burn

A little case study on using GstAnalytics from Rust

GStreamer Conference 2025

23 October 2025

Sebastian Dröge <sebastian@centricular.com>

Centricular

gstreamer

Rationale

- Most GstAnalytics code is C or Python
- How well does it work from Rust?
- How usable is burn?

Plan

- burn-based YOLOX object detection inference element
  - Produces raw tensor metas for detections from video frames
- YOLOX tensor decoder element
  - Converts raw tensors into relation metas

burn 🔥

- https://burn.dev
- PyTorch-like deep learning framework in Rust 🦀
- Lots of computation backends (CPU / GPU / NPU based)
- Model compiled & optimized for a specific backend
- "Loaders" (codegen) for PyTorch, ONNX, ... models

YOLOX element

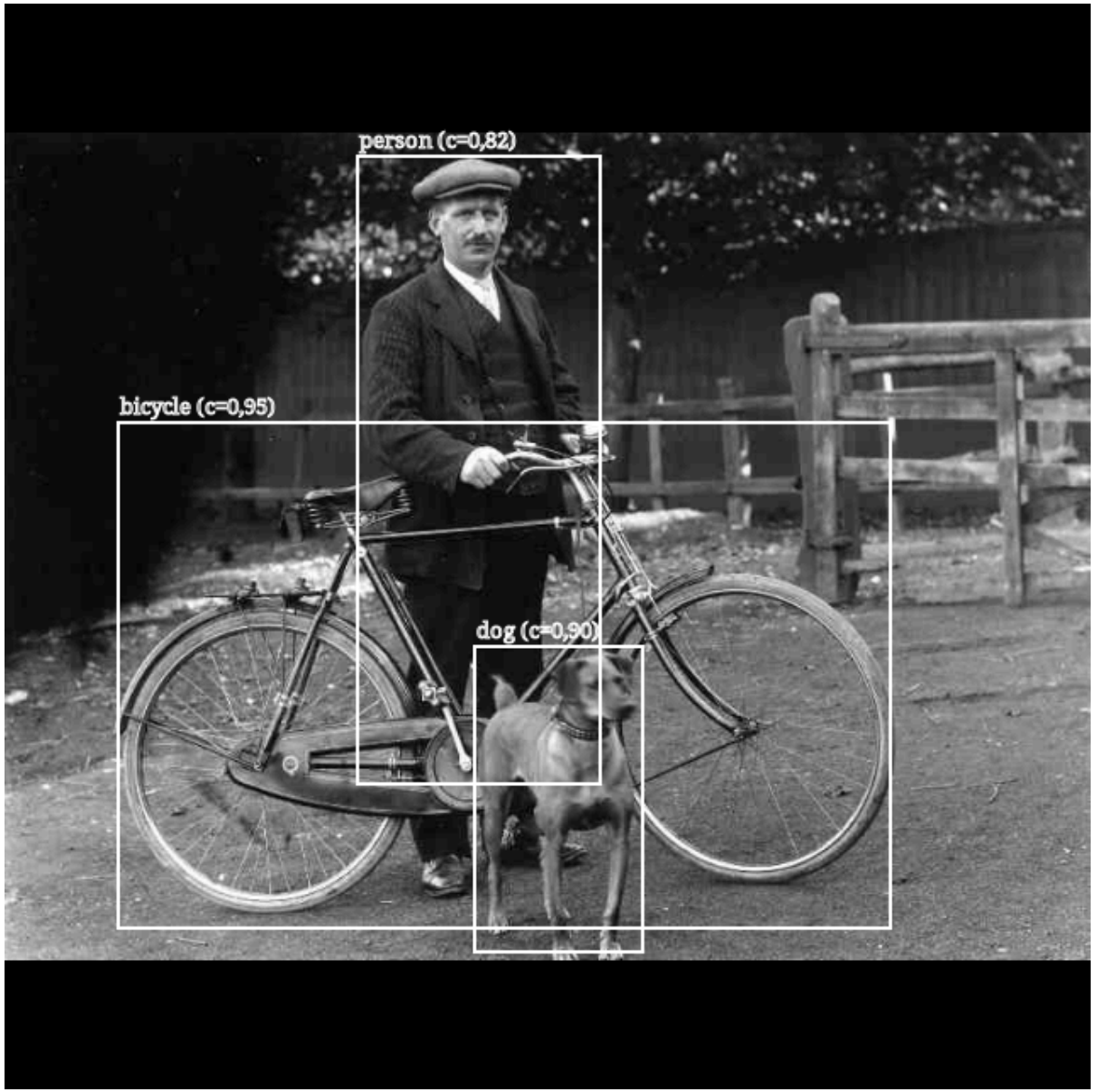
- Takes 640x640 (*) RGB input frames
  - Converts into planar internally
  - No suitable planar RGB format in GStreamer
  - 3 function calls
- Outputs input plus raw tensor meta
  - Wrapping of a 8400x85 (*) float matrix into a meta
- Basically no code except for element boilerplate and caps negotiation

YOLOX tensor decoder element

- Adds object detection and classification metas
- 100 SLOC algorithm, overall 500 SLOC
  - Comparison: C YOLOv9 tensor decoder 3x as big
  - Mostly because of C book-keeping
- Can be autoplugged by `tensordecodebin`
  - Should also work with e.g. `onnxinference`
- Output can be used by `ioutracker`, `objectdetectionoverlay`, etc.

Centricular

gstreamer



Centricular

gstreamer

Questions? Comments?

Code available at

https://gitlab.freedesktop.org/gstreamer/gst-plugins-rs/-/merge_requests/2347