Universidad de la República
Facultad de Ingeniería

Sellos LUISA

Memoria de proyecto presentada a la Facultad de Ingeniería de la Universidad de la República por

Andrés Arobba, Guillermo Sosa

en cumplimiento parcial de los requerimientos para la obtención del título de Ingeniero Electricista.

Tutor

Ignacio Ramirez ..... Universidad de la República

Tribunal

Emilio Martinez ..... Universidad de la República
Álvaro Gomez ..... Universidad de la República
Fernando Carpani..... Universidad de la República

Montevideo
lunes 29 julio, 2024

Sellos LUISA, Andrés Arobba, Guillermo Sosa.

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 124 páginas.
Compilada el lunes 29 julio, 2024.

Agradecimientos

Queremos agradecer a las siguientes personas que nos han acompañado y alentado durante el proceso de elaboración de este proyecto:

A nuestras familias y amigos por el apoyo inagotable a lo largo de todo nuestro proceso académico y en particular en este proyecto. Gracias por haber sido sostenes emocionales fundamentales durante este período.

A Nacho, nuestro tutor, por su capacidad de liderazgo y guía sin la cual no hubiese sido posible este proyecto. Gracias por la disposición, el compañerismo genuino y el compromiso con el proyecto mostrados que nos inspiran a liderar nuevos proyectos.

A los docentes Guillermo Carbajal y Emiliano Acevedo, por compartir sus conocimientos y experiencia con nosotros. Gracias por su disponibilidad y colaboración.

A los voluntarios Damián Pintos, Ivana Otero y Cristian Callero, quienes dedicaron su tiempo al proceso de etiquetado y registro manual de la información de los sellos en varios rollos de documentos. Gracias por brindarnos la información necesaria para poder comenzar con el desarrollo del presente proyecto.

A la Facultad de Ingeniería y en particular al Instituto de Ingeniería Eléctrica por brindarnos las herramientas necesarias para nuestro desarrollo académico.

Esta página ha sido intencionalmente dejada en blanco.

Resumen

El presente proyecto tiene el siguiente objetivo:

- Determinar automáticamente qué sellos contienen los documentos escaneados.

El motivo principal es contribuir al proyecto LUISA en la organización de los documentos del Archivo Berrutti, con el fin de facilitar la búsqueda digital, promoviendo la investigación y el esclarecimiento de los acontecimientos ocurridos durante la dictadura cívico-militar de nuestro país.

El proyecto LUISA es un proyecto interdisciplinario que busca extraer información relevante de documentos de la dictadura, al contar con una cantidad grande de documentos, esta extracción debe ser automática para poder realizar una investigación exhaustiva de la información allí contenida.

El Archivo Berruti es una recopilación de rollos de microfilms generados durante la dictadura cívico-militar, posteriormente encontrados, escaneados y almacenados digitalmente en el Repositorio Luisa Cuesta. Recientemente se liberó dicho repositorio para que cualquier persona pueda solicitar la consulta de los documentos.

Para cumplir con dicho objetivo se fragmentó el trabajo en tres etapas:

- **Etapa 1**: Detectar automáticamente todos los sellos presentes en los documentos.
- **Etapa 2**: Implementar un sistema de clasificación de sellos.
- **Etapa 3**: Integrar las etapas anteriores, clasificando los documentos en función de los sellos que aparezcan en estos.

El desarrollo de cada una de estas etapas no fue independiente entre sí. En la fase final del proyecto se realizó el proceso de integración entre las dos primeras donde las detecciones se convierten en las instancias a clasificar, con el fin de asociar el documento con la clase del sello correspondiente.

Es importante tener en consideración que este proyecto de fin de carrera no pretende ser un trabajo definitivo. En su lugar, busca desarrollar un prototipo capaz de ser empleado, ajustado y personalizado por usuarios para abordar las necesidades cambiantes sobre investigaciones futuras.

La etapa de detección prioriza extraer la información relevante de los documentos (los sellos presentes) realizando una búsqueda rápida a través de técnicas clásicas de *template matching*. Por otra parte, mediante el uso de inteligencia artificial, más específicamente utilizando redes neuronales convolucionales, el objetivo del clasificador es identificar características específicas de los sellos agrupándolos en distintas clases. Finalmente, la última etapa integra las anteriores y se encarga de determinar qué sellos hay en cada imagen.

Una vez finalizada la implementación de cada etapa, se realizaron las pruebas correspondientes para evaluar si cada una cumplió con los objetivos establecidos. Se verificó:

- **La detección y ubicación automática de sellos en documentos.**
- **La clasificación de sellos.**

En conclusión, en este proyecto se presenta un método de organización y estructuración de documentos del Archivo Berrutti, mediante la implementación de un clasificador de documentos en función de los tipos de sellos. Los resultados obtenidos en los ensayos experimentales son considerados satisfactorios, confirmando el cumplimiento del objetivo principal.

Lista de abreviaturas

Para facilitar la comprensión de los conceptos utilizados a lo largo del presente documento, se detalla un listado de abreviaturas junto con sus respectivos significados, como se muestra en la tabla 1.

Abreviatura	**Significado**
LUISA	Leyendo Unidos para Interpretar loS Archivos
PFC	Proyecto Fin de Carrera
FING	Facultad de Ingeniería
FIC	Facultad de Información y Comunicación
FNCC	Fast Normalized Cross Correlation
NCC	Normalized Cross Correlation
FFT	Fast Fourier Transform
PR	Precision - Recall
RAM	Memoria de Acceso Aleatorio
GPU	Unidad de Procesamiento Gráfico
SSH	Secure Shell
CNN	Convolutional Neural Network

Tabla 1: Lista con abreviaciones mencionadas a lo largo de este proyecto.

Esta página ha sido intencionalmente dejada en blanco.

Tabla de contenidos

Tabla de contenidos

Capítulo 1

Introducción

1.1. Contexto

Para comprender el contexto del presente proyecto de fin de carrera, es esencial conocer el marco histórico del tema y la tarea que desarrollan las organizaciones que trabajan en el estudio y preservación de documentos relacionados con las violaciones de derechos humanos y el funcionamiento del aparato estatal durante la dictadura cívico militar en Uruguay, período que abarcó más de una década, extendiéndose desde el año 1973 hasta 1985.

Luego del retorno a la democracia en el país han surgido diversas organizaciones lideradas por familiares de desaparecidos, estudiantes y ciudadanos motivados por la defensa de los derechos humanos. Su propósito consiste en la investigación y el esclarecimiento de los acontecimientos ocurridos en esta etapa histórica de nuestro país, en la búsqueda de la verdad, desempeñando un papel crucial en el proceso de documentación, concientización y trasmisión de la memoria histórica.

Vivimos en una sociedad fragmentada, atravesada por situaciones de violencia cada vez más frecuentes, en la que predomina la individualidad y la distancia entre los ciudadanos según afiliaciones políticas, preferencias ideológicas y factores educativos y sociales. A pesar de ello subsiste el denominador común del sentido de pertenencia a un Estado democrático, históricamente arraigado de una población que busca la paz, la libertad y el bienestar para llegar a completar el concepto de verdad histórica y justicia que logre, finalmente, reconciliar a la sociedad uruguaya.

CRUZAR (Sistema de información de archivos del pasado reciente)

CRUZAR es un proyecto de extensión de la Universidad de la República en el que participan docentes y estudiantes de la Facultad de Información y Comunicación, Facultad de Ingeniería y Facultad de Ciencias Sociales. Su objetivo principal es

aportar a la búsqueda de verdad y justicia sobre los hechos ocurridos durante el terrorismo de estado en Uruguay mediante el procesamiento de documentos de la época.

'Cruzar refiere a un proyecto de sistematización de información de archivos del pasado reciente vinculados al terrorismo de estado y graves violaciones a los Derechos Humanos. Tiene como objetivo el ordenamiento y la clasificación del material, y la elaboración de un programa informático que permita el cruzamiento de la información contenida en esos archivos. El cruzamiento facilita la investigación y el análisis de los temas referidos al terrorismo de Estado, como un aporte más en la búsqueda de la verdad.' [1]

LUISA (Leyendo Unidos para Interpretar loS Archivos)

El proyecto LUISA es un proyecto que se enmarca dentro del proyecto CRUZAR. El nombre LUISA es un acrónimo sugerido por docentes de la Facultad de Ingeniería en homenaje a Luisa Cuesta, referente de la lucha por la búsqueda de los detenidos desaparecidos durante la dictadura cívico militar ya mencionada.

Se trata de una iniciativa que combina el uso de la inteligencia artificial y la colaboración humana para digitalizar y transcribir archivos escritos durante la dictadura y posteriormente escaneados. Este proyecto busca preservar, facilitar el acceso y la investigación de estos documentos, convirtiéndolos de imágenes a formato texto y extrayendo otras características de interés. Esto permite realizar búsquedas digitales de forma ágil pudiendo trabajar con el material disponible, el cual contiene información importante para comprender la historia del pasado reciente. [2]

1.2. Motivaciones

La tarea de preservar y organizar estos documentos se ve obstaculizada por varios factores. En primer lugar, la cantidad de documentos recopilados es significativamente grande, alcanzando cerca de tres millones de registros. Pero además de la dimensión de su volumen, este material se encuentra ordenado únicamente por rollos de microfilms, resultando cualquier intento de búsqueda totalmente ineficiente.

Por lo tanto, la principal motivación para desarrollar un enfoque automatizado radica en la necesidad de gestionar eficientemente este vasto conjunto de documentos, clasificándolo según ciertos estándares para extraer información relevante de manera precisa y oportuna.

1.3. Trabajos similares

El artículo [3] identifica varios enfoques posibles para abordar problemas similares, haciendo una distinción principal entre métodos de detección de sellos y métodos

de reconocimiento o clasificación. Además, dentro de los métodos de detección, se destacan cuatro tipos principales:

- Basados en las componentes conexas.
- Basados en ventanas deslizantes.
- Basados en aprendizaje automático.
- Basados en descriptores locales.

Los métodos basados en componentes conexas binarizan la imagen para posteriormente generar una lista de componentes conexas. Luego, se calculan características de las componentes (ancho, alto, etc.) para finalmente clasificarlas. Los estudios en esta línea [4] [5] no están diseñados para conjuntos de datos donde suele haber texto superpuesto a los sellos o sellos con muy poca tinta, lo que genera componentes conexas muy variables.

Por otro lado, los métodos basados en ventanas deslizantes analizan subimágenes de la imagen original (parches), recorriendo toda la imagen y generando estadísticas locales. Los estudios en esta línea [6] [7] indican que es una buena estrategia para abordar el problema. El método de detección abordado durante este proyecto se puede clasificar dentro de esta categoría.

Los métodos basados en aprendizaje automático procesan las imágenes directamente con Redes Neuronales Convolucionales [8, capítulo 14] , estos algoritmos pueden requerir muchos datos para lograr desempeños aceptables. Los estudios en esta línea [9] [10] muestran un buen desempeño y por lo tanto también podrían ser una buena estrategia.

El algoritmo presentado en [9] utiliza la herramienta YOLO (You Only Look Once [8, capítulo 14]), esta herramienta consta de una red neuronal preentrenada originalmente pensada para imágenes naturales, para adaptarla a este problema habría que reentrenar algunas capas del modelo. Por otro lado [10] consta de una red preentrenada, cuya salida es la imagen de entrada pero borra todas las zonas que no contengan un sello. Al igual que YOLO, habría que reentrenar algunas capas para adaptarla a nuestro problema.

Por último, los métodos basados en descriptores locales son técnicas clásicas de procesamiento de imágenes. Los estudios en esta línea [11] [12] [13] [14] [15] utilizan la Generalized Hough Transform (GHT [16, capítulo 8]), o Scale-Invariant Feature Transform (SIFT [17, capítulo 11]) para extraer posibles zonas del documento donde hay sellos. Estos métodos podrían ser una buena estrategia para la detección de sellos.

Por otro lado, los métodos de reconocimiento o clasificación se basan en las características extraídas de la detección (si en esta etapa se utilizan métodos de

descriptores locales) o directamente en los parches de imágenes candidatos a sellos. Para procesar características (ancho, alto, etc.), se utilizan SVMs [8, capítulo 5] o regresión lineal [8, capítulo 4] , mientras que para procesar los parches se utilizan Redes Neuronales Convolucionales.

También se puede utilizar el histograma de distancia de puntos, como en [18]. En este trabajo, que no abarca la detección, después de utilizar dicho histograma, se realiza Principal Component Analysis (PCA [8, capítulo 8]) para reducir la dimensionalidad y luego Linear Discriminant Analysis (LDA [8, capítulo 8]) para discriminar entre sellos. Estos últimos dos son ambos métodos de aprendizaje automático. Por lo tanto, el aprendizaje automático se utiliza comúnmente para la clasificación.

Capítulo 2

Presentación del proyecto

La necesidad de abordar una gran cantidad de información sin una estructura clara ha generado la oportunidad para desarrollar soluciones automáticas que faciliten la gestión y el análisis eficiente de estos datos.

En el marco de este problema, el presente proyecto desarrolla una solución para clasificar automáticamente documentos según los sellos que contengan los mismos. Es decir, a partir de determinadas clases de sellos, determinar cuáles se encuentran en cada documento.

2.1. Base de datos: Archivo 'Berrutti'

Para el desarrollo del proyecto es primordial contar con acceso a una base de datos lo más completa posible, con información necesaria para entrenar y evaluar los modelos a implementar.

El Archivo 'Berrutti' pertenece al Ministerio de Defensa y está compuesto por unos 1500 rollos de microfilms, con más de 1000 imágenes cada uno, los cuales corresponden a archivos y documentos militares comprendidos durante el año 1968 y el 2004 aproximadamente. Dado lo sensible de la información contenida en el Archivo, fue necesario pedir permisos especiales justificando su utilización únicamente con fines académicos. Hoy en día el Archivo está disponible para todo público [1].

Se cuenta con acceso a dos conjuntos de datos:

- **Imágenes de documentos escaneados.**
- **Imágenes de sellos que aparecen en estos documentos.**

[1] https://www.gub.uy/institucion-nacional-derechos-humanos-uruguay/politicas-y-gestion/repositorio-luisa-cuesta

CLASE	CANTIDAD DE DATOS
1) Inteligencia	537
2) Junta	528
3) Juzgado	295
4) Estado Mayor	661
5) Penitenciario	505

Tabla 2.1: *Ground truth*: Tamaño de la base de datos generada previamente por estudiantes de la FIC.

En la figura 2.1 se puede observar un ejemplo de un documento escaneado.

Originalmente se cuenta con un subconjunto del Archivo Berrutti, este fue previamente procesado por estudiantes de la FIC mediante el programa de etiquetado manual Label Me y consta aproximadamente de 75.000 imágenes. Estos archivos clasificados (de ahora en más datos etiquetados) tienen asociado un archivo en formato `JSON` [2] por cada documento. Para este proyecto será de interés la información relativa a los sellos contenidos, más específicamente a la ubicación de los mismos en el documento, en la segunda imagen de la figura 3.12 se observan los recuadros azules que son las coordenadas indicadas en el JSON asociado, estos recuadros también se llamarán etiquetas. Dicho subconjunto tiene:

- 75.000 imágenes de documentos escaneados y etiquetados.
- 10.000 imágenes con sellos.
- 12.000 imágenes de sellos extraídos y además categorizados por clases.

Algunos de los tipos de sellos que serán utilizadas como apoyo a lo largo del informe se encuentran en la figura 2.2 (de ahora en más sellos de interés). La elección de estas clases de sellos se basa en la cantidad de muestras de las que se parte (tabla 2.1).

Cabe observar que de los 12.000 sellos originales, solo 2.500 aproximadamente pertenecen a las cinco clases de interés, esto quiere decir que hay aproximadamente 9.500 que no pertenecen a las clases de interés y que sus respectivas clases no tienen más de 295 muestras. Particularmente hay muchas clases que tienen una o unas pocas muestras.

[2] https://es.wikipedia.org/wiki/JSON

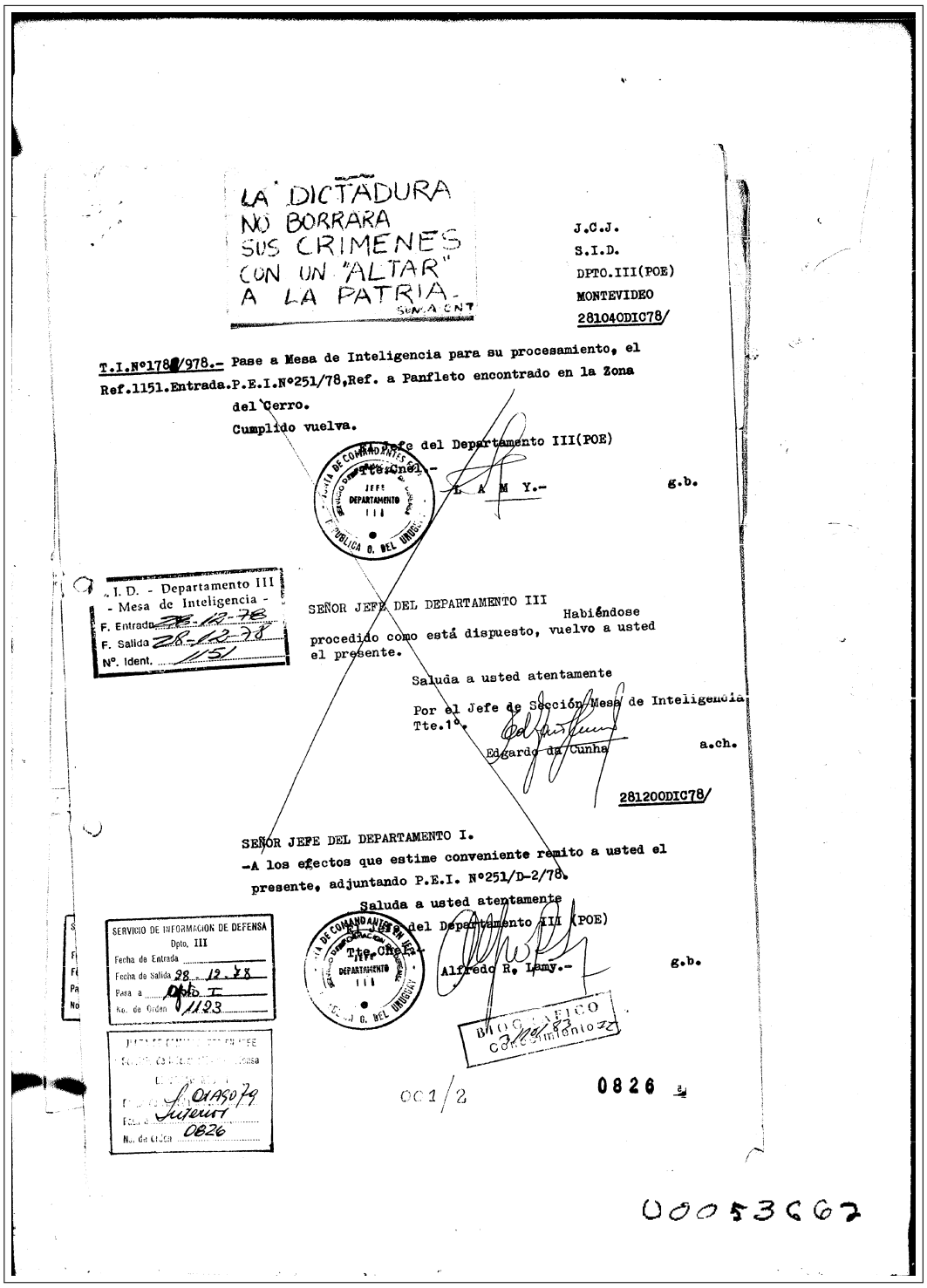


Figura 2.1: Ejemplo de un documento escaneado perteneciente al Archivo 'Berrutti'. En este documento se observan varios tipos de sellos (circulares, rectangulares) e incluso firmas. Para el análisis de este proyecto, el enfoque estará puesto en los sellos circulares (aunque puede ser aplicable a otros tipos de figuras).

(a) Inteligencia (b) Junta (c) Juzgado (d) Mayor (e) Penitenciario

Figura 2.2: Ejemplos de las cinco clases representativas de sellos a clasificar.

Con respecto a algunas características de las imágenes, todas se encuentran en representación binaria y están almacenadas en formato `TIFF` [3]. A su vez, las imágenes de los documentos escaneados abarcan una resolución de 4000 × 3000 píxeles aproximadamente, mientras que los sellos más grandes alcanzan un tamaño de 500 × 500.

2.2. Solución propuesta

En una primera instancia, se planteó la posibilidad de resolver el problema mediante inteligencia artificial utilizando redes neuronales. La idea inicial era introducir los documentos escaneados como entrada y, a través del proceso de clasificación, organizar los documentos según los sellos contenidos en ellos.

Sin embargo, esta aproximación presenta un desafío significativo: los documentos son de alta resolución (4000 × 3000), incluso escalando la imagen se necesitarían muchos parámetros y por lo tanto muchos datos de entrenamiento. Además, los sellos representan solo una pequeña porción (500 × 500) de la imagen total, utilizar toda la imagen del documento implicaría una gran cantidad de parámetros que no aportarían información relevante para la tarea de clasificación.

Por lo tanto, se propone una alternativa más eficiente: recortar los documentos para enfocarse exclusivamente en la región que contiene los sellos. Utilizando estos recortes se reduce significativamente el tamaño de entrada a la red, centrándose en la información relevante para nuestro objetivo.

Se aborda la propuesta subdividiendo el objetivo principal en tres etapas:

- **Etapa 1: Detección**. El objetivo de esta etapa es detectar automáticamente todos los sellos presentes en los documentos.
- **Etapa 2: Clasificación**. Implementar un sistema de clasificación de sellos.
- **Etapa 3: Integración**. Unificar las etapas anteriores, organizando los documentos en función de las clases de los sellos correspondientes.

[3] https://es.wikipedia.org/wiki/TIFF

En la etapa 1 se implementa el bloque de detección calculando correspondencias entre características en dos imágenes diferentes (*matching*) para localizar candidatos a sellos en los documentos. Luego, estos candidatos son procesados en la etapa 2 con el bloque de clasificación, diseñando un modelo basado en una arquitectura de redes neuronales convolucionales.

El presente proyecto no pretende ser un trabajo definitivo, sino más bien un sistema flexible y adaptable a las necesidades específicas de los usuarios (se entiende por usuarios aquellos que pretendan ordenar los documentos mediante clases de sellos de interés y requieran del uso de la herramienta desarrollada en este proyecto). La arquitectura y el diseño están concebidos con la intención de ser modificables, permitiendo su adaptación a una amplia variedad de aplicaciones. Esta flexibilidad garantiza que el trabajo pueda evolucionar junto con las demandas emergentes, asegurando su continuidad en el campo de estudio.

2.3. Métricas de evaluación

Es necesario establecer métricas fieles para la evaluación, optimización y comunicación del rendimiento de cada algoritmo implementado, esto permite que la toma de decisiones y la continuidad en la mejora de soluciones esté fundamentada por resultados que aporten información relevante.

Métrica del detector

La métrica empleada en este estudio es la curva de *Precision-Recall* (PR), ésta es comúnmente utilizada para presentar resultados en el contexto de problemas de decisión binaria, donde el objetivo es clasificar instancias en dos categorías distintas. Algunos términos comúnmente utilizados para describir los resultados de un modelo en estos problemas son:

- Verdaderos positivos (*True Positives (TP)*): sellos detectados que efectivamente son sellos.
- Falsos positivos (*False Positives (FP)*): sellos detectados como tales, pero no son sellos.
- Falsos negativos (*False Negatives (FN)*): sellos no detectados, donde hay sellos.
- Verdaderos negativos (*True Negatives (TN)*): sellos no detectados donde no hay sellos.

Para ayudar al lector se complementa esta definición con la matriz de confusión de la figura 2.3.

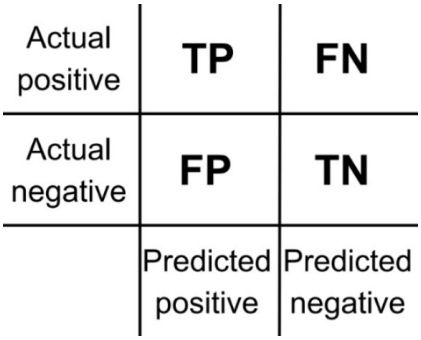


Figura 2.3: Matriz de confusión. Se comparan las predicciones generadas y los verdaderos valores de las etiquetas

Se define la métrica de *Precision* como la proporción de instancias clasificadas como positivas que son verdaderamente positivas respecto al total de instancias clasificadas como positivas. El cálculo se detalla en la ecuación (2.4).

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{2.1}$$

Por otro lado, la métrica de ratio de verdaderos positivos, más conocida como *Recall* es la proporción de instancias positivas que son clasificadas correctamente como positivas por el modelo respecto al total de instancias realmente positivas; y se calcula como en la ecuación (2.2).

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{2.2}$$

El presente problema de detección consiste en diferenciar si una imagen es considerada sello o no. La prioridad es detectar la mayor cantidad de sellos posible, a pesar de que la tasa de *FP* sea alta y esto derive en bajos valores de *Precision*. En el espacio de la curva PR 2.4 (*Precision-Recall*), esto puede verse reflejado por una curva que no se aproxime lo suficiente a la esquina superior derecha, lo cual es considerado óptimo (la esquina superior derecha indica valores de recall y precisión cercanos a uno, osea se alcanza la mayoría de verdaderos positivos y pocos falsos positivos). Sin embargo, valores altos de *Recall* demuestran que el detector es capaz de encontrar un alto porcentaje de sellos sobre los documentos, minimizando la cantidad de sellos omitidos.

Métricas del clasificador

En el contexto del aprendizaje automático, especialmente con redes neuronales, la función de pérdida (*loss*) y las métricas son dos conceptos claves para evaluar el rendimiento del modelo.

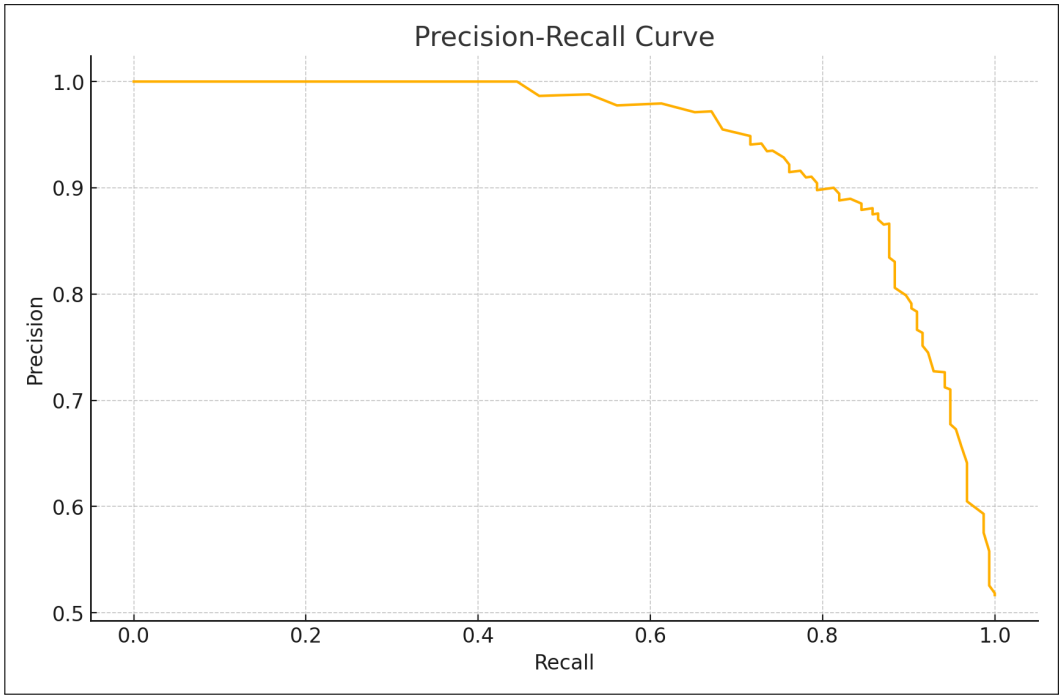


Figura 2.4: Ejemplo de curva PR.

Por un lado, la función de pérdida es una medida de error que determina cuánto se desvía el modelo de la salida deseada. Al momento de entrenar un modelo el objetivo es minimizar esta pérdida. Dicha función depende del problema y del tipo de salida esperada. Para el caso de clasificación multi-clase se suele utilizar como pérdida la entropía cruzada. Por ejemplo para una sola instancia de datos esta se define como:

$$L = -\sum_{i=1}^{N} y_i log(p_i) \tag{2.3}$$

Donde:

- N es la cantidad de clases.
- y_i es la etiqueta correspondiente a la instancia. Por ejemplo, si la etiqueta corresponde a la clase 1:
  - $y_1 = 1$
  - $y_i = 0, \forall i \neq 1$
- p_i es la probabilidad estimada por el modelo de que una instancia pertenezca a la clase i.

Por otra parte, las métricas son indicadores de rendimiento que se calculan para evaluar el desempeño del modelo en un conjunto de datos de entrenamiento y de evaluación. Se utiliza la precisión como métrica principal, si bien el cálculo es igual a la ecuación (2.4), en este contexto de clasificación los TP se corresponden con las muestras bien clasificadas y los FP con aquellas mal clasificadas. Esta métrica es útil en un clasificador ya que permite cuantificar la proporción de predicciones correctas que el modelo realiza sobre el total de predicciones realizadas.

Por otra parte, las métricas son indicadores de rendimiento que se calculan para evaluar el desempeño del modelo en un conjunto de datos de entrenamiento y de evaluación. Se utiliza la accuracy como métrica principal ya que es útil en un clasificador porque permite cuantificar la proporción de predicciones correctas que el modelo realiza sobre el total de predicciones realizadas.

$$\text{Accuracy} = \frac{\text{Predicciones correctas}}{\text{Total de predicciones}}. \tag{2.4}$$

Capítulo 3

Etapa 1 - Detección

En el presente capítulo se justifica y detalla la solución implementada para detectar la ubicación de sellos en documentos, se evalúan técnicas y se discute sobre cuál es la más apropiada en términos de precisión y velocidad de ejecución.

3.1. Descripción del problema

La detección de objetos en imágenes es un proceso en el que se identifican y localizan objetos de interés en una imagen. El emparejamiento de plantillas (*template matching*), es una técnica clásica comúnmente utilizada en este tipo de tareas. El proceso consiste en tomar una imagen de referencia, llamada plantilla, y buscar su presencia en una imagen de entrada. Se calcula una medida de similitud entre la plantilla y diferentes regiones de la imagen de entrada, y la región con mayor similitud es considerada la ubicación del objeto detectado.

En el presente caso los objetos a detectar son sellos grabados manualmente sobre una hoja de papel, los cuales no son siempre iguales y tienen deformaciones. Esto presenta dificultades al momento de aplicar este método, ya que las deformaciones pueden ser debidas a cambios de iluminación, rotaciones, escalas, excesos de tinta, texto superpuesto o incluso que en el proceso de digitalización de imágenes el escáner haya modificado parte del sello.

Esto puede llevar a problemas de falsos positivos, donde la plantilla elegida y una región de la imagen que no contiene el objeto de interés encuentran una similitud de forma errónea. También a un problema de falsos negativos, ya que una leve alteración en el sello puede llevar a que no se puede detectar.

Para hacer frente a estas dificultades es importante robustecer estas técnicas. En las siguientes secciones del capítulo se detalla el análisis de algunas posibles soluciones a esta problemática, así como también alternativas para que el proceso sea lo más

ágil y preciso posible.

3.2. Métricas de similitud entre dos imágenes

En esta sección se estudian métricas de comparación entre imágenes. ¿Cómo definir si una imagen es lo suficientemente similar a otra para catalogarla como una detección satisfactoria? Es necesario establecer cuantitativamente un límite que diferencie si una imagen es un potencial candidato a sello o no.

Para favorecer el análisis en el desarrollo de las siguientes técnicas, se toma el documento escaneado de la figura 3.1 a modo de ejemplo.

3.2.1. Correlación Cruzada Lineal *(Linear Cross-Correlation)*

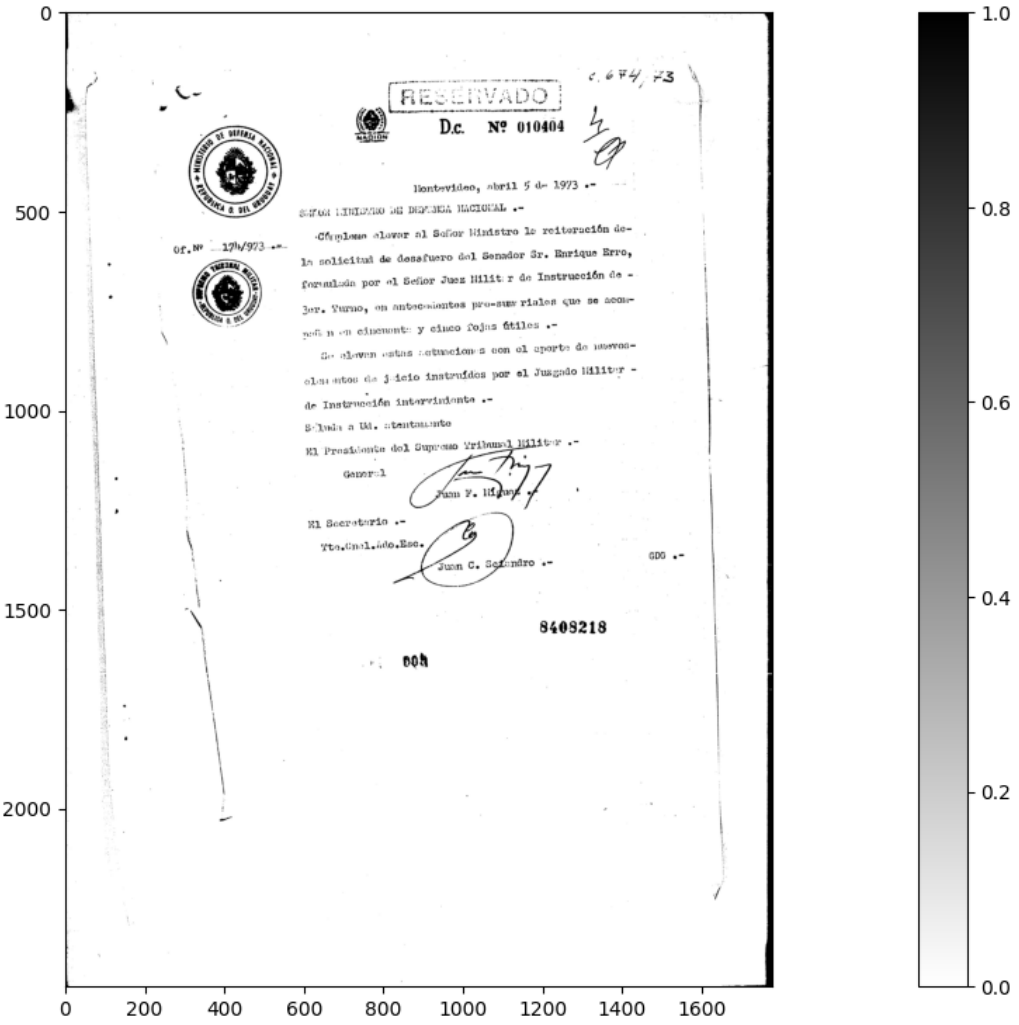
La primer medida considerada de similitud entre dos señales es la correlación cruzada lineal, deducida a partir del cálculo de la distancia euclidiana. Esta operación matemática (3.1) multiplica coordenada a coordenada una imagen de referencia t (sello o plantilla) con otra de interés f (documento) y luego suma estas multiplicaciones para evaluar la correspondencia entre las dos imágenes en diferentes posiciones de superposición. De ahora en más se considerará que la imagen de referencia es de dimensión $M_t \times N_t$ y la de interés de $M \times N$. El par de coordenadas (u, v) corresponde a la posición de la imagen de interés, mientras que (x, y) representa las coordenadas de la imagen de referencia, la cual es desplazada sobre la imagen de interés.

$$C_{tf}(u, v) = \sum_{(x,y)} f(x, y)t(x - u, y - v) = t \otimes f \tag{3.1}$$

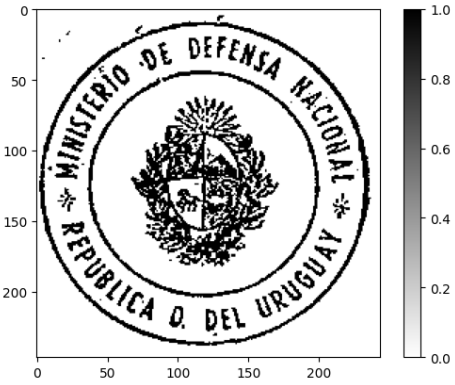
La imagen $C_{tf}(u, v)$ es del mismo tamaño que el documento y contiene en cada una de sus coordenadas el valor de la correlación entre el sello y el parche de la imagen correspondiente, a esta nueva imagen se le llama mapa de correlación.

Si se quiere realizar la correlación de una imagen con una versión desplazada de sí misma, el resultado en cada posición será la suma de los productos de los valores de los píxeles superpuestos. Resulta claro entonces que el valor más fuerte de correlación se ubique en la esquina superior izquierda de la versión desplazada, debido a la naturaleza de la operación. Allí hay un perfecto alineamiento entre la imagen original y la versión desplazada, y por lo tanto los productos de los píxeles en cada posición son los más altos cuando la imagen está superpuesta completamente. A medida que se desplaza la imagen, los píxeles dejan de coincidir perfectamente, resultando en valores de correlación cada vez menores.

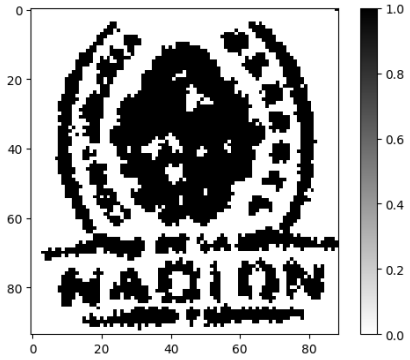
Un aspecto a destacar en este caso particular es que el valor de la correlación cuando ambas imágenes coinciden perfectamente es igual a la suma de los píxeles



(a) Documento: imagen de interés (*feature*).



(b) Sello: imagen de referencia o plantilla (*template*) presente en el documento.



(c) Sello: otro (*template*) presente en el documento.

Figura 3.1: Documento y sellos sobre los que se aplicarán las operaciones, se toman blanco y negro como 0 y 1 respectivamente.

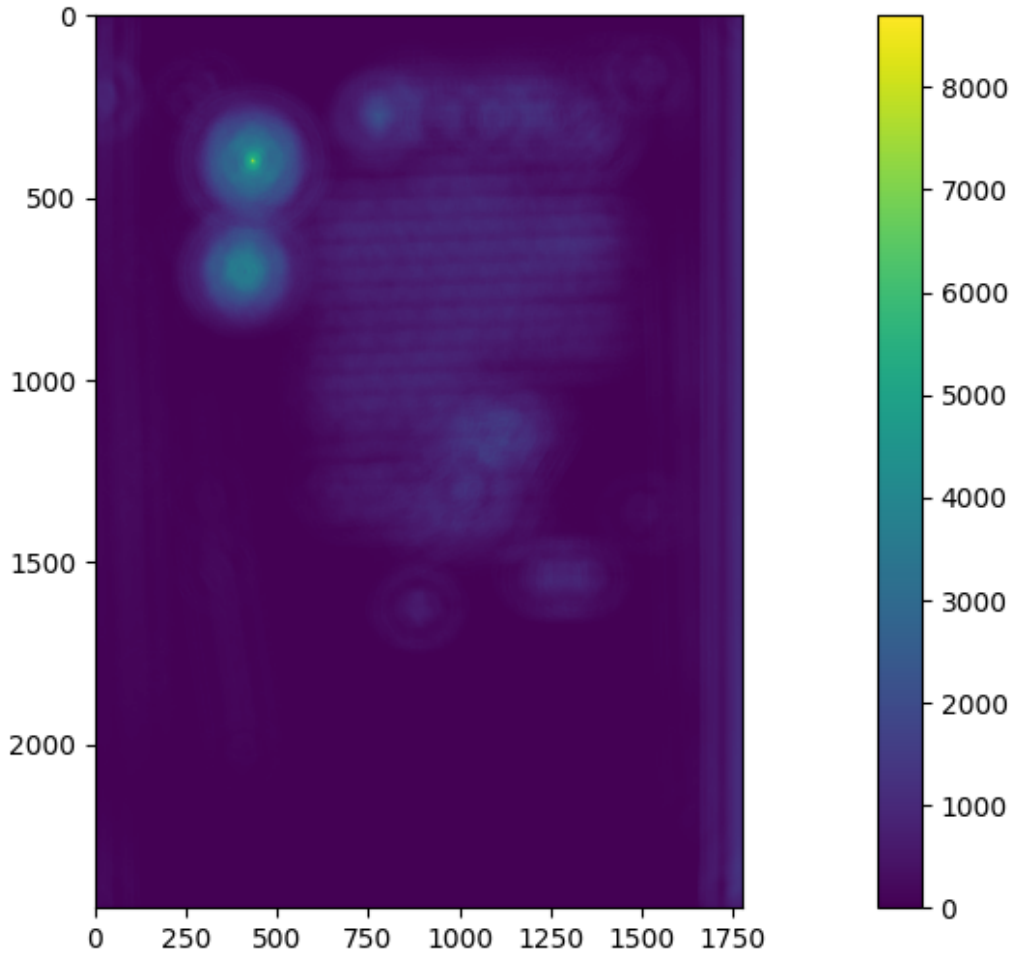


Figura 3.2: Mapa de correlación entre el documento y el primer sello de la figura 3.1. Regiones del mapa con color claro se corresponden con valores altos de correlación (como se muestra en la escala), indicando posibles zonas con sellos. El máximo valor de correlación se da en las coordenadas donde se encuentra el sello en el documento de la figura 3.1 y su valor es 8.967, cabe observar que es menor a la energía del sello (11.811) dado que no son exactamente el mismo.

de la imagen (la suma al cuadrado en este caso coincide por el hecho de trabajar con imágenes binarias), la cual se conoce como la energía de la imagen y se calcula con la ecuación (3.2). Este es el valor máximo que puede tomar esta correlación, ya que esta es una suma de píxeles multiplicados. Para el primer sello de la imagen 3.1 esta suma es igual a 11.811.

$$E_t = \sum_{x=0}^{M_t-1} \sum_{y=0}^{N_t-1} t^2(x,y) \tag{3.2}$$

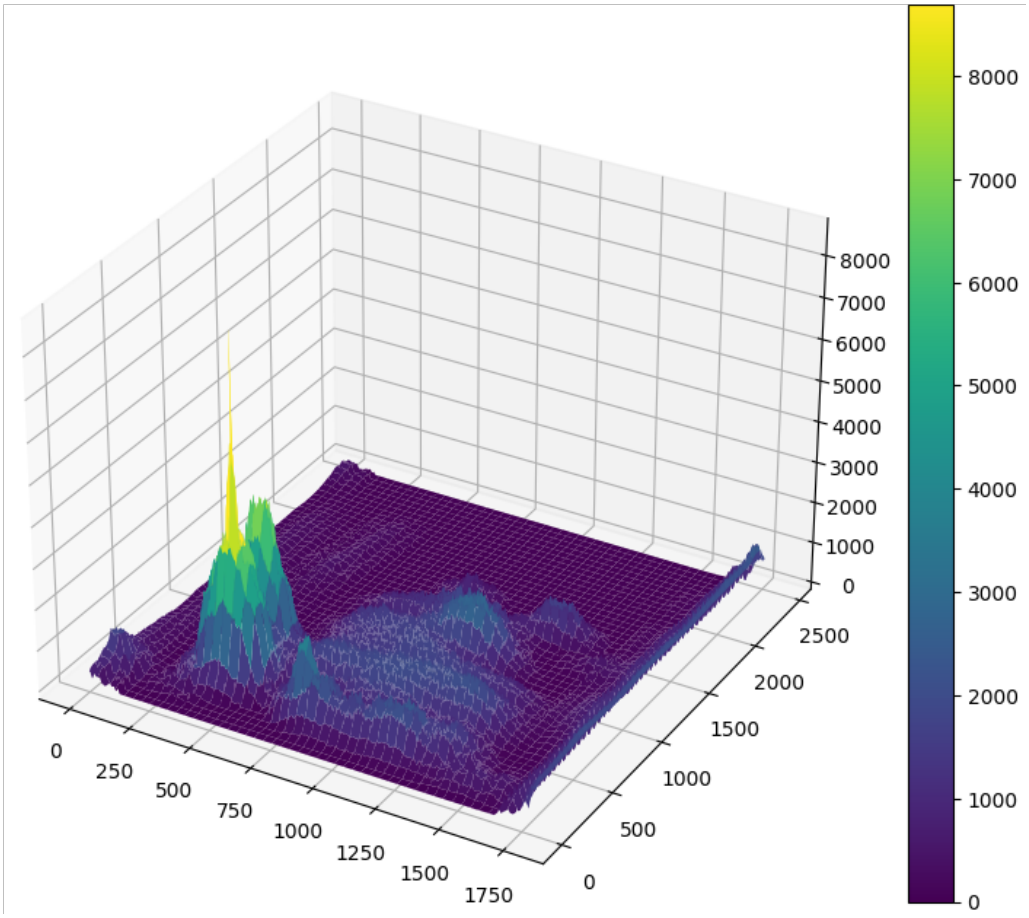


Figura 3.3: Mapa de correlación representado en tres dimensiones entre el documento y el primer sello de la figura 3.1. Se alcanza un pico de correlación de valor 8.967.

Es posible observar en la figura 3.2 el resultado de realizar este cálculo entre el documento y el primer sello de la figura 3.1. Al igual que para el caso particular mencionado, el sello es desplazado sobre toda la imagen de interés (el documento). La diferencia es que anteriormente el sello era desplazado sobre él mismo. Ahora la región de color claro que representa un alto valor de correlación coincide con la posición del sello que está presente en el documento.

Cabe destacar que, para que el cálculo de la correlación se realice sobre la totalidad de los píxeles de la imagen, la plantilla debe estar completamente solapada con la imagen de interés. Para generar cálculos de correlación sobre el borde de la imagen se asume que los píxeles que no se solapan valen cero, aportando valores nulos a la sumatoria.

Se puede observar la representación en tres dimensiones del mapa de correlación en la figura 3.3, donde la altura corresponde a los picos de correlación calculados.

A priori esta operación parece ser un primer acercamiento para medir la similitud

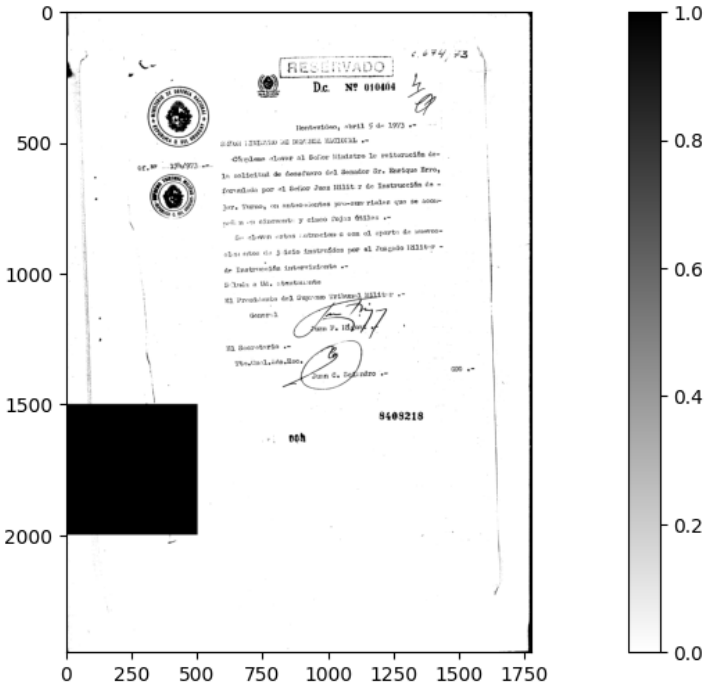


Figura 3.4: Mismo documento de la figura 3.1 con una región negra agregada (entre las filas 1500-2000 y las columnas 0-500).

entre dos imágenes. Sin embargo, presenta algunas desventajas a tener en cuenta. Idealmente se espera que el cálculo sea máximo cuando la imagen de interés es igual a la de referencia. Sin embargo, el valor más alto (la energía) también se registra al realizar la correlación con una imagen completamente negra. Esto se da porque al trabajar sobre imágenes binarias, los píxeles negros (que representan el valor 1) en la imagen de referencia coinciden con todos los píxeles negros en la de interés. A pesar de que las imágenes no sean similares, al multiplicar los píxeles correspondientes la suma será máxima, dado que la imagen completamente negra siempre aporta un factor de 1 para cada producto entre píxeles.

A modo de ejemplo, en la figura 3.4 se altera el documento anterior agregando una región totalmente negra. Esto no debería presentar cambios notorios respecto al mapa de correlación calculado anteriormente en 3.2, dado que no aporta información respecto a ningún sello. Sin embargo, tal como fue detallado, en la región negra la correlación 3.5 devuelve valores incluso más altos que en la zona donde se encuentra el sello. Este corresponde con el valor más alto y además es la energía de la imagen de referencia (11.811).

Usualmente los documentos escaneados contienen zonas predominantemente negras correspondientes a exceso de tinta, fotografías, bordes de escaneado, etc. La resolución de esta problemática es imprescindible para que los valores altos de correlación sean fieles candidatos a sellos, robusteciendo el resultado ante irregu-

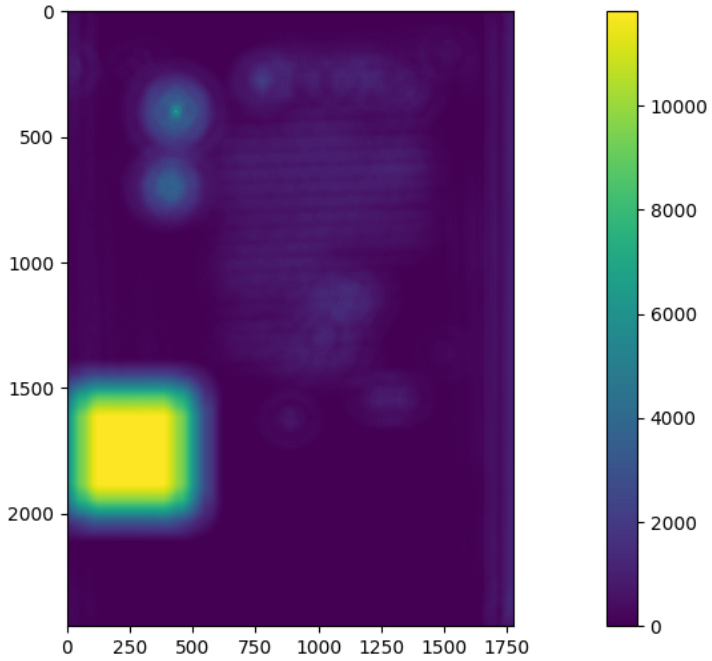


Figura 3.5: Correlación entre el documento manchado de la figura 3.4 y el primer sello de la figura 3.1. Se observa que en las zonas con alta densidad de píxeles negros se obtienen altos valores de correlación, independientemente de su similitud con la plantilla. Se observa que el valor en esta región es 11.811 (la energía del sello).

laridades.

El siguiente aspecto a tener en cuenta refiere a la energía del sello. Hasta el momento en el análisis siempre se consideró el mismo sello. Sin embargo, es fundamental reconocer que si un sello tiene más energía (en general si un sello es más grande tiene más píxeles negros y por lo tanto más energía, aunque no necesariamente es así), los valores de correlación tienden a ser mayores. Esto se debe a que aumentar la cantidad de píxeles negros, implica considerar más coordenadas durante el proceso de correlación cruzada. En consecuencia, la contribución total al valor de correlación se incrementa, conduciendo a valores de correlación más altos. En otras palabras, el rango de valores de correlación no está normalizado y varía según la energía del sello. Esto genera una dificultad para determinar si una correlación es alta o baja y no permite una comparación directa de correlaciones para diferentes imágenes de referencia.

Para evidenciar este aspecto se halla la correlación entre el documento y el segundo sello de la figura 3.1, cuya energía es 3.409.

En la figura 3.6 se observa que el máximo de la correlación es mucho menor, del orden de la energía del nuevo sello. Además, el máximo parece estar en la ubicación del sello anterior, esto se debe a que en el centro del sello original hay

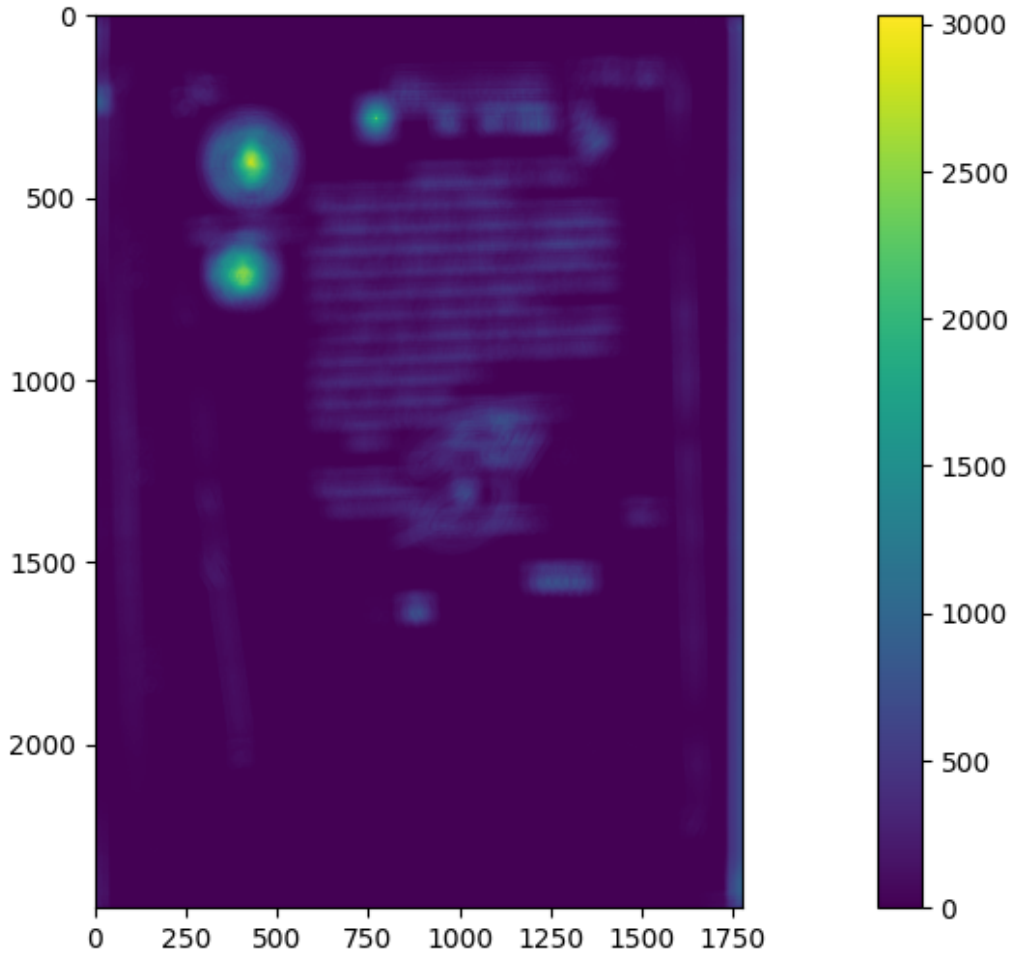
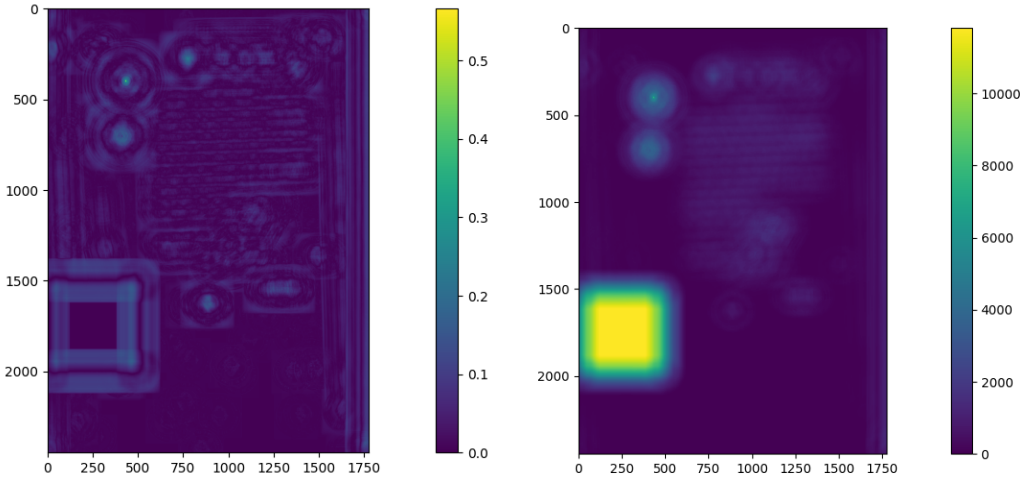


Figura 3.6: Correlación entre el segundo sello y el documento de la figura 3.1. Se visualizan tres picos de correlación en las zonas donde se ubican los tres sellos presentes en la imagen. Se registra el valor de correlación más alto con un valor de 3.409.

muchos píxeles negros. Por lo que al superponer este nuevo sello (más pequeño) en el centro del sello original, la coincidencia de píxeles negros es incluso mayor a la que hay donde se encuentra el nuevo sello. Es decir, se manifiesta nuevamente la primer desventaja planteada, sobre la energía de la imagen de interés.

3.2.2. Correlación Cruzada Normalizada (*Normalized Cross-Correlation*)

La implementación de NCC (*Normalized Cross-Correlation*) sortea las dificultades mencionadas en la sección anterior 3.2.1, normalizando el coeficiente de correlación [19]. En esta instancia se repite el cálculo pero además se divide entre la norma de la multiplicación entre las energías de ambas imágenes, obteniendo la ecuación (3.3). De está forma el rango de valores se mantiene acotado entre 0 y 1. Además, se restan las medias $\bar{f}$ y $\bar{t}$ (ver ecuaciones (3.4) y (3.5)) de las imágenes



(a) NCC (b) Correlación inicial.

Figura 3.7: Comparación entre ambas técnicas de correlación realizadas sobre el sello 3.1 y el documento alterado 3.4. El cálculo con NCC suprime correctamente los picos de correlación en la zona alterada y normaliza la escala a un rango entre 0 y 1.

penalizando aquellas de interés con energía alta. Esto soluciona el problema presentado anteriormente 3.5 sobre las regiones con mucha presencia de negro, brillo o contraste. Una observación es que la media $\bar{f}$ se calcula sobre la subregión de la imagen de interés sobre la que se desplaza el sello.

$$\gamma(u,v) = \frac{\sum_{x,y}[f(x,y) - \bar{f}_{(u,v)}][t(x-u,y-v) - \bar{t}]}{\sqrt{\left(\sum_{(x,y)}[f(x,y) - \bar{f}_{(u,v)}]^2 \sum_{(x,y)}[t(x-u,y-v) - \bar{t}]^2\right)}} \tag{3.3}$$

$$\bar{f}_{(u,v)} = \frac{1}{M_t N_t} \sum_{x=u}^{u+M_t-1} \sum_{y=v}^{v+N_t-1} f(x,y) \tag{3.4}$$

$$\bar{t} = \frac{1}{M_t N_t} \sum_{x=0}^{M_t-1} \sum_{y=0}^{N_t-1} t(x,y) \tag{3.5}$$

Al aplicar NCC al documento 3.4 que presenta la región negra, se verifica la robustez del cálculo ante los problemas anteriormente descritos. El mapa de correlación de la imagen 3.7a vuelve a indicar correctamente que la región de valor más alto es precisamente donde se ubica el sello, suprimiendo los picos de correlación en la zona alterada inicialmente. En la imagen 3.7b se contrasta la ventaja de aplicar la correlación cruzada normalizada con respecto al cálculo inicial (3.1).

Hasta el momento la intención fue robustecer el cálculo para que sea lo más acertado posible. Sin embargo, para que resulte el más apropiado y sea capaz de computar grandes cantidades de datos, es esencial optimizar su velocidad de ejecución. La ecuación (3.3) resulta computacionalmente costosa debido a que el denominador, que normaliza el coeficiente de correlación, está siendo recalculado para cada punto de la imagen, aumentando significativamente la cantidad de operaciones.

3.2.3. Correlación Cruzada Normalizada Rápida (*Fast Normalized Cross-Correlation*)

Para abordar la problemática presentada anteriormente se utilizan las técnicas descritas en [19] y [20]. Se propone un método eficiente para calcular los coeficientes de correlación que disminuyen notoriamente la cantidad de operaciones. Este método resulta en la ecuación (3.6), donde M_t y N_t corresponden a las dimensiones de la imagen de referencia t.

$$\gamma(u,v) = \frac{\sum_{x,y} f(x,y)[t(x-u,y-v) - \bar{t}]}{\sqrt{\left[\sum_{x,y} f^2(x,y) - \frac{1}{Mt.Nt}\left(\sum_{x,y} f(x,y)\right)^2\right] \sum_{x,y}[t(x-u,y-v) - \bar{t}]^2}} \tag{3.6}$$

Para el procesamiento de este cálculo se utiliza la librería de Python: `Scipy`. Esta permite calcular la correlación mediante la Trasformada Rápida de Fourier (FFT por su nombre en inglés *Fast Fourier Transform*), ofreciendo beneficios en términos de eficiencia computacional y velocidad de cálculo. La implementación de este cálculo se detalla en el algoritmo 1.

Además, $\bar{f}_{(u,v)}$ se puede calcular como la correlación entre la imagen de interés y una plantilla de unos de igual tamaño que el sello, evaluada en (u, v). Es decir, sea $z(x, y)$ una imagen de unos de tamaño $M_t \times N_t$:

$$\bar{f}_{(u,v)} = C_{zf}(u,v) = (z \otimes f)(u,v) \tag{3.7}$$

Y de manera similar:

$$\bar{f^2}_{(u,v)} = C_{zf^2}(u,v) = (z \otimes f^2)(u,v) \tag{3.8}$$

Esto se debe a que calcular la correlación con una plantilla de unos equivale a sumar valores de la imagen para cada parche de tamaño $M_t \times N_t$. A su vez, presenta la ventaja de que se puede hallar mediante la Transformada de Fourier, acelerando la operación.

Algorithm 1: Fast Normalized Cross-Correlation

Input:
- `f`: imagen de entrada (feature)
- `t`: sello (template)

Output:
- `y`: array de coeficientes de FNCC

1 **Function** `FNCC(`f, t`)`
2 $M, N \leftarrow$ Dimensión de f;
3 $M_t, N_t \leftarrow$ Dimensión de t;
4 $M_{t2} \leftarrow M_t/2$;
5 $N_{t2} \leftarrow N_t/2$;
6 $f_{mean} \leftarrow$ `dsp.correlate`$(f,$ `np.ones`$((M_t, N_t)))/(M_t \cdot N_t)$;
7 $t_{centered} \leftarrow t -$ `np.mean`(t);
8 $||t_{centered}||^2 \leftarrow$ `np.linalg.norm`$(t_{centered})$;
9 $nf2 \leftarrow$ `dsp.correlate`$(f^2,$ `np.ones`$((M_t, N_t)))$;
10 $nfc \leftarrow \sqrt{\texttt{max}(0, nf2 - M_t \cdot N_t \cdot f_{mean}^2)}$;
11 $y \leftarrow$ `dsp.correlate`$(f, t_{centered})/(nfc \cdot ||t_{centered}||^2)$;
12 **return** y

Respecto al pseudocódigo se mencionan algunas aclaraciones. La imagen de interés f es agrandada para poder calcular la correlación en los bordes sin afectar el tamaño original. Esto es debido a que la función para calcular la correlación (`dsp.correlate` [1]) se utiliza en modo `'valid'`, es decir que el calculo se realiza siempre y cuando la plantilla se encuentre totalmente solapada con la imagen de interés, como se detallaba anteriormente.

La línea 6 es una forma rápida de calcular $\bar{f}$, debido a que se calcula para toda la imagen y se utiliza la Transformada de Fourier. Cada coordenada de f_{mean} será la norma de la región por la que se desplaza el sello (por eso se correlaciona con una matriz de unos de tamaño igual que el sello).

Si bien el pseudocódigo no lo indica, en la línea 10 se añade un término de *offset* $(1 \times e^{-5})$ ya que eventualmente se encuentran regiones de la imagen de interés completamente blancas, evitando de esta manera divisiones entre cero.

[1] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.correlate.html`

Figura 3.8: Ejemplos de la variabilidad entre sellos de una misma clase: Juzgado Militar. Se observan diferencias de tamaño, cantidad de tinta, diseños de las figuras y estilos de imprenta.

3.3. Variabilidad y diversidad de sellos: desafíos y estrategias

Por naturaleza, las imágenes presentan una serie de deformaciones producto de varios factores como el deterioro de la documentación a lo largo del tiempo, la calidad y cantidad de la tinta de los sellos, su posición en la imagen, colores y diversas fuentes de escritura. Es importante también tener en consideración que los sellos utilizados pueden aparecer deformados de manera no rígida, debido al material (goma) con el que fueron marcados. Otro aspecto a destacar es que los sellos presentes en las imágenes no son exactamente iguales entre sí, ya que el diseño y el texto puede variar sensiblemente. Producto de estas diferencias es que la búsqueda de una imagen idéntica se convierte en una tarea imposible, ya que no existen dos imágenes cuyos píxeles coincidan completamente.

En la figura 3.8 se observan algunas de las características mencionadas. En primer lugar, es claro notar la variación de la cantidad de tinta en cada imagen, reflejada en la densidad de píxeles negros. A su vez, la variabilidad de tamaño y diseño de las figuras dificulta aún más la detección de patrones. Por último, es posible percibir una sensible rotación en el tercer sello de la imagen.

Otros aspectos que también se encuentran habitualmente son la presencia de texto y rayas superpuestas.

Entonces, existiendo tanta variedad entre sellos de una misma clase, ¿cuál es la mejor forma de elegir un representante de cada clase que tome en cuenta todas las irregularidades?

Utilizar directamente un sello por clase como plantilla puede resultar poco eficiente en términos de desempeño y tiempo de procesamiento. Por un lado, el cálculo NCC aplicado sobre el par documento-sello se repite para cada sello elegido como plantilla. Puesto que se utiliza un sello por clase (ver figura 2.2), el cálculo es procesado tantas veces como clases de sellos se elijan. A su vez, por las características mencionadas anteriormente, un único representante por clase resulta

Figura 3.9: Ejemplos de selloides generados como la suma de sellos de las distintas clases de sellos a detectar.

muy específico para contemplar todas las posibles irregularidades, restringiendo así la generalización del detector.

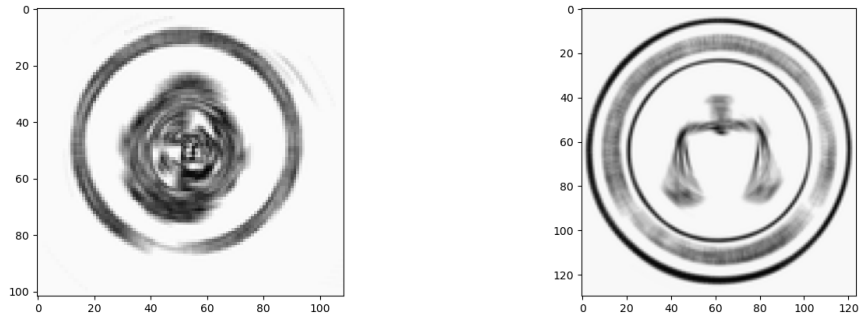
Por esta razón se propone aplicar transformaciones geométricas sobre las plantillas, con el fin de robustecer su capacidad de contemplar las irregularidades de todos los sellos de una misma clase. De esta forma, es posible construir un algoritmo eficaz y con alto grado de generalización.

3.3.1. *Selloides*

La primera estrategia de transformación consiste en suplantar las plantillas seleccionadas a partir de la generación de un único selloide. Se entiende por el concepto selloide como una fusión entre varios tipos de sellos con distintas características. Una posibilidad es sumar varios sellos de distintas clases superpuestos entre sí, como se ilustra en la figura 3.9. Luego, a partir de la creación del mismo, se calcula NCC entre una única plantilla (el selloide) y los documentos.

Principalmente con esta variación interesa profundizar en dos aspectos. Por un lado, reducir el tiempo de procesamiento, ya que menor cantidad de plantillas implica menores cálculos de NCC por documento. Por otra parte, generar una única plantilla capaz de contemplar las irregularidades y todas las clases determinadas, lo que aumenta la capacidad de generalización del algoritmo.

Una observación importante es que para el caso de extracción original (utilizando una única muestra por clase como plantilla, sin manipularlas) se obtiene a lo sumo una zona candidata por cada cálculo FNCC. Es decir, dado un sello y un documento, se halla la FNCC y si el máximo supera un umbral se guarda la zona candidata. Se implementa de esta manera porque se asume que un mismo sello no aparece dos veces en un documento y en caso de aparecer más de una vez, una



(a) Clase: Inteligencia. (b) Clase: Juzgado.

Figura 3.10: Selloides generados a partir de la suma de varias rotaciones. A partir de un sello de generan 5 versiones rotadas, a -5, -2.5, 0, 2.5 y 5 grados y luego se suman.

detección alcanza para el objetivo del proyecto.

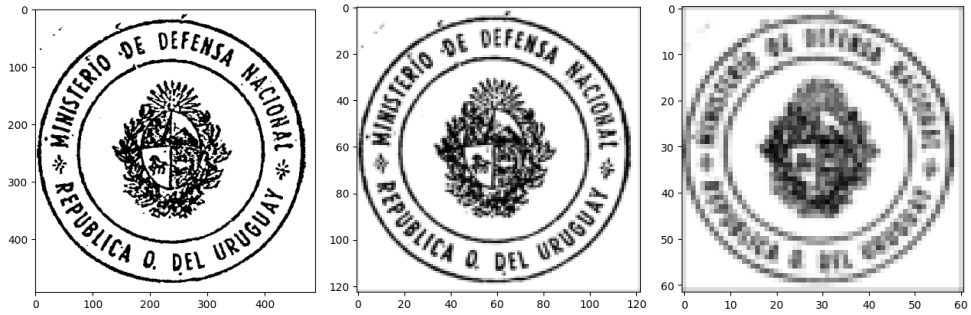
Para los selloides este proceso cambia, porque eventualmente se debe guardar más de una zona (si hay más de un sello presente). Entonces el desafío a resolver consiste en detectar varios sellos en un mismo documento con una única plantilla. Para esto, una vez calculada la FNCC entre un documento y el selloide se aborda el siguiente algoritmo:

1. Si el máximo del mapa de correlaciones es mayor al umbral establecido, se guarda el valor de la coordenada.
2. Se establece una zona candidata alrededor del máximo encontrado.
3. Se modifica el mapa de correlaciones reduciendo a cero toda la zona candidata, incluido el centro, para evitar detectar picos de correlación cercanos (correspondientes al mismo candidato).
4. Se calcula un nuevo máximo y se vuelve al primer paso del algoritmo, finalizando cuando el nuevo máximo ya no supere el umbral establecido.

3.3.2. Rotaciones

Otra transformación posible consiste en sumar varias muestras rotadas de un mismo sello, de esta forma el objetivo es contemplar todas las pequeñas rotaciones existentes producto de la posición en la que se grabó el sello.

Para generar esta variante, se parte del sello original y se rota utilizando la librería `skimage`. De forma sucesiva, se vuelve a aplicar la misma rotación sobre la imagen ya transformada. Finalmente se suman todas las transformaciones, obteniendo resultados como los de la figura 3.10.



(a) Escala 1. (b) Escala 1/4. (c) Escala 1/8.

Figura 3.11: Ejemplos del mismo sello escalado x1, x4 y x8.

3.4. Estrategias de acelerado: multi-escalados

El *multiscreening* o multiescalado es una técnica comúnmente utilizada en procesamiento de imágenes que implica analizar una imagen en múltiples resoluciones diferentes para mejorar la detección de objetos. Esto se logra examinando la imagen original y versiones escaladas de la misma, lo que permite detectar objetos de diferentes tamaños y aspectos.

En el algoritmo de la etapa 1, el tiempo de procesamiento se reduce considerablemente al escalar la imagen ya que la cantidad de operaciones es proporcional a la cantidad de píxeles. Dado que el tamaño del mapa de correlación es proporcional al tamaño de la imagen, reducir el tamaño de la imagen mediante el escalado reduce el número de operaciones requeridas para calcular la correlación. Esto resulta en una disminución significativa en el tiempo de procesamiento, lo que hace que el algoritmo sea más rápido en la detección de sellos.

Junto con esta técnica se presenta otra ventaja: disminución de ruido. Al escanear documentos, la alta resolución captura incluso los detalles más pequeños como la rugosidad de la hoja, el polvo, las manchas y los márgenes del escáner. Sin embargo, al variar el escalado de la imagen, estas imperfecciones se diluyen y su impacto se reduce significativamente, lo que contribuye a eliminar píxeles que no aportan información relevante para la detección de los sellos.

Utilizando la función `reescale` de la librería `skimage.transform` [2] se obtuvieron resultados para varias escalas: 1, 2, 4, 8 y 16. Un escalado de 1 significa que la imagen no es reducida en absoluto, mientras que en una escala de 4 la imagen reduce 1/16 su cantidad de píxeles, además se utiliza anti aliasing. Cabe aclarar que el análisis de las secciones anteriores asume imágenes binarias pero al escalar se interpola y se pasa a escala de grises, el análisis es análogo para este tipo de imágenes.

[2] `https://scikit-image.org/docs/stable/api/skimage.transform.html`

En la figura 3.11 se detalla cómo varía la composición de la imagen al escalarla por distintos factores. En primer lugar, el cambio de tamaño es inherente al escalado. El sello original de la figura (correspondiente a escalado 1) presenta 493 × 488 píxeles de alto y ancho respectivamente. Por otro lado, un factor 1/4 de escalado reduce las dimensiones a 123 × 122, siendo la cuarta parte del tamaño original, pudiendo identificar casi sin esfuerzo los mismos detalles que en la imagen original. Por último, se aplica una reducción a la octava parte. Aún en estas condiciones es posible identificar las características más generales del sello, como su forma y la presencia de alguna figura o texto, pero los detalles comienzan a perderse.

3.5. Implementación del algoritmo

La etapa 1 establecida para el presente trabajo consiste en detectar de forma automática sellos en un conjunto de documentos escaneados. El algoritmo a implementar debe realizar una lectura rápida de todos los documentos y generar como salida un conjunto de sellos extraídos de los mismos. Posteriormente, se utilizará esta información extraída para generar una base de datos que agrupe los sellos según sus respectivas clases. Esta base de datos es imprescindible para el abordaje de la siguiente etapa, que será objeto de entrenamiento y verificación del modelo de clasificación a desarrollar.

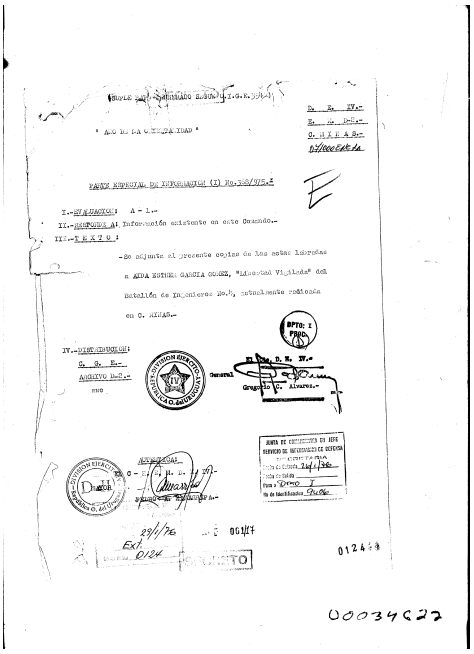
En la figura 3.12 se muestra un ejemplo de la secuencia a realizar con el algoritmo. Primero se lee una imagen de un documento escaneado y se calcula el mapa de correlación, respecto de las plantillas previamente elegidas. Aquellos picos mayores a cierto umbral definido, son recuadrados de color rojo sobre el documento y extraidos como candidatos a sellos. Luego, si se cuenta con etiquetas (rectángulos azules), se generan métricas de desempeño a partir de las distancias entre las detecciones y las etiquetas.

3.5.1. Estructura de directorios

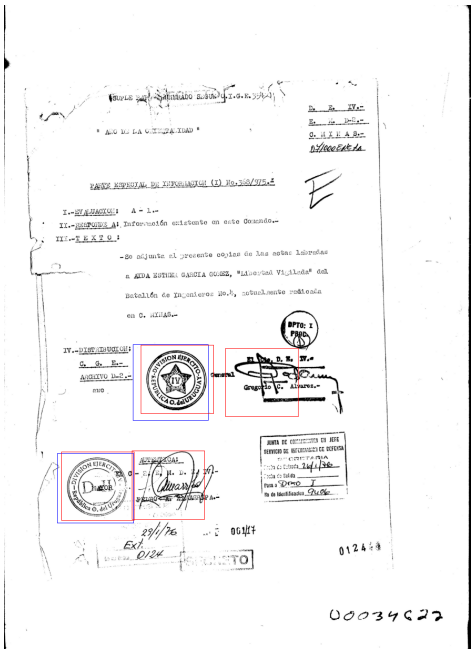
La cantidad y el tamaño de las imágenes que componen el conjunto de datos hacen que sea esencial tener una estructura de directorios organizada que facilite la lectura optimizada de los datos. Sin una estructura de directorio adecuada, el acceso a las imágenes y la manipulación de los datos pueden volverse poco eficientes, lo que resulta en tiempos de ejecución más largos.

La sección de directorios y archivos relevantes incluye:

- `listas`. Conjunto de listas con las rutas a los archivos en formato `JSON` que se corresponden con cada documento. Se utiliza para generar el conjunto de control.
- `conjunto_control`. Almacena el archivo llamado `gt.txt` que contiene las coordenadas de los centros extraídos por estudiantes de la FIC con *Label*

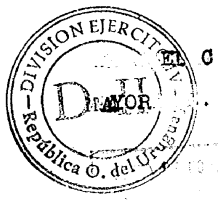


(a) Paso 1: Imagen de interés (documento) sobre la cual se buscan sellos.



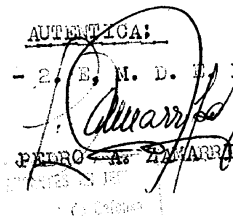
(b) Paso 2: Documento con las etiquetas (rectángulos azules) y las cuatro detecciones del algoritmo (rectángulos rojos).

(c) Primera extracción con pico de correlación 0.39.



(d) Segunda extracción con pico de correlación 0.28.

(e) Tercera extracción con pico de correlación 0.18.



(f) Cuarta extracción con pico de correlación 0.17.

Figura 3.12: Proceso de detección y extracción.

Me.

- `home`. Contiene las imágenes de los documentos sobre los que se detectan sellos.
- `sellos_a_detectar`. Contiene los sellos a utilizar como plantillas de referencia.
- `salidas_NCC`. Almacena todos los resultados obtenidos.

En la sección del apéndice A se describe con mayor detalle la estructuración del directorio.

3.5.2. Desarrollo

Una vez organizado el directorio, se inicia el proceso de codificación del desarrollo utilizando el lenguaje de programación `Python` para la detección de sellos en documentos. Es elegido `Python` por su amplia disponibilidad de bibliotecas y herramientas especializadas que facilitan la implementación para este fin.

El algoritmo se enfoca principalmente en el cálculo FNCC para cada par documento (x) - sello (s) contenidos en los directorios `home` y `sellos_a_detectar` respectivamente. A partir de un mapa de correlación calculado, se almacenan las coordenadas correspondientes a aquellos valores que superen cierto umbral a establecer. Las coordenadas son almacenadas en el archivo de texto correspondiente y representan las zonas de posibles candidatos a sellos.

Además, al contar con datos etiquetados, es posible obtener resultados comparando las detecciones generadas con las etiquetas de los sellos. Si el algoritmo es ejecutado en producción (donde no se cuenta con etiquetas) se pueden extraer los sellos detectados a partir de las coordenadas de los máximos y el tamaño del sello correspondiente.

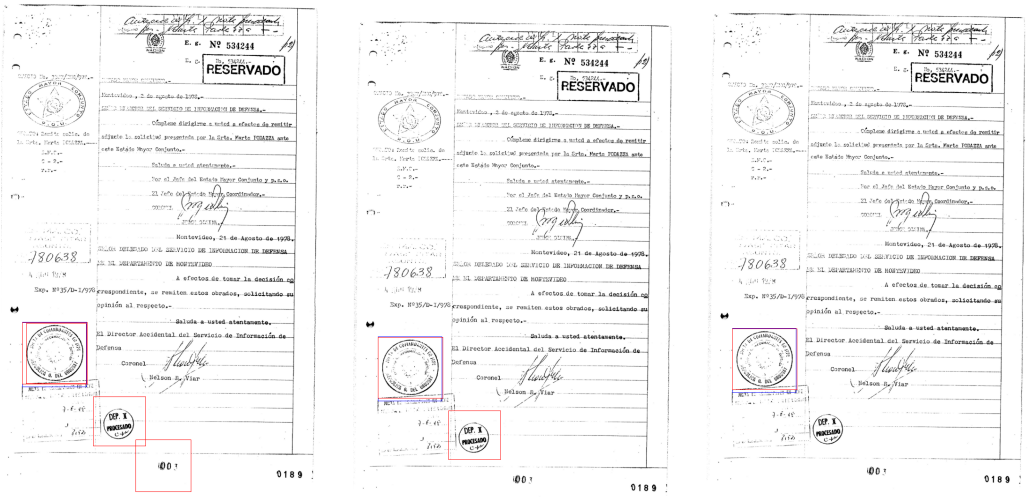
3.5.3. Umbrales de detección

Es necesario establecer un parámetro que determine cuándo una coordenada presenta un valor de correlación lo suficientemente alto para ser considerada centro de un candidato a sello. Este es el umbral mencionado anteriormente y corresponde a un valor entre 0 y 1, por ser éste el rango en el que varía la correlación.

Analizando casos límites, un valor de umbral pequeño detecta numerosos picos derivando en una detección demasiado permisiva. Esto puede generar un exceso de detecciones que no son sellos (también denominados falsos positivos). Por el contrario, cuanto más cercano sea el umbral a 1, se genera el efecto inverso: el pico de correlación debe ser mayor, resultando en un umbral más restrictivo. Al detectar pocos sellos se omiten aquellos con picos de correlación no tan altos. Mediante ensayo y error, se determinan valores que permitan una detección óptima y precisa de los sellos. Este análisis se encuentra más adelante en la sección 3.8, junto con los resultados obtenidos. Es de particular interés capturar la mayor cantidad de sellos, esto implica un umbral bajo donde la tasa de recall sea alta aunque la precisión no sea buena.

En las figuras 3.13 se visualizan ejemplos donde una misma imagen es procesada con distintos niveles de umbrales de detección. Un umbral óptimo genera un recuadro de color rojo en la misma zona en la que hay un recuadro azul. El color rojo indica las detecciones realizadas de forma automática por el algoritmo, mientras que el azul hace referencia a la base de datos generada con la herramienta *Label Me*.

Cabe destacar que no hay un único valor correcto para determinar el umbral, ya

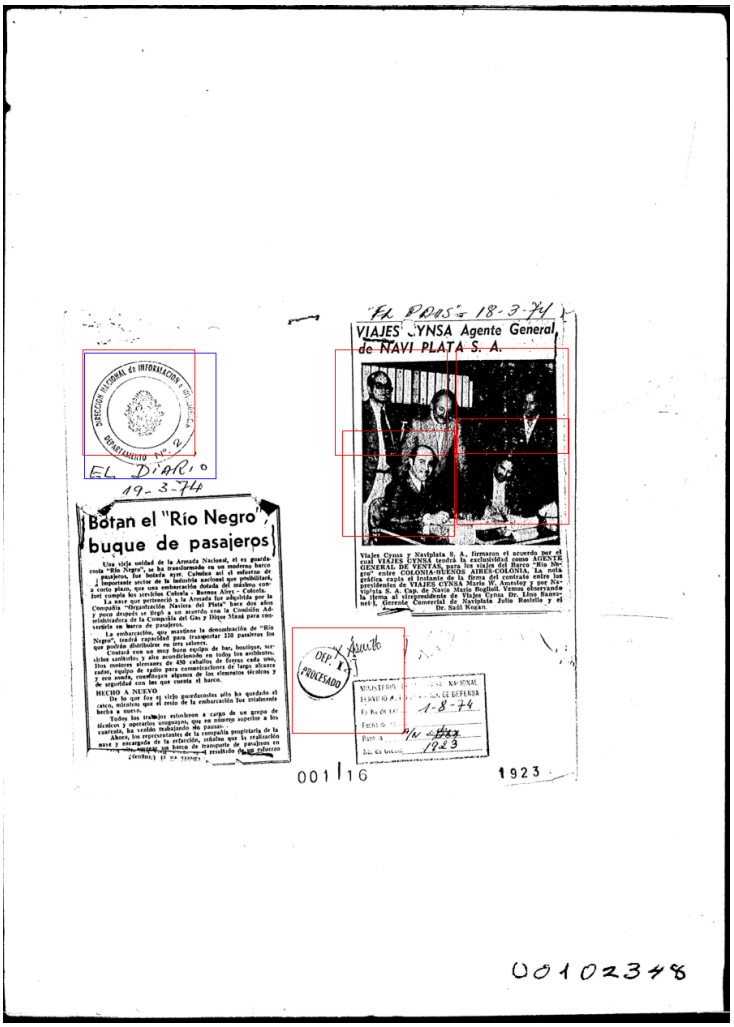


(a) Umbral bajo (b) Umbral medio (c) Umbral alto

Figura 3.13: Imágenes generadas con el algoritmo implementado. En rojo se ven las zonas candidatas detectadas y en azul el *ground truth* de referencia (*Label Me*). Se observa que para umbrales bajos las detecciones son menos precisas (la detección de más abajo desaparece para los umbrales más grandes) pero se encuentran más sellos comparado con el umbral más alto (la detección del sello chico desaparece para el umbral alto).

que existe un compromiso entre detectar lo que se desea y minimizar la cantidad de falsos positivos. Sin embargo, la prioridad radica en detectar todos los sellos posibles, a pesar de que la tasa de falsos positivos sea alta. A pesar de esto, si se quiere que las detecciones sean directamente extraídas para ser utilizadas como entradas a un clasificador (unificando etapa 1 y etapa 2) es imprescindible profundizar en la limpieza de falsos positivos para no enlentecer el algoritmo con imágenes que no son de interés.

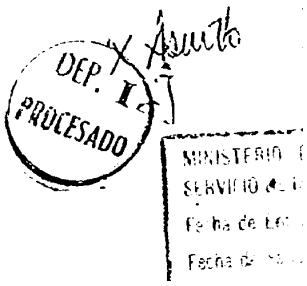
El primer filtro implementado para reducir la cantidad de falsos positivos detectados (llamados basura) fue excluir las detecciones cuyo porcentaje de píxeles blancos (o negros) sea muy alto, imágenes con porcentaje de píxeles negros por debajo del 5 % son muy blancas y por encima del 95 % son muy negras, por lo que prácticamente no se encuentran sellos en éstas. Se toma como hipótesis que ningún sello tendrá una proporción tan alta de píxeles blancos o negros. Con esto es posible filtrar una gran cantidad de falsos positivos, como los que se indican en la figura 3.14. Esto es a excepción de aquellos falsos positivos cuya proporción de píxeles blanco/negro sea similar a la de los verdaderos sellos no es detectada por el filtro, lo cual anticipa que será necesario continuar con la profundización sobre dicha limpieza.



(a) Documento con las etiquetas (En azul) y las detecciones del algoritmo (En rojo).

(b) Primera extracción.

(c) Segunda extracción



(d) Tercera extracción

Figura 3.14: Detección y extracciones luego del filtro que elimina detecciones con alto porcentaje de píxeles negros. Dentro del documento se observa una imagen, dentro de esta hay cuatro falsos positivos de los cuales uno solo (b) logra pasar el filtro de porcentaje de negro.

3.6. Análisis de operaciones

En esta sección se evalúa el costo computacional de las distintas funciones del algoritmo. Dado que el tiempo computacional depende de la capacidad de procesamiento de los equipos en los que se ejecutan los programas, no se hace referencia a tiempos absolutos sino a órdenes de operaciones.

Para el siguiente análisis, D será la cantidad de documentos y S la cantidad de sellos utilizados como plantilla.

3.6.1. Escalas

La primer variable a tener en cuenta para analizar el tiempo computacional es el escalado a utilizar. Dado que el algoritmo calcula la FNCC entre dos imágenes, el tamaño de estas en un factor relevante del costo computacional que cuanto mayor es el escalado, menor peso tiene la FNCC.

El costo computacional no solo está determinado por la complejidad del cálculo de la FNCC, sino que también se ve afectado por otras operaciones realizadas en cada iteración del algoritmo. Estas operaciones incluyen la lectura y escritura de archivos, así como el procesamiento de imágenes y gráficas, aunque esta última es menos relevante ya que generalmente se realiza una sola vez durante la ejecución del algoritmo.

3.6.2. Métodos de detección

Solo sellos

Para el caso en que solo se utilizan sellos como plantillas (como los de la figura 2.2), el programa carga los mismos en memoria RAM y luego itera sobre los documentos, hallando en cada iteración la FNCC y escribiendo en los archivos los resultados correspondientes. De esta forma, la cantidad de veces que se calcula la FNCC es igual a la cantidad de sellos multiplicada por la cantidad de documentos.

$$\#FNCC = D \cdot S$$

Solo selloide rotado

Para el caso en que se utilizan los selloides rotados como los de la figura 3.10, al comienzo del algoritmo se requiere procesamiento de imágenes para rotar y combinar (hay un selloide por cada sello), pero luego el algoritmo continúa igual que con los sellos, se cargan en RAM y se itera sobre los documentos. De esta forma la cantidad de veces que se calcula la FNCC es la misma que para los sellos.

$$\#FNCC = D \cdot S$$

Solo selloide sumado

Para el selloide sumado (ver ejemplos de la figura 3.9), al principio del algoritmo se requiere cierto procesamiento para generar el selloide (se considera un único selloide en este análisis, pero cabe la posibilidad de que sean más). Luego carga este en memoria RAM y se itera sobre los documentos, hallando una única vez la FNCC para cada documento. Así es que la cantidad de veces que se calcula la FNCC es igual a la cantidad de documentos.

$$\#FNCC = D$$

Varias muestras por sello

Si se tienen por ejemplo M muestras de cada sello, se puede pensar que se tienen $M.S$ sellos distintos a utilizar como plantillas, de esta forma la cantidad de veces que se calcula la FNCC sería igual que el primer caso de solo sellos pero multiplicado por la cantidad de muestras M por cada sello.

$$\#FNCC = D \cdot S \cdot M$$

Métodos combinados

Es evidente notar que si se ejecuta el programa para una combinación de los casos anteriores (por ejemplo: para sellos y el selloide generado por sumas), la cantidad de cálculos de la FNCC será la suma de los necesarios para cada caso individual ($\#FNCC = D \cdot (S + 1)$ para sellos y selloide sumado).

3.7. Validación: datos de entrenamiento

Para evaluar el proceso de detección se trabaja con dos bases de datos diferentes a modo de comparar el desempeño del modelo. Previo a definir la composición de ambas bases, es necesario definir el concepto de *ground-truth* (verdad de referencia).

Se entiende por *ground-truth* un conjunto de datos que se considera lo suficientemente confiable. En el contexto de visión por computadora, se refiere a los datos etiquetados que se consideran correctos, sirviendo como punto de comparación para evaluar un algoritmo. El desempeño del algoritmo se evalúa comparando sus predicciones con esta base de referencia. A su vez, desviaciones respecto del *ground-truth* pueden indicar áreas en las que el algoritmo necesita mejorar o puede destacar dificultades específicas en la tarea de procesamiento. Por tanto, además de ser utilizado para la evaluación del modelo, el *ground-truth* es fundamental para la fase de entrenamiento.

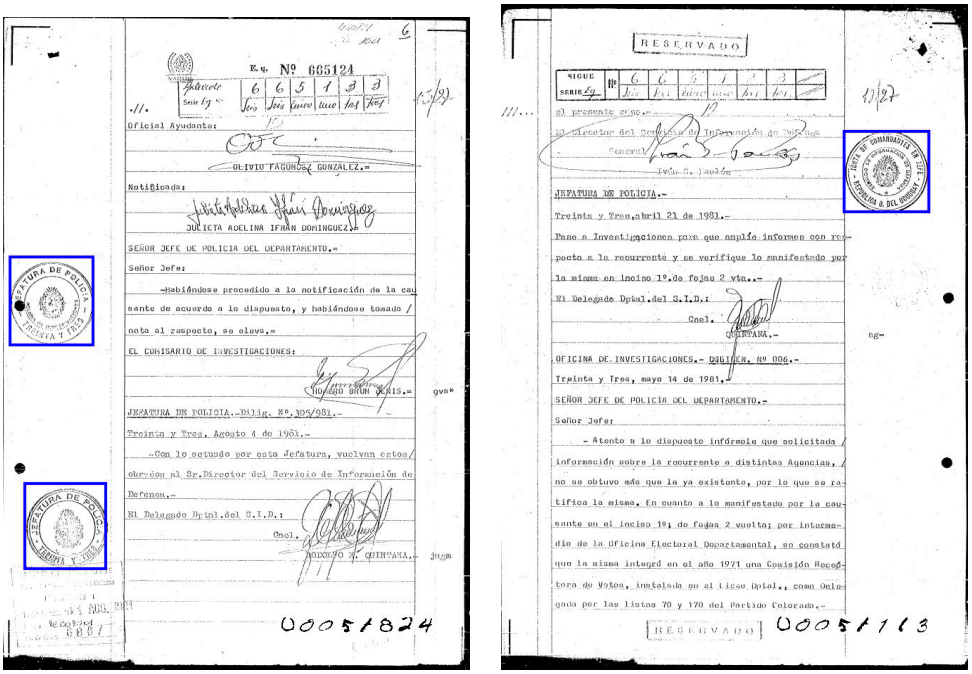


Figura 3.15: Ejemplos de documentos contenidos en el *ground-truth* de referencia. En azul se recuadran los sellos marcados por estudiantes mediante la aplicación de (*Label Me*). Notar que el sello pequeño que se encuentra en la zona superior de la primera imagen no fue marcado.

Base inicial

Inicialmente se cuenta con una base de datos que contiene detecciones generadas manualmente por estudiantes de la Facultad de Información y Comunicación (FIC). Esta base fue creada mediante el uso de la aplicación *Label Me*: una herramienta de anotación de imágenes que permite a los usuarios etiquetar y anotar objetos contenidos en imágenes. Consiste en una interfaz de usuario simple en la que los usuarios pueden cargar imágenes, dibujar regiones y asociar etiquetas con esas regiones. En la figura 3.15 se exponen algunos de los documentos marcados por estudiantes.

Las anotaciones son exportadas en formato `JSON` para su posterior procesamiento. Estos archivos `JSON` contienen información tal como el nombre, rollo y año correspondientes al documento, la ubicación de los sellos presentes, y otra información que no es relevante para la detección de sellos.

Sin embargo, es fundamental destacar que esta base escogida como *ground-truth* no es perfecta. La marca de los objetos en las imágenes no siempre está señalada de forma precisa; algunas de estas marcas no están centradas alrededor del sello que representan y, en algunos casos, hay sellos que no están marcados. A modo de ejemplo, en la figura 3.16 se ilustran dos casos que suelen ocurrir con frecuencia.

En el primer documento, se omite la detección de un sello que debería estar remar-

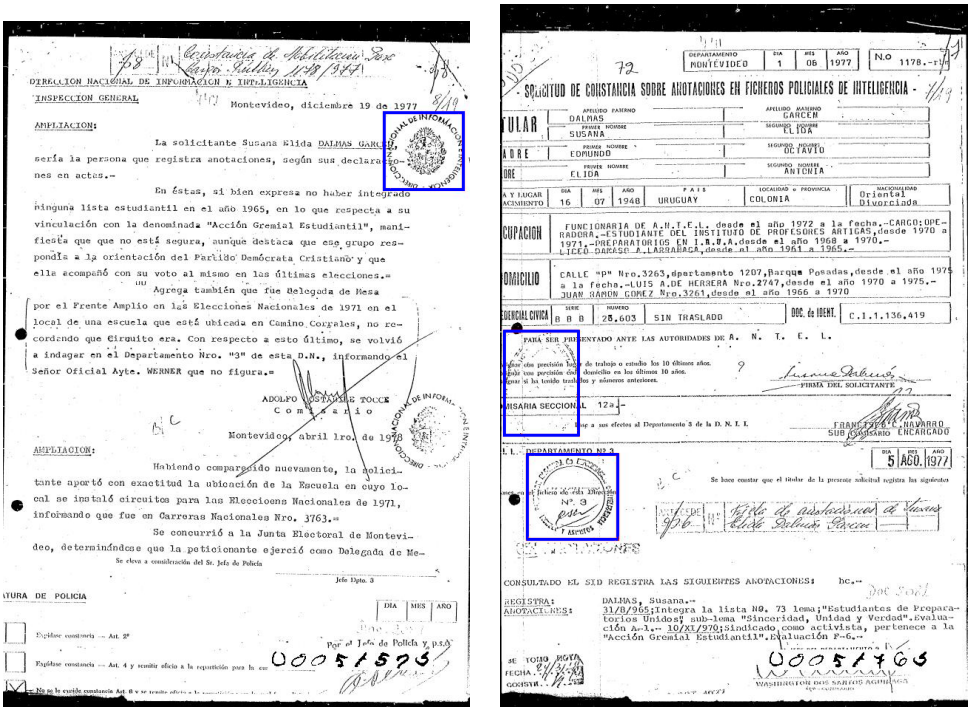


Figura 3.16: Ejemplos de documentos contenidos en el *ground-truth* de referencia. En azul se recuadran los sellos marcados por estudiantes mediante la aplicación de *Label Me*. En la primera imagen se ve un ejemplo de un sello que no fue recuadrado. En la segunda imagen se recuadra un sello que difícilmente pueda ser apreciado incluso por un humano, cabe aclarar que a pesar de éste último tener una forma ovalada y no circular, se busca detectar estos sellos también.

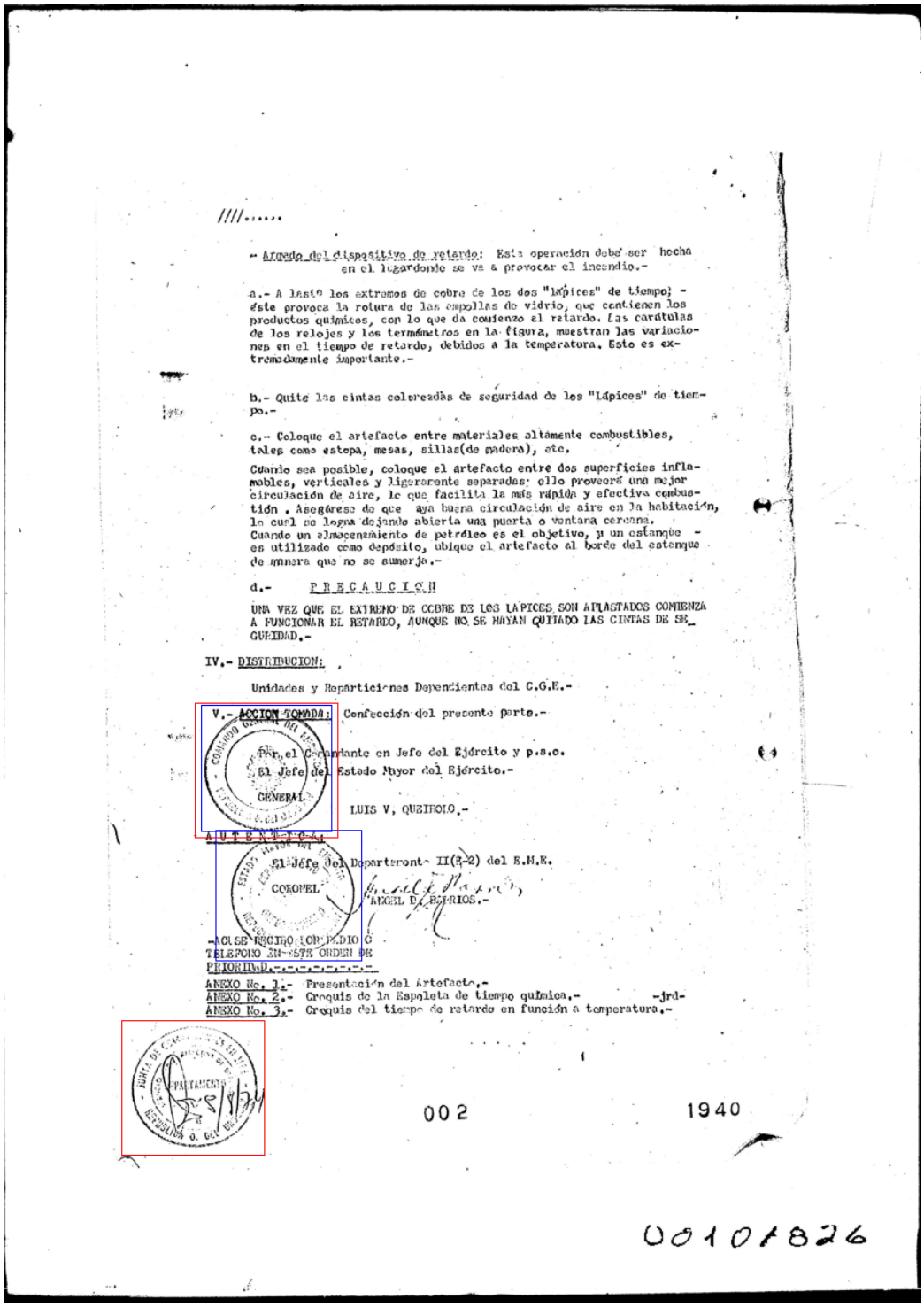
cado. Mientras que en el siguiente, se recuadra un sello irreconocible incluso para el ojo humano. Esto proporciona una indicación clara de las limitaciones a tener en consideración al momento de interpretar los resultados y evaluar el rendimiento.

Un caso práctico de lo mencionado se puede observar en la figura 3.17, donde se presentan tres situaciones: la presencia de un verdadero positivo, un falso positivo y un falso negativo. Si bien el algoritmo logró detectar correctamente dos sellos, dado que uno no está etiquetado, no será contemplado en la métrica. A su vez, no se logró detectar un sello que sí fue etiquetado, a pesar de la mala calidad del mismo. Aquí se explica por qué los resultados presentados reflejan una tendencia en lugar de proporcionar una representación precisa de la realidad.

Base sintética

A raíz de las imperfecciones encontradas en la base anterior, se propone generar muestras sintéticas que estén exentas de errores y puedan conformar una base confiable para conocer con mayor exactitud cuál es el desempeño del modelo.

Esta base está compuesta por documentos en blanco (podría ser también docu-



////.....

- Armado del dispositivo de retardo: Esta operación debe ser hecha en el lugardonde se va a provocar el incendio.-

a.- A lasto los extremos de cobre de los dos "lápices" de tiempo; - éste provoca la rotura de las ampollas de vidrio, que contienen los productos químicos, con lo que da comienzo el retardo. Las carátulas de los relojes y los termómetros en la figura, muestran las variaciones en el tiempo de retardo, debidos a la temperatura. Esto es extremadamente importante.-

b.- Quite las cintas coloreadas de seguridad de los "Lápices" de tiempo.-

c.- Coloque el artefacto entre materiales altamente combustibles, tales como estopa, mesas, sillas(de madera), etc.

Cuando sea posible, coloque el artefacto entre dos superficies inflamables, verticales y ligeramente separadas; ello proveerá una mejor circulación de aire, lo que facilita la más rápida y efectiva combustión , Asegúrese de que aya buena circulación de aire en la habitación, lo cual se logra dejando abierta una puerta o ventana cercana. Cuando un almacenamiento de petróleo es el objetivo, y un estanque - es utilizado como depósito, ubique el artefacto al borde del estanque de manera que no se sumerja.-

d.- P R E C A U C I O N

UNA VEZ QUE EL EXTREMO DE COBRE DE LOS LAPICES SON APLASTADOS COMIENZA A FUNCIONAR EL RETARDO, AUNQUE NO SE HAYAN QUITADO LAS CINTAS DE SEGURIDAD.-

IV.- DISTRIBUCION:

Unidades y Reparticiones Dependientes del C.G.E.-

V.- ACCION TOMADA: Confección del presente parte.-

Por el Comandante en Jefe del Ejército y p.s.o.
El Jefe del Estado Mayor del Ejército.-

GENERAL

LUIS V. QUEIROLO.-

A U T E N T I C O:

El Jefe del Departamento II(2-2) del E.M.E.

CORONEL

ANGEL D. RIOS.-

ACUSE RECIBO POR MEDIO DE TELEFONO SIN-ESTE ORDEN DE PRIORIDAD.-

ANEXO No. 1.- Presentación del Artefacto.-
ANEXO No. 2.- Croquis de la Espoleta de tiempo química.- -jrd-
ANEXO No. 3.- Croquis del tiempo de retardo en función a temperatura.-

002

1940

00101826

Figura 3.17: Ejemplo de un caso particular que refleja la problemática de una base que no está correctamente etiquetada. En rojo se observan las detecciones del algoritmo y en azul las etiquetas generadas con *Label Me*. En este caso se cuenta un verdadero positivo (detección de arriba), un falso negativo (detección del medio) y un falso positivo (detección de abajo) cuando en realidad hay dos verdaderos positivos (detecciones de arriba y abajo) y un falso negativo (detección del medio).

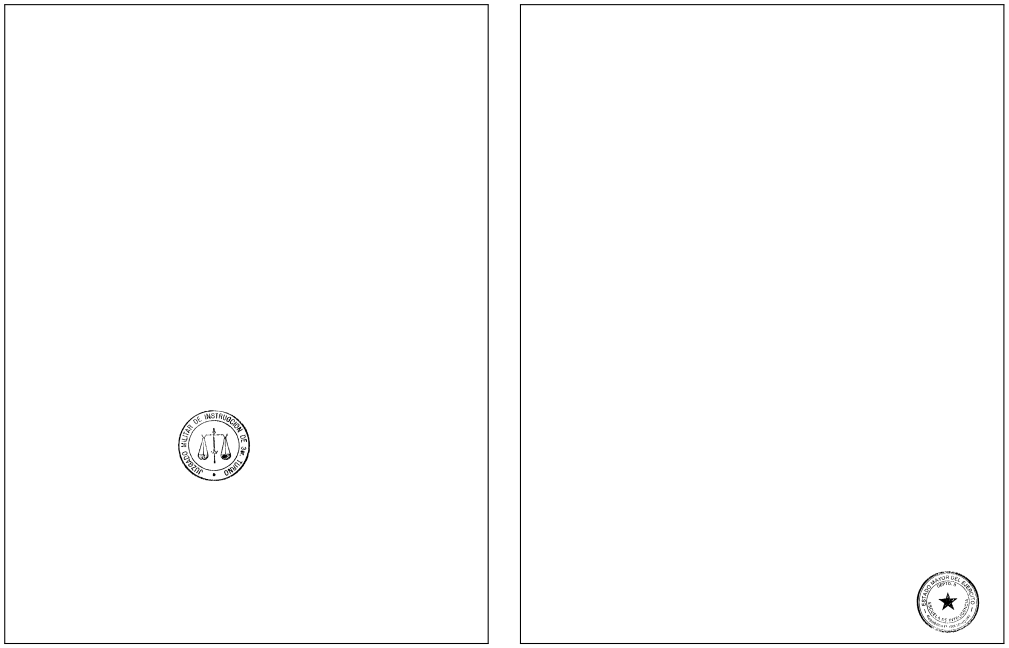


Figura 3.18: Ejemplos de documentos generados contenidos en la base sintética.

mentos negros, pero se busca emular hojas en blanco que son más frecuentes) donde cada uno contiene únicamente un sello. Los sellos se obtienen de una base previamente clasificada por voluntarios y estudiantes de la FIC. El sello se encuentra ubicado de manera aleatoria en cada documento, rotado un ángulo aleatorio y degradado o erosionado por un pequeño factor también azaroso. En la figura 3.18 se observan algunos de estos documentos sintéticos generados.

Al igual que en la base original, cada documento de la base sintética fue creado con un archivo en formato `JSON` asociado que lleva la información específica de la ubicación del sello. La ventaja en este caso radica en que, al momento de evaluar el modelo, se tendrá certeza de que los datos etiquetados son consistentes y los resultados reflejarán de manera confiable el desempeño. Esto significa que la información proporcionada en los archivos `JSON` es precisa y fiable en todos los casos, a diferencia de lo que sucede con la base original.

Por otra parte, dada la simplicidad de los documentos (un sello sobre un fondo blanco), se espera obtener resultados optimistas. El objetivo es conocer qué capacidad de detección tiene el modelo en un escenario ideal.

¿Cuándo una detección es considerada correcta?

Se propone utilizar una medida para determinar si una detección se encuentra lo suficientemente cercana a un verdadero sello para ser considerada correcta. Para esto se calcula la distancia entre el centro de las coordenadas registradas en el archivo `JSON` y el centro de la región detectada por el algoritmo. Cuando la distancia

entre ambos centros es menor a un umbral establecido, es posible afirmar que la detección está lo suficientemente solapada al objeto señalado con *Label Me*.

En este caso en que los sellos no difieren considerablemente en tamaño, esto es similar a la métrica denominada *Intersection over Union* (IoU, también conocida como índice de Jaccard [3]) la cual mide el área de intersección entre las regiones de las etiquetas y las detecciones.

Estas coordenadas que contienen los archivos `JSON` se corresponden con los vértices de las zonas delimitadas por los estudiantes. Por lo tanto, es necesario calcular las coordenadas del centro de dicha región para poder ser comparado con el centro detectado.

Se realizó una evaluación empírica basada en la observación y la experiencia adquirida para determinar el valor óptimo del umbral para dicha distancia. Este valor corresponde a 50 píxeles a escala uno entre ambos centros, considerando que tanto la detección como la etiqueta no son exactas. Con este criterio, una distancia mayor resulta en un candidato incorrecto. Cabe observar que en caso de escalar la imagen el umbral se debe escalar por el mismo factor.

3.8. Resultados

Una vez detallado el funcionamiento del algoritmo junto con todos sus componentes, se procede a determinar cuál de las opciones es la más eficiente en términos de tiempo de respuesta y desempeño.

Para esta sección se utilizan dos conjuntos de *ground-truth*: parte de la base inicial y la base sintética mencionada en la sección anterior 3.7.

Del total de los datos etiquetados se tomó aproximadamente el 5 % para generar resultados, éste subconjunto cuenta con 2.334 imágenes en las que hay 382 sellos en total, de estos hay 183 que pertenecen a las clases de interés 2.2.

A su vez, sobre este subconjunto se experimenta una modificación. La llamaremos base modificada y consta de los mismos documentos pero las etiquetas corresponden a sellos únicamente de las clases de interés, es decir que de los 382 sellos originales (183 de las clases de interés y 199 de otras clases) quedan únicamente los 183 de las clases de interés. Con esto se busca investigar si el algoritmo funciona mejor sobre las clases de interés o si es capaz de detectar cualquier clase de sellos independientemente de las plantillas utilizadas.

[3] `https://en.wikipedia.org/wiki/Jaccard_index`

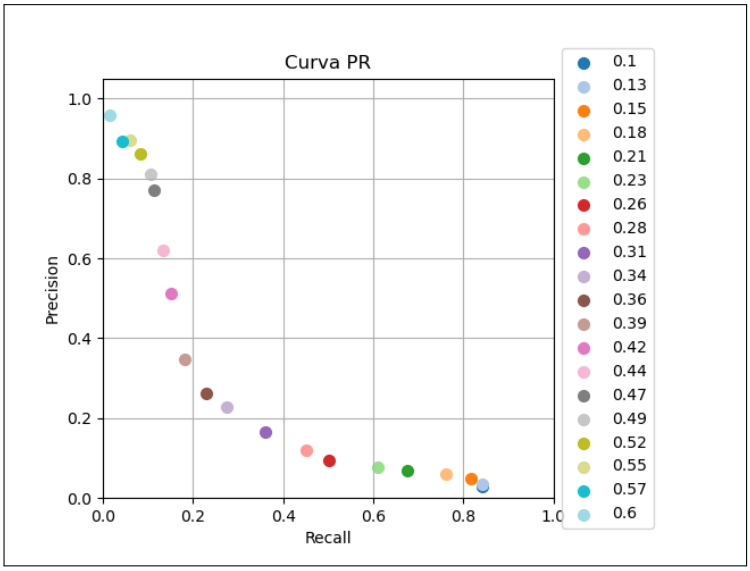


Figura 3.19: Resultados para distintos niveles de umbral, sobre *ground-truth* original. Se utiliza como plantilla *un sello por clase*. Se observa lo explicado en la figura 3.13, para umbrales bajos la precisión es baja pero se detecta gran porcentaje de los sellos, mientras que para umbrales altos la precisión es alta pero se pierden sellos.

3.8.1. Experimentos sobre la base de datos inicial

Aplicación de una plantilla por cada clase

Para la primera prueba se utiliza como plantilla el conjunto de sellos de la figura 2.2. Esto significa que para cada documento se repiten 5 cálculos de FNCC (uno correspondiente a cada plantilla).

Se ejecuta el algoritmo sobre la base inicial original (sin modificaciones) con el fin de evaluar la capacidad del algoritmo para identificar correctamente los sellos en diferentes contextos. Se obtiene la curva *Precision-Recall* de la figura 3.19, donde se observa el desempeño para distintos umbrales. Cabe recordar que el umbral determina si un pico de correlación hallado en una coordenada es suficientemente fuerte para ser considerada potencial zona de sello.

A primera vista se observa que umbrales de valor alto (entre 0.47 y 0.6) corresponden a valores de precisión altos, pero *recall* bajo. Esto tiene sentido ya que un alto valor de umbral es más selectivo, eligiendo una baja cantidad de candidatos pero acertando a una tasa alta entre 80 % y 100 %.

Por otra parte, para umbrales muy permisivos se obtienen altos valores de *recall*. Se alcanza una tasa del 85 % pero con una precisión menor al 10 %, lo que resulta en una gran cantidad de falsos positivos.

Es importante considerar que la utilización de un único representante por clase

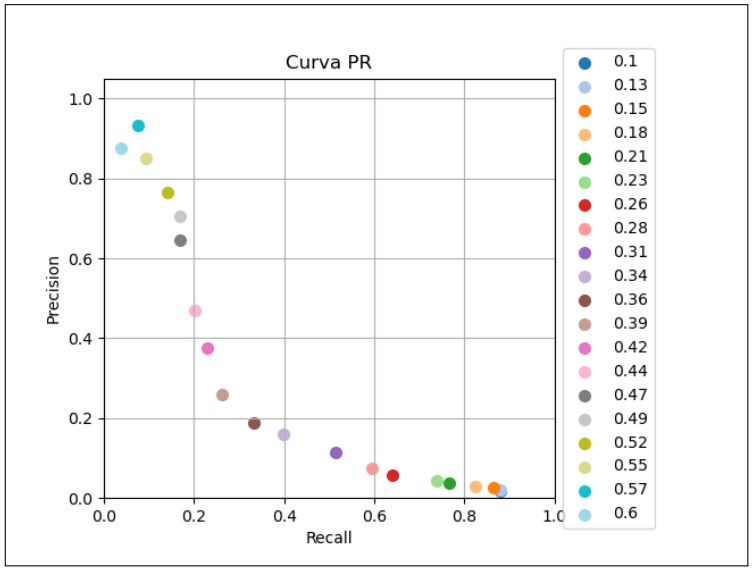


Figura 3.20: Resultados para distintos niveles de umbral, sobre un *ground-truth* modificado para las clases determinadas. Se utiliza como plantilla *un sello por clase*.

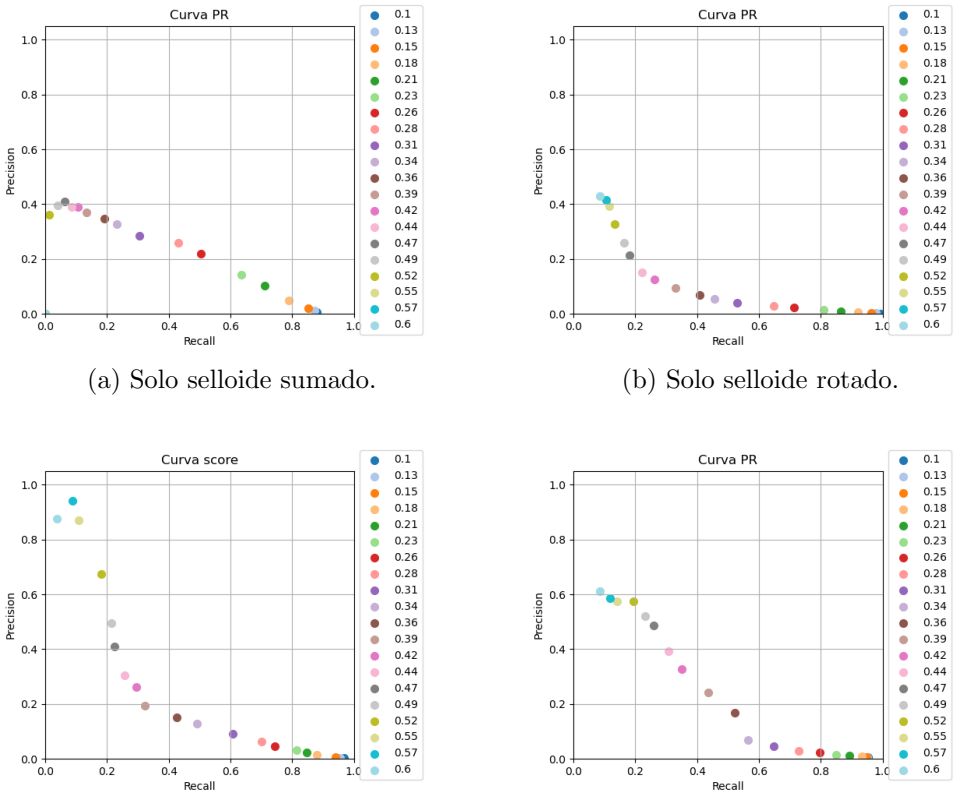
como plantilla podría resultar muy específico y no lograr contemplar todas las posibles irregularidades de cada clase. A priori esto podría presentar dos limitantes: por un lado, que el detector se reduzca a encontrar únicamente sellos de las clases correspondientes a las plantillas elegidas como referencia (desestimando otras clases de sellos no contempladas). Y por otro lado, que se especifique aún más en hallar sellos que tengan características similares a las plantillas.

Sin embargo, en la práctica el algoritmo es capaz de abarcar un rango más amplio de variedades. Recordando que la base está conformada aproximadamente por la mitad de sellos de interés y la otra mitad por sellos de otras clases; y notando valores de *recall* mayores al 50 %, se concluye que el algoritmo detecta sellos por fuera del conjunto de sellos de interés.

Por ejemplo, para el umbral de 0.18 en la figura 3.19 se obtiene un *recall* es mayor a 80 %. Por tanto en el supuesto caso que se hayan detectado todos los sellos de interés (50 %), el 30 % restante correspondería a sellos de otras clases. Esto ocurre a pesar de que se consideren las plantillas únicamente de las clases de interés.

Una forma de verificar si efectivamente se detectaron todos los sellos de interés, es ejecutando el algoritmo para el *ground-truth* modificado. Donde se toman como verdaderas únicamente aquellas detecciones sobre sellos pertenecientes a las clases de interés. Si en este caso se tuviera un alto desempeño, podría concluirse que el detector tiene capacidad para diferenciar entre clases de sellos (ya que todos son de características generales similares).

En la figura 3.20 se observa un desplazamiento hacia la derecha y abajo de todos



(a) Solo selloide sumado.

(b) Solo selloide rotado.

(c) Sellos y selloide sumado.

(d) Varias muestras por sello.

Figura 3.21: Resultados para los distintos métodos de detección sobre el *ground-truth* original.

los puntos en comparación con el *ground-truth* original. Dicho desplazamiento resulta en un mayor *recall* y menor precisión, siendo coherente con la modificación realizada en el *ground-truth*. Esto se debe a que continúan detectándose algunos sellos que ya no forman parte de la base de referencia, considerándose ahora como falsos positivos (menos precisión).

Por otra parte, el aumento de *recall* se debe a que la cantidad de sellos a detectar en el *ground-truth* modificado es menor y pertenecen únicamente a las clases de interés. Sin embargo, el desempeño no es perfecto como para afirmar que el detector es capaz de distinguir sellos por categoría con tanta exactitud.

Aplicación de selloides

Las siguientes pruebas realizadas son para las funciones mencionadas en las secciones anteriores. Se analizan estos casos únicamente para el *ground-truth* inicial, dado que el *ground-truth* modificado presenta un comportamiento muy similar al experimento anterior. Se ilustran todos los resultados para distintas combinaciones en la figura 3.21.

En primer lugar, si se toma de referencia 80 % de *recall*, los valores más bajos de precisión se encuentran para el selloide rotado 3.21b y para varias muestras por sello 3.21d.

A pesar de lograr altos valores de *recall*, el uso de selloides generados con rotaciones no logró detectar regiones específicas que caracterizan los sellos, sino que detectaba múltiples zonas (entre ellas una tasa excesivamente alta de falsos positivos). Por otra parte, ocurre un comportamiento similar al utilizar varias muestras por cada clase de sello, pero a su vez se agrega la problemática de que tal cantidad de plantillas enlentece significativamente el tiempo de procesamiento del detector.

Cabe recordar que el objetivo de esta etapa es detectar la mayor cantidad de sellos de un documento. Si bien no es prioritario obtener resultados altos de precisión, se esperan valores superiores al 5 %.

La precisiones más altas (tomando como referencia 80 % de *recall*) se consiguen para el selloide sumado 3.21a con un valor próximo al 10 %, y a su vez se destaca la reducción de tiempos producto de utilizar una única plantilla.

Otra observación que surge de analizar las curvas P-R 3.21 es que la utilización de selloides como plantilla puede reducir considerablemente la precisión (para valores de umbral altos), producto del algoritmo detallado previamente en 3.3.1. Al admitir más de una detección por plantilla aumenta el margen de error. Esto genera un compromiso entre el desempeño en la precisión y la velocidad de procesamiento, la cual es menor para este caso (por ser una única plantilla).

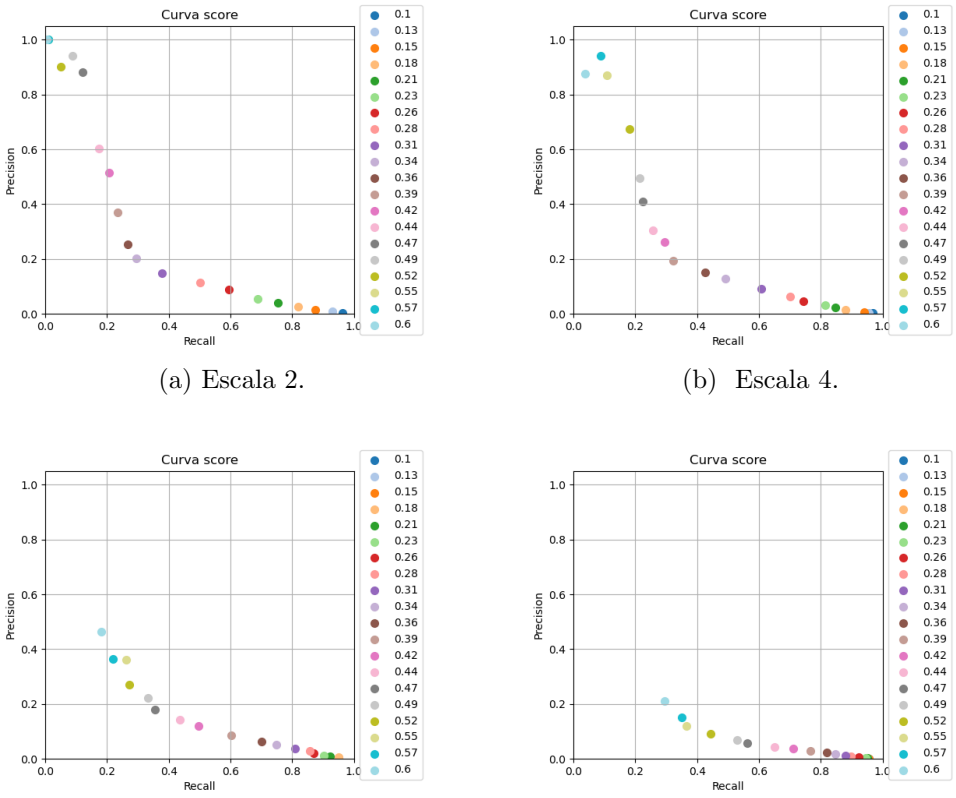
Escalas.

En la figura 3.22 se muestran los resultados para distintas escalas, donde los mejores se obtienen para las escalas 4 y 8. El impacto más significativo en el escalado se refleja en términos del tiempo de ejecución, tal como se mencionó en la sección 3.4, logrando reducir el mismo considerablemente entre las escalas empleadas.

Por otra parte, con el escalado 16 se pierde demasiada información del sello como para que sea detectado correctamente, viéndose reflejado en una precisión notoriamente menor al resto. Siempre es importante notar que se desprecian del análisis aquellos umbrales cuya precisión no supera el 5 %.

¿Qué ocurre con los sellos que no son detectados con ningún método?

Luego de analizar los desempeños de todos los casos, se observa que los mejores resultados con precisión aceptable (mayor a 5 %) no superan el 90 % de recall. El 10 % restante de los sellos a detectar no logró ser detectado satisfactoriamente con ningún método. Se podría concluir que el algoritmo no es lo suficientemente capaz de cumplir su función de detectar todos los sellos existentes. Sin embargo, del conjunto de sellos no detectados se reiteran algunos patrones en particular que son convenientes analizar.



(a) Escala 2. (b) Escala 4.

(c) Escala 8. (d) Escala 16.

Figura 3.22: Resultados para distintas escalas sobre el *ground-truth* modificado, se utilizó como plantilla sellos y selloide sumado.

En primer lugar se implementa una funcionalidad para almacenar en un directorio aquellas imágenes cuyo resultado sea considerado falso negativo, aislándolas del resto y facilitando su análisis. En la figura 3.23 se observan algunos ejemplos de documentos que el detector consideró que contenían falsos negativos.

Recorriendo los documentos con falsos negativos, se observó que en la amplia mayoría se reiteran dos comportamientos:

- Falta excesiva de tinta en el sello.
- Texto, firmas y rayas superpuestas sobre el sello.

Estas características en los sellos reducen notoriamente la posibilidad de que sean detectados correctamente. Incluso en ocasiones resulta casi imperceptible para el ojo humano identificar la presencia de un sello (como en el primer documento de la figura 3.23).

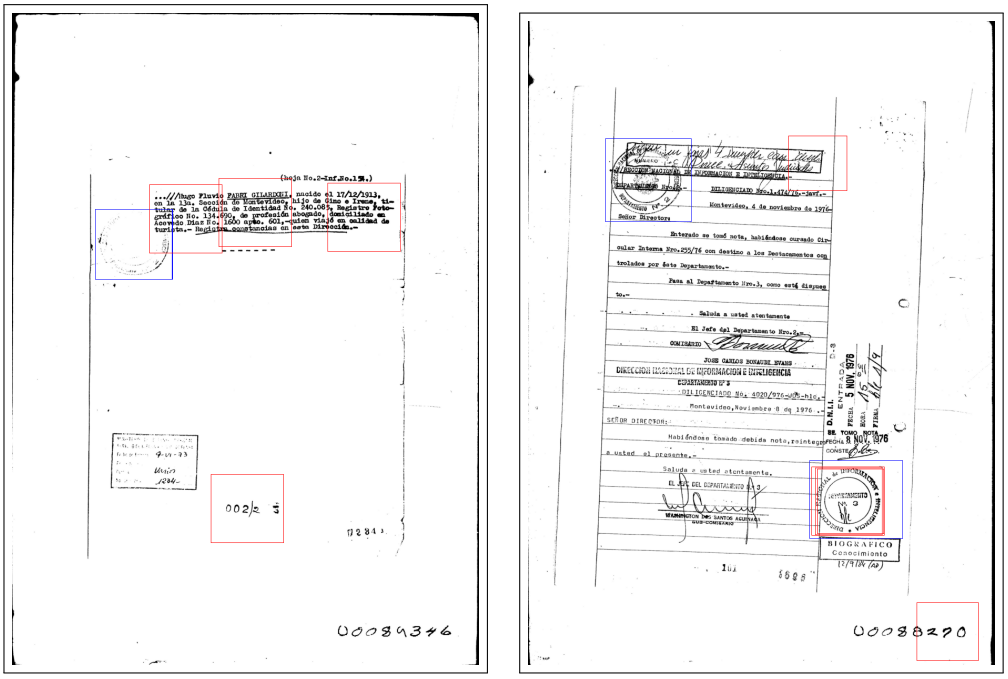


Figura 3.23: Ejemplos de falsos negativos y positivos para un umbral bajo. Los recuadros de color rojo corresponden a las detecciones del algoritmo. En azul se señalan las etiquetas, particularmente interesan aquellas no detectadas (falsos negativos).

También cabe mencionar que en la figura 3.23 se observan detecciones en lugares de la imagen con alta presencia de píxeles blancos, esto se debe a que el umbral establecido es bajo y que la NCC al penalizar zonas con muchos píxeles negros también favorece zonas con baja cantidad de píxeles negros. De esta manera, si hay alto grado de coincidencia entre la plantilla y la zona de pocos píxeles negros, la correlación logra ser mayor que los umbrales bajos (que se deben utilizar para observar los falsos negativos).

De esto se desprende la necesidad de ejecutar el algoritmo sobre una base donde cada etiqueta se corresponda efectivamente con un sello en buen estado, a modo de evaluar la capacidad máxima del detector.

3.8.2. Experimentos sobre la base de datos sintética

Para estos experimentos se utilizan como plantillas una muestra de cada clase de interés pero ahora el detector es aplicado sobre la base generada artificialmente. Como se mencionaba anteriormente, esta base está libre de texto, cuadros, firmas u otras figuras que puedan aparecer además del sello y entorpecer el calculo de la correlación. A su vez, todos los documentos se generan con sellos legibles para evitar falsos negativos por falta de tinta u otras irregularidades.

En la figura 3.24 se presentan las curvas de PR correspondientes a la base de

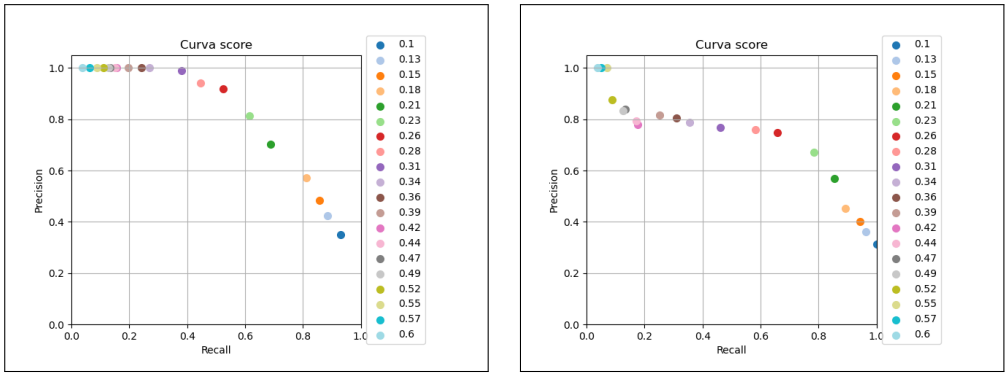


Figura 3.24: Resultados para distintos niveles de umbral, sobre la base sintética original (a la izquierda) y sobre la base sintética modificada a sellos pertenecientes a las clases utilizadas en las plantillas (a la derecha). Se utilizan como plantillas *un sello por clase*.

datos sintética. La primera imagen muestra la curva obtenida al utilizar la versión original de la base, es decir con sellos de todas las clases. Por otro lado, la segunda imagen representa las curva generada utilizando la base modificada, utilizando únicamente sellos de las clases de interés.

Como era de esperar, se observa una mejora significativa en la precisión y el *recall* en comparación con los experimentos realizados en la base de datos anterior, aumentando hasta un 40 % y un 90 % respectivamente. Estos resultados subrayan la importancia de contar con una base de datos correctamente etiquetada, y respaldan la capacidad del algoritmo para desempeñar su función de detección.

3.9. Conclusiones

A modo de resumen, esta sección abarcó un proceso integral que implicó la exploración de diversas técnicas y métricas, su implementación en un algoritmo de detección de sellos, y la evaluación de diferentes estrategias para mejorar su rendimiento. Esta fase sentó las bases para la implementación de un clasificador de sellos, tarea a profundizar en la siguiente etapa.

En términos de resultados y de acuerdo con lo esperado, se logró extraer sellos de documentos satisfactoriamente utilizando el algoritmo implementado. A su vez, el tiempo de procesado es considerado aceptable, entre un segundo y cinco segundos por documento, dependiendo de la capacidad del equipo y de si se generan imágenes de depuración o no.

Los mejores resultados obtenidos fueron de 80 % *recall* y precisión de 5 % aproximadamente. Es importante destacar que estos resultados fueron logrados sobre una base de datos que presenta ciertas limitaciones y no es considerada perfecta en términos de calidad y completitud. A raíz de los resultados con la base creada sintéticamente, se afirma que con una base de datos de mejor calidad, los resul-

tados pueden ser aún más altos. De todas formas se considera a los resultados obtenidos como indicativos y sujetos a posibles mejoras con una base de datos más confiable y representativa de la realidad.

Si bien se logró una precisión relativamente baja de aproximadamente 10 % para valores de *recall* aceptables, el enfoque principal es lograr una alta tasa de esta última. Es decir, asegurarse de que la mayoría de los sellos presentes en las imágenes sean detectados, aún a costa de un mayor número de falsos positivos o 'basura' en los resultados. Esto significa que el algoritmo es capaz de detectar la presencia de sellos en las imágenes en su amplia mayoría.

Un aspecto a remarcar es la importancia de escalar las imágenes, estrategia que optimizó el algoritmo tanto en términos de desempeño como de tiempos de ejecución. No hubiera sido posible procesar las imágenes en su resolución original a un tiempo aceptable. La velocidad de procesamiento se optimizada al aplicar un escalado en las imágenes de 1/4. A su vez, este factor de escalado es el que ha arrojado los resultados más destacados con respecto al compromiso entre tiempo y precisión.

Finalmente, es posible concluir que el algoritmo de correlación implementado cumple su función de detección para cualquier tipo de sellos que se asimilen a las plantillas seleccionadas, siempre y cuando estos se encuentren en condiciones legibles (con cantidad adecuada de tinta y sin demasiado texto superpuesto). Sin embargo, se reafirma la hipótesis de que este método no es el ideal para diferenciar sellos en categorías, resultando en un modelo débil si se empleara como clasificador: desafío que se aspira a superar en el siguiente capítulo.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 4

Etapa 2 - Clasificación

En la presente etapa se pretende clasificar de forma automática imágenes de sellos según un conjunto de clases de interés. Como punto de partida se determina el problema a resolver así como también los datos a utilizar. A lo largo de las siguientes secciones se presenta también el abordaje escogido y las técnicas exploradas.

4.1. Descripción del problema

A la tarea de clasificar instancias en categorías diferentes se la denomina como problema de clasificación multi-clase, donde cada instancia (en este caso, una imagen de un sello) debe ser asignada a una y sólo una de estas clases. A partir de un sello dado como entrada, se busca determinar a cuál de las categorías de la figura 2.2 pertenece. Este conjunto de clases fue escogido priorizando aquellas con mayor cantidad de muestras a modo de contar con suficientes ejemplos para entrenar y evaluar correctamente el modelo de clasificación.

Hubiera sido preferible disponer de una mayor variedad de sellos que abarcaran una gama más amplia de figuras y formas, en lugar de limitarse exclusivamente a sellos circulares. Sin embargo, si bien las clases escogidas se asemejan en su forma circular, todas contienen figuras que las distinguen del resto. Esto permite evaluar la capacidad del modelo para discernir entre diversos tipos de sellos. Además, la mayoría de los sellos que se encuentran en los documentos son circulares, siendo sólo algunas las clases que presentan formas rectangulares y ovaladas.

La resolución propuesta implica el uso de algoritmos de aprendizaje automático, particularmente de aprendizaje profundo, como lo son las redes neuronales convolucionales (CNN por su nombre en inglés *Convolutional Neural Network*). En la figura 4.1 se ilustra un ejemplo de un sello requerido a la entrada y su predicción correspondiente a la salida del algoritmo.

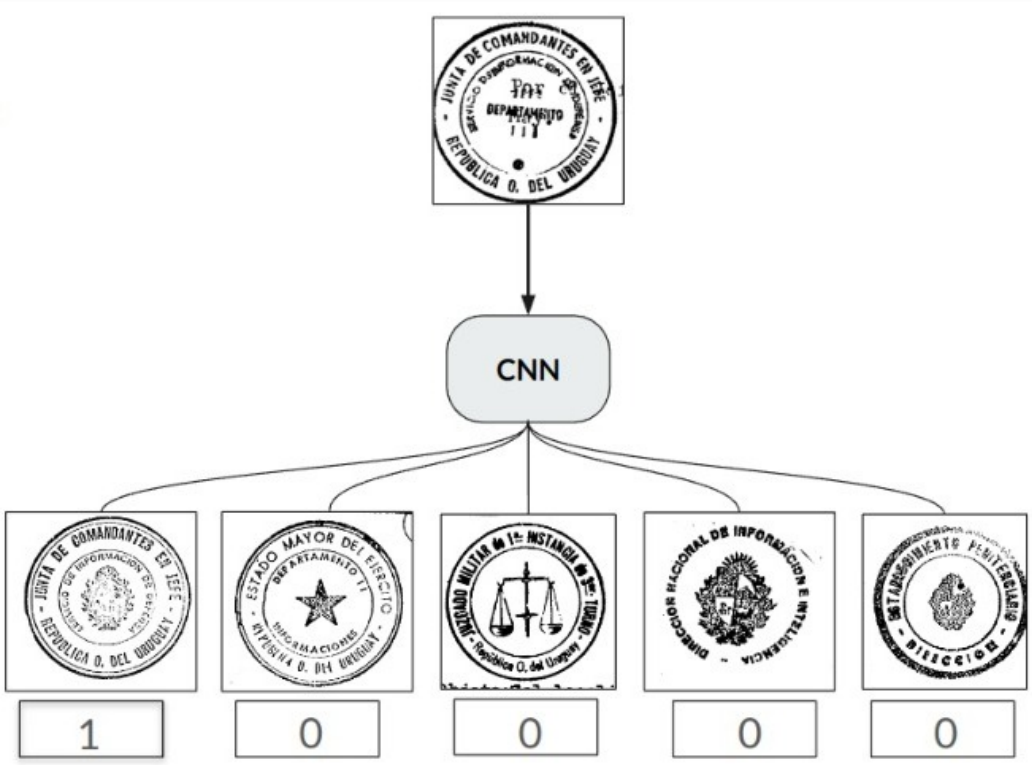


Figura 4.1: Diagrama de entrada/salida para el modelo de redes neuronales convolucionales. La entrada es una imagen y la salida un vector de ceros y un uno. El valor uno de la salida indica que la entrada se predice como de clase Junta.

Las CNN son un algoritmo especialmente adecuado para tareas de visión por computadora ya que han demostrado ser muy eficaces para problemas de reconocimiento de patrones y clasificación de imágenes. Estas redes se inspiran en la arquitectura y el funcionamiento del cerebro humano, y pueden aprender representaciones jerárquicas de las características visuales de las imágenes automáticamente sin necesidad de una extracción manual [8, capítulo 14].

4.2. Gestión de la base de datos

La creación de una base de datos adecuada es un paso esencial para construir un modelo que sea preciso y confiable. Para esta sección se parte de las cinco clases con mayor cantidad de muestras. Esta base se genera sumando las muestras previamente etiquetadas por estudiantes de la FIC con las generadas etiquetando manualmente extracciones obtenidas en la etapa de detección descrita en el capítulo anterior.

Una observación relevante es que los datos de la tabla 4.1 son los obtenidos luego de varias iteraciones de los algoritmos implementados y la limpieza manual de sus extracciones. Algunas pruebas realizadas durante este capítulo se realizaron con

CLASE	CANTIDAD DE DATOS
1) Inteligencia	801
2) Junta	2102
3) Balanza	638
4) Estado Mayor	1079
5) Penitenciario	812

Tabla 4.1: *Ground truth*: Tamaño de la base de datos.

bases previas de menor cantidad de datos.

4.2.1. Preparación para el entrenamiento del modelo

Antes de proceder al entrenamiento del modelo, es necesario preparar los datos para garantizar que estén en condiciones adecuadas y que el modelo pueda aprender de manera efectiva, lo que nos permitirá obtener resultados más precisos y confiables.

Preprocesamiento

La amplia variedad de tamaños de las imágenes es un problema al momento de procesar las zonas candidatas mediante una red neuronal, dado que el modelo a implementar admite a la entrada vectores de datos de tamaño fijo. Por tanto, el primer ajuste a realizar consiste en la estandarización de los tamaños de las imágenes de entrada. La solución propuesta implica deformar estas lo menos posible, agrandando las mismas y asignando píxeles blancos hasta igualar el tamaño del sello más grande de toda la base. Eventualmente puede haber un sello que sea el más largo y otro que sea el más ancho, en cuyo caso se ajustan las imágenes al largo del sello más largo y ancho del sello más ancho. Se implementa así ya que la variabilidad de tamaño intra-clase no es grande, es decir una clase de sello es casi siempre del mismo tamaño, puede variar en cantidad de tinta y fuente pero pocas veces de tamaño.

A su vez, se normalizan los valores de los píxeles a un rango entre 0 y 1 para garantizar que todas las características contribuyan equitativamente durante el aprendizaje del modelo.

Subconjuntos de entrenamiento, validación y prueba

Se procede a dividir el conjunto de datos de manera no estratificada en tres subconjuntos: entrenamiento, validación y prueba. La separación se realizó en dos pasos:

primero se dividió el total reservando el 20 % correspondiente para el subconjunto de prueba, que se utilizará únicamente para evaluar los modelos. Luego, los datos restantes se dividieron en 80 % para entrenamiento y 20 % para validación, quedando formada la base con las siguientes proporciones:

- Entrenamiento: 64 %
- Validación: 16 %
- Prueba: 20 %

Cabe aclarar que los números de las matrices de confusión de la sección 4.5 no son todos obtenidos utilizando la última base de datos, durante el proyecto esta fue aumentada y limpiada. Algunas de las pruebas de las secciones siguientes se realizaron sobre versiones antiguas de la base de datos por lo tanto la cantidad de datos no coincide exactamente.

4.3. Diseño e Implementación del Modelo base

En esta sección se describe la arquitectura de la red entrenada. Se detalla su estructura en términos de capas, indicando la distribución y el tipo de las mismas.

Inicialmente, la estrategia para comenzar a construir el modelo final involucra la creación de un modelo básico desde cero, capaz de funcionar completamente de punta a punta. A medida que se avanza en este proceso, se espera mejorar gradualmente el desempeño del mismo mediante optimización de parámetros y exploración de diversas técnicas.

Las decisiones tomadas en la arquitectura del modelo se basaron en las estructuras más clásicas utilizadas para resolver problemas de clasificación de imágenes mediante redes neuronales convolucionales. En primera instancia, se optó por una secuencia de algunas capas convolucionales con un incremento progresivo en el número de filtros intercaladas con capas `MaxPooling2D` [1]. Esto es una elección común en la construcción de CNN debido a su capacidad para extraer características relevantes de las imágenes mientras reduce la dimensionalidad. Finalmente se añade una capa densa seguida de la salida con cinco neuronas (una por cada clase de sellos a clasificar).

A partir del modelo base, fue realizada una búsqueda de hiperparámetros a modo de encontrar la combinación que maximice el rendimiento en el conjunto de datos de validación. Para esto fue utilizada la función de python `RandomizedSearchCV` [2]. Los hiperparámetros sobre los que se aplica la búsqueda son: la cantidad de

[1] `https://keras.io/api/layers/pooling_layers/max_pooling2d/`

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html`

capas y filtros de las mismas, el tamaño de la capa densa final, cantidad de épocas y tamaño de lote. Los rangos de la búsqueda se detallan a continuación:

- Cantidad de capas convolucionales: 2, 3 y 4.
- Cantidad de filtros por capa: (32, 64), (16, 32, 64), (32, 64, 128) y (16, 32, 64, 128).
- Cantidad de neuronas de la capa densa: 64, 128 y 256.
- Cantidad de épocas: 10, 15 y 20.
- Tamaño de lote: 8, 16, 32 y 64.

4.4. Técnicas de optimización

Sobre el mejor modelo hallado se exploran las siguientes técnicas como posibles mejoras de refinamiento y se evalúa el impacto que tiene cada uno de ellos en el resultado.

4.4.1. Autoencoders

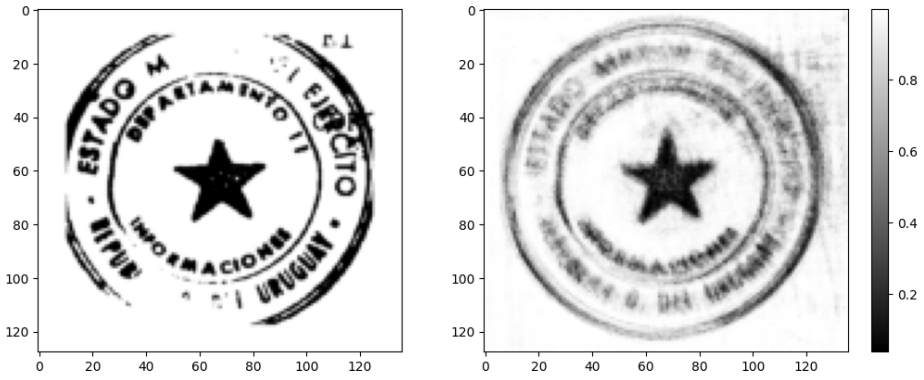
La siguiente técnica estudiada e implementada son los codificadores automáticos (más conocidos como *autoencoders*) [21].

Los *autoencoders* son una clase de redes neuronales no supervisadas comúnmente empleadas como herramientas de preentrenamiento en tareas de aprendizaje automático. Consisten en dos bloques principales: el codificador y el decodificador. Durante el entrenamiento, los *autoencoders* toma los datos de entrada, los codifica minimizando la cantidad de parámetros y los reconstruye en la salida, con el objetivo de minimizar la diferencia entre la entrada y la reconstrucción.

La arquitectura del codificador es igual a las capas convolucionales de la arquitectura base y la del decodificador es un espejo de esta, por ejemplo tres capas convolucionales de 64, 32 y 16 filtros intercaladas con capas de `UpSampling2D` [3] para tener la misma dimensión a la salida y a la entrada.

Una vez entrenado el *autoencoder*, se utiliza únicamente el codificador como una red preentrenada sin supervisado. A estas capas se le agrega la capa densa de 128 neuronas y la capa de salida de cinco neuronas y se entrena con los datos etiquetados. Esto eventualmente presenta la ventaja de que se aprovechan tanto los datos etiquetados como no etiquetados: los primeros se utilizan para el entrenamiento del clasificador y para el preentrenamiento del *autoencoder* se utilizan todos los datos. En este caso no se cuenta con datos no etiquetados pero sí se cuenta con muchos

[3] `https://keras.io/api/layers/reshaping_layers/up_sampling2d/`



(a) Original. (b) Reconstruida.

Figura 4.2: A la izquierda se ve el sello original y a la derecha el reconstruido mediante un autoencoder. Se observa que la reconstrucción de la misma mantiene características relevantes de la imagen original.

sellos que no son de las clases de interés, por lo que esto permite aprovechar todos estos sellos.

A su vez, lograr que la salida del codificador sea de tamaño menor a la imagen permite comprimir la imagen de manera óptima.

En la figura 4.2 se observa que el *autoencoder* aprende a reconstruir características relevantes de los sellos, eliminando la presencia de ruido pero también difuminando información de alta frecuencia como las letras. La figura reconstruida se obtiene a partir del decodificador, es decir que la imagen se codifica y luego se decodifica obteniendo la reconstrucción.

4.4.2. Aumentado de datos

Esta estrategia es comúnmente conocida como *data augmentation* y se basa en la ampliación de la cantidad y diversidad de datos disponibles para el entrenamiento de modelos. Consiste en aplicar transformaciones aleatorias y controladas a los datos de entrenamiento originales, generando así nuevas muestras que cumplan con las mismas características pero presentan variaciones en su apariencia. Esto es útil cuando se cuenta con una cantidad de datos muy limitada como para lograr capturar toda la complejidad del mundo real. Esta limitación fue el motivo principal para considerar su implementación, pudiendo ser una herramienta útil para enriquecer los datos y mejorar la capacidad de generalización del modelo.

En el contexto de clasificación de sellos, se propone aumentar los datos mediante morfologías, rotaciones y traslaciones, emulando las deformaciones típicas de los sellos. A modo de ejemplo, una dilatación de los píxeles negros emularía un sello con más tinta, mientras que una rotación representaría la variabilidad de ángulos

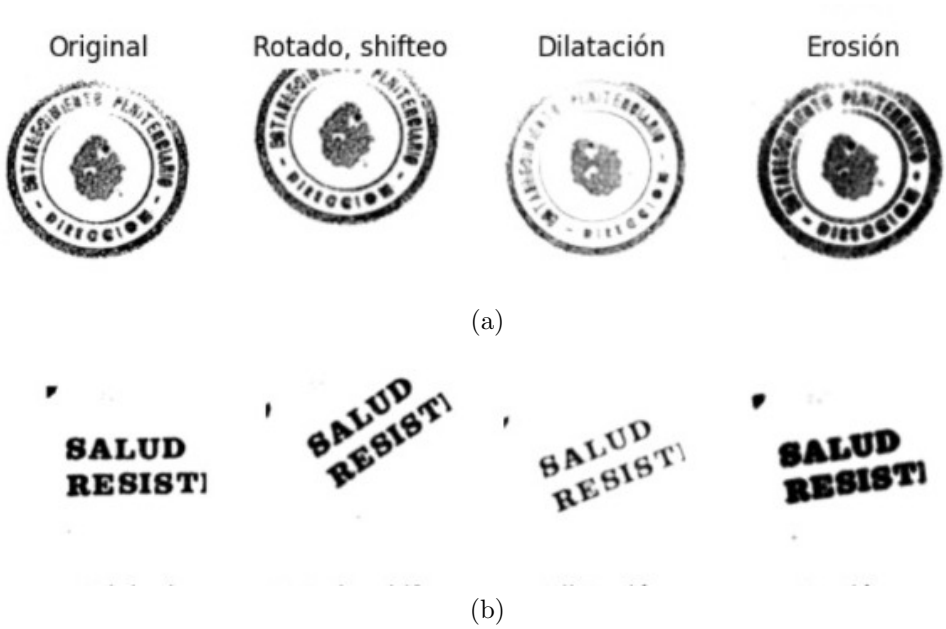


Figura 4.3: Distintos tipos de transformaciones aplicadas imágenes. En particular, la dilatación y la erosión parecen estar al revés, pero esto se debe a la convención utilizada: valor cero como blanco y uno como negro.

desde los cuales se estampa el mismo. Algunos ejemplos se ilustran en la figura 4.3.

La imágenes generadas reflejan una amplia cantidad de variaciones que podrían surgir en un contexto natural. Este proceso enriquece significativamente nuestra base de datos y mejora la capacidad de nuestro modelo para generalizar a posibles escenarios reales.

Sin embargo, este aumento en la cantidad de datos no está exento de desafíos. A medida que nuestra base de datos crece, también lo hace el consumo de recursos computacionales. Por esto es importante tener en consideración que la cantidad de memoria RAM disponible en nuestro entorno de trabajo es limitada. En este caso es esencial aplicar esta herramienta durante la fase de entrenamiento, ya que permite generar nuevas muestras sobre la marcha y de manera dinámica sin sobrecargar la memoria significativamente. Para esto es utilizada la función `ImageDataGenerator` proporcionada por la biblioteca `Keras` en Python.

4.5. Resultados

En esta sección se comparan los resultados obtenidos para las distintas técnicas de entrenamiento, así como también las optimizaciones propuestas. A su vez, se exploran variantes en la base de datos para analizar la capacidad de aprendizaje del modelo.

Modelo base con búsqueda de hiperparámetros

Los primeros resultados se obtuvieron con el modelo base implementado. El objetivo de la creación de este modelo consiste en establecer un sistema que funcione de manera integral desde el inicio hasta el final del proceso, priorizando la simplicidad del modelo por sobre el rendimiento. De esta manera, se establece como punto de partida una base sólida que facilite la comprensión del problema y permita realizar pruebas, ajustes y refinamientos de manera progresiva.

La búsqueda de hiperparámetros realizada arroja los siguientes valores que optimizan el desempeño del modelo:

- Cantidad de capas convolucionales: 3.
- Cantidad de filtros por capa: 16, 32 y 64.
- Cantidad de neuronas de la capa densa: 128.
- Cantidad de épocas: 20.
- Tamaño de lote: 8.

A pesar de ser un modelo de aprendizaje inicial, se obtuvieron resultados poco esperados a priori: el algoritmo ya logra alcanzar buenos resultados sobre el conjunto de validación. El desempeño entrenando con validación cruzada es de 98 % de *accuracy* sobre el conjunto de validación (ver resultados del entrenamiento en la sección del apéndice B), obteniendo la matriz de confusión de la figura 4.4. Una observación es que en esta parte se puede aprovechar mejor los datos de validación, ya que sobre el conjunto de train se realiza validación cruzada, entonces el resultado debería ser aún mejor. Se implementa de esta manera para poder comparar entre distintos modelos sobre el mismo conjunto de validación, en algunos de ellos se utiliza validación cruzada y en otros no.

Además, este fue el mejor resultado obtenido de todos los experimentos, por lo tanto la arquitectura de red utilizada es la de la figura 4.5.

Experimento: Aumentado de datos

De los distintos modelos, métodos y parámetros de entrenamiento explorados, uno de los resultados más destacados se logró realizando un aumentado de datos. Entrenando con los mejores hiperparámetros encontrados inicialmente, se alcanzó un desempeño del 95 % en el conjunto de validación (ver resultados del entrenamiento en la tabla del apéndice B.2).

A partir de los subconjuntos organizados en la sección 4.2.1, se obtienen las predicciones ilustradas en la matriz de confusión en la figura 4.6 sobre el subconjunto de validación.

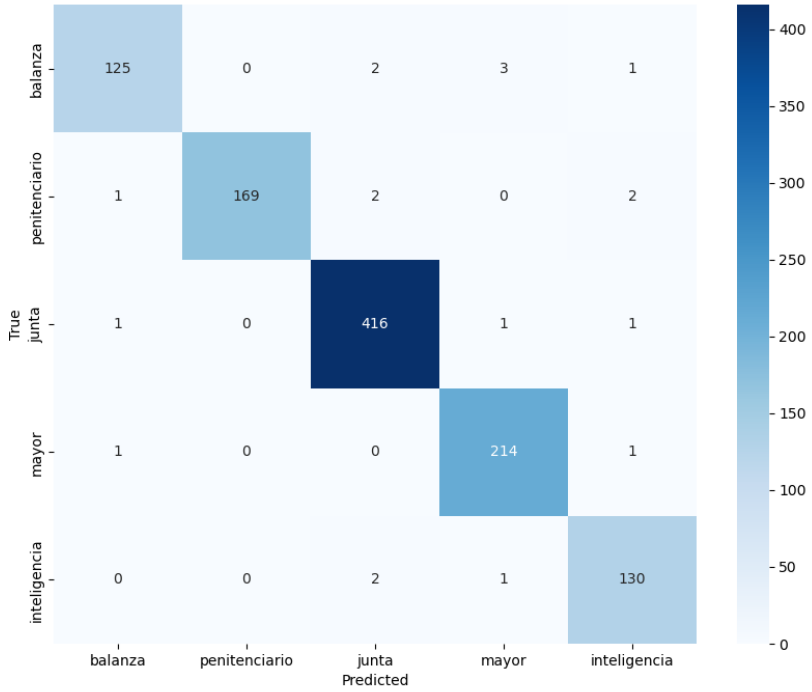


Figura 4.4: Matriz de confusión del modelo base con los hiperparámetros ajustados mediante `Grid Search`. Los valores sobre la diagonal indican la cantidad de predicciones correctas de cada clase, la predominancia de estas indica un buen desempeño del modelo.

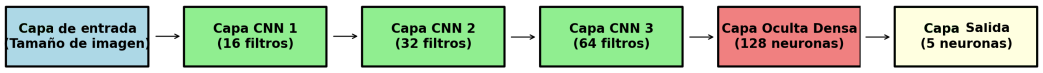


Figura 4.5: Arquitectura de la CNN con mejores resultados.

A raíz de los datos extraídos, se destaca la predominancia de las predicciones sobre la diagonal de la matriz. Esto refleja que el clasificador predijo correctamente la mayoría de las clases respecto a las verdaderas etiquetas de las muestras. Cabe destacar que la mayoría de las predicciones incorrectas se dan entre las clases *mayor* y *junta*. Particularmente estas clases presentan algunas muestras que pueden resultar similares, como las de la figura 4.7. Probablemente al aplicar el aumentado de datos sobre estas muestras se generen variantes aún más similares entre sí, dificultando en estos casos la capacidad del algoritmo para predecir correctamente.

Esta problemática se resuelve eliminando de la base de datos aquellas muestras que no representen sus respectivas clases a la perfección, aunque el modelo no estaría aprendiendo sobre una base completamente ilustrativa de la realidad.

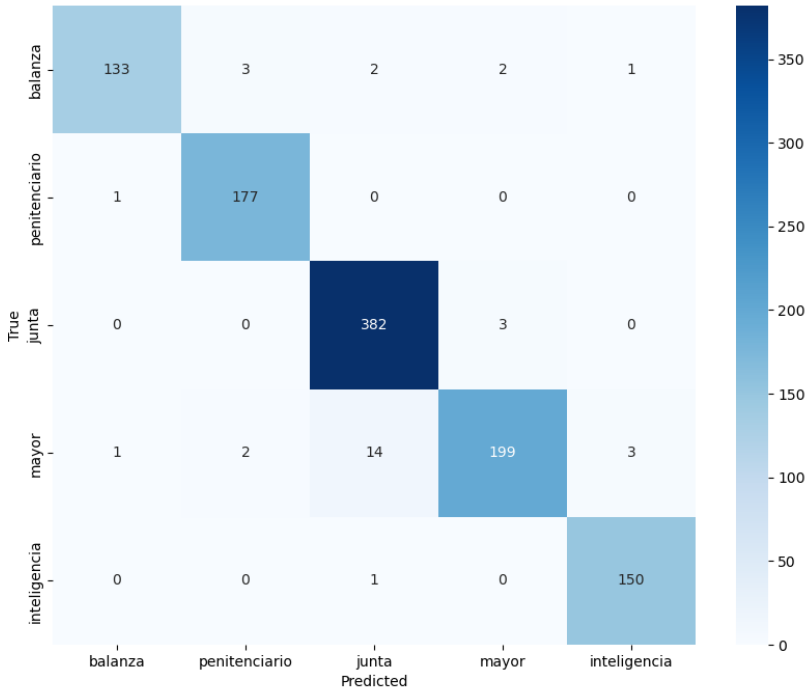
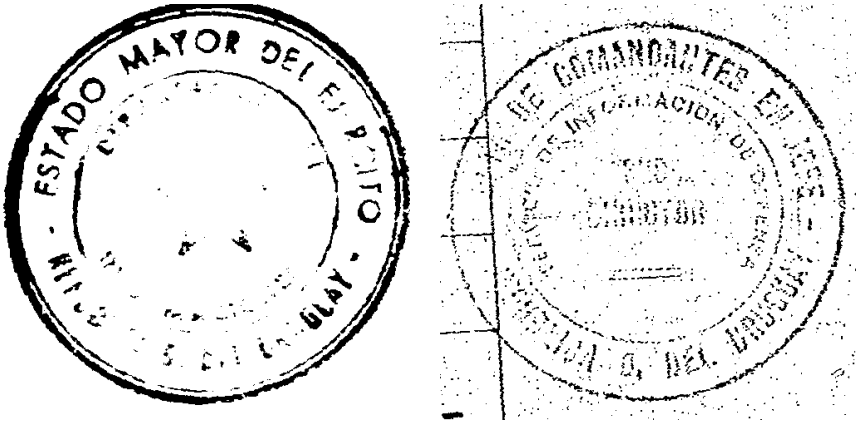


Figura 4.6: Matriz de confusión del modelo base utilizando `data augmentation`. La pérdida de desempeño se debe a que varias muestras de la clase Mayor se predicen incorrectamente como de clase Junta.



(a) Clase: mayor. (b) Clase: junta.

Figura 4.7: Ejemplos extraídos de la base de datos utilizada para el entrenamiento. Se observa la similitud entre ambas muestras ya que la característica principal de la clase Mayor (la estrella del centro) aparece borrada. Por casos como estos se asume que al aplicar aumentado de datos el desempeño baja, ya que genera muestras que puedan parecerse más aún.

Experimento: Autoencoder

En este caso, el experimento aplicando autoencoder alcanzó un desempeño del 88 % en el conjunto de validación. Para este modelo en particular, el valor de *accuracy* sobre el conjunto de entrenamiento es considerablemente mayor al de validación como se observa en la tabla B.1, indicando que el modelo posiblemente esté sobre-ajustándose demasiado a los datos de entrenamiento. El proceso de codificación del autoencoder logra ser efectivo en cuanto a la reducción de parámetros, pero no logra generalizar tan bien como se esperaba.

Experimento: Quitar muestras a una clase

Se investiga el impacto de variar la cantidad de muestras por clase. La pregunta que se busca responder a través de este experimento es: ¿A qué cantidad se puede reducir un conjunto de muestras de una clase sin que afecte significativamente el rendimiento del clasificador? Es decir que si el usuario deseara clasificar una nueva clase, ¿cuántas muestras deberá etiquetar manualmente para entrenar un modelo que logre generalizar?

La prioridad en esta implementación, además de alcanzar un desempeño y tiempos de ejecución adecuados, es requerir de la menor manipulación posible por parte del usuario a modo de simplificar la tarea. Por tanto, es deseable que el proceso de etiquetado sea simple.

Por ejemplo, se analiza el caso entrenando únicamente 40 muestras de la clase *balanza* (inicialmente se tienen aproximadamente 400) y manteniendo la cantidad de muestras del resto de las clases. Se obtiene la matriz de confusión de la figura 4.8 con una precisión de aproximadamente 80 % sobre esta clase.

Para este ensayo en particular no fue aplicado ningún método de balanceo de clases, por lo que se ve que con algunas decenas de muestras ya es posible obtener resultados aceptables. Esto puede ser de interés si se desea clasificar una nueva clase de sellos y no se cuenta con suficientes muestras etiquetadas previamente, ni con el tiempo necesario para hacerlo.

Experimento: Clase de sellos ovalados

La intención en este experimento es analizar la capacidad que tiene el modelo de aprender a diferenciar características globales, como las formas de los sellos. Hasta el momento, toda la base de datos estaba formada por imágenes de sellos circulares (aunque cada clase contiene una figura en particular que la diferencia del resto). En la figura 4.9 se muestra un ejemplo de un sello ovalado, actualmente se cuenta con 150 instancias de estos sellos.

Entrenando con todas las muestras disponibles se obtiene la matriz de confusión de la figura 4.10.

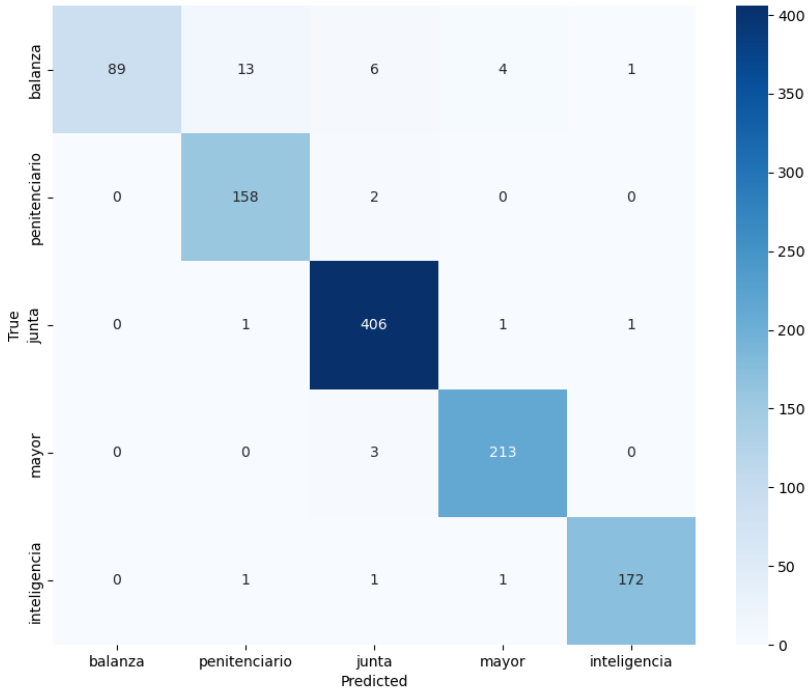


Figura 4.8: Matriz de confusión luego de reducir 10 veces la cantidad de muestras de entrenamiento de la clase de balanza. Para los experimentos anteriores el accuracy sobre esta clase ronda el 95 %, para este experimento baja al 79 % .

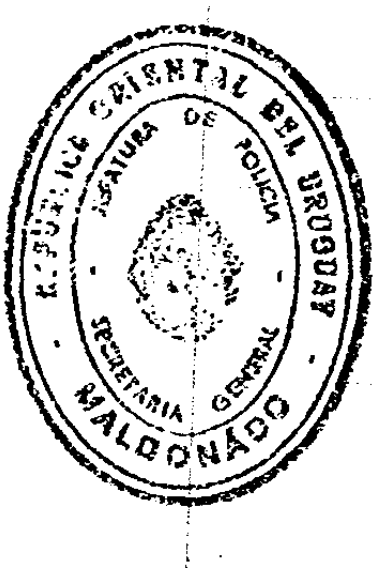


Figura 4.9: Muestra de un sello ovalado, nueva clase a clasificar.

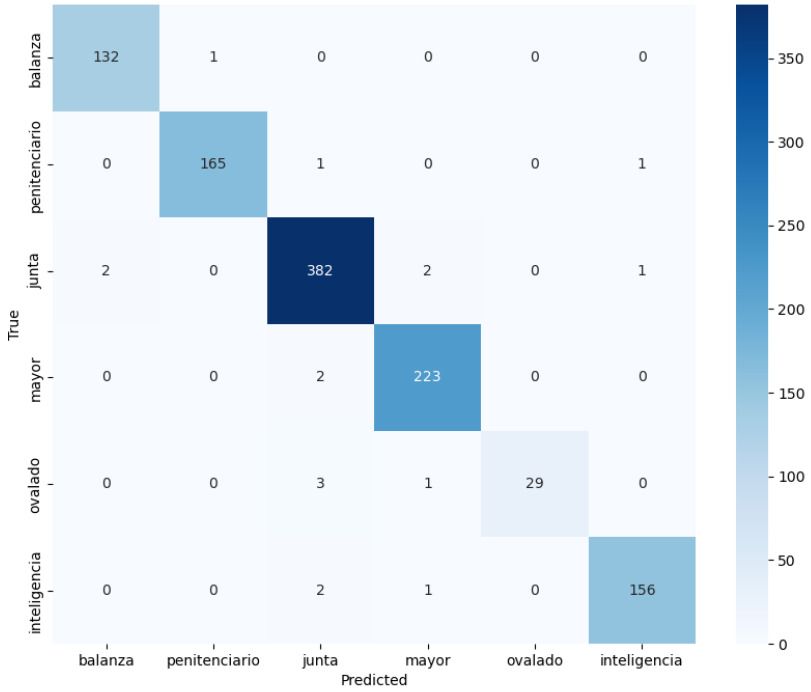


Figura 4.10: Experimento: Matriz de confusión entrenando con todas las muestras de la clase Ovalado. El accuracy sobre esta clase es de 88 %.

Dado que los resultados son considerados satisfactorios, se reduce la cantidad de muestras del subconjunto de entrenamiento de la misma forma que en el experimento anterior, esta vez hasta diez muestras de la nueva clase. La matriz de confusión de la figura 4.11 comienza a arrojar predicciones menos aceptables sobre esta clase, aunque dada la cantidad de muestras resulta ser un buen desempeño.

Esto nos da algún indicio de que para una nueva clase que presente características particulares que se diferencien del resto (en este caso la forma ovalada), basta contar con algunas pocas decenas de muestras para obtener buenos resultados.

4.6. Conclusiones

En esta sección se llevó a cabo el proceso de exploración, implementación y optimización de un modelo de redes neuronales convolucionales para encontrar el clasificador que mejor se adapte a los objetivos planteados.

En primer lugar, se concluye que la cantidad de muestras disponibles en la base de datos resulta ser adecuada para abordar la tarea de clasificación de manera

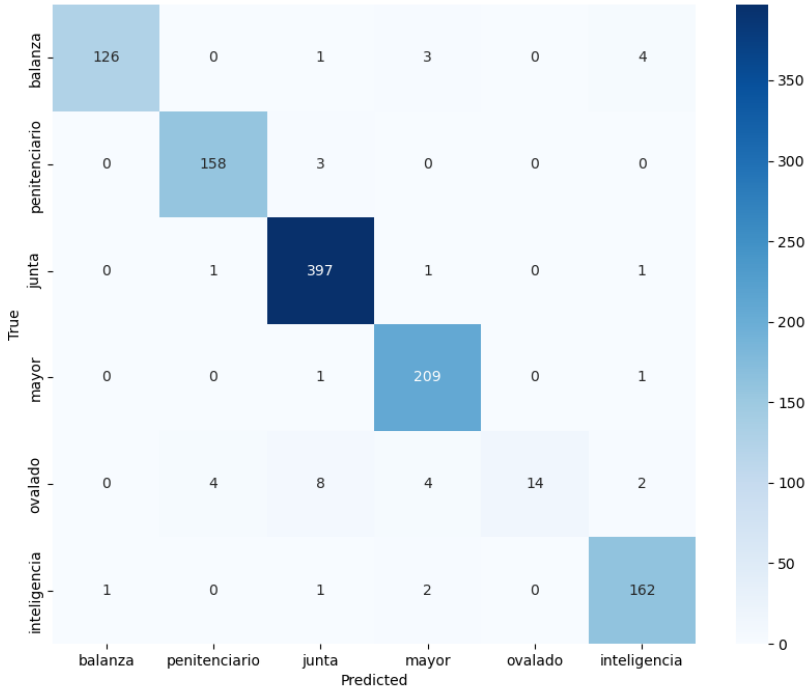


Figura 4.11: Matriz de confusión entrenando con diez muestras de la clase ovalada. El accuracy sobre esta clase ahora es 44 %.

efectiva. Esto se evidencia en la capacidad que logró el modelo para generalizar y aplicar lo aprendido durante el entrenamiento a nuevas instancias. Este éxito en la generalización verifica que el modelo ha logrado capturar las características más relevantes de los datos de entrenamiento, discriminando satisfactoriamente entre las diferentes clases escogidas a pesar de la poca variabilidad entre ellas por ser todas de forma circular.

Haber logrado mantener rendimientos destacados reduciendo la cantidad de muestras refuerza la capacidad de aprendizaje del algoritmo. Al reducir el número de muestras en el conjunto de entrenamiento y validación, se reduce también la cantidad de datos que necesitan ser etiquetados, aprovechando tiempo y recursos. Esto hace que el proceso de recopilación y preparación de datos sea más eficiente, simplificando la tarea de etiquetado para el usuario.

Otra observación radica en la capacidad de clasificación del modelo al distinguir entre sellos circulares únicamente y al incluir también sellos ovalados. La inclusión de distintas figuras geométricas facilita significativamente el aprendizaje del modelo, lo cual contribuye a lo mencionado anteriormente, donde es posible utilizar aún menos muestras durante el entrenamiento.

Por otra parte, a raíz de los resultados obtenidos en la sección 4.5 se observa que todos los métodos alcanzaron un accuracy mayor o igual a 90 % en validación, destacándose el modelo implementado a partir de la búsqueda de hiperparámetros, obteniendo 98 % de accuracy en el conjunto de validación y posteriormente el mismo resultado en prueba, lo que indica la capacidad de generalización del modelo. De todas formas cabe destacar el experimento obtenido entrenando con data augmentation, las transformaciones realizadas con esta técnica emulan de manera fiel posibles casos reales que se presentan en los documentos, convirtiéndose así en una herramienta muy útil para la mejora del rendimiento y la robustez de nuestros modelos de clasificación, incluso podría ser una buena técnica para aplicar en caso de tener pocas muestras de una clase.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 5

Etapa 3 - Integración

Recordando el alcance del presente proyecto, el objetivo final consiste en implementar un programa que determine qué sellos están contenidos en cada documento. Hasta el momento este problema fue descompuesto en dos etapas detalladas en los capítulos anteriores:

1. Detector de sellos en documentos.
2. Clasificador de sellos por categorías.

En esta sección se detalla el procedimiento de integración de ambas etapas, unificando el proceso a modo de simplificar y agilizar la tarea para el usuario que utilice esta aplicación en investigaciones futuras. Este procedimiento implica haber realizado previamente un etiquetado manual de los sellos, tarea que eventualmente será necesaria para clasificar nuevas clases de sellos. En caso que se desee clasificar una nueva clase pero se cuente con una cantidad insuficiente de etiquetas, es posible replicar este procedimiento para actualizar la base de datos.

5.1. Adaptación

En este proceso, la salida generada por el detector de sellos, debe ser transformada de manera adecuada para servir como entrada al clasificador. Esto implica que el detector debe ser capaz de identificar sellos en un documento y extraer las regiones específicas, generando la base de datos que será utilizada para entrenar el clasificador.

El desafío que se presenta al integrar ambas etapas es que la base generada a partir de las extracciones del detector probablemente no sea perfecta. Habrá detecciones que no se corresponderán con las clases de interés debido a que el desempeño del

mismo no está exento de fallas. Si así lo fuera, nuestro objetivo principal ya estaría prácticamente resuelto con las etapas anteriores.

Para la adaptación de la base de datos se ejecutó el detector en varios documentos, generando zonas candidatas entre las que se encuentran:

- Sellos pertenecientes a las clases de interés.
- Sellos correspondientes a otras clases.
- Detecciones incorrectas (denominadas como basura). Partes de documentos que no contienen ningún sello.

El clasificador implementado en el capítulo anterior está diseñado para recibir a la entrada únicamente imágenes correspondientes las clases establecidas a la salida. Por tanto, este debe ser modificado para contemplar todas las detecciones encontradas, incluyendo aquellas que no son de interés.

5.1.1. Desbalance

Es importante destacar la importancia del umbral del detector, que cumple un rol fundamental ya que de él depende la cantidad de basura a detectar. Cuanto mayor es el umbral menor será la cantidad de basura, pero como contraparte se reducirá la cantidad de sellos detectados.

Dado que interesa identificar la mayor cantidad de sellos posibles, se opta por un umbral preferentemente bajo para los valores de correlación. Esto resulta en una relación desproporcionada entre los sellos y las detecciones incorrectas, siendo estas últimas mayoritarias. A partir las curvas P-R de la sección 3.8 se observa que la precisión es menor al 10 % si se desea un *recall* mayor al 80 %, de esta manera el detector extrae más de 10 imágenes basura por cada imagen de un sello. Si se utilizan umbrales más bajos para obtener mayor *recall*, el desbalance es aún mayor.

Una opción para tratar este problema es generar un primer clasificador binario, que distinga entre sellos y basura, considerando la proporción de muestras de ambas clases y aprovechando que la estructura de los sellos es identificable incluso en imágenes de poca resolución. Esto puede generar cierta inmunidad frente al desbalance y también frente al umbral elegido, ya que en principio este último es un parámetro variable del sistema, por lo tanto el desbalance no es de un valor fijo.

5.1.2. Preprocesado

El algoritmo del detector genera zonas candidatas del mismo tamaño que el sello utilizado como plantilla. Es decir, dado un documento y un sello se recortan las

zonas candidatas del documento. Dicho recorte es del tamaño de cada plantilla, y como todas las plantillas tienen distinto tamaño, las zonas candidatas también.

En este caso, el proceso de estandarización de tamaños implica igualar todas las muestras de sellos a las dimensiones de la imagen más alta y más ancha dentro del conjunto de plantillas. Aplicando el mismo razonamiento que en la etapa de preprocesado del capítulo anterior 4.2.1, se ajustan todos los tamaños agregando píxeles blancos en los bordes.

5.1.3. Limpieza

Un problema recurrente es que muchas veces distintas plantillas generan la misma zona candidata, obteniendo muestras repetidas. Por ejemplo, en la figura 5.1 se puede ver una detección duplicada sobre el mismo sello. Esto generará dos muestras básicamente iguales salvo por la diferencia de tamaño.

A priori, se podría interpretar como una posible técnica de aumentado de datos, pero a la hora de entrenar es importante que estas muestras no pertenezcan a distintos conjuntos. De lo contrario, se estaría evaluando el modelo sobre algunas de las mismas imágenes con las que se entrena, cometiendo el llamado husmeo de datos (*data snooping*). La solución a este problema se resuelve eliminando estos 'duplicados' a partir de un *script* de Python.

Luego de extraer las zonas candidatas con el detector y eliminar aquellas muestras duplicadas, se procede con el etiquetado de los sellos según las clases establecidas.

5.1.4. Etiquetado

Al considerar la tarea como un problema supervisado, es necesario que los datos de la base se correspondan con una etiqueta que asigne la clase a la que pertenecen.

El etiquetado manual es un proceso arduo y consume mucho tiempo, especialmente cuando se trata de grandes conjuntos de datos. Sin embargo, es indispensable para garantizar que el modelo de aprendizaje automático tenga suficiente información para aprender patrones y realizar predicciones precisas. La calidad de las imágenes y la precisión de las etiquetas son cruciales para el rendimiento del modelo.

El detector extrae zonas candidatas a un directorio específico, estas se encuentran sin clasificar. El etiquetado se realiza manualmente reordenando las extracciones en carpetas que llevan el nombre de cada una de las clases. Para este proceso de clasificación de la base de datos se cuenta con una interfaz sencilla a nivel de la terminal de comandos para organizar las nuevas imágenes en las carpetas correspondientes. En la sección del apéndice B se describe más en detalle el modo de utilización de la interfaz.

A este conjunto de datos etiquetados se lo complementa con la base que se contaba

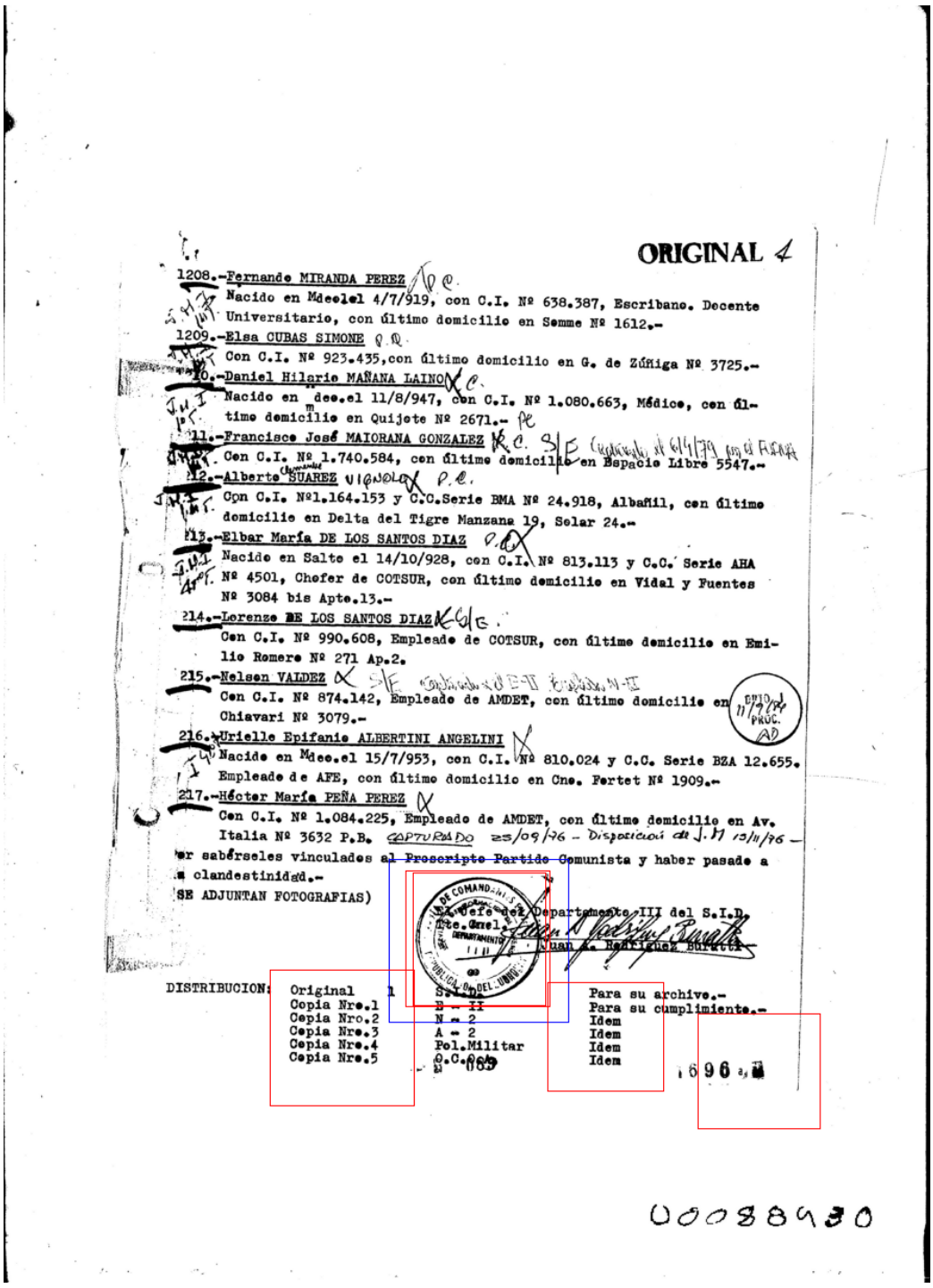


Figura 5.1: Ejemplo del detector utilizando 5 plantillas (recuadro rojo). El mismo sello (recuadro azul) es detectado 2 veces, generando un duplicado. Además del duplicado, esta detección genera tres imágenes que no son sellos, la estrategia para lidiar con esto es considerar estas imágenes como una nueva clase: basura.

inicialmente (recordar tabla 2.1) duplicando la cantidad de muestras, además de generar muestras de las otras clases necesarias para integrar las etapas (clases otro y basura). Se obtiene la base definitiva (`actual`) que se detalla en la tabla 5.1.

CLASE	**CANTIDAD DE DATOS**	
	inicial	**actual**
1) Inteligencia	537	801
2) Junta	528	2.102
3) Juzgado	295	638
4) Estado Mayor	661	1.079
5) Penitenciario	505	812
6) Otro	-	1.660
7) Basura	-	22.113

Tabla 5.1: Tamaño de la base de datos en función de las clases inicialmente y luego de ampliada. La clase *Otro* refiere a todos los sellos que no pertenecen al conjunto de datos de interés. Notar el desbalance a considerar entre la clase *basura* y las demás.

Nuevamente, esta base de datos es la generada luego de varias iteraciones de los algoritmos, algunas pruebas de este capítulo se realizan con bases previas con menor cantidad de datos.

Al generar la base de datos a partir de las extracciones del detector, se observó que los tamaños son considerablemente menores a los recortes generados con LabelMe (`inicial`) manualmente. Es decir, las extracciones del detector logran capturar el sello aprovechando el tamaño de la imagen, con un margen para bordes considerablemente pequeño en comparación con algunos recortes realizados por los estudiantes (puede observarse esta diferencia en la figura 5.1, donde el recuadro azul es de tamaño notoriamente mayor al resto). Esto resulta en un costo de procesamiento menor. Para obtener el mayor aprovechamiento posible, se procesó con el detector aquellos documentos previamente etiquetados con LabelMe, con el fin de redimensionar las extracciones al tamaño estándar.

5.2. Implementación

El enfoque de las arquitecturas propuestas está puesto en cómo clasificar las imágenes que no corresponden a ninguna de las clases de interés de manera óptima,

tomándolas como otras clases dentro del clasificador. De esta manera se debe obtener un clasificador que discrimine entre cada una de las siguientes tres categorías:

- Sellos que pertenecen a las clases de interés.
- Sellos que están por fuera de las clases de interés
- Detecciones incorrectas consideradas basura.

El programa a implementar necesita la misma estructura de directorios implementada hasta el momento (para la sección de detección 3.5.1) y además dos carpetas necesarias para la etapa de clasificación:

- `base`. Contiene la base de datos necesaria en caso de entrenar un nuevo modelo.
- `modelo`. Se almacena el modelo del clasificador entrenado en formato de archivo `H5`.

Las arquitecturas de los clasificadores implementados constan de tres capas convolucionales `Conv2D` de 16, 32 y 64 filtros respectivamente con tres capas `MaxPooling2D` intercaladas. Finalmente se añade una capa oculta de 128 neuronas junto con la capa de salida. Se decide esta arquitectura dadoslos resultados que se obtienen con el clasificador de la etapa 2 (salvo la capa de salida es la misma arquitectura).

5.2.1. Tratamiento de imágenes no relevantes

En el contexto de esta clasificación, sabiendo que a la entrada del clasificador se tienen tanto imágenes de sellos como imágenes detectadas incorrectamente, ¿qué sistema es el más efectivo para discriminar ambos tipos de imágenes? En las siguientes secciones se detallan dos métodos para tratar con las imágenes consideradas basura y los sellos por fuera del conjunto de interés.

Sistema de clasificación en cascada

El siguiente enfoque implementado implica el uso de dos clasificadores en cascada. El primero es un clasificador binario que llamaremos *filtro de basura*, y se enfoca específicamente en determinar si la imagen de entrada es considerada sello o no. La estructura de este sistema se ilustra en la figura 5.2. Dado que el clasificador es binario la capa de salida consta de dos neuronas: una asociada a los sellos y otra a la basura.

Luego, el siguiente clasificador únicamente diferencia las clases de los sellos, suponiendo que en esta instancia la basura ya fue extraída con el *filtro de basura* quedando descartada para el resto del análisis. Aquellas que sean consideradas

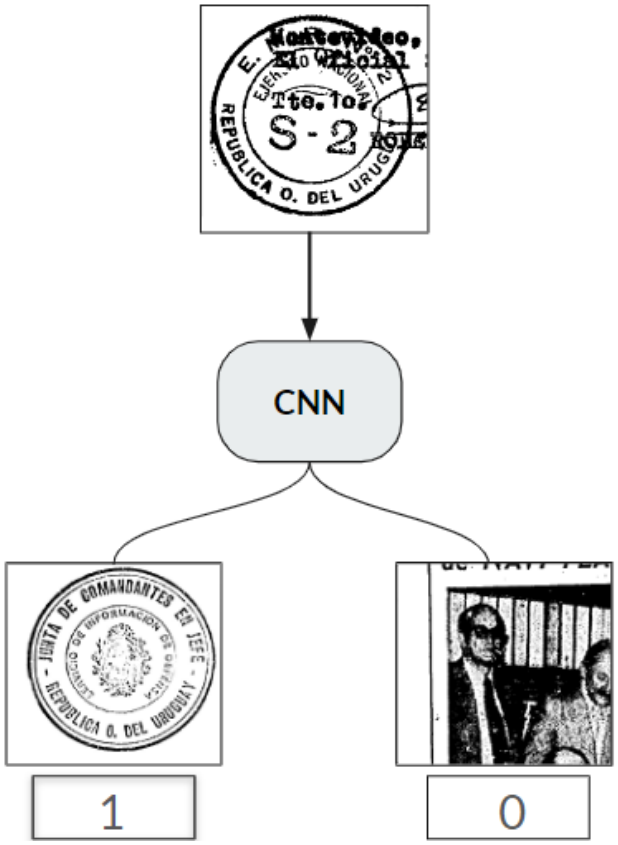


Figura 5.2: Esquema del *filtro de basura*. Se muestra como entrada un sello y a la salida su predicción, diferenciándola de la basura. Las clases corresponden a sellos (todos los tipos de sellos) y basura.

sellos, pasarán por el segundo clasificador representado en el diagrama de la figura 5.3, similar al implementado en la etapa anterior. La única diferencia es que se debe agregar una nueva clase *otros* que represente a todos los sellos que no pertenezcan a ninguna de las clases de interés. La capa de salida de este clasificador consta de seis neuronas, una por cada clase de interés y otra para el resto de clases.

Con la combinación de estos dos niveles de clasificación se busca una optimización del proceso de reconocimiento y clasificación de los sellos. Al incorporar un filtrado binario que actúa como una barrera inicial, identificando de manera eficiente los sellos auténticos entre una gran cantidad de imágenes irrelevantes (categorizados como *basura*). Seguido de un análisis detallado y especializado que garantiza una clasificación más precisa y confiable de los sellos auténticos identificados previamente.

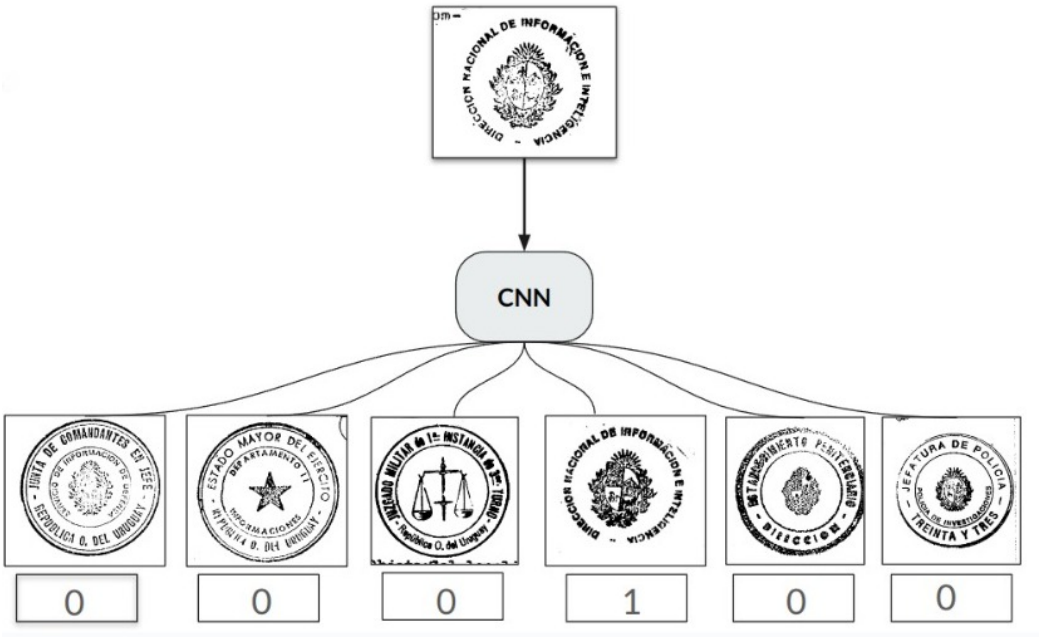


Figura 5.3: Esquema del clasificador solo de sellos. Ejemplo de una predicción correcta de la clase Inteligencia. Las primeras 5 clases corresponden a las de interés y la última representa a todas las otras clases (clase "otros").

Clasificador de sellos y basura

Esta arquitectura consta de un único clasificador que se encarga de distinguir todas las categorías mencionadas anteriormente. Es decir que en la capa de salida se tiene un total de siete neuronas: cinco correspondientes a las clases de interés, una para el resto de los sellos que no pertenecen a ninguna de las cinco anteriores (clase 'otros') y la restante para la clase basura.

En el diagrama de la figura 5.4 se puede diferenciar a simple vista que la primera clase corresponde a imágenes que no son sellos (basura). Por otro lado, el resto de las clases son las de interés a excepción de la última, que representa todos los sellos que no pertenecen a ninguna de las categorías de interés.

5.3. Resultados

Se comparan los resultados obtenidos según los métodos descritos anteriormente en la sección 5.2.1 para determinar qué arquitectura es más adecuada según el objetivo planteado.

Método 1: Clasificadores en cascada

En la figura 5.5 se puede observar la matriz de confusión del *filtro de basura* indicando las predicciones del clasificador y comparándolas con las verdaderas eti-

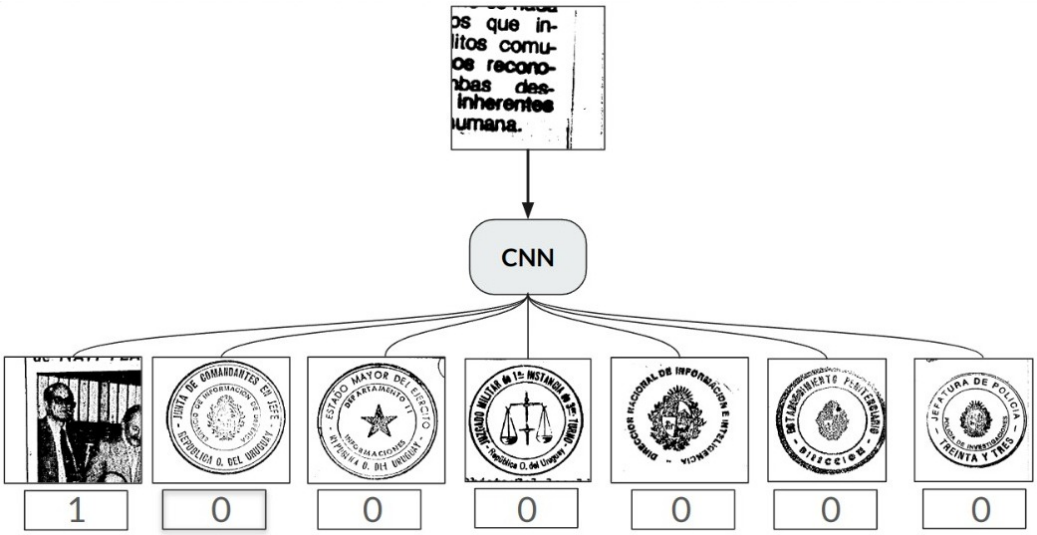


Figura 5.4: Esquema del clasificador de sellos y basura. Ejemplo de una predicción correcta de basura. A la salida, la primera clase corresponde a basura, las siguientes 5 a las clases de interés, y la última corresponde a la clase 'otros'.

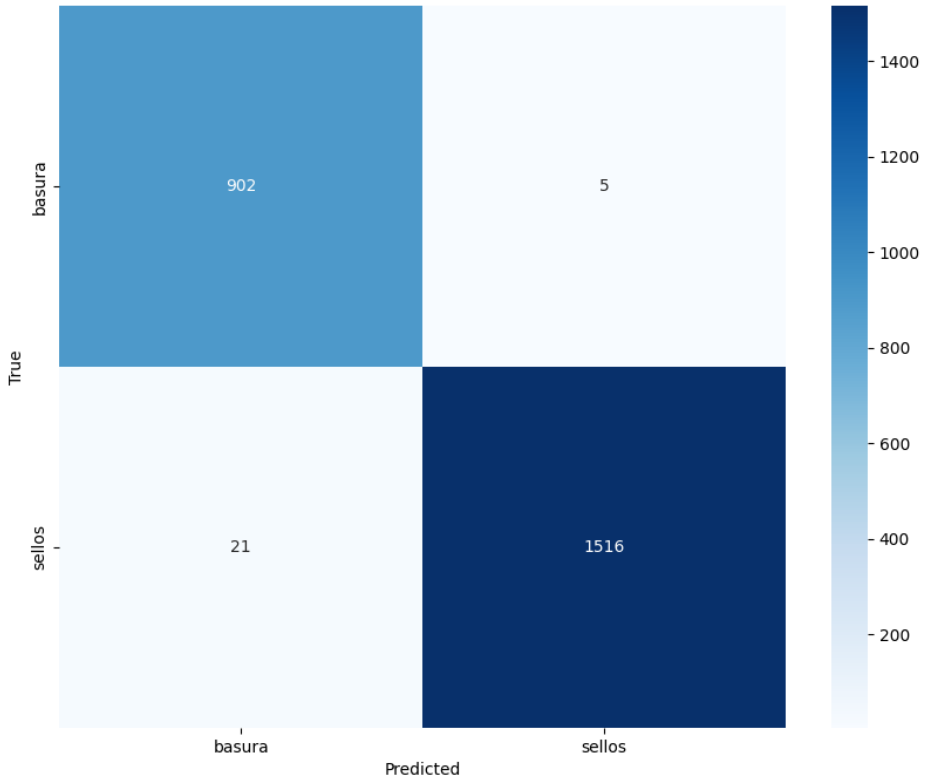


Figura 5.5: Matriz de confusión del *filtro de basura*: el primer clasificador del Método 1.

Figura 5.6: Predicciones erróneas del *filtro de basura*. Se observan sellos cuya extracción no permite visualizarlos completamente. Por otra parte, se observan figuras con estructuras similares a las de los sellos de interés que fueron consideradas como sellos. Estas predicciones se encuentran en menor proporción y son las más complejas de solucionar, ya que requiere hilar en más detalles para considerar que imágenes con estructuras muy similares a los sellos de interés también pueden ser basura.

quetas de las muestras. El *accuracy* obtenido en el conjunto de validación es casi ideal, superando el 98 %. Sobre este conjunto el modelo predijo incorrectamente 26 muestras, una cantidad prácticamente despreciable, posicionando a este método como un potencial candidato a ser implementado en el modelo final.

Visualizando algunas de las predicciones erróneas en la figura 5.6, la mayoría de las predicciones incorrectas repiten algunos patrones en particular. Por ejemplo, existen sellos cuya extracción no permite visualizarlos completamente, por lo que el clasificador los considera como basura. Puede ser razonable considerar como basura aquellos sellos que en su proceso de extracción hayan sido cortados.

Luego, a partir de las predicciones de sellos obtenidas del filtro, se toman como entrada para el siguiente clasificador de la cascada que se encarga de categorizar los sellos según sus clases. En la figura 5.7 se ilustra la matriz de confusión cuyo desempeño del clasificador obtuvo 92 % de *accuracy* sobre el conjunto de validación.

Si bien el desempeño en general resulta muy acertado a simple vista, es posible diferenciar un comportamiento inusual en la clase *junta*, que se confunde con la clase *otro* (si bien la proporción es muy baja). Una posible justificación puede deberse a que los sellos de esta clase frecuentan más de lo usual la presencia de texto y firmas escritas con puño sobre los mismos.

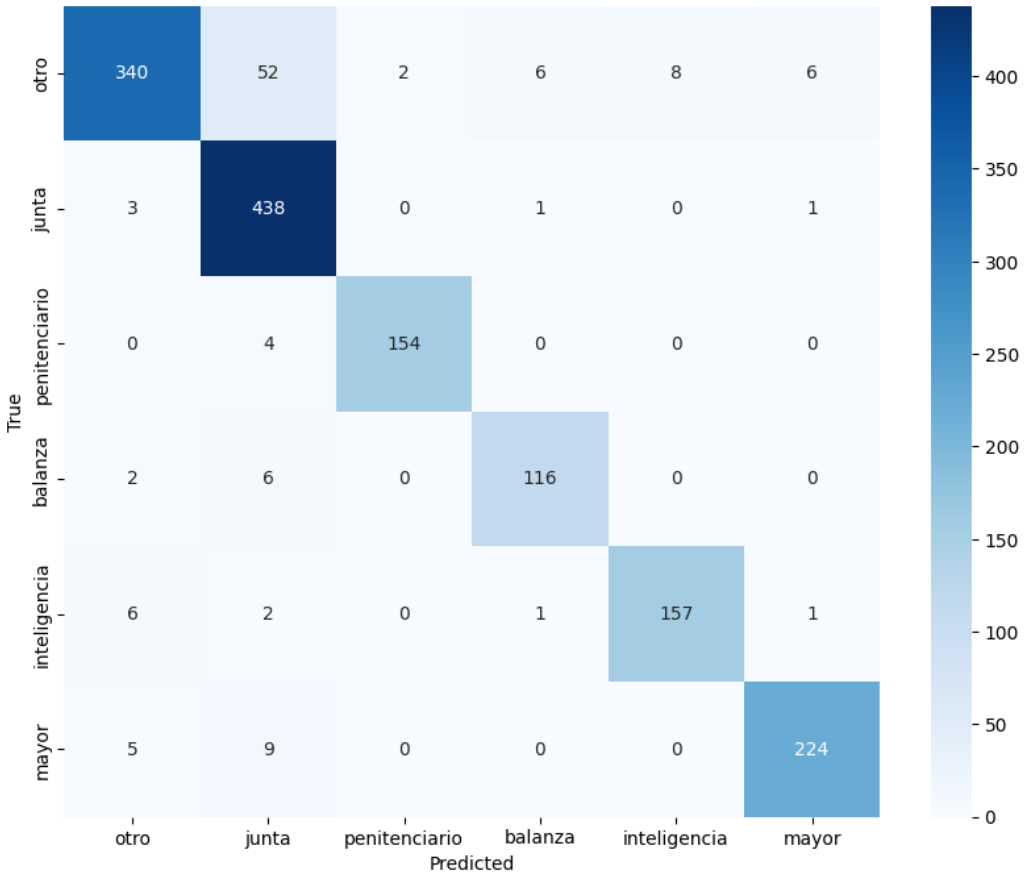


Figura 5.7: Matriz de confusión del clasificador *solo de sellos*: el segundo clasificador del Método 1.

En la figura 5.8 se observan algunas de las predicciones erróneas. Se puede afirmar que varios de los sellos de la base no están en las mejores condiciones. Esto justifica el accionar del clasificador, entendiendo que la calidad de las imágenes es suficiente para que el *filtro* anterior lo clasifique como sello, pero presenta dificultades para determinar la clase (tarea difícil de percibir aún hasta con el ojo humano).

Método 2: Clasificador de sellos y basura

Nuevamente se utiliza el modelo base encontrado en la sección 4.5, con la diferencia que a la salida se contemplan tanto las clases de interés como las que no y la basura. La matriz de confusión de la figura 5.9 indica predicciones muy favorables. Esto quiere decir que con un único modelo de redes neuronales sería posible clasificar por completo la salida del detector.

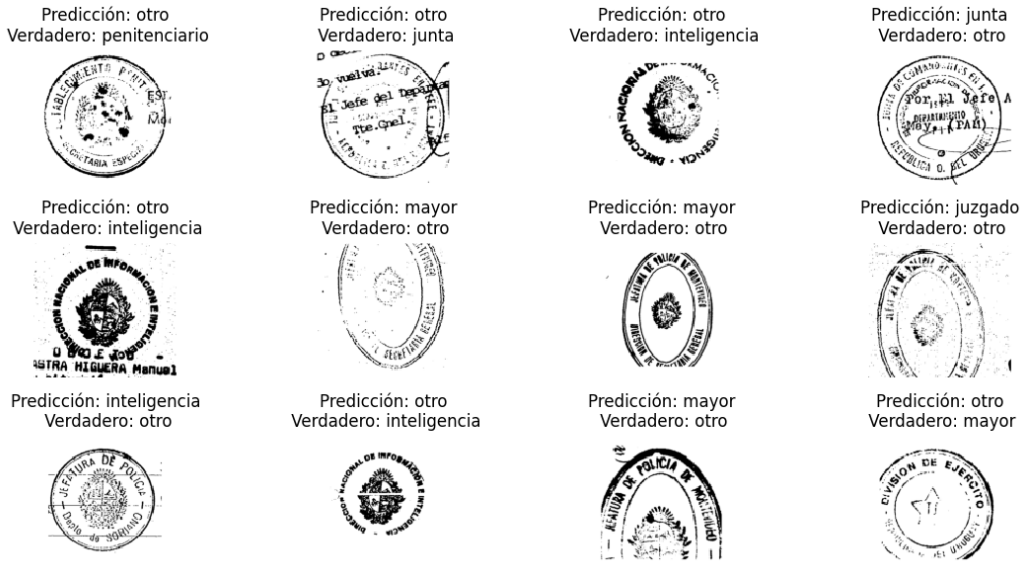


Figura 5.8: Predicciones erróneas del clasificador *solo de sellos*. Se observan varios sellos en condiciones poco aceptables con exceso de texto, manchas y rayas. También hay varios sellos con falta de tinta y otros con tinta en exceso.

Método 3: Clasificadores en cascada: *filtro de basura* con clasificador de sellos y basura.

Los métodos anteriores muestran resultados buenos sobre las bases de datos en que se prueban pero al utilizarlos en un escenario real, donde hay un amplio desbalance de clases a favor de la clase basura, el desempeño no es tan bueno como parece. Sobre todo para el método 1, ya que por más que el *filtro de basura* funcione bien, una pequeña porción de una clase mayoritaria pueden ser muchas muestras y afectar considerablemente el desempeño del algoritmo. Dado esto, se decide probar un nuevo sistema en cascada, utilizando el *filtro de basura* y además del clasificador de sellos y basura, permitiendo errores en el filtro que luego se pueden corregir con el segundo clasificador.

Para este método durante el entrenamiento del *filtro de basura* se aumentó la cantidad de imágenes de basura al orden de decenas de miles, ajustando el modelo para que aprenda sobre un escenario más real. De esta forma se obtiene cierta inmunidad al ruido, ya que las imágenes basura que no logre clasificar el filtro serán muy pocas en comparación a las que se detectan 5.10.

Dado el buen desempeño obtenido hasta el momento, en esta prueba se agregaron dos clases nuevas: inteligencia con circunferencia y mayor conjunto, con 448 y 64 muestras respectivamente. Se puede ver un ejemplo de cada clase en la figura 5.11. Estas muestras fueron recolectadas de las extracciones generadas con el detector y sumadas a algunas iniciales que clasificaron estudiantes de la FIC. La matriz de confusión obtenida se puede ver en la figura 5.12.

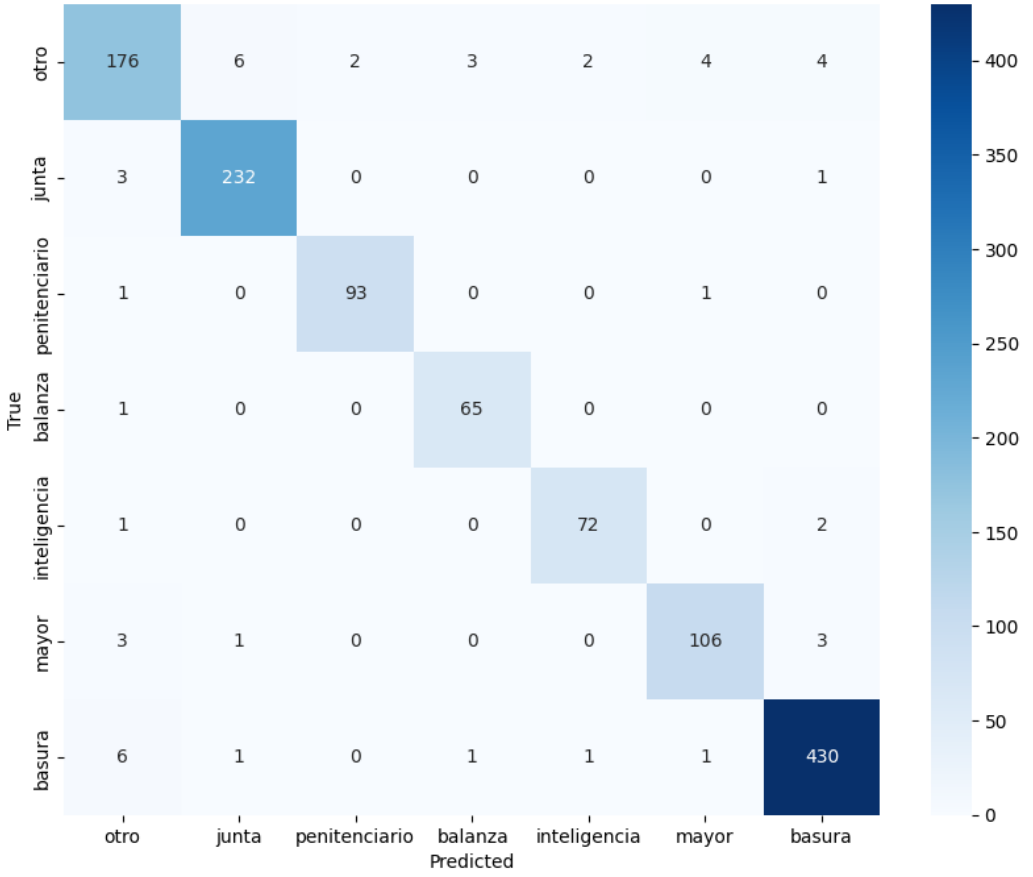


Figura 5.9: Matriz de confusión del clasificador de sellos y basura del Método 2.

5.4. Trazabilidad documental

A modo de facilitar posteriores investigaciones, interesa que los usuarios tengan la posibilidad de realizar una búsqueda por categorías, en función de los sellos, contengan los documentos. Por lo tanto una vez clasificados, se vinculan sus categorías con los documentos correspondientes que contienen a cada uno de estos sellos.

Para esto, a la salida del clasificador se genera de forma automática un archivo en formato `CSV` (como el de la figura 5.13) que contendrá asociado a cada documento un vector binario, cuya dimensión es igual a la cantidad de clases. El documento contendrá un sello de la categoría cuyo valor sea `1`, y para los casos en los que no aparezca, se asigna el valor nulo a la clase correspondiente.

A modo de ejemplo, en la figura 5.14 se observa la imagen 766 del rollo 99. Idealmente, la predicción de esta imagen contendrá un sello de la clase balanza y un sello de la clase otros.

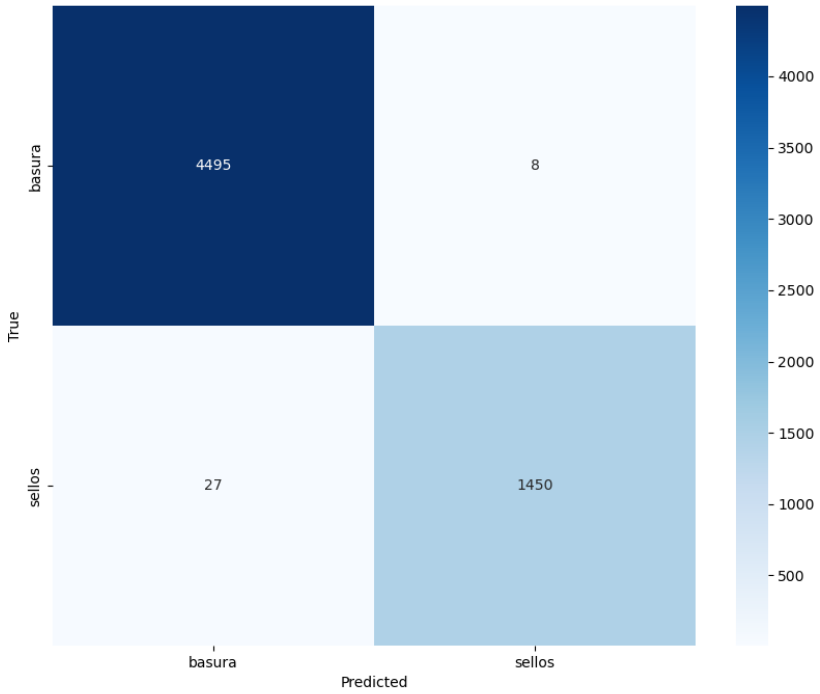


Figura 5.10: Matriz de confusión del *filtro de basura* para un escenario más real (aproximadamente tres veces más datos de basura que de sellos).

(a) Clase: Inteligencia cfa.

(b) Clase: Estado Mayor conjunto.

Figura 5.11: Ejemplos de las nuevas clases clasificadas.

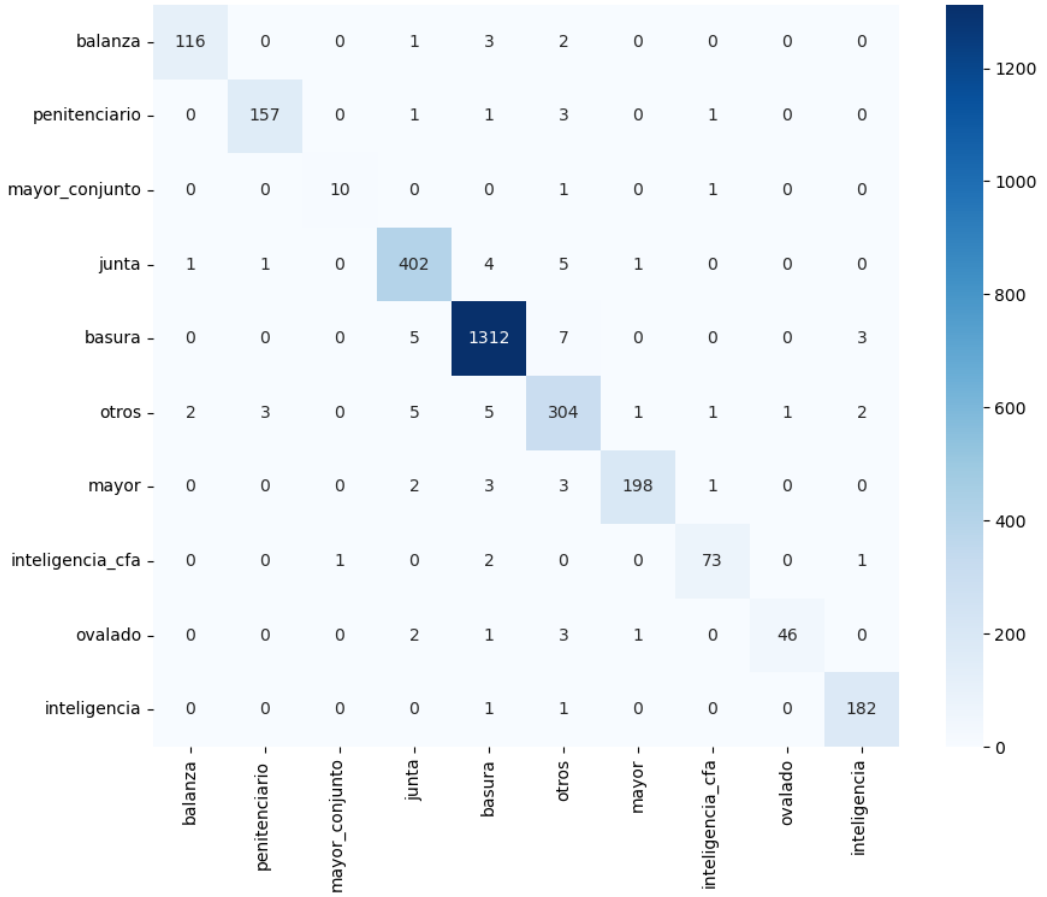
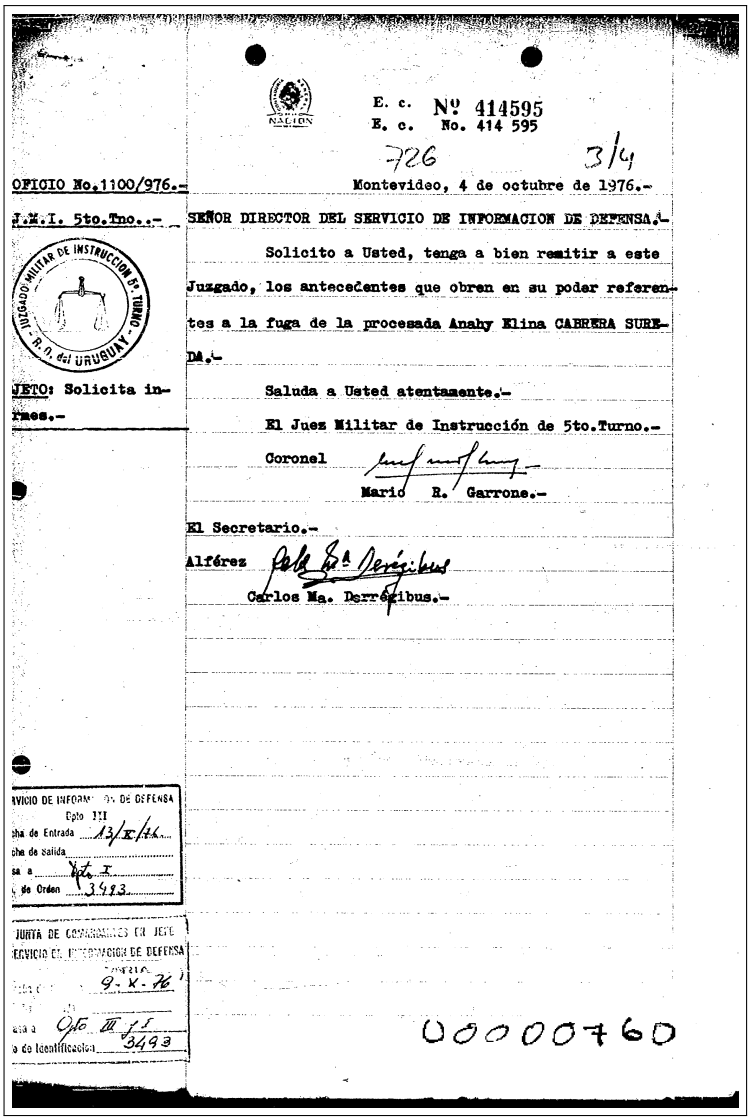


Figura 5.12: Matriz de confusión del clasificador de sellos y basura para muchas clases. Se destaca la gran cantidad de basura clasificada correctamente, además del buen desempeño en general para todas las clases.

	A	B	C	D	E	F	G	H	I	J	K	L
1		junta	penitenciario	juzgado	inteligencia	mayor	basura	otros	balanza	mayor_conjunto	inteligencia_cfa	ovalado
2	0099_0875											
3	0099_0766							1	1			
4	0099_0936											
5	0099_1383											
6	0099_0059											
7	0099_0801											
8	0099_0558						1					
9	0099_0883											
10	0099_0947	1										

Figura 5.13: Archivo en formato `CSV` generado para nueve imágenes. La fila tres indica que para la imagen 766 del rollo 99 se predice un sello de la clase otros y uno de la clase balanza.


E. c. Nº 414595
E. c. No. 414 595
726 3/4

OFICIO No.1100/976.-
Montevideo, 4 de octubre de 1976.-

J.M.I. 5to.Tno..-
SEÑOR DIRECTOR DEL SERVICIO DE INFORMACION DE DEFENSA.-

Solicito a Usted, tenga a bien remitir a este Juzgado, los antecedentes que obren en su poder referentes a la fuga de la procesada Anahy Elina CABRERA SUREDA.-

JUZGADO MILITAR DE INSTRUCCION 5to. TURNO - R.O. del URUGUAY

ASUNTO: Solicita informes.-

Saluda a Usted atentamente.-

El Juez Militar de Instrucción de 5to.Turno.-

Coronel
Mario R. Garrone.-

El Secretario.-

Alférez
Carlos Ma. Derregibus.-

SERVICIO DE INFORMACIONES DE DEFENSA
Depto III
Fecha de Entrada 13/X/76
Fecha de salida
Pasa a Depto I
Nº de Orden 3493

JUNTA DE COMANDANTES EN JEFE
SERVICIO DE INFORMACIONES DE DEFENSA
Fecha de entrada 9-X-76
Pasa a Depto III 5
Nº de Identificación 3493

00000760

Figura 5.14: Imagen 766 del rollo 99 con presencia de dos sellos. Observando la predicción de la figura 5.13 se verifica que el sello de balanza y el sello de la clase otros están contenidos en el documento. Dada la implementación, para verificar las detecciones y clasificaciones correspondientes, esto es necesario navegar por los directorios detallados en el apéndice C.

Capítulo 6

Conclusiones

Esta sección está destinada a evaluar si el presente proyecto de fin de carrera de ingeniería eléctrica culminó con éxito. A su vez, se hará mención a potenciales mejoras a ser incorporadas a futuro, con el fin de favorecer a la optimización del sistema desarrollado en lo que respecta a su eficiencia temporal y precisión.

En primer lugar, en base a los resultados analizados a lo largo del presente informe, es posible afirmar que se logró el objetivo principal establecido inicialmente. A partir de una base de documentos escaneados, fue posible reordenarlos satisfactoriamente en un sistema de carpetas en función de las clases de sellos que cada uno contiene.

La implementación de *template matching* empleando técnicas clásicas de correlación cruzada permitió reducir notoriamente la dimensión de parámetros del problema, llevando un problema de clasificación de imágenes de dimensión 4000×3000 a uno de 500×500. Esta aplicación fue capaz de indicar exitosamente las zonas de los documentos en las que se encuentran los sellos, extrayendo esta información relevante de un tamaño adecuado para ser tratado en un problema de clasificación.

Por otra parte, el diseño de la arquitectura de redes neuronales convolucionales desarrollada fue capaz de aprender de los datos extraídos y de generalizar para nuevas instancias. A pesar de las imágenes incorrectas introducidas por la etapa anterior, el modelo es lo suficientemente robusto para ser inmune ante dicha presencia, al mismo tiempo que cumple su función de diferenciar distintas clases de sellos según las características particulares de cada una.

De esta forma se puede afirmar que la decisión de combinar métodos clásicos con métodos de inteligencia artificial implementada fue acertada. Por un lado, utilizar únicamente *template matching* para este problema no es una alternativa eficiente por la variabilidad que presentan los sellos, dificultando la capacidad de clasificación. Por otra parte, reducir el problema a uno exclusivamente con inteligencia

artificial presenta el problema de la dimensionalidad mencionado anteriormente. Entrenar un modelo con imágenes de muy alta resolución, en la que solo una pequeña región de las mismas contiene la información relevante hace que el mismo sea lento e ineficiente.

En conclusión, este proyecto ha completado con éxito el objetivo propuesto. El diseño adecuado de la estructura establecida ha permitido abordar de manera efectiva un problema complejo, empleando las herramientas apropiadas para enfrentar cada tarea en particular.

6.1. Trabajos futuros

Con respecto al bloque del detector, el proceso para mejorar el valor de *recall* obtenido sería iterar hasta sobreajustarse a los falsos negativos, a modo de reconocer patrones en los que falla el detector, para alcanzar valores de *recall* más cercanos a uno.

En términos de velocidad de procesamiento, se recomienda profundizar con técnicas de multi escalado, es decir, se podrían encontrar candidatos a sellos sin necesidad de escalar a tan alta resolución, agilizando significativamente el tiempo de procesamiento sin necesidad de impactar en el rendimiento. Si bien se evaluó el algoritmo para varias escalas, es posible modificar el mismo para que durante la misma ejecución escale progresivamente cada documento hasta encontrar un sello.

Quedó pendiente explorar el uso de plantillas únicamente con circunferencias, por la predominancia de los sellos con esta forma. Detectar únicamente la presencia de circunferencias en una imagen podría ser un hallazgo beneficioso que mejore tanto en velocidad como en rendimiento, ya que incluso en imágenes de baja resolución pueden apreciarse formas circulares.

Por otra parte, para el bloque correspondiente al clasificador es posible analizar con mayor profundidad la clase *otros*. Por ejemplo, diseñar una implementación aplicando vecinos más cercanos (*k-nn*) sobre las muestras clasificadas como esta clase. Probablemente se identifiquen agrupaciones entre distintas clases de sellos, aumentando la capacidad de clasificación del algoritmo. Esto podría ser útil cuando se desee clasificar nuevas clases, pudiendo incluso eliminar el proceso de etiquetado manual (convirtiéndose en un problema de clasificación no supervisado). Es decir, la clase otros incluye sellos de muchas clases, se podría estudiar si implementando vecinos más cercanos se pueden separar automáticamente estas clases.

Aunque el clasificador produce una salida binaria, en realidad la capa de salida contiene neuronas con activación softmax, éstas generan probabilidades que no se han considerado pero podrían ser útiles.

Otra posible mejora podría ser entrenar los modelos con mayor resolución, actualmente se cuenta con modelos que escalan x4. Esto generaría modelos con un

alto costo computacional que enlentecería el proceso, pero cuyo desempeño podría mejorar considerablemente. Para esto es necesario cargar la base por lotes durante el entrenamiento, ya que guardar grandes cantidades de datos directamente en memoria resulta ineficiente.

También se puede mejorar el hecho de tener un problema con clases desbalanceadas como por ejemplo mediante el uso del parámetro *class_weights*. Para esto se debe definir un umbral de detección fijo, ya que de este depende la cantidad de basura que llegará al clasificador. También se podría utilizar aumentado de datos exclusivamente sobre clases que a priori se conocen como minoritarias, lo cual generaría datos sintéticos logrando equilibrar el problema.

Se podría probar YOLO. Dado que este está pensado para imágenes naturales se deberían reentrenar los bloques preentrenados de la red, y para esto se estima que se precisarían muchas más muestras de las actuales. De todas formas, se podría implementar un detector con YOLO, donde no se clasifiquen los distintos sellos sino que se detecten todos sin importar su clase (igual que el detector implementado) y las detecciones posteriormente clasificarlas con una CNN como las que ya se probaron.

Por último, la salida del sistema genera un archivo `CSV` que permite saber qué sellos se encuentran en cada imagen pero no permite saber la ubicación del mismo en la imagen, se implementó así para tener una guía rápida únicamente con esta información. Si bien se puede saber qué sellos clasifica en cada clase navegando por los directorios explicados en el apéndice C puede resultar poco eficiente esta búsqueda de verificación, lo mismo con la ubicación de los mismos en el documento. Para esto se podría generar además de el archivo `CSV`, otro en formato `JSON` (similar a los generados mediante `Label Me`) para saber en qué coordenadas de la imagen se encuentran cada uno de los sellos determinados.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 7

Contribuciones

Durante la realización del proyecto se aumentó manualmente la base de datos, se clasificaron manualmente sellos de las clases de interés 2.2, otros tipos de sellos se clasificaron dentro de una misma clase (otros) e imágenes de basura para poder entrenar los modelos que contemplan esta clase. La cantidad de datos inicial y actual se ve en la tabla 5.1 por lo que se puede decir que se aportaron 4.566 imágenes de sellos, la cantidad de horas dedicadas a la clasificación manual fue de 320 horas aproximadamente.

Para generar este aporte de datos se clasificaron a mano extracciones del detector, para esto se implementa la interfaz gráfica de la figura 7.1.

Existe la posibilidad de no contar con suficientes muestras de una clase de sellos de interés (que interesa saber qué documentos contienen esa clase de sellos), para conseguir nuevas muestras se puede utilizar esta interfaz, siguiendo un procedimiento similar al mencionado anteriormente.

La interfaz de la figura (7.1) requiere que el usuario utilice los comandos que se ilustran en la terminal de comandos, en función de la clase de sello que se observa en el visualizador de imágenes. Este proceso reordena las extracciones de los sellos (automáticamente almacenadas en el directorio `sellos_extraídos`) almacenándolas en las carpetas correspondientes que se observan en la figura.

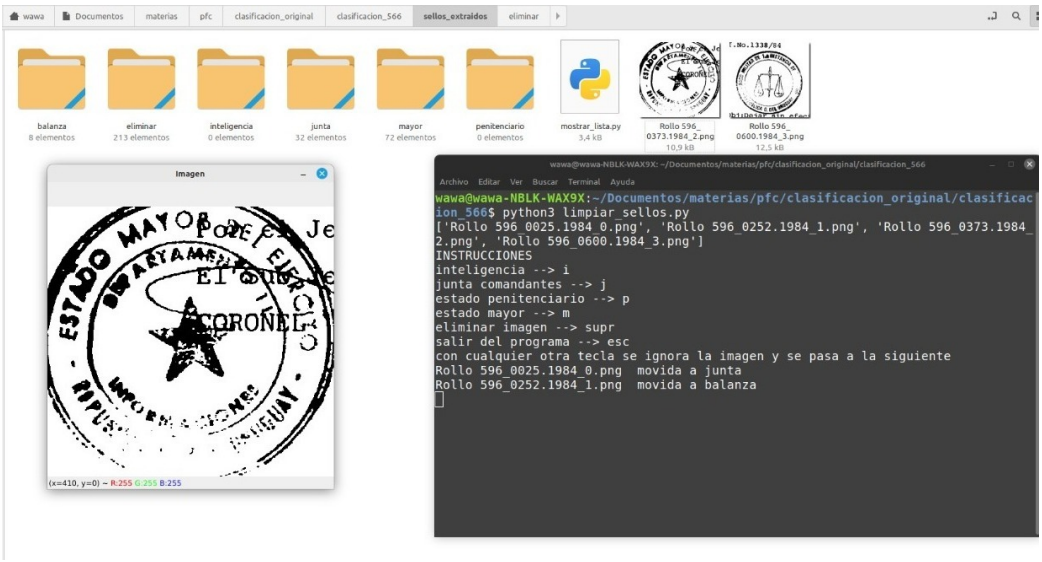


Figura 7.1: Interfaz de clasificación manual. En el ejemplo se mueve el archivo `Rollo 596_0025.1984_0.png` a la carpeta junta y `Rollo 596_0252.1984_1.png` a balanza, quedando dos archivos restantes por clasificar.

Apéndice A

Funcionamiento del detector

En esta sección se profundizará sobre el sistema implementado para el detector de la etapa 1 (desarrollado en el capítulo 3), detallando la organización del directorio y su contenido, el cual se encuentra alojado en el repositorio de *gitlab* [22].

Inicialmente se cuenta con la estructura de directorio que se ilustra en la figura A.1. El directorio `home` (ver figura A.2) debe contener una carpeta con las imágenes de los documentos en los que se desea detectar sellos (junto con sus archivos en formato `JSON` correspondientes, en caso de estar etiquetados). Por otra parte, `sellos_a_detectar` almacena las plantillas a utilizar para la búsqueda de correlación.

Estas dos carpetas son necesarias para poder ejecutar el algoritmo principal de esta etapa: `find_seal_con_selloide.py`.

Un ejemplo de la ejecución de este proceso se observa en la figura A.3, donde se generan las siguientes carpetas:

- `conjunto_control`

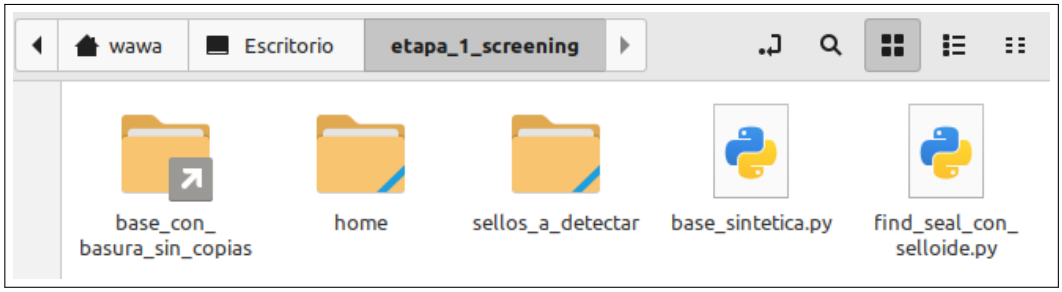


Figura A.1: Imagen ilustrativa de la disposición del directorio previo a realizar una ejecución de detección (Etapa 1).

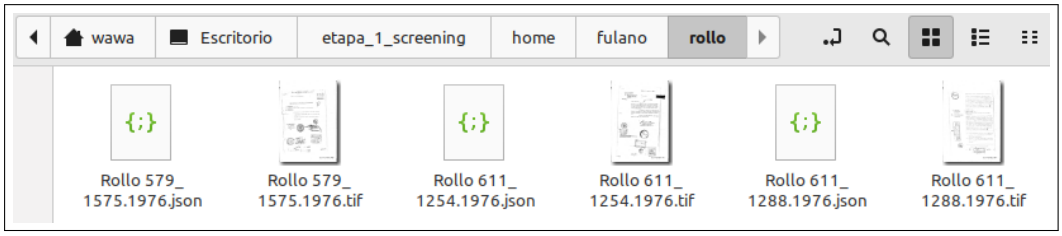


Figura A.2: Estructura de la carpeta home. Observar la estructura de dos carpetas previo a los archivos `JSON` y `TIFF`.

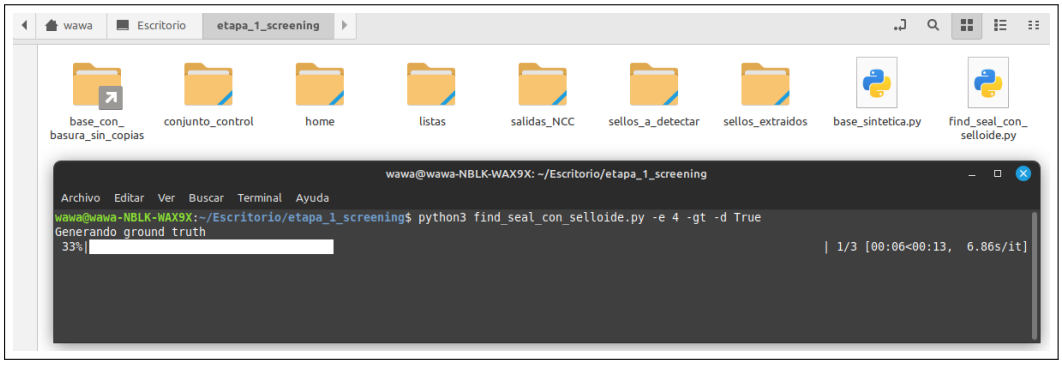


Figura A.3: Representación de la ejecución del programa y las carpetas generadas. Es importante destacar que los umbrales de detección por defecto es una lista de 0.10 a 0.60 con 18 umbrales intermedios.

- `listas`
- `salidas_NCC`
- `sellos_extraidos`

`Listas` se genera con el parámetro `-gt=True` y crea un archivo `todo.txt` que contiene las rutas hacia los archivos `JSON` (se visualiza un ejemplo en la figura A.4). Sobre estos se realiza la lectura para extraer las coordenadas de los sellos, para ser registrados en `conjunto_control`.

El directorio `conjunto_control` es generado luego de utilizar el parámetro `-gt=True`

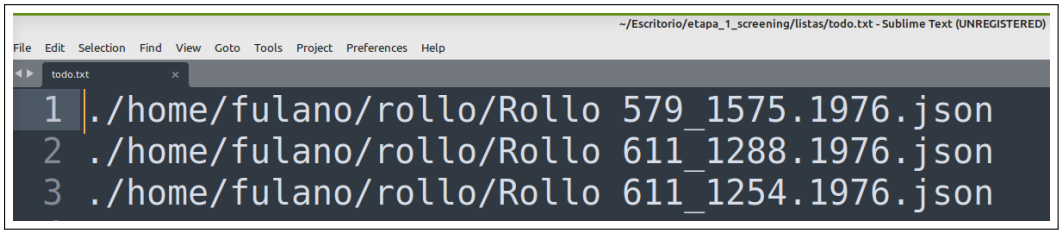


Figura A.4: Archivo `todo.txt` que contiene las rutas a los archivos `JSON`.

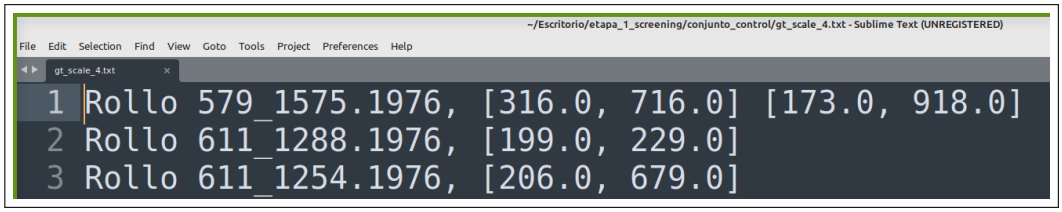


Figura A.5: Archivo de *ground-truth* generado a partir de los archivos `JSON` en el directorio `home`, contiene las coordenadas de los centros de los sellos asociados a cada imagen.

Figura A.6: Directorio asociado a la ejecución de la figura A.3. El nombre lleva la fecha y hora de ejecución (19/04/2024 a las 13:25:42), y el tipo de plantilla utilizado (solo sellos).

desde la terminal de comandos y cumple la función de registrar etiquetas para calcular métricas de evaluación. Contiene un archivo de texto como el de la figura A.5 con los centros de los sellos asociados a cada documento, los cuales fueron extraídos de los archivos `JSON` asociados a cada documento. Cabe recordar que estos fueron previamente etiquetados por alumnos de la FIC mediante la aplicación *Label Me*.

Por último, el directorio `salidas_NCC` se genera automáticamente y contiene los resultados del detector representados en la figura A.6. En este directorio se almacenan los siguientes resultados referidos al detector:

1. `debugeo`: Carpeta opcional para observar resultados intermedios y posibles casos de falsos negativos.
2. `escala`: Para cada escalado aplicado y umbral seleccionado, se genera un archivo `txt` que contiene las coordenadas de los picos de correlación hallados con el algoritmo para cada documento A.7. A su vez, se presenta la curva *Presicion-Recall* a modo de evaluar el desempeño del detector.
3. `logs.txt`: Se registra información sobre parámetros utilizados (umbral, escalado, fecha y hora de inicio de ejecución del algoritmo). También informa en qué documentos han sido encontrado sellos y con qué valor de correlación.

La carpeta `debugeo` se observa en la figura A.9 y contiene (en caso de utilizar el parámetro `debug=True`):

- `falsos_negativos`: Se almacenan imágenes de los documentos cuyos sellos no lograron ser detectados con el algoritmo. Esto involucra tanto a las

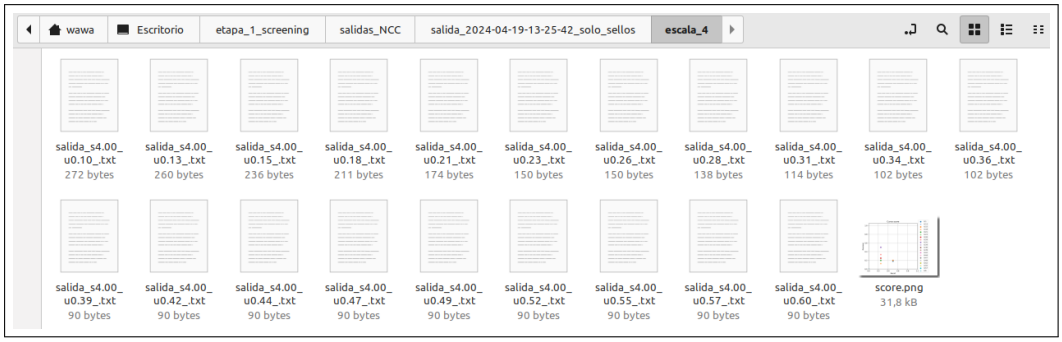


Figura A.7: Directorio con los archivos generados para cada umbral de detección (20 umbrales por defecto). En cada uno de estos se encuentran los centros de las detecciones para cara imagen (ver figura A.8). La comparación de estos archivos y el archivo *ground-truth* de la figura A.5 genera el archivo de `score.png`

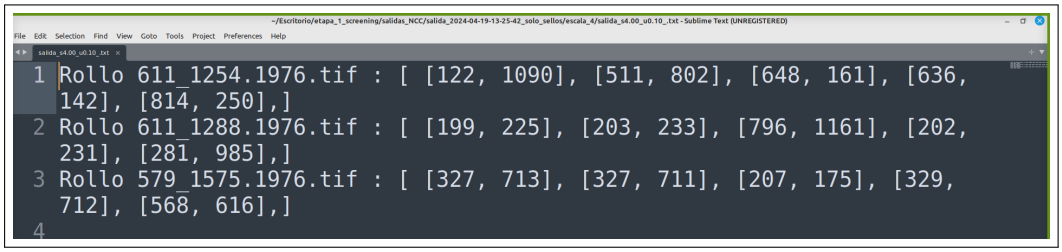


Figura A.8: Archivo generado para el umbral 0.10. Este asocia a cada imagen los centros de las detecciones halladas.

imágenes de 'debugeo' para el menor umbral (asumiendo que si un sello no es detectado con el menor umbral, entonces no se detectará con umbrales mayores) como también a las imágenes originales y sus respectivos `JSON`.

- `selloide`: Ilustra las imágenes de las plantillas utilizadas, representando el resultado de los selloides generados (en caso de ser utilizados).
- Imágenes de los documentos con las detecciones halladas y las del *ground-truth* marcadas sobre los mismos. Estas ilustraciones estarán escaladas según el parámetro `s` escogido.

Luego, la carpeta `sellos_extraidos` almacena todas las detecciones extraídas de los documentos en caso de ejecutar el programa con `-es=True`. Este parámetro es fundamental si se desea continuar con el proceso de clasificación de sellos.

Por último, en la figura A.1 se observa el directorio `base_con_basura_sin_copias` (ver figura A.10) y el archivo `base_sintetica.py`. Esta funcionalidad es opcional para el caso en que se desee crear una base sintética, a modo de evaluar el desempeño del algoritmo implementado. Si se ejecuta el archivo `.py` se generan (a partir de los sellos del directorio) imágenes sintéticas en la carpeta `home`.

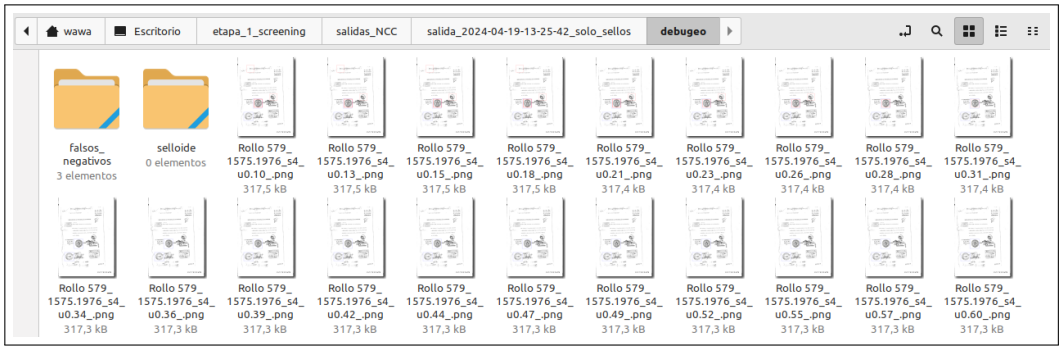


Figura A.9: Directorio debugeo.

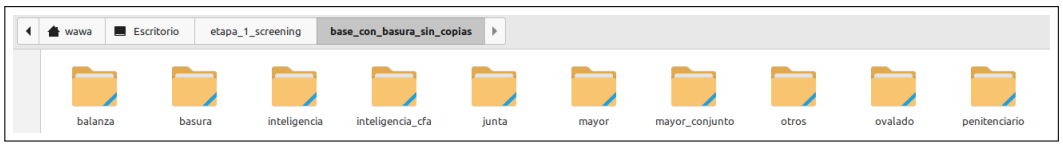


Figura A.10: Base de sellos. Contiene una carpeta por cada clase de sello con sus respectivas muestras.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice B

Funcionamiento del clasificador

En esta sección se detalla el procedimiento correspondiente al proceso de clasificación de la etapa 2 (introducido en el capítulo 4). Se definen librerías y funciones utilizadas en la arquitectura, se indican los pasos necesarios para llevar a cabo la clasificación sobre nuevas categorías y se muestra la estructura de directorios. Además se describe el entorno de trabajo utilizado.

Entorno de trabajo - Google Colab

En esta sección se encuentra todo aquello relativo al hardware y las plataformas utilizadas para desarrollar y entrenar la red.

Es importante señalar que, para llevar a cabo el proceso de implementación de forma ágil, el entorno de trabajo utilizado en esta etapa requiere componentes tales como GPU y memoria RAM. Una cantidad suficiente de RAM facilita la carga y manipulación de grandes conjuntos de datos durante el preprocesamiento y la fase de entrenamiento de los modelos. Por otro lado, el acceso a GPU acelera significativamente el tiempo de entrenamiento de los modelos gracias a su velocidad de procesamiento.

En vista de las limitaciones de estos componentes en los dispositivos locales accesibles, se propone trabajar inicialmente en el entorno de *Google Colab*. Un entorno de desarrollo alojado en los servidores de Google que permite escribir, ejecutar y compartir código en Python de manera gratuita. A su vez, proporciona acceso gratuito a recursos de computación, incluidas GPUs, lo que lo hace útil para la etapa de diseño e implementación de la arquitectura de la red. Esta decisión está alineada con el objetivo de desarrollar un prototipo que sea indicativo de la efectividad del modelo en un entorno de producción.

A pesar de las ventajas descritas, la versión libre presenta algunas dificultades tales como la capacidad de memoria RAM proporcionada (12 GB), la cual limitará la

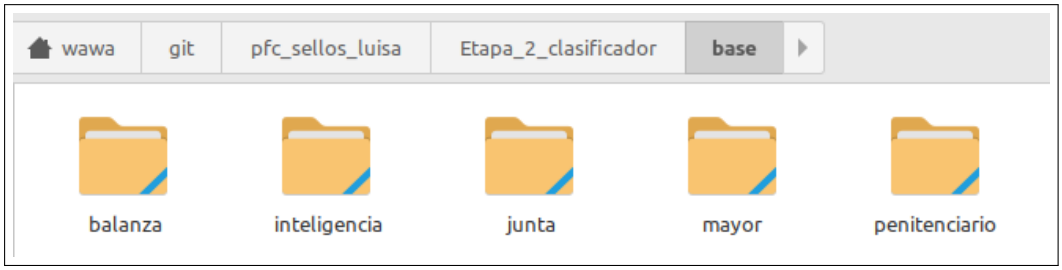


Figura B.1: Imagen ilustrativa de la disposición del directorio base con los datos de entrenamiento: sellos agrupados en carpetas según categorías.

capacidad de entrenamiento y el manejo de grandes cantidades de datos. Por otra parte, el uso de GPU también es limitado si es usa excesivamente en un corto período de tiempo. Esto no solo retrasa nuestra capacidad de manejar grandes volúmenes de datos, sino que también ralentiza el proceso de implementación.

La base de datos es almacenada en *Google Drive*, ofreciendo así una solución de almacenamiento compatible con el entorno a utilizar y accesible desde cualquier ubicación con conexión a internet.

Estructuración de archivos y carpetas

En primer lugar, es necesario contar con un conjunto de datos de entrenamiento etiquetados. Estos son esenciales ya que permiten al modelo aprender, pudiendo tomar decisiones en base a nuevos datos no vistos previamente.

Los datos se almacenan en el directorio `base` y contienen una carpeta por cada clase con las muestras de sellos correspondientes. En la figura B.1 se ilustra el directorio con algunas clases representativas.

La carpeta `base` es la única necesaria para ejecutar el archivo `clasificador.py`, el cual se encarga de entrenar el modelo de aprendizaje y generar resultados en base a los datos de prueba reservados para la evaluación del modelo. Estos resultados se generan dentro de una carpeta en el directorio llamado `Entrenamiento` que contiene los siguientes resultados (ver figura B.2):

- `clasificador.h5`: Archivo con el modelo entrenado listo para ser utilizado en un entorno de producción, para predecir clases de sellos.
- `confusion.png`: Matriz de confusión generada. Compara las predicciones de los datos de prueba (no utilizados para el entrenamiento del modelo) con las verdaderas etiquetas de los mismos.
- `incorrectas.png`: Imagen ilustrativa de ejemplos clasificados incorrectamente por el modelo.

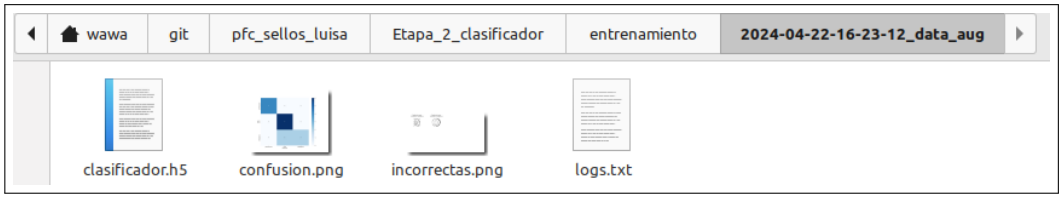


Figura B.2: Imagen ilustrativa del directorio `Entrenamiento` con los resultados generados luego de ejecutar `clasificador.py`

Época	Velocidad	Loss	Accuracy	Val_loss	Val_accuracy
1	3s 16ms/step	1.1815	0.7181	0.4691	0.8152
2	2s 14ms/step	0.3271	0.8824	0.4018	0.8655
3	2s 12ms/step	0.2159	0.9286	0.3566	0.8765
4	2s 12ms/step	0.1409	0.9538	0.3137	0.8902
5	1s 11ms/step	0.0992	0.9707	0.3109	0.8930
6	1s 11ms/step	0.0818	0.9751	0.5267	0.8390
7	1s 11ms/step	0.0721	0.9780	0.4604	0.8628
8	1s 10ms/step	0.0599	0.9817	0.3756	0.8884
9	1s 11ms/step	0.0596	0.9824	0.3503	0.8875
10	1s 11ms/step	0.0480	0.9854	0.4091	0.8838

Tabla B.1: Evaluación del modelo utilizando la técnica de *autoencoder*: Visualización de las métricas sobre los conjuntos de entrenamiento y validación en función de las épocas.

- `logs.txt`: Destinado a guardar los hiperparámetros utilizados y la fecha finalización de la ejecución.

Ensayos

A continuación se presentan resultados obtenidos sobre una selección de ensayos realizados, a partir de los cuales hemos tomado decisiones con respecto a qué técnicas incorporar al modelo final.

En la tabla B.1 se visualiza el resultado en función de las épocas de implementar *autoencoders* sobre la arquitectura de redes neuronales descrita en el capítulo 4.

Luego, en la tabla B.2 se muestran los resultados para la implementación del modelo con aumentado de datos, alcanzando un 95 % de accuracy en validación.

La tabla B.3 contiene la evolución de las métricas a lo largo de 20 épocas, utilizando los hiperparámetros seleccionados con `Grid Search`.

Clasificar nuevas categorías

Siguiendo la premisa establecida al comienzo del presente informe, el enfoque de este proyecto está diseñado para ser modificado y mejorado para futuras investi-

Época	Velocidad	Loss	Accuracy	Val_loss	Val_accuracy
1	69s 627ms/step	1.7279	0.3918	1.0450	0.5466
2	67s 618ms/step	1.2393	0.5076	0.9161	0.6480
3	68s 628ms/step	1.1450	0.5464	1.0139	0.6026
4	68s 632ms/step	0.9402	0.6294	0.4808	0.8287
5	64s 592ms/step	0.7375	0.7169	0.5162	0.8298
6	67s 615ms/step	0.6039	0.7796	0.5064	0.8462
7	43s 394ms/step	0.4857	0.8207	0.3227	0.9091
8	36s 328ms/step	0.4299	0.8458	0.2789	0.9056
9	36s 331ms/step	0.3410	0.8880	0.2234	0.9347
10	36s 331ms/step	0.3497	0.8755	0.1832	0.9394
11	36s 331ms/step	0.2853	0.9050	0.1622	0.9534
12	36s 335ms/step	0.2900	0.9006	0.1213	0.9569
13	36s 334ms/step	0.2309	0.9248	0.2046	0.9324
14	35s 321ms/step	0.2514	0.9227	0.1439	0.9464
15	36s 333ms/step	0.2147	0.9318	0.1769	0.9452
16	36s 333ms/step	0.2190	0.9271	0.1231	0.9592
17	35s 324ms/step	0.1918	0.9327	0.1878	0.9499
18	35s 326ms/step	0.2028	0.9338	0.1057	0.9662
19	34s 318ms/step	0.1716	0.9446	0.1187	0.9627
20	35s 324ms/step	0.1635	0.9446	0.1166	0.9569

Tabla B.2: Evaluación del modelo para los hiperparámetros encontrados mediante *Grid Search* y entrenando con data augmentation: Visualización de las métricas sobre los conjuntos de entrenamiento y validación en función de las épocas.

gaciones. Esta flexibilidad cumple con el propósito de adaptarse a las necesidades cambiantes y las exigencias de mejora continua.

Una posible modificación es la elección de los tipos de sellos que interesen clasificar. Cabe recordar que a lo largo del proyecto se utilizaron cinco clases de sellos a modo representativo. Dependiendo de qué categorías se desee entrenar, será necesario almacenar las muestras en una carpeta con el nombre de la clase correspondiente, dentro del directorio `base` como se ve en la figura B.1.

Época	**Velocidad**	**Loss**	**Accuracy**	**Val_loss**	**Val_accuracy**
1	58s 135ms/step	0.3106	0.8974	0.0749	0.9744
2	59s 138ms/step	0.0378	0.9886	0.0731	0.9767
3	57s 133ms/step	0.0091	0.9965	0.0752	0.9814
4	62s 144ms/step	0.0041	0.9985	0.0553	0.9860
5	59s 138ms/step	0.00020152	1.0000	0.0594	0.9883
6	60s 140ms/step	0.00002022	1.0000	0.0603	0.9872
7	59s 137ms/step	0.00000955	1.0000	0.0606	0.9883
8	61s 141ms/step	0.00000671	1.0000	0.0617	0.9883
9	59s 137ms/step	0.00000491	1.0000	0.0632	0.9883
10	58s 136ms/step	0.00000368	1.0000	0.0645	0.9883
11	60s 140ms/step	0.00000280	1.0000	0.0661	0.9872
12	65s 152ms/step	0.00000215	1.0000	0.0676	0.9872
13	102s 238ms/step	0.00000167	1.0000	0.0693	0.9872
14	72s 167ms/step	0.00000130	1.0000	0.0707	0.9872
15	63s 147ms/step	0.00000101	1.0000	0.0720	0.9872
16	89s 208ms/step	0.00000079	1.0000	0.0737	0.9872
17	93s 217ms/step	0.00000062	1.0000	0.0754	0.9872
18	104s 242ms/step	0.00000049	1.0000	0.0770	0.9872
19	95s 221ms/step	0.00000038	1.0000	0.0788	0.9860
20	92s 214ms/step	0.00000030	1.0000	0.0804	0.9860
Accuracy on test data: **98.23 %**					

Tabla B.3: Evaluación del modelo para los hiperparámetros encontrados mediante *Grid Search*: Visualización de las métricas sobre los conjuntos de entrenamiento y validación en función de las épocas. En la última fila se observa el desempeño sobre el conjunto de prueba.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice C

Integración

En esta sección se explica el funcionamiento del algoritmo que integra las etapas anteriores. En esta etapa, a partir de una imagen de entrada, se genera un archivo en formato `CSV` que indica qué sellos se encuentran en cada imagen.

La composición de directorios necesaria es similar a las de las etapas anteriores, se puede observar en la figura C.1. La carpeta `base` cumple la misma función que la explicada en el apéndice B, contiene la base de datos clasificada para entrenar un modelo de clasificación. Para esta etapa, además de los directorios que se observan en la figura B.1 es necesario que existan los directorios `basura` y `otros`.

Por otra parte, la carpeta `filtro_basura` contiene el modelo de clasificación binaria, este modelo junto con el código `entreno_filtro.py` son las nuevas incorporaciones para esta etapa. Si se desea entrenar un nuevo filtro, se debe ejecutar el algoritmo `entreno_filtro.py`, generando una carpeta dentro de `filtro_basura` con el nuevo modelo y matriz de confusión (como se observa en la figura C.2). Luego, para utilizar el nuevo modelo generado, se debe reemplazar el modelo existente en `filtro_basura`.

Las carpetas `home` y `sellos_a_detectar` son las mismas que fueron detalladas en el apéndice A. Por último, la carpeta `modelo` contiene el modelo de clasificación

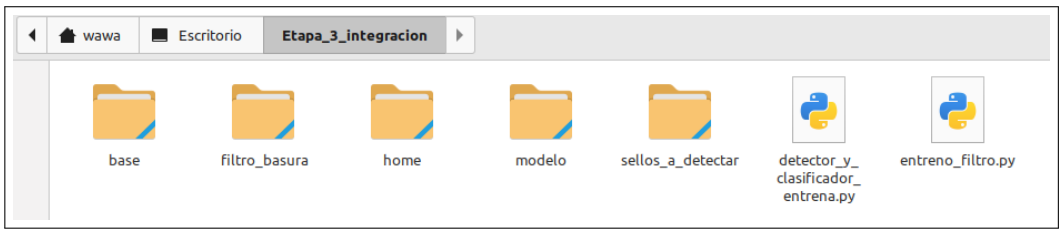


Figura C.1: Estructura de directorios para la detección y clasificación de imágenes.

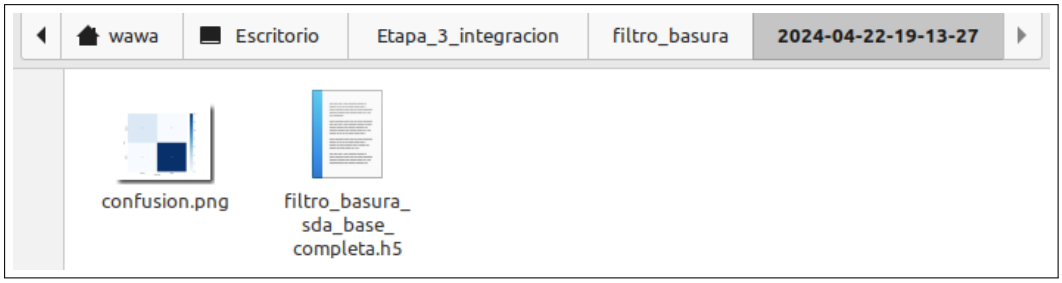


Figura C.2: Archivos generados luego de ejecutar `entreno_filtro.py`.

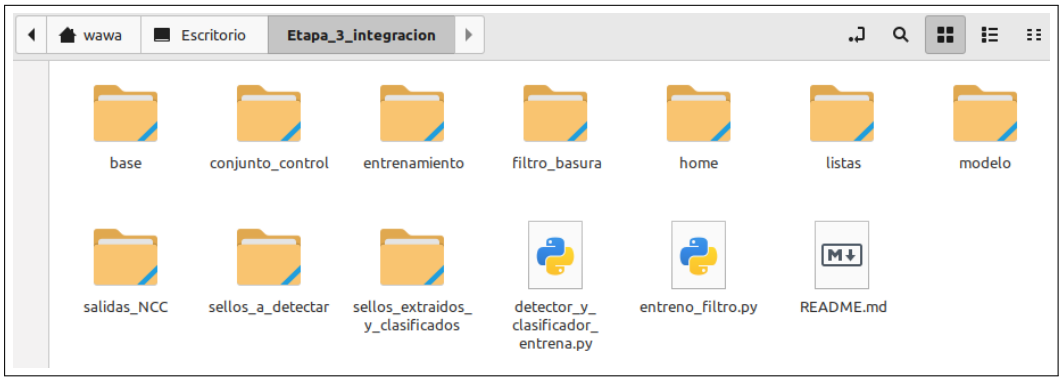


Figura C.3: Archivos generados luego de entrenar un nuevo modelo y clasificar con este.

entrenado que se utiliza en caso de no entrenar uno nuevo para clasificar. Es decir, si se cuenta con un modelo de clasificación, se debe reemplazar el existente en el directorio por el nuevo modelo a utilizar.

Los directorios generados luego de correr `detector_y_clasificador_entrena.py` se pueden observar en la figura C.3. Estas carpetas ya se observaron en capítulos anteriores salvo la carpeta `modelo`, esta cumple con la misma función que la carpeta `entrenamiento` del apéndice B. En esta se guarda el modelo a utilizar durante la clasificación, así como también las carpetas ordenadas por fecha (en las que se guardan los nuevos modelos de clasificación entrenados junto con su matriz de confusión y otra información del entrenamiento).

Referencias

[1] Cruzar. página oficial. `https://cruzar.edu.uy/`, Abril 2024.

[2] Luisa. página oficial. `https://mh.udelar.edu.uy/luisa/`, Abril 2024.

[3] Partha Pratim Roy Alireza Alaei and Umapada Pal. Logo and seal based administrative document image retrieval: A survey. *Computer Science Review*, 2016.

[4] Matthias Schulte-Austum Zhe Li and Martin Neschen. Fast logo detection and recognition in document images. *2010 20th International Conference on Pattern Recognition Pattern Recognition (ICPR), 2010 20th International Conference on. :2716-2719 Aug, 2010*, 2010.

[5] Mathieu Delalandre The Anh Pham and Sabine Barrat. A contour-based method for logo detection. *2011 International Conference on Document Analysis and Recognition Document Analysis and Recognition (ICDAR), 2011 International Conference on. :718-722 Sep, 2011*, 2011.

[6] Mathieu Delalandre Alireza Alaei and Nathalie Girard. Logo detection using painting based representation and probability features. *2013 12th International Conference on Document Analysis and Recognition Document Analysis and Recognition (ICDAR), 2013 12th International Conference on. :1235-1239 Aug, 2013*, 2013.

[7] Dimosthenis Karatzas Marçal Rusiñol, Vincent Poulain D'Andecy and Josep Lladós. Classification of administrative document images by logo identification. *Graphics Recognition. New Trends Challenges; 2013, p49-58, 10p*, 2013.

[8] Aurélien Géron. *Hands-on Machine Learning with Scikit-Learn, Keras TensorFlow*. O'Reilly, 2019.

[9] D. Ershovaa A. Gayera and V. Arlazarov. Fast and accurate deep learning model for stamps detection for embedded devices. *Pattern Recognition Image Analysis; Dec2022, Vol. 32 Issue 4, p772-779, 8p*, 2022.

[10] Muhammad Imran Malik Faisal Shafait Paul Lukowicz Sheraz Ahmed Junaid Younas, Muhammad Zeshan Afzal. D-star: A generic method for stamp segmentation from document images. *Proceedings of the 8th International Conference on Computer Recognition Systems CORES 2013; 2013, p383-392, 10p*, 2013.

[11] Balasubramanian Raman Priyanka Singh and Partha Pratim Roy. Detection of seal and signature entities with hierarchical recovery based on watermark self embedding in tampered digital documents. *Displays*, 2018.

[12] Paweł Forczmański and Andrzej Markiewicz. Low-level image features for stamps detection and classification. *Proceedings of the 8th International Conference on Computer Recognition Systems CORES 2013; 2013, p383-392, 10p*, 2013.

[13] Paweł Forczmański and Andrzej Markiewicz. Stamps detection and classification using simple features ensemble. *Mathematical Problems in Engineering. 8/30/2015, Vol. 2015, p1-15. 15p.*, 2015.

[14] Umapada Pal Partha Pratim Roy and Josep Lladós. Document seal detection using ght and character proximity graphs. *Pattern Recognition. Jun2011, Vol. 44 Issue 6, p1282-1295. 14p.*, 2011.

[15] Panpan Tang and Yuxin Peng. Exploiting distinctive topological constraint of local feature matching for logo image recognition. *Neurocomputing*, 2017.

[16] Wilhelm Burger and Mark J. Burge. *Digital Image Processing.* Springer, 2016.

[17] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing.* Pearson Prentice Hall, 2017.

[18] Pawel Forczmański and Dariusz Frejlichowski. Classification of elementary stamp shapes by means of reduced point distance histogram representation. *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR) ICDAR Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on. 01:248-253 Nov, 2017*, 2017.

[19] Kai Briechle and Uwe D. Hanebeck. Template matching using fast normalized cross correlation. `https://isas.iar.kit.edu/pdf/SPIE01_BriechleHanebeck_CrossCorr.pdf`.

[20] J.P. Lewis. Fast template matching. `http://scribblethink.org/Work/nvisionInterface/vi95_lewis.pdf`.

[21] Jonathan Masci, Ueli Meier, Dan Cire¸san, and J¨urgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. `https://people.idsia.ch/~ciresan/data/icann2011.pdf`.

[22] Andres Arobba, Guillermo Sosa, and Ignacio Ramirez. Repositorio de gitlab. `https://gitlab.fing.edu.uy/andres.arobba/pfc_sellos_luisa`, Abril 2024.

Esta página ha sido intencionalmente dejada en blanco.

Índice de tablas

Esta página ha sido intencionalmente dejada en blanco.

Índice de figuras

Índice de figuras

Esta es la última página.
Compilado el lunes 29 julio, 2024.