**Informe sobre la puesta en marcha y portabilidad de YOLOsv7 en UNCaLens**

## **1. Restricciones de la Aplicación**

La aplicación en desarrollo tiene los siguientes requisitos clave:

* Debe funcionar sin conexión a Internet.
* No puede depender del repositorio original de YOLOv7 para la inferencia.
* Debe ser liviana y evitar almacenar archivos innecesarios.
* Necesita permitir el cambio dinámico de modelos de IA.

El problema principal surge cuando el modelo .pt, entrenado en el repositorio original de YOLOv7, requiere dependencias e importaciones propias del entorno donde fue generado, impidiendo su ejecución independiente. Como en el caso de la IA “best\_4class.pt” la cual es imposible hacer andar fuera de su repositorio original.  
Los creadores no diseñaron YOLOv7 para ejecutarse directamente fuera del repositorio sin modificaciones, pero dan herramientas de exportación como TorchScript, ONNX y TensorRT para hacerlo más portátil.

## **2. Soluciones factibles**

### **Opción 1: Guardar solo los pesos del modelo (state\_dict)**

* En el entrenamiento, guardar únicamente los pesos sin la arquitectura completa:  
  torch.save(model.state\_dict(), "modelo\_pesos.pt")

En la aplicación, definir la arquitectura manualmente y cargar los pesos:  
model = YOLOv7Model()

model.load\_state\_dict(torch.load("modelo\_pesos.pt", map\_location="cpu"))

model.eval()

* **Ventaja:** Evita dependencias innecesarias del repositorio original.
* **Desventaja:** Se debe definir la arquitectura manualmente en la aplicación.

### **Opción 2: Convertir a TorchScript (.torchscript.pt)**

Transformar el modelo .pt a .torchscript.pt con:  
scripted\_model = torch.jit.trace(model, torch.randn(1, 3, 640, 640))

* torch.jit.save(scripted\_model, "modelo.torchscript.pt")
* **Ventaja:** Elimina dependencias del repositorio, mejora la portabilidad y optimiza la inferencia.
* **Desventaja:** Algunos parámetros internos del modelo pueden quedar estáticos.

## **3. Consideraciones Importantes**

### **Sobre el guardado de solo los pesos**

* Es necesario definir la arquitectura en la aplicación antes de cargar los pesos.
* Se debe asegurar que la arquitectura en la aplicación coincida con la del entrenamiento.
* Permite mayor flexibilidad en la modificación de parámetros dinámicos.

### **Sobre TorchScript**

* **¿La conversión afecta la precisión de la IA?**
  + No debería haber cambios significativos si la conversión se hace correctamente.
  + Se debe verificar que la conversión no elimine operaciones dinámicas esenciales.
* **¿Impacta en el consumo de memoria y CPU/GPU?**
  + **CPU:** Puede mejorar el rendimiento al optimizar el grafo computacional.
  + **GPU:** Sigue siendo compatible con CUDA, sin cambios drásticos en rendimiento.
* **¿Se pueden modificar parámetros en tiempo de ejecución?**
  + Si los parámetros se definen como atributos dentro del modelo (self.conf\_threshold = 0.25), TorchScript los congela y no pueden cambiarse.

Para evitar esto, deben pasarse como argumentos en el forward():  
def forward(self, x, conf\_threshold: float):

* + Si el modelo original cargaba configuraciones externas, TorchScript ya no podrá acceder a ellas.