



Universidad Nacional de Rosario  
Facultad de Ciencias Exactas, Ingeniería y Agrimensura  
T.U.I.A  
Computer Vision

### **Trabajo final:**

**“Detección de cartas españolas para cálculo del envido  
en el juego Truco “**

Santiago Ferrero

2024

# Introducción

El presente proyecto se enfoca en el desarrollo de un sistema de reconocimiento de cartas españolas utilizando técnicas avanzadas de visión por computadora, con el objetivo específico de calcular los puntos del envido en el popular juego de cartas argentino, Truco. Para lograr este objetivo, se llevó a cabo un proceso exhaustivo que incluyó la creación de un conjunto de datos específico, el entrenamiento de modelos de detección de objetos, y la evaluación del rendimiento de estos modelos en distintas etapas del desarrollo.

La solución entregada incluye una carpeta denominada `dataset`, que contiene las imágenes y etiquetas divididas en los conjuntos de entrenamiento y prueba.

Asimismo, se proporcionan cuatro notebooks en formato `.ipynb`:

`personalizacion_dataset_y_analisis_exploratorio`,  
`entrenamiento_dataset_no_aumentado`, `aumentacion_y_entrenamiento` y  
`00_run_envido`, cada uno de los cuales aborda aspectos específicos del desarrollo y evaluación del modelo.

Además, se incluye una carpeta llamada `00.alumnus_practico` que contiene un dataset en formato YOLO, diseñado para poner a prueba el modelo en un entorno controlado. Dentro de la carpeta `out`, se encuentran los archivos de salida generados por el modelo durante la ejecución del notebook `00_run_envido`. Estos archivos incluyen los resultados de la detección de cartas, las coordenadas de los bounding boxes (bbox) y los niveles de confianza asociados a cada predicción. Finalmente, los modelos entrenados están almacenados en la carpeta `model/weights`, listos para su implementación y uso en futuros desarrollos.

Entre estos modelos, se destacan los siguientes:

- `best (12).pt`: Corresponde al modelo entrenado con el dataset sin aumentar, el cual ha mostrado las mejores métricas en las evaluaciones.
- `last (10).pt`: Este es el modelo entrenado con el dataset aumentado.
- `last (10).engine`: Este archivo contiene el modelo entrenado con el dataset aumentado, optimizado mediante TensorRT.

El informe está dividido en secciones que detallan el rendimiento de cada modelo. La primera parte se enfoca en el análisis del modelo entrenado con el dataset sin aumentar. Posteriormente, se presentan los resultados obtenidos con el modelo aumentado.

# Proceso de Evaluación de Datos

## Creación de un Dataset Colaborativo

En la fase inicial del proyecto, se desarrolló un dataset colaborativo en el que cada estudiante del curso participó activamente, contribuyendo con al menos 30 imágenes de cartas españolas, anotadas en formato YOLO para su detección. En mi caso particular, capturé y anoté un total de 40 imágenes utilizando la herramienta Labelme, asegurándome de que cada imagen estuviera etiquetada con la mayor precisión posible y siguiendo los estándares establecidos para el proyecto. Para convertir estas anotaciones al formato requerido, utilicé la librería [labelme2yolo](#), generando los archivos `.txt` correspondientes que contenían las coordenadas de los bounding boxes (bbox) y las clases de las cartas.

Estos archivos se almacenan en un repositorio Drive, y se envían a los profesores para que los revisen. Todos los estudiantes del curso, un total de 44, realizaron un proceso similar, y los enlaces a cada uno de los datasets fueron recopilados en un documento compartido que puede verse en el siguiente link [+ Drives VC\\_TUIA\\_2024 \(Respuestas\)](#).

Este enfoque colaborativo fue fundamental para asegurar una amplia diversidad de imágenes y escenarios en los que se presentan las cartas, enriqueciendo así el conjunto de datos final.

## Generación de un Dataset Personalizado

Con el objetivo de consolidar un dataset que incluyera los aportes de todos los estudiantes, decidí descargar cada una de las carpetas colaborativas a mi computadora local. Este proceso resultó ser laborioso y consumió una cantidad considerable de tiempo, dado que requería la gestión manual de los archivos y su posterior subida a mi cuenta de Google Drive. Una vez completada la transferencia, puede accederse a estos fácilmente desde Google Colab, a través de la librería `google.colab`, realizando `drive.mount()`.

Antes de proceder con el entrenamiento, realicé una verificación exhaustiva para asegurarme de que el número de imágenes coincidiera exactamente con el número de etiquetas ([labels](#)), y que ambas coincidencias tuvieran nombres idénticos. Este proceso no estuvo exento de dificultades, ya que durante la descarga y carga de archivos, algunos de ellos se perdieron debido a limitaciones de memoria u otros fallos técnicos. Para minimizar la pérdida de datos, me dediqué a corregir manualmente los nombres de archivos que no coincidían entre las imágenes y las etiquetas, así como a buscar y agregar aquellos archivos faltantes.

Además, detecté que algunos estudiantes habían anotado sus imágenes en formatos distintos, como YOLOv8 y YOLOv5. Para garantizar la consistencia en el entrenamiento del modelo, desarrollé un script que convierte aquellas que originalmente contenían vértices con 8 valores al formato de 5 valores (xywh), y validaba las anotaciones, asegurando que los bounding boxes no excedieran los límites de las imágenes. Esta unificación fue crucial para asegurar que el

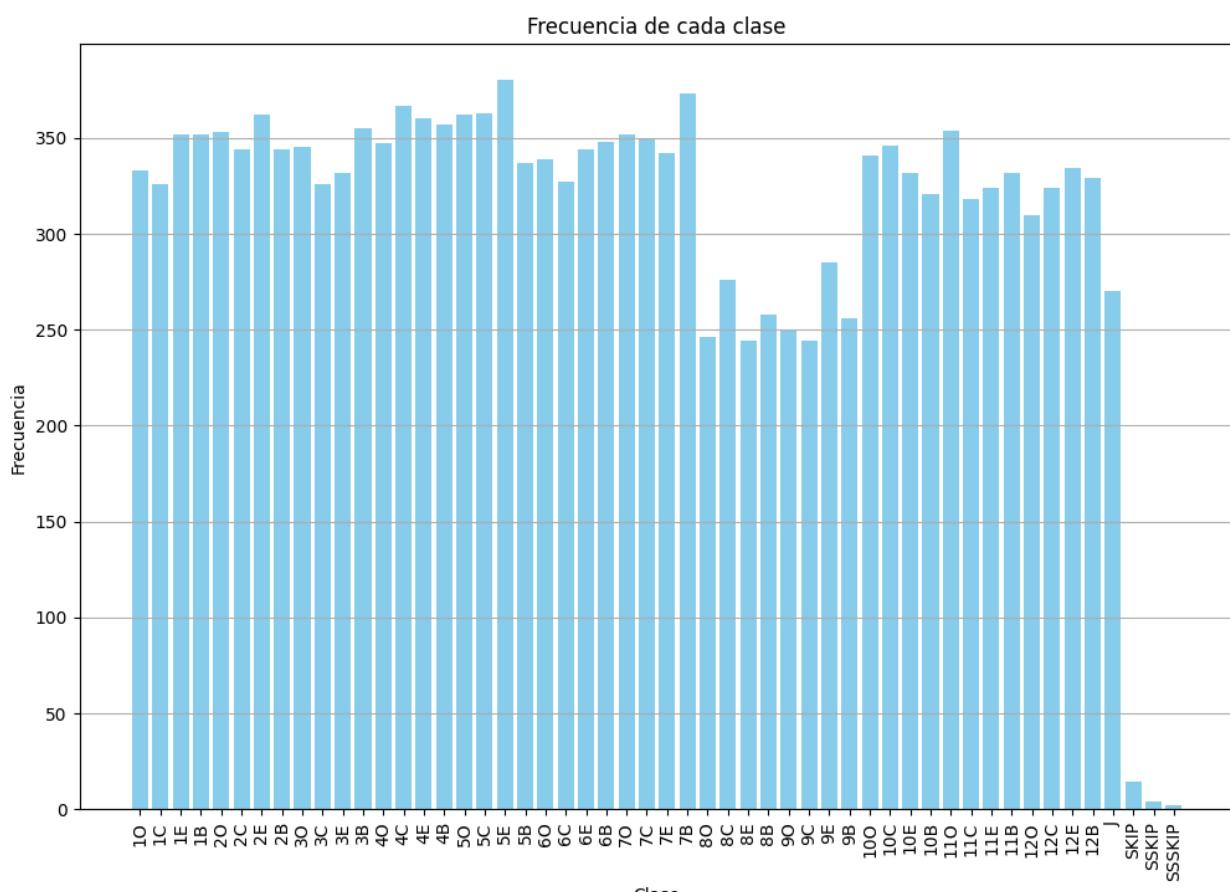
entrenamiento se realizará sin problemas y que el modelo pudiera procesar correctamente todas las imágenes.

Con los archivos `.txt` ya estandarizados, procedí a dividir el dataset en dos subconjuntos: `train` y `test`. Utilicé un 20% de los archivos para el conjunto de prueba (`test`), que resultó en 498 imágenes y 498 archivos `.txt`, y el 80% restante para el conjunto de entrenamiento (`train`), con un total de 1335 imágenes y 1335 archivos `.txt`. Esta división fue realizada de manera aleatoria para garantizar que ambos conjuntos fueran representativos y equilibrados.

## Análisis exploratorio

Antes de proceder con el entrenamiento del modelo, realicé un análisis exploratorio exhaustivo del dataset para comprender mejor la distribución de las clases y asegurarme de que las anotaciones de los bounding boxes fueran precisas tras el proceso de unificación de formatos.

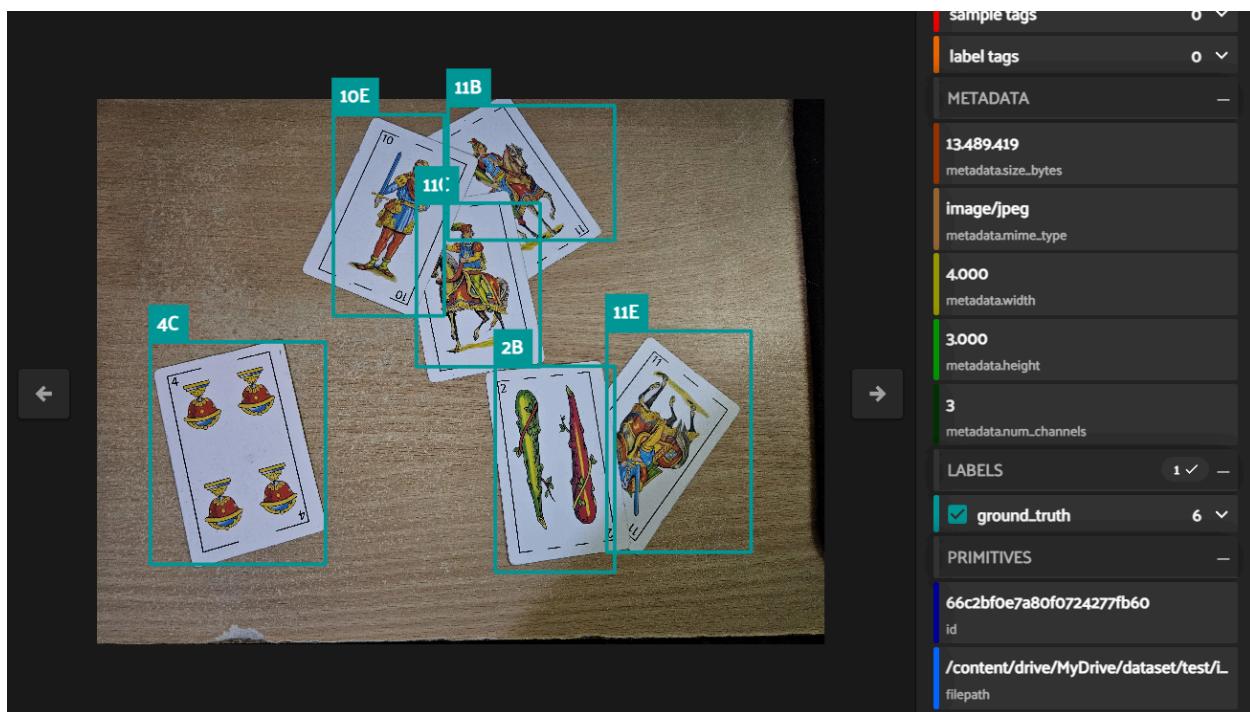
Primero, examiné la distribución de las clases de cartas dentro del dataset. Como era de esperarse, observé una menor cantidad de imágenes correspondientes a los números 8 y 9, ya que estas cartas no se utilizan en el juego de Truco. De manera similar, las cartas con la etiqueta "J" representan comodines, que tampoco se usan en Truco, y los valores etiquetados como "SKIP" al final resultaron ser errores de anotación. Aparte de estas excepciones, las demás clases de cartas mostraron cantidades relativamente uniformes, con un rango que oscila entre 320 y 370 imágenes por clase.



Durante la inspección, identifiqué un total de 22 archivos .txt con errores en las anotaciones. A pesar de estos errores, decidí no eliminarlos del conjunto de datos. La razón detrás de esta decisión fue que, durante el entrenamiento, el modelo YOLO ignora automáticamente las anotaciones que no cumplen con los criterios, lo que minimiza el impacto negativo de estas inconsistencias en el rendimiento final del modelo.

Otro aspecto importante del análisis fue evaluar el tamaño de las imágenes. Esto es crucial para decidir la resolución a la que se deben redimensionar las imágenes durante el entrenamiento del modelo YOLO. Revisé las dimensiones de todas las imágenes, prestando especial atención a las medidas mínima, máxima y promedio para descartar que todas sean muy chicas o muy grandes. Los resultados mostraron una amplia diversidad en los tamaños de las imágenes, lo que puede complicar el entrenamiento si no se estandariza la resolución.

Para verificar la calidad del dataset después de estos ajustes, utilicé la herramienta FiftyOne, que permite una visualización detallada de las imágenes junto con sus respectivos bounding boxes. Esta herramienta resultó invaluable para confirmar que las correcciones y unificaciones realizadas previamente eran adecuadas. Al revisar las imágenes del conjunto de prueba (test), pude comprobar que los bounding boxes estaban correctamente alineados con las cartas, lo que indica que el dataset estaba listo para ser utilizado en el entrenamiento del modelo.



# Entrenamiento del Modelo sin aumentar

Decidí entrenar el modelo utilizando exclusivamente los datos disponibles, sin aplicar técnicas de aumento de datos. Esto me permitió evaluar el rendimiento del modelo YOLOv8 de Ultralytics en su forma más básica, proporcionando una línea base sobre la cual se podrían aplicar mejoras en el futuro.

El entrenamiento se llevó a cabo empleando el modelo YOLOv8, reconocido por su alta eficiencia en tareas de detección de objetos. Para este propósito, configuré un entorno de entrenamiento en Google Colab. Preparé un archivo de configuración en formato YAML, donde especificué los parámetros clave del modelo, como las rutas a los datasets, las clases a detectar, y las configuraciones específicas de la red neuronal, como el tamaño de la imagen y el número de épocas.

Para garantizar un entrenamiento efectivo, opté por utilizar una GPU de alta capacidad en Colab, accediendo a este recurso a través del pago de unidades de procesamiento. Esta inversión me permitió llevar a cabo un entrenamiento inicial de 50 épocas, seguido de un reentrenamiento de 20 épocas adicionales. Este reentrenamiento se realizó con el objetivo de afinar aún más los pesos del modelo y mejorar su rendimiento en el conjunto de validación.

## Evaluación del Rendimiento del Modelo

Los resultados obtenidos durante el proceso de validación fueron los siguientes:

- **mAP50:** 0.806
- **mAP50-95:** 0.647
- **Precisión:** 0.852
- **Recall:** 0.711
- **F1-score:** 0.775

Estas métricas nos proporcionan una visión integral del desempeño del modelo:

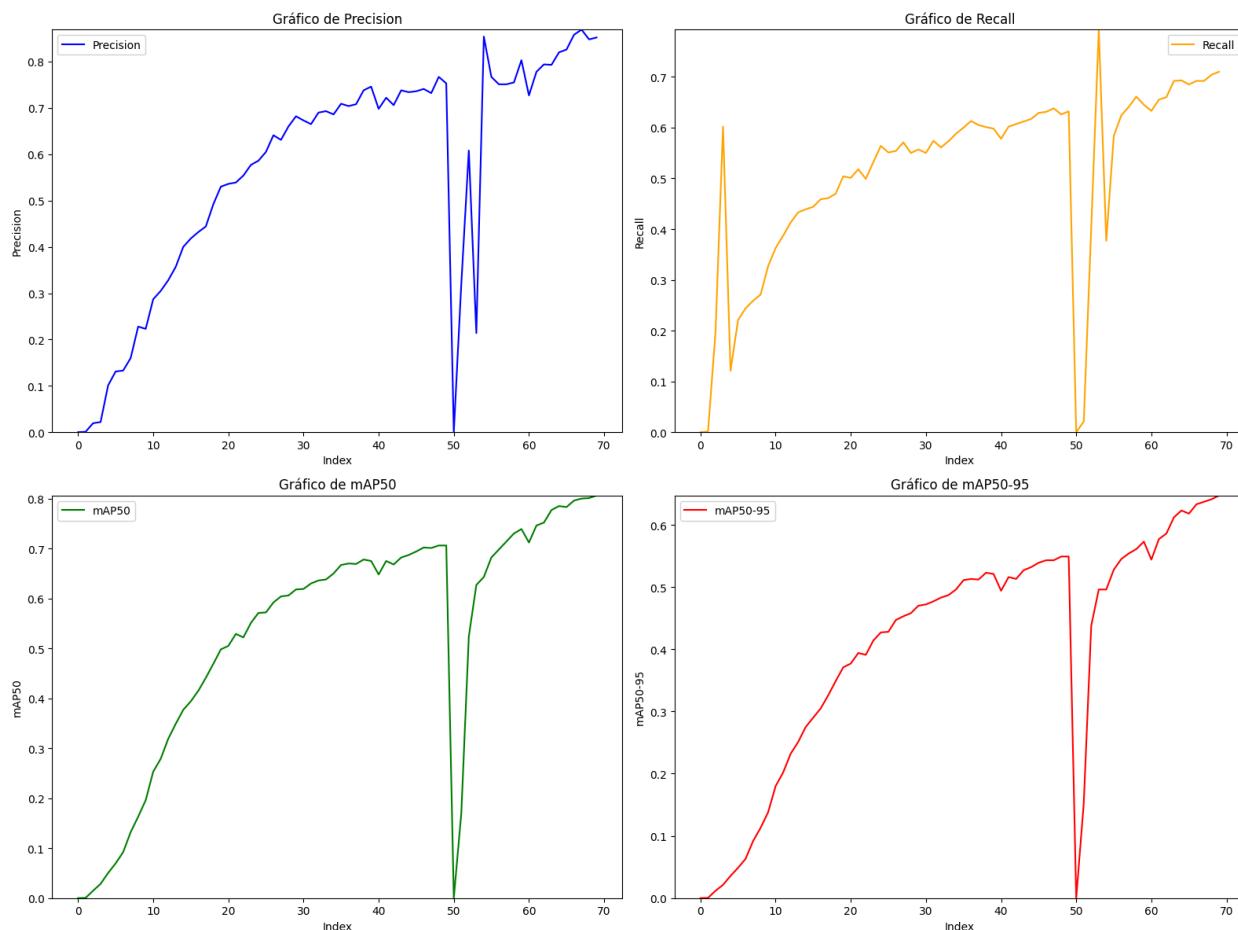
- **mAP50 (Mean Average Precision @ 50%):** Un valor de 0.806 indica que el modelo tiene un alto rendimiento en la detección de objetos cuando se utiliza un umbral de IoU (Intersection over Union) del 50%. Esto significa que el modelo es capaz de localizar correctamente la mayoría de las cartas con una precisión considerable.
- **mAP50-95:** Este valor, que promedia la precisión para distintos umbrales de IoU (entre 50% y 95%), es de 0.647. Aunque es inferior al mAP50, esto es esperado, ya que la métrica es más exigente al evaluar el rendimiento del modelo en distintos niveles de precisión.

- **Precisión:** Con un valor de 0.852, esta métrica indica que el modelo comete pocos errores al clasificar correctamente las cartas que detecta, es decir, tiene una alta tasa de aciertos en las detecciones positivas.
- **Recall:** El recall de 0.711 sugiere que el modelo es capaz de detectar la mayoría de las cartas presentes en las imágenes, aunque todavía hay un margen de mejora para capturar todos los objetos relevantes.
- **F1-score:** El F1-score de 0.775, que es una combinación de precisión y recall, proporciona un balance entre ambas métricas, mostrando que el modelo logra un buen equilibrio entre detectar correctamente las cartas y minimizar los falsos positivos.

En resumen, estas métricas reflejan un rendimiento sólido del modelo en su estado actual, sin aumentaciones de datos. Sin embargo, los resultados también sugieren que hay espacio para mejorar, especialmente en términos de recall y mAP50-95, donde la aplicación de técnicas adicionales, como la aumento de datos o el ajuste de hiperparámetros, podría contribuir a un desempeño aún mejor.

## Análisis de Métricas a lo Largo del Entrenamiento

Para evaluar la evolución de las métricas a lo largo de las épocas, se recolectaron datos del log de información del entrenamiento para crear gráficos. Desafortunadamente, no se guardaron las imágenes generadas automáticamente durante el entrenamiento, que incluían gráficos clave como las funciones de pérdida tanto para el entrenamiento como para la validación. Estos gráficos son útiles para determinar si el modelo presenta **overfitting**.

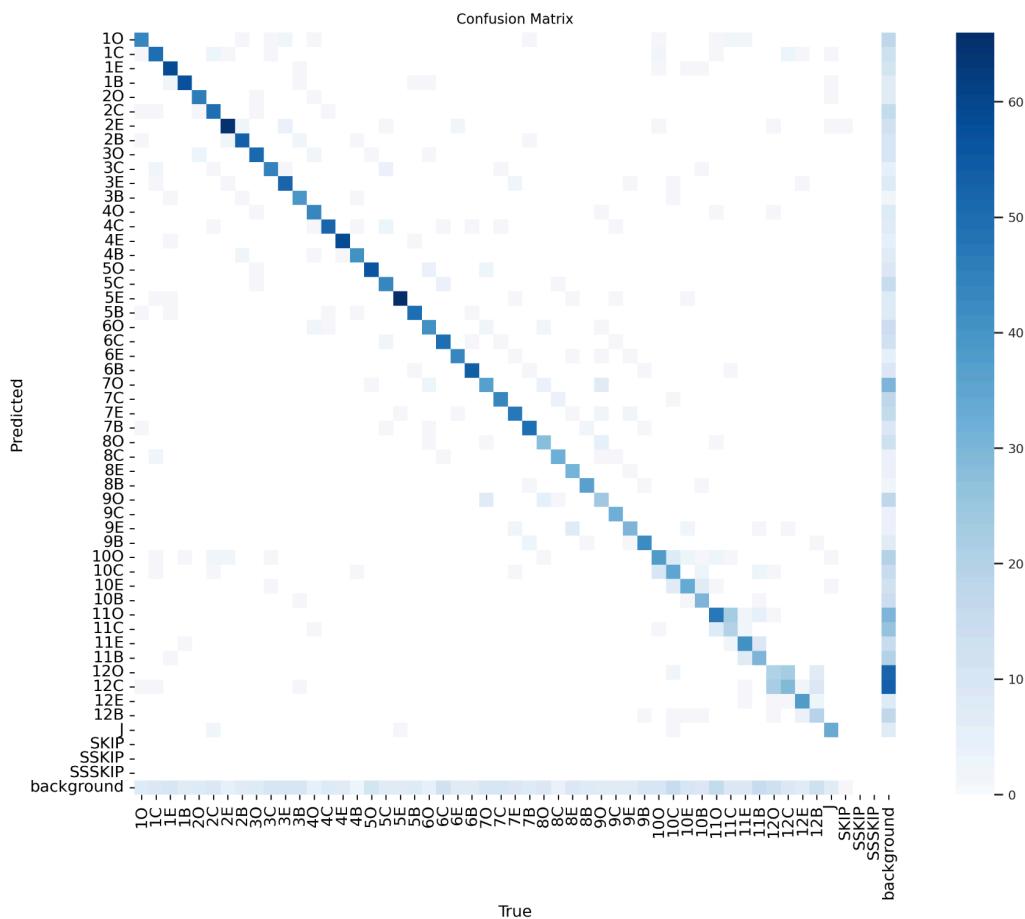


Se observa que todas las métricas mejoran con el transcurso de las épocas. En el reentrenamiento, la primera época muestra resultados casi nulos, pero rápidamente recupera los valores anteriores gracias a los pesos guardados previamente.

## Imágenes generadas en la evaluación del Modelo

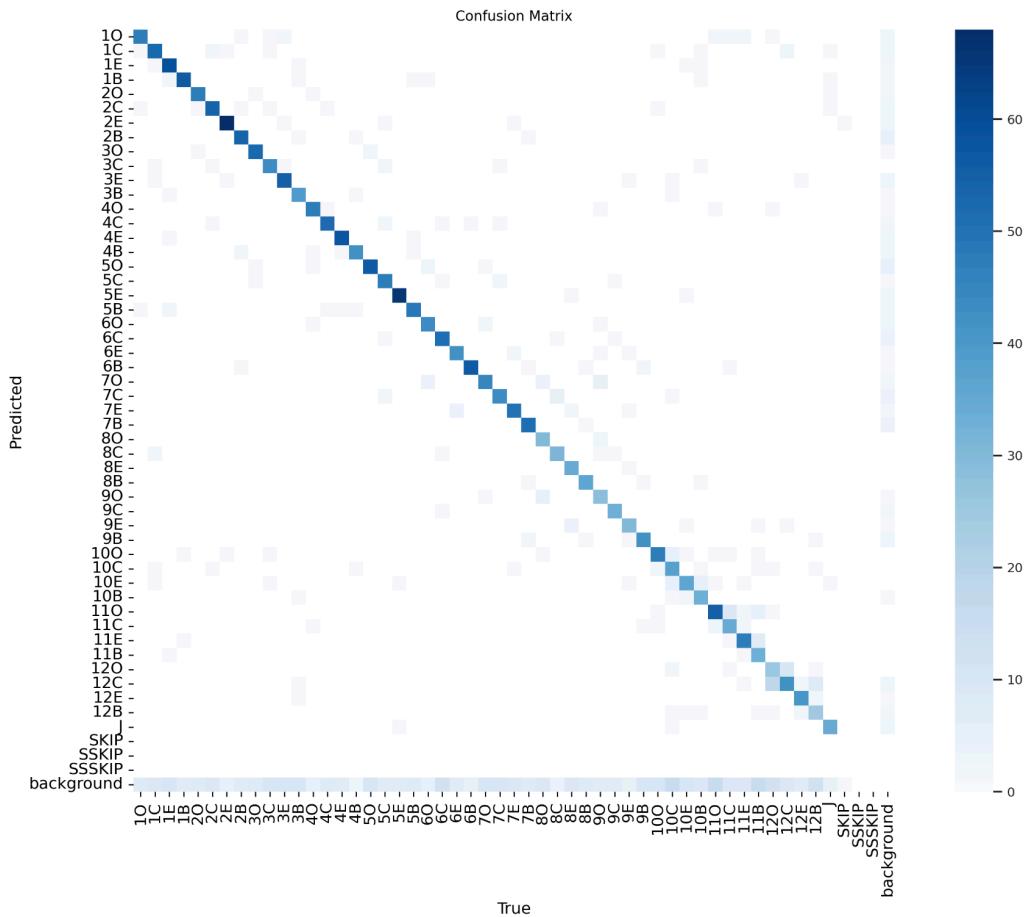
Una vez terminado el entrenamiento, se evalúa el modelo con el dataset de validación, y se generan automáticamente algunas imágenes que permiten ver cómo se desempeña el modelo.

La matriz de confusión presentada refleja el desempeño del modelo en la clasificación de las cartas españolas para el juego de Truco. En este caso, cada eje representa las clases de cartas, con las clases verdaderas en el eje horizontal (True) y las clases predichas en el eje vertical (Predicted).



La mayoría de las cartas están bien predichas, como lo indican las celdas más oscuras en la diagonal principal, que representan un mayor número de predicciones correctas para esas clases específicas. Sin embargo, también se observa una cantidad significativa de cartas

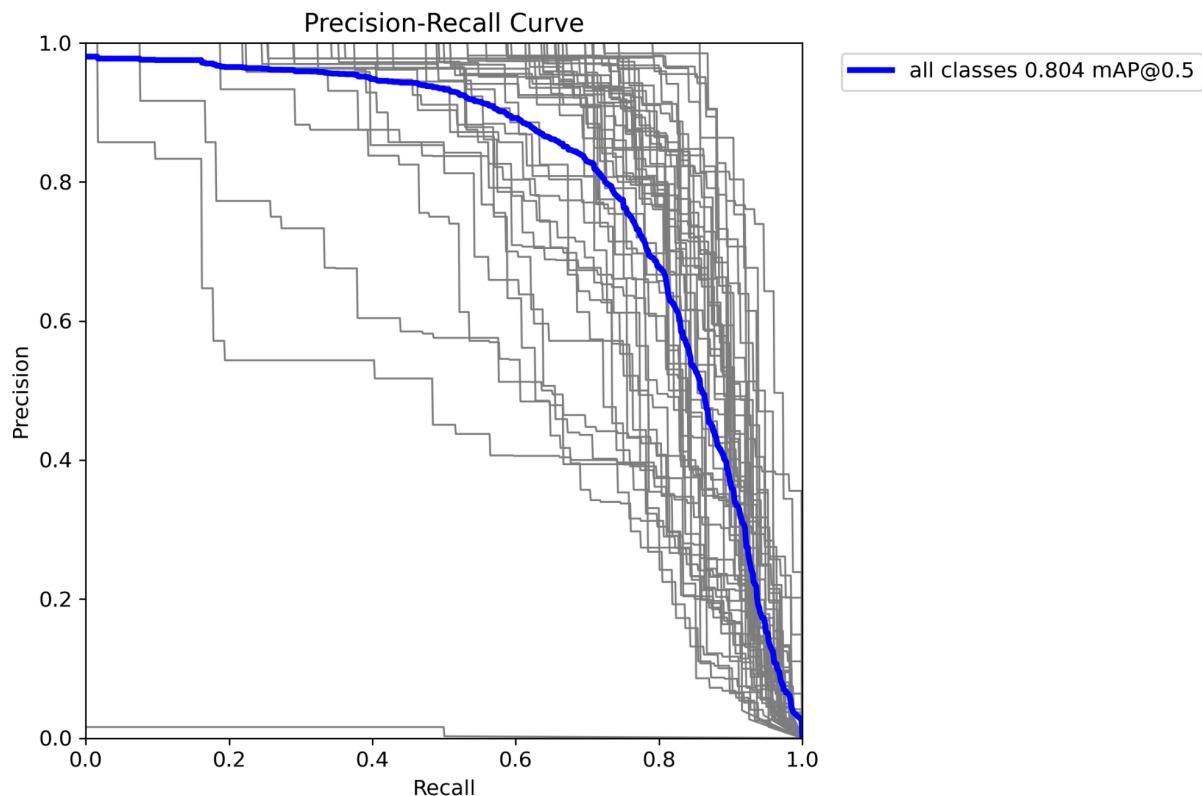
clasificadas incorrectamente como "background", lo que sugiere que hay varias cartas que no se detectan adecuadamente.



Esta matriz es de la evaluación usando el parámetro `agnostic_nms=True`, vemos como mejora notablemente la cantidad de cartas predicha como background, lo que quiere decir que muchas cartas que no se identificaban ahora si lo hacen correctamente.

Este parámetro indica que el proceso de NMS (supresión de máximos no máximos sin considerar la clase) no discrimina entre clases distintas cuando decide cuáles detecciones mantener y cuáles suprimir. Esto es especialmente útil en casos donde diferentes clases de cartas están muy juntas en la imagen, ya que permite que el modelo mantenga múltiples detecciones en lugar de suprimirlas, incluso si pertenecen a diferentes clases.

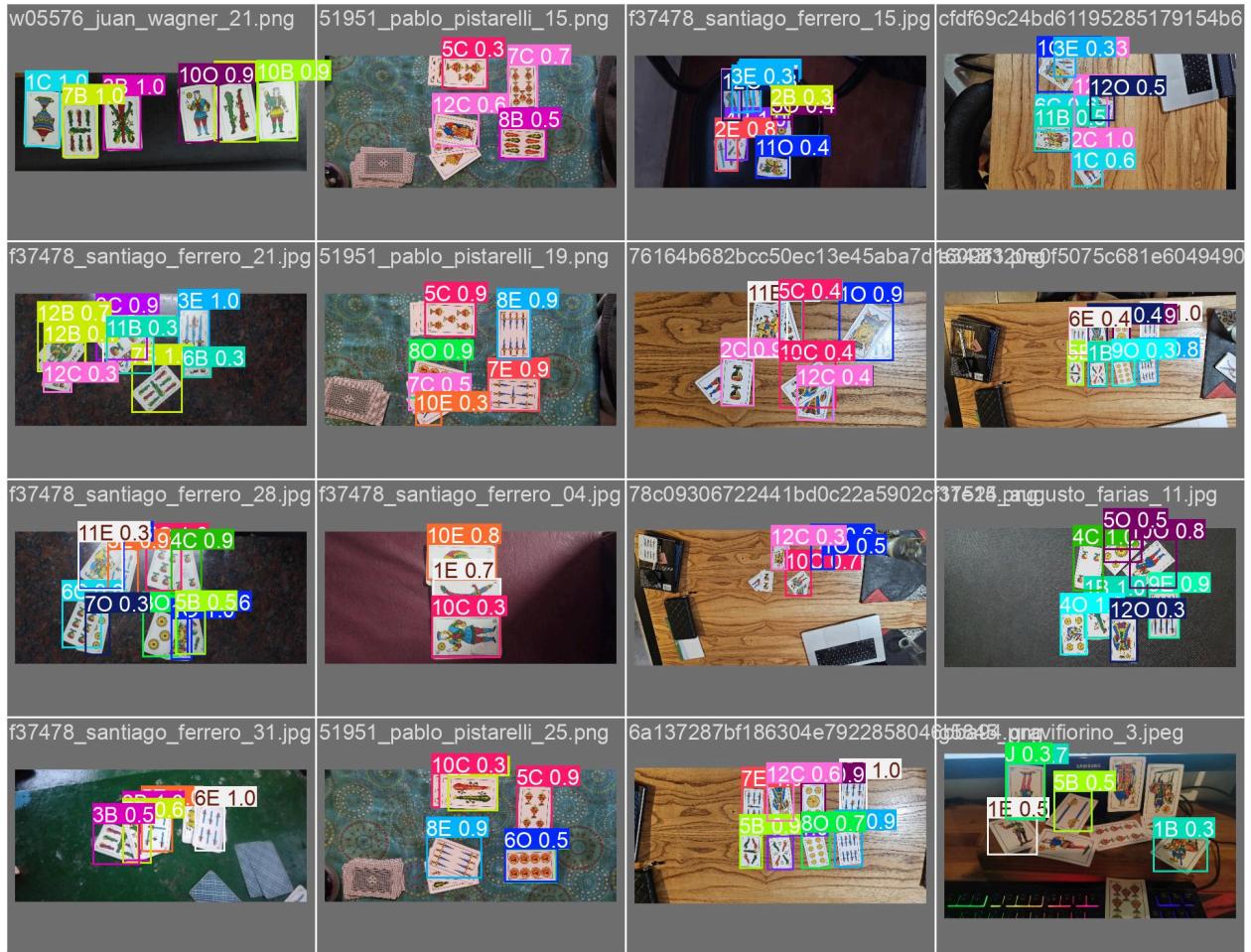
La curva Precision-Recall proporciona información clave sobre el rendimiento del modelo de detección de objetos, especialmente en contextos donde las clases están desbalanceadas o el interés está más centrado en los positivos.



La curva azul muestra la relación entre precisión y recall para todas las clases del modelo. La curva se mantiene relativamente alta en la parte superior izquierda, lo que indica que el modelo es capaz de mantener una alta precisión incluso a niveles altos de recall.

Sin embargo, al aumentar el recall hacia 1, la precisión disminuye. Esto es típico, ya que un alto recall implica identificar más verdaderos positivos, pero también puede aumentar el número de falsos positivos, reduciendo así la precisión.

Se presentaron imágenes con las detecciones realizadas por el modelo para realizar una evaluación visual de su desempeño, observando la precisión y fiabilidad en la detección de las cartas.



Muchas de las cartas se reconocen correctamente, pero también tenemos algunas cartas con más oclusión que no se llegan a detectar, y con algunas cartas muy similares puede confundirse la etiqueta, como con el 5 y el 3 de copa.

## Generación de un Dataset con Datos Aumentados

Con el objetivo de mejorar la robustez del modelo, se creó un nuevo dataset utilizando técnicas de aumentación de datos, empleando la biblioteca Albumentations. La aumentación de datos consiste en aplicar transformaciones a las imágenes originales para crear nuevas versiones que simulan variaciones posibles en el mundo real, como cambios en el color, brillo, contraste, o rotaciones. Estas variaciones permiten que el modelo se entrene en un conjunto de datos más diverso, mejorando su capacidad de generalización.

## Funciones Utilizadas en la Aumentación de Datos

Las funciones seleccionadas para la aumentación de datos en este proyecto fueron las siguientes:

- `HorizontalFlip(p=0.5)`: Esta función realiza una inversión horizontal de la imagen con una probabilidad del 50%.
- `RandomBrightnessContrast(p=0.3)`: Ajusta aleatoriamente el brillo y el contraste de la imagen con una probabilidad del 30%.
- `RGBShift(r_shift_limit=30, g_shift_limit=30, b_shift_limit=30, p=0.3)`: Aplica un desplazamiento aleatorio a los canales de color (Rojo, Verde, Azul) con un límite de 30 unidades, y una probabilidad del 30%.
- `OneOf([MotionBlur(p=0.5), Sharpen(), Emboss()], p=0.5)`: Esta función aplica una de las tres transformaciones posibles (`MotionBlur`, `Sharpen`, `Emboss`) con una probabilidad combinada del 50%. Estas transformaciones imitan situaciones en las que la imagen puede estar desenfocada, resaltada o tener un relieve, lo que podría ocurrir si la carta se movió mientras se tomaba la foto.

Estas funciones fueron aplicadas únicamente al conjunto de entrenamiento, generando una nueva imagen aumentada por cada imagen original, duplicando efectivamente el tamaño del dataset de entrenamiento. Las nuevas imágenes aumentadas fueron etiquetadas con un prefijo “`aug_`” para diferenciarlas de las originales.

Además, durante la aplicación de estas transformaciones, se actualizan los bounding boxes (`bboxes`) asociados a cada carta, asegurando que sigan cubriendo correctamente las cartas en las imágenes modificadas.

## Evaluación del Rendimiento del Modelo Aumentado

Para verificar que la aumentación no afecta negativamente los `bboxes`, se creó una carpeta donde se almacenaron imágenes aleatorias junto a sus respectivos `bboxes`, los cuales fueron graficados y revisados manualmente.

Este tipo de verificación asegura que las aumentaciones aplicadas mantienen la integridad de las etiquetas y no afectan la calidad del entrenamiento.



Imagen aumentada con los bounding boxes correctos.

## Resultados del Entrenamiento con Datos Aumentados

Se entrenó nuevamente un modelo YOLOv8, esta vez utilizando un dataset aumentado. La aumentación de datos, que incluye técnicas como rotaciones, escalados y ajustes de brillo o contraste, se empleó para mejorar la capacidad del modelo de generalizar a nuevas variaciones de los datos. Sin embargo, debido a las limitaciones de recursos computacionales, el entrenamiento se limitó a 15 épocas iniciales, seguidas de 4 épocas adicionales de reentrenamiento.

Las métricas obtenidas fueron las siguientes:

- **mAP50:** 0.573
- **mAP50-95:** 0.500
- **Precisión:** 0.623
- **Recall:** 0.547
- **F1-score:** 0.558

A primera vista, estas métricas son inferiores a las obtenidas en el modelo entrenado sin aumentación de datos. Sin embargo, al comparar estas métricas con las del modelo anterior en la misma época 15, se observa que el modelo entrenado con aumentación presenta mejores

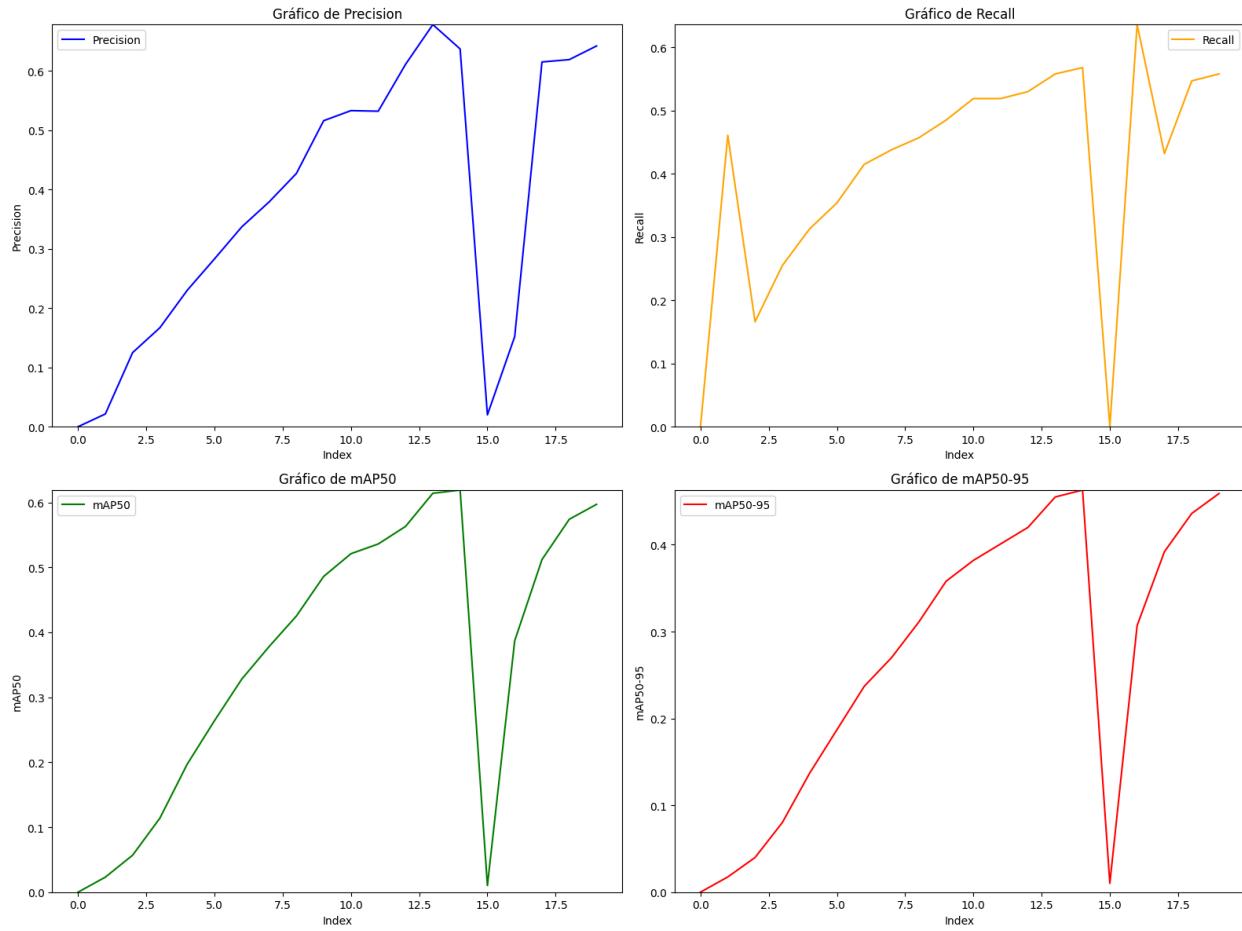
resultados. Esto sugiere que el uso de la aumentación de datos permite que el modelo aprenda más rápido, incluso con un número limitado de épocas.

- **mAP50 (Mean Average Precision @ 50%):** Un valor de 0.573 a la época 15, aunque menor que el obtenido en el entrenamiento completo sin aumentación, es superior en comparación directa con la misma época en el modelo anterior. Esto indica que la aumentación ha permitido al modelo captar de manera más efectiva las características de las cartas en un período más corto.
- **mAP50-95:** El valor de 0.500 refleja una mejora en la precisión promedio para distintos umbrales de IoU en comparación con la misma época del modelo no aumentado, sugiriendo que el modelo se adapta mejor a variaciones más complejas gracias a la aumentación.
- **Precisión y Recall:** La precisión de 0.623 y el recall de 0.547 son mejores a las métricas equivalentes del modelo anterior en las primeras etapas del entrenamiento, lo que indica que el uso de datos aumentados ayuda al modelo a identificar y clasificar correctamente las cartas más rápidamente.
- **F1-score:** El F1-score de 0.558, que equilibra precisión y recall, confirma que el modelo entrenado con datos aumentados logra un mejor equilibrio en las primeras etapas del entrenamiento en comparación con el modelo anterior.

En resumen, aunque las métricas finales del modelo entrenado con datos aumentados son inferiores debido al menor número de épocas, la comparación con el modelo anterior en la misma etapa del entrenamiento revela que la aumentación de datos es beneficiosa. El modelo con datos aumentados aprende más rápido, lo que sugiere que, con más épocas de entrenamiento, podría superar al modelo entrenado sin aumentación.

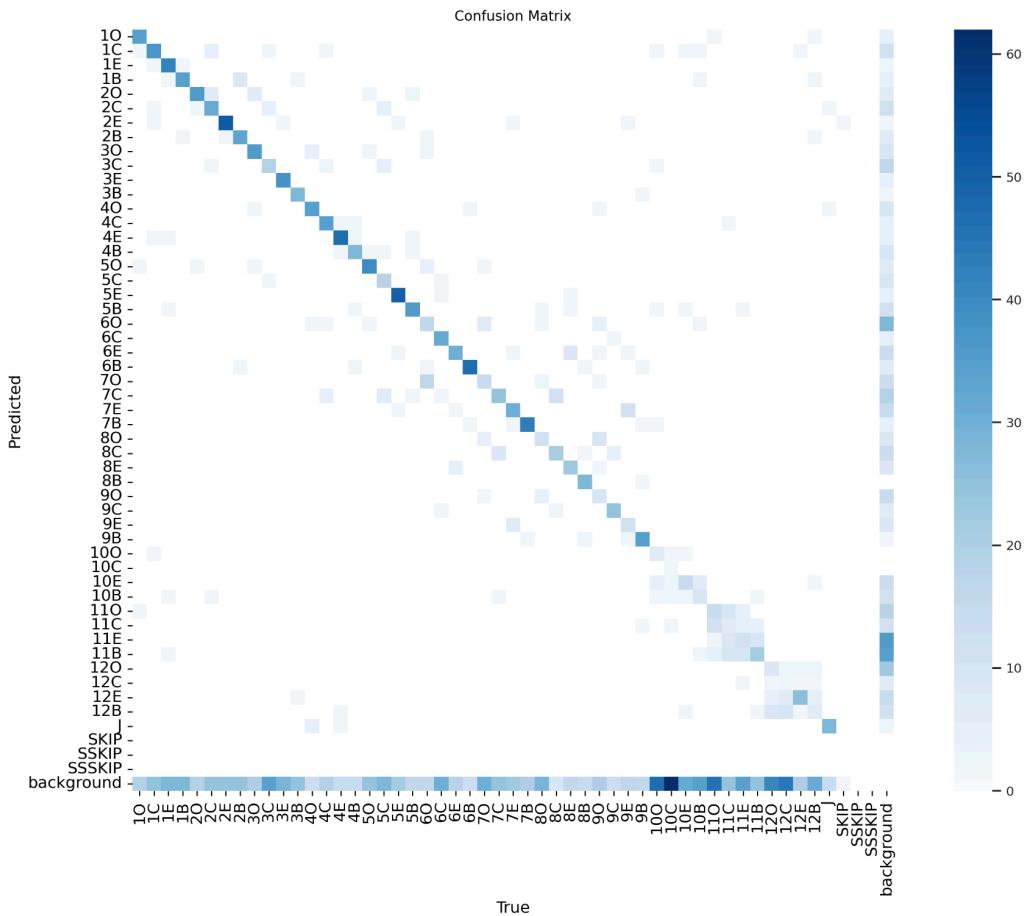
## Imágenes generadas en la evaluación del Modelo Aumentado

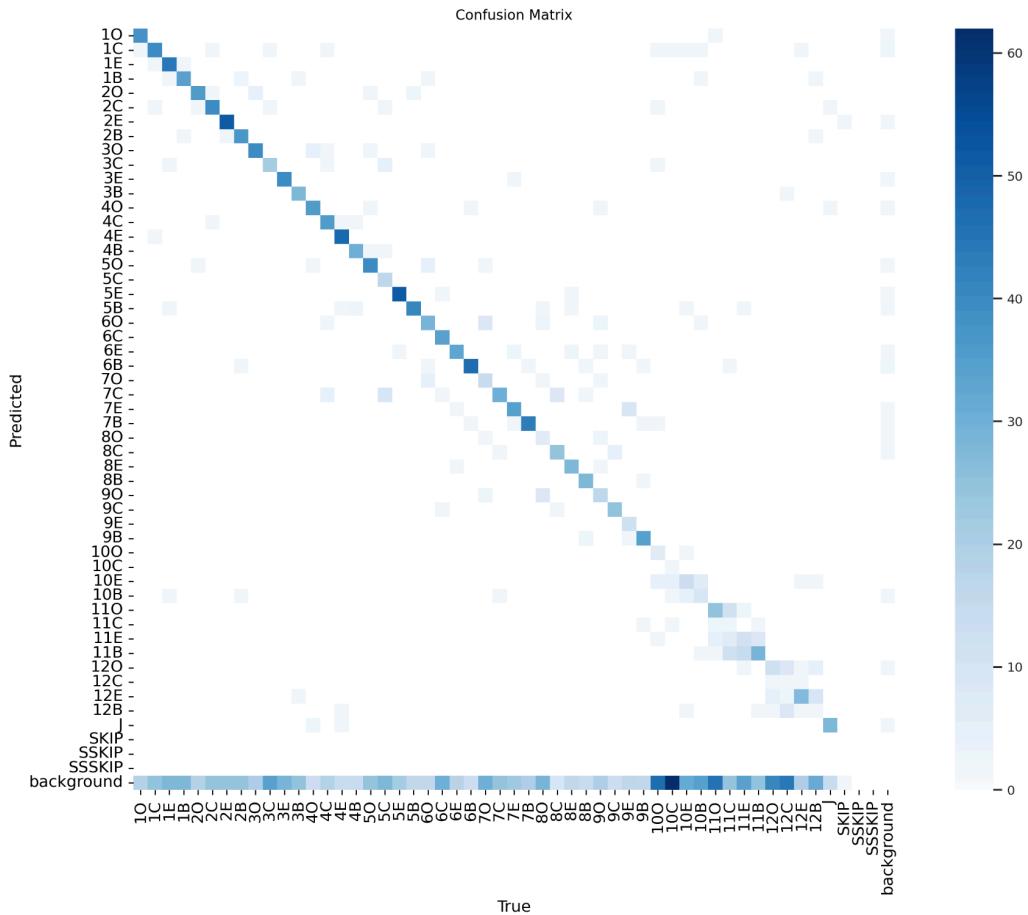
Debido a que tampoco se guardaron las imágenes generadas en el entrenamiento, para analizar la evolución de las métricas a lo largo de las épocas, seguimos un proceso similar al utilizado en la evaluación del modelo sin aumentación de datos.



Como el reentrenamiento consistió sólo de 5 épocas, podemos ver que no llega a mejorar las métricas del entrenamiento anterior.

En la matriz de confusión de este modelo, se observa un mayor número de celdas oscuras fuera de la diagonal principal. Esto indica un rendimiento inferior, con el modelo confundiendo varias cartas, especialmente las de valor 10, 11 y 12.

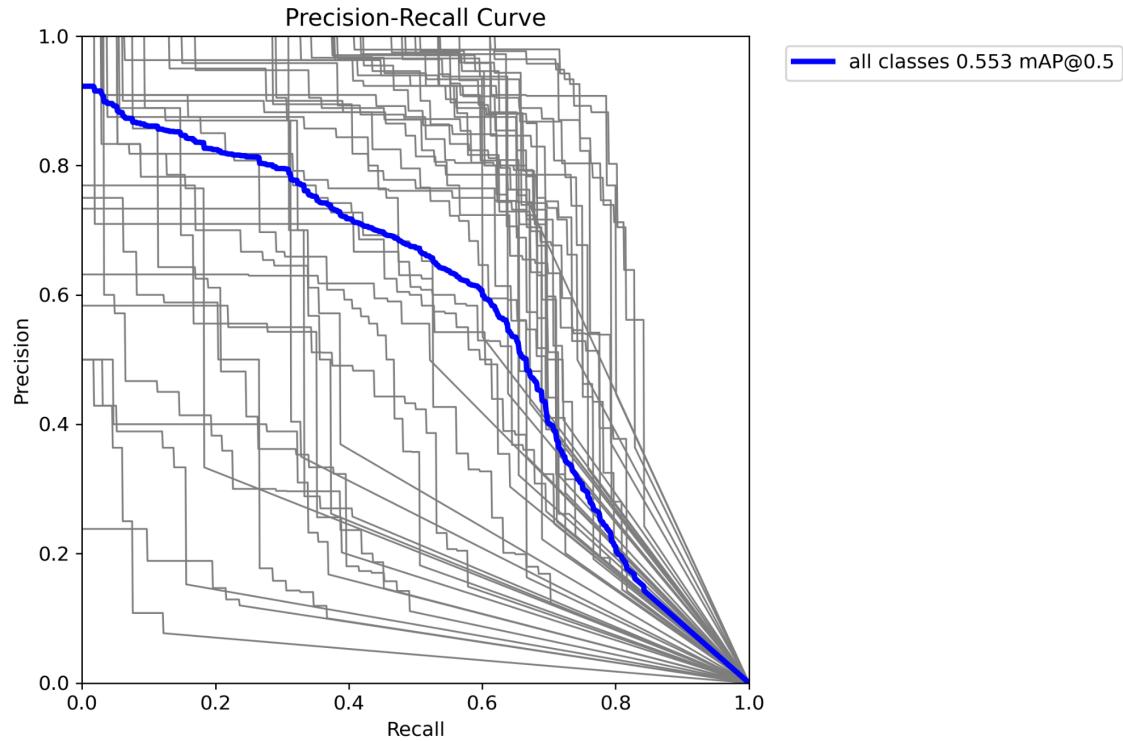




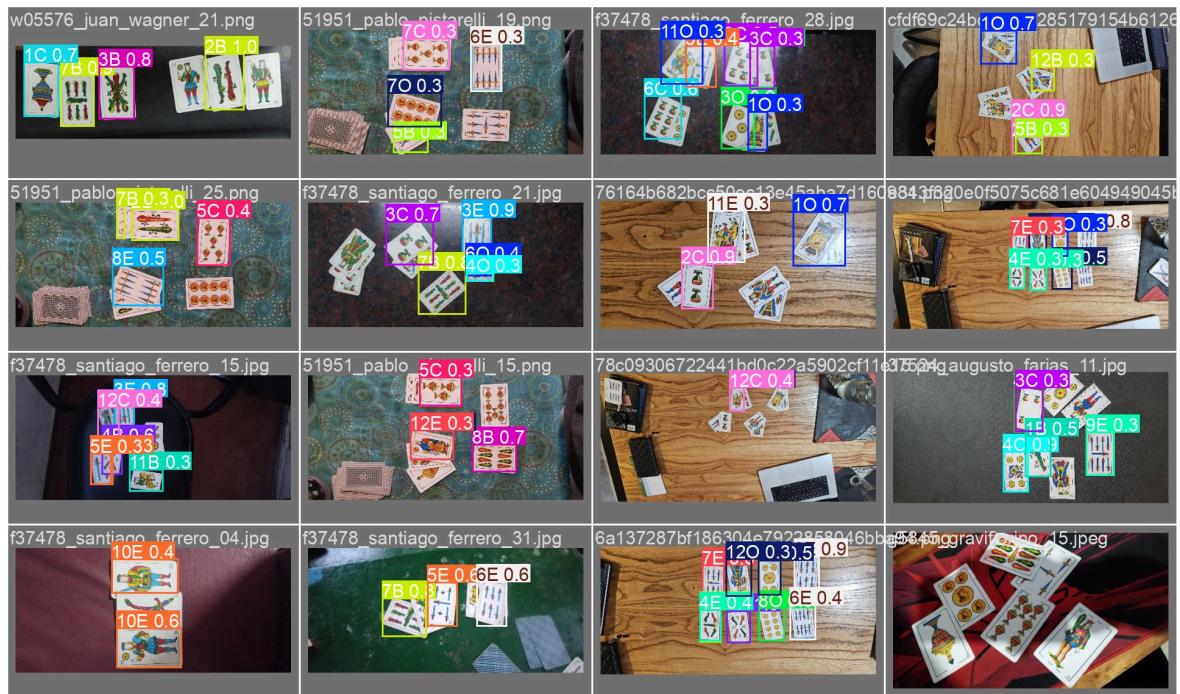
Aunque la implementación del parámetro `agnostic_nms=True` mejora la situación, seguimos detectando muchas cartas de manera incorrecta.

Además, aunque la cantidad de cartas predichas incorrectamente se reduce notablemente, todavía es muy alta la cantidad de cartas que no se detectan.

La curva Precision-Recall refleja un rendimiento no muy bueno del modelo, con una mAP (mean Average Precision) de 0.553 a un umbral de IoU de 0.5. Este resultado sugiere que el modelo no muestra una precisión razonable, y aún hay margen para mejoras significativas.



Batch de imágenes con cartas detectadas por el modelo:



Al comparar las imágenes generadas por el modelo aumentado con las de la evaluación anterior, se observa una disminución en el número de cartas correctamente detectadas. Además, se detecta un número considerable de cartas de manera incorrecta, lo que subraya la necesidad de entrenar al modelo por más épocas.

## Evaluación y cálculo del Envido

### Evaluación e Inferencia

El modelo fue evaluado en un conjunto de prueba separado, proporcionado por la cátedra, para medir su rendimiento en un entorno controlado. Se utiliza el modelo aumentado, pero puede cambiarse fácilmente cambiando el nombre del modelo.

Además, para cada imagen procesada, se generó un archivo TXT en formato YOLO que incluía el nivel de certeza en la detección. También se creó un archivo JSON que contenía, para cada imagen, la cantidad de cartas detectadas, cuáles eran estas cartas, y el cálculo de los puntos del envido.

```
"IMG_20240630_173828513_HDR.jpg": {
    "total_cards": 3,
    "cards": {
        "E": [
            7
        ],
        "C": [],
        "B": [
            3,
            7
        ],
        "O": []
    },
    "points": 30,
    "figure": "B"
},
"IMG_20240630_174333447.jpg": {
    "total_cards": 1,
    "cards": {
        "E": [],
        "C": [],
        "B": [
            7
        ],
        "O": []
    },
    "points": 7,
    "figure": "B"
},
```

Este es un extracto del JSON generado donde tengo una imagen con 3 cartas y se calculan los puntos del envido que son 30 de basto, y otra imagen con solo una carta que es el 7 de basto.

## Optimización del modelo

Utilizamos el modelo entrenado con los datos aumentados para evaluar los tiempos de inferencia y determinar cuánto se podía optimizar utilizando TensorRT. Este proceso se detalla en el notebook '00.run\_envido.ipynb'.

En este notebook, primero se recorren todas las imágenes de prueba y se grafican junto con los bounding boxes detectados. Luego, en una segunda fase, se recorren las mismas imágenes, pero en lugar de graficar, se calcula el tiempo promedio y los FPS.

El rendimiento sin optimización con CPU era de 6 FPS. Al utilizar una GPU, el rendimiento mejoró a 9 FPS. Con la optimización mediante TensorRT, se alcanzó un procesamiento de 11 FPS. Aunque no se logró un rendimiento extremadamente alto, se observó una mejora significativa en comparación con el procesamiento solo en CPU.

# Conclusiones

Este proyecto ha demostrado la viabilidad de utilizar técnicas de visión por computadora para el reconocimiento de cartas españolas, con el objetivo específico de calcular los puntos del envío en el juego de Truco. A lo largo del proceso, se abordaron varias etapas clave, incluyendo la creación de un dataset colaborativo, la personalización y unificación de los datos, el análisis exploratorio, y el entrenamiento de modelos de detección de objetos con y sin aumentación de datos.

El análisis inicial de los datos reveló la importancia de una adecuada preparación del dataset, incluyendo la verificación de las anotaciones y la estandarización del tamaño de las imágenes. Estas precauciones aseguraron que el modelo YOLOv8 pudiera aprovechar al máximo la información disponible.

El entrenamiento del modelo sin aumentación de datos arrojó resultados sólidos, con métricas como un mAP50 de 0.806 y un F1-score de 0.775, lo que indica que el modelo pudo identificar y clasificar correctamente las cartas en la mayoría de los casos. Sin embargo, cuando se entrenó un segundo modelo utilizando un dataset aumentado, se observó que, aunque las métricas finales fueron inferiores debido al menor número de épocas, el modelo mostró un aprendizaje más rápido. Esto se reflejó en el hecho de que, al comparar las métricas del modelo aumentado en la época 15 con las del modelo sin aumentación en la misma etapa, las primeras eran superiores.

Este hallazgo subraya el beneficio de la aumentación de datos en el proceso de entrenamiento, permitiendo que el modelo adquiera una mayor capacidad de generalización desde las primeras etapas. No obstante, las limitaciones computacionales impidieron llevar el entrenamiento del modelo aumentado al mismo número de épocas que el modelo sin aumentación, lo que probablemente habría permitido superar las métricas iniciales.

En conclusión, el proyecto no solo logró su objetivo principal, sino que también destacó la importancia de técnicas avanzadas como la aumentación de datos en la mejora del rendimiento de modelos de visión por computadora. Con más recursos computacionales, se espera que un modelo entrenado con datos aumentados y un mayor número de épocas pueda alcanzar e incluso superar el desempeño del modelo inicial, haciendo la solución aún más robusta y aplicable a escenarios reales de juego de Truco.