## **Reflexiones sobre el manejo de variables runtime en la inferencia**

### **¿Qué hago con el padding que sale del letterbox?**

Cuando uso letter\_box en el preprocess, me devuelve el padding. Este padding lo necesito sí o sí en el postprocess para eliminarlo cuando ya tengo los vectores de salida y quiero llevar todo de vuelta al espacio de la imagen original.

El problema es que todo el ciclo de inferencia lo hago a través del controlador. Y no estoy seguro si está bien que el padding viaje por ahí. Pero tampoco quiero romper el diseño modular que tengo. ¿Estoy cometiendo un error si dejo que el controlador transporte esa info?

### **¿Y si meto las variables de runtime dentro del objeto config?**

Estaba pensando en aprovechar el objeto config que ya tengo para meter ahí las variables que se generan o modifican en tiempo de ejecución:

* El padding
* La confianza que setea el cliente
* En un futuro: el color de los boxes, la fuente del texto, etc

¿Está mal meter eso ahí? Técnicamente no viene del JSON, pero config es un objeto, no tiene por qué venir todo desde un archivo. Mientras no lo serialice como si fuera parte del JSON original, no estaría mintiendo.

### **¿No sería mejor separar claramente lo que viene del JSON y lo que vive en runtime?**

Sí, probablemente sí. Capaz es tener algo como:

config.runtime.padding

config.runtime.confidence\_threshold

config.runtime.box\_color

Entonces:

* Lo del JSON sigue quedando en config como hasta ahora.
* Todo lo que cambié el cliente o se genere en tiempo real va a config.runtime

Así no lleno de basura la parte de config que sí representa el archivo.

### **¿Cómo evito que se vuelva un basurero global?**

Eso es clave. No quiero que config.runtime se vuelva un lugar donde tiro cualquier variable suelta. Tiene que estar bien controlado:

* Solo van cosas que cambian en tiempo de ejecución o que dependen de la imagen que entra
* No deberían mezclarse las responsabilidades (no meter ahí paths, ni lógica del modelo, ni nombres de adaptadores, etc)

Y además, todo lo que meta ahí tiene que tener un motivo: o lo necesito en postprocess, o lo necesita el cliente, o se usa para debug.

### **¿Cuál es el plan de acción entonces?**

1. Terminar todos los módulos y probar todo para asegurar que la estructura base funciona.
2. Decidir si realmente me conviene usar objetos para representar el config (como Pydantic models) o si sigo con dicts puros.
3. Hacer que el controlador sea el único responsable de leer el JSON y construir el objeto config.
4. Introducir config.runtime como contenedor exclusivo de datos mutables en tiempo de inferencia.

Y todo esto, sin perder de vista el orden ni convertirlo en un Frankenstein. Que siga siendo legible y fácil de debuggear.