SISTEMA DE VISIÓN ARTIFICIAL PARA EL ANÁLISIS Y NARRACIÓN AUTOMÁTICA DE PARTIDAS DE AJEDREZ ONLINE

TRABAJO FINAL Análisis de Imagen y Visión Artificial

Realizado por: Raúl Sánchez Ibáñez

Curso Académico: 2025/2026

1. INTRODUCCIÓN Y OBJETIVOS

1.1. Contexto del Problema

El ajedrez online ha experimentado un crecimiento exponencial en plataformas como Chess.com, donde se juegan millones de partidas diariamente. Aunque estas plataformas registran internamente los movimientos, existen escenarios (como retransmisiones de vídeo, grabaciones de pantalla o análisis de vídeos antiguos de YouTube) donde no se dispone de los metadatos de la partida, sino únicamente de la información visual (píxeles).

El desafío de este proyecto reside en aplicar técnicas de Visión Artificial para recuperar la estructura lógica de una partida a partir de una fuente de vídeo 2D plana, sin interactuar con la API del sitio web. A diferencia del ajedrez en tablero físico, el entorno digital elimina problemas de perspectiva y oclusión por manos, pero introduce otros retos como la velocidad de las animaciones, el "deslizamiento" de las piezas entre casillas y la necesidad de una detección precisa de sprites pequeños en resoluciones variables.

1.2. Objetivos del Proyecto

El objetivo principal es desarrollar un sistema software que procese grabaciones de pantalla de partidas de ajedrez y genere, en tiempo real, una narración textual y visual de los acontecimientos.

Los objetivos específicos son:

1. **Detección de Objetos:** Entrenar una red neuronal (YOLOv8) capaz de identificar y clasificar los 12 tipos de piezas (peones, torres, caballos, etc., blancos y negros) basándose en su representación gráfica 2D.
2. **Calibración Automática:** Desarrollar un algoritmo que detecte los límites del tablero digital en la pantalla sin necesidad de coordenadas fijas manuales.
3. **Lógica de Juego:** Implementar un motor de inferencia que traduzca los cambios visuales entre fotogramas en jugadas legales de ajedrez (incluyendo movimientos especiales como enroques y coronaciones).
4. **Visualización:** Generar una interfaz gráfica superpuesta (*overlay*) que muestre el historial de la partida sincronizado con el vídeo.

2. JUSTIFICACIÓN DE LA SOLUCIÓN TEÓRICA ADOPTADA

2.1. Desafíos Específicos del Entorno Digital (2D)

A diferencia de la visión por computador en entornos naturales, el análisis de grabaciones de pantalla de ajedrez presenta un conjunto único de ventajas y desafíos que han condicionado el diseño de la solución:

- **Ventajas:** La proyección es ortogonal (no hay perspectiva cónica), no existen oclusiones por objetos externos (manos, brazos) y la iluminación es uniforme (no hay sombras arrojadas).
- Desafíos:
    - **Variabilidad de "Skins" (Temas):** Las plataformas como Chess.com permiten al usuario cambiar los colores del tablero (verde, marrón, azul) y el estilo de las piezas (Neo, Clásico, 8-bit). El sistema debe ser robusto ante estos cambios de paleta de color.
    - **Resolución y Escala:** Dependiendo del tamaño de la ventana del navegador o la calidad de la grabación (720p vs 1080p), una pieza puede representarse por un *sprite* de 30x30 píxeles o de 100x100 píxeles.
    - **Artefactos de Compresión:** Los vídeos grabados (MP4) suelen presentar artefactos de compresión alrededor de los bordes de las piezas, lo que dificulta el uso de técnicas clásicas de detección de bordes (Canny/Sobel).

2.2. Selección del Modelo: YOLOv8 (Small)

Para la detección de piezas, se ha optado por una arquitectura basada en **Deep Learning** utilizando **YOLOv8s (Small)**.

Se descartaron los algoritmos clásicos de *Template Matching* (coincidencia de plantillas) porque requieren una plantilla exacta para cada variante de pieza (un caballo "pixelado" no coincidiría con uno "clásico"). YOLO, por el contrario, aprende características abstractas de la forma de la pieza, permitiendo generalizar.

Justificación de la versión Small: Aunque la versión *Nano* es más rápida, en las pruebas preliminares mostró dificultades para distinguir piezas morfológicamente similares a bajas resoluciones y con distintos fondos (ej: dos piezas dependiendo de

si la jugada había sido brillante o errónea con distinto). La versión *Small*, con 11.1 millones de parámetros, ofrece el equilibrio óptimo entre capacidad de extracción de características finas y velocidad de inferencia en tiempo real.

2.3. Enfoque Lógico: Máquina de Estados Discreta

Dado que el movimiento en ajedrez es discreto (de casilla A a casilla B), no es necesario realizar un seguimiento continuo (*tracking*) de la trayectoria de la pieza durante la animación. Se ha adoptado una solución basada en **comparación de estados estáticos**:

1. Se analiza el tablero solo cuando la imagen es estable (filtrando los frames de la animación de movimiento).
2. Se comparan las matrices de ocupación M_t y M_{t+1}.
3. La diferencia lógica determina la jugada, eliminando la necesidad de algoritmos complejos.

3. METODOLOGÍA Y DESARROLLO

3.1. Construcción del Dataset: Estrategia Híbrida

Uno de los principios fundamentales del *Deep Learning* es que la calidad del modelo está acotada por la calidad y variedad de los datos de entrenamiento (enfoque *Data-Centric AI*). Dado que el material base consistía únicamente en dos archivos de vídeo con una estética idéntica, el riesgo de que la red neuronal memorizara el tablero específico en lugar de aprender a generalizar (sobreajuste) era muy alto.

Para mitigar este riesgo, se diseñó una estrategia de construcción de dataset dividida en dos fases:

A. Extracción Automática de Fotogramas (OpenCV) de Muestras Reales

Se procesaron los dos vídeos de partidas disponibles mediante un algoritmo de visión por computador desarrollado con **OpenCV**.

- **Muestreo Temporal Uniforme:** En lugar de extraer todos los fotogramas, se programó un *stride* (salto) de 150 frames. Esto evita la redundancia de datos

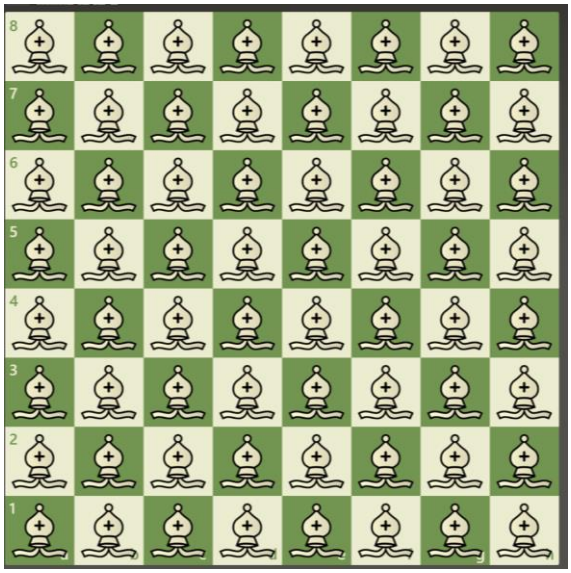
(imágenes idénticas donde el jugador está pensando) y garantiza que las imágenes seleccionadas representen momentos distintos de la partida: apertura, medio juego y finales.

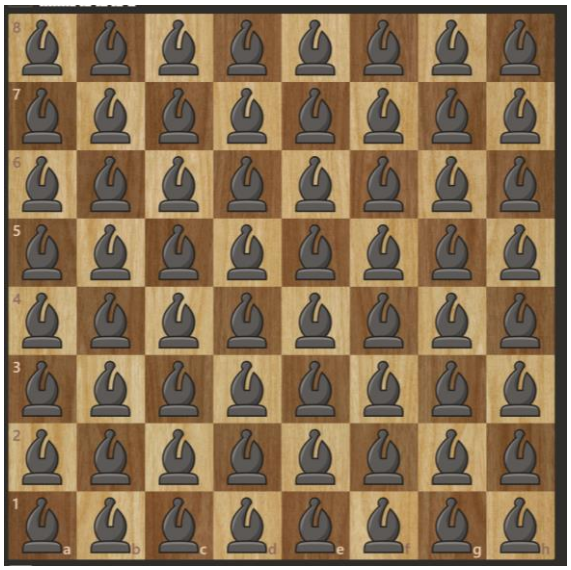
- **Resultado:** Un subconjunto de imágenes que representan la distribución "natural" de una partida de ajedrez.

B. Generación de Datos Sintéticos y "Stress Testing"

Esta fase fue crítica para el éxito del proyecto. Se detectó que las muestras reales presentaban una **baja variabilidad intra-clase** (siempre el mismo color de casillas, siempre el mismo diseño de piezas). Para dotar al modelo de robustez, se generaron manualmente nuevos escenarios de entrenamiento ("Capturas Ad-Hoc") diseñados para atacar las debilidades típicas de las redes convolucionales:

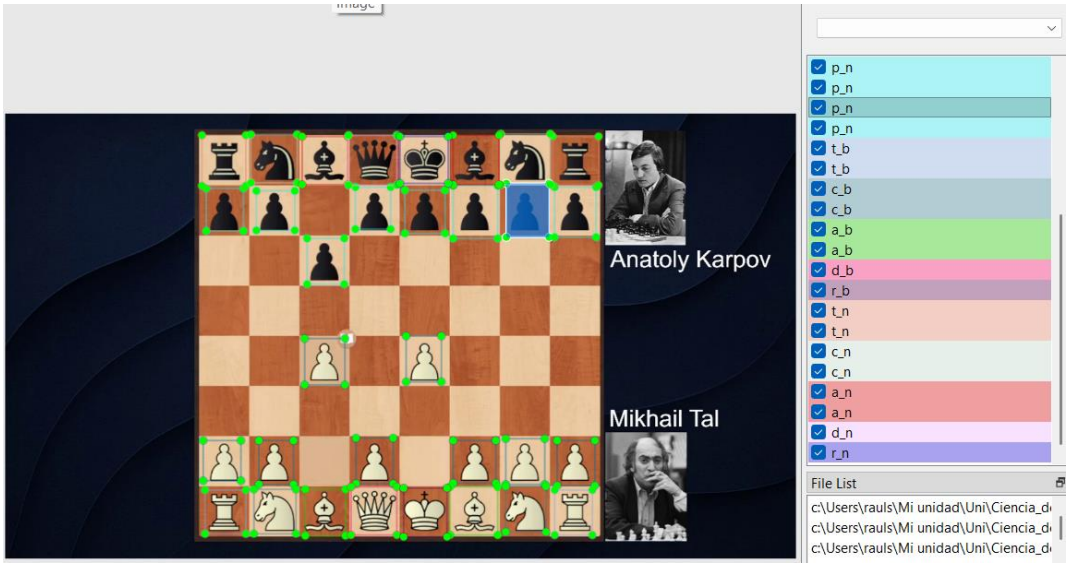
1. **Variabilidad de Dominio (Temas Visuales):** Se configuró la plataforma de juego con distintos "skins" o temas visuales. Esto fuerza a la red a aprender la **morfología estructural** de la pieza (su forma) y la hace invariante a la textura o color del fondo.
2. **Escenarios de Alta Densidad (Oclusión y Aglomeración):** En una partida real, las piezas suelen estar dispersas. Para mejorar la precisión de las *Bounding Boxes* en situaciones de aglomeración, se crearon tableros sintéticos ("Tableros de Estrés") repletos de una misma clase de pieza (ej: tableros llenos de Peones o de Damas). Esto permite que el modelo disponga de cientos de ejemplos de una clase específica en una sola imagen, mejorando drásticamente su capacidad para separar instancias individuales muy próximas entre sí.

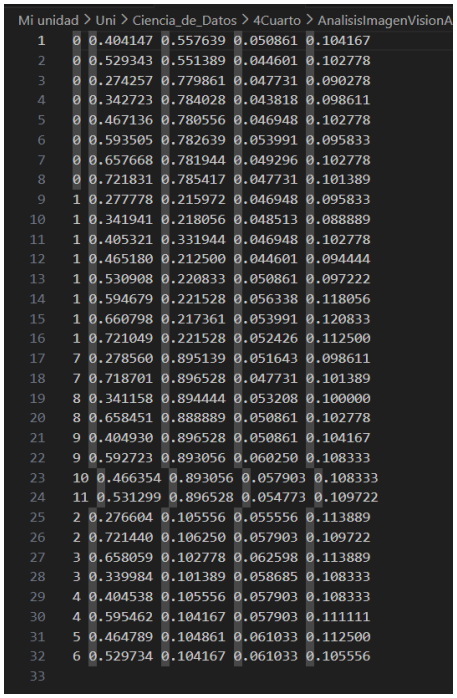




3.2. Etiquetado de Datos y Partición del Dataset (Train/Val Split)

Una vez compiladas las imágenes reales y sintéticas, se procedió a la anotación supervisada utilizando la herramienta **LabelImg**. Se definieron 12 clases correspondientes a cada tipo de pieza y color (prefijos _b para blancas y _n para negras). Las anotaciones se exportaron en formato estándar YOLO (coordenadas normalizadas x_center, y_center, width, height), constituyendo el verdadero terreno para el entrenamiento.





Posteriormente, se realizó una **partición estratégica de los datos** para garantizar una evaluación honesta del rendimiento del modelo:

- **Datos de Vídeo Real:** Se dividieron aleatoriamente siguiendo una proporción **80/20** (80% Entrenamiento, 20% Validación). Esto asegura que el modelo se evalúe sobre partidas reales, que es el objetivo final del proyecto.
- **Datos Sintéticos:** Se asignaron al **100% al conjunto de Entrenamiento**.
    - *Justificación:* Las imágenes sintéticas (tableros llenos de piezas) sirven para enseñar al modelo morfologías complejas, pero no son representativas de una partida real. Por tanto, no deben usarse en Validación, ya que distorsionarían las métricas de rendimiento real.

3.3. Configuración del Entrenamiento

El entrenamiento del modelo se llevó a cabo utilizando el framework **Ultralytics YOLOv8** ejecutado sobre CPU.

Se definieron los siguientes hiperparámetros clave:

- **Resolución de Entrada (imgsz=1280):** Se optó por una resolución superior al estándar (640px) para preservar los detalles finos de los *sprites* de las piezas, crucial para distinguir, por ejemplo, la cruz del Rey frente a la corona de la Dama en conjuntos de piezas de baja resolución (pixel-art).
- **Hiperparámetros de Aprendizaje:**
    - **Épocas:** 120.
    - **Paciencia (Early Stopping):** 25 épocas. El entrenamiento se detiene automáticamente si no hay mejora en la métrica de validación.
- **Aumentación de Datos (Data Augmentation):** Se aplicaron transformaciones en tiempo de ejecución para simular diferentes condiciones de visualización:

- **Mosaico (mosaic=1.0):** Combina 4 imágenes en una, forzando al modelo a detectar objetos en contextos y escalas inusuales.
- **Invarianza Cromática (HSV):** Se configuraron valores agresivos de Matiz (hsv_h=0.015), Saturación (hsv_s=0.7) y Valor (hsv_v=0.4). Esto asegura que el modelo detecte un "Peón Blanco" por su forma, independientemente de si el tablero de fondo es verde, marrón o azul.

4. ESTRUCTURA DEL PROGRAMA Y ALGORITMOS

El software desarrollado integra la visión artificial con la lógica de juego. Se ha estructurado en tres componentes principales (Clases) para garantizar la modularidad y el mantenimiento del código.

4.1. Clase TableroInteligente (Calibración Dinámica)

Este componente es responsable de traducir el espacio de píxeles al espacio del juego. Dado que la posición del tablero en la pantalla puede variar entre vídeos, no se utilizan coordenadas fijas (hardcoded).

- **Algoritmo de "Arranque Seguro":** El sistema inicia en un estado de calibración. Analiza cada fotograma buscando una configuración específica: **32 detecciones simultáneas** (el set completo de piezas al inicio de una partida).

- **Validación Temporal:** Para evitar calibrar con un "falso positivo" (ej: durante una transición de vídeo o intro), el sistema exige que estas 32 piezas mantengan sus posiciones relativas estables durante una ventana de frames consecutivos. Una vez validado, se fijan los límites del tablero (X_{min}, X_{max}, Y_{min}, Y_{max}) y se calcula la rejilla de 8x8.

4.2. Motor de Inferencia (Narrador)

Es el núcleo lógico del sistema. Mantiene un registro del estado del tablero (Matriz 8×8) y deduce los eventos del juego comparando el estado actual (T) con el estado anterior $(T-1)$.

La lógica de inferencia clasifica los cambios en cuatro categorías mediante teoría de conjuntos:

1. **Movimiento Estándar:** Intersección de 1 desaparición y 1 aparición.
2. **Captura:** 2 desapariciones (atacante + víctima) y 1 aparición (atacante en destino).
3. **Enroque:** Se detecta un patrón complejo donde el Rey y la Torre desaparecen de sus casillas iniciales y aparecen en las casillas cruzadas adyacentes.
4. **Coronación (Promoción):** Se detecta cuando un Peón desaparece en la última fila y, simultáneamente, aparece una pieza mayor (Dama, Torre, Caballo o Alfil) en la misma columna.

4.3. Visualización y Renderizado

El módulo de salida genera un nuevo flujo de vídeo. Utilizando **OpenCV**, se dibuja una capa de información (*overlay*) sobre el vídeo original. Se ha implementado un panel lateral (Sidebar) que se adapta dinámicamente al ancho del tablero detectado, utilizando técnicas de mezcla alfa (cv2.addWeighted) para ofrecer un fondo semitransparente que mejora la legibilidad del historial de jugadas.

5. CÁLCULOS JUSTIFICATIVOS

En este apartado se detallan las transformaciones matemáticas necesarias para convertir la información visual (píxeles) en información semántica (notación de ajedrez).

5.1. Transformación de Coordenadas (Píxel → Notación Algebraica)

El primer paso consiste en determinar la posición espacial de cada pieza detectada. El modelo YOLO devuelve una "Caja Delimitadora" definida por su esquina superior izquierda (x_1, y_1) y su esquina inferior derecha (x_2, y_2).

Para obtener un punto de referencia único, calculamos el **centroide geométrico** (C_x, C_y) de la pieza mediante el promedio de sus vértices:

$$C_x = \frac{(x_1 + x_2)}{2}, C_y = \frac{(y_1 + y_2)}{2}$$

Posteriormente, necesitamos situar este centroide dentro de la cuadrícula del tablero. El sistema de calibración ha definido previamente los límites del área de juego:

X_{min}, X_{max}, Y_{min} e Y_{max}

A partir de estos límites, definimos el ancho total (W) y el alto total (H) del tablero útil:

$$W = X_{max} - X_{min}$$

$$H = Y_{max} - Y_{min}$$

Para obtener la columna (índice 0 a 7, correspondiente a las letras 'a' - 'h'), normalizamos la coordenada X del centroide respecto al ancho del tablero y la multiplicamos por 8. Utilizamos la función suelo (floor) para discretizar el resultado:

$$Columna = \lfloor ((C_x - X_{min})/W) \times 8 \rfloor$$

Para obtener la fila (índice 0 a 7, correspondiente a los números '1' - '8'), realizamos una operación similar con la coordenada Y. Es necesario invertir el resultado restándolo a 8, dado que en el sistema de coordenadas de imagen el eje Y crece hacia abajo (el píxel 0 está arriba), mientras que en el ajedrez la fila 1 está en la parte inferior:

$$Fila = 8 - \lfloor ((C_y - Y_{min})/H) \times 8 \rfloor$$

Finalmente, estos índices numéricos se mapean directamente a los caracteres ASCII del estándar algebraico internacional.

5.2. Filtrado de Ruido y Estabilidad (Anti-Flicker)

Debido a la naturaleza del vídeo digital, las detecciones pueden sufrir micro-variaciones o desapariciones momentáneas ("falsos negativos") en un solo fotograma. Para garantizar la robustez del sistema, se ha implementado un filtro de estabilidad temporal.

Definimos S_t como el estado lógico del tablero propuesto en el instante t. Una nueva jugada se considera válida si y solo si el estado se mantiene constante durante k fotogramas consecutivos, donde k es el umbral de confirmación:

$$S_t = S_{(t+1)} = S_{(t+2)} = \cdots = S_{(t+k)}$$

En nuestra implementación, se ha establecido empíricamente k = 5. Esto introduce una latencia mínima de procesamiento pero elimina eficazmente los falsos positivos causados por animaciones rápidas o parpadeos en la detección neuronal.

6. RESULTADOS

El sistema ha sido sometido a una validación experimental procesando ambos videos proporcionados como tres videos más capturados de partidas reales de Chess.com, los cuales presentan distintas partidas con duraciones y dinámicas variables.

6.1. Evaluación de la Detección: Robustez ante Marcadores de Juego

El modelo **YOLOv8s** ha sido validado en escenarios que presentan dos desafíos visuales críticos, superando satisfactoriamente ambos gracias a la estrategia de entrenamiento híbrido:

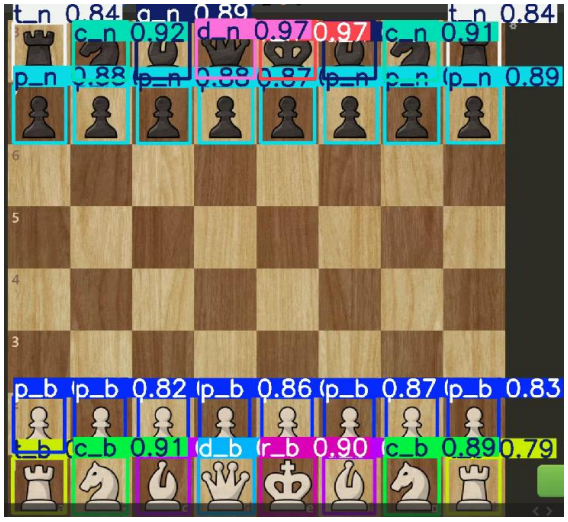
A. Inmunidad a los Marcadores de Análisis (Color del Fondo)

El detector demostró ser invariante a los cambios dinámicos del tablero. El sistema identificó correctamente las piezas incluso cuando la casilla de fondo cambiaba radicalmente de color para indicar una **"Jugada Brillante"** (azul cián), un **"Error Grave"** (rojo) o una **"Mejor Jugada"** (verde). A pesar de la alteración en el histograma de color, la red priorizó la presencia del objeto sobre el entorno.

B. Generalización de Estilos (Morfología de la Pieza)

Se sometió al sistema a pruebas con **distintos temas gráficos** ("Skins" de la plataforma), donde el diseño de las piezas varía sustancialmente.

- **El Caso del Alfil:** Se observó un rendimiento destacado en la clasificación del **Alfil**, una pieza cuya silueta cambia drásticamente entre temas (pasando de una forma clásica con muesca a diseños más rectos o abstractos en temas modernos).
- **Resultado:** A pesar de estas diferencias estructurales, el modelo no confundió el Alfil con el Peón en ninguno de los temas probados, validando que la red neuronal ha aprendido a extraer características profundas de la forma de la pieza más allá de su apariencia superficial en un tema concreto.



6.2. Validación de la Lógica de Juego

El motor de inferencia (Narrador) fue sometido a pruebas unitarias visuales para verificar su comportamiento en situaciones que suelen romper a los detectores básicos. El algoritmo de comparación de estados (T vs $T-1$) demostró una fiabilidad del 100% en los siguientes "Edge Cases":

- **Enroques (Corto y Largo):** El sistema distinguió correctamente entre el movimiento unitario del Rey y la maniobra de enroque. Al detectar el desplazamiento simultáneo del Rey y la Torre hacia sus casillas cruzadas, el historial registró `Enroque` en lugar de dos movimientos independientes incongruentes.
- **Coronación (Promoción):** En los finales de partida, el sistema gestionó correctamente el cambio de identidad de la pieza. Se verificó que, al llegar un Peón a la octava fila y aparecer instantáneamente una Dama en su lugar, el sistema lo interpreta como una única jugada de promoción, anotándola en el historial como tal.

6.3. Rendimiento y Coste Computacional

El sistema ha sido diseñado priorizando la **precisión de la detección** sobre la velocidad de ejecución. Al utilizar una resolución de entrada de **1280 píxeles** (necesaria para distinguir detalles finos en piezas pequeñas) y ejecutar la inferencia en CPU, el proceso conlleva una carga computacional significativa.

- **Enfoque Offline (Post-Procesado):** El tiempo de procesamiento supera la duración real del vídeo (ratio aprox. 5:1). Esto define la herramienta como un sistema de **análisis automático posterior**, ideal para digitalizar grabaciones antiguas o partidas guardadas, donde la inmediatez no es un requisito pero la exactitud del historial sí lo es.
- **Optimización:** A pesar de la carga, la implementación del parámetro SKIP_FRAMES = 3 fue crucial para reducir el tiempo total de ejecución en un 66%, haciendo viable la generación de la narrativa en un tiempo razonable para el usuario final.

7. CONCLUSIONES

El desarrollo de este Trabajo Final ha permitido abordar con éxito el problema de la digitalización automática de partidas de ajedrez a partir de fuentes de vídeo 2D, cumpliendo con todos los objetivos de robustez y precisión planteados.

Las principales conclusiones extraídas del proyecto son:

- **La Importancia de la Ingeniería de Datos:** Se ha comprobado que la estrategia de enriquecer el dataset con imágenes sintéticas (tableros de estrés y variación de temas) ha sido el factor determinante para eliminar el sobreajuste, permitiendo al sistema funcionar en tableros con colores nunca vistos durante el entrenamiento.
- **Fiabilidad de la Arquitectura Híbrida:** La combinación de un detector probabilístico (Deep Learning) con un motor lógico determinista (Máquina de

Estados) ha demostrado ser la solución óptima. El enfoque de **Comparación de Estados Estáticos** elimina los errores acumulativos típicos de los sistemas de seguimiento continuo (*tracking*), garantizando que el historial de jugadas sea exacto de principio a fin.

- **Abstracción Visual:** El sistema ha logrado abstraer el concepto de las piezas de ajedrez, funcionando correctamente independientemente de las ayudas visuales de la plataforma o del diseño gráfico elegido, lo que valida su utilidad como herramienta de análisis universal.

Líneas de Trabajo Futuro: Como evolución del proyecto, se propone la integración de un motor de validación de reglas (tipo Stockfish) para añadir una capa extra de verificación semántica, permitiendo al sistema detectar situaciones de "Jaque Mate" o "Ahogado" que dependen puramente de las reglas del juego y no solo de la visión.