Sistema de Detección de Figuras Geométricas con YOLO

Sistema completo para detectar figuras geométricas y sus colores usando YOLO, Flask y HTML.

🚀 Características

- ✅ Detección de figuras geométricas (triángulos, cuadrados, rectángulos, círculos, etc.)
- ✅ Identificación de colores (rojo, azul, verde, amarillo, etc.)
- ✅ Procesamiento de imágenes desde archivo, URL o arrastrar y soltar
- ✅ Procesamiento de videos MP4
- ✅ Interfaz web moderna y responsive
- ✅ API REST con Flask

📋 Requisitos Previos

- Python 3.8 o superior
- pip (gestor de paquetes de Python)
- Cámara web (opcional)

🔧 Instalación

1. Clonar o crear el proyecto

bash

```
mkdir yolo-detector
cd yolo-detector
```

2. Crear entorno virtual (recomendado)

bash

```
python -m venv venv

# Activar en Windows:
venv\Scripts\activate

# Activar en Linux/Mac:
source venv/bin/activate
```

3. Instalar dependencias

bash

```
pip install -r requirements.txt
```

4. Estructura de carpetas

Crea la siguiente estructura:

```
yolo-detector/
│
├── app.py              # Backend Flask
├── requirements.txt    # Dependencias
├── templates/
│   └── index.html      # Frontend HTML
├── uploads/            # Carpeta para archivos subidos (se crea automáticamente)
└── processed/          # Carpeta para archivos procesados (se crea automáticamente)
```

5. Descargar modelo YOLO

El modelo YOLOv8n se descargará automáticamente la primera vez que ejecutes la aplicación. Si prefieres descargarlo manualmente:

bash

```
python -c "from ultralytics import YOLO; YOLO('yolov8n.pt')"
```

🎯 Uso

1. Iniciar el servidor

bash

```
python app.py
```

El servidor se iniciará en `http://localhost:5000`

2. Abrir el navegador

Abre tu navegador y ve a: `http://localhost:5000`

3. Usar la aplicación

Opción 1: Subir Archivo

- Arrastra y suelta una imagen o haz clic para seleccionar
- Haz clic en "Procesar Imagen"
- Visualiza los resultados con las detecciones

Opción 2: Desde URL

- Pega la URL de una imagen
- Haz clic en "Procesar desde URL"

Opción 3: Video

- Selecciona un archivo de video (MP4, AVI, MOV)
- Haz clic en "Procesar Video"
- Descarga el video procesado con las detecciones

📡 API Endpoints

POST /detect/image

Detecta objetos en una imagen.

Formas de enviar:

1. **Archivo subido (multipart/form-data):**

bash

```
curl -X POST http://localhost:5000/detect/image \
  -F "file=@imagen.jpg"
```

2. **Desde URL (JSON):**

bash

```
curl -X POST http://localhost:5000/detect/image \
  -H "Content-Type: application/json" \
  -d '{"url": "https://ejemplo.com/imagen.jpg"}'
```

3. **Base64 (JSON):**

bash

```bash
curl -X POST http://localhost:5000/detect/image \
  -H "Content-Type: application/json" \
  -d '{"image_base64": "..."}'
```

Respuesta:

json

```json
{
  "success": true,
  "detections": [
    {
      "shape": "Círculo",
      "color": "Rojo",
      "confidence": 0.85,
      "bbox": [100, 150, 200, 250]
    }
  ],
  "processed_image": "..."
}
```

POST /detect/video

Procesa un video completo.

bash

```bash
curl -X POST http://localhost:5000/detect/video \
  -F "file=@video.mp4"
```

Respuesta:

json

```
{
  "success": true,
  "video_path": "/download/processed_video.mp4",
  "detections": [...],
  "total_frames": 300
}
```

GET /download/<filename>

Descarga un archivo procesado.

🎨 Personalización

Modificar detección de colores

En `app.py`, función `detect_color()`, ajusta los rangos HSV:

python

```
if avg_hue < 10 or avg_hue > 170:
    return "Rojo"
elif 10 <= avg_hue < 25:
    return "Naranja"
# ... añadir más colores
```

Cambiar modelo YOLO

Puedes usar otros modelos YOLO:

python

```
# YOLOv8 nano (más rápido)
model = YOLO('yolov8n.pt')

# YOLOv8 small (balance)
model = YOLO('yolov8s.pt')

# YOLOv8 medium (más preciso)
model = YOLO('yolov8m.pt')

# YOLOv8 large (más preciso pero más lento)
model = YOLO('yolov8l.pt')
```

Ajustar clasificación de formas

En `app.py`, función `classify_shape()`:

python

```
def classify_shape(contour):
    peri = cv2.arcLength(contour, True)
    approx = cv2.approxPolyDP(contour, 0.04 * peri, True)
    vertices = len(approx)

    # Añadir más formas
    if vertices == 8:
        return "Octágono"
```

🐛 Solución de Problemas

Error: "No module named 'ultralytics'"

bash

```
pip install ultralytics
```

Error: "CUDA not available"

Si no tienes GPU NVIDIA, YOLO usará CPU automáticamente (más lento pero funcional).

Error de CORS

Verifica que `flask-cors` esté instalado y configurado en `app.py`.

Video no se procesa

- Verifica que el formato sea MP4, AVI o MOV
- Asegúrate de tener suficiente espacio en disco
- El procesamiento de videos puede tomar tiempo

Imágenes no se cargan desde URL

- Verifica que la URL sea accesible públicamente
- Algunas URLs requieren autenticación

📊 Rendimiento

- **Imagen (1920x1080):** ~1-3 segundos
- **Video (30 fps, 10 seg):** ~30-60 segundos
- Depende de tu hardware (CPU/GPU)

🔒 Seguridad

Para producción, considera:

1. Agregar autenticación
2. Limitar tamaño de archivos
3. Validar tipos de archivo
4. Usar HTTPS
5. Implementar rate limiting

🤝 Contribuciones

¡Las contribuciones son bienvenidas! Siéntete libre de mejorar el código.

📝 Notas

- El modelo YOLOv8n es ligero y rápido, ideal para pruebas
- Para mejor precisión, usa modelos más grandes (yolov8m, yolov8l)
- La detección de colores usa espacio HSV para mayor precisión
- Las formas se detectan usando aproximación de contornos

📧 Soporte

Si tienes problemas, verifica:

1. Que todas las dependencias estén instaladas
2. Que el servidor Flask esté corriendo
3. Que los puertos no estén bloqueados por firewall
4. Los logs en la consola de Python

🎓 Recursos Adicionales

- Documentación YOLO
- OpenCV Tutoriales
- Flask Documentation

¡Disfruta detectando figuras geométricas! 🎉