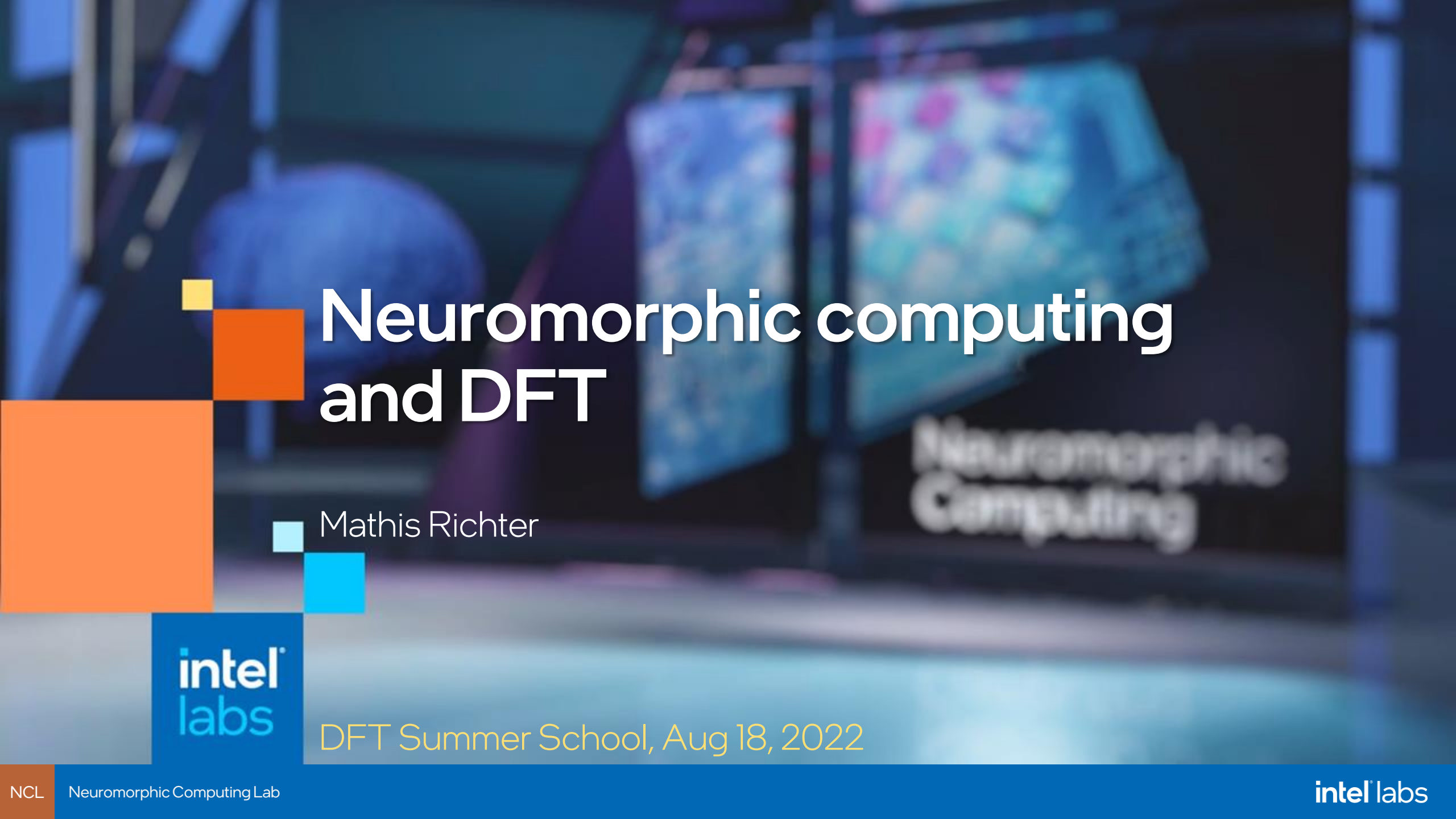


Neuromorphic computing and DFT

Mathis Richter

DFT Summer School, Aug 18, 2022

Legal Information

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.

Your costs and results may vary.

Results have been estimated or simulated.

Intel technologies may require enabled hardware, software or service activation.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Applications of DFT

- Fundamental research: How does it work?
- Application oriented research: Can we use it to solve problems?

- How can we use neural dynamics and DFT to solve problems?
- What is the most fitting problem domain?
  - Autonomous robotics

Autonomous robotics

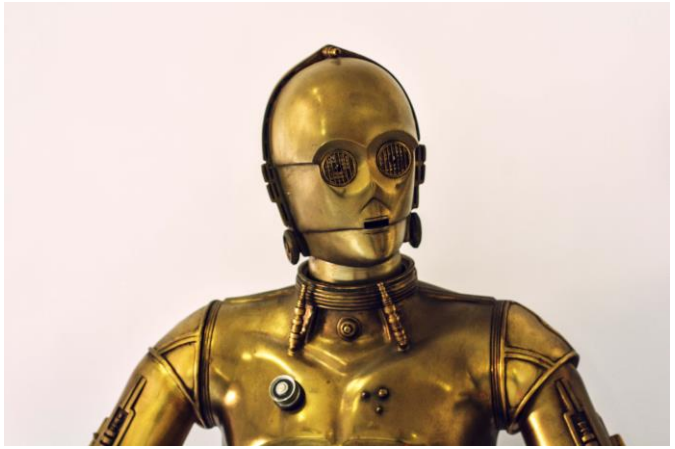


Photo by Lyman Hansel Gerona on Unsplash

Photo by Jason Blackeye on Unsplash

Photo by Roberto Nickson on Unsplash

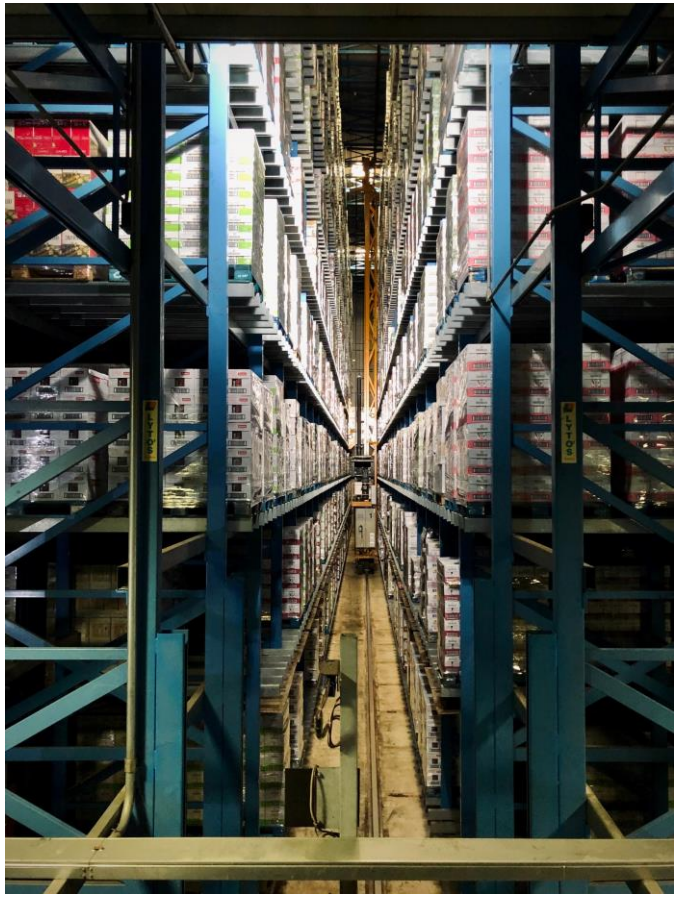


Photo by Arno Senoner on Unsplash

Challenges to applications in robotics

- Many (many) open questions
  - Perception
  - Abstraction
  - Learning
  - Autonomy
- On the implementation level
  - *Size, weight*
  - *Power*
  - *Speed*

Mismatch between hardware and neural networks

- CPUs
  - Few (4-8)
  - Fast (2,300,000,000,000 instructions per second)
  - Medium functionality (~100 instructions)
  - Random access memory (access any information)
- Neurons
  - Many (~86 billion)
  - Slow (10 ms time scale)
  - Minimal functionality (leak, integrate, fire)
  - Memory “in place” (only access to local information)

Neuromorphic computing

- Fundamentally rethink hardware based on findings from biology and neuroscience
  - Energy efficiency
  - Speed
  - Novel functionality / application areas
- Heterogeneous community
- Beginning in the late 1980s with sensors, later processors
- Today: Toward neuromorphic algorithms and applications

Neuromorphic sensors

- Vision: Dynamic vision sensor (DVS)
  - Sparse data
  - High temporal resolution
  - High dynamic range

- Audio: Silicon cochlea
- Olfaction
- Haptics

https://inivation.com/products/customsolutions/videos/

Neuromorphic processors

- Different designs
  - Industry (Intel Loihi, IBM TrueNorth)
  - Academic labs (e.g., Manchester (SpiNNaker), INI Zurich (ROLLS), Heidelberg (BrainScaleS)
  - Start ups (SynSense, BrainChip, Innatera, GrAI matter labs)
- Different technologies
  - Digital CMOS
  - Mixed signal CMOS
  - New materials

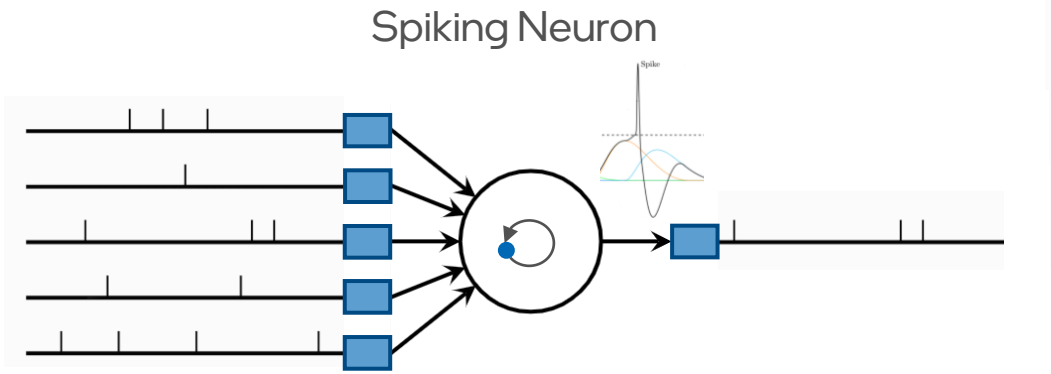
Core principles

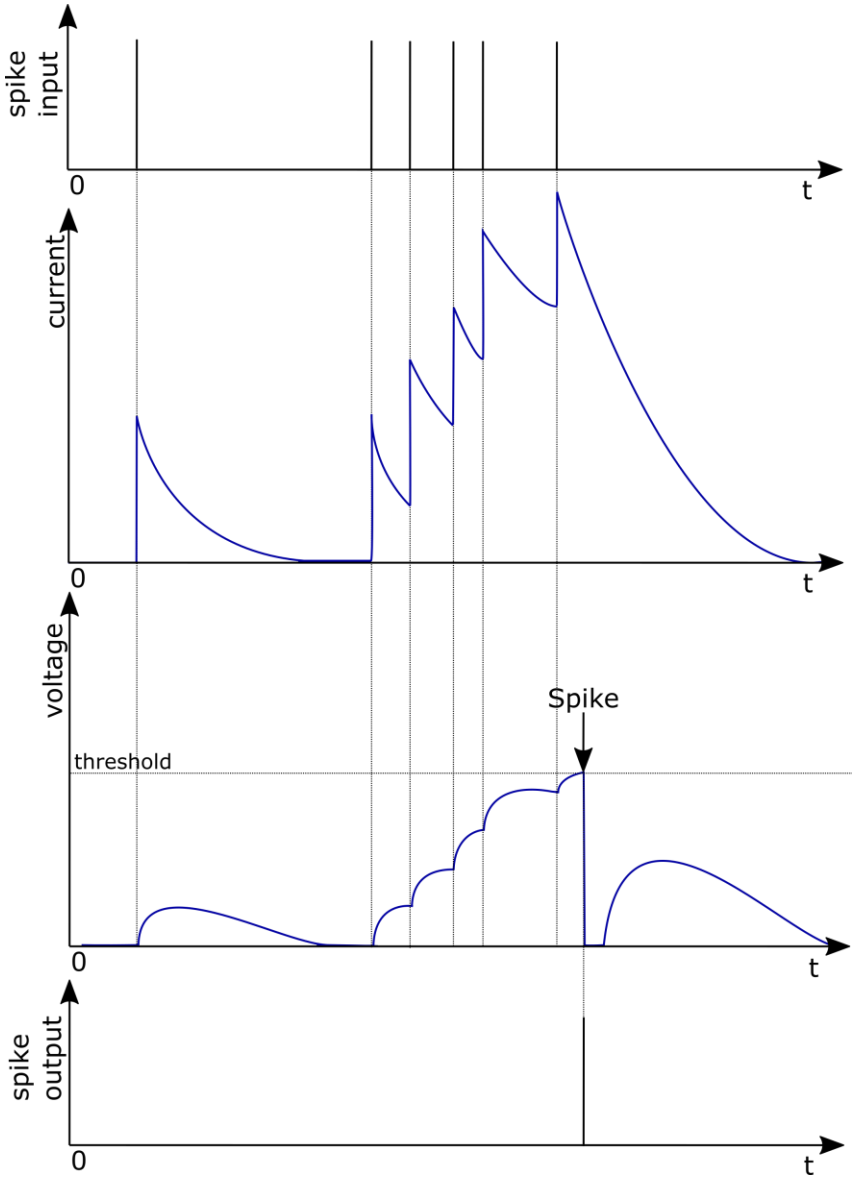
- Event-based communication
  - Only communicate “change”
  - Sparse communication
  - -> Energy efficiency
- Massive parallelism
  - Dedicated circuitry for every neuron/pixel
  - -> Speed
- Function from connectivity
  - Minimal units (neurons, pixels)
  - Overall function emerges from connectivity of units
  - Modularity
  - Recurrence
- Learning
  - Synaptic plasticity
  - Local learning rules

Spiking neurons

- Leaky integrate-and-fire (LIF) neuron

$$\dot{v}_i(t) = -\frac{1}{\tau_v} v_i(t) + I_i(t) - \theta_i \sigma_i(t) + b_i$$





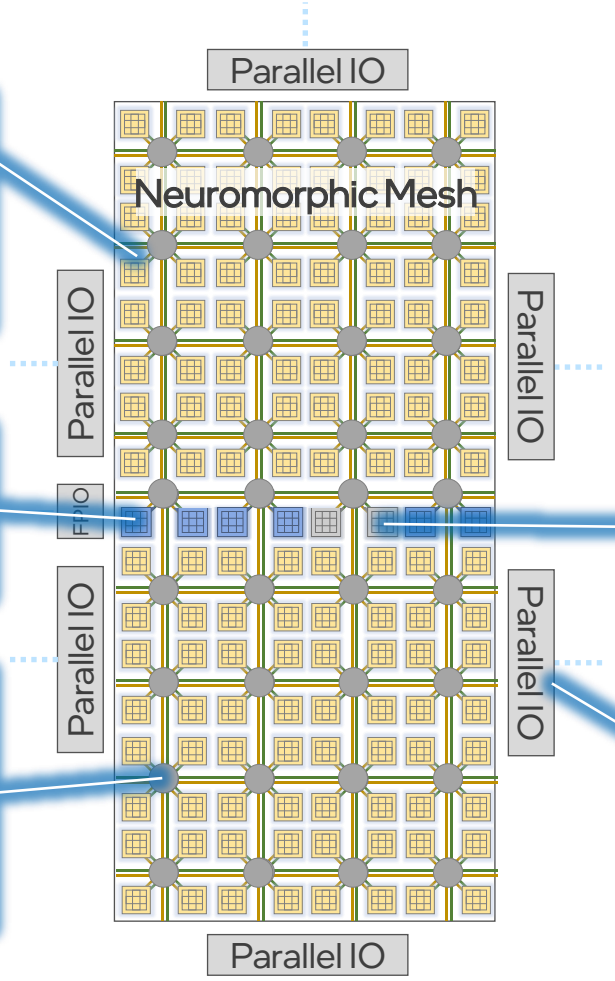
Loihi 2 chip architecture

Core and tile based architecture

128x Neuromorphic core
Asynchronous design
Programmable neuron model
Programmable learning
Up to 128kB synaptic memory
Up to 8192 neurons

6x Embedded Processor cores
Efficient spike-based communication
Data encoding/decoding
Network configuration

Low overhead NoC fabric
8x16-core 2D mesh
Scalable to 1000s of cores
Dimension order routed
Two physical fabrics
Acceleration for handshaking between cores

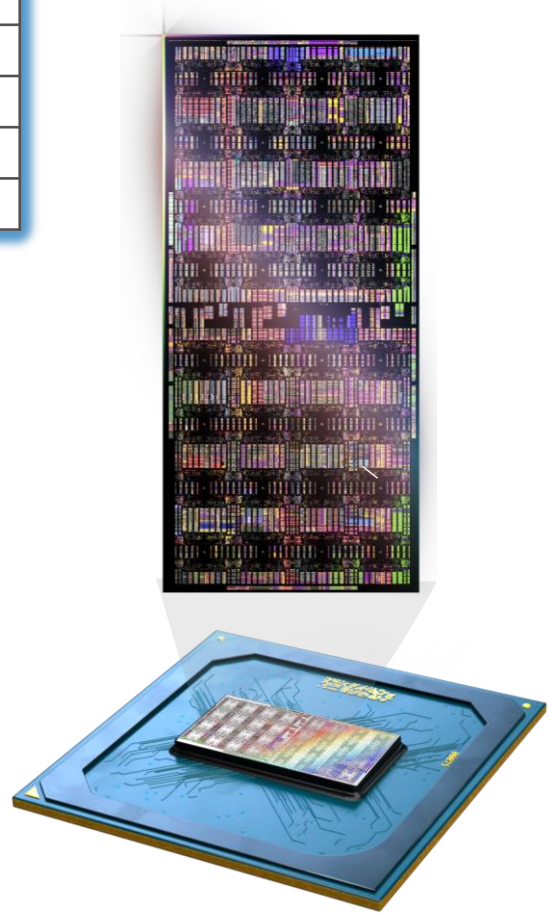


2x IO cores
Asynchronous design
Handle communication between chips and IO to external devices
On-chip fanout of remote spikes

6x Parallel off-chip interfaces
Faster chip-to-chip links
3D scaling
Support for standard synchronous protocols and event-based vision sensors

Technology:	Intel 4
Die Area:	31 mm^2
NmC cores:	128 cores
x86 cores:	6 cores
Max # neurons:	1M neurons
Max # synapses:	120M synapses
Transistors:	2.3 billion

31 mm^2 in Intel 4



Loihi 1 systems

Pohoiki Springs

768 chips

Kapoho Bay

2 chips

AER interfaces

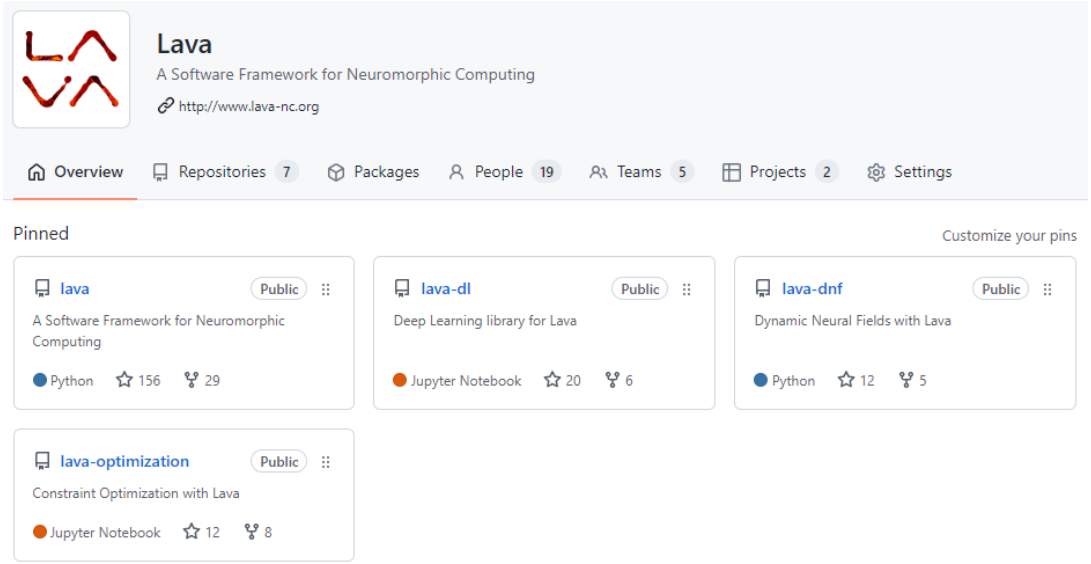
Nahuku

32 chips

Arria 10 FPGA

Lava

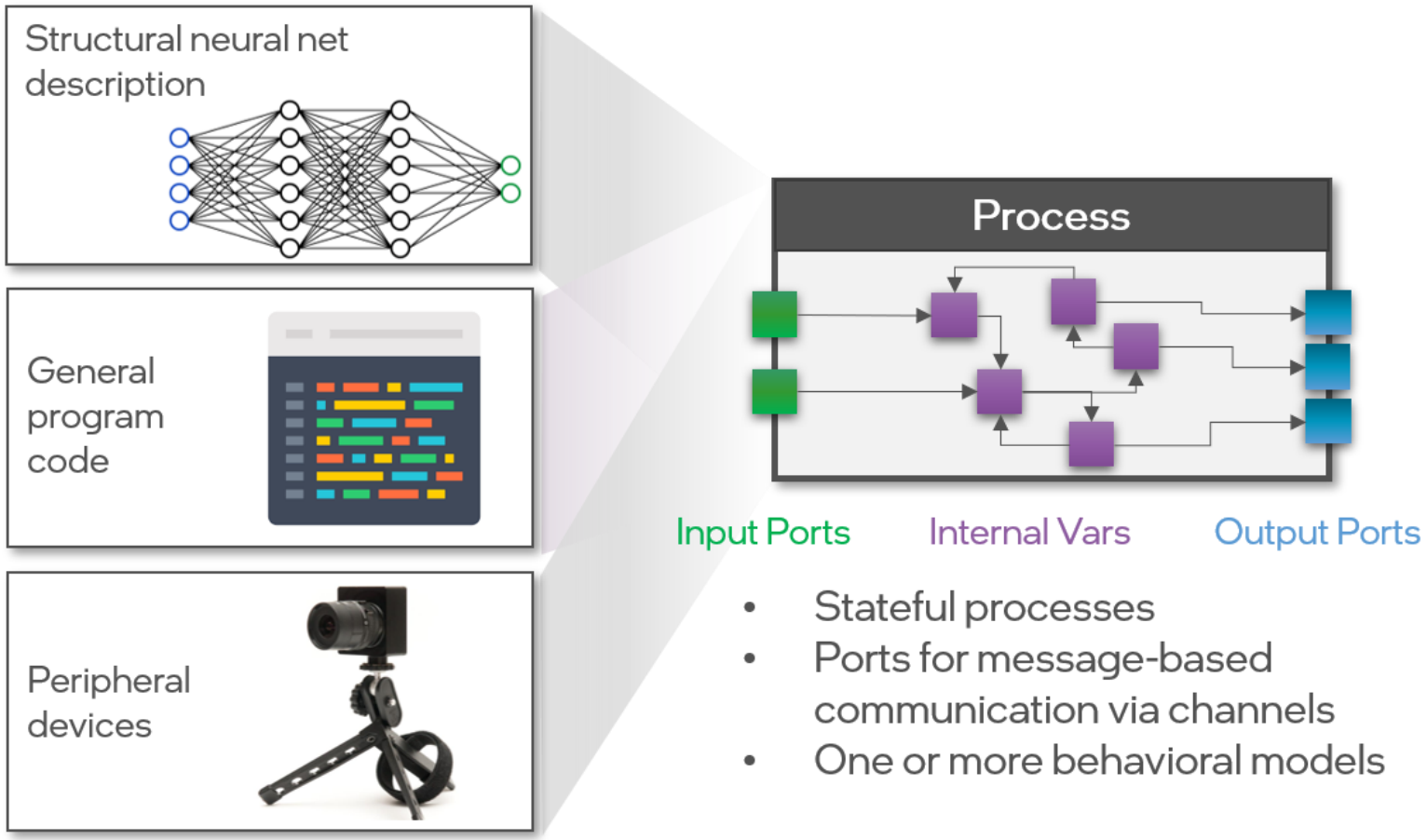
- Open source software framework for neuromorphic computing
- Spiking neural networks and middleware (agnostic of compute)
- Supports execution on Loihi 2 and CPU (simulation)
- Python-based



https://lava-nc.org

https://github.com/lava-nc

Lava Processes

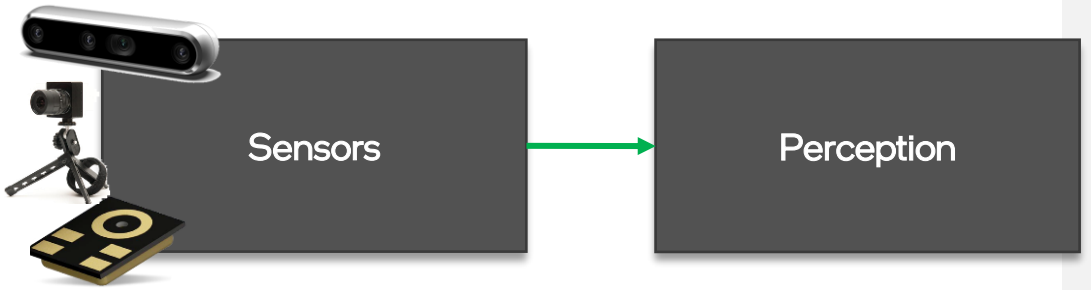


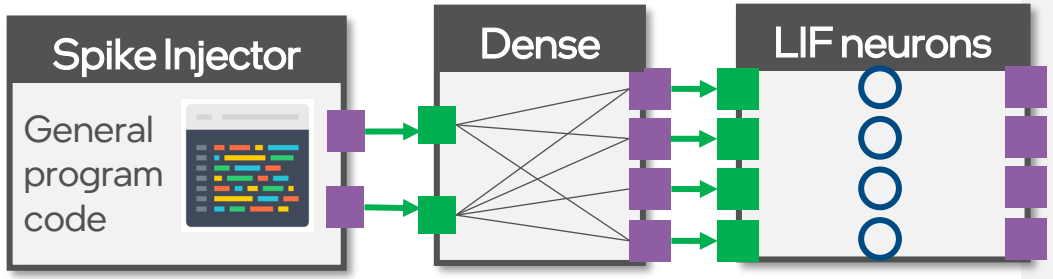
- Stateful processes
- Ports for message-based communication via channels
- One or more behavioral models

Application Development

A simple canonical example

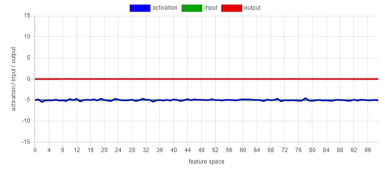
```
# Import processes from any library
from lava.proc.io.input import SpikeInjector
from lava.proc.dense.process import Dense
from lava.proc.lif import Lif

# Initialize processes via costum API with specific parameters
source = SpikeInjector(...<params>...)
dense = Dense(...<params>...)
lif = LIF(...<params>...)

# Connect processes via directional input/output ports
source.spks_out.connect(dense.spks_in)
dense.spk_out.connect(lif.spks_in)

# Helper configuration class
from lava.magma.core.run_conditions import RunSteps
from lava.magma.core.run_configs import Loihi2HwCfg

# Compile and execute model on Loihi 2 for 100 time steps
lif.run(condition=RunSteps(num_steps=100),
        run_cfg=Loihi2HwCfg())

# Terminate
lif.stop()
```

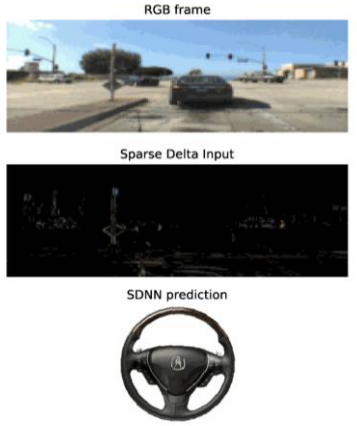




Lava libraries

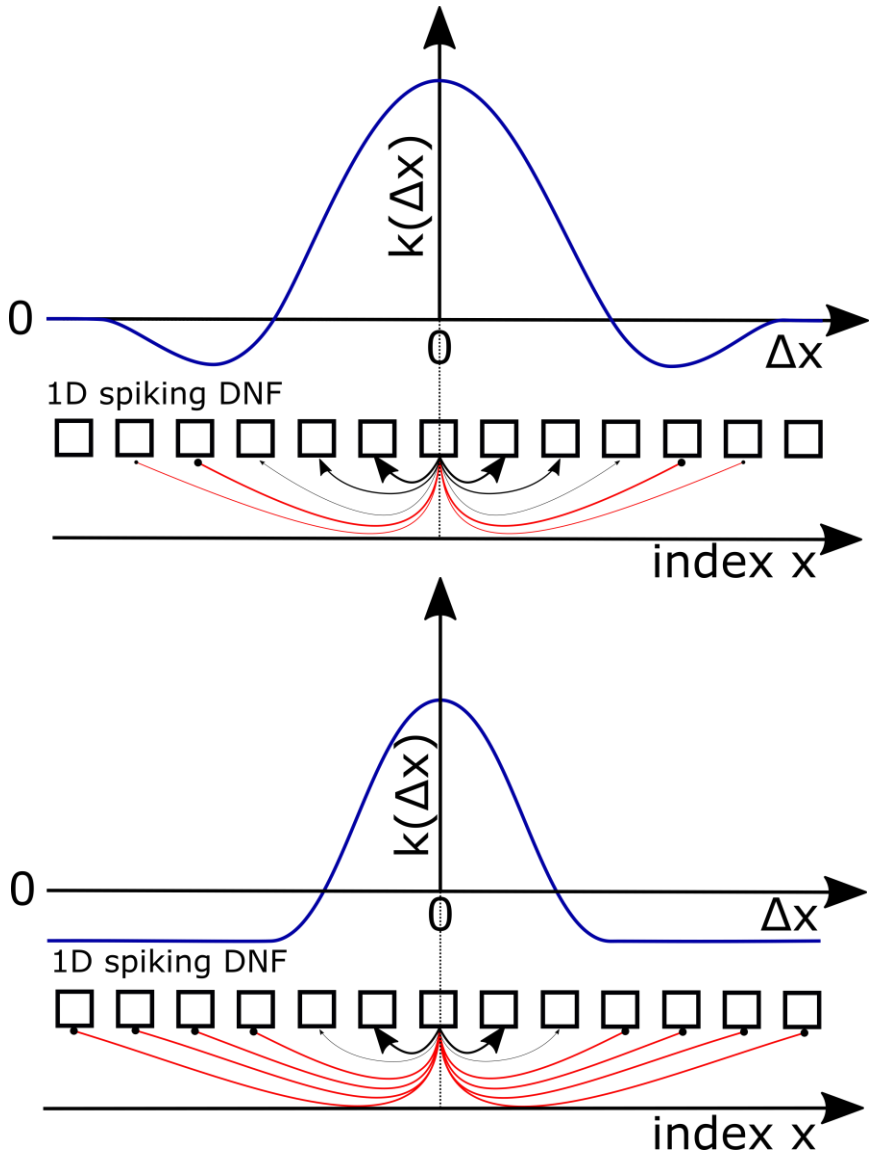
- Lava-dnf
  - Design neural populations with attractor states
  - Build architectures using the attractor modules
- Lava-dl
  - Deep learning tools for event-based networks
  - Offline training
    - For direct training rich set of neuron dynamics
    - For accelerated rate coded training of binary SNNs
  - Inference
    - On various backends using Lava
- Lava-optim
  - Modeling and solving optimization problems with SNNs





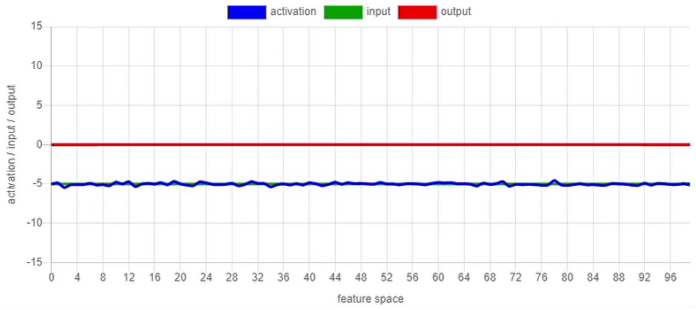
Neuromorphic DFT

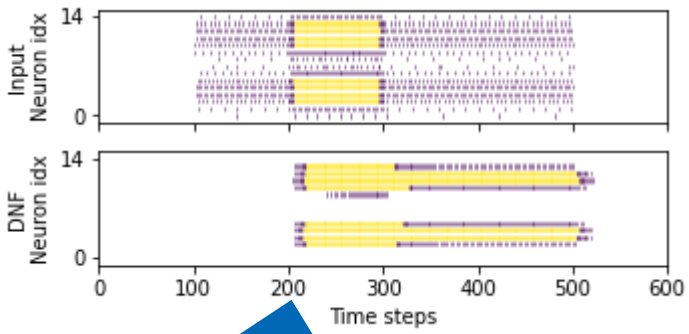
- DNF dimensions are sampled with LIF neurons
- Connectivity strength is sampled from continuous kernel function



DFT instabilities

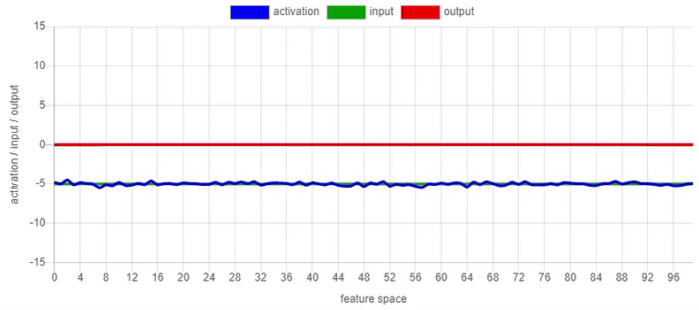
Detection

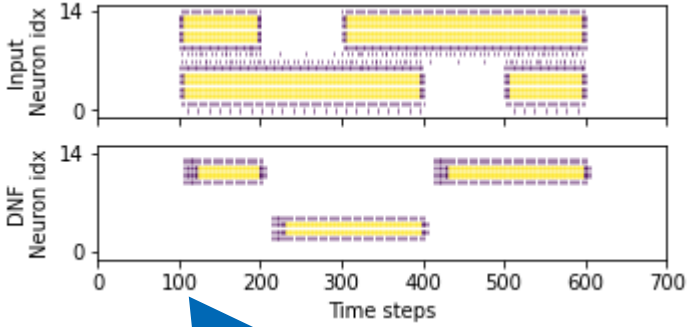




Stabilized detection decision

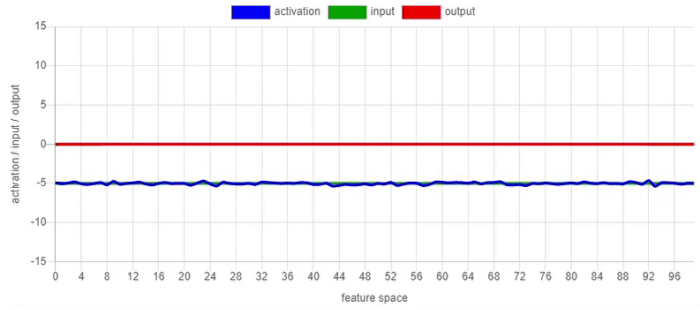
Selection

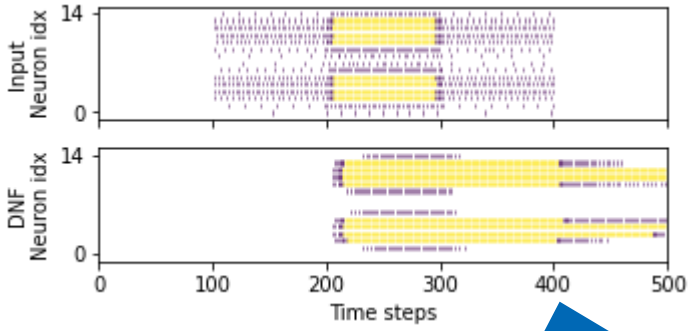




Stabilized selection decision

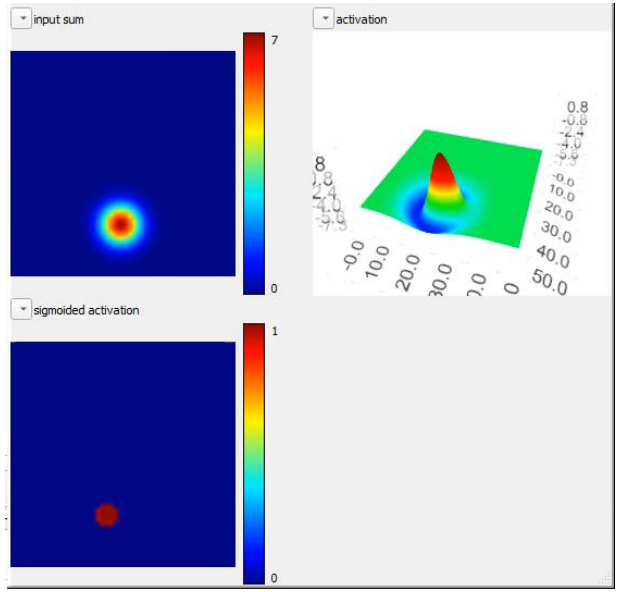
Memory

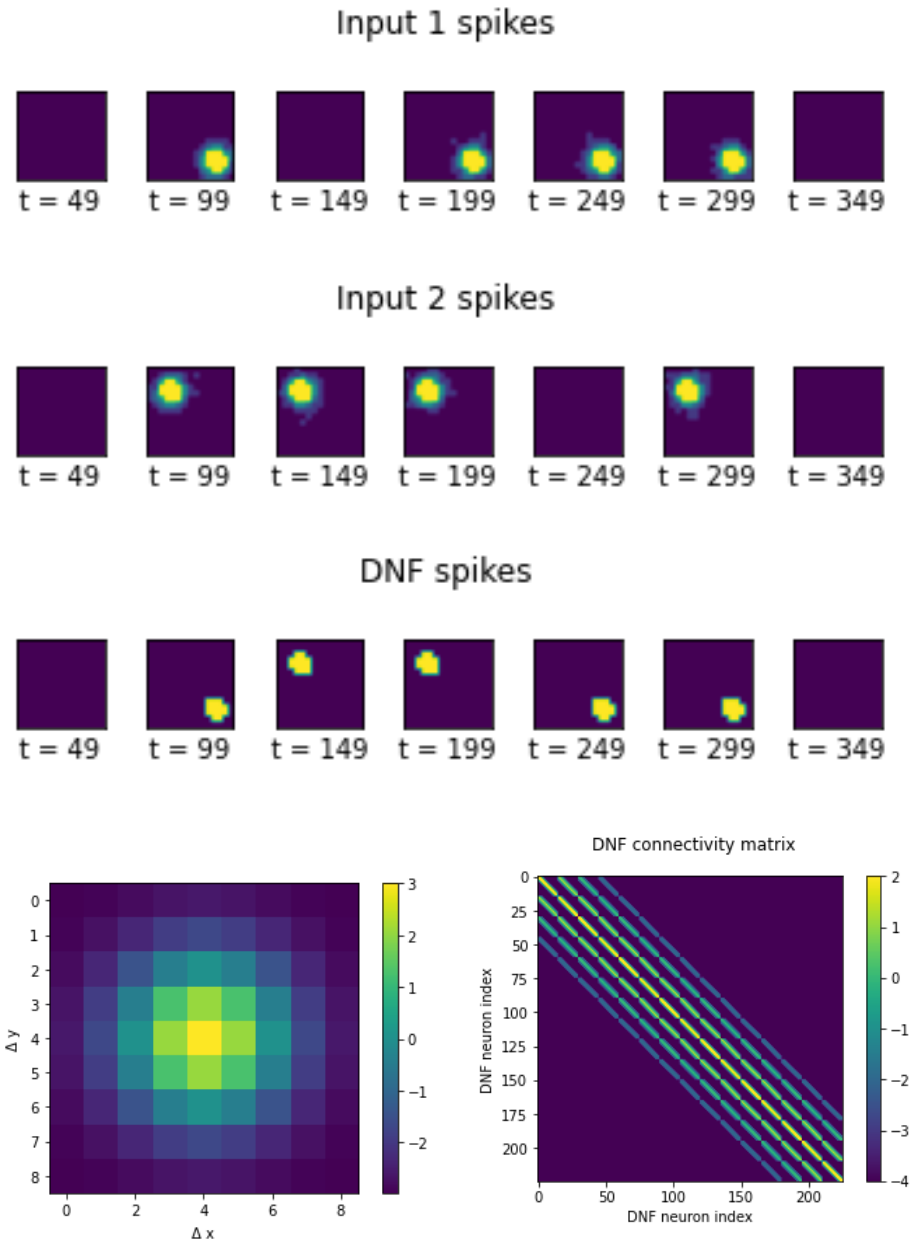




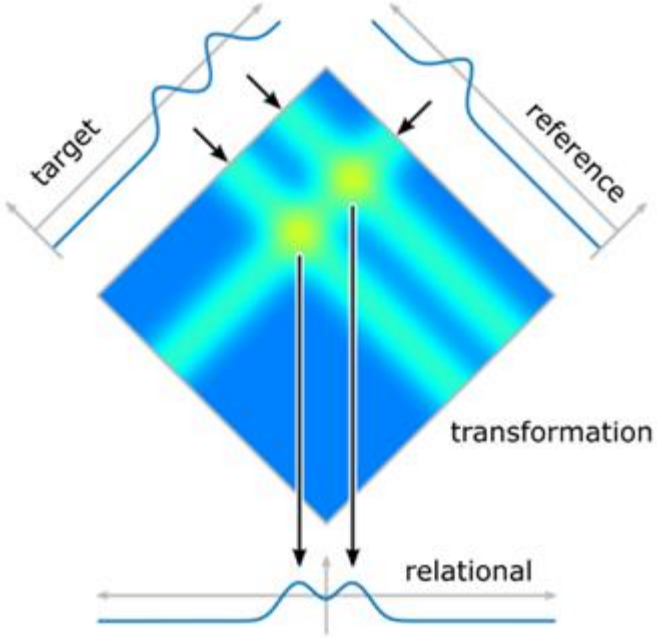
Working memory representation

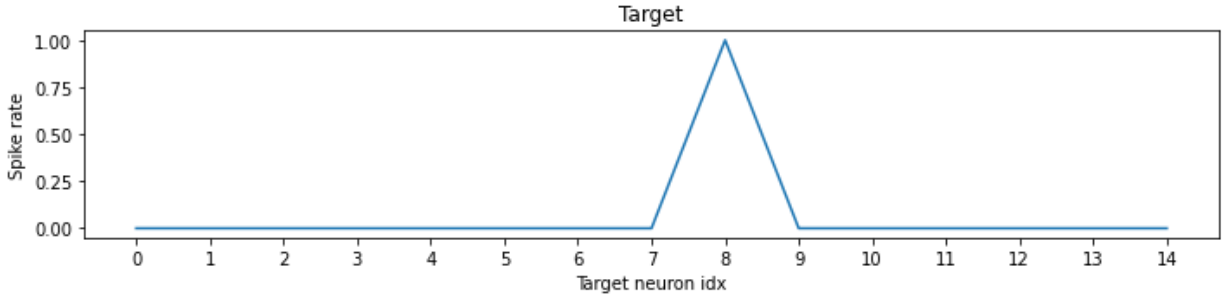
Higher dimensional DNFs

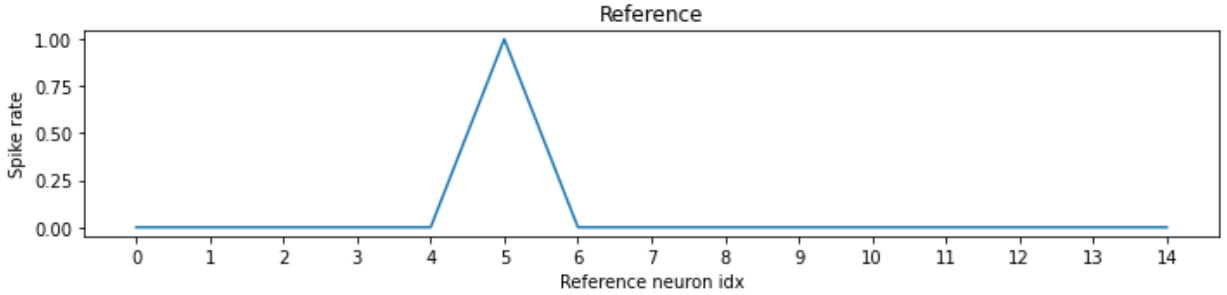


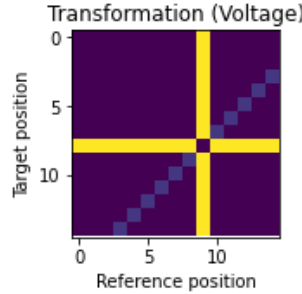


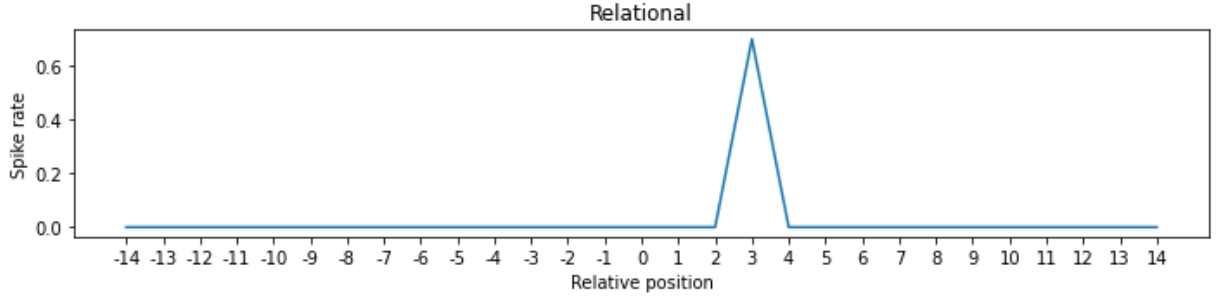
Coordinate transform





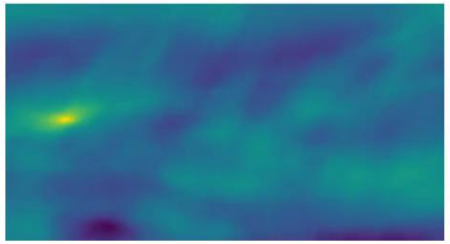






Demo: Stabilizing object tracking on Loihi

Input

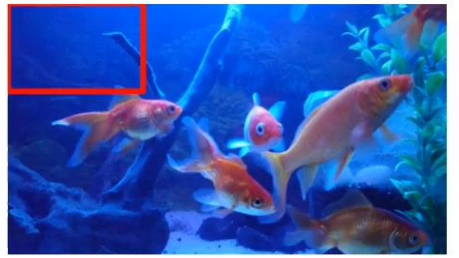


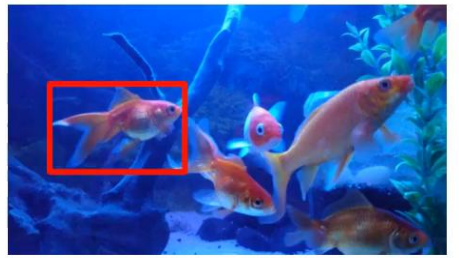
Salience
(input bias)

State
(voltage)

Output
(spikes)

Smooth
tracking

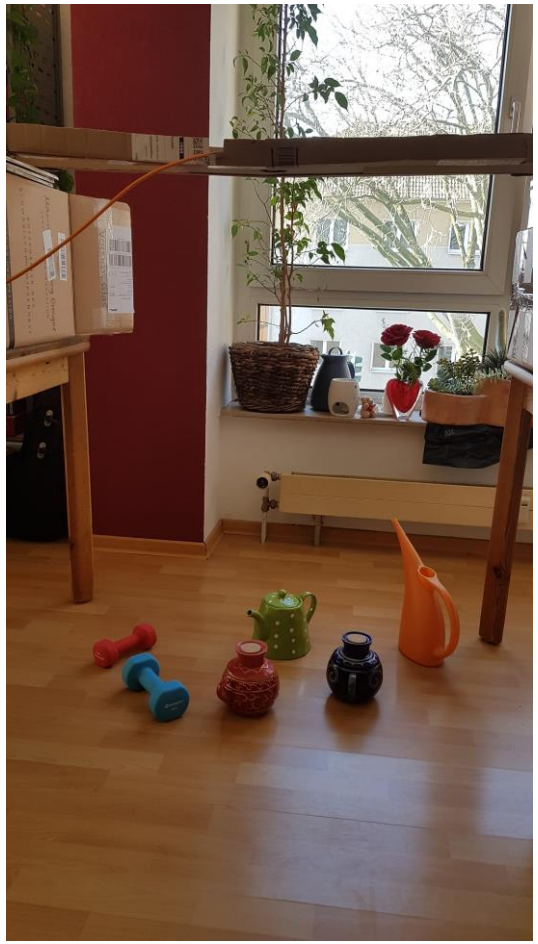


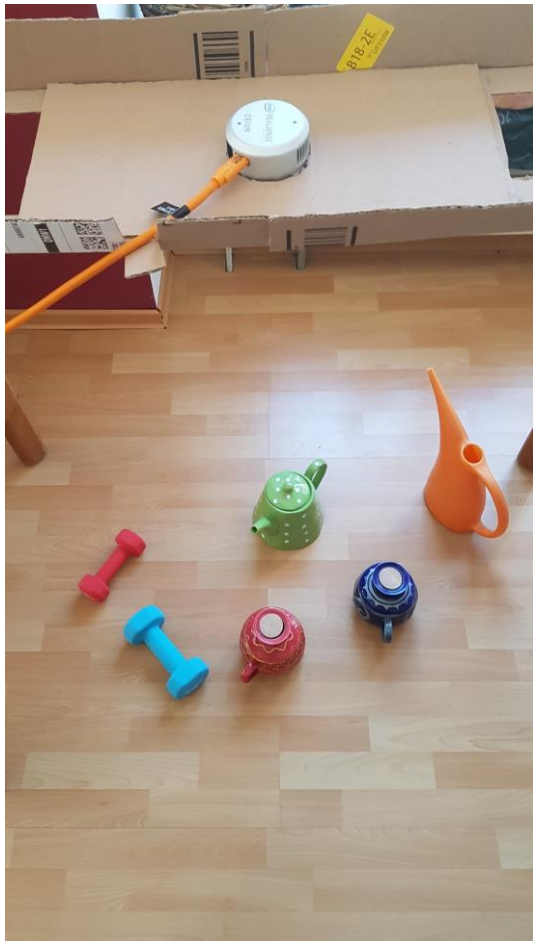


Baseline

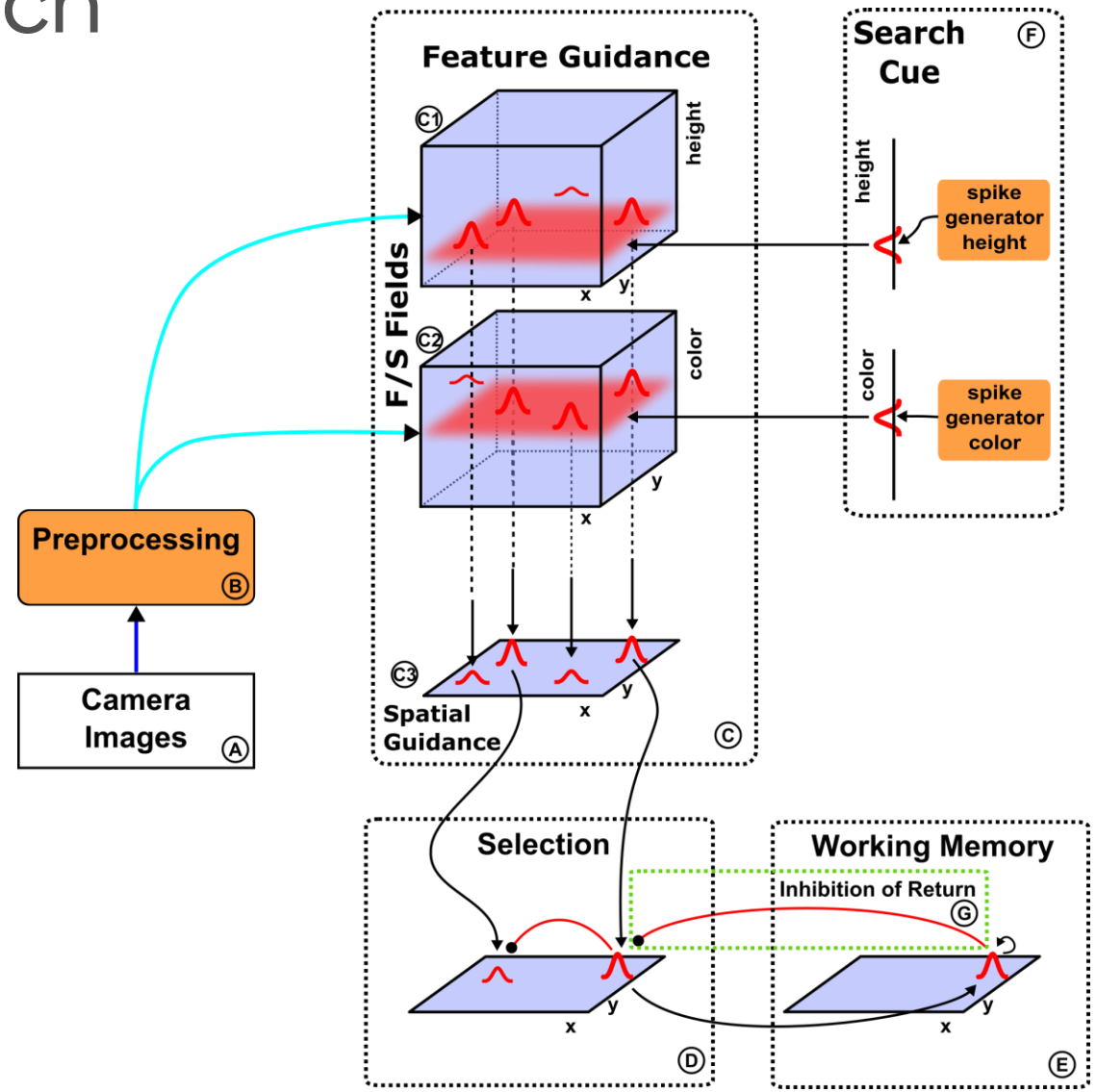
Visual search - Setup

Visual search - Setup



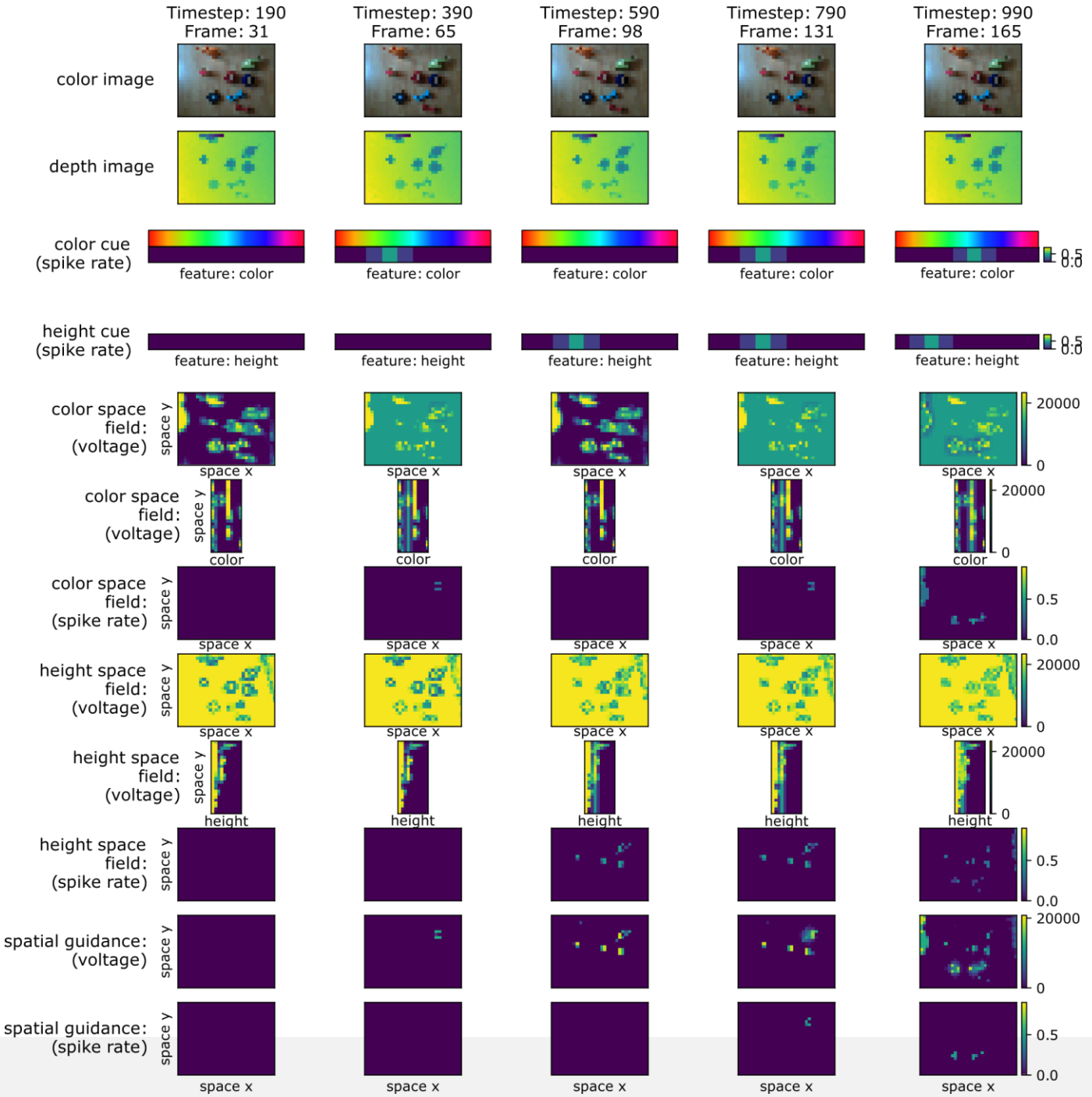


Visual search

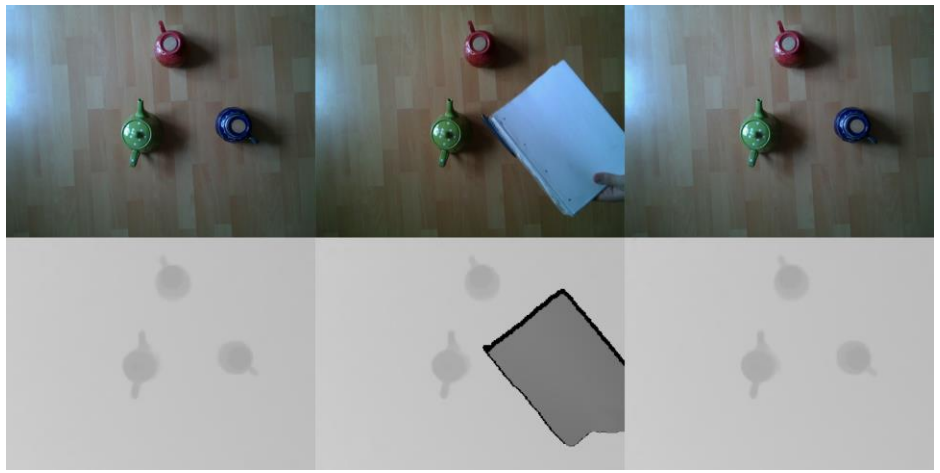


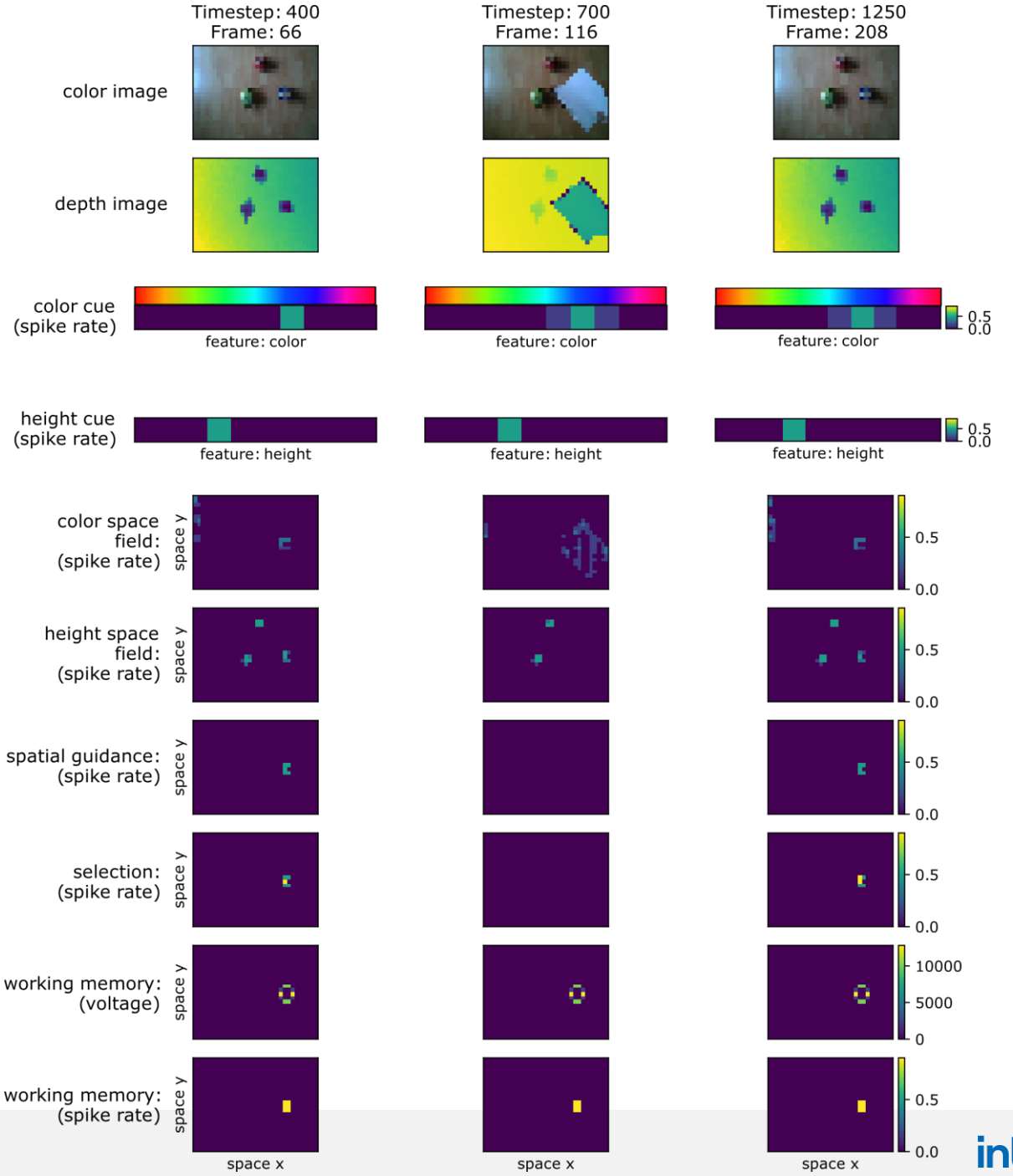
Master thesis Lennart Keil, 2022

Visual search for features

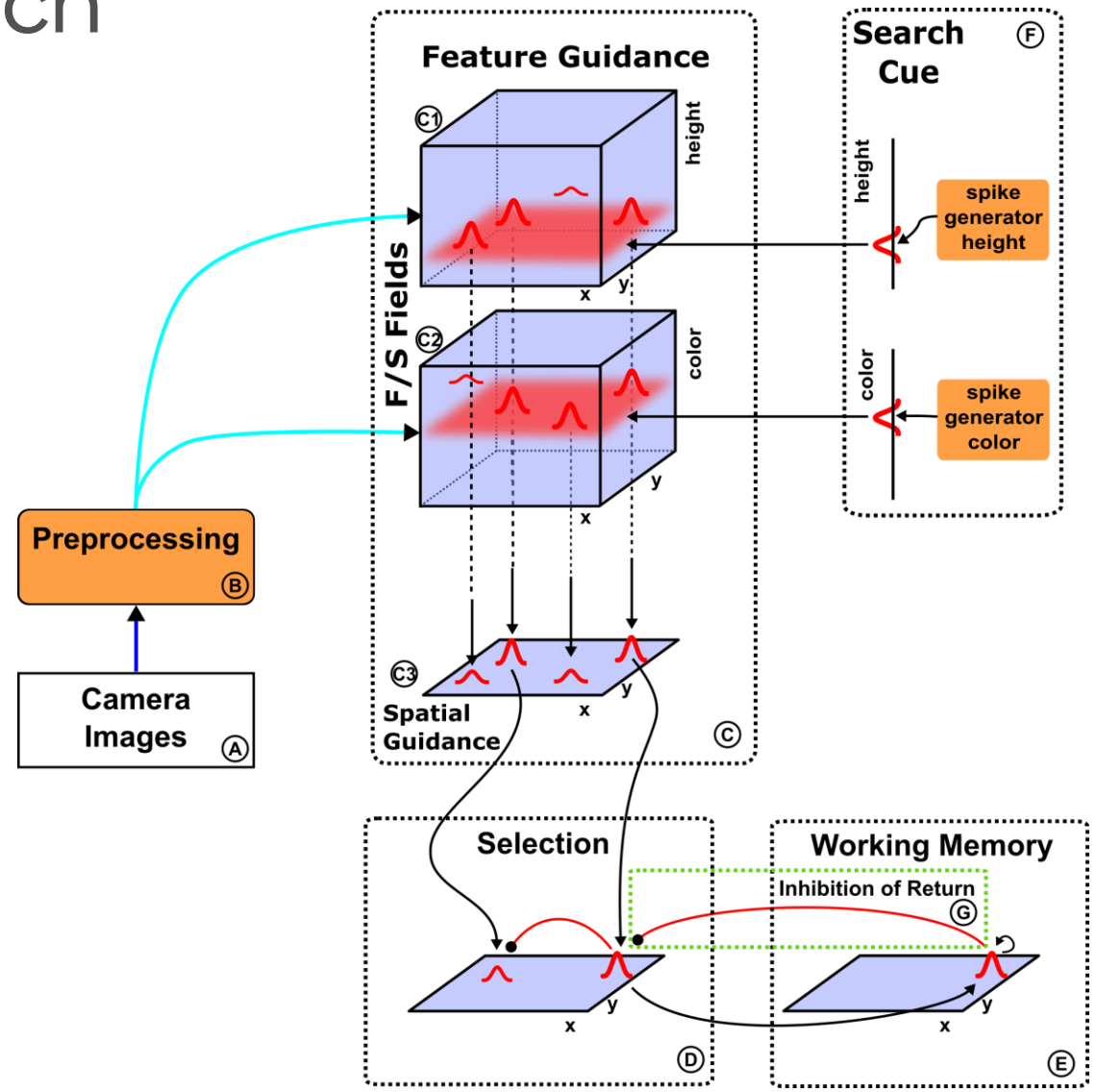


Visual search with working memory





Visual search



All spatial dimensions: 32x24 neurons

Color/Height feature dimension: 10 neurons

Kernel size F/S fields: 5x5x1

Entire architecture:

17,684 neurons

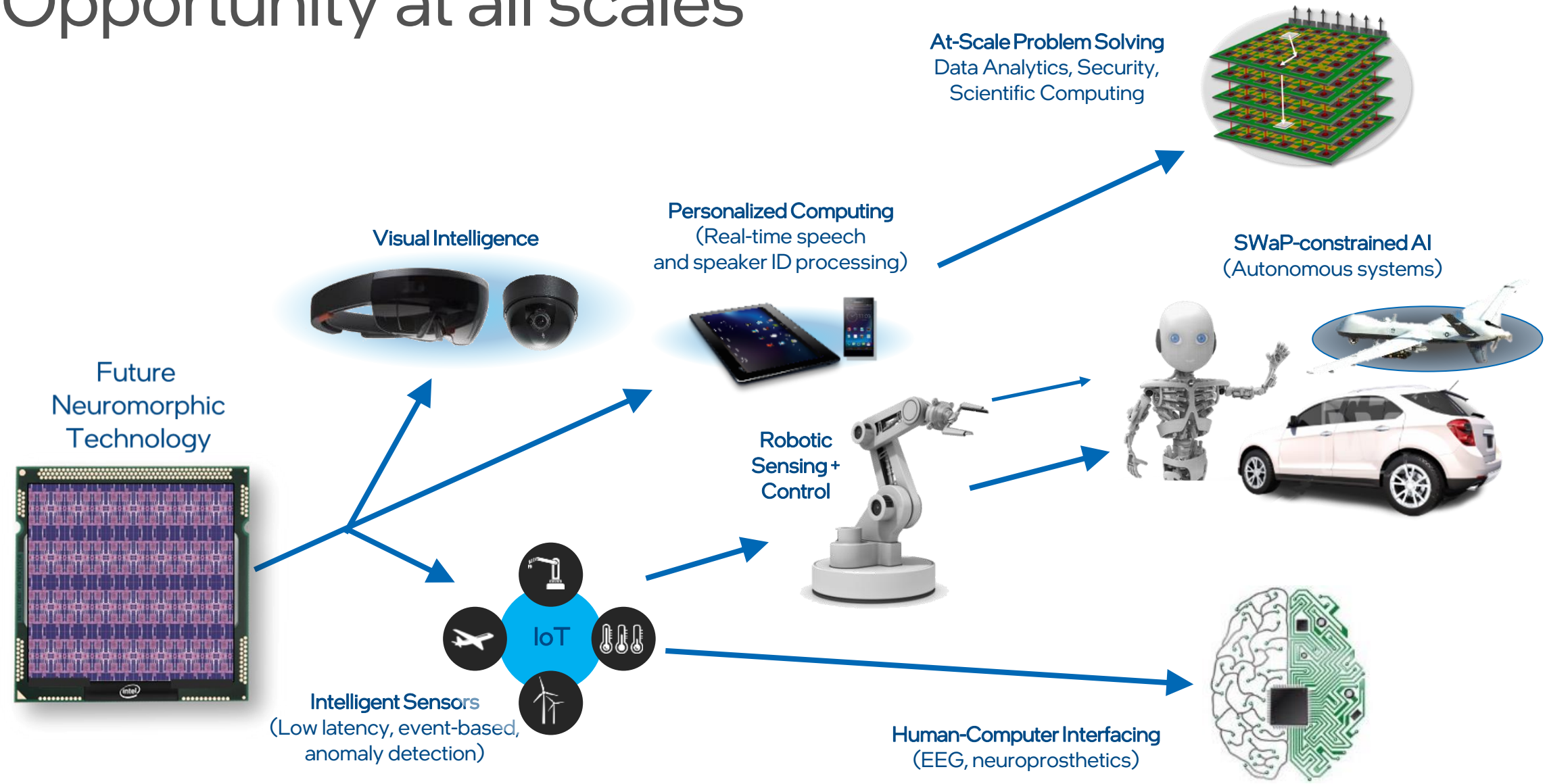
(Scaled to 320x240: 1.7 million neurons)

Master thesis Lennart Keil, 2022

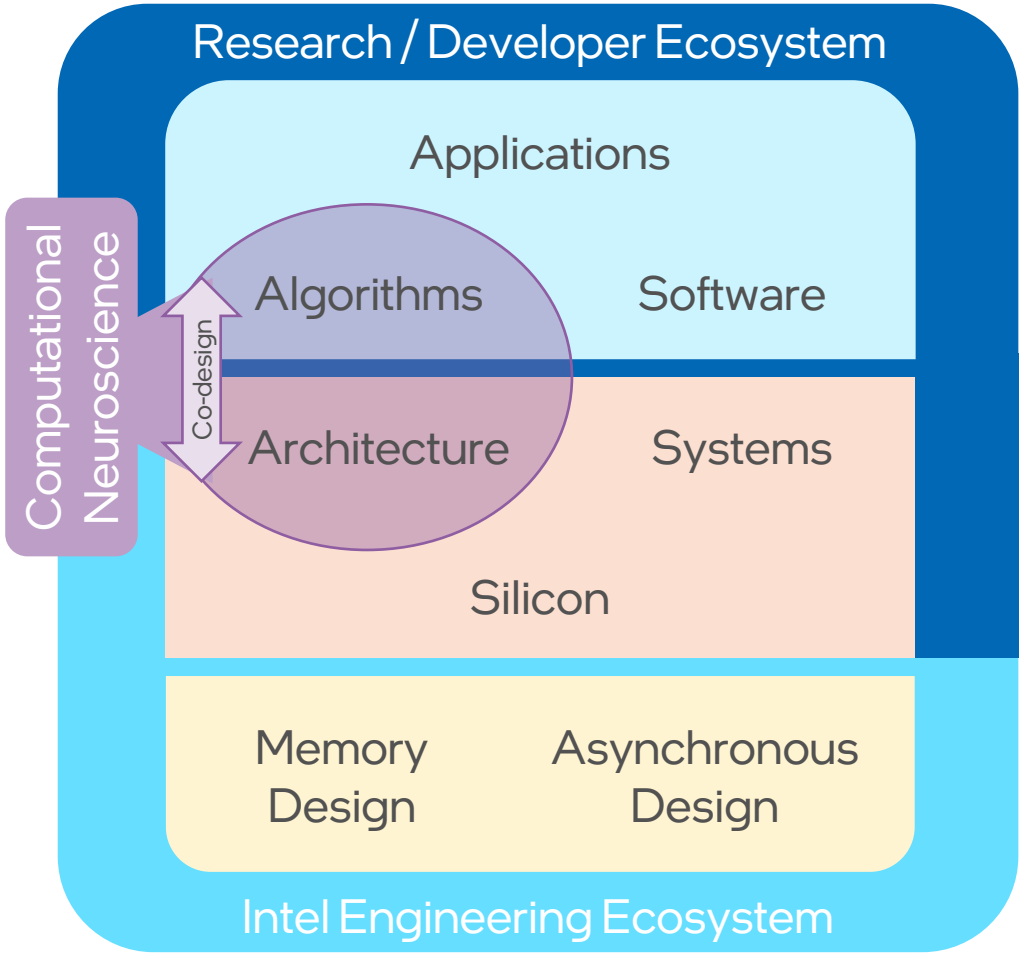
Outlook

- Make DNF implementation more efficient
  - Possibly find a replacement for rate-code
  - Optimize the number of neurons and connections
    - In particular for coordinate transforms and higher dimensional DNFs
- Combine DNFs with powerful perception front-end
  - CNNs for object recognition
- Incorporate learning into DFT elements
- Make available through lava-dnf
- Build architectures for autonomous robots

Opportunity at all scales



Building the Neuromorphic Computing ecosystem

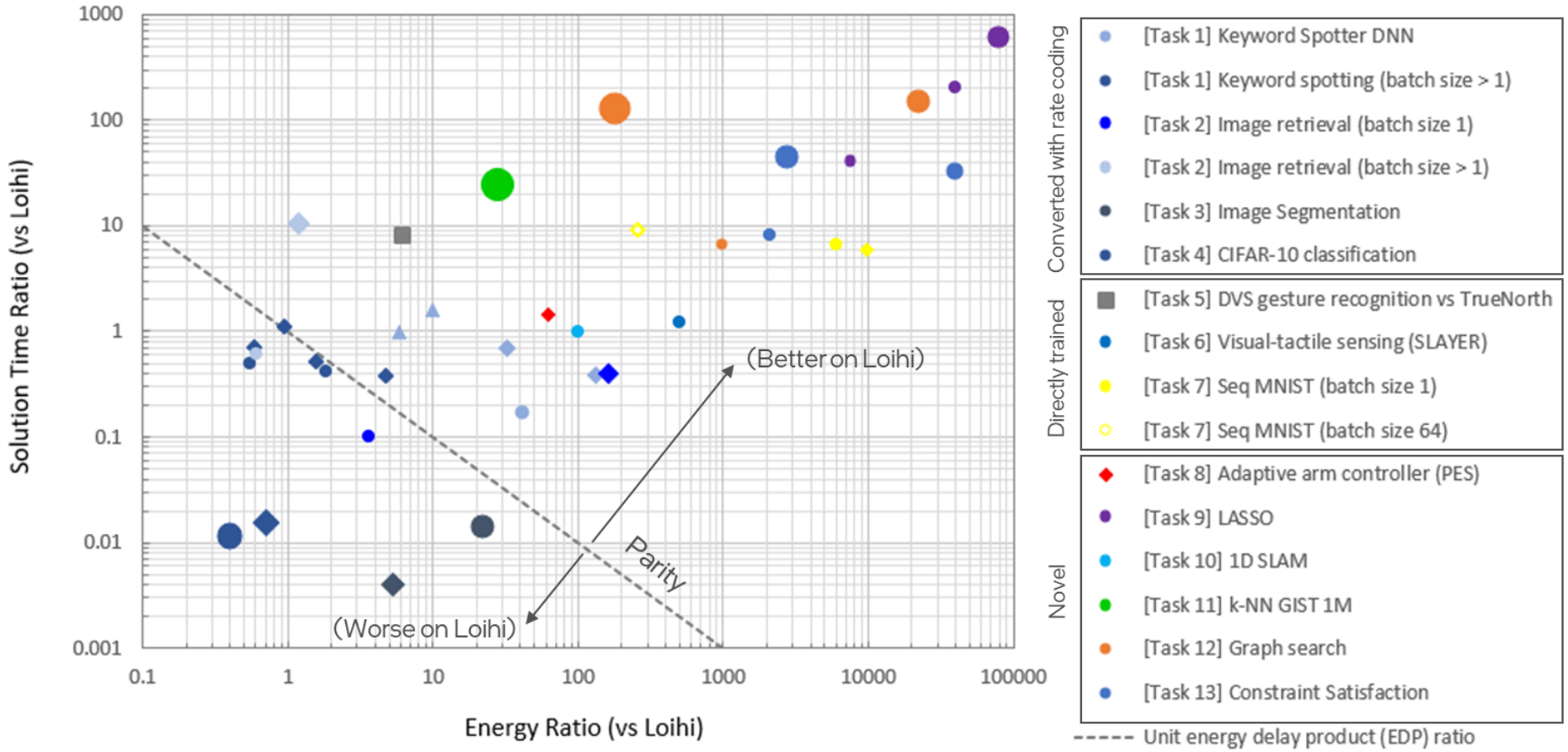


Email inrc_interest@intel.com to get involved!

Thank You!

Email inrc_interest@intel.com for more information

Neuromorphic computing: Benchmarking



Circles: Intel Core or Xeon CPUs; diamonds: Nvidia GPUs; triangles: Intel Movidius Neural Compute Stick; crosses: IBM's TrueNorth architecture. Blue: feed-forward rate-coded networks; red/orange: backpropagation-trained; gray: feed-forward networks with online learning; green: attractor networks; yellow: SNNs exploiting temporal coding or precise spike timing. Results may vary.

Davies et al., IEEE Proc. 2021