

Building Heterogeneous Platforms for End-to-end Online Learning Based on Dataflow Computing Design

Benoit Corda, Clément Farabet, Marco Scoffier and Yann Lecun



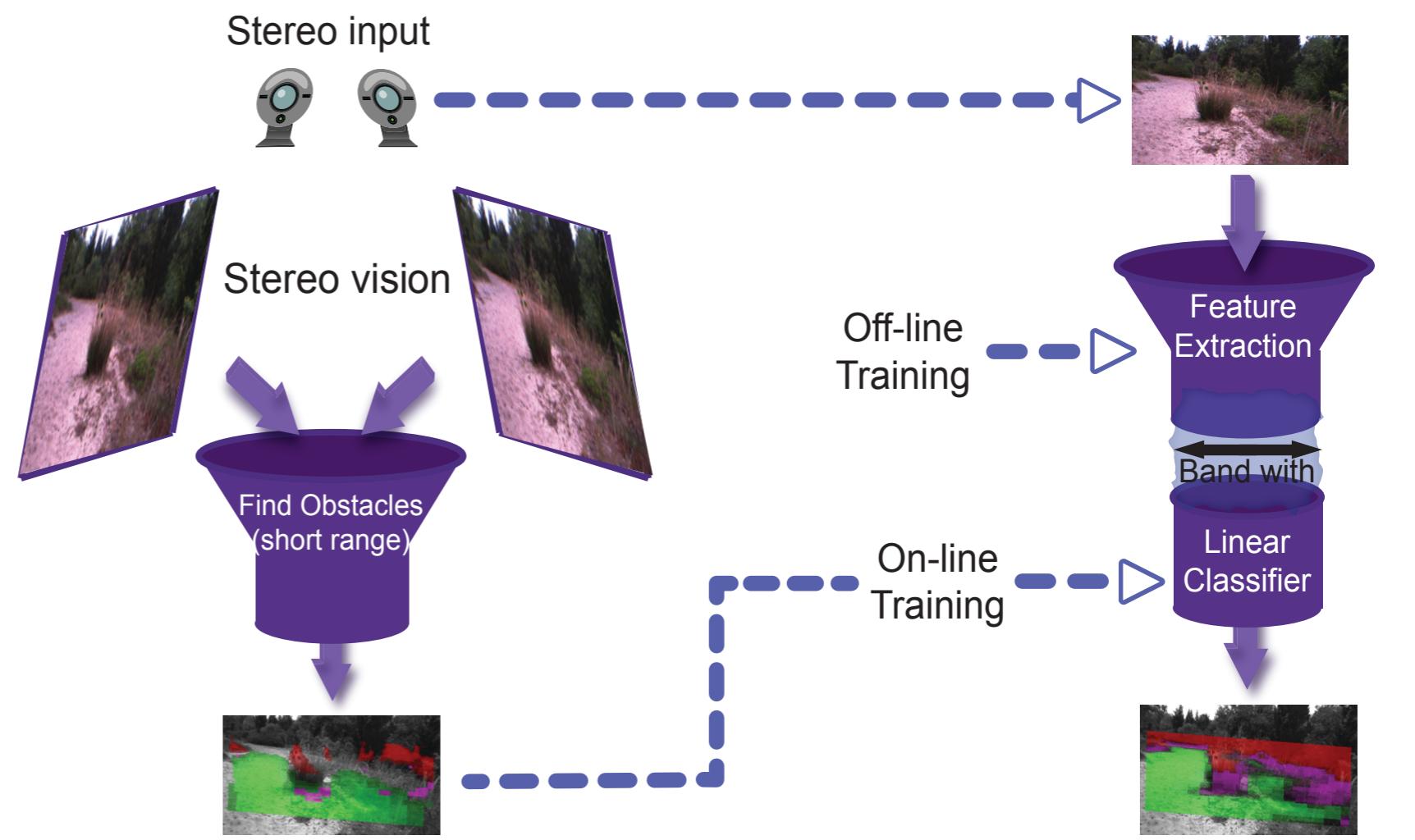
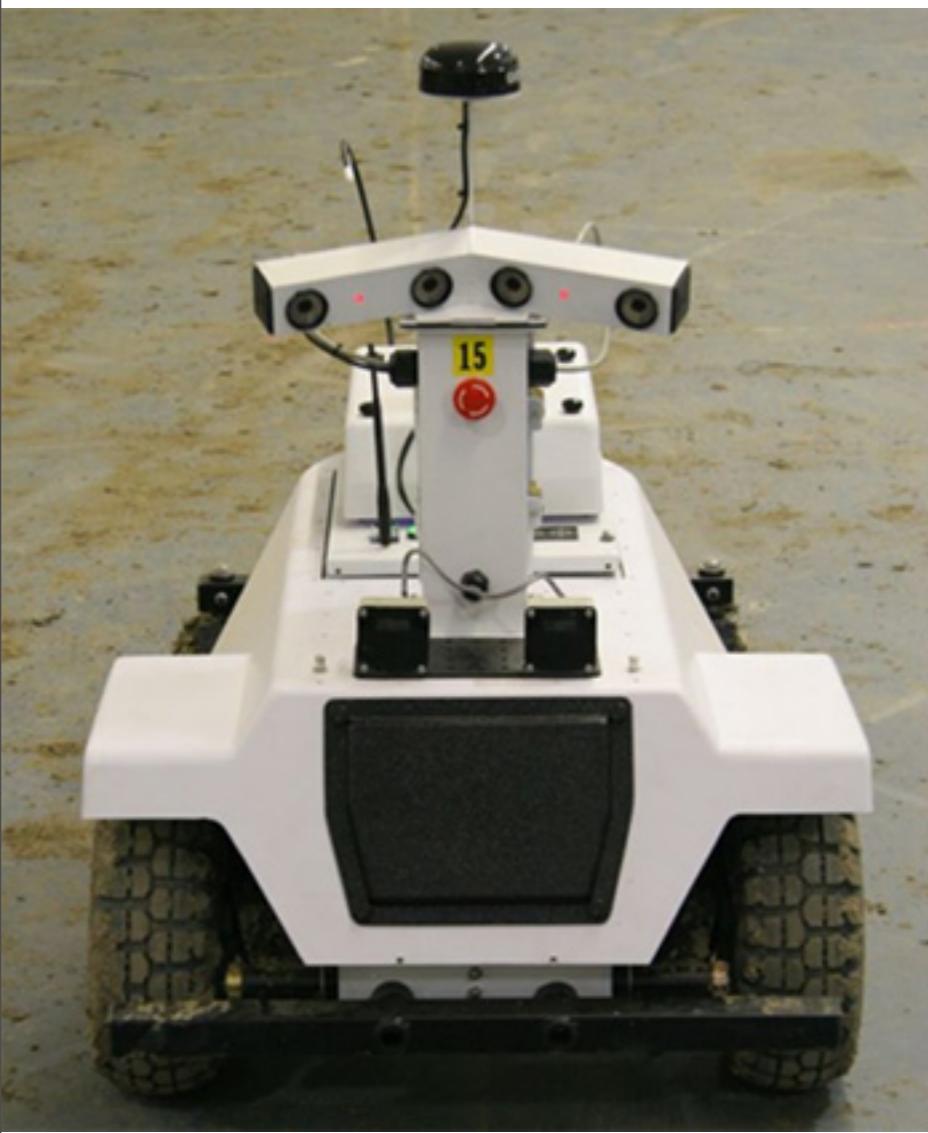
NEW YORK UNIVERSITY

net > SCALE
Technologies, Inc.

Yale University



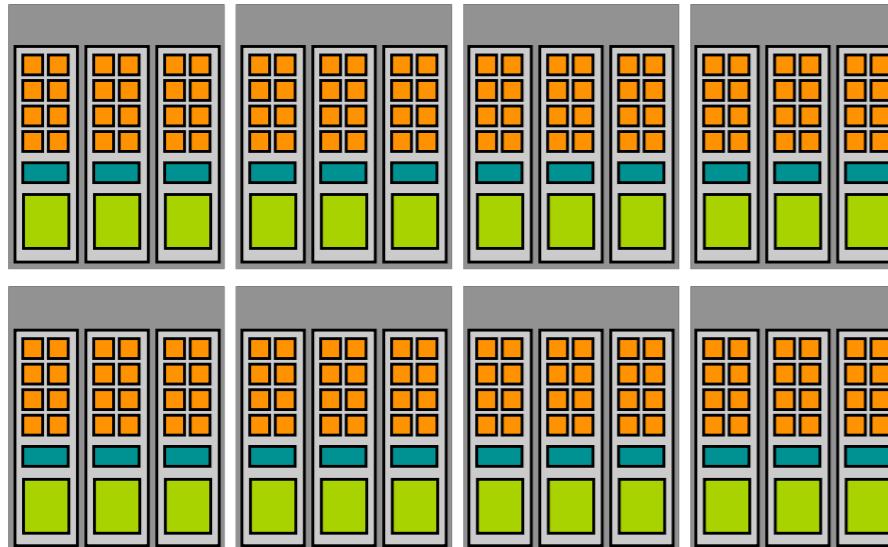
LAGR: a Near-to-far obstacle detection



Inference : 1 frame/sec

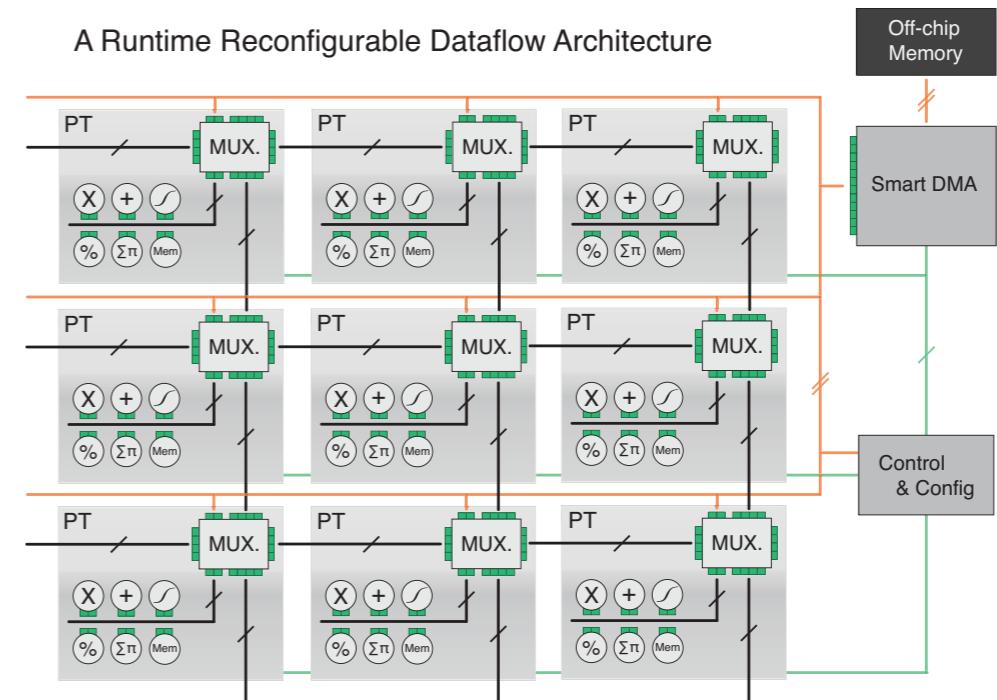
Leveraging stream-processing commodities

GPU Multiprocessors



- Power: 220W
- 294 GOP/sec observed
- Floating point precision

FPGA Compute Grid



- Power: 10W
- 147 GOP/sec observed
- Fixed point precision

Parallelization

- Dataflow design
- Meta programming to generate optimal code
- Interface with CPU abstracted
- Modular design

xFlow: a way to share your models with others

xFlow
Code

```
# set I/Os


# encoder
&encoder := {
    # declare I/Os
    input in = array(1,...)
    output out = array(32,...)
    # internals
    convol_out = array(32,...)
    # a filter bank
    &linear_filter_bank(DIM = 2,
        in = in,
        out = convol_out)
    # a non-linear function
    &math_nn(x = convol_out,
        sigmoid<x> = out)
}

# decoder
&decoder := {
    # declare I/Os
    input in = array(32,...)
    output out = array(1,...)

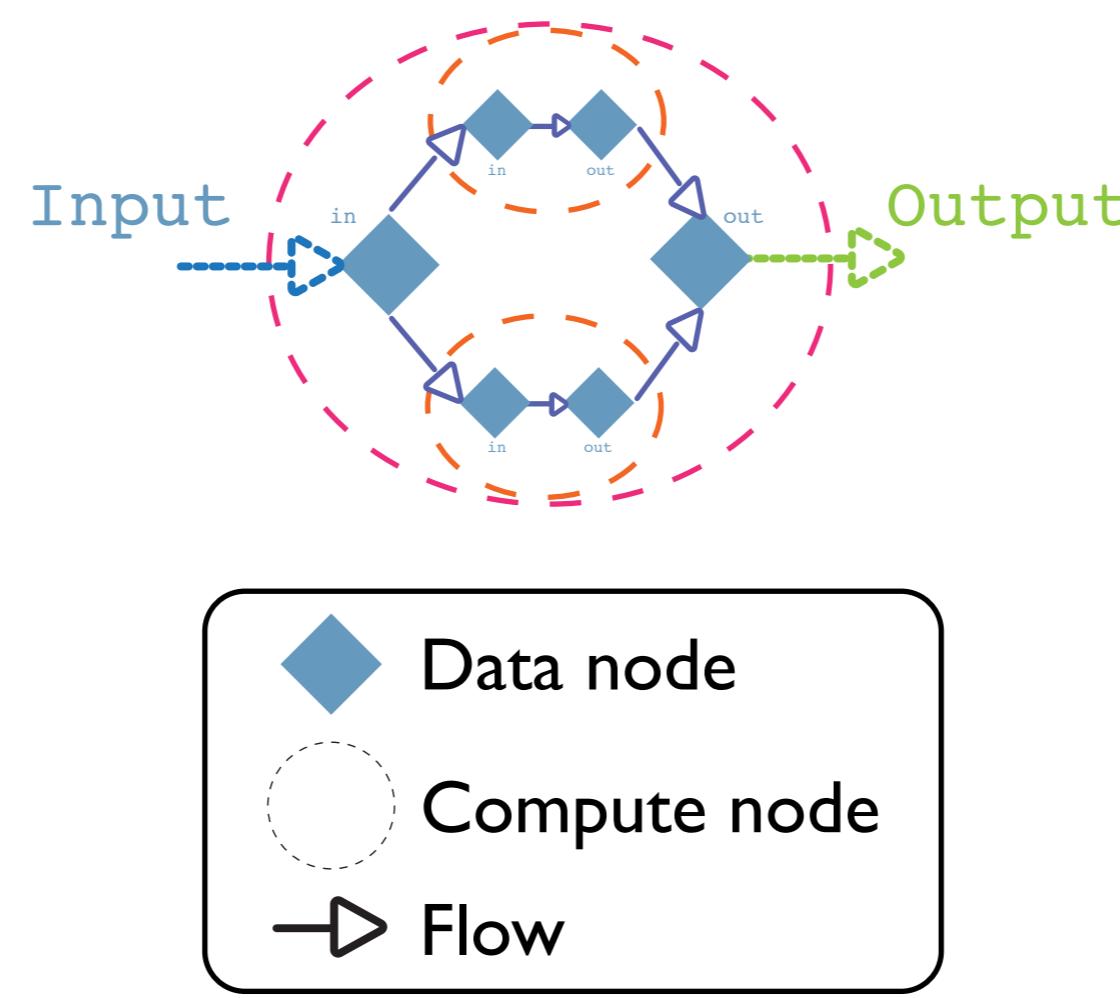
    # a filter bank
    &linear_filter_bank(DIM = 2,
        MODE = "full",
        in = in,
        out = out)
}

# instantiate encoder
output encoder_out = array(32,...)
&encoder(in = in1, out = encoder_out)

# copy output
code = array(32,...)
&flow(x = encoder_out, copy<x> = code)

# instantiate decoder
&decoder(in = code, out = decoder_out)
```

Dependency
Graph



Architecture
implementation

