Rapid Development of AI-Powered Embedded Vision Solutions—Without a Team of Experts

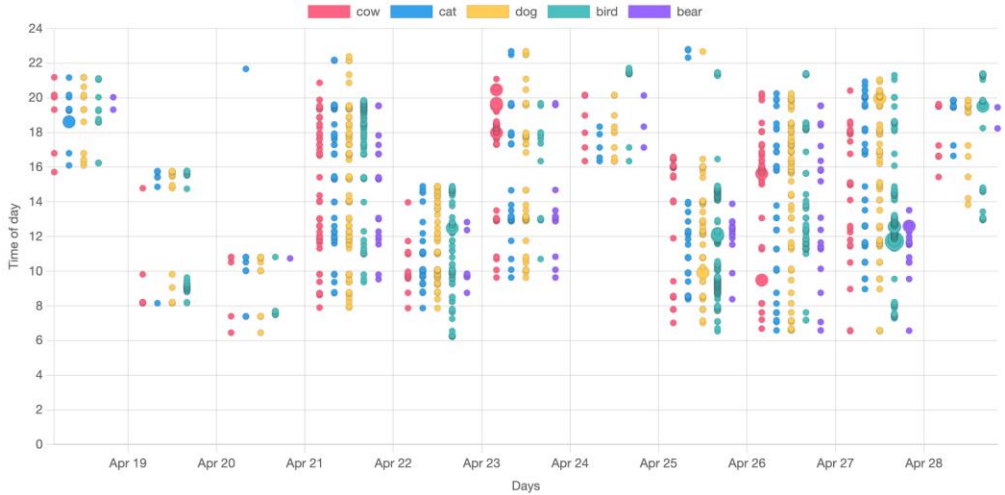
Marcel Wouters

Senior Back-end Engineer

Network Optix

Network Optix

YOLOv8 Model to Visualisation



Components

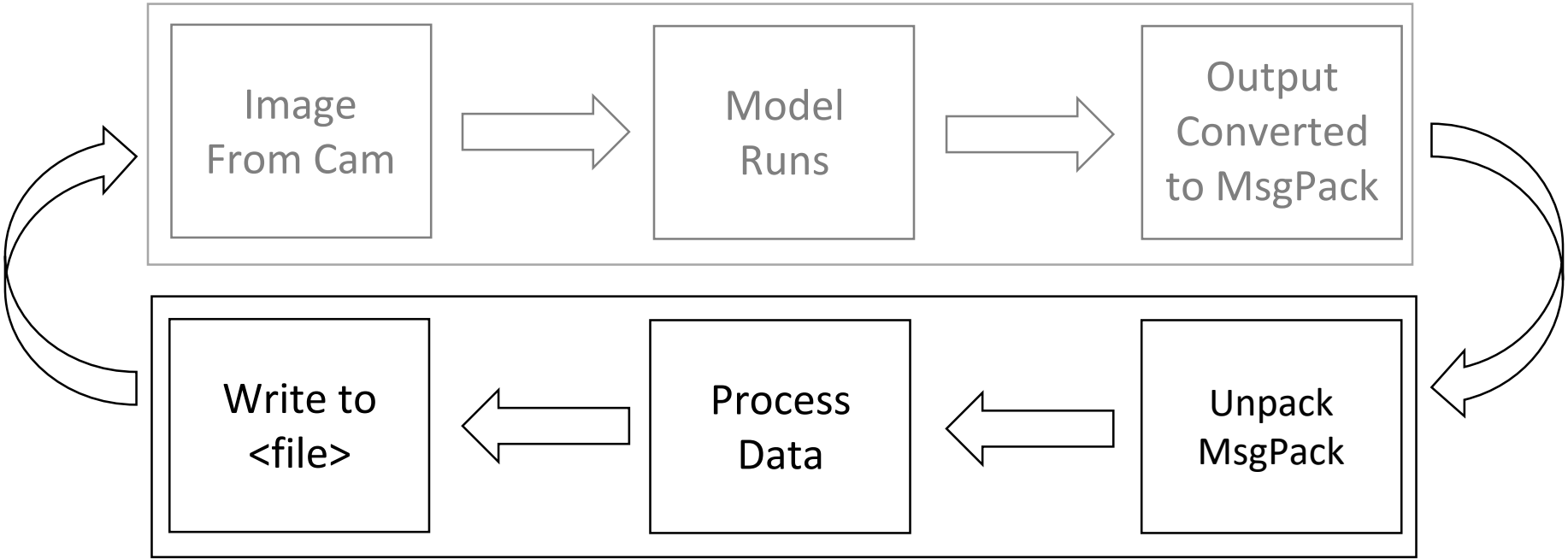
Installable components:

- Linux Server
- MetaVMS Server
- AI plugin
- MetaVMS Client

Custom components:

- Development postprocessor
- Data capture with Python script
- Data visualisation with chart.js

Custom Component – Data Capturing



Postprocessors

- Per server only 1 postprocessor per type is started
- Unix domain sockets are used for communication
- Data is encoded in MsgPack format
- Next inference will wait for postprocessor to finish
- Option to receive raw input data
- Option to send updated model output data back

Development Postprocessor

- Can be found in the integration-sdk repository “postprocessor-broadcast-server” (soon)
- Convert MsgPack input to JSON
- Easy connect to using TCP
- Allows for many readers
- Cannot read image data nor send data back

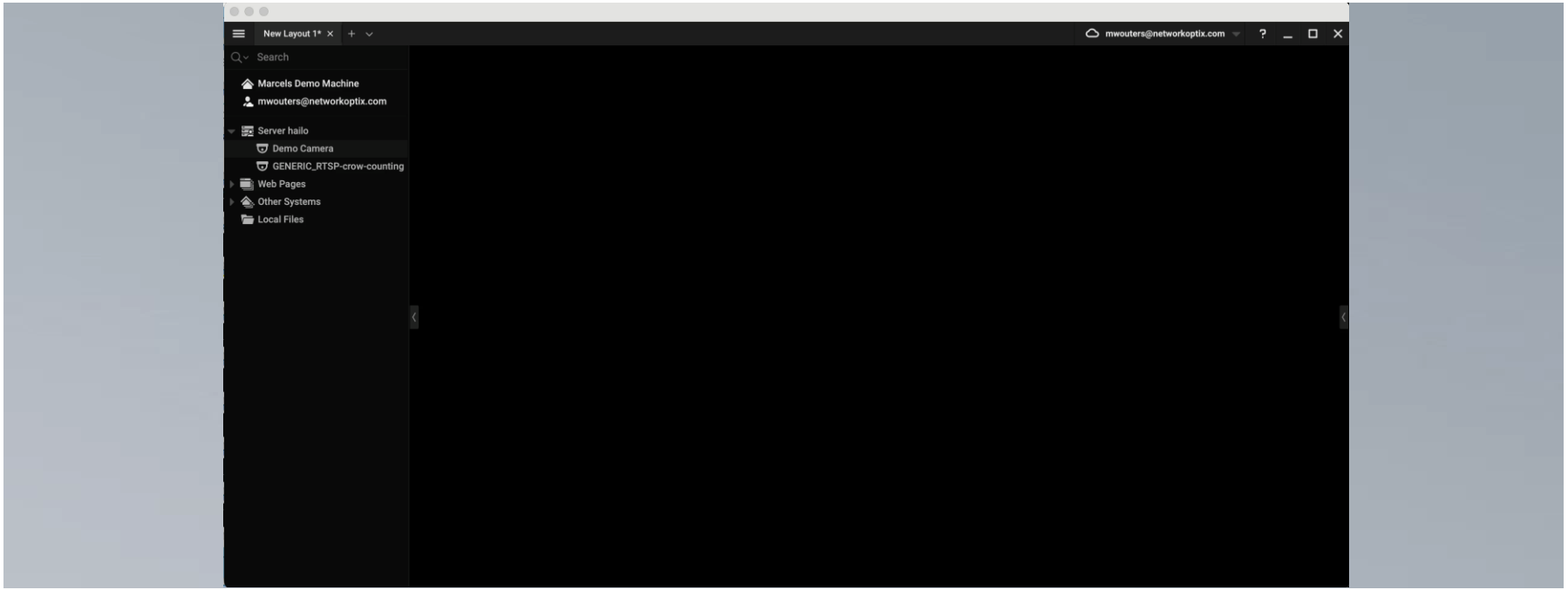
Installing postprocessor-broadcast-server

```
$ git clone https://github.com/scailable/sclbl-integration-sdk.git
Cloning into 'sclbl-integration-sdk'…
$ cd sclbl-integration-sdk/postprocessor-broadcast-server
$ # follow instructions for building and installation
```

```
{   “Name”:"PostProcessor Broadcast Server",
    “Command”:”/opt/…/postprocessor-broadcast-server”,
    “SocketPath”:"/tmp/postprocessor-broadcast-server.sock",
    "ReceiveInputTensor": false,
    "ReceiveBinaryData": false,
    "NoResponse": true }
```

Network Optix

Configure a Pipeline in the AI Cloud



Broadcast Server Output

$ nc localhost 7100
{“DeviceID":"740aef91-d1b0-b2a8-6f5a-e6f716599757",
“DeviceName”:"Demo Camera”, “Timestamp”: 2025-05-06T16:55:58.768+02:00”,"Width":1920,"Height":1080,
"IndexIndex":0,"Counts":null, “BBoxes_xyxy":{"bench":[[770.89484, 508.4393,1006.5688,588.6948]]},”ObjectsMetaData":null,
"Scores":null}

{"DeviceID":"740aef91-d1b0-b2a8-6f5a-e6f716599757",
"DeviceName":"Demo Camera", "Timestamp": "2025-05-06T16:55:58.888+02:00","Width":1920,"Height":1080,
"IndexIndex":0,"Counts":null, "BBoxes_xyxy":{"vase":[[1436.2456, 736.65295,1498.6831,818.2757]]},”ObjectsMetaData":null,
"Scores":null}

…continues like this…


```
{
  "DeviceID": "740aef91-d1b0-b2a8-6f5a-…",
  "DeviceName": "Demo Camera",
  "Width":1920,
  "Height":1080,
  "Timestamp": "2025-05-06T16:55:54.755+02:00",
  "BBoxes_xyxy": {
    "mouse": [[
      1435.3203, 735.72766, 1497.7578, 817.35034
    ]]
  }
}
```


Network Optix

Capturing Data for Visualisation

```
ANIMALS = ['bird', 'cat', 'dog', 'bear', 'cow']

def process_message(json_obj: dict) -> None:
   timestamp = strptime(json_obj['Timestamp'][:18], '%Y-%m-%dT%H:%M:%S')

    json_line = {}
    if 'BBoxes_xyxy' in json_obj:
        for animal in ANIMALS:
            if animal in json_obj['BBoxes_xyxy']:
               json_line[animal] = len(json_obj['BBoxes_xyxy'][animal])

    if len(json_line) > 0:
        json_line['timestamp'] = timestamp.strftime('%Y-%m-%dT%H:%M:%S.%f')
        line = json.dumps(json_line)
        with open('/tmp/detections.jsonl', 'a') as f:
            f.write(line + '\n')
```

Broadcast Server Loop

```
# Regular invocation
if __name__ == "__main__":
    processor = TCPJsonLineProcessor(host='localhost', port=7100)
    try:
        processor.connect()
        processor.process_stream(process_message)
    except KeyboardInterrupt:
        print("Stopping...")
    finally:
        processor.close()
```

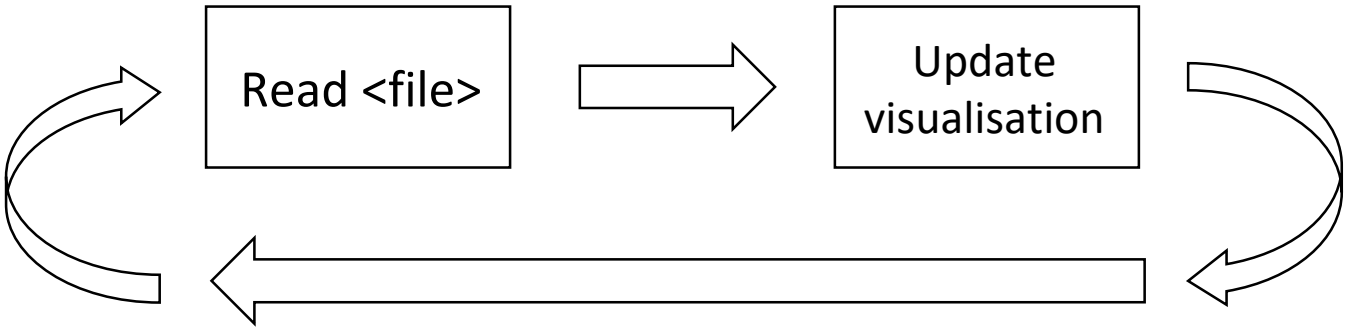
Regular Python Postprocessor Loop

```
def main():
    # ...
    while True:
        try:
            input_message, _ = communication_utils.waitForSocketMessage(server)
        except socket.timeout:
            continue

        # Parse input message
        input_object = communication_utils.parseInferenceResults(input_message)

        # Process the message
        process_message(input_object)
```

Custom Component – Data Visualisation



HTML

```
<html>
  <head>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/luxon@3"></script>
    <script src="https://cdn.jsdelivr.net/npm/chartjs-adapter-luxon@1"></script>
    <script>
        // code here
    </script>
  </head>
  <body>
      <canvas id="graph"></canvas>
  </body>
</html>
```

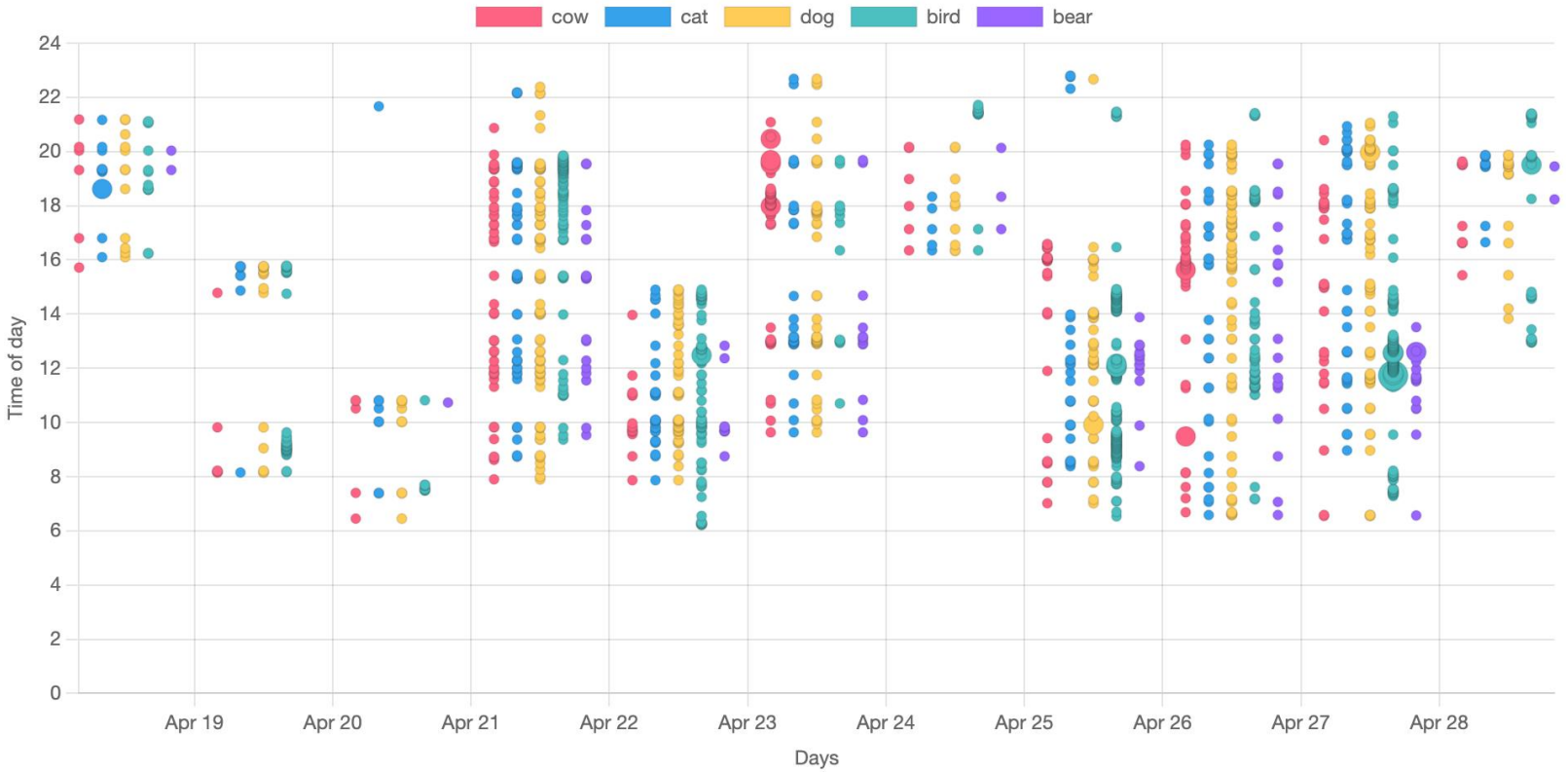
Javascript

```
// detections.jsonl:
//   {"bird": 1, "timestamp": "2025-04-28T21:24:05.000000"}
//   {"cat": 1, "timestamp": "2025-04-28T21:24:15.000000"}
const response = await fetch('detections.jsonl');
const text = await response.text();
const lines = text.split('\n');

// parse JSONL lines and aggregate into whole minutes
const detections = parseAndAggregateLines(lines);

for (const detection of detections) {
    await updateChartDataWithDetection(data, detection);
}
```

Data Visualisation Result



To Recap

- Fast and easy iteration with our setup
- Engineer workarounds or retrain model
- Maintaining hardware resources is not my thing
- My cats are either really scary and/or really big

Resources

Developer resources

https://meta.nxvms.com/

Integration SDK repository

https://github.com/scailable/sclbl-integration-sdk

AI cloud (need Nx account)

https://admin.sclbl.nxvms.com/

2025 Embedded Vision Summit

Visit Network Optix at booth 302

Suggestion for next talk:

“OAAX: One Standard for AI Vision on Any Compute Platform”

Robin van Emden, Thursday, 11:25 am