



Universidad Nacional de Rosario

Facultad de Ciencias Exactas, Ingeniería y  
Agrimensura

T.U.I.A

Computer Vision

2024

TOMÁS NAVARRO MIÑÓN || N-1257/2

## **Objetivo**

El objetivo de este trabajo es elaborar un software que, basado en modelos de reconocimiento de objetos y programación, sea capaz de calcular los puntos del “envido” del juego argentino de cartas conocido como “Truco” a partir del reconocimiento de cartas españolas. Se recorrerá al menos una vez el ciclo de vida completo de un entrenamiento, desde el análisis del juego y la problemática, el diseño de la captura de datos, la captura de datos, anotación, selección, aumentación, y elaboración de los diferentes conjuntos de datos.

## **Información**

El envido es un canto que se utiliza para desafiar al equipo contrario a mostrar la suma de las cartas de mayor valor que tienen en su mano. El envido se obtiene de la suma de dos cartas del mismo palo. A esta suma se le agregan 20 puntos. Por ejemplo, si tienes un 5 y un 6 del mismo palo, la suma sería 11. Agregando los 20 puntos, tendrías un total de 31 puntos en el envido. El puntaje mínimo es 0 (tres cartas negras de distinto palo), y el máximo es 33 (un 6 y un 7 del mismo palo).

## **Proceso:**

### **1. Creación de un dataset colaborativo:**

- Cada alumno realizó un aporte individual de al menos 30 imágenes reales de un mazo de cartas españolas, como un aporte a dataset colaborativo. Esto llevó a un dataset amplio y con imágenes con alta variabilidad. En mi caso llene con 50 imágenes ya que 30 consideraba que era poco para poder entrenar un modelo YOLO
- Estas imágenes fueron tomadas por celulares, en cada una de ella había alrededor de 5 a 8 cartas etiquetadas.
- Cada imagen debía ser etiquetada generando un bounding box en cada carta que aparecía allí. La norma que tomó el curso fue 1C para ancho de copa, 1E para ancho de espada, es decir el número correspondiente al valor de la carta y la letra al palo y J para el comodín.
- Cada alumno subió sus imágenes a un drive personal, el cual fue compartido para ser utilizado por cada compañero

## 2. Creación de un dataset personal:

- Teniendo en cuenta la propensión a errores debido a la cantidad de alumnos, pudimos chequear con otro compañero cuáles eran los drives que estaban en un formato correcto. De allí se descargó con cada link los archivos de imágenes y labels (donde se encuentran las etiquetas).

## 3. Procesamiento del dataset:

- Luego de tener listas las carpetas de images y labels, procedí a procesar las mismas, validando que no haya archivos erróneos.
- Convertí las etiquetas al formato YOLO: class x\_center y\_center width height

```
label_errors_dir = r'dataset/label_errors'
backup_dir = os.path.join(label_errors_dir, 'backup')

os.makedirs(backup_dir, exist_ok=True)

def is_yolov8_format(label):
    lines = label.strip().split('\n')
    for line in lines:
        parts = line.split()
        if len(parts) != 5:
            return False
    return True

def convert_to_yolov8(label):
    lines = label.strip().split('\n')
    converted_label = ''
    for line in lines:
        parts = line.split()
        if len(parts) == 9:
            class_id = parts[0]
            x_center1 = float(parts[1])
            y_center1 = float(parts[2])
            width1 = float(parts[3])
            height1 = float(parts[4])
            x_center2 = float(parts[5])
            y_center2 = float(parts[6])
            width2 = float(parts[7])
            height2 = float(parts[8])

            x_center = (x_center1 + x_center2) / 2
            y_center = (y_center1 + y_center2) / 2
            width = (width1 + width2) / 2
            height = (height1 + height2) / 2

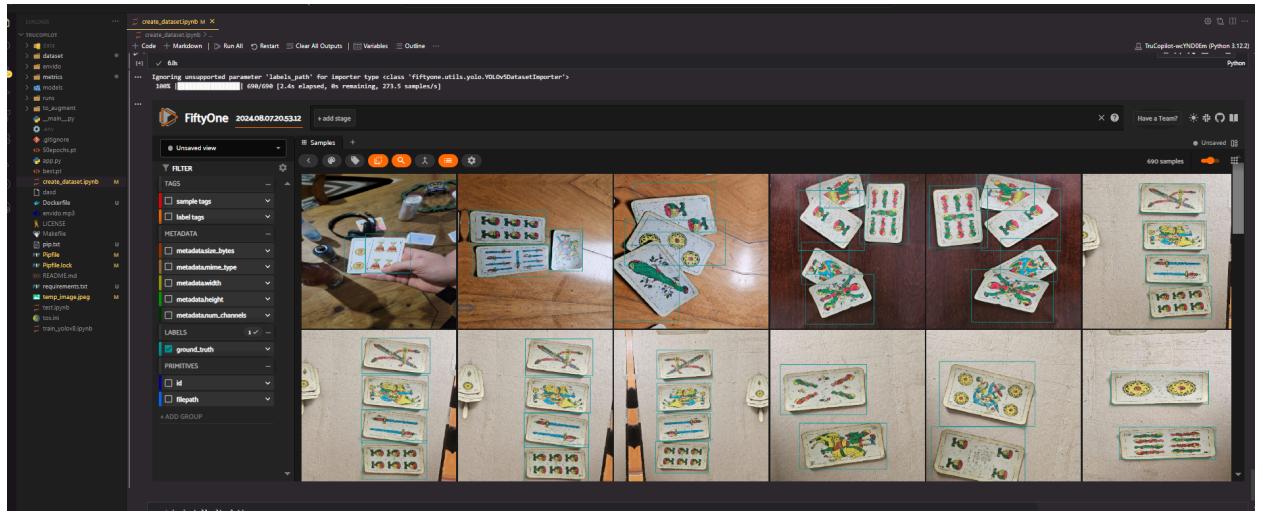
            converted_label += f'{class_id} {x_center} {y_center} {width} {height}\n"
        else:
            print(f"Warning: Unexpected label format: {line}")
    return converted_label.strip()

for filename in os.listdir(label_errors_dir):
    if filename.endswith('.txt'):
        label_path = os.path.join(label_errors_dir, filename)
        with open(label_path, 'r') as f:
            label = f.read()

        if not is_yolov8_format(label):
            shutil.copy(label_path, os.path.join(backup_dir, filename))

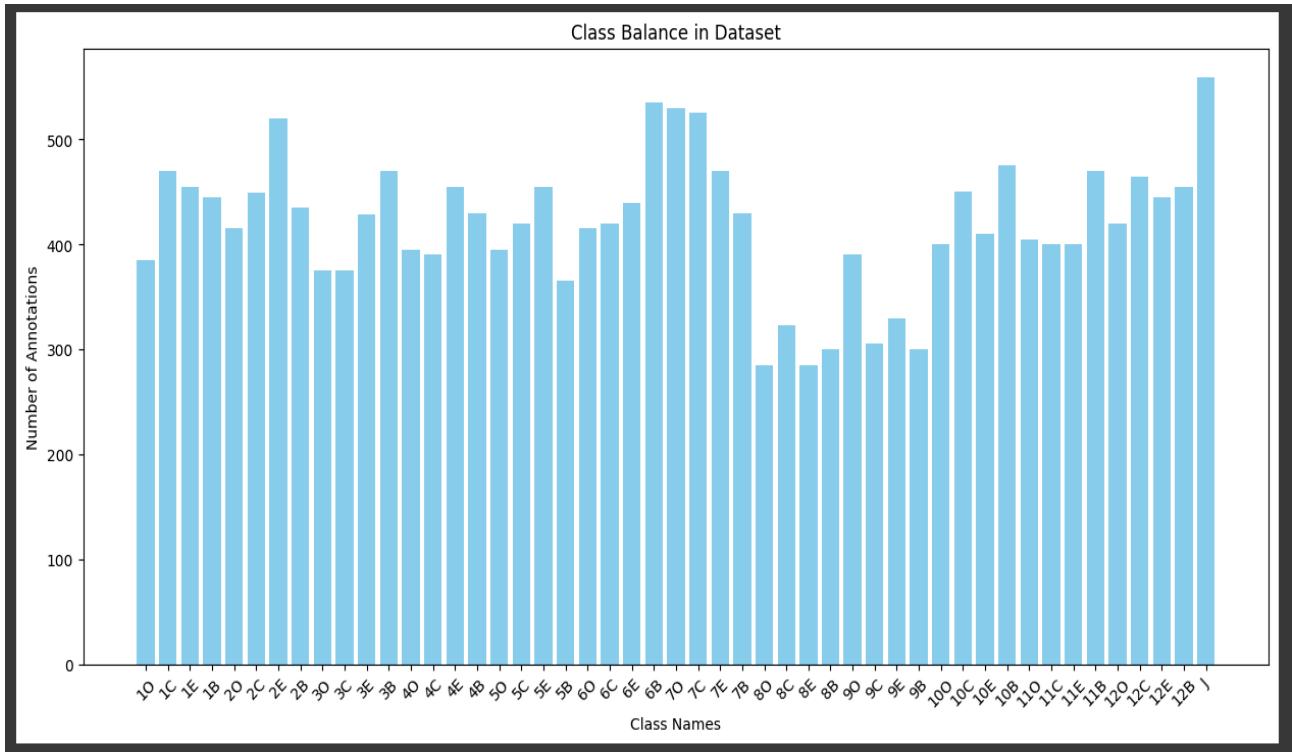
        converted_label = convert_to_yolov8(label)
        with open(label_path, 'w') as f:
            f.write(converted_label)
        print(f"Archivo {filename} convertido a formato YOLOv8 y original respaldado")
    else:
        print(f"Archivo {filename} ya está en formato YOLOv8")
```

- Eliminé fotos sin labels
- Renombré los archivos eliminando la letra ñ y tildes para no generar inconvenientes
- Realicé un resize a las imágenes para reducir el tamaño de los archivos.
- Chequee que estás transformaciones hayan sido de manera correcta utilizando la herramienta FiftyOne. Con esta herramiento he podido visualizar el dataset por completo viendo que en todos los casos se podía ver las cartas con sus respectivas etiquetas



#### 4. Chequeo de balance de clases

- Es importante para estos modelos de clasificación utilizar datasets balanceados para obtener mejores métricas.



- En este caso podemos ver que hay un buen balance. También podemos notar que las cartas 8 y 9 tienen menos imágenes. Esto se debe a que no todos los compañeros etiquetaron estas cartas. Podemos seguir adelante con el proceso ya que no hay un gran desbalance y para el envío no se utilizan las mismas

## 5. Aumentación del dataset

- Luego de tener procesado el dataset elegí aumentarlo con la librería albumenations para generar más imágenes para el entrenamiento. Aquí abajo un fragmento de código usando la mencionada librería

```

# Cargar imágenes y etiquetas
def save_augmented_data(image, label, image_name, label_name, aug_images_dir, aug_labels_dir):
    cv2.imwrite(os.path.join(aug_images_dir, image_name), image)
    with open(os.path.join(aug_labels_dir, label_name), 'w') as f:
        f.write(label)

transform = A.Compose([
    A.HorizontalFlip(p=0.5),
    A.RandomBrightnessContrast(p=0.2),
    A.Rotate(limit=10, p=0.5),
    A.ShiftScaleRotate(shift_limit=0.1, scale_limit=0.2, rotate_limit=10, p=0.5)
], bbox_params=A.BboxParams(format='yolo', label_fields=['class_labels']))

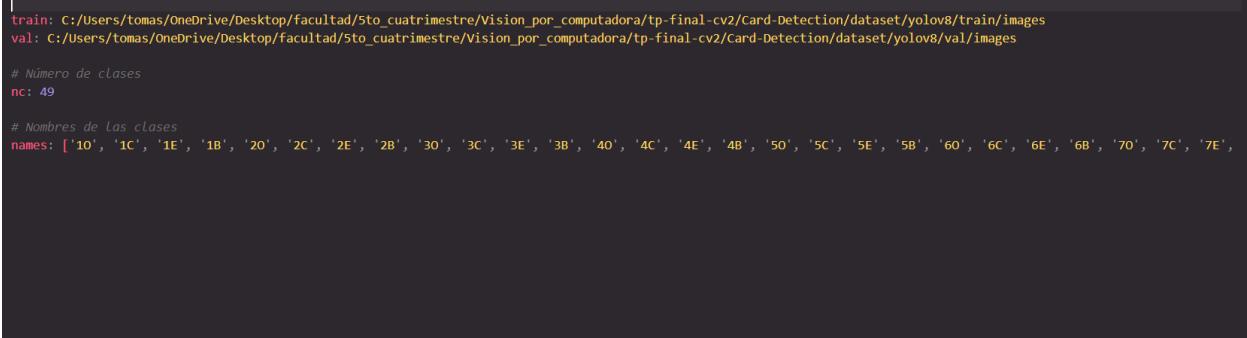
```

- Primero dividí el dataset en val y train, eligiendo 30% para val y 70% para train. Elegí estos valores ya que la aumentación se le realiza solo a train.

- A las imágenes de train les realicé una aumentación agregando cambios de color, tonalidades y ruido. Cabe aclarar que las transformaciones de color no son muy bruscas para no generar cambios muy grandes en el color de las ya que nos interesa el color para la predicción de palos.

## 6. Entrenamiento

- Luego de tener el dataset aumentado y el archivo YAML creado (este archivo contiene las rutas a las imágenes y labels de train y test y también los nombres de las clases) procedí a entrenar el modelo YOLOV8 este archivo YAML, en esta foto es para entrenarlo local. Debido al costo computacional que tiene poder re-entrenar estos modelos, lo que hice fue entrenarlo en Kaggle. Kaggle nos permite usar sus tarjetas gráficas que se encuentran en la nube durante 30 hs sin costo alguno.



```

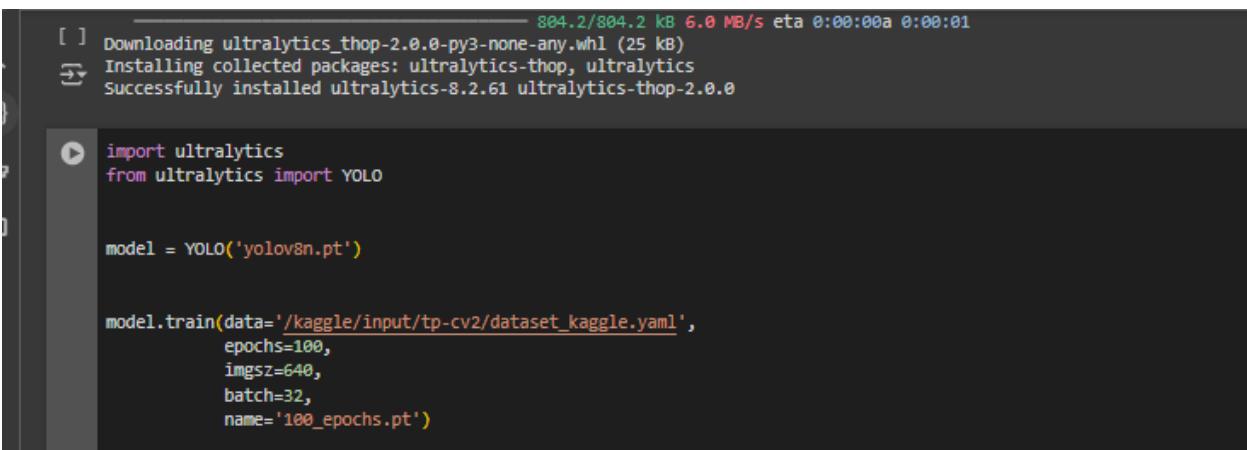
train: C:/Users/tomas/OneDrive/Desktop/facultad/5to_cuatrimestre/Vision_por_computadora/tp-final-cv2/Card-Detection/dataset/yolov8/train/images
val: C:/Users/tomas/OneDrive/Desktop/facultad/5to_cuatrimestre/Vision_por_computadora/tp-final-cv2/Card-Detection/dataset/yolov8/val/images

# Número de clases
nc: 49

# Nombres de las clases
names: ['10', '1C', '1E', '1B', '20', '2C', '2E', '2B', '30', '3C', '3E', '3B', '40', '4C', '4E', '4B', '50', '5C', '5E', '5B', '60', '6C', '6E', '6B', '70', '7C', '7E',

```

- Para ello le pasé el image size de 640 (tamaño al que redimensioné las imágenes) un batch size de 32 durante 100 épocas



```

[ ] Downloading ultralytics_thop-2.0.0-py3-none-any.whl (25 kB)
[ ] Installing collected packages: ultralytics-thop, ultralytics
[ ] Successfully installed ultralytics-8.2.61 ultralytics-thop-2.0.0

import ultralytics
from ultralytics import YOLO

model = YOLO('yolov8n.pt')

model.train(data='/kaggle/input/tp-cv2/dataset_kaggle.yaml',
            epochs=100,
            imgsz=640,
            batch=32,
            name='100_epochs.pt')

```

## 7. Evaluación del modelo

- En esta sección analicé las métricas obtenidas por el modelo YOLO

class	Images	Instances	Box(P)	R	mAP50	mAP50-95
all	690	4271	0.951	0.912	0.966	0.719
10	68	68	0.956	0.971	0.985	0.742
1C	97	99	0.99	0.953	0.985	0.744
1E	99	99	0.98	0.998	0.995	0.779
1B	94	94	0.919	0.962	0.974	0.771
20	95	95	0.959	0.916	0.964	0.709
2C	100	100	0.952	0.933	0.966	0.714
2E	120	120	0.985	0.892	0.972	0.71
2B	82	82	0.948	0.988	0.995	0.723
30	81	81	0.927	0.897	0.964	0.713
3C	73	73	0.932	0.863	0.96	0.691
3E	81	81	0.965	0.951	0.974	0.748
3B	91	91	0.988	0.912	0.982	0.715
40	88	88	0.931	0.939	0.94	0.683
4C	74	74	0.957	0.909	0.967	0.726
4E	99	99	0.989	0.96	0.969	0.734
4B	96	96	0.942	0.92	0.969	0.734
50	73	73	0.911	0.92	0.984	0.759
5C	72	72	0.963	0.833	0.942	0.703
5E	75	75	0.971	0.953	0.981	0.679
5B	80	80	0.961	0.949	0.975	0.749
60	86	86	0.914	0.942	0.966	0.75
6C	97	97	0.934	0.871	0.941	0.742
6E	92	92	0.971	0.908	0.962	0.709
6B	106	106	0.98	0.936	0.99	0.722
70	106	106	0.968	0.903	0.983	0.728
7C	121	121	0.922	0.893	0.959	0.729
7E	94	94	0.979	0.886	0.975	0.706
7B	103	103	0.966	0.886	0.97	0.757
80	61	61	0.854	0.928	0.984	0.717
8C	63	63	0.911	0.828	0.916	0.698
8E	68	68	0.937	0.898	0.968	0.686
8B	69	69	0.967	0.986	0.984	0.726
90	91	91	0.948	0.968	0.992	0.744
9C	70	70	0.983	0.857	0.935	0.703
9E	63	63	0.889	0.889	0.956	0.687
9B	66	66	0.955	0.955	0.971	0.688
100	79	79	0.944	0.911	0.974	0.716
10C	86	86	0.939	0.872	0.97	0.708
10E	92	92	0.961	0.859	0.968	0.699
10B	104	104	0.978	0.85	0.955	0.702

110	80	80	0.934	0.884	0.962	0.71
11C	82	82	0.962	0.918	0.969	0.768
11E	72	72	0.978	0.764	0.931	0.674
11B	81	81	0.956	0.827	0.878	0.667
120	95	95	0.937	0.937	0.976	0.699
12C	89	89	0.924	0.91	0.978	0.744
12E	93	93	0.997	0.968	0.987	0.741
12B	90	90	0.943	0.918	0.96	0.709
J	106	106	0.96	0.967	0.982	0.747

- **Interpretación de las métricas globales:**

1. **Box(P) - Precisión de la caja (0.951):**
  - Esto indica que el modelo tiene una alta precisión al predecir las cajas delimitadoras para todas las clases ya que el 95.1% de las predicciones de las cajas son precisas.
2. **R - Recall (0.912):**
  - Este valor muestra que el modelo es capaz de detectar el 91.2% de las instancias presentes en las imágenes.
3. **mAP50 (0.966):**
  - El valor de mAP50 (Media de Precisión Promedio con umbral IoU de 0.5) es 0.966. Esto significa que el modelo tiene una alta precisión promedio al predecir las cajas delimitadoras con un umbral de 0.5.
4. **mAP50-95 (0.72):**
  - La métrica mAP50-95 mide la precisión promedio sobre un rango de umbrales de IoU (de 0.5 a 0.95). Un valor de 0.72 indica que el modelo mantiene una buena precisión promedio incluso con umbrales más estrictos.

Conclusiones:

- Clases con Buen Desempeño: Las clases tienen buenos valores de mAP50 y mAP, mostrando un desempeño sólido en términos de detección.
- Clases con Bajo Desempeño: Las clases como 8s, 9s, muestran bajos valores en de mAP50-95 indicando que el modelo tiene dificultades para detectar estas clases con precisión.
- General: El modelo es relativamente bueno en la detección de muchas clases, pero hay margen para mejorar en algunas áreas específicas, especialmente en clases con baja precisión y recall.

## Predicciones



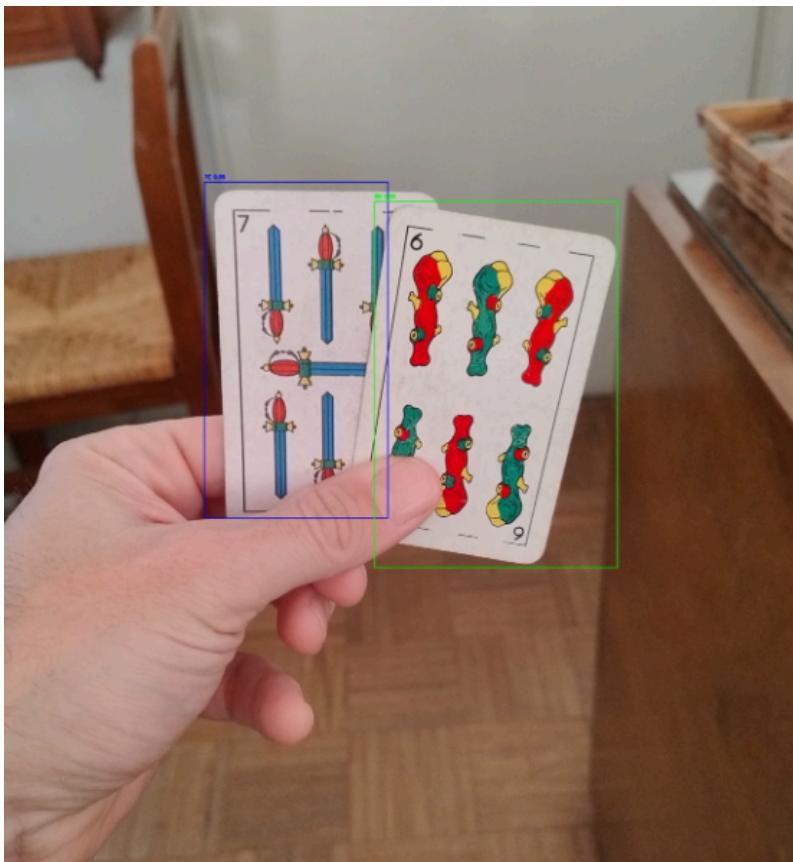
## Inferencia

La cátedra nos otorgó un Colab en el cual realicé diversas operaciones

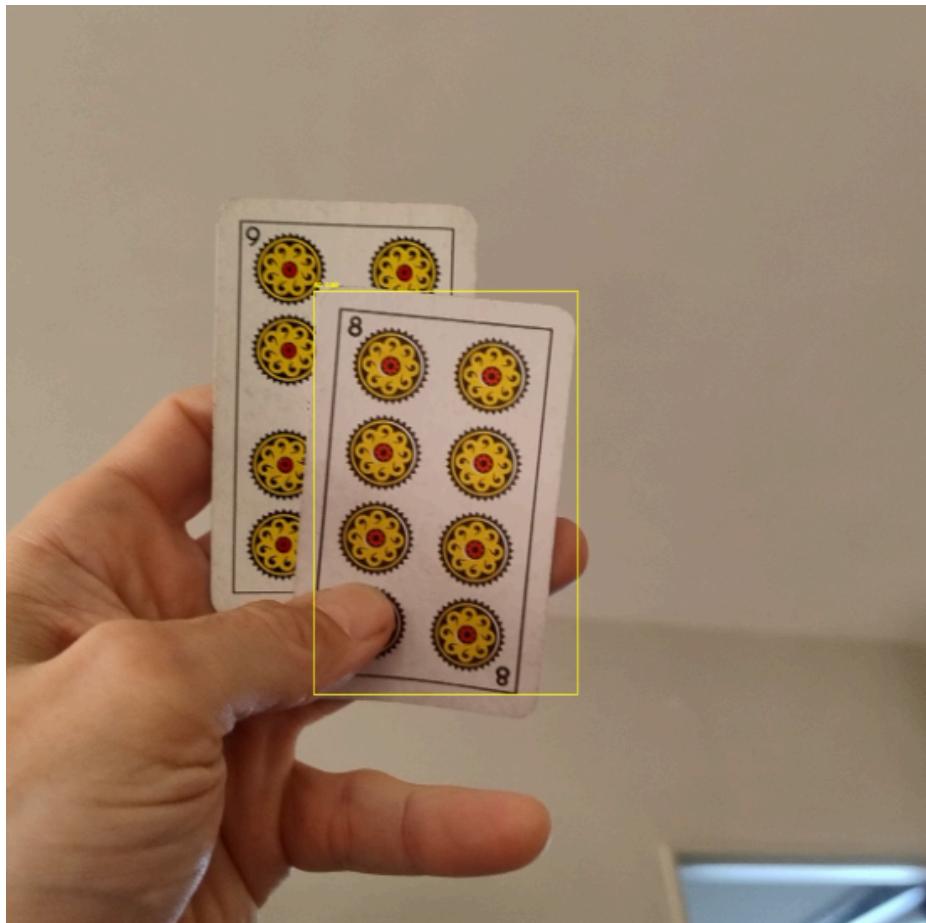
1. Predicciones de imágenes: Se predijo con el modelo entrenado diversas imágenes y se guardó los resultados en formato yolo con el número de clase, centro en x, centro en y, ancho, alto y confidence
2. Función para calcular envido: Creé la función que tiene la lógica para sumar y devolver los puntos de envido en una imagen
3. Prueba: Se realizó el testeо en las imágenes
4. Guardar métricas: Se guardó en un archivo envido.json las cartas que el modelo ve en la imagen, separada por palo y el envido que tiene.



```
{  
    "/content/drive/MyDrive/Trabajo Practico CV2 Toma\u0301s Nava":  
        "total_cards": 3,  
        "cards": {  
            "E": [  
                7  
            ],  
            "C": [],  
            "B": [  
                3,  
                6  
            ],  
            "O": [],  
            "J": []  
        },  
        "points": 29,  
        "figure": "B"  
    }  
}
```



```
0: 640x512 1 6B, 1 7E, 193.6ms
Speed: 7.0ms preprocess, 193.6ms inference, 1.9ms postprocess per image at shape (1,
{
    "/content/drive/MyDrive/Trabajo Practico CV2 Toma\u0301s Navarro Min\u0303o\u0306
        "total_cards": 2,
        "cards": {
            "E": [
                7
            ],
            "C": [],
            "B": [
                6
            ],
            "O": [],
            "J": []
        },
        "points": 0,
        "figure": "N/A"
    }
}
```



```
0: 640x512 1 80, 242.4ms
Speed: 12.9ms preprocess, 242.4ms inference, 3.0ms postprocess
{
    "/content/drive/MyDrive/Trabajo Practico CV/00000000000000000000000000000000.jpg": {
        "total_cards": 1,
        "cards": {
            "E": [],
            "C": [],
            "B": [],
            "O": [
                8
            ],
            "J": []
        },
        "points": 0,
        "figure": "N/A"
    }
}
```