

# GStreamer for Machine Vision & Beyond



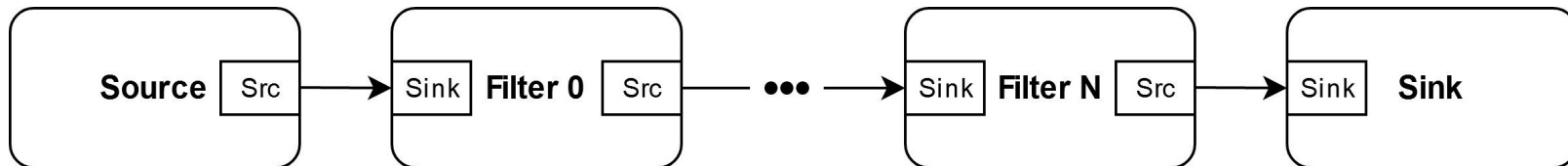
<https://github.com/BrianOfrim/gstreamer-pyspin-src>

# What is GStreamer?

An open source framework for constructing multimedia pipelines

Pipelines consist of connected elements of 3 basic types:

- **Sources:** Elements that **produce** data
  - cameras, microphones, media files, network
- **Filters:** Elements that **manipulate** data
  - compression, transcoders, re-formatters, overlay, augmentations, processing
- **Sinks:** Elements that **consume** data
  - display, files, network stream, real time processing applications



# Example Video Pipeline

```
$ gst-launch-1.0 pypinsrc ! videoconvert ! xvimagesink sync=false
```

**pypinsrc** - produces images from Spinnaker compatible cameras

**videoconvert** - ensures data from pypinsrc can be consumed by xvimagesink

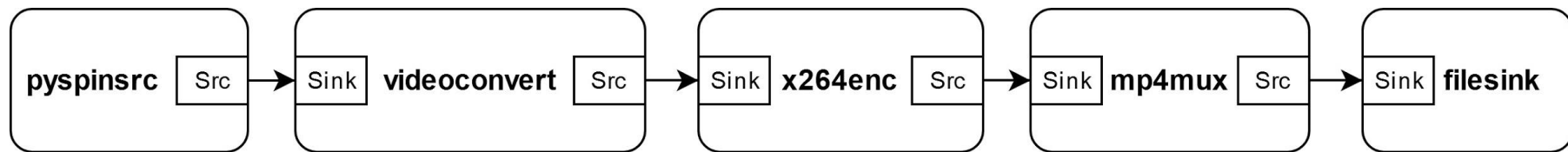
**xvimagesink** - displays the images



# Compression

# Sample Compression Pipeline

```
$ gst-launch-1.0 pypinsrc ! videoconvert ! x264enc ! mp4mux ! filesink location="pypinsrc_xh264.mp4"
```



**x264enc** - encodes raw video into H.264 compressed data

**mp4mux** - merges audio and video streams into .mp4 file format

**filesink** - writes incoming data into a local file

# Hardware Compression Plug-ins

Fully and easily utilize available hardware resources

Due to the popularity of multimedia applications, most devices have dedicated hardware for encoding (compressing) and decoding (decompressing) video streams.

This hardware can help to:

- **accelerate** encoding/decoding
- **free up the cpu** for other tasks
- **improve energy efficiency** when compared to software implementations

# Nvidia Linux for Tegra

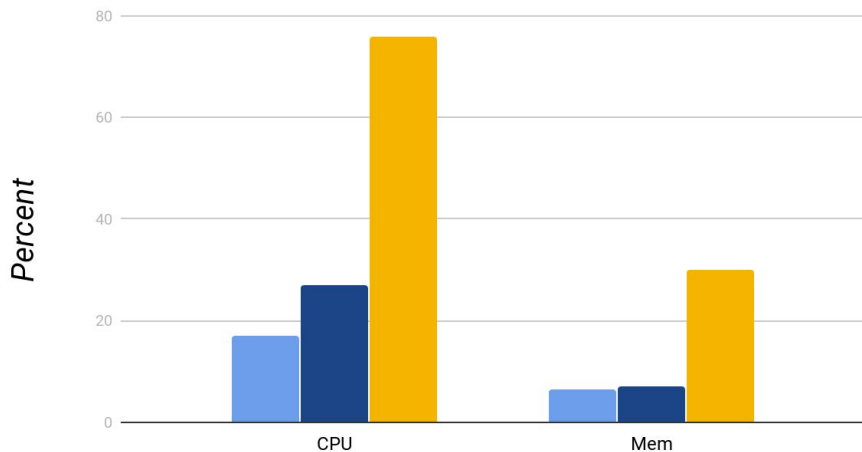


Nvidia hardware accelerated video encoding (compression)

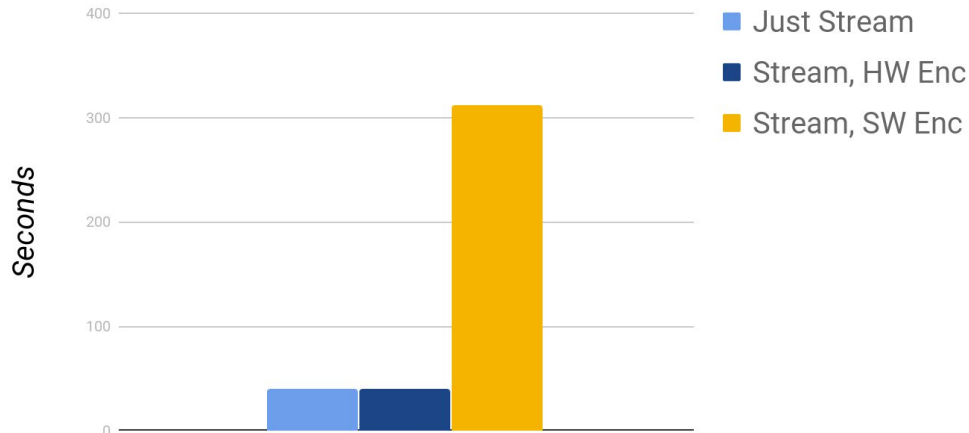
**H.264 benchmark:** BFS-U3-31S4C-C, 2048x1536, BGRA, 25fps = ~314MB/s, 1000 frames

**Computer:** Nvidia Jetson Nano

Resource Usage (Lower is Better)



Runtime (Lower is Better)



# Intel Graphics - VAAPI

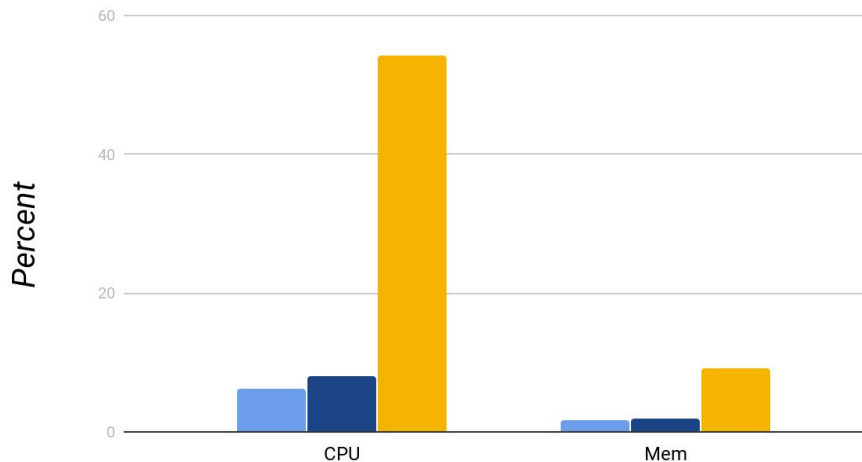


Intel hardware accelerated video encoding (compression)

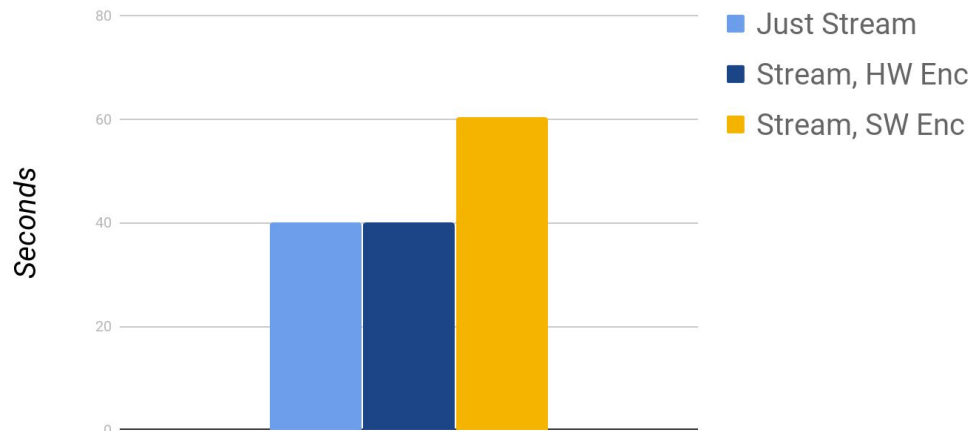
**H.264 benchmark:** BFS-U3-31S4C-C, 2048x1536, BGRA, 25fps = ~314MB/s

**Computer:** Intel i5-8350U CPU @ 1.70GHz, UHD Graphics 620

Resource Usage (Lower is Better)



Runtime (Lower is Better)





# Network Live Streaming

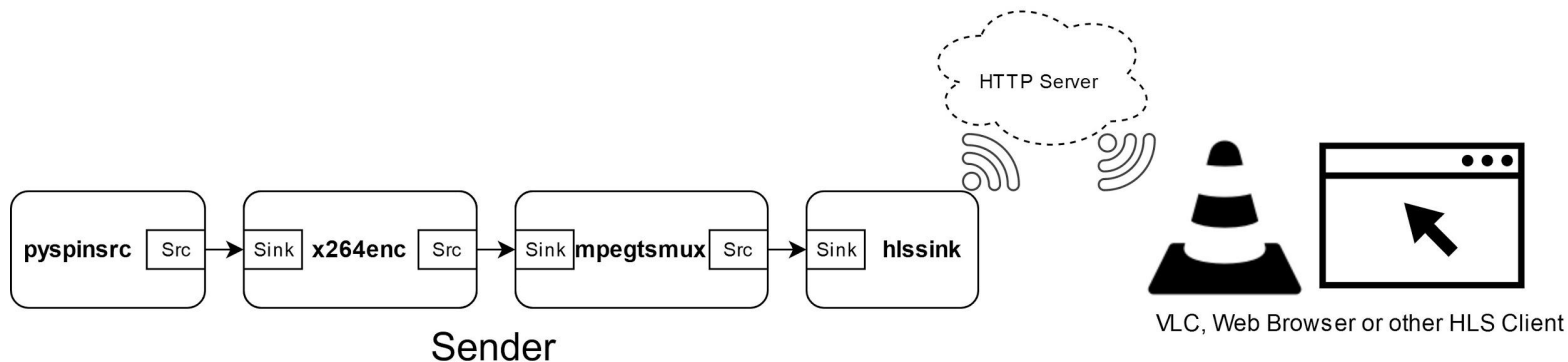
# HLS (HTTP Live Streaming)

Most useful for compatibility, versatility, and quality at the expense of latency

**Sender** uploads short video file segments to a server

**Receiver** downloads these segments from the server and plays them back

**Example application:** Broadcasting over the internet to many types of devices

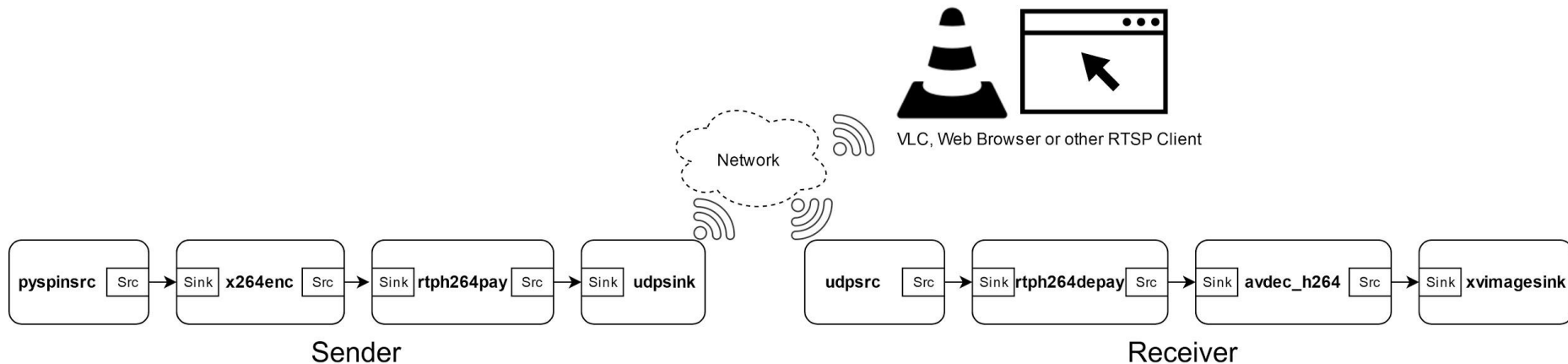


# RTP/RTSP (Real-time Transport Protocol)

Most useful for streaming data with minimal latency on a local network

Can be configured to do a 1 (sender) to N (receivers) multicast

**Example applications:** Streaming video over an ad-hoc network from a robot

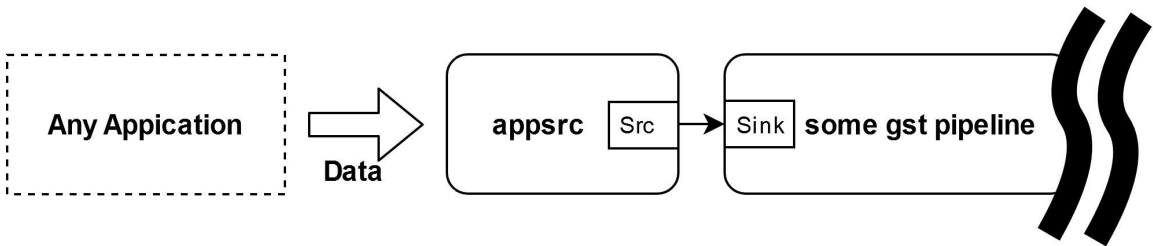


# Integrating with Applications

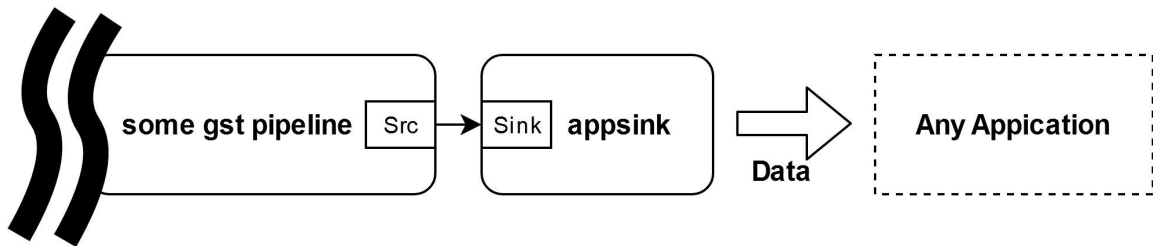
# Appsrc / Appsink

GStreamer has plug-ins that allow for any application to act as a source or sink

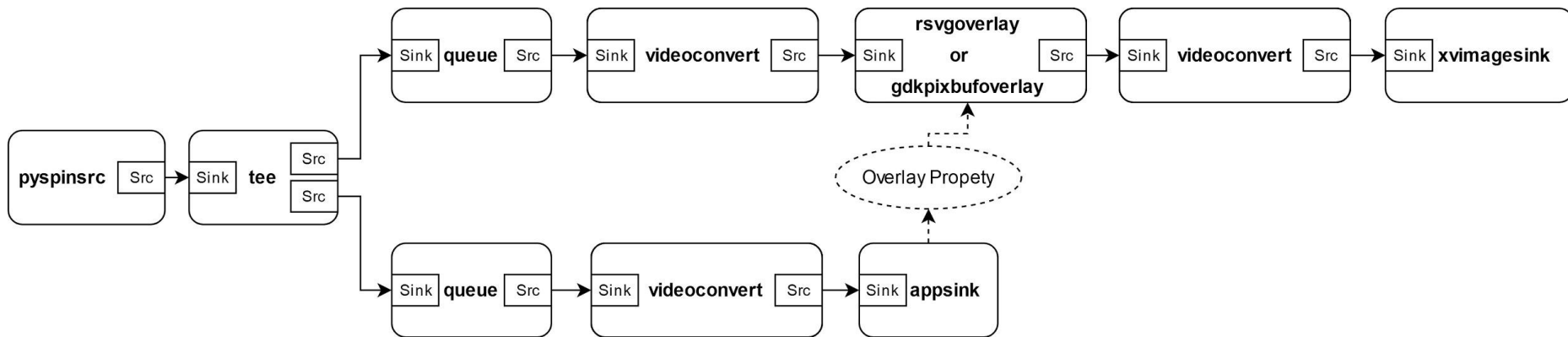
appsrc - allows an application to feed data buffers to a pipeline



appsink - allows an application to receive data buffers from a pipeline



# Example Image Processing and Overlay Pipeline



**tee** - splits data to multiple pads

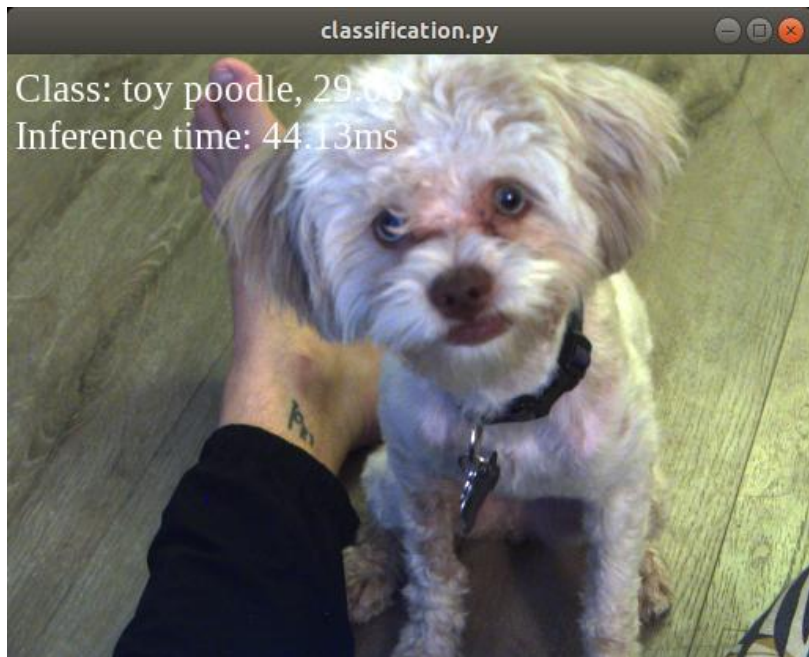
**queue** - data queue

**rsvgoverlay** - overlays SVG graphics over the video stream

**gdkpixbufoverlay** - overlays an image on the video stream

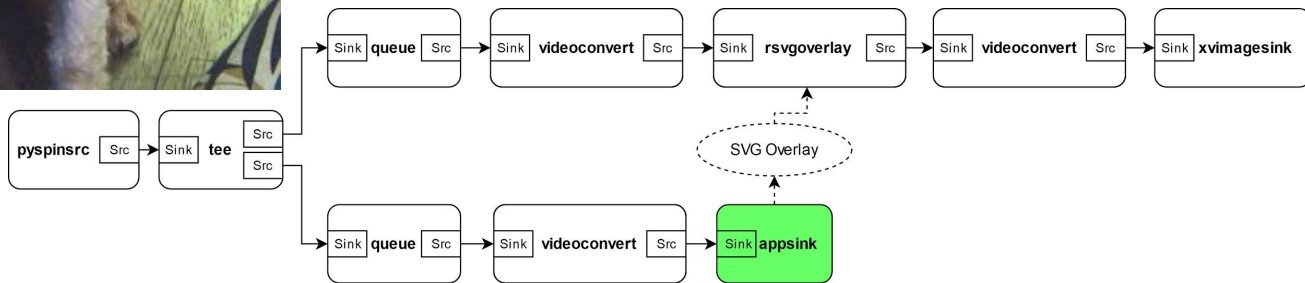
**appsink** - allows an application, in this case python, to access image data buffers

# Appsink Example: Classification

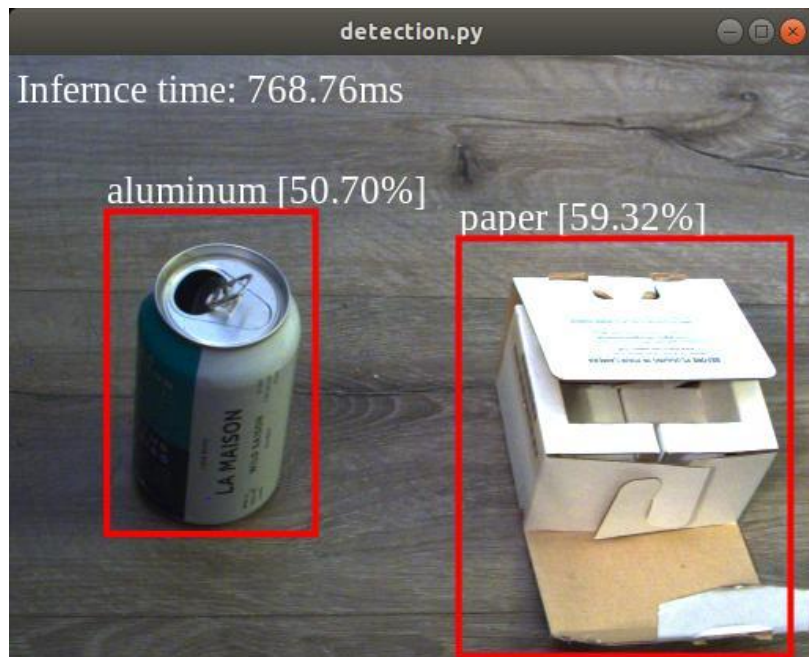


Pre-trained mobilenet\_v2 classifier from:

<https://github.com/pytorch/vision/blob/master/torchvision/models/mobilenet.py>

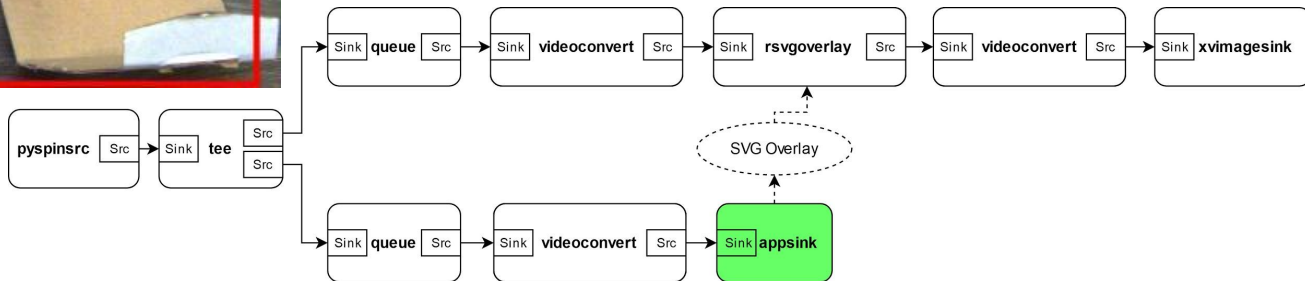


# Appsink Example: Recycling Detection



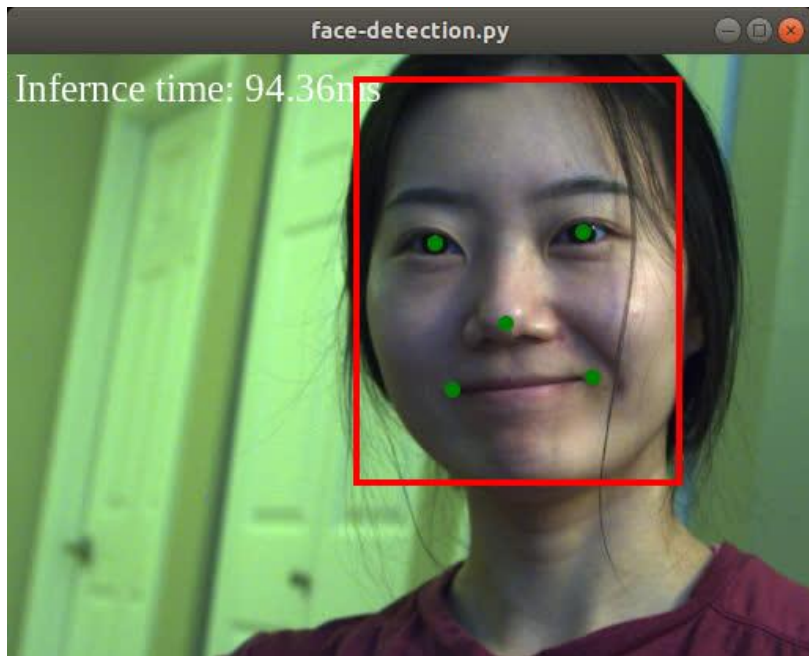
Recycling detection trained via the Boja process:

<https://github.com/BrianOfrim/boja>



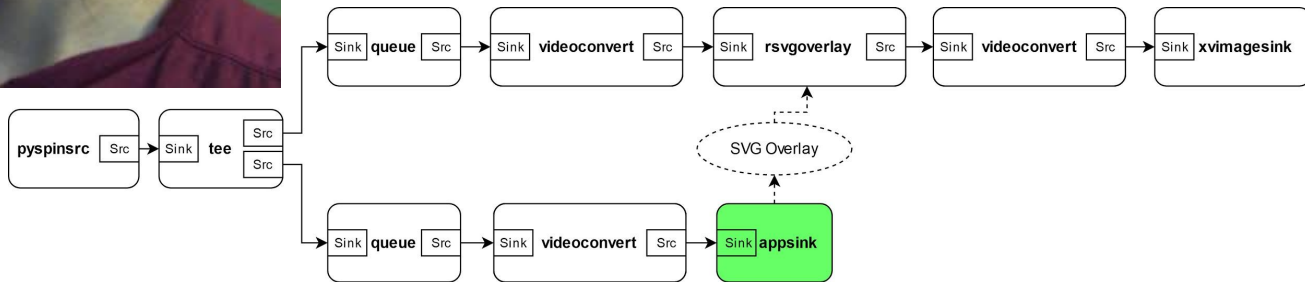


# Appsink Example: Face Detection



Face Detection model from:

<https://github.com/timesler/facenet-pytorch>

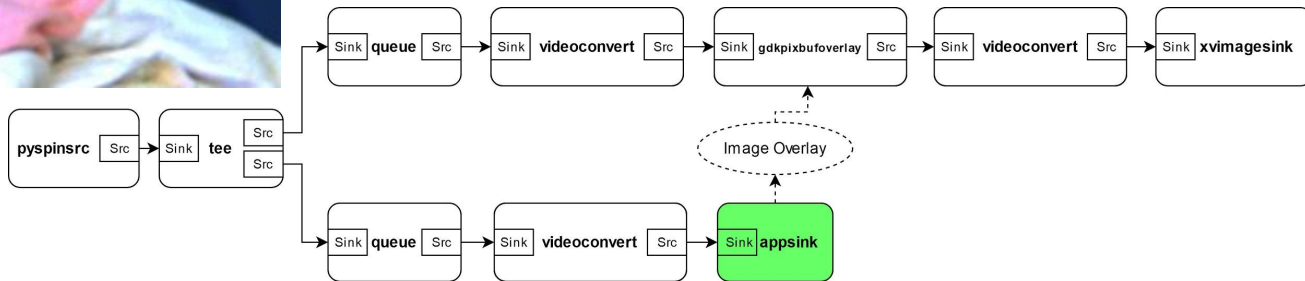


# Appsink Example: Segmentation



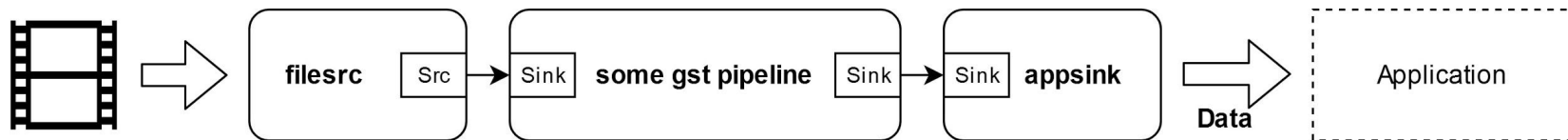
Pretrained segmentation model from:

<https://github.com/pytorch/vision/tree/master/torchvision/models/segmentation>



# Faster and Easier Application Development

Stream from a video to help with development and testing



Stream from a camera for deployment

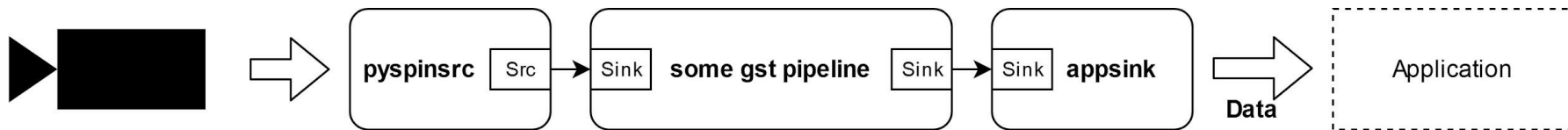


# Deploy Application on the Edge or a Server

Application does not need to know or care where the image stream is coming from

Image source can be a local camera or a real time network video stream

## Edge:



## Server:

