UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Análisis de anotaciones de partidas de ajedrez

Trabajo Fin de Grado

Grado en Ciencia de Datos

AUTOR/A: Bono Monreal, Alberto

Tutor/a: Martínez Hinarejos, Carlos David

CURSO ACADÉMICO: 2022/2023

A mi iaia, por hacer de un juego de verano,
una pasión para toda una vida.

Resum

En les partides oficials a ritme clàssic d'escacs cal apuntar les jugades en una fulla d'anotacions (una fulla que proporciona l'organització del torneig) de manera manual, perquè està totalment prohibit l'entrada de dispositius electrònics a les sales de joc per a evitar ajudes externes. Aquest full d'anotacions serveix perquè, en acabar la partida, es puguin reproduir movent en el tauler (virtual o físic) les jugades apuntades. Aquest procés pot aplegar a ser llarg i tediós. La idea a desenvolupar implica una automatització d'aquest procés, en aquest cas, després de realitzar una foto a una fulla d'anotacions es reproduiria automàticament la partida en un tauler virtual, sense haver de moure l'usuari jugada a jugada en el seu ordinador o mòbil. Aquest procés requereix de, a través, d'una tecnologia de *Computer Vision* i *HandWritten Character Recognition* analitzar el format de la fulla d'anotacions, per posteriorment extraure les jugades escrites a mà i reproduir-les. En aquest procés de reconeixement es poden implementar restriccions per la situació de la partida, sent això últim la novetat respecte a altres treballs sobre reconeixement de text manuscrit. Per a això, s'ha de tindre en compte que la jugada reconeguda ha de ser legal en el context dels escacs en la posició actual del tauler.

Paraules clau: Escacs, Planella, *Computer Vision, HandWritten Character Recognition*

Resumen

En las partidas oficiales a ritmo clásico de ajedrez hay que apuntar las jugadas en una planilla (una hoja que proporciona la organización del torneo) de manera manual, pues está totalmente prohibido la entrada de dispositivos electrónicos a las salas de juegos para evitar ayudas externas. Esta planilla sirve para que, al acabar la partida, se puedan reproducir moviendo en el tablero (virtual o físico) las jugadas apuntadas. Este proceso puede llegar a ser largo y tedioso. La idea a desarrollar implica una automatización de este proceso; en este caso, tras realizar una foto a una planilla, se reproduciría automáticamente la partida en un tablero virtual, sin tenir que mover el usuario jugada a jugada en su ordenador o móvil. Este proceso requiere de, a través de una tecnología de *Computer Vision* y *Hand-Written Character Recognition*, analizar el formato de las planillas, para posteriormente extraer las jugadas escritas a mano y reproducirlas. En este proceso de reconocimiento se pueden implementar restricciones por la situación de la partida, siendo esto último la novedad con respecto a otros trabajos existentes sobre reconocimiento de texto manuscrito. Para ello, se debe tener en cuenta que la jugada reconocida debe ser legal en el contexto del ajedrez en la posición actual del tablero.

Palabras clave: Ajedrez, Planilla, *Computer Vision, HandWritten Character Recognition*

Abstract

In official games at a classic chess rhythm, the players have to write down the moves on a scoresheet manually (provided by the tournament organization), since the entry of electronic devices into the game rooms is totally prohibited to avoid external aid. This spreadsheet is used to reproduce the movements on a virtual or physical board at the end of the game. The idea to be developed implies the automatization of this process; in this case, after taking a photo of the scoresheet, the game would automatically be reproduced in a virtual board, without the users having to do every movement in their computer or smartphone. This process requires to analyze the format of the scoresheet using Computer Vision y HandWritten Character Recognition to then extract the handwritten movements and reproduce them. In this recognition process, restrictions can be applied according to the game's situation which is the innovation with respect to other works. For this purpose, every recognized movement must be legal in the context of chess in the current board position.

Key words: Chess, Scoresheet, Computer Vision, HandWritten Character Recognition

ÍNDICE

ÍNDICE DE FIGURAS

CAPÍTULO 1
Introducción

El 15 de mayo de 1495 se produjo a manos de Francesch Vicent la primera publicación relacionada con el juego del ajedrez. Bajo el nombre "Llibre dels jochs partits dels schacs en nombre de 100", Francesch Vicent proclamó la ciudad de Valencia como cuna del ajedrez moderno [1].

El ajedrez es un juego popular en todo el mundo, presente en muchas casas a través de ese tablero que de alguna manera es un baúl de los recuerdos. Transmitido de generación en generación, de abuelos y padres a hijos y nietos, en esas tardes que para muchos han marcado el deparar de toda una vida. Su popularidad ha sido tasada por diversas fuentes en más de 600 millones de personas, haciéndolo uno de los juegos más conocidos en el mundo. Sin embargo, el número de personas federadas es ampliamente inferior debido a que esta parte del ajedrez se orienta más a la competición.

Muchos aficionados al ajedrez recuerdan la primera victoria de una máquina sobre un humano (Deep Blue vs Gary Kasparov en 1997 [2]), todo un hecho histórico que marcó el devenir de este deporte. Desde aquel entonces, tecnología y ajedrez han ido ligados, mejorando la calidad de las partidas entre aficionados y alcanzando niveles inimaginables en encuentros donde se enfrentan las máquinas mejor programadas.

El ajedrez es un juego catalogado como juego de información completa, pues ambos oponentes cuentan con toda la información posible durante el transcurso de la partida. La cantidad de posibles partidas es superior al orden de átomos en el mundo, haciendo de este un juego extremadamente complejo y sin solución a día de hoy.

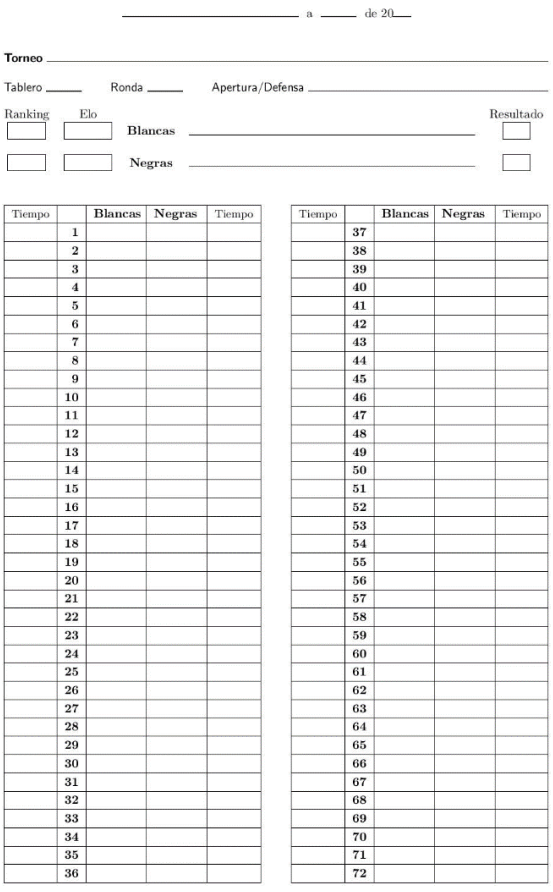
En el ajedrez **federado y competitivo** existen todavía tradiciones ampliamente arraigadas y lejos de la tecnología.

Toda partida de ajedrez debe ser jugada lejos de cualquier tipo de dispositivo electrónico. Esto se debe a que cualquier persona con conexión a internet dispone de acceso a lo que en ajedrez se conoce como "módulos". Estos módulos son programas como Deep Blue, de un nivel extremadamente superior al de cualquier humano, que podrían ser utilizados durante una partida adulterando la competición e incurriendo en una falta grave al deporte.

Toda partida de ajedrez debe ser anotada en una hoja proporcionada por la organización del torneo denominada planilla. En esta planilla se anota obligatoriamente el nombre del oponente y todos los movimientos de las piezas blancas y negras. De modo voluntario se puede añadir como información extra el lugar de juego, la fecha y la ronda. Esta planilla es recogida por la organización al finalizar cada partida, quedándose los jugadores con una copia/calco de ella. Esta copia sirve para, al finalizar la partida, poder analizar y estudiar las diferentes posiciones que haya o hubiesen podido surgir durante el transcurso de la misma. La representación gráfica de esta planilla podría ser la Figura 1.

Todo tablero de ajedrez es de 8x8, un total de 64 casillas, 32 blancas y 32 negras, divididas en 8 columnas que van de la "a" a la "h" y en 8 filas enumeradas del 1 al 8. Esto permitió la creación de un sistema internacional y adaptado a todos los idiomas para que todas las jugadas sean escritas de la misma manera. Toda jugada sigue el siguiente orden. En primer lugar, se anota la pieza que se ha movido (caballo, alfil, torre, dama o rey), a excepción del peón, que no se anota. Posteriormente, la columna a la que se ha movido y por último la fila. Un ejemplo de movimiento podría ser Th8, que indicaría que la torre (T) se ha movido a la columna h fila 8.

Y es a través de esta planilla y su contenido, donde nace la creación del enlace (uno entre tantos posibles) entre la ciencia de los datos y el ajedrez.



_____ a ____ de 20____

Torneo _____

Tablero ____ Ronda ____ Apertura/Defensa _____

Ranking Elo Resultado

Blancas _____

Negras _____

Tiempo		Blancas	Negras	Tiempo
	1			
	2			
	3			
	4			
	5			
	6			
	7			
	8			
	9			
	10			
	11			
	12			
	13			
	14			
	15			
	16			
	17			
	18			
	19			
	20			
	21			
	22			
	23			
	24			
	25			
	26			
	27			
	28			
	29			
	30			
	31			
	32			
	33			
	34			
	35			
	36			

Tiempo		Blancas	Negras	Tiempo
	37			
	38			
	39			
	40			
	41			
	42			
	43			
	44			
	45			
	46			
	47			
	48			
	49			
	50			
	51			
	52			
	53			
	54			
	55			
	56			
	57			
	58			
	59			
	60			
	61			
	62			
	63			
	64			
	65			
	66			
	67			
	68			
	69			
	70			
	71			
	72			

Figura 1: Ejemplo de planilla

La planilla, aun siendo una hoja utilizada en un contexto muy específico, no deja de ser un folio con una cierta estructura donde se puede encontrar texto manuscrito y que se puede representar como una imagen. Una imagen es un conjunto de datos visuales compuestos por una gran cantidad de píxeles. Cada píxel representa un punto discreto en una cuadrícula espacial, así como un valor de intensidad luminosa. Estos valores de intensidad se utilizan para representar los colores y las características visuales de la imagen.

La relación entre las imágenes y la ciencia de datos radica en la capacidad de extraer información relevante y significativa a partir de los datos visuales, en este caso jugadas que nos permitan reproducir una partida. Los algoritmos de procesamiento de imágenes y visión por computadora permiten analizar y comprender el contenido de las imágenes, como identificar objetos o reconocer rostros [22]. Estos algoritmos se basan en técnicas de aprendizaje automático [3] donde se entrenan modelos con grandes conjuntos de datos de imágenes para reconocer y comprender automáticamente las características visuales.

En concreto, el *HandWritten Character Recognition* (HWCR) es un campo de investigación muy activo, que combina ideas tanto de la visión por computador como del procesamiento del lenguaje natural [25]. A diferencia del reconocimiento de texto impreso, la escritura a mano presenta una serie de características únicas que hacen que la tarea sea mucho más difícil que el reconocimiento caracteres a ordenador (OCR) [30].

El reto que supone el reconocimiento de la escritura manuscrita radica principalmente en la gran variabilidad que puede presentar la escritura de una persona a otra. Para ello, además de decodificar visualmente una imagen en una secuencia de caracteres, varios trabajos adoptan modelos lingüísticos para reducir esta ambigüedad innata de los caracteres manuscritos, haciendo uso de información contextual y semántica [23].

En general, el diseño de un sistema de aprendizaje eficaz y generalizable es un reto constante, puesto que es extremadamente complejo modelar los diferentes tipos de escritura. Las redes neuronales (NN) [3], entre una variedad de otros sistemas de aprendizaje, se han utilizado para el reconocimiento de la escritura a mano desde los inicios de la disciplina, con un alcance que va desde subtareas más simples, como el reconocimiento de un solo dígito, hasta la transcripción de libros enteros. Tras el auge del aprendizaje profundo y sus aplicaciones, los últimos avances en HWTR están monopolizados por las redes neuronales profundas (DNN) [16].

1.1 Motivación

La motivación de este proyecto nace de la necesidad de automatizar un proceso que lleva siendo manual desde hace más de 200 años. Hoy en día, las grandes industrias trabajan a diario para poder automatizar procesos, principalmente tareas simples y repetitivas. Y no sólo en industria, sino también en deportes como el ajedrez.

Esta tarea simple y repetitiva es el caso de lo que entre ajedrecistas se llama como "pasar la partida". Esta tarea consiste en reproducir (actual y principalmente en un software de ajedrez con un tablero virtual) una partida que haya sido anotada durante el transcurso de un torneo de ajedrez.

Esto se hace por dos motivos principales; uno, para mejorar el nivel ajedrecístico analizando las propias partidas en busca de posibles errores y mejoras. Este proceso suele realizarse con la ayuda de los módulos anteriormente comentados y se trata de una motivación personal.

Y dos, como método para compartir información y conocimiento. Son muchas las organizaciones que deciden subir a internet todas las partidas jugadas en su torneo con el objetivo de que todo el mundo pueda disfrutarlas. Un torneo con formato suizo, que es el más habitual entre los torneos, a 7 rondas con 100 participantes, supondría transcribir manualmente 350 partidas. Esto, además de tedioso, requiere de una persona que se dedique exclusivamente a realizarlo, incurriendo en un gasto de personal con el que federaciones, clubs y personas no suelen contar.

Por ello, a través de la tecnología HWCR e incluyendo aspectos técnicos de ajedrez en el proceso, se podría pasar de una duración de unos 15-20 minutos por partida a lo que simplemente cuesta hacer una foto de una planilla. Esto supondría una mejora en el proceso y dotaría de una mayor capacidad para poder compartir partidas de manera más rápida y efectiva.

1.2 Objetivos

El proyecto se divide en dos grandes objetivos;

1. Reconocimiento de la estructura de la planilla mediante el uso de técnicas clásicas de tratamiento de imágenes.
2. Análisis y predicción de las jugadas escritas a mano mediante el uso de modelos de aprendizaje automático.

Entrando en detalle, el primer punto se refiere a la capacidad del software desarrollado de reconocer la estructura de una planilla, es decir, identificar el lugar donde se escriben las jugadas, sabiéndolo diferenciar de otros espacios de la planilla como puede ser la columna donde indica el número de jugada o la columna para especificar el tiempo que queda en el reloj.

El segundo punto hace referencia al uso de modelos pre-entrenados para reconocer texto manuscrito incorporando en el proceso conocimiento ajedrecísitico. Es decir, el texto predicho tiene que ser una jugada de ajedrez legal y posible teniendo en cuenta la situación del tablero.

1.3 Impacto esperado

La automatización de procesos monótonos representa un avance tecnológico significativo, pues permite eliminar errores debido al factor humano, que suele ser considerablemente mayor que el error de las máquinas. En el contexto de este proyecto, el desarrollo de un software de este tipo permitiría a miles de jugadores de ajedrez capturar todas sus partidas con un simple clic utilizando la cámara de sus dispositivos móviles. Y no solo eso, sino que también proporcionaría a las organizaciones y federaciones de ajedrez una mayor facilidad y una herramienta para exponer y compartir de manera transparente las partidas disputadas en sus torneos.

Dado el enfoque ambicioso de este proyecto y su total falta de similitudes existentes, su repercusión sería amplia, generando un gran interés entre miles de personas y abriendo las puertas a futuros proyectos y mejoras. La combinación de estas características, junto con un mercado inicial tan diverso, resulta en un impacto potencialmente inmenso para el proyecto. Incluso si la obtención del producto final presenta desafíos, la cantidad de opciones y áreas de investigación que se desprenden del proyecto representa una oportunidad única.

Este trabajo tiene el potencial de revolucionar el mundo del ajedrez mediante la aplicación de tecnologías innovadoras, además de poder asentar las bases para futuras investigaciones y desarrollos en el campo de la visión por computadora y el reconocimiento de caracteres en ajedrez. Por otra parte, al combinar estas áreas de estudio en un solo proyecto, se fomenta la interdisciplinariedad y la sinergia entre campos diversos, lo que a su vez puede inspirar nuevas aplicaciones en otros ámbitos. El trabajo trasciende el ámbito académico y se extiende a la comunidad ajedrecística y a la industria tecnológica, ofreciendo nuevas oportunidades de innovación y mejoras en el campo del análisis de partidas de ajedrez y la automatización de procesos relacionados.

1.4 Estructura del documento

El trabajo se ha organizado en capítulos, y dentro de ellos en secciones. Esta sección pretende mostrar el proceso realizado de forma concisa. A continuación, se realiza una breve explicación de cada capítulo que conforma el trabajo.

En el primer capítulo, llamado *Introducción,* se realiza una breve presentación de los temas relacionados con el trabajo. Primeramente, se hace un repaso e introducción al deporte del ajedrez, su historia y ciertas características que lo hacen particular, para posteriormente explicar la conexión que guarda con la ciencia de datos y más concretamente con el reconocimiento de texto manuscrito. En segundo lugar, se plantean los objetivos a conseguir en el trabajo para finalizar el capítulo con el impacto esperado en caso de conseguir estos objetivos.

En el segundo capítulo, *Estado del Arte*, se realiza un estudio del estado de los trabajos relacionados con el proyecto desde tres puntos de vista: desde el ajedrez y el papel que juega en el equipo hasta las técnicas clásicas como base del análisis de imágenes, pasando por las tecnologías más modernas como puede ser la librería de Python PyLaia [18].

En tercero capítulo, *Análisis del problema*, en primer lugar, se describe con un mayor grado de profundidad los objetivos a cumplir, profundizando en ellos, además de especificar la razón detrás de cada uno de los procesos. Posteriormente, se explica la base de datos de imágenes con la que se ha trabajado, las características generales de las imágenes, y cómo se ha realizado su etiquetado. En último lugar, y siendo este el principal aspecto desarrollado en este capítulo, la metodología del proyecto, basada principalmente en el constante proceso de mejora del software, empezando con unos resultados desilusionantes para acabar con unos prometedores.

En el cuarto capítulo, *Conocimiento extraído y Evaluación de Modelos*, se presentarán y evaluarán los diferentes modelos utilizados para la predicción del texto manuscrito. Al término de este capítulo, se incluirá una sección conclusiva en la cual se reflejará el conocimiento adquirido a través de los modelos, así como las necesidades y desafíos identificados en virtud de los resultados obtenidos.

En el quinto capítulo, *Futuros trabajos*, se plantearán todas las mejoras que deberían implementarse en el trabajo para poder comercializar la idea. Estas mejoras requieren de unos ciertos aspectos como conocimiento experto o ayuda externa. Cada una de estas mejoras está respaldada por una justificación lógica y sólida que se proporcionará de manera detallada.

En el sexto capítulo, *Conclusiones,* se procederá a realizar una recapitulación exhaustiva de los objetivos que se lograron satisfactoriamente y se proporcionará un resumen conciso de los resultados que se obtuvieron en cada uno de los procesos implementados en el desarrollo del trabajo.

El séptimo y último capítulo, *Legado y relación del trabajo desarrollado con los estudios cursados,* contendrá una explicación sucinta del legado que deja el proyecto y de las distintas competencias transversales que se han adquirido durante el grado y que se han aplicado efectivamente en este trabajo.

Para concluir, se añadirán dos apéndices que contienen información complementaria a la presentada en el cuerpo principal del documento. Estos apéndices están destinados a enriquecer la comprensión del lector y a proporcionar un contexto adicional que respalde y enriquezca las conclusiones extraídas.

CAPÍTULO 2
Estado del arte

A continuación, se describe una revisión del estado del arte referente a los tres grandes componentes de este proyecto.

El primero de ellos es el ajedrez. A pesar de tratarse de texto manuscrito cabe destacar la importancia de incluir conocimiento de ajedrez en el proceso y por ello es esencial revisar los trabajos relacionados con este campo.

En segundo lugar, el *HandWritten Character Recognition*. El reconocimiento preciso y confiable de caracteres manuscritos sigue siendo un desafío, debido a la variabilidad inherente en la forma y estilo de escritura de cada individuo. En una mayor capacidad de predicción se encuentran las tecnologías de *Optical Character Recognition* (OCR). Estas tecnologías cuentan con la facilidad de no encontrase sujetas al trazo o estilo de la letra y presentan grandes casos de éxito, por ejemplo, reconociendo matrículas de vehículos [4].

En tercer y último lugar, las técnicas clásicas de análisis de imágenes. Estas técnicas son la base de todo sistema de aprendizaje automático y mejora de visionado de imágenes. En este caso de estudio serán utilizadas para el reconocimiento de la estructura de la planilla donde se encuentran escritas las jugadas a predecir.

2.1 Ajedrez y ciencia de datos

El ajedrez lleva unido al mundo de la tecnología desde hace más de medio siglo, cuando en 1950 el matemático e informático Alan Turing creó el primer programa de ajedrez. Este programa no llegó hasta donde se esperaba, pues ningún ordenador de la época era capaz de soportarlo. No fue hasta 1957 cuando un empleado de IBM escribió el primer programa de ajedrez completamente decodificable por ordenador que funcionaba en un IBM 704 [5].

La gran mayoría de esfuerzo y dinero invertido por empresas en el sector del ajedrez ha ido dedicado a crear máquinas con un mayor nivel de juego que las ya existentes, dejando de lado otras oportunidades que ofrece este deporte. Este es el caso de Google, que en 2017 sacó al mercado un módulo llamado Alphazero. AlphaZero es una red neuronal que fue capaz de aprender a jugar al ajedrez únicamente enfrentándose contra sí misma llegando a ser capaz de vencer a los mejores jugadores del mundo en menos de 24 horas de entrenamiento. Actualmente es considerado uno de los mejores módulos que existen [6].

A pesar de ello, en los últimos años ha nacido una nueva tendencia, a raíz también del auge en este tipo de tecnologías, basada en fusionar *computer vision* y ajedrez.

C*omputer vision* es una disciplina que engloba el análisis de imágenes y vídeos desde el punto de vista informático, desglosando las imágenes en datos, más concretamente en píxeles. De esta disciplina surgen diferentes vertientes; en este caso, la más explorada hasta el momento relacionado con el ajedrez competitivo es el análisis de vídeo [43].

Actualmente, la retransmisión de torneos de ajedrez se realiza mediante un tablero electrónico especializado y utilizado únicamente con este fin. Este tablero cuenta con sensores para detectar la situación de las piezas en el tablero y enviarle la información a un ordenador al cual está conectado mediante corriente eléctrica. El ordenador en cuestión recibe la información y es una persona la encargada de actualizar la información recibida por el tablero en una página web para que el resto del mundo pueda seguir las partidas del torneo en *streaming*. Este proceso es rudimentario, además de necesitar de una tecnología basada en sensores bastante cara.

Por todo ello se decidió unir el análisis de vídeo y el ajedrez, empezando a surgir proyectos para el reconocimiento de tableros mediante el uso de una cámara. Esto facilitaría la retransmisión de partidas en línea, abarataría costes de producción y abriría nuevos campos, como la posibilidad de jugar *online* mediante un tablero físico.

Son diversos los equipos de trabajo que han luchado por conseguir este objetivo, y de ellos cabe destacar dos en concreto. El primero, la fundación de informática y ciencias de la decisión que consiguió en 2020 desarrollar un algoritmo con un 95% de certeza en detección del tablero y posición [7]. En segundo lugar, un trabajo individual realizado por Alper Karayaman [8] que permite el uso de un tablero físico para jugar *online* en la plataforma en *streaming open source* por excelencia, lichess.org[1].

Estos y otros proyectos relacionados no han sido puestos totalmente en práctica. En contadas ocasiones las organizaciones dan la oportunidad de probar este tipo de softwares, principalmente debido al desconocimiento unido a este tipo de tecnologías. Solo algunas de ellas han tenido la oportunidad de ser probadas, con gran éxito, como la desarrollada por el español Isaac Lozano [9] en el campeonato de España de aficionados en 2022.

Teniendo en cuenta lo anteriormente expuesto, es evidente los esfuerzos que se están realizando por mejorar, abaratar e implementar nuevas tecnologías en la retransmisión de partidas de ajedrez. Sin embargo, existen otros procesos en ajedrez que no están automatizados, y es aquí donde entra en juego la tecnología de reconocimiento de texto manuscrito.

2.2 HandWritten Character Recognition (HWCR)

En la situación actual, el reconocimiento de caracteres manuscritos se ha vuelto cada vez más importante debido a la creciente demanda de digitalización de documentos, automatización de tareas administrativas y mejora de la accesibilidad en dispositivos móviles. Además, la capacidad de reconocer y transcribir texto escrito a mano de manera precisa y eficiente tiene aplicaciones en diversas áreas, como el reconocimiento de formularios [10], la autenticación de firmas [19] o la lectura de jugadas anotadas en una planilla de ajedrez.

En términos de enfoques y algoritmos utilizados en el reconocimiento de caracteres manuscritos, ha habido un cambio significativo hacia técnicas basadas en el aprendizaje automático en los últimos años. Los enfoques tradicionales, como los basados en reglas heurísticas y clasificadores estadísticos, han sido superados por métodos más sofisticados, como las redes neuronales convolucionales (CNN) [26][27][29] y las redes neuronales recurrentes (RNN) [28][31]. Estas técnicas aprovechan la capacidad de las redes neuronales para aprender automáticamente patrones y características relevantes de las

[1] The best free, adless Chess server. (s. f.-c). lichess.org. https://lichess.org/

imágenes de los caracteres manuscritos.

Un caso de éxito ampliamente reconocido y usado en el ámbito académico y que en su momento revolucionó la concepción de este campo es el analizado en el artículo de LeCun et al. (1998) [11], donde los autores presentan un enfoque basado en CNN para el reconocimiento de dígitos manuscritos en el conjunto de datos MNIST, logrando una precisión sobresaliente.

El caso de reconocimiento de jugadas de ajedrez es ciertamente más complejo que el de MNIST. La gran mayoría de casos de éxito en el reconocimiento de caracteres no incluyen conocimiento en su proceso; sin embargo, existen ciertos casos de estudio orientados principalmente al reconocimiento de texto antiguo que requiere de cierto conocimiento del lenguaje para su correcta predicción.

Este es el caso del reconocimiento de caracteres manuscritos en contextos históricos, donde se han desarrollado enfoques que aprovechan el conocimiento histórico y las características estructurales de los textos. Un caso de éxito sería el presentado en el artículo Toledo et al. (2019) [12], donde se integra una Red Neuronal Convolucional para modelar información visual de las imágenes de palabras junto con una red de memoria a largo plazo bidireccional para modelar la relación entre las palabras.

Todos estos casos de éxito tienen un denominador común que es la utilización de redes neuronales para predecir el texto.

Las redes neuronales pre entrenadas han demostrado ser una herramienta invaluable en el reconocimiento de caracteres escritos a mano. Estos modelos han sido entrenados en conjuntos de datos extensos y representativos, lo que les permite capturar patrones complejos en la escritura a mano y realizar inferencias precisas en tiempo real [13]. Gracias a la capacidad de estas redes pre entrenadas para reconocer y transcribir caracteres escritos a mano con alta precisión, se ha logrado acelerar significativamente el proceso de transcripción manual y facilitar la búsqueda y el análisis de información en documentos escritos a mano de gran volumen.

Algunas desventajas de las redes neuronales pre entrenadas en el reconocimiento de caracteres manuscritos son que, aunque estos modelos han sido entrenados en conjuntos de datos amplios, es posible que no capturen la diversidad completa de estilos de escritura, especialmente en contextos específicos o regionales. Esto puede llevar a errores de reconocimiento en casos atípicos o en escritura poco común. Además, la dependencia de modelos pre entrenados puede limitar la adaptabilidad de las redes a diferentes dominios o idiomas, lo que requiere una personalización adicional y el ajuste de parámetros específicos para cada caso de uso.

Es por ello por lo que estos modelos deben usarse sobre tareas específicas a través de herramientas que permiten su adaptación a la tarea y la obtención de la transcripción de texto basándose en estas redes adaptadas a la situación qué se está tratando. Existen múltiples herramientas de este tipo; entre ellas, PyLaia [18], es flexible, de código abierto, independiente del dispositivo en el que se ejecuta y se puede utilizar para expresar una amplia variedad de experimentos, incluido el entrenamiento y la inferencia sobre modelos de redes neuronales profundas convolucionales y recurrentes.

Uno de los casos de uso más destacados de PyLaia es su aplicación en la digitalización de documentos antiguos o escritos a mano, como archivos históricos o manuscritos de investigaciones científicas [14].

2.3 Técnicas clásicas para el análisis de imágenes

En el campo del análisis de imágenes, las técnicas clásicas desempeñan un papel fundamental al proporcionar un marco sólido y bien establecido para abordar una amplia variedad de problemas. Estas técnicas han sido utilizadas y estudiadas durante décadas, lo que ha dado lugar a un conjunto de algoritmos y métodos confiables que han demostrado su eficacia en numerosas aplicaciones [20][21].

Una de las ventajas de las técnicas clásicas es su capacidad para extraer características relevantes de las imágenes de manera eficiente y robusta. Por ejemplo, el algoritmo de detección de bordes de Canny se ha utilizado ampliamente para identificar los contornos y bordes de los objetos en una imagen. Un ejemplo de este algoritmo mejorado fue presentado por Rong et al. (2014) [15], donde propuso una mejora para luchar contra posible ruido en las imágenes. Esta técnica ha sido aplicada con éxito en diversos campos, como la detección de objetos en sistemas de vigilancia [21] y el análisis de imágenes médicas para identificar estructuras anatómicas [20].

Las técnicas clásicas son una potencial herramienta para detectar estructuras y esquemas en una cierta imagen. En el caso de las planillas, las jugadas se pueden ver como texto encapsulado en una tabla. Las tablas han sido el origen de muchos debates acerca de cómo afrontar su lectura y reconocimiento. En Zanibbi et al. (2004) [17], el autor ofrece un resumen completo sobre el reconocimiento de tablas. El estudio aborda diferentes aspectos relacionados con la detección y el análisis de tablas en documentos, cubriendo modelos, observaciones, transformaciones e inferencias. Los autores exploran una amplia gama de técnicas utilizadas en el reconocimiento de tablas, incluyendo métodos basados en reglas, métodos estructurales y métodos de aprendizaje automático. El artículo también examina las dificultades y desafíos comunes asociados con la detección y extracción precisa de información de tablas en documentos.

En el caso que nos atañe, la tabla que compone la planilla se verá analizada bajo los niveles de iluminación con el objetivo de detectar las zonas más oscuras que corresponden con las líneas de separación entre jugadas.

CAPÍTULO 3
Análisis del problema

En este capítulo se presentará de manera más detallada la problemática y la descripción del trabajo. A continuación, se presentarán las imágenes seleccionadas para su estudio, análisis y etiquetado. Por último, la metodología seguida para afrontar los objetivos a conseguir y junto a ella el desarrollo del programa informático para afrontar la problemática anteriormente expuesta.

3.1 Descripción del trabajo

En el capítulo 1, en la sección de los objetivos, se hizo una breve introducción a los principales puntos a conseguir de este trabajo. En esta sección se ampliará la información ya expuesta entrando en un mayor grado de detalle acerca de cada uno de los objetivos.

El primer objetivo es el reconocimiento de la estructura de la planilla mediante el uso de técnicas clásicas de tratamiento de imágenes. Este objetivo hace referencia al hecho de lograr discernir el lugar donde se encuentran las jugadas anotadas en la planilla. El resultado del proceso visualmente debería ser el representado en la Figura 2.

Tiempo		Blancas	Negras	Tiempo
	1			
	2			
	3			
	4			
	5			

Figura 2: Localización (remarcado en rojo) del espacio de jugadas en una plantilla

Los espacios en blancos son los aprovechados para escribir las jugadas en el sistema internacional anteriormente explicado. Una planilla común de ajedrez cuenta con dos columnas de 40 jugadas por columnas, haciendo un total de 80 jugadas y de 160 espacios donde poder escribir jugadas.

La identificación de estos espacios se realizará mediante el uso de técnicas clásicas basadas en el análisis de la luminosidad. Esto se debe a que las zonas donde se encuentran las líneas horizontales y verticales son más oscuras, es decir, de menor luminosidad, en comparación con el resto de las zonas de la planilla.

Este proceso podría haberse llevado a cabo mediante técnicas de aprendizaje profundo, pero debido a la escasez de recursos para poder etiquetar y transcribir planillas manualmente resultaba poco factible. Asimismo, al tratarse de un campo tan concreto las posibilidades de aplicar modelos pre-entrenados para el reconocimiento de la tabla quedaba fuera del alcance del proyecto, optándose finalmente por una solución clásica.

Tras esta detección se podrían determinar los puntos de cruce de estas líneas, lo que permitiría extraer las casillas donde se encuentra el texto. De hecho, la medida de calidad de este proceso se va a basar en la correcta detección de los cruces en la planilla, al ser el factor clave para la extracción posterior de la información textual.

En segundo lugar, y una vez estos espacios hayan sido localizados, se procederá a aplicar un modelo pre-entrenado de HWCR. La novedad de este trabajo vendrá de las limitaciones del contexto, pues en este caso se descartará toda predicción que no sea una jugada legal de ajedrez. Para ello se irá simulando la partida de manera ficticia y de cada posición se extraerán todas las jugadas posibles legales. El modelo no se limitará a estas jugadas legales, pues podría haber habido un error en la predicción anteriormente cometido y que pudiese verse rectificado posteriormente, si no que de entre todas las posibilidades que el modelo proponga sólo serán vistas como opción aquellas que sean legales.

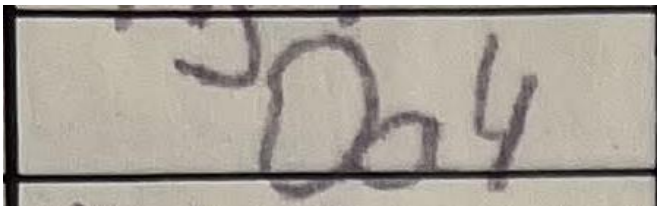


Figura 3: Ejemplo de anotación de jugada

En la imagen de la Figura 3 se puede observar una jugada, en este caso Da4, que significa que la dama se ha movido a la columna "a" fila 4. Una vez el modelo sea aplicado, predicciones como Do4 serán descartadas debido a que la columna "o" en ajedrez no es una columna existente. Sin embargo, que sea descartada no significa que el modelo no la dé como posibilidad.

3.2 Imágenes y datos

Para la realización de este proyecto se decidió trabajar con 11 planillas etiquetadas y transcritas manualmente más 59 únicamente transcritas. La razón detrás de esta decisión está basada principalmente en que cada planilla debía someterse a un etiquetado manual de los píxeles donde se encontraban todos los cruces, además de, por supuesto, transcribir las partidas enteras. Cada planilla contaba de media, dependiendo de la foto, con unos 300 cruces; esto, unido al hecho de repasar cada etiquetado en busca de posibles errores, implica una cantidad de tiempo ingente.

La selección de estos cruces y posterior guardado de la información fue realizado mediante el software Adobe Illustrator, unido a un pequeño *script* de JavaScript para guardar la información en un archivo de texto con los cruces marcados.

Toda imagen es de 4032x3024 píxeles, con una media de duración de las partidas de entre 20 hasta 50 movimientos. Una partida media de ajedrez cuenta con alrededor de 30 movimientos.

En la Figura 1 se muestra la estructura de una planilla de ajedrez; esta imagen es una representación idónea, pues la hoja de papel con la que se queda el jugador o jugadora es un calco de color amarillo, quedándose la otra custodiada por la organización. En la Figura 4 se observa un caso real de una planilla conservada al finalizar la partida.

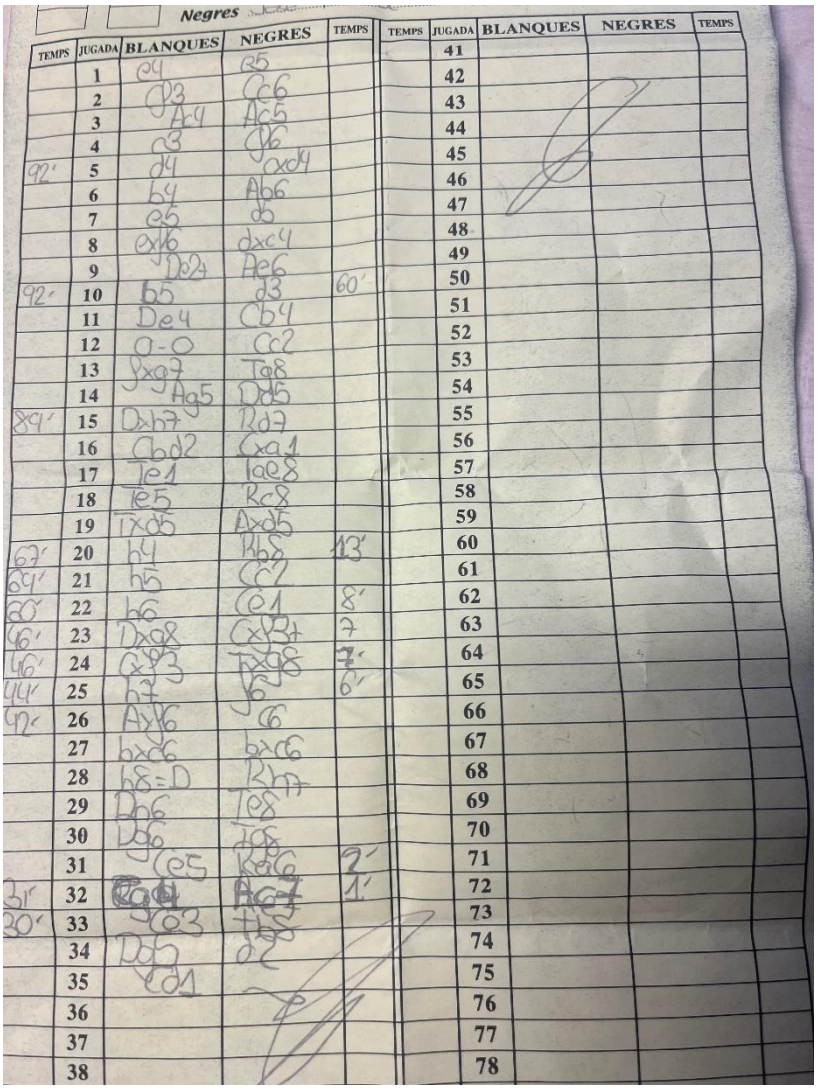


Figura 4: Ejemplo de calco de planilla

Todas las imágenes que conforman la base de datos de este proyecto han sido realizadas de tal manera que solo se vea la tabla con sus respectivas jugadas, omitiendo información adicional, como puede ser el nombre del adversario. Al fin y al cabo, esta información es intrascendente para la transcripción de la partida.

Por otra parte, esta imagen en concreto forma parte de la base de datos de imágenes con la que se ha trabajado. En ella se ilustra a la perfección las dificultades derivadas del problema, como puede ser la aparición de firmas (ambos jugadores deben firmar la planilla para dar su aprobación sobre la partida, siendo esta una mera formalidad), zonas con diferentes luminosidad (como la zona inferior derecha y la superior izquierda), jugadas no encuadradas en su casilla (como la jugada 33 de las negras), dobleces e irregularidades en el papel (como en la línea entre la 33 y 34), además de líneas no completamente rectas, que inician en un cierto punto y acaban a una altura significativamente distinta.

3.3 Metodología y desarrollo

En esta sección se realizará un estudio detallado de todos los procesos realizados, paso a paso, haciendo hincapié en los problemas que hayan surgido, cómo se han planteado, y la solución a ellos. Los resultados de estos procesos serán mostrados en el siguiente capítulo.

Cada proceso, problema que haya podido surgir en él, y su solución contarán con subapartados dentro de esta sección para facilitar y proporcionar una estructura sencilla al documento.

3.3.1 Detección de cruces

El primer paso, como se ha explicado anteriormente, es conseguir que el programa desarrollado sea capaz de ubicar los lugares dentro de la planilla donde se encuentran las jugadas escritas. Este proceso se puede ver gráficamente en la Figura 2.

Para conseguir este objetivo se decidió realizar una detección de cruces en la planilla. Una cruz no es más que la intersección entre dos líneas, una vertical y otra horizontal, de la planilla. Dentro de este proceso, la primera dificultad radicaba en detectar correctamente todas las zonas similares a las marcadas en la Figura 5.

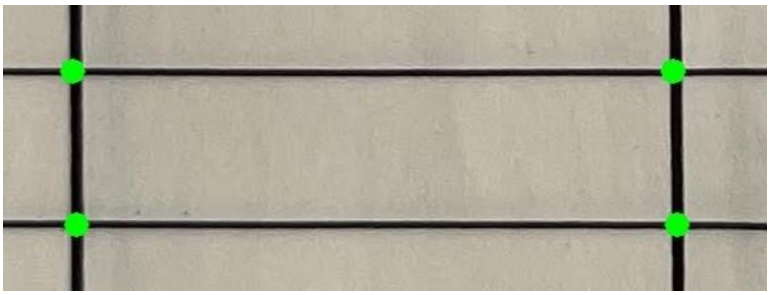


Figura 5: Ejemplo de cruces (marcados en verde)

Las cruces son los puntos marcados con un círculo de color verde, y el "rectángulo" que forman sería donde se encontraría la jugada a predecir.

Para poder llegar a localizar estos cruces se hizo un análisis de la luminosidad tanto verticalmente como horizontalmente. Este estudio perseguía el objetivo de poder detectar las líneas, que son las zonas más oscuras de la imagen, para posteriormente unirlas y formar una planilla creada artificialmente.

La imagen se recorre vertical y horizontalmente de diez en diez píxeles con un *step* de cinco píxeles. El *step* es sinónimo de solapamiento, es decir, en una primera iteración se recorren las diez primeras columnas o filas, y en la segunda iteración en vez de pasar directamente a la undécima se recorrería de la sexta a la decimoquinta. De esta manera, si la línea está entre los píxeles 24 y 34, se podría conseguir detectar una mayor parte de ella, pues sin el *step* los pasos que se darían no serían lo suficientemente flexibles. Durante este recorrido se realiza la media de intensidad de luminosidad de los píxeles columna a columna o fila a fila, obteniendo 10 números por iteración, a los cuáles se les aplica de nuevo una media. En total se acaba obteniendo un *array* de longitud 3024/5 para las

columnas y 4032/5 para las filas. La división no da un resultado exacto, esto se debe a que en la última iteración se cogen los píxeles sobrantes, cuatro en el primer caso y cinco en el segundo.

Una vez se tienen los dos vectores unidimensionales, uno por eje, se realiza la búsqueda de dónde se encuentran las líneas. Esta búsqueda se basa en encontrar los mínimos locales de los *arrays*, pues cuanto más oscuro menor media de intensidad de luminosidad. El mínimo local se busca con un cierto margen; este margen principalmente se debe a que las líneas de las planillas se encuentran bastante separadas y así se evita etiquetar como mínimo local a los conocidos como puntos de silla.

Conocidos los índices de los mínimos locales se procede a crear una imagen en blanco con sólo líneas negras correspondientes a los mínimos locales por eje, es decir, una imagen con la detección de líneas horizontales y otra con la detección de las verticales. La combinación de estas imágenes daría como resultado la estructura de la planilla.

Los resultados quedaron alejados de los esperado, enfrentándonos al **primer problema** en el proceso, las **líneas verticales**.

Con la imagen creada, se mostraban unos grandes resultados detectando las líneas horizontales, todo lo contario a las verticales. Seguramente, debido a una menor presencia de líneas verticales en la planilla, en el proceso de búsqueda de los mínimos locales el programa detectaba bajones de luminosidad, que en ningún caso era por las líneas si no por cuestiones de cómo estaba realizada la imagen. Este problema se puede ver claramente en la Figura 6.

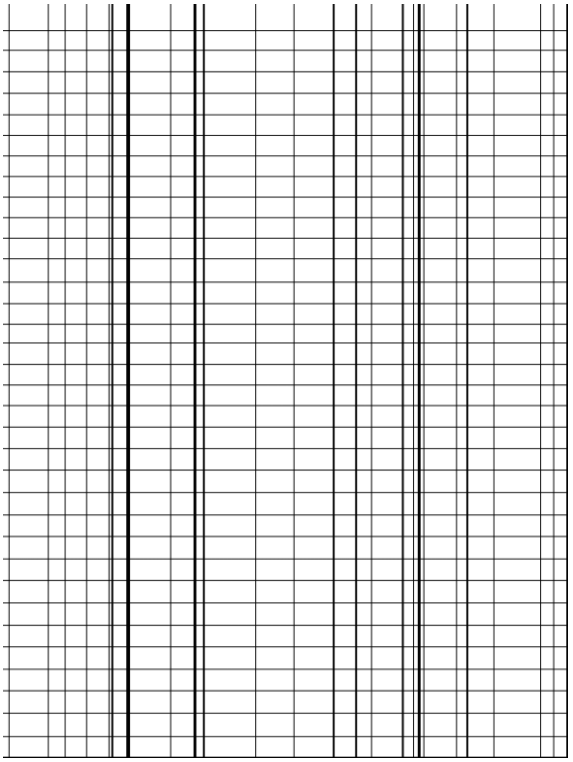


Figura 6: Primer resultado basado en el analisis de iluminosidad

Este problema fue enfocado desde dos puntos de vista. El primer de ellos, fue tratar de mejorar o incorporar conocimiento al proceso de la búsqueda de mínimos locales. Esto quedó descartado por no encontrar una manera idónea de incorporar conocimiento, como podría ser definir una serie de líneas verticales fijas, debido a que todo conocimiento iría sujeto a cómo estaba tomada la imagen. Al fin y al cabo, obligar al usuario a tomar la foto de una cierta manera hacía que el proyecto pasara de una idea general y abierta, a un proyecto limitado y con restricciones.

Por ello se planteó un segundo enfoque. Esta **solución** se basaba en el uso de **diferencias relativas** para la detección de cruces.

Antes de desarrollar la solución se requiere realizar un paso previo. Las imágenes creadas en base a líneas negras sacadas de los índices de los vectores con mínimos locales tienen la característica de ser binarias. Es decir, todo píxel de la imagen es o 0, que corresponde al negro, o 255, correspondiente al blanco. Si al unir las imágenes en vez de hacer una unión de las líneas se realiza una operación para quedarse con el valor máximo se obtendrá los puntos donde se cruzan las líneas, pues sólo donde coincidan dos píxeles a 0 se mantendrá el 0, el resto será a 255. Es decir, como resultado final de este proceso se obtendrán imágenes prácticamente blancas con pequeños puntos negros referentes a los supuestos cruces.

Sin embargo, debido a la deficiente detección de las líneas verticales hay más puntos de los debidos, y por eso es necesario acudir a las diferencias relativas.

El proceso actúa como sigue. Cada punto en negro será analizado y puesto sobre la imagen original, creando una ventana de 30x30 píxeles sobre él. Una vez creada esta ventana se realizarán una serie de operaciones para comprobar si la ventana de la imagen original se trata de una cruz.

- Siendo v_05 y h_05 la intensidad media de las 5 primeras filas/columnas de la ventana de 30x30
- Siendo v_13_18 y h_13_18 la intensidad media de la 13º a la 18º fila/columna de la ventana de 30x30
- Siendo v_25_30 y h_25_30 la intensidad media de la 25º a la 30º fila/columna de la ventana de 30x30

La ventana debe de cumplir las siguientes condiciones. Estas han sido creadas con cierta laxitud pues el error de no detectar un cruce es mayor que el de predecirlo sin que lo sea.

1.- Las primeras y últimas filas/columnas deben ser luminosamente hablando parecidas. El denominador será siempre el menor de ambas variables; en este caso, se ha escogido arbitrariamente las variables referentes a las primeras filas/columnas.

$$\frac{|v\_05 - v\_25\_28|}{v\_05} < 0.25 \; ; \; \frac{|h\_05 - h\_25\_28|}{h\_05} < 0.25$$

2.- Las diferencias entre los extremos y la filas/columnas centrales deben de ser mayores a 0.5. En este caso el denominador será la columna central pues es la supuestamente más pequeña en valor de luminosidad.

$$\frac{|v\_05 - v\_13\_18|}{v\_13\_18} > 0.5 \; ; \; \frac{|h\_05 - h\_13\_18|}{h\_13\_18} > 0.5$$

$$\frac{|v\_25\_30 - v\_13\_18|}{v\_13\_18} > 0.5 \; ; \; \frac{|h\_25\_30 - h\_13\_18|}{h\_13\_18} > 0.5$$

Si estas operaciones dan el resultado esperado se confirmaría que se trata de una cruz. Una vez realizado este proceso se descartan los puntos que no han cumplido las condiciones anteriormente indicadas y permanecen los que sí. Como último proceso se eliminan los puntos repetidos, considerando punto repetido todo aquel punto que tenga a menos de 25 píxeles otro punto. Este proceso hace referencia a la necesidad de eliminar ruido y redundancia en lo referente a la cantidad de puntos obtenidos. Unos primeros resultados a modo de *beta* desvelaron la gran suma de puntos que estaban siendo evaluados. Debido a la proximidad entre píxeles, tales como píxeles con las coordenadas (1357,658) y (1359,659), se detectaban cruces doblemente, generando información repetitiva y no necesaria.

Una vez se tienen los cruces deseados estos deben ser comparados con los etiquetados manualmente de las 11 planillas anteriormente comentadas. Gráficamente, los resultados pueden examinarse en la Figura 8, representados por una matriz de confusión.

Una matriz de confusión (Figura 7), también conocida como matriz de error, es una tabla que se utiliza frecuentemente en estadística y aprendizaje automático para visualizar el rendimiento de un algoritmo de clasificación. Esta tabla se organiza de tal forma que cada columna de la matriz representa las instancias de la clase verdadera, mientras que cada fila representa las instancias de la clase predicha. La matriz de confusión se divide en cuatro cuadrantes diferentes que representan los cuatro tipos de predicciones posibles que se pueden hacer:

- Verdaderos positivos (TP): Los casos en los que el algoritmo predijo la clase positiva correctamente.
- Verdaderos negativos (TN): Los casos en los que el modelo predijo la clase negativa correctamente.
- Falsos positivos (FP): También conocidos como errores de Tipo I, son los casos en los que el modelo predijo la clase positiva incorrectamente.
- Falsos negativos (FN): También conocidos como errores de Tipo II, son los casos en los que el modelo predijo la clase negativa incorrectamente.

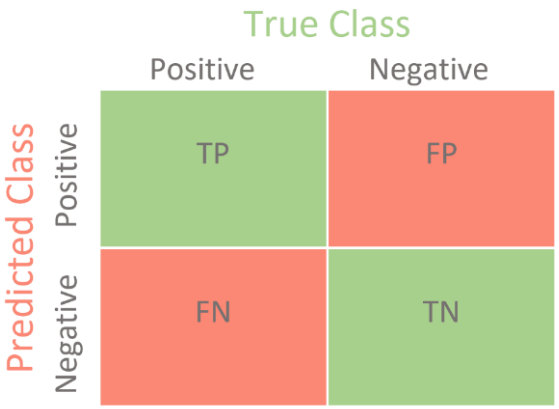


Figura 7: Matriz de confusión

En el caso de la Figura 8 cabe reseñar que el 0 de los *True Negatives* es debido a que los cruces primeramente detectados y posteriormente rechazados por las diferencias relativas no fueron añadidos en el *dataframe* utilizado para obtener esta matriz, pues no aportaban información de valor a los resultados. El *dataframe* final consiste en la unión de los puntos predichos como verdaderos por el algoritmo, más aquellos puntos etiquetados manualmente no incluidos en la predicción. Estos últimos puntos sólo fueron añadidos en caso de que no existiera una predicción cercana a ellos, es decir, se dejó un cierto margen de error de 10 píxeles a las predicciones. Por otra parte, debe destacarse la gran cantidad de *False Negative*. Estos 2915 puntos son puntos de cruce que el algoritmo debería haber detectado y no lo consiguió. Por último, la cantidad de cruces dependerá de la cantidad de puntos estimados por el algoritmo, por ello la suma total de puntos puede variar ligeramente en futuras muestras de resultados. Esta variación vendrá en la parte de los *False Positive*, pues la cantidad de puntos etiquetados (primera columna de la Figura 8) es invariable.

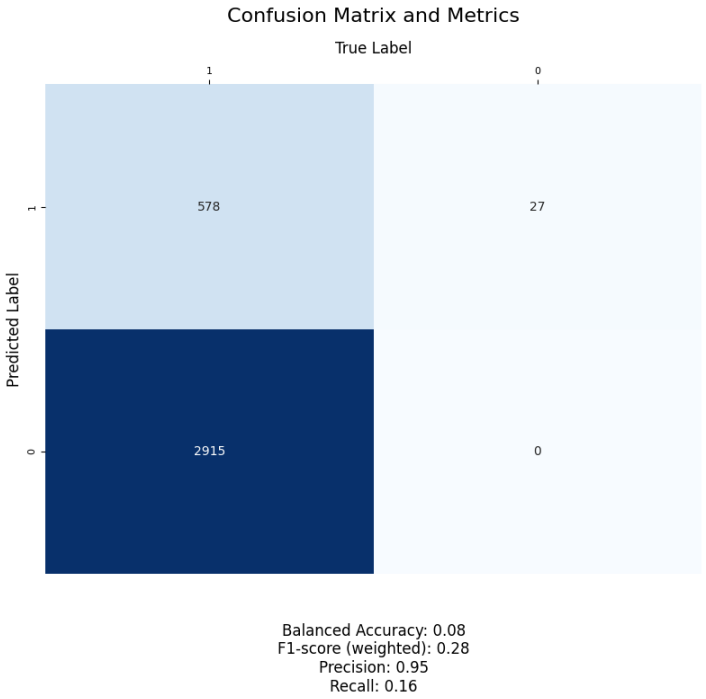


Figura 8: Matriz de confusión de la primera aproximación del problema.

Asimismo, la Figura 8 viene con cuatro métricas en la parte inferior. La primera se corresponde a la *Balanced-Accuracy*, métrica utilizada para los casos con clases desbalanceadas, es decir, donde existen más casos en una de las opciones, como es la situación del problema. Su fórmula corresponde a las siguientes ecuaciones.

$$Balanced - Accuracy = \frac{Sensitivity + Specificity}{2}$$

$$Sensitivity = \frac{TP}{TP + FN} \; ; \; Specificity = \frac{TN}{TN + FP}$$

En segundo lugar, el *F1-Score (weighted)*, de aplicación similar a la métrica anterior y que aporta valor a modo de segunda opinión. La diferencia con el *F1-Score* tradicional es que calcula la métrica para cada *label* y encuentra su media ponderada. Su fórmula está basada en la *precision* y *recall* definidas posteriormente.

$$F1 - Score(Weighted) = \frac{Precision \cdot recall}{Precision + recall}$$

La *precision* es una métrica que cuantifica la proporción de predicciones positivas correctas realizadas por el modelo de todas las predicciones positivas. En otras palabras, de todas las veces que el modelo predice la clase positiva, cuántas de esas veces la predicción es correcta. La precisión es una medida útil cuando el coste de los falsos positivos es alto. Por otro lado, el *recall*, también conocido como sensibilidad, es una métrica que cuantifica la proporción de verdaderos positivos que fueron identificados correctamente por el modelo. En otras palabras, de todas las instancias positivas que existen, cuántas de ellas el modelo predijo correctamente. El *recall* es una medida útil cuando el coste de los falsos negativos es alto.

$$Precision = \frac{TP}{TP + FP} \; ; \; Recall = \frac{TP}{TP + FN}$$

Teniendo en cuenta los valores de estas métricas y la gran cantidad de Falsos Negativos, quedó en evidencia la necesidad de mejorar el proceso. Esta mejora estaba fundamentada en la necesidad de conseguir que el algoritmo detectase la no-linealidad de las rectas verticales y horizontales que conforman la planilla.

Este progreso vendría a través de un pequeño cambio en el análisis de luminosidad de la planilla. Este análisis se realizaba en toda la planilla, suponiendo un problema con las líneas que empiezan y acaban en diferentes alturas. Por ello, se decidió realizar el análisis dividiendo la imagen en sub-imágenes, en concreto en 72 imágenes de 336x378. Además, se redujo el *step* a cuatro píxeles. El resto de procesos del algoritmo, como la evaluación de los cruces mediante diferencias relativas y la posterior extracción de resultados se mantuvo intacta.

Los resultados de la Figura 9 muestran un evidente avance y progreso respecto a los mostrado en la Figura 8. Sin embargo, y teniendo en cuenta que un falso negativo podría suponer el no reconocimiento del lugar de una jugada, los resultados continuaban sin ser esperanzadores.

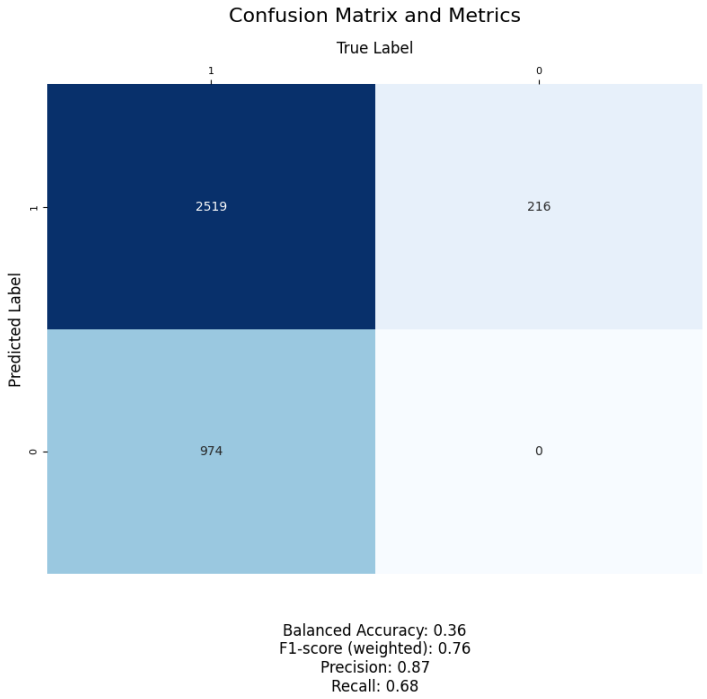


Figura 9: Matriz de confusión mejora sub-imágenes

3.3.2 Cambios en el algoritmo

Tras obtener los resultados anteriormente comentados quedó clara la necesidad de realizar cambios drásticos en el algoritmo de detección de cruces. Previamente a comentar estos cambios, durante la realización de los mismos se detectó un punto de mejora evidente posterior a la obtención de resultados, a modo de post-proceso. Este post-proceso está basado en dos pilares fundamentales:

1. La adición de puntos basado en la distancia regular existente entre los puntos verticales.
2. Cambios en el proceso del cálculo de las diferencias relativas.

El primero se basa en el aprovechamiento de la estructura de la planilla y su regularidad entre las distancias de los cruces verticales. Una simple tarea de mejora de detección de cruces incluiría añadir puntos en base a la distancia entre un punto y el siguiente. Por ejemplo, si la distancia entre puntos es 10 (calculada en base a la mediana de distancia entre puntos de una vertical), y hay dos puntos con un valor en el eje "y" de diez y treinta respectivamente, es evidente que es necesario añadir un punto en el veinte.

Este proceso es posible realizarlo de arriba abajo y viceversa, pero no en horizontal, pues las distancias entre puntos son irregulares. La clave de este pequeño post-proceso reside en un correcto cálculo de la mediana de las distancias en una misma vertical, siendo necesario recorrer las verticales de 24 en 24 píxeles y con un solapamiento, para evitar saltarse una horizontal y evitar falsas horizontales, es decir, horizontales que contengan falsos negativos.

El segundo, dado los resultados anteriores, nace de la necesidad de realizar cambios en todo proceso que lo permita. En este caso para el cálculo de las diferencias relativas, en la imagen de 30x30 píxeles creada en base a cada punto evaluable se realizará un proceso de umbralización. Umbralizar una imagen, en este caso de manera binaria, es un proceso en el que se establece un valor límite para dividir los píxeles en dos categorías: aquellos cuyo valor está por encima del umbral y aquellos cuyo valor está por debajo. Esto ayuda a resaltar características de la imagen y simplificar su análisis o segmentación. El proceso umbralizará la imagen en base a la media de intensidad, resaltando de una manera más exagerada la existencia de una posible cruz.

Ambos post-procesos son aplicables no sólo en el nuevo algoritmo sino también en los anteriormente comentados, pero estos últimos parten de una base de malos resultados que no mejorarían lo necesario para ser considerados algoritmos óptimos.

El nuevo algoritmo desarrollado debía ser robusto ante los cambios de iluminación, debido a que al tratarse de fotos tomadas por un usuario no se puede garantizar una calidad y condiciones ideales. Además, el algoritmo debe ser lo suficientemente flexible para crear trazos de líneas no lineales, es decir, el software necesitaba poder modelar correctamente aquellas líneas que debido a irregularidades en la planilla o imagen no eran rectas.

A continuación, se va a realizar la explicación técnica de los procesos que conforman el nuevo algoritmo.

El primer proceso, y dado los exitosos, pero no suficientes resultados en el anterior algoritmo, se dividió la imagen verticalmente en diez partes iguales, eliminando el resto de los píxeles restantes para trabajar con un cierto margen. Esta división vertical busca encontrar las líneas horizontales de la planilla. El trabajo para encontrar las líneas verticales se hará seguidamente a este.

Una vez se tiene las divisiones se calcula la intensidad de luminosidad media para cada fila (línea horizontal). Al trabajar con imágenes de 4032x3024 se obtendrán 4032 valores guardados en un vector por cada sub-imagen.

Estos valores se utilizan para decidir qué líneas expandir, que serán aquellas horizontales con una intensidad media menor, es decir, más oscura que la media de intensidades de las 10 líneas más cercanas que la rodean. Estas líneas serán las candidatas a ser una posible línea horizontal de la planilla.

Conocidas las posibles líneas se fija en cada una de ellas un píxel semilla. Este píxel semilla corresponderá al píxel de menor valor de todo el vector. En la Figura 10 estos píxeles quedan marcados en negro, y las flechas indican la línea horizontal marcada como posible candidata.

El píxel semilla será la base de la que partirá la expansión del resto de píxeles. Esta expansión empezará con el algoritmo moviéndose hacia la derecha del píxel semilla, examinando su píxel seguidamente posterior y los 3 píxeles superiores e inferiores a este.

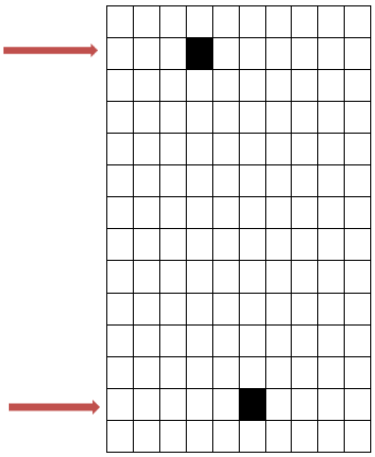


Figura 10: Ejemplo de píxel semilla

Sobre estos píxeles se creará una pequeña ventana de 10x10 píxeles, se calculará su intensidad de luminosidad media, y esta pasará a ser nuestro umbral. Si alguno de los píxeles posteriores a la semilla es más oscuro que el umbral entonces lo marcamos como la continuación de la línea que estamos trazando. En caso contrario, se vacía la lista donde se estaban acumulando estos píxeles y se pasa a la siguiente línea marcada (flecha roja en la Figura 10).

Volviendo al caso favorable donde el píxel ha sido menor que el umbral calculado a través de la ventana creada artificialmente, se procede a realizar la misma secuencia de tareas, pero tomando el siguiente píxel. Finalmente, se detectará como línea horizontal si en el proceso se ha podido trazar una línea hasta los límites de la siguiente línea vertical marcada, pero con un cierto margen. Al finalizar en la expansión a derecha del píxel semilla, este proceso se hace también hacia la izquierda, con el fin de trazar horizontales conectadas en el caso de que hacia uno de los lados falle la expansión erróneamente por circunstancias de la imagen.

El píxel semilla sería el primer cuadrado de la Figura 11, las flechas representarían la expansión de los píxeles. El segundo píxel rojo sería el menor al umbral calculado, y la última expansión correspondería al caso desfavorable donde ninguno de los píxeles estudiados es menor al umbral.

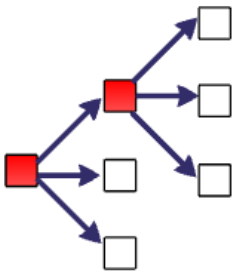


Figura 11: Representación del proceso de expansión de píxeles

Finalizada la detección de líneas horizontales se procede a realizar la eliminación de aquellas líneas que no se encuentren a una mínima distancia más un cierto margen de la siguiente. Esta distancia es calculada como la mediana entre las distancias de todas las líneas.

Realizado este post-proceso se realiza el mismo procedimiento con las líneas verticales, tomando como píxeles semilla todos los de las horizontales detectadas en el proceso previo. Este procedimiento el cuello de botella en cuanto a coste temporal del algoritmo se refiere. La razón es sencilla y es la altura de las líneas, pues al ser tan largas, el proceso de expansión de los píxeles se vuelve lento. Cabe reseñar que las distancias entre líneas verticales son irregulares y por ello no es posible aplicar el mismo post-proceso que con las horizontales. El resultado de este algoritmo daría lugar a una imagen como la de la Figura 12.

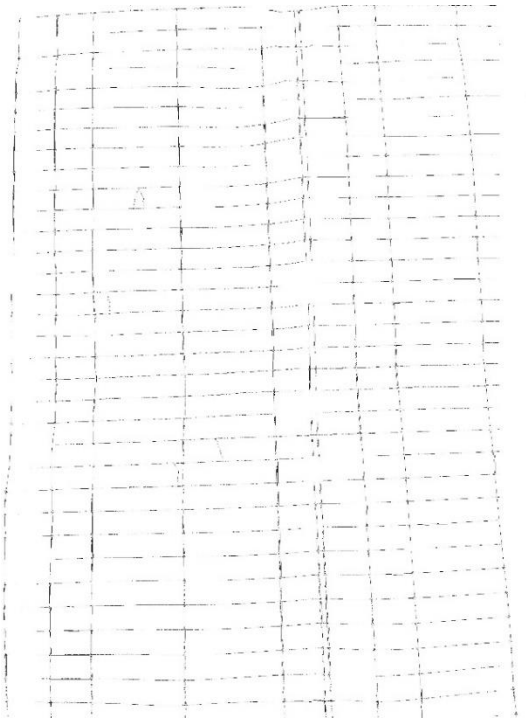


Figura 12: Imagen artificial de la estructura de la planilla con el nuevo algoritmo.

La Figura 12 es una representación no perfecta, pero sí muy aproximada, esperanzadora y con una evidente mejora respecto a las anteriores versiones del algoritmo, de la estructura de una planilla. Como se ve, es capaz de modelar la curvatura de las líneas y los desperfectos de la planilla.

De esta imagen el valor se encuentra en los puntos donde coinciden las verticales y horizontales, pues se trata de los puntos de cruce. Cada uno de estos puntos es la esquina del espacio donde se encuentran encapsuladas las jugadas manuscritas.

Por ello se decidió extraer las líneas verticales y horizontales de manera independiente para posteriormente juntarlas a través del método *cv2.max*. La función cv2.max() es parte de la biblioteca OpenCV [24], una popular biblioteca de visión por computadora en Python. Toma dos imágenes de entrada (o matrices) del mismo tamaño y tipo, y compara cada píxel en las dos imágenes. Para cada píxel, selecciona el valor máximo de los dos píxeles correspondientes en las dos imágenes y genera una nueva imagen (o matriz) con esos valores máximos.

Esa imagen generada artificialmente es una imagen con pequeños puntos negros referentes a los cruces, sobre la cual se realizará el proceso de detección de cruces a través de diferencias relativas. Este proceso no difiere en ningún aspecto del anterior algoritmo.

Tras ejecutar todas las tareas/procesos el programa devolvió los resultados mostrados en la Figura 13.

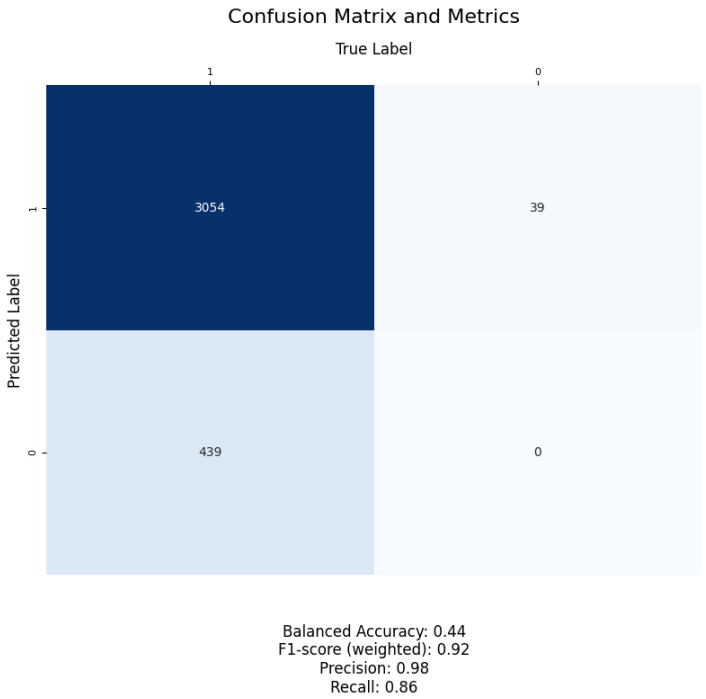


Figura 13: Resultados finales algoritmo

Tras conseguir superar la marca de 0.85 en *recall* y 0.90 en *precision* y *f1-score* se declaró el algoritmo apto para la posterior fase del proyecto.

Como añadido, en el *Apéndice A* se puede observar los resultados tomando en cuenta únicamente los cruces donde se encuentran los rectángulos donde pueden ir anotadas jugadas, sin tener en cuenta el resto de los puntos etiquetados.

En conclusión, el algoritmo es capaz de detectar con una gran fiabilidad la gran mayoría de cruces, además de ser escalable y asentar una base y unos conocimientos mediante el uso de técnicas clásicas para el reconocimiento de planillas de ajedrez.

CAPÍTULO 4
Conocimiento extraído y evaluación de modelos

En este capítulo se expondrán los diferentes modelos utilizados para la predicción del texto manuscrito y el uso y papel de PyLaia en este proceso. Cada modelo tendrá su propia sección dentro del capítulo con su explicación inicial, los datos con los que han sido entrenados, las características de la red, los resultados otorgados y sus respectivas limitaciones.

4.1 PyLaia

PyLaia [18] es una librería de Python de código abierto para la transcripción automática de documentos manuscritos e impresos. Es utilizada principalmente en el área de Procesamiento de Imágenes y Visión por Computadora, especialmente en tareas de transcripción automática de texto en imágenes de documentos. Esto incluye el reconocimiento de texto en documentos históricos, manuscritos y otros tipos de documentos en los que la transcripción manual puede ser difícil, lenta o propensa a errores. Las características principales y lo que ofrece PyLaia es una plataforma para el entrenamiento y evaluación de redes neuronales con datos personalizados. También se integra a la perfección con otros *frameworks* como PyTorch [32] además de ser modular y extensible pues permite a los usuarios adaptar sus funcionalidades según sea necesario.

PyLaia como *toolkit* recibió el apoyo financiero del centro de investigación *Pattern Recognition* and *Human Language Technology* (PRHLT) formado por investigadores de la Universitat Politècnica de València (UPV) en las áreas de interacción multimodal, reconocimiento de patrones, procesamiento de imágenes (análisis de imágenes, visión por computador, reconocimiento de texto escrito a mano, análisis de documentos) y procesamiento del lenguaje.

Todo modelo de HWCR se divide en tres grandes fases:

1. **Preparación de las imágenes:** cuando se va a entrenar cualquier modelo los datos de entrada requieren de un cierto pre-proceso. En el caso de las imágenes suele consistir en ajustar los píxeles de altura, su formato (blanco y negro, color o binarizadas), su transcripción y su división, es decir, si es un párrafo de un texto, su división en líneas.
2. **Entrenamiento del modelo:** como cualquier modelo de *Machine Learning* se necesita de un entrenamiento y ajuste de hiperpárametros. Aquí y en la siguiente etapa es donde entra PyLaia en juego.
3. **Evaluación del modelo:** todo modelo debe ser evaluado mediante diferentes métricas según el tipo de modelo y las circunstancias del problema. Los modelos de HWCR tienen las suyas propias que posteriormente serán analizadas.

El código asociado al segundo y tercer punto se puede encontrar en el apéndice B.

Tras esta explicación de PyLaia, sus funcionalidades y los procedimientos de uso, se procede a explicar los diferentes modelos entrenados mediante esta plataforma.

4.2 Red Neuronal Rodrigo

Esta red neuronal profunda recibe su nombre debido a que está entrenada en base a más de 20000 imágenes correspondientes al manuscrito de 1545 llamado "Historia de España del arçobispo Don Rodrigo" escrito en castellano antiguo por un único autor. Su letra tiene influencias góticas, pero tiene un estilo principalmente, como lo llaman los expertos, humanista.

Las 20000 imágenes con las que se realizó el entrenamiento corresponden a segmentos de líneas de este libro, como la observable en la Figura 14. Estas imágenes además son binarias; por ello las futuras casillas donde se encuentren las jugadas serán binarizadas por la media para tener un formato similar a las seleccionadas para entrenar el modelo. La siguiente fórmula representa el cálculo de la media acompañada posteriormente por los condicionales de la binarización.

$$AVG(I) = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} I(i,j)$$

- Sea I una imagen es escala de grises de dimensiones MxN
- Sea I (i, j) la intensidad del píxel en la posición i,j.
- I (i, j) = 255 si AVG(I) <= I (i, j)
- I (i, j) = 0 si AVG(I) > I (i, j)

La red neuronal entrenada combina capas convolucionales con unidades recurrentes, lo cual la hace particularmente adecuada para el procesamiento de imágenes de texto, donde es necesario capturar tanto características locales como dependencias a largo plazo entre los elementos de la secuencia.

Figura 14: Ejemplo de línea del libro de Rodrigo

Las capas convolucionales son responsables de extraer características locales de la imagen y las del modelo tienen las siguientes características.

- ***Kernel Size*** (3x3). El tamaño del núcleo se refiere a las dimensiones del filtro que se desliza sobre la imagen de entrada para extraer características. Un tamaño de 3x3 es comúnmente utilizado, ya que es lo suficientemente grande para capturar patrones y lo suficientemente pequeño para ser computacionalmente aceptable.
- ***Num Features*** [16,16, 32, 32]. Esto determina la cantidad de filtros en cada capa convolucional. A medida que avanzamos en la red, el número de características aumenta, permitiendo que la red capture características más complejas.

- ***Batch Normalization****.* Activado. Este es un método utilizado para que el entrenamiento sea más rápido y estable mediante la normalización de las capas por recentrado y reescalado.
- ***Pool size*** [2,2,0,2]. Esto se refiere a la operación de agrupación (*pooling*) que sigue a la convolución, y que tiene el efecto de reducir las dimensiones espaciales de la salida.

Las capas recurrentes (RNN) son capaces de mantener un estado que puede capturar información sobre elementos previos en la secuencia.

- **LSTM.** Las redes LSTM (*Long Short-Term Memory*) [44] son un tipo especial de RNN que son más efectivas en el aprendizaje de dependencias a largo plazo.
- **Unidades RNN:** 512. Esto determina la dimensión del estado oculto en las capas LSTM.
- ***Número de Capas RNN*****:** 3. Tenemos tres capas de LSTM apiladas, lo que puede ayudar a capturar características más complejas.

Como última característica, típica de la gran mayoría de redes neuronales entrenadas para el HWCR, las imágenes de entrada a la red deben tener 64 píxeles de altura. Sin requisito para el ancho de la imagen.

Tras definir las características de la red neuronal se debe definir cómo se va a realizar el entrenamiento.

En primer lugar, se hizo uso de una GPU para agilizar el entrenamiento. Hoy en día las unidades de procesamiento gráfico son un componente principal del ordenador que ha cogido un papel protagonista en este tipo de entrenamientos.

En segundo lugar, el uso de la tasa de aprendizaje, o también conocida como *learning rate*; es un hiperparámetro que controla cuánto se ajustan los pesos y sesgos de la red en respuesta a los errores estimados. En otras palabras, determina el tamaño de los pasos que se toman para minimizar la función de pérdida (o de error) durante el proceso de optimización, como, por ejemplo, en el algoritmo de descenso de gradiente.

Cuando se actualizan los pesos de la red, típicamente se hace en la dirección opuesta al gradiente de la función de pérdida con respecto a esos pesos. La tasa de aprendizaje determina cuán grandes o pequeños son estos ajustes.

Si la tasa de aprendizaje es muy alta, los pesos pueden cambiar drásticamente, lo que podría hacer que el algoritmo oscile o incluso diverja y no converja a una solución. Por otro lado, si la tasa de aprendizaje es muy baja, el algoritmo podría necesitar demasiadas iteraciones para converger, lo que significa que el entrenamiento sería muy lento o podría atascarse en un mínimo local y no encontrar una buena solución.

En este caso esta ha sido definida a 0.003, que tiende a ser bastante pequeña, pero se ajusta a las necesidades del modelo.

Por último, lo que se conocen como *epochs,* se establece a 60. Durante un *epoch*, la red neuronal aprende ajustando sus pesos con el objetivo de reducir el error en las predicciones. Cuando se establece que el número de *epochs* es igual a 60, significa que el algoritmo de entrenamiento hará 60 pasadas completas por todo el conjunto de datos. Esto implica:

1. **Más oportunidades de aprendizaje**: la red tiene más oportunidades de aprender y ajustar sus pesos en base a los datos.
2. **Mayor tiempo de entrenamiento**: con un mayor número de *epochs*, el tiempo de entrenamiento será más largo, ya que se requieren más iteraciones sobre el conjunto de datos.

Tras realizar definir todo lo anteriormente comentado se obtiene un modelo, con sus respectivos hiperparámetros, que debe ser evaluado. Para evaluar modelos de HWCR se utilizan dos métricas principalmente, *Character Error Rate* (CER) y *Word error Rate* WER.

El cálculo del CER se basa en el concepto de distancia de Levenshtein [45], en el que contamos el número mínimo de operaciones a nivel de carácter necesarias para transformar el texto verdadero (también conocido como texto de referencia) en el resultado del HWCR. Se representa con esta fórmula.

$$CER = \frac{S + D + I}{N}$$

- S = Número de sustituciones
- D = Número de supresiones
- I = Número de inserciones
- N = Número de caracteres del texto de referencia (también conocido como texto original)

El denominador N puede calcularse alternativamente con: N = S + D + C (donde C = número de caracteres correctos).

La fórmula de la WER es la misma que la de la CER, pero la WER opera a nivel de palabra. Representa el número de sustituciones, supresiones o inserciones de palabras necesarias para transformar una frase en otra.

La WER suele estar correlacionada con la CER (siempre que las tasas de error no sean excesivamente altas), aunque se espera que el valor absoluto de la WER sea superior al de la CER.

En el caso del modelo de Rodrigo estas métricas presentaban valores de CER=1.968 WER=7.687 para el test de dicha tarea. Estos valores indican una *performance* del modelo excelente.

Una vez se tiene el modelo entrenado y evaluado se procede a realizar la predicción en base a la extracción de casillas con sus respectivas jugadas.

Lo óptimo para esta tarea sería haber entrenado una red neuronal para ello, pero cabe recordar la escasez de datos con la que estamos trabajando. Las 11 planillas etiquetadas manualmente más las 59 sin etiquetar, a una media de 30 jugadas, no son suficientes para entrenar una red neuronal.

Adicionalmente, este tipo de redes neuronales requieren de ordenadores de grandes prestaciones para ser entrenados; por ello, contar con modelos ya entrenados como Rodrigo reducía la cantidad de horas de gasto computacional.

Los resultados no fueron los esperados, obteniendo resultados carentes de significado;

Figura 15: Ejemplo de casilla binarizada

En el ejemplo de la Figura 15 el modelo debería de predecir "Ab5". Lejos de la realidad la salida del modelo era "omm". Una salida sin ningún sentido ni próxima a la realidad.

Esta distancia entre lo que debería ser y lo que el modelo predice está fundamentada en los siguientes puntos que hacen que el modelo no obtenga resultados lógicos.

- **Estilo de escritura:** haciendo una comparativa entre la Figura 14 y 15 es evidente la diferencia entre el estilo de escritura. Los textos entrenados son del siglo 15 y las jugadas son escritas por diferentes personas en los últimos 3 años. Esto provoca confusiones entre letras.
- **Rectángulo y casilla:** las jugadas puestas a predecir se encuentran rodeadas por un rectángulo que tras la binarización es negro, mientras que en las imágenes de entrenamientos son pocos los píxeles a negros que no forman parte. Esto podría hacer que el modelo se confundiese.
- **Palabras y jugadas:** por último, el modelo espera unas imágenes con palabras con un significado y las jugadas no lo son.
- **Falta de símbolos**: las predicciones de este tipo de modelos pasan por un proceso que se llama decodificación. Es decir, a tu modelo ya entrenado se le pasan unas ciertas imágenes y éste lo que predice no son caracteres, si no números, que corresponde a unas ciertas letras. Este archivo de correspondencias, en el caso de Rodrigo, no contenía ni símbolos especiales que aparecen en las jugadas como el "+" que indica jaque o "=" que indica coronación, además de tampoco contener letras mayúsculas, reduciendo las posibilidades de que la salida del modelo sea una jugada legal.

Teniendo en cuenta los aspectos anteriormente comentados, quedó descartada la opción de utilizar Rodrigo como un modelo válido para predecir las jugadas en cuestión. Por ello se pasó a buscar nuevas vías más robustas ante los aspectos anteriormente comentados. Para ello se buscaron modelos que contuvieran en la transcripción de sus imágenes, números, minúsculas y mayúsculas y caracteres especiales.

4.3 Red Neuronal HisClima

Para superar las problemáticas presentadas con el anterior modelo se propuso utilizar un modelo entrenado con una mayor variedad de vocabulario, símbolos e imágenes. En concreto, el corpus empleado para evaluar las ideas propuestas en este trabajo es la primera versión de la base de datos HisClima [46]

Este corpus se ha compilado a partir del diario a bordo del Jeannette, un barco que navegó por el océano Ártico desde julio de 1818 hasta febrero de 1881. Durante esta expedición, los marineros registraron varias veces al día información climática, velocidad, temperatura, coordenadas, forma de las nubes, etc.

En este cuaderno, encontramos que cada fecha anotada suele corresponder a dos páginas contiguas: la página izquierda, que contiene información tabular sobre las condiciones meteorológicas a determinadas horas del día, y la página derecha, que contiene información diversa sobre los acontecimientos ocurridos ese día. Así pues, este corpus se compone de 419 páginas, de las que 208 corresponden a páginas tabulares y las otras 211 corresponden a páginas descriptivas. Ejemplos de ambos tipos de páginas en la Figura 16.

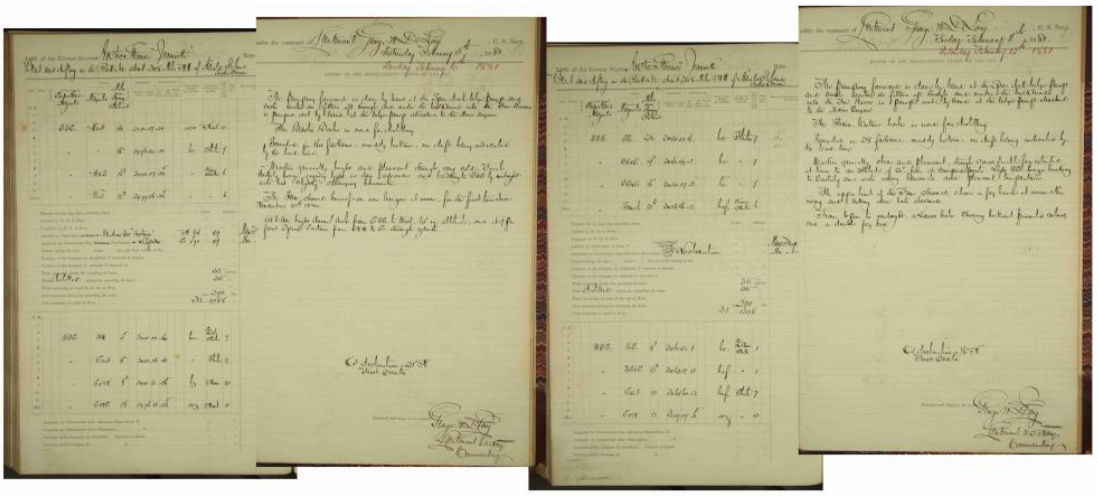


Figura 16: Ejemplos de cuadernos con anotaciones sobre las condiciones meteorológicas.

Estos documentos conllevan algunos retos relacionados con distintas áreas, como el reconocimiento de la escritura y la extracción de información. Una de las principales dificultades del análisis automático de la disposición es que la información incluida en una celda a veces se sustituye por comillas (") cuando los datos son los mismos que los de la misma columna. Estas comillas son muy cortas y a veces son difíciles de detectar. Otras dificultades de maquetación que se pueden encontrar son: datos relacionados con una celda que en realidad están escritos en las celdas superior y en las celdas inferiores, nombres de columnas tachados, palabras escritas entre celdas, y diferente número de filas en cada tabla.

En cuanto a la transcripción automática y la extracción de información, estos documentos presentan algunas dificultades a nivel óptico, léxico, sintáctico y semántico.

A nivel óptico porque las comillas y los números cortos pueden ser difíciles de transcribir. Algunas dificultades que pueden encontrarse a nivel léxico son: números escritos como superíndices, diferente notación para la misma información (por ejemplo, la palabra *nimbus* puede encontrarse escrita como *nim*, *nimb* o *nimbus*). A nivel sintáctico por la falta de contexto. Cada celda suele contener sólo un número o una palabra y su contenido no está relacionado con la información de las anteriores celdas de la misma fila. Y a nivel semántico, porque la interpretación de cada dato depende de su posición en la tabla.

Obsérvese que cada columna tiene una semántica que no está relacionada con las columnas siguientes y anteriores, sino que el valor de cada fila está condicionado por los valores de la misma columna en las filas anteriores.

Todas estas páginas fueron transcritas y utilizadas para entrenar al modelo en cuestión, incluyendo variedad de palabras, simbología e interpretaciones. Todo ello hace que el modelo sea entrenado con mayor diversidad y capacidad de predecir un mayor rango de palabras. Además, aunque la diferencia entre estos textos y las jugadas es de unos doscientos años, la manera de escribir no dista de una manera tan significativa a la actual.

Este modelo es de reciente creación y sigue las tendencias actuales propuestas como óptimas por la comunidad científica. Las tecnologías de HWCR más avanzadas se basan, como en el caso de Rodrigo, en modelos de redes neuronales profundas de varias capas convolucionales seguidas de una o más capas de RNN compuestas de "neuronas" especiales llamadas *Bidirectional Long Short Term Memory (BLSTM) units* [34] que son la evolución de las pasadas *Long short-term memory* (LSTM).

Las LSTM introducen un diseño de celda de memoria especializado que permite que la información sea almacenada, leída y escrita a lo largo de diferentes pasos de tiempo. Esto se logra mediante una estructura que incluye puertas de entrada, olvido y salida. La puerta de entrada determina cuánta información nueva se incorporará a la celda de memoria. La puerta de olvido decide cuánta información se descartará de la celda de memoria. Y, finalmente, la puerta de salida determina cuánta información de la celda de memoria se utilizará para calcular la salida en el paso de tiempo actual. Sin embargo, las redes LSTM estándar tienen una limitación en el sentido de que procesan la secuencia de datos en una sola dirección, lo que significa que la salida en un paso de tiempo dado solo puede depender de los elementos anteriores en la secuencia. La representación gráfica de esta explicación se puede encontrar en la Figura 17.

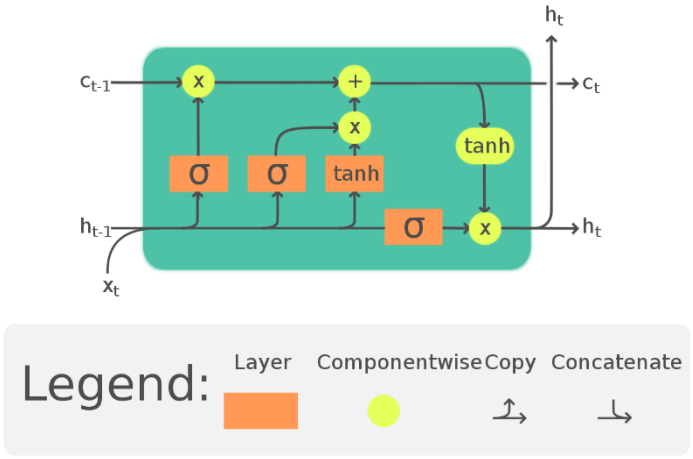


Figura 17: Ejemplo del diseño de una LSTM

Aquí es donde las redes BLSTM entran en juego. Las BLSTM amplían el concepto de LSTM al introducir una segunda capa de celdas LSTM que procesan la secuencia de datos en la dirección opuesta. De esta manera, las BLSTM pueden capturar información contextual tanto de las partes anteriores como de las futuras de la secuencia. La información de las dos direcciones se combina para generar una representación más rica de cada elemento de la secuencia.

Por último, una capa de salida calcula una estimación de las probabilidades de cada carácter del alfabeto de entrenamiento más un símbolo especial "sin carácter". La arquitectura general de arquitectura general se conoce como redes neuronales convolucionales recurrentes (CRNN) [31].

En Graves et al. (2009) se introdujo una versión más compleja y "multidimensional" de la arquitectura BLSTM, dando lugar a las llamadas redes neuronales recurrentes multidimensionales (MDLSTM), que se hicieron bastante populares durante algunos años debido a su superioridad sobre las redes neuronales recurrentes. Sin embargo, un trabajo más reciente [31] ha demostrado que, configurando adecuadamente la pila de capas convolucionales y recurrentes, se puede obtener una precisión HTR similar con arquitecturas más sencillas y eficientes.

Una CRNN se entrena por descenso de gradiente estocástico con el método RMSProp. El descenso por gradiente estocástico es una técnica de optimización ampliamente utilizada en el entrenamiento de redes neuronales y otros algoritmos de aprendizaje automático. El objetivo del entrenamiento es minimizar una función de pérdida, que cuantifica cuán lejos están las predicciones del modelo de los datos reales. El gradiente se refiere a la derivada de esta función de pérdida con respecto a los parámetros del modelo, y señala en qué dirección los parámetros deben ajustarse para minimizar la función de pérdida. El término estocástico implica que, en lugar de utilizar todo el conjunto de datos para calcular el gradiente, se toma un solo ejemplo de entrenamiento al azar en cada iteración para calcular una aproximación del gradiente. Esto hace que el proceso de entrenamiento sea más rápido y menos costoso computacionalmente, aunque más ruidoso. La representación gráfica de esto se puede ver en la Figura 18.

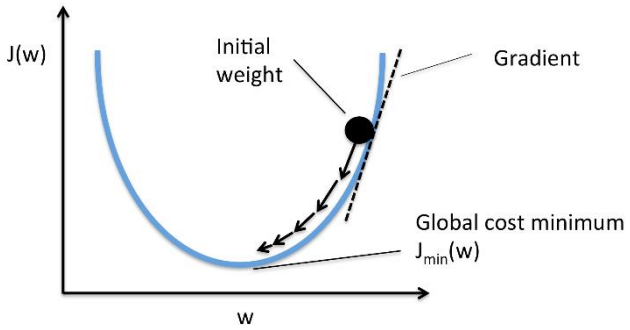


Figura 18: Descenso por gradiente estocástico

El RMSProp es un método que adapta la tasa de aprendizaje de cada uno de los parámetros del modelo de forma individual. Lo hace mediante el seguimiento de un promedio móvil del cuadrado de los gradientes y, a continuación, divide la tasa de aprendizaje por la raíz cuadrada de este promedio. Esto tiene el efecto de normalizar el gradiente, lo que a menudo resulta en un entrenamiento más estable y rápido.

Por otra parte, se utilizaron técnicas de *Dropout* para reducir el sobreajuste del entrenamiento, lo que se ha demostrado que mejora eficazmente la precisión del reconocimiento [34].

La arquitectura exacta de una CRNN típica está definida a través de un gran número de hiperparámetros. Estos incluyen, al menos, el tamaño de la imagen de entrada, en este caso y típicamente con 64 píxeles de altura, el número de capas convolucionales, el número de filtros en cada capa, los tamaños de los *kernels* y los factores de reducción de resolución (*max-pooling*) de estos filtros, el número de capas recurrentes, el tipo de

unidades recurrentes (BLSTM u otras), el número de estas unidades en cada capa recurrente, los tipos de funciones de activación utilizadas en cada capa, y el número de diferentes caracteres a predecir en la capa de salida. Además, se necesitan otros hiperparámetros para especificar los detalles del entrenamiento de la CRNN, incluyendo, al menos, el tamaño de los *mini-batches*, la tasa de *dropout*, la tasa de aprendizaje, el criterio de convergencia (posiblemente utilizando un conjunto de desarrollo) y el número de posibles iteraciones adicionales utilizando los datos de desarrollo.

Claramente, optimizar tal cantidad de hiperparámetros para cada tarea de reconocimiento de texto escrito a mano es un cuello de botella importante en el modelado. Con el objetivo de superar este cuello de botella, el conjunto de herramientas PyLaia ofrece definiciones de arquitectura predeterminadas y los correspondientes hiperparámetros de entrenamiento, que generalmente permiten en la práctica obtener modelos ópticos precisos para documentos escritos a mano en la mayoría de idiomas, períodos históricos y estilos de escritura.

Para el entrenamiento sólo se hicieron pequeñas variaciones a la configuración predeterminada de PyLaia. En general, a menos que se indique lo contrario, los modelos ópticos CRNN incluyen un conjunto de cuatro capas convolucionales y tres capas recurrentes, cada una con 256 unidades BLSTM. Los detalles completos de la arquitectura y el entrenamiento están incluidos en los scripts disponibles para cada conjunto de datos en GitHub[2]. Un ejemplo gráfico de cómo podría ser una de estas redes es el representado en la Figura 19.

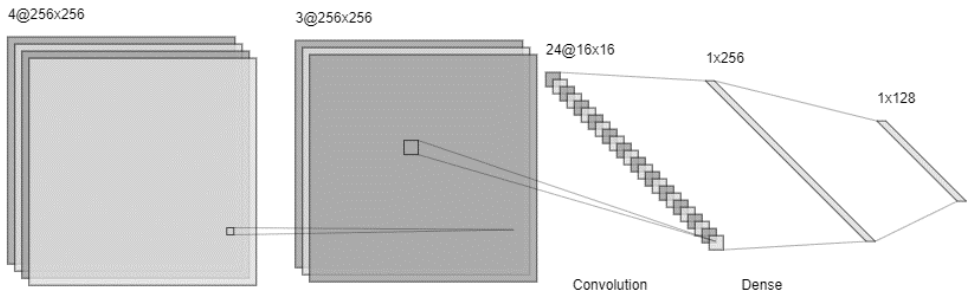


Figura 19: Ejemplo CRNN

Las variaciones que uno le quiera dar a este tipo de configuraciones vienen normalmente de la experiencia de saber qué hiperparámetros son los más adaptables dependiendo del problema que se esté tratando. Asimismo, estas modificaciones son prácticamente imperceptibles en los resultados finales.

Una vez realizado el entrenamiento se evaluó el modelo mediante las mismas métricas anteriormente comentadas, el WER y CER. Todos los resultados se pueden ver en [35].

Preparado el modelo, se aplicó sobre las casillas con texto manuscrito, no sin previamente realizar varios pre-procesos sobre las imágenes. Por un lado, toda imagen de entrada a la red debe tener una altura de 64 píxeles. Por ello se tuvieron que ajustar todas las imágenes para cumplir este requisito.

Por otro lado, el entrenamiento se realizó indicando que las imágenes estaban en modo "L", es decir, en blanco y negro. Por ello las casillas también se pusieron a blanco y negro para adaptarse y parecerse lo máximo posible a las del entrenamiento.

[2] Prhlt. (s. f.). GitHub - PRHLT/htr-contests-exps: Scripts for experiments using HTR competition datasets.

Los resultados volvieron a no adaptarse al formato de las jugadas, esta vez, aproximadamente en una de cada diez jugadas acertaba una letra, quizás fruto del azar. Esta situación se puede ver en la figura 20. Tómese en cuenta que acertar una letra de una jugada no aporta prácticamente ninguna esperanza, más allá de saber la pieza que se va a mover, y eso en los casos que se acierta. En conclusión, los resultados volvieron a ser desfavorables.

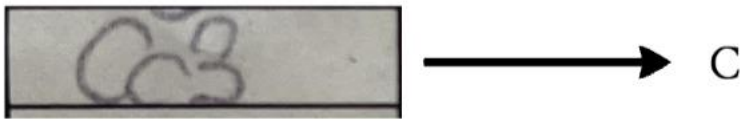


Figura 20: Jugada: Cc3 - Predicción: C

Teniendo en cuenta estos resultados, el proyecto pasaba de nuevo, a una fase donde era imprescindible entender el motivo detrás de estas predicciones tan alejadas, pues no hay cabida para la mejora sin una compresión de porqué fallan los modelos.

Los motivos, y a pesar de haberse intentado no caer en ellos de nuevo no se alejan de los anteriores, y son los siguientes:

- **Estilo de escritura:** los años donde se escribieron los textos de la base de datos de entrenamiento distan entre 100 y 200 años respecto al 2022-2023. Esto provoca una diferencia de escritura no tan grande como la anterior, pero sí lo suficientemente significativa para distorsionar los resultados de las predicciones. Esta problemática se vio tras realizar la decodificación, pues previo a ella se asumió estilos de escritura suficientemente similares.
- **Rectángulo y casilla:** las jugadas a predecir se encuentran rodeadas por un rectángulo que tras la binarización es negro, mientras que en las imágenes de entrenamientos son pocos los píxeles a negro que no forman parte del propio texto. Esto podría hacer que el modelo se confundiese.

La falta de símbolos quedó solucionado respecto al anterior modelo pues en estas transcripciones había una gran variedad de mayúsculas, minúsculas, números y caracteres especiales como el guión. Por otra parte, al haber datos navieros dentro de las imágenes, dentro de las transcripciones se encontraban “palabras” como N86, con el significado más que probable de 86 grados al norte. Este tipo de anotaciones guardan cierta similitud con las jugadas de ajedrez; a pesar de ello el modelo no fue capaz de modelar el problema.

4.4 Modelo Oculto de Markov

Probados los modelos pre-entrenados y atañéndose a los resultados obtenidos quedaba en evidencia la necesidad de buscar otros métodos para la predicción de jugadas.

Los Modelos Ocultos de Markov, abreviados HMM del inglés *Hidden Markov Models,* se han utilizado con éxito en el procesamiento y reconocimiento del habla [36]. El reconocimiento de palabras manuscritas tiene muchas similitudes con el del habla. Ambos implican el procesamiento de cadenas de símbolos con límites ambiguos y considerables variaciones en el aspecto de los símbolos.

Estos modelos fueron sustituidos por las redes neuronales puesto que, en la actualidad, no existe un buen algoritmo de segmentación que pueda separar todas las letras a la perfección. Esta es la principal limitación del propuesto en [37]. Si bien

observando los buenos resultados obtenidos mediante técnicas clásicas para el reconocimiento de la planilla, se intentó aplicar esta técnica efectiva pero ya no tan utilizada para modelar el problema.

La explicación matemática y teórica de este modelo queda fuera del alcance de este proyecto, ésta se puede encontrar en [3]. No obstante, se debe introducir unos mínimos conceptos para entender este tipo de modelos.

Un HMM es una quíntupla $M = (\Sigma, Q, \pi, A, B)$ donde:

- Σ es un conjunto de símbolos "observables". En cada instante $t = 1, 2, \ldots, T$ emite un símbolo, que se denota con x_t.
- Q es un conjunto de estados. En cada instante $t = 1, 2, \ldots, T$ está en uno de sus estados, denotado q_t.
- $\pi \in R^Q$ es un vector de probabilidades iniciales. M elige q_1 según π: $\pi_q = P(q_1 = q)$
- $A \in R^{Q \times Q}$ es una matriz de probabilidades de transición (entre estados). M elige q_{t+1} basándose en qt y A: $A_{q,q} = P(q_t+1 = q' \mid q_t = q)$.
- $B \in R^{Q \times \Sigma}$ es una matriz de probabilidades de emisión (de símbolos). M elige x_t basándose en q_t y B: $B_{q,\sigma} = P(x_t = \sigma \mid q_t = q)$.

Expresado con palabras, un HMM es una extensión de una cadena de Markov, que es un tipo especial de proceso estocástico discreto en el que la probabilidad de que ocurra un evento depende solamente del evento inmediatamente anterior. En cambio, en un HMM, se agregan estados ocultos a esta estructura. Estos estados ocultos no son directamente observables, pero producen observaciones que sí pueden ser medidas.

Un aspecto clave de los HMMs es el uso de dos tipos de probabilidades: probabilidades de transición y emisión. Las probabilidades de transición representan las probabilidades de moverse de un estado oculto a otro, mientras que las probabilidades de emisión se refieren a la probabilidad de una observación específica que se origina de un estado oculto.

Los algoritmos son fundamentales para operar con HMMs, dentro de ellos hay tres que son los principales.

- El algoritmo de Viterbi [3] es un algoritmo de programación dinámica utilizado para encontrar la secuencia de estados más probable que resulta en una secuencia de observaciones dadas, considerando un Modelo Oculto de Markov. Sea Q la secuencia de estados y O la secuencia de observaciones, el objetivo del algoritmo de Viterbi es encontrar la secuencia de estados que maximiza P(Q|O). Dado un HMM con un conjunto de estados S, y una secuencia de observaciones $O = \{o_1, o_2, \ldots, o_t\}$, el algoritmo define la probabilidad de la secuencia parcial hasta el tiempo t, q_t, como $v_t(q_t) = \max P(q_1, q_2, \ldots, q_t, o_1, o_2, \ldots, o_t)$. El algoritmo mantiene un camino óptimo hasta cada estado en cada instante de tiempo y utiliza programación dinámica para evitar el cálculo de caminos subóptimos.
- El algoritmo *Forward-Backward* [3] es un algoritmo que se utiliza para calcular las probabilidades posteriores de los estados ocultos en un Modelo Oculto de Markov. A diferencia del algoritmo de Viterbi, que encuentra el camino más probable, el algoritmo Forward-Backward calcula la probabilidad de estar en un estado específico en un momento dado, dada la secuencia de

observaciones. El algoritmo consta de dos partes:

a. *Forward*: calcula la probabilidad de las observaciones hasta el tiempo t, y la probabilidad de estar en un estado particular i en el tiempo t, denotado como $\alpha_t(i)$.

b. *Backward*: calcula la probabilidad de las observaciones restantes a partir del tiempo t + 1 hasta el final, dado que estamos en el estado i en el tiempo t, denotado como $\beta_t(i)$.

Las probabilidades posteriores se calculan utilizando las partes *forward* y *backward* como: $P(q_t = i \mid O) = \alpha_t(i) * \beta_t(i) / P(O)$.

- El algoritmo Baum-Welch [47] es un algoritmo de optimización basado en el algoritmo *Forward-Backward* y se utiliza para entrenar Modelos Ocultos de Markov. Es un caso especial del algoritmo de *Expectation-Maximization* (EM). Cuando tenemos una secuencia de observaciones, pero no conocemos la secuencia de estados, el algoritmo Baum-Welch se puede utilizar para estimar los parámetros del modelo (matriz de transición de estados, matriz de emisión y probabilidades iniciales) que maximizan la probabilidad de las observaciones dadas. Durante la fase de *Expectation*, el algoritmo calcula las probabilidades posteriores de los estados utilizando el algoritmo *Forward-Backward*. En la fase de *Maximization*, reestima los parámetros del modelo utilizando estas probabilidades. El algoritmo itera entre estas dos fases hasta que converja a un conjunto de parámetros que maximiza la probabilidad de las observaciones.

Introducidos los HMM, éstos en combinación con redes neuronales para el reconocimiento de texto manuscrito puede potenciar significativamente el rendimiento en términos de precisión y eficiencia. Por ello se decidió juntar un HMM con las redes pre-entrenadas previamente analizadas. A continuación, se presenta un esquema general para incorporar un modelo pre-entrenado de red neuronal en un HMM.

1. **Extracción de Características:** se empieza por extraer características de las imágenes de texto manuscrito. Las características pueden incluir elementos como contornos, direcciones de trazo y densidades de píxeles. De esta parte se encarga el modelo pre-entrenado
2. **Modelado de HMM:** se construye un modelo HMM para representar secuencias de caracteres o palabras. Cada estado en un HMM podría representar un carácter o un conjunto de características de caracteres. Los HMMs son especialmente buenos para modelar secuencias temporales y espaciales, lo cual es crítico en el reconocimiento de texto manuscrito, ya que los caracteres pueden tener diversas formas y estilos.
3. **Integración de Red Neuronal y HMM:** se integra la red neuronal dentro del HMM utilizando las salidas de la red neuronal (por ejemplo, las probabilidades de las diferentes clases de caracteres) como las probabilidades de emisión de los HMM. Esto permite que el HMM haga uso de la capacidad de discriminación de la red neuronal mientras modela las dependencias secuenciales entre caracteres.
4. **Decodificación:** Se utilizan algoritmos como el algoritmo de Viterbi para encontrar la secuencia más probable de estados que mejor explique las observaciones (en este caso, las características de las imágenes de texto manuscrito). Esto dará como resultado la secuencia de caracteres reconocida.

Realizados estos pasos se obtuvieron todas las predicciones para las jugadas que seguían sin mejorar lo ya comentado. Todos estos resultados adversos condujeron a una clara conclusión cómo era la necesidad de aplicar conocimiento ajedrecístico en el proceso.

Para abordar este desafío, se ha empleado una combinación de Modelos Ocultos de Markov (HMM) con un modelo de lenguaje específico para las jugadas de ajedrez. Este modelo de lenguaje es una lista que contiene todas las opciones posibles de jugadas de ajedrez. A priori podría parecer una lista breve; sin embargo, hay que tener en cuenta que hay 6 piezas diferentes en el tablero, cada pieza puede ir a las 64 casillas, puede ir a ellas capturando otra pieza (se le añade una "x" en la anotación), puede hacer jaque, y puede ir capturando y haciendo jaque a la vez. Además de otras situaciones especiales tales como que dos piezas iguales pueden ir a la misma casilla. En total hablamos de más de 5700 posibilidades.

Para aumentar la precisión en el reconocimiento de jugadas de ajedrez escritas a mano, es fundamental que el sistema no solo sea capaz de identificar caracteres y símbolos, sino que también comprenda la estructura y reglas sintácticas de la notación de ajedrez. Es aquí donde la integración de un modelo de lenguaje con los Modelos Ocultos de Markov se vuelve esencial.

La integración de ambos modelos se realiza mediante la combinación de las probabilidades de emisión de los HMMs con las probabilidades del modelo de lenguaje. Concretamente, durante el proceso de decodificación, cuando el HMM genera una secuencia de estados candidatos basada en las observaciones, el modelo de lenguaje se utiliza para reevaluar la probabilidad de cada secuencia candidata.

Este modelo puede parecer el óptimo, pues es el único hasta ahora en el que se ha incorporado conocimiento en su proceso; sin embargo, al tener todas las jugadas la misma probabilidad de salida y el modelo de HMM no ser capaz de reconocer el estilo de escritura, la salida era totalmente azarosa y con resultados fallidos.

Teniendo en cuenta todos los modelos analizados, el **conocimiento extraído** de todos ellos es la necesidad de la creación de un modelo únicamente entrenado con jugadas de ajedrez. Ningún modelo pre-entrenado ni técnicas como los modelos de Markov han sido capaces de modelar la problemática y no es más que por lo diferente, peculiar y compleja que es en sí.

Las jugadas de ajedrez tienen un contexto, una forma de anotación, un estilo de escritura y un vocabulario prácticamente único en el mundo. Por ello debe tratarse de tal forma y crear un modelo específico para este contexto.

Bien es sabido que la creación de un modelo no es una tarea sencilla. Este punto será tratado en el siguiente capítulo correspondiente a los futuros trabajos. Esta tarea conllevaría la creación de una base de datos, ya no de planillas de ajedrez, si no de recortes de casillas con jugadas de ajedrez manuscritas, y es este etiquetado el que hace la tarea especialmente tediosa y larga.

Este recortado de la jugada se podría realizar mediante el algoritmo creado para el reconocimiento de la planilla; aun así, al no tener una fiabilidad del 100% haría la tarea imprecisa.

Adicionalmente, y a pesar de no haberse llevado a cabo en este proyecto, se podría intentar incorporar conocimientos en los modelos de Rodrigo y HisClima para comprobar si existe capacidad de mejora en ellos. Este proceso es técnicamente complejo y fuera de los conocimientos del grado.

Además, PyLaia como *toolkit* ofrece una gran herramienta para el entrenamiento modelos como los comentados; sin embargo, su total control y entendimiento requiere de una capacidad amplia de exploración y asimilación de conceptos, pues en ciertos momentos la documentación no está suficientemente detallada.

Unido a todo esto, PyLaia está desarrollada en Python, pero su uso es a través de la línea de comandos de Linux [39], aumentando la complejidad en ciertos momentos por cuestiones de dependencias entre sistemas operativos y librerías.

CAPÍTULO 5
Trabajos futuros

A medida que la ciencia avanza incesantemente hacia nuevos horizontes, es imperativo reconocer que las investigaciones actuales son solamente un eslabón en la cadena infinita del conocimiento. La sección de “Futuros trabajos” juega un papel cardinal en esta travesía académica, actuando como puente entre el presente y lo que está por venir. Este segmento no solo enfatiza las oportunidades para expandir y mejorar los hallazgos actuales, sino que también invita a la comunidad científica a abordar preguntas aún sin respuesta y a explorar territorios desconocidos. En un mundo donde los problemas evolucionan con una rapidez vertiginosa, la adaptabilidad y la visión prospectiva son herramientas cruciales para los investigadores. La sección de "Futuros trabajos" representa una brújula que guía la dirección de la indagación científica, asegurando que siga siendo relevante, rigurosa y en sintonía con las demandas cambiantes de la sociedad.

En concreto, en este capítulo se hablará de las posibilidades de mejora de este trabajo, las posibilidades no exploradas pero que deben serlo y de todo lo necesario para que un proyecto de *Computer Vision* y ajedrez pueda tener éxito y renombre.

El reconocimiento de la planilla como tabla a través de técnicas clásicas puede considerarse exitoso. Todo éxito es mejorable, y con el algoritmo desarrollado pasa lo mismo. La perfección es prácticamente inalcanzable, en este caso sería el 100% de *accuracy*; sin embargo, hasta llegar a ella quedan pasos por darse.

La principal y mejor opción para conseguir esa deseada mejora es la aplicación de técnicas de aprendizaje automático en la tarea como se ha hecho en anteriores ocasiones [40][41]. Las técnicas clásicas han demostrado seguir siendo una buena manera de reconocer tablas, pero computacionalmente son costosas. El actual algoritmo, dependiendo de las características del ordenador donde se ejecute, tarde entre un minuto y medio y dos. Esto lleva a pensar en las nuevas tecnologías, principalmente en las redes neuronales, que están acaparando el mercado y toda la atención de las grandes compañías. Estos modelos pueden reportar mayor precisión en la predicción, pero es necesario la creación de una base de datos con fotografías de planillas y su etiquetado y transcripción. Posteriormente será debatido como debe crearse esta base de datos.

En el capítulo 4 quedó demostrada la ineficacia de los modelos pre-entrenados y los modelos ocultos de Markov, y que por ello era necesario construir un modelo específico capaz de modelar este problema.

Para entrenar una red se requiere de una cantidad ingente de datos, en este caso de imágenes de jugadas de ajedrez encapsuladas en casillas que conforman una planilla. Como se ha ido comentando a lo largo de este trabajo, la cantidad de imágenes con las que contábamos era reducida, en total 70 planillas de las cuales solamente 11 fueron etiquetadas manualmente indicando los cruces de la planilla.

El motivo detrás de esto no es otro que, una vez finalizada la partida, lo primero que se hace entre la mayoría de los ajedrecistas es comentar y transcribir la partida. Una vez finalizado este proceso la planilla deja de ser útil pues ya se tiene la información en el dispositivo móvil u ordenador, no se necesita más el papel y, o se tira la planilla o se conserva durante un cierto tiempo si se guarda un especial cariño a la partida.

Además, la gente es ligeramente recelosa a compartir información sobre sus partidas. Una parte muy importante de una partida de ajedrez es la apertura, es decir, los primeros movimientos. Cada cual tiene una manera de empezar la partida y existe una estrategia altamente compleja entorno a esto para que la gente no sepa cuáles son tus aperturas.

Realizando unos cálculos rápidos, suponiendo que se necesitan alrededor de 30000 jugadas para entrenar una red lo bastante buena como para modelar el problema, y teniendo en cuenta que una partida de ajedrez tiene de media 30 jugadas, se necesitarían 1000 planillas aproximadamente. Proporcionando una visión de lo que esto significa, los jugadores federados de la comunidad valenciana que más partidas juegan al año rondan las 100 partidas, es decir, necesitarían 10 años para conseguir las 1000 planillas.

Así pues, este número de partidas puede ser conseguido exclusivamente con la ayuda de las organizaciones y federaciones de ajedrez. En las competiciones oficiales todo jugador acepta una serie de puntos estándares a todos los torneos como la aceptación del derecho de la organización a compartir sus partidas en caso de lo que consideren oportuno. En la realidad, solo las partidas que se retransmiten en *streaming* acaban siendo publicadas. Además, las organizaciones y federaciones están obligadas, según el reglamento de competición de cada comunidad autónoma, a conservar las planillas durante un cierto tiempo.

Teniendo en cuenta todo lo comentado, organizaciones y federaciones juegan un rol clave en la creación de esta base de datos, pues son las únicas con la capacidad de recabar tal cantidad de información además de estar habilitadas para compartirla legalmente. Para ello, se necesitaría contactar con una organización privada o una federación, explicar el proyecto y durante un torneo fotografiar las planillas de todas las partidas una vez finalizadas. En un torneo de 200 personas con 9 rondas se crearían 900 planillas diferentes, es decir, en dos torneos se tendría el número de planillas, incluso sobrepasándolo, para formar la base de datos.

Ahora bien, la foto de las planillas no es suficiente. Toda planilla requiere pasar un proceso de extraer las casillas con jugadas manuscritas. Este proceso es el descrito en el capítulo 3 de este trabajo. Su *precision* y *recall* es superior a 0.85 en ambos casos; sin embargo, ciertas casillas no consigue extraerlas de manera exacta provocando una posible pérdida de información.

Esta casuística se puede plantear desde dos puntos de vista: el primero de ellos consiste en no tener en cuenta esta información, confiar en el algoritmo y que extraiga todas las casillas posibles y después ir descartando las que hayan sido erróneamente reconocidas. Este enfoque ahorra mucho tiempo en comparación con el otro enfoque posible que sería el etiquetado manual. Apuntar todos los cruces de una planilla es totalmente exasperante, y a pesar de asegurar un 100% de *accuracy* sólo las grandes empresas como Google, Amazon o Facebook tienen esa capacidad de etiquetado.

Conseguida esta base de datos pasaríamos a la parte del reconocimiento del texto. Cómo se ha podido comprobar, los modelos pre-entrenados y las técnicas clásicas no son capaces de adaptarse al problema y sus predicciones carecen de utilidad práctica.

Para ello se debería entrenar una red neuronal de unas características parecidas a las ya presentadas, pero únicamente entrenadas con imágenes de jugadas. Esto aseguraría la totalidad del vocabulario de ajedrez y una red exclusivamente válida para este contexto, a la vez de excepcionalmente especializada en ajedrez.

Una vez entrenada esta red debe ser evaluada con las métricas CER y WER y ver si sus resultados son óptimos. En caso de que no lo viesen estaríamos ante un punto crítico del proyecto, y su viabilidad se vería cuestionada.

En el supuesto de que la evaluación del modelo fuese satisfactoria, supuesto totalmente razonable, llegaríamos al momento de introducir conocimiento al proceso. Esta parte del proceso requiere de expertos en HWCR, pues queda fuera del alcance de cualquier proyecto desarrollado en un grado.

Este conocimiento puede venir desde dos perspectivas diferentes, dependientes ambas dos de la capacidad del algoritmo anteriormente desarrollado en reconocer las casillas, ya sea el algoritmo actual basado en técnicas clásicas o un futuro algoritmo basado en redes neuronales.

El primer criterio, como criterio más restrictivo en cuanto al espacio de búsqueda, es conseguir acotar la predicción a que la jugada sea legal teniendo en cuenta la situación del tablero. Es decir, en la posición inicial del ajedrez existen únicamente 20 opciones por cada color. Por lo tanto, la salida de la primera predicción debe estar obligatoriamente entre estas 20. Es de suma importancia reseñar que lo que se hace no es reducir el número de salidas, si no de todas las salidas que da el modelo como posibilidades con su correspondiente probabilidad de aparición se escogerá como jugada elegida aquella que sea legal y tenga mayor probabilidad. La representación gráfica de esta mejora se puede ver en la Figura 21, donde se debería descartar la primera y cuarta predicción debido a que no existen las columnas “l” ni “p”, quedándote entonces con e4 como jugada legal en el primer movimiento de las blancas con mayor predicción.

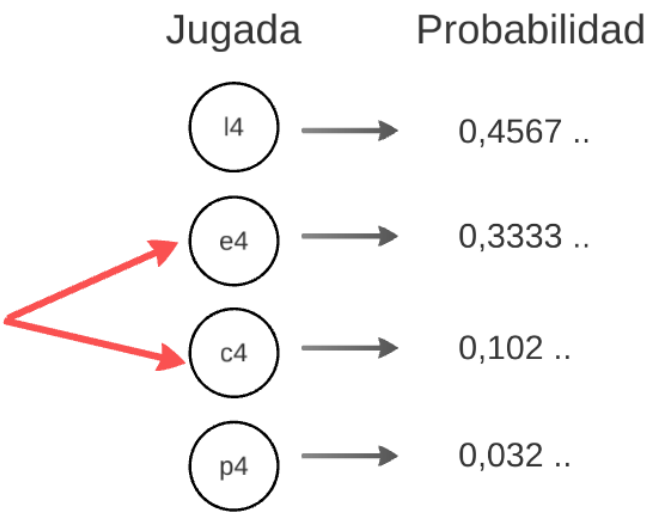


Figura 21: Restringiendo el espacio de búsqueda

Las desventajas de este enfoque provienen de la posibilidad de una predicción incorrecta. Si la primera jugada de la partida se ha predicho erróneamente las posteriores predicciones se verán afectadas por este error, arrastrándose durante toda la partida. Esto podría producir que una planilla donde todas menos la primera jugada fuesen a ser

predichas correctamente pase a ser una partida totalmente diferente a la escrita. En conclusión, este método podría obtener excelentes resultados, pero es bastante arriesgado.

En consecuencia, se puede usar el modelo de lenguaje creado con las 5700 posibilidades de jugadas para los modelos de Markov pero enfocado a las redes neuronales. Actualmente, ese modelo les da el mismo peso y probabilidad a todas las jugadas; sin embargo, no todas las jugadas aparecen con la misma frecuencia en las partidas de ajedrez. Por ejemplo, la jugada inicial "e4" que indica que el peón de rey se mueve dos casillas al frente es mucho más común que la jugada "0-0-0+", que indicaría enroque largo dando jaque.

Para ello se necesitaría una base de datos con las transcripciones de millones de partidas y qué mejor que la creada por Isaac Lozano llamada Yottabase [9]. Esta base de datos es pública y mediante técnicas de *web scraping* se podría extraer los millones de partidas que existen en esta página web.

Una vez extraídas las partidas se debería realizar un análisis de la frecuencia de aparición de cada jugada para posteriormente asignarle un cierto peso. Con estos pesos asignados a la hora de realizar la predicción se podría tener en cuenta cuál es la probabilidad de que una cierta jugada aparezca, combinado evidentemente con el trazo de la jugada.

A modo de experimentación se podría probar a incluir como conocimiento el número de la jugada. En ajedrez, los alfiles y los caballos suelen ser las primeras piezas en moverse, mientras que las torres y las damas suelen tardar más. Este conocimiento puede aportar información valiosa a la hora de predecir el quinto movimiento de una partida, pues se sabría que es más posible que se mueva un caballo que una dama y por lo tanto la letra "C" sería más común que apareciese que la "D".

Todas estas propuestas requieren de grandes habilidades en programación, aprendizaje automático, procesamiento del lenguaje y altos conocimientos en redes neuronales, que sólo pueden ser obtenidos en el campo de la investigación científica.

CAPÍTULO 6
Conclusiones

En esta último capítulo previo al legado y a la relación del trabajo desarrollado con los estudios cursados se expondrán las conclusiones y un análisis obtenido de los diferentes procesos realizados.

Este trabajo final de grado se puede dividir en dos partes principales. La primera de ellas, y marcado como primer objetivo, es el reconocimiento de una planilla de ajedrez. Una planilla de ajedrez vista desde un ángulo puramente conceptual es una tabla con información manuscrita en ella.

El reconocimiento de tablas ha sido ampliamente tratado en la literatura. En este caso, y adaptándose a los conocimientos vistos en el grado y más concretamente en la asignatura de análisis de imágenes y vídeos, se eligió decantarse por las técnicas clásicas como método para alcanzar este objetivo.

Se realizaron diversos intentos basados en la binarización de la imagen, tanto en la imagen completa como por zonas, pero por cuestiones de luminosidad de las fotografías no pudo llevarse a cabo.

Otro de los intentos llevados a cabo consistió en el análisis de la luminosidad de las planillas. Al fin y al cabo, el objetivo final era encontrar los puntos de cruces y para ello se necesitaba encontrar las líneas de la tabla. Estas líneas eran de color negro, provocando una bajada en el valor de intensidad luminosa cuando se llegaba a ellos. Por ello se trató de buscar los diferentes mínimos locales de la imagen tanto vertical como horizontalmente, tanto en la imagen completa como dividiéndola por zonas; sin embargo, y principalmente debido a un problema con las líneas verticales, este proceso no resultó fructuoso.

Tras estos intentos se dio con la pieza clave de puzzle y fue la expansión de píxeles vertical y horizontalmente. Este algoritmo, unido a las diferencias relativas para la detección de las casillas, creaba una estructura de planilla análoga a la original. Los resultados de las métricas satisficieron las expectativas iniciales conteniendo una *precision* superior a 0.90 y un *recall* superior a 0.85. Los resultados finales pueden encontrarse en la matriz de confusión de la Figura 13. La única limitación unida a este algoritmo es el coste temporal, de unos dos minutos, siendo este inferior al tiempo que se tarda en transcribir manualmente una planilla.

Una vez localizado la gran parte de texto manuscrito se debía proceder a su reconocimiento y predicción. Esta tarea correspondiente al segundo objetivo del trabajo fue tratada de tres maneras diferentes.

En primer lugar, el modelo pre-entrenado de Rodrigo. Este modelo no fue capaz de modelar el problema debido a la gran diferencia entre estilos de escritura y la falta de simbología para ciertas jugadas.

En segundo lugar, el modelo HisClima. Este modelo contenía estilos de escritura diversos y más recientes. Su entrenamiento fue con diversos textos y las métricas de evaluación avalaban su rendimiento. No obstante, su aplicación en el contexto de las jugadas de ajedrez no fue el esperado, y quedó descartado como solución factible.

Como última opción, se hizo una prueba con los métodos previos al *boom* de las redes neuronales como lo fueron los modelos ocultos de Markov. De nuevo, los resultados no reportaron ninguna predicción válida.

En conclusión, el objetivo inicial del reconocimiento de la planilla puede darse como cumplido dado el resultado de las métricas. Respecto al segundo objetivo, a pesar de no haberse alcanzado, cabe reseñar los diferentes intentos realizados y el conocimiento extraído de ellos que permitirá en próximos trabajos conocer la dificultad del modelado del problema y la necesidad de crear una red neuronal entrenada exclusivamente con jugadas de ajedrez.

Finalmente, a nivel personal, el ajedrez es mi pasión desde que a los 6 años mi iaia me enseñó a jugar. Haber podido trabajar en un proyecto relacionado con ello, asentar unas bases para futuros trabajos relacionados con ello, y haber podido expandir mi conocimiento respecto a los modelos de reconocimiento de escritura ha sido una experiencia definitivamente positiva.

CAPÍTULO 7
Legado y relación del trabajo desarrollado con los estudios cursados

El legado de este proyecto son los cimientos de futuros trabajos relacionados con el *computer vision* y el ajedrez. El beneficio que se puede sacar de él, en caso de materializarse en una aplicación funcional, es el renombre que aporta dentro de la comunidad científica y ajedrecística, acompañado probablemente de un reporte monetario.

En cualquier caso, la futura base de datos, el modelo de predicción de texto y la aplicación serían *open source.* El conocimiento, y más aún cuando es tan novedoso, debe ser compartido. De hecho, una opción para generar cierto ruido sobre el proyecto, siempre y cuando se encuentre creada/disponible una base de datos de jugadas, es plantear una competición de Kaagle [42] con un cierto *prizepool* para comprobar hasta dónde puede llegar la precisión de un modelo de reconocimiento de texto.

Organizaciones, jugadores y federaciones están empezando a interesarse por este proyecto y todo los relacionados con las nuevas tecnologías, pues como se ha comentado, en ciertos aspectos el ajedrez sigue anclado en el pasado.

Respecto a la relación que guarda este trabajo con los estudios cursados principalmente se relaciona con la asignatura de cuarto “Análisis de imágenes y video”. Otra asignatura relacionada podría ser la también realizada en cuarto llamada “Técnicas escalables en aprendizaje automático”. Esta asignatura era el pico de una serie de asignaturas relacionadas con el *machine learning* y modelos de regresión y clasificación.

Por otra parte, las competencias transversales desarrolladas han sido, por parte de la asignatura de análisis de imágenes y video, el análisis y resolución de problemas, más concretamente la resolución de problemas sobre imágenes, y el conocimiento de problemas contemporáneos.

Dentro de esta última competencia transversal, gran parte del trabajo ha consistido en leer artículos científicos para documentarse y conocer cuál era el estado del arte tanto en el reconocimiento de tablas como de texto manuscrito. Ambos temas, a pesar de ser de plena actualidad, no son tan tendencia como la inteligencia artificial o los modelos de lenguaje como GPT.

En el caso de las competencias transversales relacionadas con la asignatura de técnicas escalables en aprendizaje automático destaca principalmente la “instrumental específica”.

Esta competencia hace referencia al uso de software específicos para tareas relacionados con el aprendizaje automático. PyLaia encaja perfectamente en esta descripción.

Por último, ambas asignaturas en sus apartados de las competencias que se deben adquirir hacen hincapié en el desarrollo de habilidades de aprendizaje necesarias para emprender estudios posteriores con un alto grado de autonomía. Un trabajo final de grado requiere de una gran autonomía y de una gestión eficaz del tiempo para entregar dentro de los *deadlines* establecidos.

Bibliografía

[1] Garzón, J. A. (2005). The Return of Francesch Vicent: The History of the Birth and Expansion of Modern Chess.

[2] Pandolfini, B. (1997). Kasparov and Deep Blue: The Historic Chess Match Between Man and Machine. Simon and Schuster.

[3] Bishop, C. M., & Nasrabadi, N. M. (2006). Pattern recognition and machine learning (Vol. 4, No. 4, p. 738). New York: springer.

[4] Kakani, B. V., Gandhi, D., & Jani, S. (2017, July). Improved OCR based automatic vehicle number plate recognition using features trained neural network. In 2017 8th international conference on computing, communication and networking technologies (ICCCNT) (pp. 1-6). IEEE.

[5] Luqman, H. M., & Zaffar, M. (2016, May). Chess brain and autonomous chess playing robotic system. In 2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC) (pp. 211-216). IEEE.

[6] Tian, Y., Ma, J., Gong, Q., Sengupta, S., Chen, Z., Pinkerton, J., & Zitnick, L. (2019, May). Elf opengo: An analysis and open reimplementation of alphazero. In International conference on machine learning (pp. 6244-6253). PMLR.

[7] Czyzewski, M. A., Laskowski, A., & Wasik, S. (2020). Chessboard and chess piece recognition with the support of neural networks. Foundations of Computing and Decision Sciences, 45(4), 257-280.

[8] Karayaman. (s. f.). GitHub - karayaman/lichess-with-a-real-board: Lichess.org client for real life chess boards. GitHub. https://github.com/karayaman/lichess-with-a-real-board

[9] Osorio, I. L. (2022, 14 noviembre). YottaChess - New era of online chess broadcasting system. YottaChess. https://www.yottachess.com/live

[10] Darmatasia, & Fanany, M. I. (2017). Handwriting recognition on form document using convolutional neural network and support vector machines (CNN-SVM). https://doi.org/10.1109/icoict.2017.8074699

[11] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

[12] Toledo, J., Carbonell, M., Fornés, A., & Lladós, J. (2019). Information extraction from historical handwritten document images with a context-aware neural model. Pattern Recognition, 86, 27-36.

[13] Mocholí Calvo, C. (2018). Development and experimentation of a deep learning system for convolutional and recurrent neural networks (Master Thesis, Universitat Politècnica de València).

[14] Anitei, D. (2021). Reconocimiento automático de un censo histórico impreso sin recursos lingüísticos.

[15] Rong, W., Li, Z., Zhang, W., & Sun, L. (2014, August). An improved CANNY edge detection algorithm. In 2014 IEEE international conference on mechatronics and automation (pp. 577-582). IEEE.

[16] Cichy, R. M., & Kaiser, D. P. (2019). Deep Neural Networks as Scientific Models. Trends in Cognitive Sciences, 23(4), 305-317. https://doi.org/10.1016/j.tics.2019.01.009

[17]Zanibbi, R., Blostein, D., & Cordy, J. R. (2004). A survey of table recognition: Models, observations, transformations, and inferences. Document Analysis and Recognition, 7, 1-16. https://doi.org/10.1007/s10032-004-0120-9
[18]Jpuigcerver. (s. f.). GitHub - jpuigcerver/PyLaia: A deep learning toolkit specialized for handwritten document analysis. GitHub. https://github.com/jpuigcerver/PyLaia
[19]Ribeiro, B., Gonçalves, I., Santos, S., & Kovacec, A. (2011). Deep Learning Networks for Off-Line Handwritten Signature Recognition. En Lecture Notes in Computer Science (pp. 523-532). Springer Science+Business Media. https://doi.org/10.1007/978-3-642-25085-9_62
[20]Gagnon, L., Lalonde, M., Beaulieu, M., & Boucher, M. C. (2001, July). Procedure to detect anatomical structures in optical fundus images. In Medical imaging 2001: Image processing (Vol. 4322, pp. 1218-1225). SPIE.
[21]Chen, X., Ling, J., Wang, S., Yang, Y., Luo, L., & Yan, Y. (2021). Ship detection from coastal surveillance videos via an ensemble Canny-Gaussian-morphology framework. The Journal of Navigation, 74(6), 1252-1266.
[22]Forsyth, D. A., & Ponce, J. (2002). Computer vision: a modern approach. prentice hall professional technical reference.
[23]Shubh, Agarwal, M., Hassan, F., Pandey, G., & Ghosh, S. (2021). Handwriting recognition using deep learning. In Emerging Trends in Data Driven Computing and Communications: Proceedings of DDCIoT 2021 (pp. 67-81). Springer Singapore.
[24]OpenCV: OpenCV modules. (s. f.). https://docs.opencv.org/4.x/index.htmlç
[25]Chowdhary, K., & Chowdhary, K. R. (2020). Natural language processing. Fundamentals of artificial intelligence, 603-649.
[26]El-Sawy, A., El-Bakry, H., & Loey, M. (2017). CNN for handwritten arabic digits recognition based on LeNet-5. In Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2016 2 (pp. 566-575). Springer International Publishing.
[27]Simard, P. Y., Steinkraus, D., & Platt, J. C. (2003, August). Best practices for convolutional neural networks applied to visual document analysis. In Icdar (Vol. 3, No. 2003).
[28]Medsker, L., & Jain, L. C. (Eds.). (1999). Recurrent neural networks: design and applications. CRC press.
[29]Targ, S., Almeida, D., & Lyman, K. (2016). Resnet in resnet: Generalizing residual architectures. arXiv preprint arXiv:1603.08029.
[30]Mithe, R., Indalkar, S., & Divekar, N. (2013). Optical character recognition. International journal of recent technology and engineering (IJRTE), 2(1), 72-75.
[31]J. Puigcerver, "Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition" 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, Japan, 2017, pp. 67-72, doi: 10.1109/ICDAR.2017.20.
[32]PyTorch. (s. f.). https://pytorch.org/
[33]Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., & Schmidhuber, J. (2008). A novel connectionist system for unconstrained handwriting recognition. IEEE transactions on pattern analysis and machine intelligence, 31(5), 855-868.
[34]Bluche, T. (2015). Deep neural networks for large vocabulary handwritten text recognition (Doctoral dissertation, Paris 11).
[35]Sánchez, J. A., Romero, V., Toselli, A. H., Villegas, M., & Vidal, E. (2019). A set of benchmarks for handwritten text recognition on historical documents. Pattern Recognition, 94, 122-134.
[36]Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE, 77(2), 257-286.

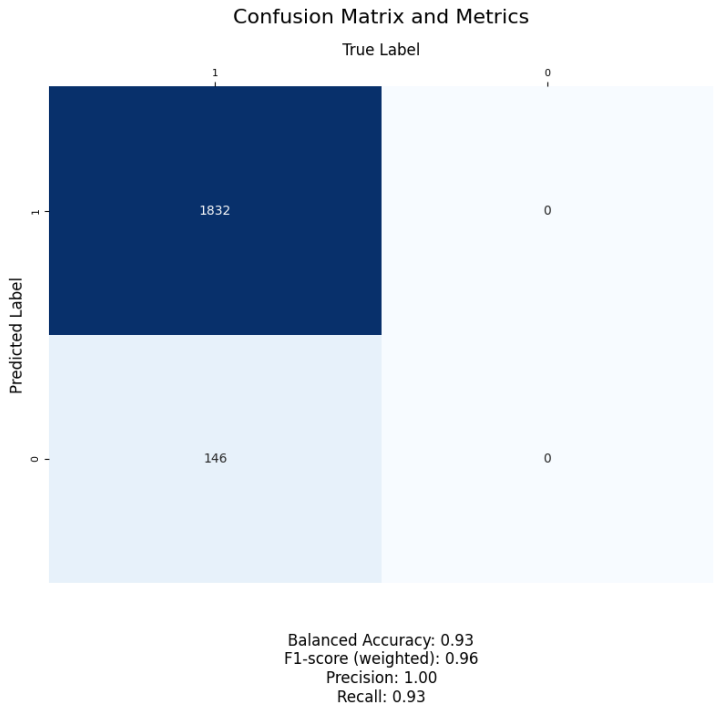
[37] Kundu, A., He, Y., & Bahl, P. (1989). Recognition of handwritten word: first and second order hidden Markov model based approach. Pattern recognition, 22(3), 283-297.
[38] Saul, L., & Pereira, F. (1997). Aggregate and mixed-order Markov models for statistical language processing. arXiv preprint cmp-lg/9706007.
[39] Linux.org. (2023, 16 junio). Linux.org. https://www.linux.org/
[40] Andrés, J., Prieto, J. R., Granell, E., Romero, V., Sánchez, J. A., & Vidal, E. (2022, May). Information extraction from handwritten tables in historical documents. In Document Analysis Systems: 15th IAPR International Workshop, DAS 2022, La Rochelle, France, May 22–25, 2022, Proceedings (pp. 184-198). Cham: Springer International Publishing.
[41] Constum, T., Kempf, N., Paquet, T., Tranouez, P., Chatelain, C., Brée, S., & Merveille, F. (2022, May). Recognition and Information Extraction in Historical Handwritten Tables: Toward Understanding Early 20 th Century Paris Census. In Document Analysis Systems: 15th IAPR International Workshop, DAS 2022, La Rochelle, France, May 22–25, 2022, Proceedings (pp. 143-157). Cham: Springer International Publishing.
[42] Kaggle: Your Machine Learning and Data Science Community. (s. f.). https://www.kaggle.com/
[43] Gibson, J. D., & Bovik, A. (Eds.). (2000). Handbook of image and video processing.
[44] Graves, A., & Graves, A. (2012). Long short-term memory. Supervised sequence labelling with recurrent neural networks, 37-45.
[45] Yujian, L., & Bo, L. (2007). A normalized Levenshtein distance metric. IEEE transactions on pattern analysis and machine intelligence, 29(6), 1091-1095.
[46] Verónica Romero and Joan Andreu Sánchez. “The HisClima database: historical weather logs for automatic transcription and information extraction”. In: 2020 25th International Conference on Pattern Recognition (ICPR). IEEE. 2021, pp. 10141–10148.
[47] Yang, F., Balakrishnan, S., & Wainwright, M. J. (2017). Statistical and computational guarantees for the Baum-Welch algorithm. The Journal of Machine Learning Research, 18(1), 4528-4580.

APÉNDICE A
Resultados algoritmo

En esta sección se añadirán todos los resultados que aportan información provechosa al proyecto, pero quedan fuera de los resultados necesarios para el seguimiento del proyecto.

A.1 Resultados algoritmo reducido

En la Figura 13 se han presentado los resultados finales del algoritmo de reconocimiento de la estructura de la planilla; sin embargo, la importancia del reconocimiento radica en la capacidad del programa en detectar los rectángulos representados en la Figura 3, correspondientes al lugar de escritura de las jugadas. Los resultados presentan una mayor precisión en comparación con toda la planilla.



Estos resultados son engañosos, pues al solamente tratarse los puntos clave de la imagen se ha descartado incluir falsos positivos, pues, por ejemplo, el programa podría incluir como falso positivo una cruz que simplemente no forma parte de la cuadrícula que encapsula la jugada. Aún así cabe destacar el gran recall obtenido y los pocos falsos negativos.

APÉNDICE B
Código PyLaia

En esta sección se añadirán capturas del código que se debe ejecutar por la línea de comando de Linux en un entorno válido para PyLaia. En el caso de tener Windows en su dispositivo se deberá instalar el subsistema de Linux para Windows, y crear un environment con la versión 3.8 de Python para trabajar con PyLaia. Además, se deberán instalar librerías como SRILM u OpenFST. Sus instalaciones no son triviales.

- **CREACIÓN DEL MODELO:**

```
pylaia-htr-create-model \
--fixed_input_height 64 \
--save_model true \
--common.train_path ./models/Optical \
--common.model_filename Rodrigo.net \
--logging.level INFO --crnn.rnn_units 512 \
--crnn.rnn_layers 3 --crnn.use_masks true \
--crnn.cnn_batchnorm [true,true,true,true] \
--crnn.cnn_kernel_size [3,3,3,3] \
--crnn.cnn_num_features [16,16,32,32] \
--crnn.cnn_dilation [1,1,1,1] --crnn.num_input_channels 1 \
--crnn.cnn_poolsize [2,2,0,2] --crnn.rnn_type LSTM \
--crnn.lin_dropout 1 \
data/lists/symbols_train.lst
```

- **ENTRENAMIENTO DEL MODELO:**

```
pylaia-htr-train-ctc --trainer.gpus 1 \
--logging.level INFO --logging.to_stderr_level INFO \
--logging.filepath train-crnn.log \
--common.experiment_dirname . \
--common.train_path models/Optical \
--common.model_filename Rodrigo.net \
--data.batch_size 15 --optimizer.learning_rate 0.0003 \
--trainer.max_epochs 60 --train.delimiters ["<space>"] \
data/lists/symbols_train.lst [data/Corpus_clean_lines/] \
data/text/train_char.txt data/text/val_char.txt
```

- **DECODIFICACIÓN:**

```
pylaia-htr-decode-ctc --trainer.gpus 1 \
--common.model_filename models/Optical/Rodrigo.net \
--common.experiment_dirname models/Optical \
--img_dirs [data/Corpus_clean_lines] \
--data.batch_size 40 \
data/lists/symbols_train.lst data/lists/test.lst | \
sort -k1 > results/test_chars.hyp
```

APÉNDICE C
Relación ODS

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. Fin de la pobreza.			x	
ODS 2. Hambre cero.			x	
ODS 3. Salud y bienestar.			x	
ODS 4. Educación de calidad.			x	
ODS 5. Igualdad de género.			x	
ODS 6. Agua limpia y saneamiento.			x	
ODS 7. Energía asequible y no contaminante.			x	
ODS 8. Trabajo decente y crecimiento económico.	x			
ODS 9. Industria, innovación e infraestructuras.	x			
ODS 10. Reducción de las desigualdades.			x	
ODS 11. Ciudades y comunidades sostenibles.			x	
ODS 12. Producción y consumo responsables.			x	
ODS 13. Acción por el clima.			x	
ODS 14. Vida submarina.			x	
ODS 15. Vida de ecosistemas terrestres.			x	
ODS 16. Paz, justicia e instituciones sólidas.			x	
ODS 17. Alianzas para lograr objetivos.			x	

En el mundo del ajedrez donde cada movimiento resuena con estrategia y destreza, encontramos un paralelismo intrigante con el avance de las sociedades modernas hacia un futuro sostenible. El presente trabajo a través de una tecnología de reconocimiento de tablas y texto manuscrito se centra en el *ODS 8. Trabajo decente y crecimiento económico* y el *ODS 9. Industria, innovación e infraestructuras.*

Respecto al octavo objetivo, esta futura aplicación reduciría la cantidad de tiempo invertido en la transcripción de partidas. Eso haría que los trabajadores que se dedican a ello en los torneos tuvieran más tiempo para asegurar que los participantes del torneo se encuentren jugando bajo las mejores condiciones posibles, además de ser capaces de atender con mayor agilidad cualquier incidencia que pudiera surgir.

El noveno objetivo relacionado sería industria, innovación e infraestructuras. En general, la automatización de procesos manuales es una innovación pues mejora la eficiencia de una tarea y evita el error del factor humano. Mejorar la eficacia de un proceso puede promover el desarrollo de nuevas tecnologías en la industria del ajedrez y servir como motor económico.

En conclusión, este trabajo tiene una relación significativa con los objetivos ODS número ocho y nueve, debido a que puede contribuir al trabajo decente, el crecimiento económico sostenible, la innovación tecnológica y la mejora de la eficiencia en el entorno del ajedrez.