología propuesta y 9,238 obtenidos con el flujo genérico desarrollado siguiendo las recomendaciones de Tesseract-

Predicción Actual	0	1	Todos
0	825	0	825
1	9	263	272
Todos	834	263	1097

(a) Matriz de confusión. Se espera que un buen modelo clasifique la respuesta 0 como 0 y la respuesta 1 como 1, lo que se manifestaría con mayor densidad de observaciones en la identidad de la tabla. En este caso se tienen 9 errores de predicción que corresponde con imágenes con la rotación correcta (1) etiquetados como no correctamente rotados (0).

Métrica	
ROC	0.999361
Recall	0.966912
Precisión	0.992453
Precisión observada(real):	0.247949
Mejora en precisión:	4.002650

(b) Métricas de desempeño del modelo. La ROC es una medida que nos dice que tan bien clasifica nuestro modelo, se espera que el valor sea cercano al 1 (clasificación perfecta), toma en cuenta tanto verdaderos positivos y minimización de falsos positivos. Tanto la ROC, el Recall y la Precisión tienen valores cercanos a 1 lo cual es un buen indicador, comparando con la precisión teórica de este problema (0.25, la probabilidad de acertar aleatoriamente en la rotación), el modelo mejora 4 veces la precisión observada, lo cual se puede traducir en que usar el modelo es 4 veces mejor que no usarlo (hablando de precisión).

Tabla 3.12: Métricas de desempeño del modelo detector de rotación

etiqueta_calidad_texto	No. de Fichas encontradas
buena_calidad	14,387
no_tan_buena_calidad	868

Tabla 3.13: Conteo de la calidad detectada en los ficheros. La etiqueta buena_calidad se le asigna a las imágenes que durante la evaluación del modelo detector de la correcta rotación tuvieron una probabilidad igual o mayor a 0.7.

angle_predicted	No. de Fichas encontradas
0	12357
90	1885
-90	951
180	62

Tabla 3.14: Las rotaciones de los ficheros encontrados

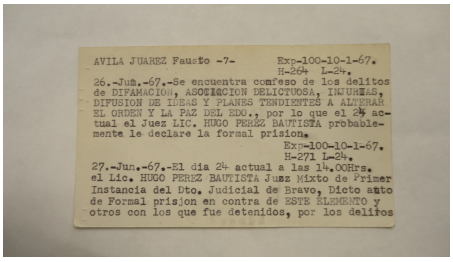
OCR que pueden ser consultados en las secciones 2.4 y 3.4. Lo anterior válida la hipótesis de que una serie de pre-procesamientos específicos a este conjunto de datos, mejorarán la calidad de la extracción

de la clave de expediente.

Longitud Expediente	No. de Fichas detectadas
Sin expediente detectado	4,577
3	4,772
4	4,107
5	1,799

Tabla 3.15: Expedientes y su longitud encontrada usando la metodología propuesta. El número de fichas con clave de expediente detectado son 10,687.

Estos datos son útiles para historiadores y público en general porque permiten buscar palabras claves mediante los nombres, organizaciones y expedientes obtenidos mediante la extracción de texto con OCR del pipeline que sugerimos (ver figuras 3.8, 3.9, 3.10), proceso que con anterioridad se realizaba de forma manual. También es posible construir redes de información entre los expedientes que podrían permitir el conocer más sobre el paradero de las víctimas de desaparición forzada y el papel de las instituciones en ello.


AVILA JUAREZ Fausto -7- Exp-100-10-1-67.
H-264 L-24.
26.-Jun.-67.-Se encuentra comfeso de los delitos de DIFAMACION, ASOCIACION DELICTUOSA, INJURIAS, DIFUSION DE IDEAS Y PLANES TENDIENTES A ALTERAR EL ORDEN Y LA PAZ DEL EDO., por lo que el 25 actual el Juez LIC. HUGO PEREZ BAUTISTA probablemente le declare la formal prision.
Exp-100-10-1-67.
H-271 L-24.
27.-Jun.-67.-El dia 24 actual a las 14.00Hrs. el Lic. HUGO PEREZ BAUTISTA Juez Mixto de Primer Instancia del Dto. Judicial de Bravo, Dicto auto de Formal prision en contra de ESTE ELEMENTO y otros con los que fue detenidos, por los delitos

(a) Fichero de la DFS

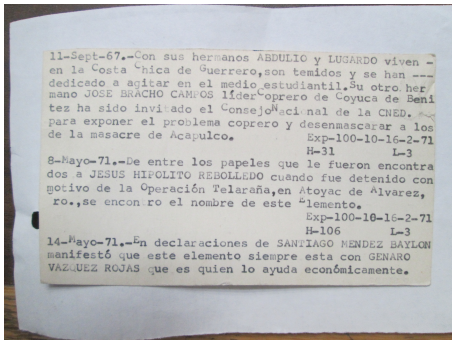
AVILA JUAREZ Tausto =-7- .Exp-100-10-1=07, — o H.2604- Lo2k,
26.-Juñ,-67,-Se encuentra comfeso de los delitos
- de DIFANMACIO:, ASOTIECION DELICTUOS£4.—IiUnBAS,
— DIFUSION DE IDEAS Y PLANES TENDIENTES Á ALTERAN
— EL ORDEN Y LA PAZ DEL EDO. , por lo que el 25 ac=
— tual el Juez LIC, HUGO PEREZ BAUTISTA probable=
— mente le declare la formal prisiono
| | EXp=100-10=1=67.
H-271 I=2lt,
27.,=Juno=67.-El dia 24 actual a las I1l,ooHrs, —
el Lic, HUCO PEREZ BAUTIST£ Juzz Mixto de ?rimer
Instancia del Dto, Judicial de Bravo, Dicto aúto
— de Formal prisjon en contra de ESTE ELEMENTO y
“otros con los que fue detenidos, por los delitos

(b) Texto obtenido con la metodología

Atributo	Valor
Entidades en mayusculas (RegEx)	[AVILA JUAREZ , H, DIFANMACIO, ASOTIECION DELICTUOS,
DIFUSION DE IDEAS Y PLANES TENDIENTES ,	ALTERAN, EL ORDEN Y LA PAZ DEL EDO, LIC, HUGO PEREZ BAUTISTA ,
I, HUCO PEREZ BAUTIST, ESTE ELEMENTO]	
Clave de expediente (RegEx)	100-10-1=07

(c) Atributos extraídos del texto mediante la herramienta RegEx

Figura 3.8: Fichero ejemplo no.1 evaluado en la metodología propuesta. En este caso, la lectura del fichero contiene entidades importantes para su explotación como lo son nombres de personas, nombres de asociaciones y clave de expediente. Además, el formato es el esperado, por consiguiente, su extracción ha sido capaz de extraerlas.



(a) Fichero de la DFS

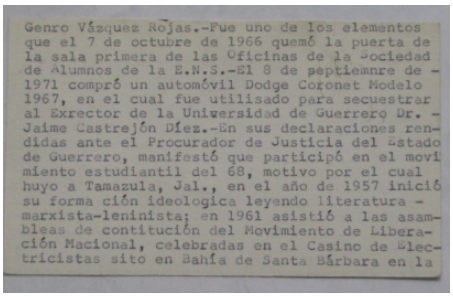
ll-Sept-67.-Con sus hermanos ABDULIO y LUGARDO viven -
en la Costa “nica de Guerrero, son temidos y se han ==
dedicado a agitar en el medio estudiantil.Su otro her”
mano JOSE BRACHO CAMPOS líder “oprero de Coyuca de Beni
tez ha sido invitado el ConsejoNaci: nal de la CNEDo. «
para exponer el problema coprero y desenmascarar a los
de la masacre de Acapulco, — Exp-100-10-16-2-71
| | H=31 L.3
E-layo=71.-De entre los papeles que le fueron encontra
dos a JESUS HIPOLITO REBOLLEDO cuando fue detenido con
uotivo de la Operación Telaraña,en Atoyac de Alvarez,
— ro.,8e encon.ro el nombre de este “lemento. | |
| u Exp-100-10-16=2.-71:
| | H-106 L3
| l4layo=712 En declaraciones de SANTIAGO MENDEZ BAYLON
manifestó que este elemento siempre esta con GENARO —
VACLUEZ ROJAS cue es quien lo ayuda económicamenteo

(b) Texto obtenido con la metodología

Atributo	Valor
Entidades en mayusculas (RegEx)	[ABDULIO , LUGARDO , JOSE BRACHO CAMPOS , L, JESUS HIPOLITO REBOLLEDO, SANTIAGO MENDEZ BAYLON, GENARO, VACLUEZ ROJAS]
Clave de expediente (RegEx)	100-10-16-2-71

(c) Atributos extraídos del texto mediante la herramienta RegEx

Figura 3.9: Fichero ejemplo no.2 evaluado en la metodología propuesta. En este caso, la lectura del fichero contiene entidades importantes para su explotación como lo son nombres de personas y clave de expediente. Además, el formato es el esperado, por consiguiente, su extracción ha sido capaz de extraerlas.


Genro Vázquez Rojas.-Fue uno de los elementos que el 7 de octubre de 1966 quemó la puerta de la sala primera de las Oficinas de la Sociedad de Alumnos de la E.N.S.-El 8 de septiemnre de - 1971 compró un automóvil Dodge Coronet Modelo 1967, en el cual fue utilisado para secuestrar al Exrector de la Universidad de Guerrero Dr. - Jaime Castrejón Díez.-En sus declaraciones rendidas ante el Procurador de Justicia del Estado de Guerrero, manifestó que participó en el movimiento estudiantil del 68, motivo por el cual huyo a Tamazula, Jal., en el año de 1957 inició su forma ción ideologica leyendo literatura - marxista-leninista; en 1961 asistió a las asambleas de contitución del Movimiento de Liberación Macional, celebradas en el Casino de Electricistas sito en Bahía de Santa Bárbara en la

(a) Fichero de la DFS

— Genro Vázquez Rojas.-Fue uno de los elementos
— que el 7 de octubre de 1966 quemó la puerta de
; la sala primera de las Oficinas de la “ociedad .
de álumnos de la E.N.5.-El 8 de septiemnre de -
- 1971 compró un automóvil Dodge Coronet fodelo
1967, en el cual fue utilisado para secuestrar -
— al Exrector de la Unizersidad de Guerrero Dr. —
" Jaime Castrejón Díez.-En sus declaraciones ren-
; dídas ante el Procurador de Justicia del ¿stado
— de Guerrero, manifestó que participó en el movi
— miento estudiantil del 68, motivo por el cual
- huyo a Tamazula, Jal., en el año de 1957 inició
. su forma ción ideologica leyendo literatura -
- Marxista-leninista; en 19261 asistió a las asam-
: bleas de contitución del iovimiento de Libera-
— ción Macional, celebradas en el Casino de Llec-
— tricistas sito en Bahía de Santa Bárbara en la”
e —

(b) Texto obtenido con la metodología

Atributo	Valor
Entidades en mayusculas (RegEx)	[E]
Clave de expediente (RegEx)	No detectado

(c) Atributos extraídos del texto mediante la herramienta RegEx

Figura 3.10: Fichero ejemplo no.3 evaluado en la metodología propuesta. La lectura de la ficha incluye entidades como nombres de personas y lugares, a pesar de ello, el proceso RegEx no es capaz de extraerlas, ya que no están en mayúsculas, en la sección 3.5.4 mencionamos este hallazgo como un problema a resolver en trabajos futuros.

CONCLUSIONES

El objetivo principal del presente trabajo, fue el de proponer una metodología para la explotación de los ficheros de la DFS. Esta metodología fue presentada durante el capitulo 2 y aplicada mostrando los resultados en el capitulo 3. De esta manera, se demuestran satisfactoriamente las hipótesis, en primer lugar como resultado de la presente tesis, se implementó una metodología utilizando aprendizaje de máquina para la extracción del texto, con énfasis en las claves de expediente, de las fotografías de los ficheros de la DFS. Adicionalmente, mostramos que una serie de pre-procesamientos específicos a este conjunto de datos, como discriminador de fichas, detector del área de la ficha y detector la rotación, mejorarán la calidad de la extracción de la clave de expediente al mejorar la extracción de 60% a 70%.

Por la forma en la que fueron generadas las fichas de la DFS requieren un tratamiento especial para la extracción de texto. El resultado de este trabajo es, por tanto, la metodología conformada por: Discriminador de Fichas, Detector del área de la Ficha, Lectura del OCR, Detector de rotación, Detector de la calidad del OCR y extracción de entidades usando la herramienta RegEx. Todas estas etapas resuelven problemas distintos encontrados en los ficheros de la DFS.

En esta tesis se demuestra que es posible solucionar el problema de la identificación del área de texto utilizando modelos tradicionalmente usados para identificación de objetos como lo es yolov5. Además, como un resultado indirecto, obtuvimos que aparte de identificar el área correspondiente a las fichas, es posible emplear la probabilidad del modelo para inferir si la imagen corresponde a la de un fichero de la DFS.

Asimismo, obtuvimos evidencia en las secciones 3.3 y 3.4 de que la rotación afectaba significativamente la calidad de la lectura, por lo cual propusimos un modelo analítico que nos permitiera identificar la mejor rotación para la imagen y de forma indirecta una métrica de calidad de lectura.

Como conclusión directa de la colaboración con historiadores, familiares de desaparecidos e instituciones como la AGN y CNB para la realización de este proyecto, resalta la urgente necesidad y responsabilidad de llevar los avances tecnológicos, tanto de aprendizaje de máquina como de otras herramientas, a la búsqueda de justicia. De

manera inversa, se reconoce el papel del contexto social e histórico como requisito para una correcta implementación de estas herramientas, en todas las fases de la generación de un desarrollo tecnológico. A futuro, es indispensable continuar las colaboraciones entre autoridades, víctimas, investigadores del fenómeno y expertos tecnológicos en la resolución de problemas sociales, los cuales conciernen y afectan a todos.

Durante la realización del trabajo fue necesario profundizar en el contexto histórico de los hechos atestiguados por los documentos que se tuvo el deber de analizar. Este proceso, que constó de reuniones con historiadores, familiares de víctimas, documentos y libros en los que apoyamos la base de nuestro marco teórico, nos permitió sensibilizarnos con el problema. A criterio propio, se puede concluir la necesidad de una aproximación distinta de parte de perfiles de formación afines a las Ciencias de la Computación. Para el autor de esta tesis el lema institucional tomó un significado distinto y los valores universitarios parecieran ser más grandes de lo que se pensaba al inicio de este trabajo, tan es así que no se quiso finalizar el trabajo sin dejar huella del impacto personal que se tuvo al abrir los ojos a una realidad oculta, que por azares del destino se tuvo el privilegio de no vivir. Sin embargo, la conclusión que se desea dejar en los lectores de este trabajo es que los desaparecidos están ahí, y sus víctimas son huérfanos, hermanos, padres, vecinos, profesores, etc. No es un fenómeno minúsculo, los estragos de la guerra sucia los seguimos viviendo hoy y explica, en gran manera, la descomposición de nuestro tejido social en el México contemporáneo. No es un problema ajeno a sus familiares, cuando se pierde a alguien, de esta forma las víctimas son todos y quienes han perdido son todos.

PERSPECTIVAS

Quedaría plantear cuáles son las líneas de investigación futuras. Como continuación natural de este trabajo, sería el estudio y profundización en cada uno de los subproblemas encontrados. Por ejemplo, en la parte de extracción de entidades se pueden encontrar modelos más complejos como los basados en BERT (Devlin y Toutanova, 2018) que, a diferencia de la herramienta RegEx, destacan en flexibilidad y adaptabilidad.

También resultaría interesante usar modelos como el propuesto con yolov5 para encontrar áreas especificas de los ficheros que les interesen a quienes trabajan con ellos como lo es el expediente y las tachaduras. Tanto para los modelos de procesamiento de lenguaje natural y los de detección de objetos, se vuelve útil tener datos anotados. Un esfuerzo que durante el desarrollo del trabajo demuestra ser de utilidad para expandir el abanico de herramientas aplicables a los ficheros.

A ANEXO

Dentro de este anexo depositaremos piezas de código o figuras que por su complejidad y extensión no era práctico incluir dentro de la estructura normal del trabajo.

A.1 WEB SCRAPING

Los códigos A.1 y A.2 fueron desarrollados para la descarga de los ficheros de la DFS, encontrados dentro del proyecto archivosdelarepresion.org. El código se asegura de mantener la misma estructura de carpetas y nombres que el origen para que de ser necesario los resultados de su explotación puedan ser empatados usando la url como llave.

Código A.1: Función recursiva para descargar los Ficheros de la DFS desde la página archivosdelarepresion.org (parte 1).

```
import bs4
import requests
import os
import urllib.request

file_counter = 0 #Variable used to debug
def download_directory(url, output):
    """
    This function does a recursive search on the url parameter.
        Depending on the type of file found, the algorithm makes
        two decisions:
    * Images: they are stored inside the output route
    * Directories: the function is executed again this time using
         the path of the directory found as a parameter
    """
    # Force python to identify the variable as global (previusl
        defined)
    global file_counter

    #Gets url content
    r = requests.get(url)

    #Check if the request was not made correctly
    if not(r.status_code == requests.codes.ok):
        return
    #Parse the plain text to html to be processed
    data = bs4.BeautifulSoup(r.text, "html.parser")

    #If the page has less than 3 files, the folder has no useful
        information
    if(len(data.find_all("tr")) <3):
        return
```

Código A.2: Función recursiva para descargar los Ficheros de la DFS desde la página archivosdelarepresion.org (parte 2).

```
    #For each file ignoring footer and header
    for tr in data.find_all("tr")[3:-1]:
        #The file name and path are contained with an 'a' tag
        a = bs4.BeautifulSoup(str(tr), "html.parser").find("a")
        #url and name are useful
        href = str(a["href"])
        name = str(a.text)

        #Defining the output and input path it works for images
            or folders
        output_path = output+"/"+name
        input_path = url+"/"+href

        #Sometimes getting info might throw an error is useful
            for debugging
        try:
            #Way to identify a folder
            if("/" in name):
                #Way to create the folder if it is not created
                    yet
                if(os.path.exists(output_path) == False):
                    os.mkdir(output_path)
                #Apply flow again
                download_directory(input_path, output_path)
            else:
                #Only if the image is not already downloaded
                if(os.path.exists(output_path) == False):
                    r2 = requests.get(input_path)
                    open(output_path, 'wb').write(r2.content)
                #Count the image in global variable
                file_counter+=1
                print(file_counter, end="\r")
        except Exception as e:
            #Log error and file
            print(e)
            print(output_path, input_path)
            pass;
home = "https://archivo.archivosdelarepresion.org/Ficheros/"
output_path = "/media/juan/DATA/archivos_represion/Ficheros/"
#Execute all process
download_directory(home, output_path)
```

A.2 PRE-PROCESAMIENTO DE LAS IMÁGENES

A.2.1 *Primera versión de la clase PipelineImg*

La primera versión de la clase contiene una implementación resultado de la interpretación del flujo sugerido por los autores de tesseract para pre-procesar imágenes(ver código A.3).

Código A.3: Clase que implementa el pipeline sugerido dentro de la sección 2.4 (solo primeras lineas de la clase)

```
import cv2
import imutils
import numpy as np
from matplotlib import pyplot as plt

class PipelineImg:
    """
    This class was developed with the aim of storing the
        necessary functions to implement the pre-processing flow
        in an image in order to prepare it for OCR reading.
    """
    @staticmethod
    def do_full_pipeline(img, height=900, auto_canny_sigma=0.33,
        bordersize=15,*,debug=False, figsize=(10, 10)):
        """
        This function applies the preprocessing flow used for DFS
             files
        @param: img image that will be pre-processed (required)
        @param: height the final height used to resize the image
            (deault 900)
        @param: auto_canny_sigma sigma param used to border
            detection (default 0.33)
        @param: border_size the size in pixels of the border
            added to the document (default 15)
        @param: debug flag used to control img printing, True is
            Yes, False is No (default False)
        @param: figsize size of the plotting for each image (
            widht, height), this value is ignored if debug is
            False (default (10,10))

        :result Image
        """
        #This part applies resizing part
        res_img = PipelineImg.resizing_img(img, height = height)
        #This part applies binarization part to resized image
        bin_img = PipelineImg.binarization_img(res_img)
        #This part applies denoise part to binarized image
        den_img = PipelineImg.denoise_img(bin_img)
        #This part applies border delimitation algorithm to
            denoised image
        unb_img = PipelineImg.drop_border_img(den_img, original=
            res_img, sigma=auto_canny_sigma, bordersize=
            bordersize)

        #If debug flag is True, this part plot images for each
            previus step
        if debug:
            PipelineImg.plot_imgs_grid([img, res_img, bin_img,
                den_img, unb_img],
                ["original", "resize", "binarizated", "denoised",
                    "unbordered + border"], figsize=figsize)
        #The function returns unb_img which is the last image in
            the pre-processing flow
        return unb_img
```

A.3 MODELO DE DETECCIÓN DE FICHAS

A.3.1 *Etiquetado de fichas para entrenamiento*

Para el etiquetado de los datos se modelo una tabla que pudiera servir como recipiente(ver sección 2.7), la tabla fue almacenada en el sistema gestor de bases de datos MySQL.

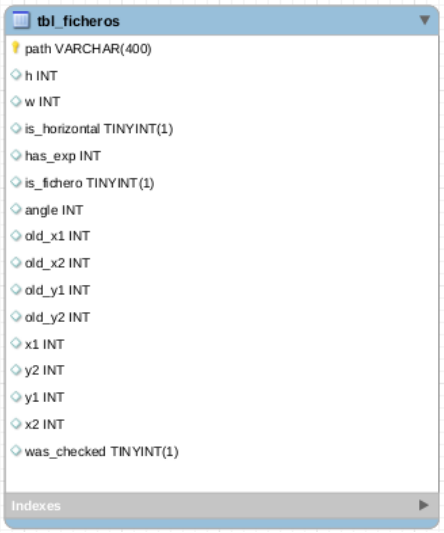


Figura A.1: Tabla diseñada para almacenar las anotaciones realizadas desde la aplicación en Flask.

Una vez con la base cargada con los datos de las fichas, se hizo disponible una aplicación en Flask que anotaba las fichas e iba registrando sus datos en la base de datos. La base de datos mostraba de forma aleatoria las fichas (sobre el supuesto de qué imágenes almacenadas en directorios cercanos tienen contenido y características similares, discutido en la sección A.1) y daba prioridad a aquellas fichas con las que se tuvo problemas al ejecutar el OCR.

El código utilizado puede verse en A.4, A.5 y A.6.

Campo	Descripción
path	Llave primaria, contiene la ruta de la imagen.
h	Longitud en píxeles correspondiente a la altura de la imagen.
w	Longitud en píxeles correspondiente al ancho de la imagen.
is_horizontal	Flag que contiene 1 si la imagen fue tomada horizontalmente.
has_exp	Flag que contiene 1 si la imagen contiene un expediente, 0 en otro caso.
is_fichero	Flag que contiene 1 si la imagen contiene un Fichero, 0 en otro caso.
angle	El angulo en grados que tiene que ser rotada la imagen para mostrar el fichero de forma horizontal.
old_x1	Coordenada x1 obtenida del flujo tradicional de detección de bordes (útil para facilitar la captura manual en la aplicación Flask).
old_x2	Coordenada x2 obtenida del flujo tradicional de detección de bordes (útil para facilitar la captura manual en la aplicación Flask).
old_y1	Coordenada y1 obtenida del flujo tradicional de detección de bordes (útil para facilitar la captura manual en la aplicación Flask).
old_y2	Coordenada y2 obtenida del flujo tradicional de detección de bordes (útil para facilitar la captura manual en la aplicación Flask).
x1	Coordenada x1 obtenida del anotador humano de la aplicación Flask, corresponde a las coordenadas del rectángulo que encierra el fichero dentro de la imagen.
y2	Coordenada y2 obtenida del anotador humano de la aplicación Flask, corresponde a las coordenadas del rectángulo que encierra el fichero dentro de la imagen.
y1	Coordenada y1 obtenida del anotador humano de la aplicación Flask, corresponde a las coordenadas del rectángulo que encierra el fichero dentro de la imagen.
x2	Coordenada x2 obtenida del anotador humano de la aplicación Flask, corresponde a las coordenadas del rectángulo que encierra el fichero dentro de la imagen.
was_checked	Flag que contiene 1 si la imagen ya ha sido anotada, 0 en caso contrario.

Tabla A.1: Diccionario de datos de la tabla tbl_ficheros. Tabla usada para almacenar las anotaciones de los ficheros de la DFS (ver sección 2.7.1).

Código A.4: Servicios implementados en Flask para consultar fichas y guardar el etiquetado de sus coordenadas y atributos (parte 1).

```
from flask import Flask, request, render_template, jsonify
import os
import mysql.connector
import pandas as pd
from flask_cors import CORS

#This line makes the connection to the database.
db_connection = mysql.connector.connect(
  host="localhost",
  user="python_user",
  passwd="password",
  database="database_ficheros"
)


cursor = db_connection.cursor()

#This line generates a Flask server and tells the class where it
    can find the html code of the templates used in the
    application.
app = Flask(__name__, template_folder = "templates")
#This lines allows users to execute locally deactivating security
     CORS Policy (https://cloud.google.com/apigee/docs/api-
    platform/reference/policies/cors-policy?hl=es-419#:~:text=
    Cross%2Dorigin%20resource%20sharing%20(CORS,is%20enforced%20
    by%20all%20browsers.)
CORS(app)
app.config['CORS\_HEADERS'] = 'Access-Control-Allow-Origin'
app.config['SEND_FILE_MAX_AGE_DEFAULT'] = 0

def connect():
    """
        This function re-establishes the connection to the
            database in case it is lost.
    """
    db_connection = mysql.connector.connect(
      host="localhost",
      user="python_user",
      passwd="password",
      database="database_ficheros"
    )

    cursor = db_connection.cursor()
```

Código A.5: Servicios implementados en Flask para consultar fichas y guardar el etiquetado de sus coordenadas y atributos (parte 2).

```
#The decorators tell the web server that the show_index function
    would provide the html code for the root and index paths.
@app.route('/')
@app.route('/index')
def show_index():
    """
        This function performs a query to the database and
            returns the necessary data to show a file in the
            application without annotating
    """

    #Database connection validation
    if db_connection is None:
        connect()
    else:
        pass

    #This line gets a random file that hasn't been annotated yet,
         and giving priority to files where we couldn't find
        files with the traditional flow
    cursor.execute("select * from tbl_ficheros where was_checked
         = 0 order by has_exp asc, RAND() limit 1 ")

    #Get columns name
    columns = [col[0] for col in cursor.description]

    #Get the row information in a dictionary structure
    fichero_info = [dict(zip(columns, row)) for row in cursor.
        fetchall()][0]

    #This line gets the number of files currently annotated and
        the ones that are missing
    cursor.execute("select was_checked, count(*) from
        tbl_ficheros group by was_checked ")
    progress = {}
    for el in cursor.fetchall():
        progress[el[0]] = el[1]
    fichero_info["faltan"] = progress[0]
    fichero_info["hechos"] = progress[1]

    #Send information to the front-end using index.html template
    return render_template("index.html", fichero_info =
        fichero_info)
```

Código A.6: Servicios implementados en Flask para consultar fichas y guardar el etiquetado de sus coordenadas y atributos (parte 3).

```
#This decorator tells the server that the foo function will
    process requests made by the application to save the image
    annotation.
@app.route('/update_fichero', methods=['POST'])
def foo():
    """
        This function stores the information annotated by the web
             annotator in the database
    """
    try:
        #Initialize the array that will be stored in the database
            , the value 1 is for was_checked field
        array_values=[1]

        #For each value we will obtain the value recorded in the
            application
        row={}
        fields = ["has_exp", "is_fichero", "angle", "x1",
                    "x2", "y1", "y2", "path"]
        for fld in fields:
            if fld == "path":
                row[fld] = request.form[fld]
            else:
                row[fld] = int(request.form[fld] )
            array_values.append(row[fld])

        #This linez stores the result of the annotation
        sql = "UPDATE tbl_ficheros SET  was_checked = {}, has_exp
            = {},  is_fichero = {},  angle = {},  x1 = {}, x2 =
            {}, y1 = {}, y2 = {} WHERE path = '{}'".format(*
            array_values)
        cursor.execute(sql)

        #Saves changes perdormed into the database
        db_connection.commit()
        #Tells the front-end that the request has been successful
        return {"status":True}
    except Exception as e:
        #Tells front-end that petition has been unsuccessful and
            return the error
        return {"status":False, "msg":str(e)}

#This line init the server in localhost ip (127.0.0.1 or
    localhost) in port 5000
app.run(host="0.0.0.0", port=5000)
```

A.3.2 *Métricas de entrenamiento del modelo detector de fichas (yolov5)*

Las métricas resultado de evaluar en el conjunto de entrenamiento el modelo entrenado se pueden encontrar en las tabla A.2, A.3, A.4 y A.5.

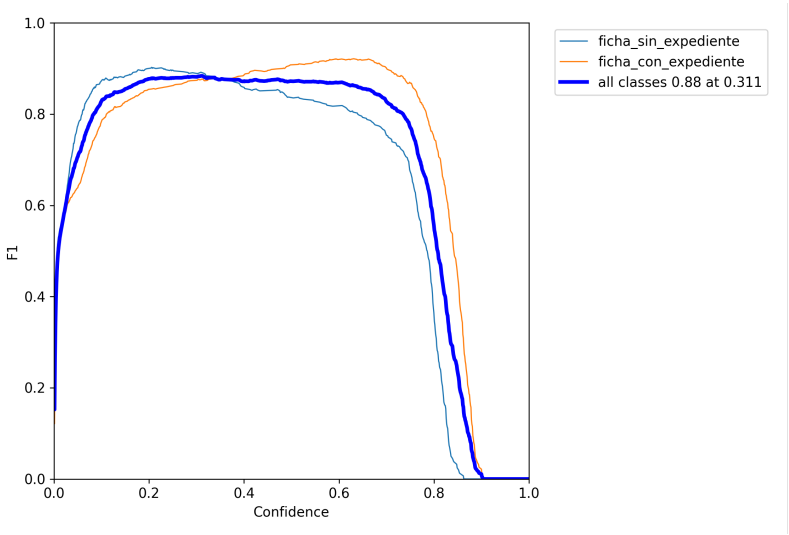


Figura A.2: Curva F1 del modelo final en el conjunto de validación. El valor F1 se usa para combinar las medidas de precisión y recall en un único valor. La medida toma el valor 1.0 cuando existe una clasificación perfecta (precisión 1.0 y recall 1.0). La curva nos muestra el valor F1 resultado de utilizar distintos cortes en la probabilidad del modelo detector de fichas (etiquetado como *Confidence*).

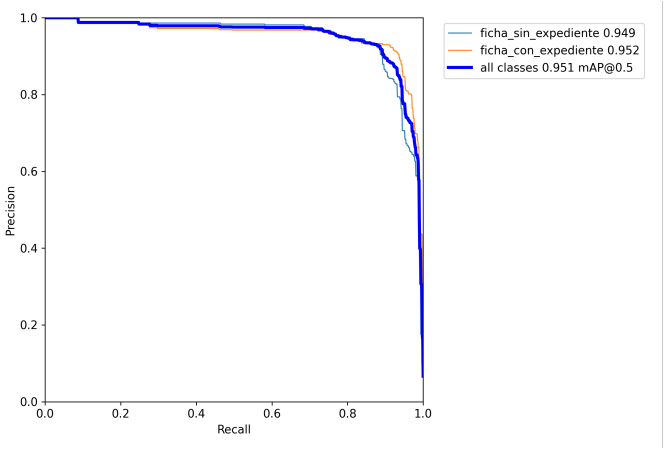


Figura A.3: Curva Precisión vs Recall del modelo final en el conjunto de validación. Al igual que lo hace la medida F1, para observar adecuadamente las medida de precisión o recall se necesitan ver de forma conjunta. Altos niveles de precisión con niveles altos de recall son buenas métricas de desempeño. En este caso en particular podemos observar que el modelo en su recall 0.8 todavía cuenta con precisiones cercanas al 0.9.

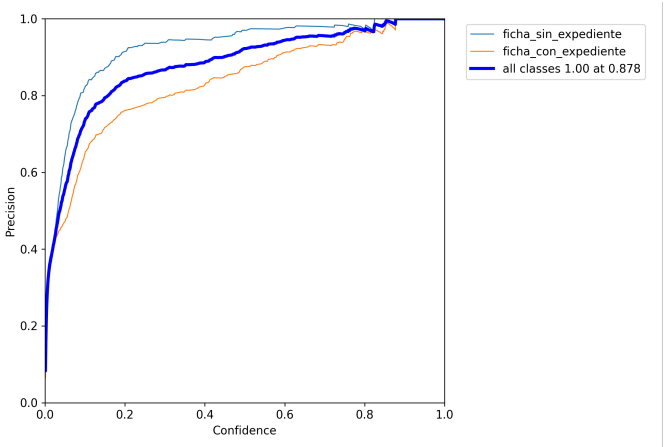


Figura A.4: Métricas de desempeño del modelo final yolov5 sobre el conjunto de validación: Curva Precisión vs Confianza del modelo final en el conjunto de validación. En el corte 0.5 de la variable de probabilidad obtenemos precisiones cercanas al 0.9, métricas en conjunto con recall y F1 consideradas aceptables.

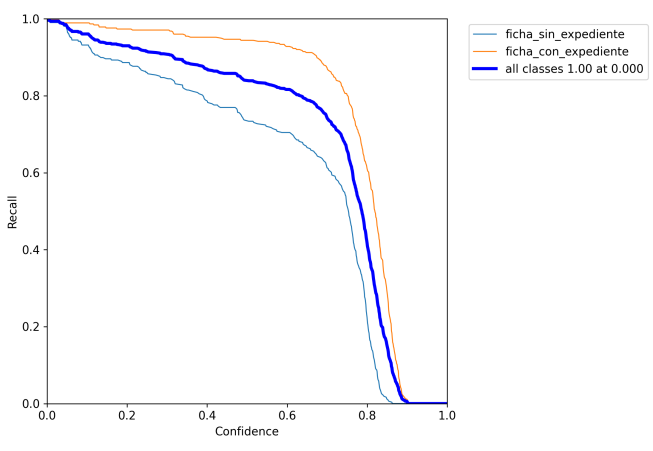


Figura A.5: Métricas de desempeño del modelo final yolov5 sobre el conjunto de validación: Curva Recall vs Confianza del modelo final en el conjunto de validación. En cuanto a esta métrica, se puede deducir que inclusive usando el corte 0.4 ya obtenemos arriba del 80% de las observaciones pertenecientes a nuestra clase a predecir. En caso de darle prioridad a la captura de los ficheros podría sacrificarse la precisión usando un corte igual o menor a 0.4, con tal de asegurar la mayor cantidad de ficheros procesados.

A.3.3 *Implementación del flujo completo en GitHub*

El código desarrollado para la realización de este trabajo se puede encontrar en el repositorio: https://github.com/datadogmx/files_dfs_ocr_reader. Dentro del repositorio se encuentran las instrucciones necesarias para instalar las librerías y binarios de los que depende el pipeline completo. Además, se disponibilizó un notebook de jupyter que muestra cómo se deben de usar las librerías creadas para cada paso de la metodología y una implementación para evaluar en paralelo, la cual puede ser muy útil para implementaciones en cantidades masivas de información.

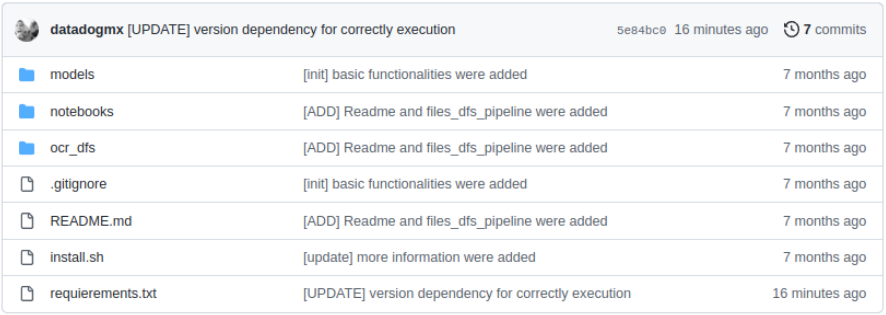


Figura A.6: Portada del repositorio.

README.md

Pipeline to extract text from DFS Files

This repository is the result of "Uso de Aprendizaje de Máquina para el análisis de los ficheros de la Dirección Federal de Seguridad (DFS)".

Examples

Example of all pipeline execute in files

Installation

The install.sh file will download yolov5 especific version used to train file_dfs_detector.

Execute install.sh: chmod +x install.sh & ./install.sh
Install requeriments: pip install -r requirements.txt

Structure

Module	Description
ocr_dfs.files_dfs_pipeline.py	This module contains functions and classes developed to implement all pipeline suggested for the authors to detect and extract dfs files information.
ocr_dfs.files_dfs_detector.py	This module contains functions and classes developed to detect orientation, dfs files into images.
ocr_dfs.text_dfs_analyzer.py	This module contains functions and classes developed to implement ocr text quality.
ocr_dfs.clean_image_pipeline.py	This module contains functions and classes developed to pre-process dfs files images.

Figura A.7: Archivo README.md del repositorio

A.4 RECOMENDACIONES EMITIDAS AL AGN PARA LA DIGITALIZACIÓN DE DOCUMENTOS HISTÓRICOS

El 15 de diciembre de 2021 se mantuvo una reunión con representantes del AGN con el objetivo de emitir recomendaciones para la digitalización de archivos de importancia histórica resguardados en la AGN. La reunión se realizón antes de la finalización de este trabajo con el objetivo de adelantarnos a la futura digitalización de documentos dentro del AGN, se les presento un borrador del presente trabajo. Tanto la Dra. Mariana Esther Martínez Sánchez como un servidor, transmitieron los siguientes puntos:

A.4.0.1 *Factores que afectan la correcta lectura (no controlables)*

- Deterioro de los documentos originales.
- Letra borrosa o poco nítida.
- Letras fragmentadas o solapadas.
- Tipografías extrañas.
- Documentos con contenido complejo (por ejemplo periódicos, documentos que aunque tienen un tamaño estandar suelen complementar su contenido con texto e imagenes, juegan con tipografias y sobre una misma linea puede haber mas de un contenido distinto)

A.4.0.2 *Factores que afectan la correcta lectura (controlables)*

- Rotación del documento: se recomienda que durante el proceso de digitalización masiva se homologue el ángulo para todas las imágenes, de preferencia alinear las letras horizontal a la toma. Sugerencia realizada a partir de los resultados en la sección 3.3.
- Fondos sucios o con garabatos: se recomendó el uso de un solo fondo, blanco y sin suciedad en caso de las fotografía. Sugerencia tomada de "Improving the quality of the output" (Tesseract-OCR, Consultado en 2022).
- Iluminación y sombras: se recomendó controlar la iluminación evitando sombras dentro de la imagen, se sugirió el uso de escáneres. Sugerencia tomada de la sección 2.2.2.2.
- Calidad de la imagen: recalcamos la importancia de la calidad de las imágenes durante la explotación por computadora. Resultado de los problemas encontrados y que dieron resultado a la metodología propuesta en el presente trabajo.

BIBLIOGRAFÍA

AGN. (Fecha de consulta: 2022). ¿Qué hacemos? https://www.gob.mx/agn/que-hacemos

Artículo 19 , COMVERDAD. (2020). Archivos de la represión. https://archivosdelarepresion.org/

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Buades, A., Coll, B., & Morel, J.-M. (2011). Non-Local Means Denoising [https://doi.org/10.5201/ipol.2011.bcm_nlm]. *Image Processing On Line*, *1*, 208–212.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., & Varoquaux, G. API design for machine learning software: experiences from the scikit-learn project. In: In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, 108–122.

CDHDF. (2016). Documental Guerrero: Memoria y verdad. https://cdhcm.org.mx/evento/documental-guerrero-memoria-y-verdad/

CNI. (2019). Conoce los antecedentes del CNI. https://www.gob.mx/cni/documentos/antecedentes-215445

Castellanos, L. (2013). *México armado. 1943-1981*. Ediciones Era.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, *2*(4), 303–314.

Devlin, M.-W. L. K., Jacob Chang, & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dwyer, B. (2021). Getting started with Roboflow. https://blog.roboflow.com/getting-started-with-roboflow/

FEMOSPP. (2006). Informe histórico a la sociedad mexicana. *The National Security Archive*.

Flask. (Consultado en 2022). Flask. https://palletsprojects.com/p/flask/

Gareth, J., Daniela, W., Trevor, H., & Robert, T. (2021). *An introduction to statistical learning: with applications in R* (Second). Springer.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning* [http://www.deeplearningbook.org]. MIT Press.

Hastie, R. F. J. H., Trevor Tibshirani, & Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction* (Vol. 2). Springer.

Jocher, G. (2021). GPU RAM requirements for training · discussion 5777 · ultralytics/yolov5. https://github.com/ultralytics/yolov5/discussions/5777

Jocher, G., Stoken, A., Borovec, J., NanoCode012, ChristopherSTAN, Changyu, L., Laughing, Tkianai, Hogan, A., Lorenzomammana, & et al. (2020). *ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements*. Zenodo. https://doi.org/10.5281/ZENODO.4154370

Luebker, M. (2010). *Inequality, income shares and poverty: the practical meaning of Gini coefficients* (tech. rep.). International Labour Organization.

Mehta, P., Bukov, M., Wang, C.-H., Day, A. G., Richardson, C., Fisher, C. K., & Schwab, D. J. (2019). A high-bias, low-variance introduction to machine learning for physicists. *Physics reports*, *810*, 1–124.

Montaño, C. I. V., Eugenia Allier y Ortega. (2017). México, 1968: violencia de Estado. Recuerdos del horror. *Theomai*, (36), 78–94.

Prasad, L. (2018). What is the difference between model hyperparameters and model parameters? https://datascience.stackexchange.com/questions/14187/what-is-the-difference-between-model-hyperparameters-and-model-parameters

Quezada, S. A. (2014). *La charola: una historia de los servicios de inteligencia en México*. Editorial Ink.

Rosebrock, A. (2015). Zero-parameter, automatic canny edge detection with python and opencv. https://pyimagesearch.com/2015/04/06/zero-parameter-automatic-canny-edge-detection-with-python-and-opencv/

Salazar Anaya, B., Delia Hernández. (2006). Guía del Fondo de la Secretaría de Gobernación. Sección Dirección de Investigaciones Políticas y Sociales, 1920-1950.

Singh, A., Bacchuwar, K., & Bhasin, A. (2012). A Survey of OCR Applications. *International Journal of Machine Learning and Computing*, *2*, 314–318. https://doi.org/10.7763/ijmlc.2012.v2.137

Tesseract Community. (2018). Optimal image resolution (DPI/PPI) for Tesseract 4.0.0 and eng.traineddata? https://groups.google.com/g/tesseract-ocr/c/Wdh_JJwnw94/m/24JHDYQbBQAJ?pli=1

Tesseract-OCR. (Consultado en 2022). Improving the quality of the output. https://tesseract-ocr.github.io/tessdoc/ImproveQuality

Tesseract-ocr. (s.f). tesseract: Tesseract Open Source OCR Engine (main repository).

Toledo, S. (2016). Informe final de actividades de la Comisión de la Verdad para la investigación de las violaciones a los Derechos Humanos durante la Guerra Sucia de los años sesenta y se-

tenta del estado de Guerrero. https://con-temporanea.inah.gob.mx/expediente_h/fragmentos

UNESCO. (2015). La ciencia al servicio de la sociedad. https://es.unesco.org/themes/ciencia-al-servicio-sociedad#:~:text=Lacienciaofrecesolucionespara,grandesmisteriosdelahumanidad.&text=Tieneunpapelfundamentaldel,aumentanuestracalidaddevida.

Venegas, J. M. (2000). Se creará comisión de transparencia sobre el 68: Fox. *La Jornada*.

Vicente Ovalle, C. (2019). Tiempo suspendido: una historia de la desaparición forzada en México, 1940-1980. *Tiempo suspendido*, 1–359.

Yankelevich, J. (2018). El canto del cisne de la FEMOSPP: la única condena a un perpetrador de la guerra sucia en Mexico. *Unpublished Manuscript*.

Ávila, D. (2020). Documentos de DGIPS y DFS pueden ser consultados sin restricciones: AGN. *PROCESO*.