intel labs

Lava Software Framework

INRC Spring Workshop 2022

Andreas Wild

Neuromorphic Computing Lab

intel

LAVA

What motivates Lava's design goal?

Design goals

Multi-Paradigm

Multi-Abstraction

Multi-Platform

Real-world applications

Characteristics of real-world applications

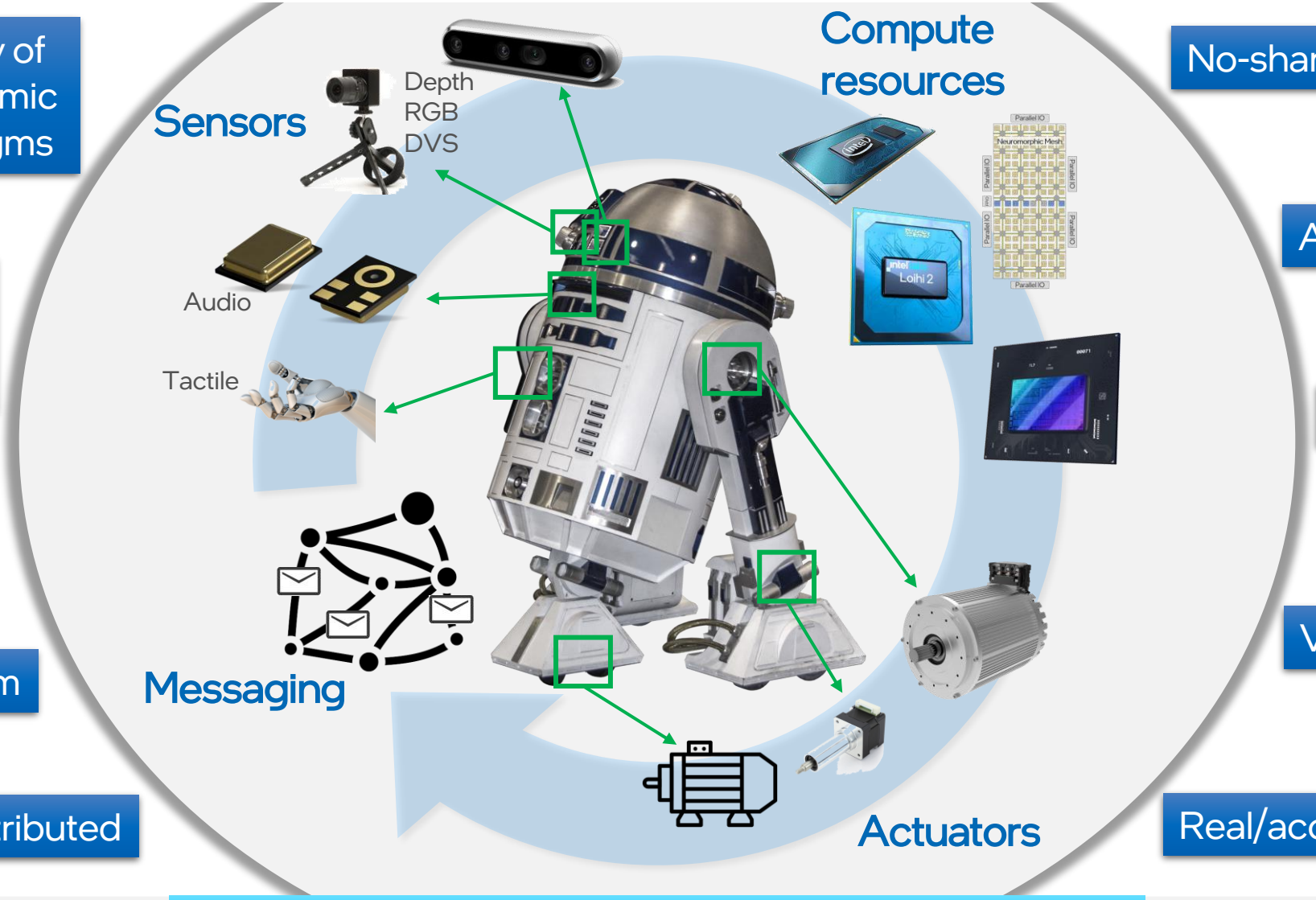
Variety of algorithmic paradigms

Variable precision arithmetic

Heterogenous

Parallelism

Distributed



No-shared memory

Asynchronous

Low latency

Variable speeds

Real/accelerated -time

What should productive SW framework look like?

Design goals and requirements for Lava

Multi-paradigm	Multi-abstraction	Multi-platform	Accessibility
App-specific algorithm libraries	Stable & HW agnostic abstractions	Asynchronous, parallel, distributed programming	Debugging & visualization
Model library	Algorithms tailored to compute resources	Seamless communication across platforms	Power and performance profiling
Composability	Programmability at different levels of abstraction	Application prototyping on CPU	Open source with permissive licensing
Existing framework support*	Hierarchical behavioral descriptions		

No existing framework satisfies all these requirements!

Lava

A new SW framework for neuromorphic computing

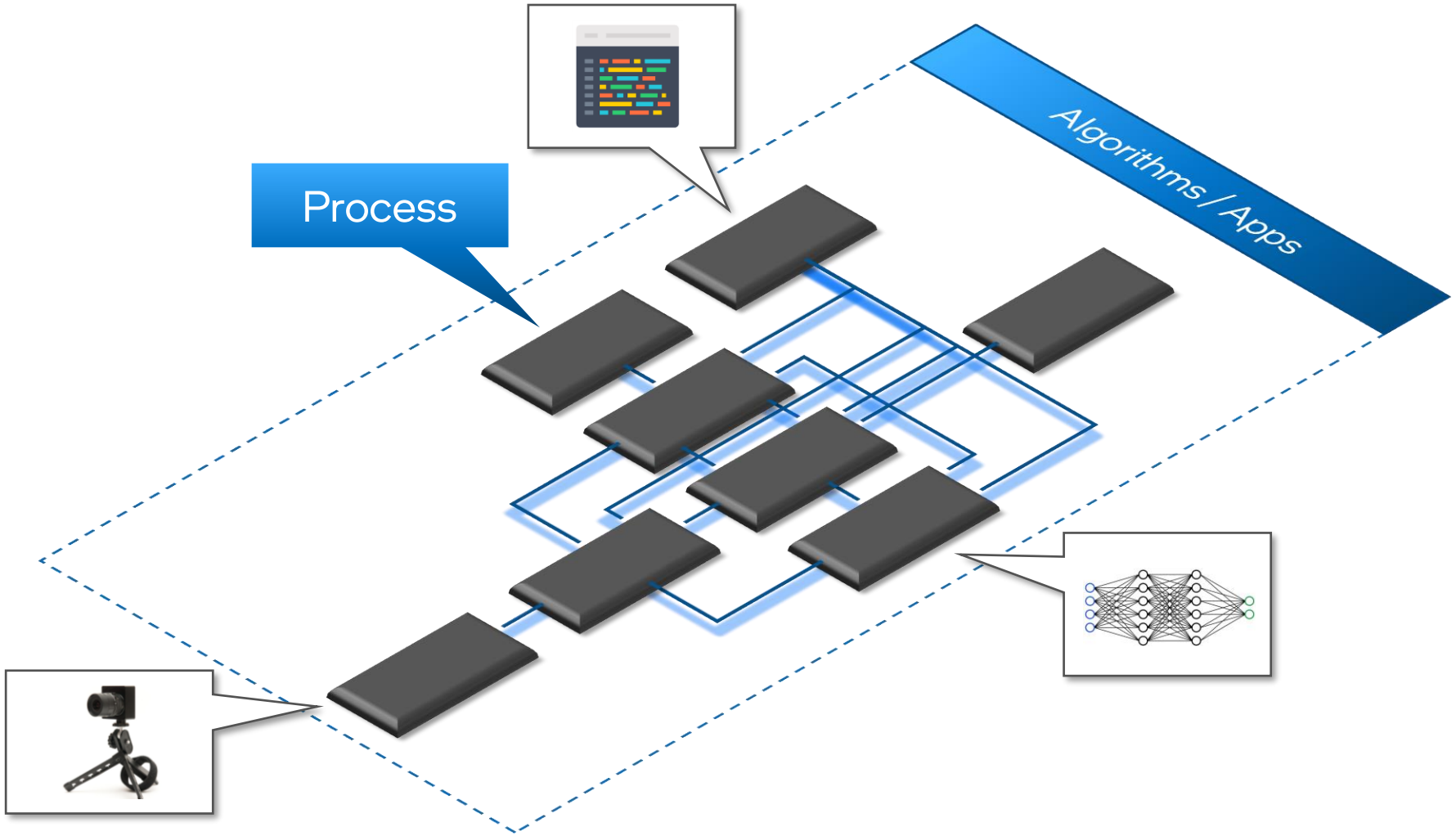
Vision: Enable mainstream adoption of neuromorphic technologies!

Mental model for thinking about computing

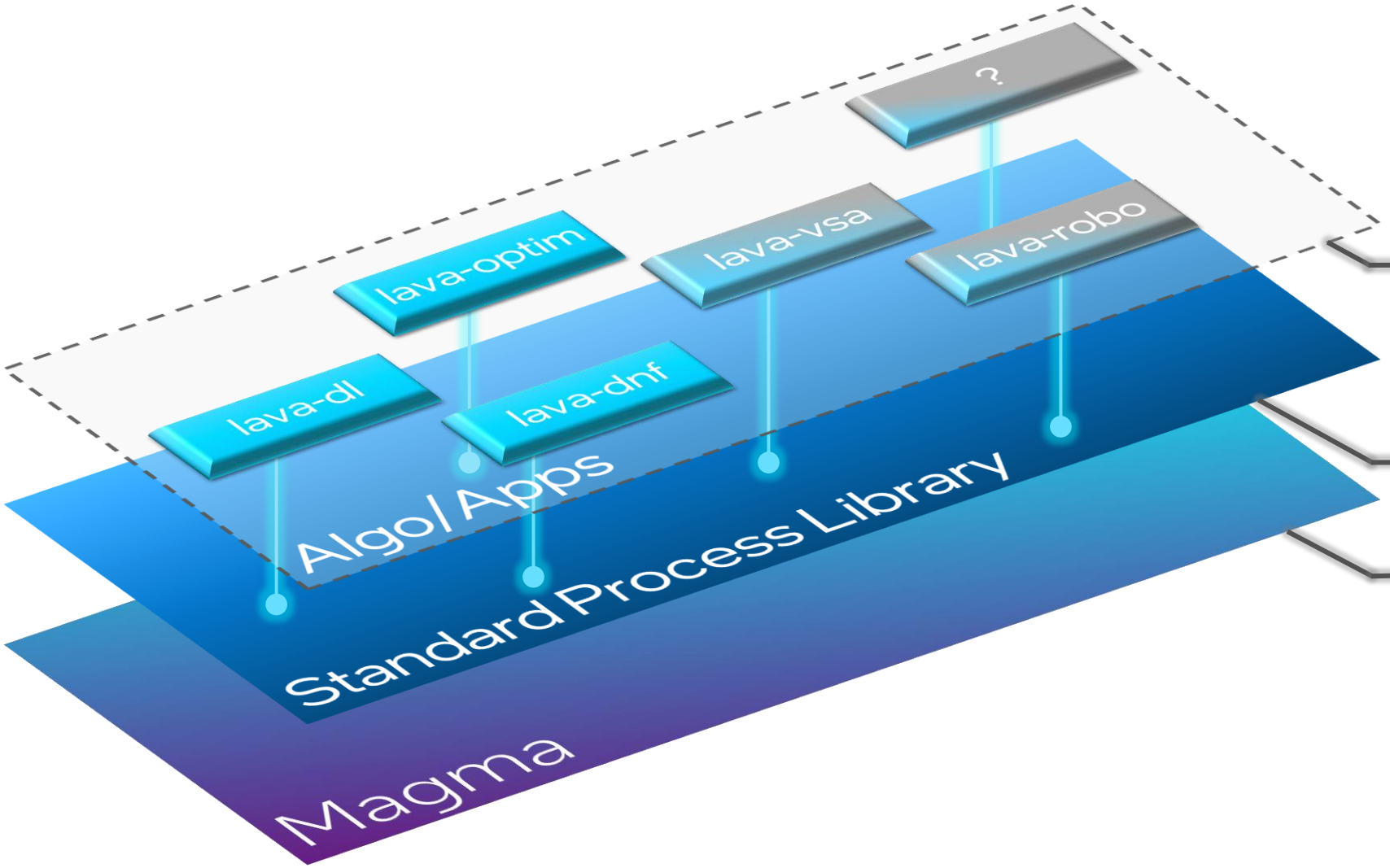
+

SW tools for developing algorithms

Processes: Lava's most fundamental abstraction



What does Lava do for you?



Algo/App libraries:

- Productive APIs
- Pre-built models
- Process generators

Library of generic process building blocks

HW abstraction

The Lava Software stack

Algo/App libraries — BSD

- Optimization
- DNF
- DL
- ...

Standard Process Library — BSD

Future Third Party Interfaces*

Nengo ROS

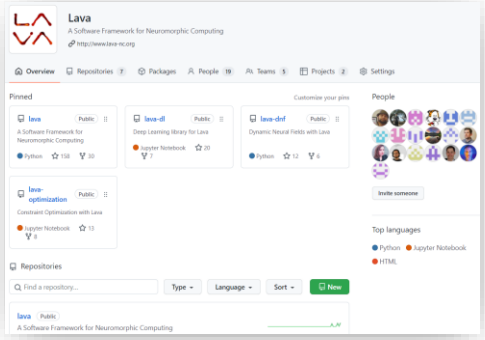
Magma

- *Communicating Sequential Process* API — BSD
- Compiler — BSD/LGPL/Proprietary
- Runtime — LGPL/Proprietary

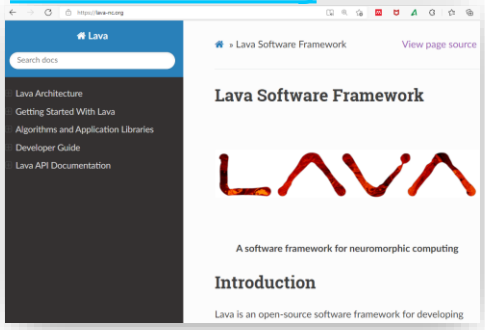
Utilities

- Profiler — BSD
- Visualizer — BSD
- Float2Fixed — BSD
- Validator — BSD

www.github.com/lava-nc

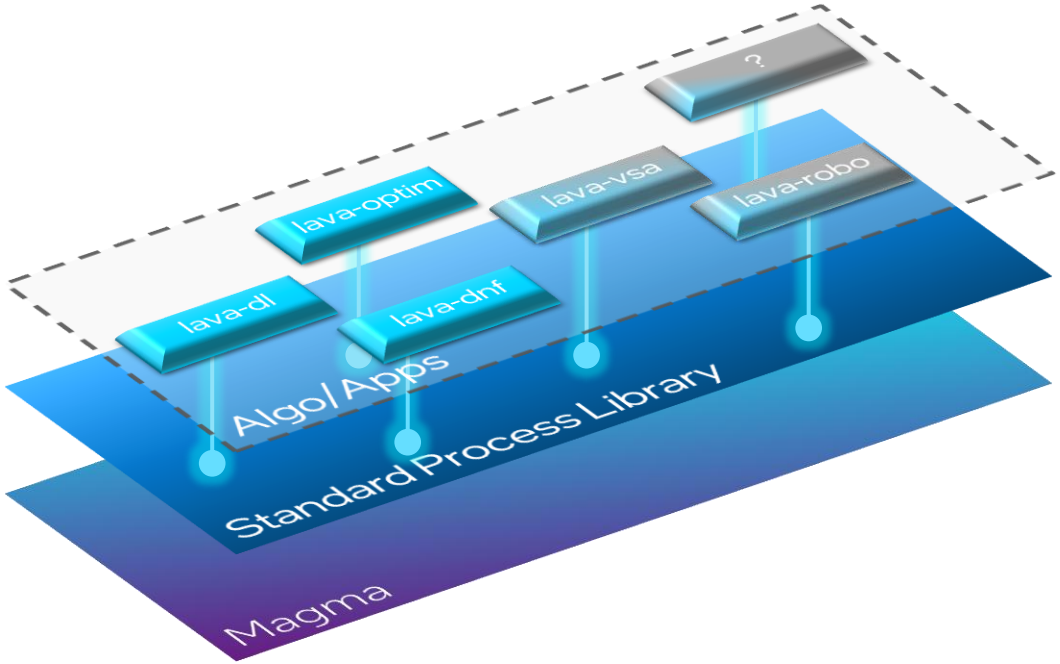


www.lava-nc.org



*TensorFlow, the TensorFlow logo and any related marks are trademarks of Google Inc. | PyTorch, the PyTorch logo and any related marks are trademarks of Facebook, Inc. | The "nine dots" ROS logo is a trademark of Open Robotics.

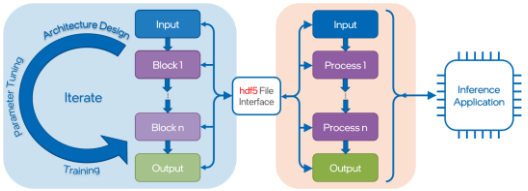
Lava's 3 layers of abstraction



Lava algorithm libraries

lava-dl

- Direct & HW-aware training of event-based DNNs
- Rich neuron model library (feed-forward & recurrent)

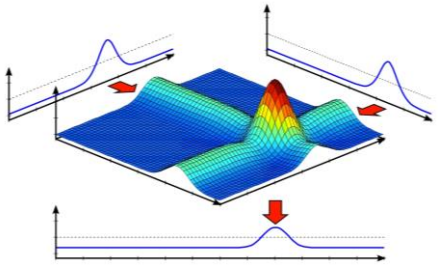


lava-optim

- Family of constraint optimization solvers
- Today: QP, CSP, QUBO
- Future: MPC, LCA, ILP, ...
- Standalone use or as part of AI applications

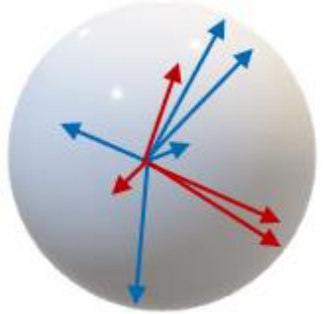
lava-dnf

- Design models with attractor dynamics
- Stabilize temporal data
- Selective data processing
- Dynamic working memories



lava-vsa (WIP)

- API for algebraic model description for VSAs
- Library of data types and operations (composition, binding, factorization, ...)

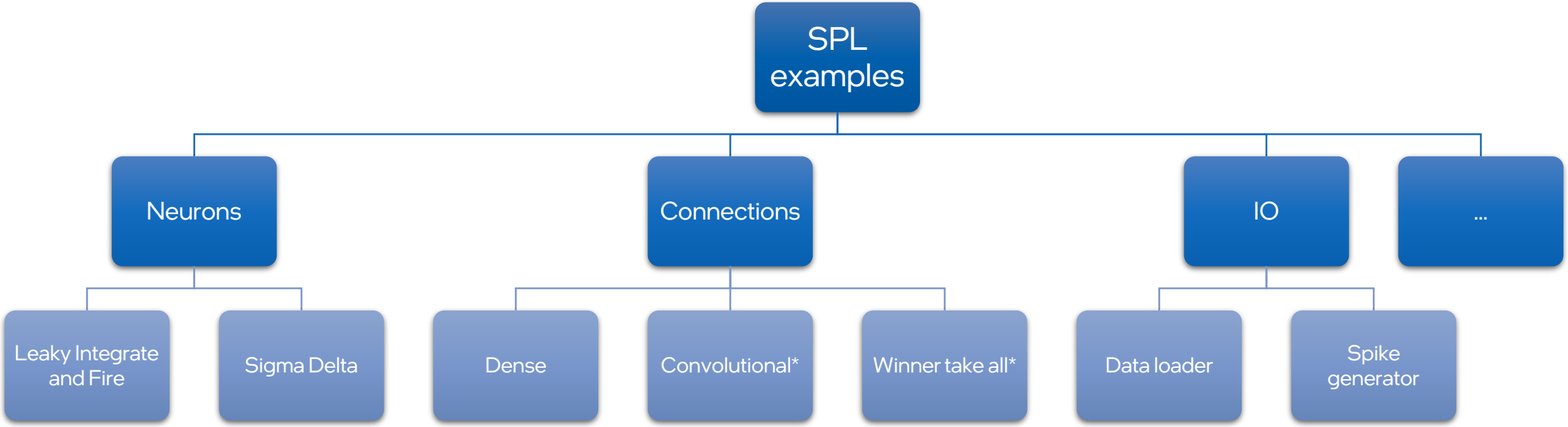


Future directions

- lava-io (sensor/actuator interfaces)
- lava-robotics (control, planning, simulator/ROS/YARP interfaces)
- lava-evolve (evolutionary training methods)
- lava-continual (continual online learning methods)
- lava-ui (graphical network creation, visualization, debugging)
- Signal processing
- Off-the-shelf apps (segmentation, tracking, keyword detection, ...)

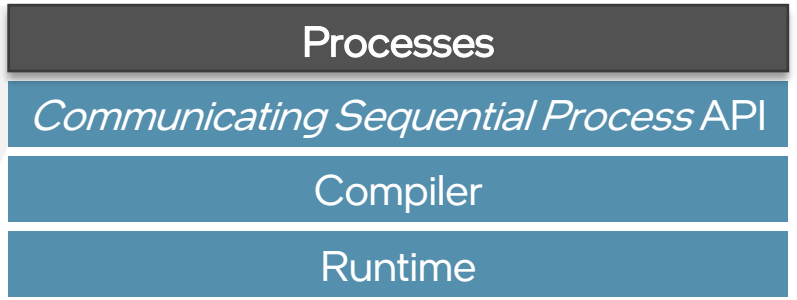
Standard Process Library

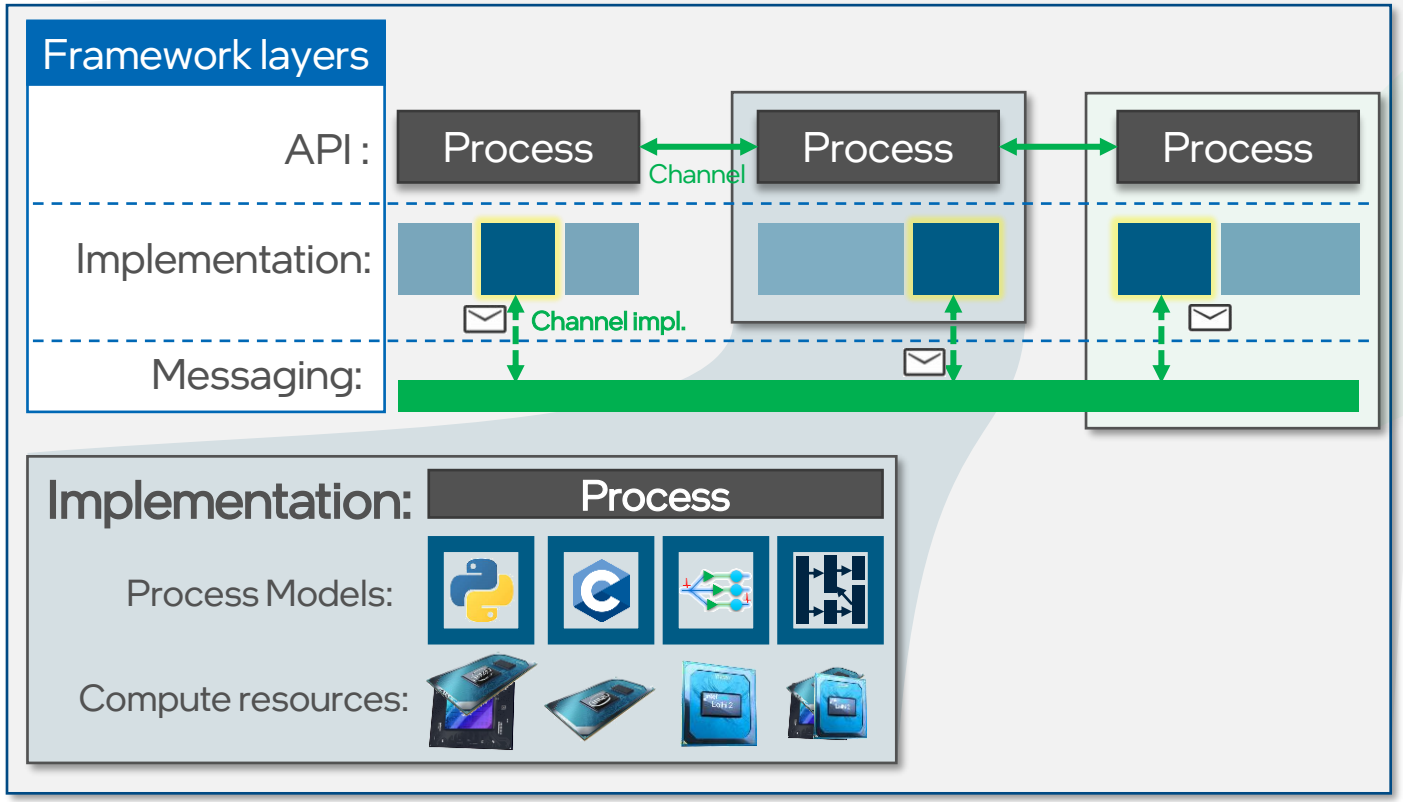
- Library of generic processes
- Used by other processes and libs
- HW-agnostic API
- Quickly expanding

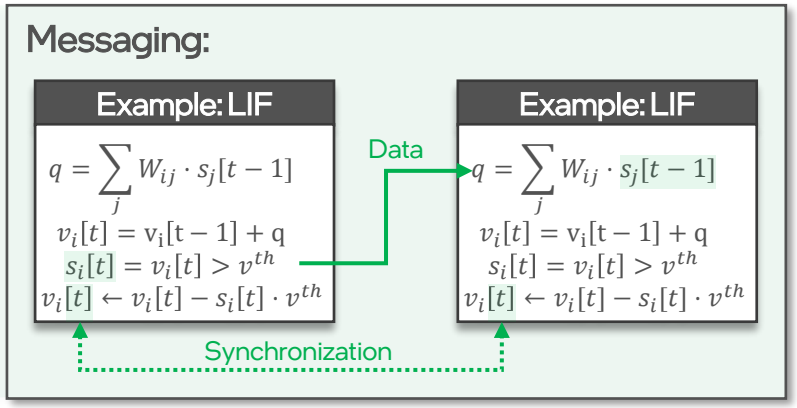


*WIP

Magma







Multi-platform:

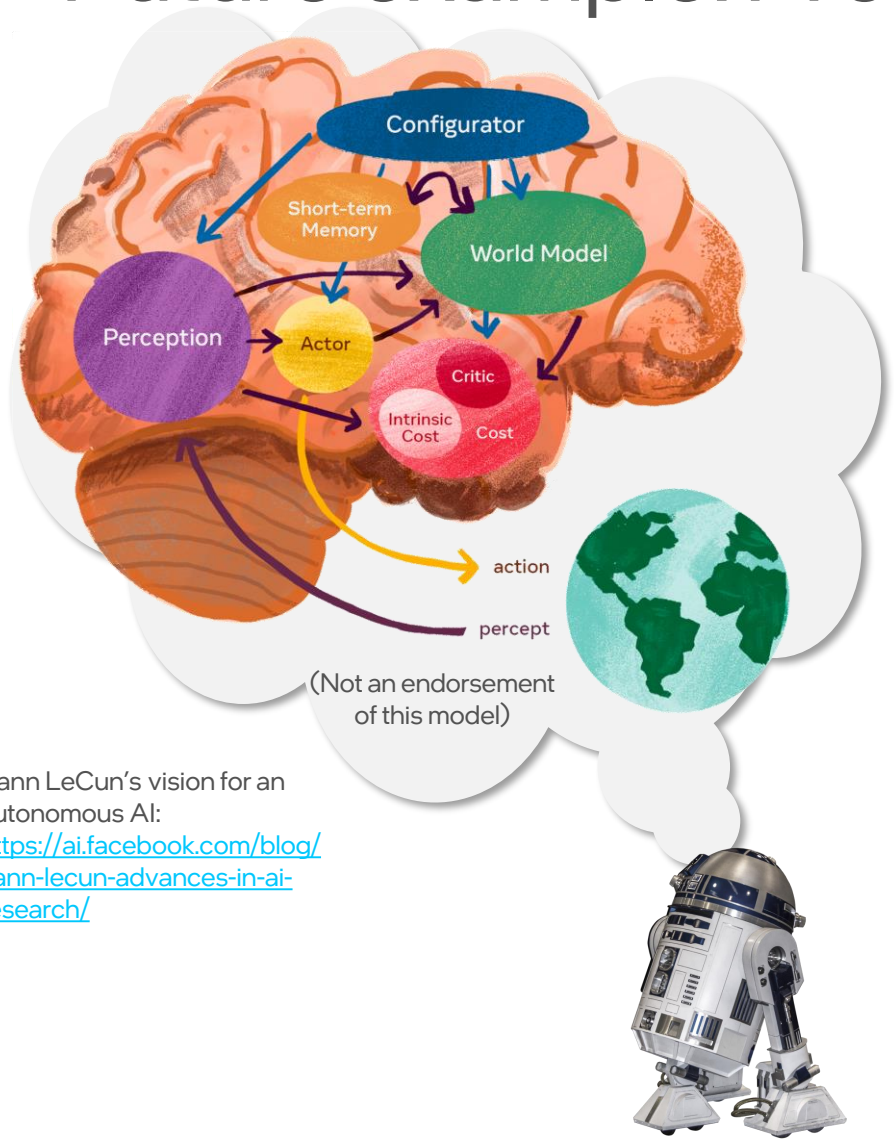
- Asynchronous, distributed computational Processes communicating via channels
- Universal message passing-backend:
  - Data exchange
  - Synchronization

Multi-abstraction:

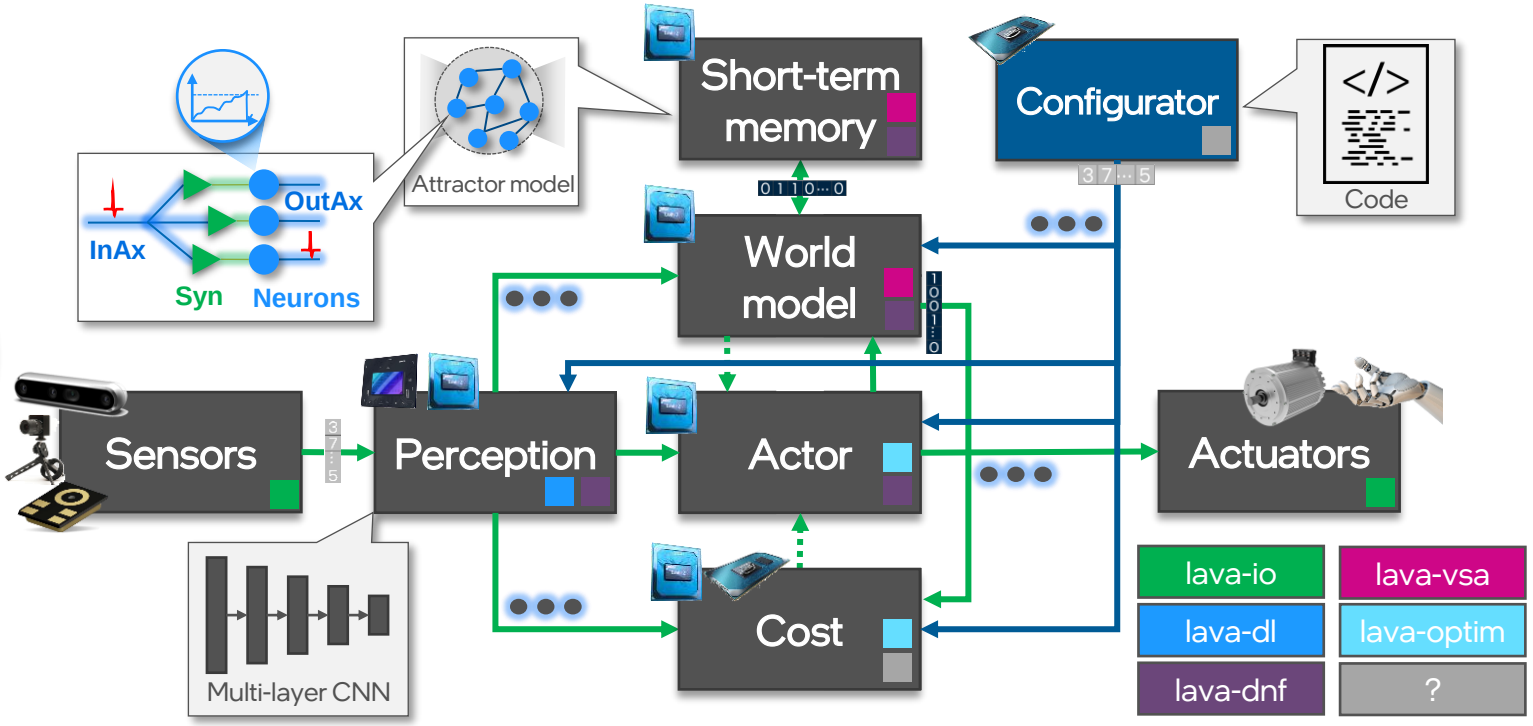
- HW-agnostic Process APIs
- HW-specific or hierarchical implementations
- Extensible at HW, Process and App level

Application development with Lava

Future example: A cognitive app in Lava



Yann LeCun's vision for an autonomous AI: https://ai.facebook.com/blog/yann-lecun-advances-in-ai-research/



Key features

- Sensors | Computational modules | Actuators == Lava Processes
- Process have different behavioral implementations or sub-structure
- Computational modules generated by Lava libraries
- Processes execute on most suitable compute resource
- Event-based computation & communication via messages over channels
- Automatic conversion between coding strategies
- Interoperable with other frameworks

Future example: A cognitive app in Lava

Import

```python
# Import processes from any library
from lava.lib.io.input import DVS
from lava.lib.dl.models import VisualFeatureExtractor
from lava.lib.agents.actor import Actor
...
```

Initialization

```python
# Initialize processes via custom API with spec
dvs = DVS(..<params>..)
vfe  = VisualFeatureExtractor(..<params>..)
act  = Actor(..<params>..)
...
```

Connectivity

```python
# Connect processes via directional input/outpu
dvs.spikes_out.connect(vfe.spikes_in)
vfe.spikes_out.connect([act.input, ...])
...
```

Execution

```python
# Helper configuration classes
from lava.magma.core.run_conditions import RunC
from lava.magma.core.run_configs import Loihi2HwCfg

# Compile and execute model on Loihi 2 HW continously
act.run(condition=RunContinuous(),
        run_cfg=Loihi2HwCfg())
```

Inspection

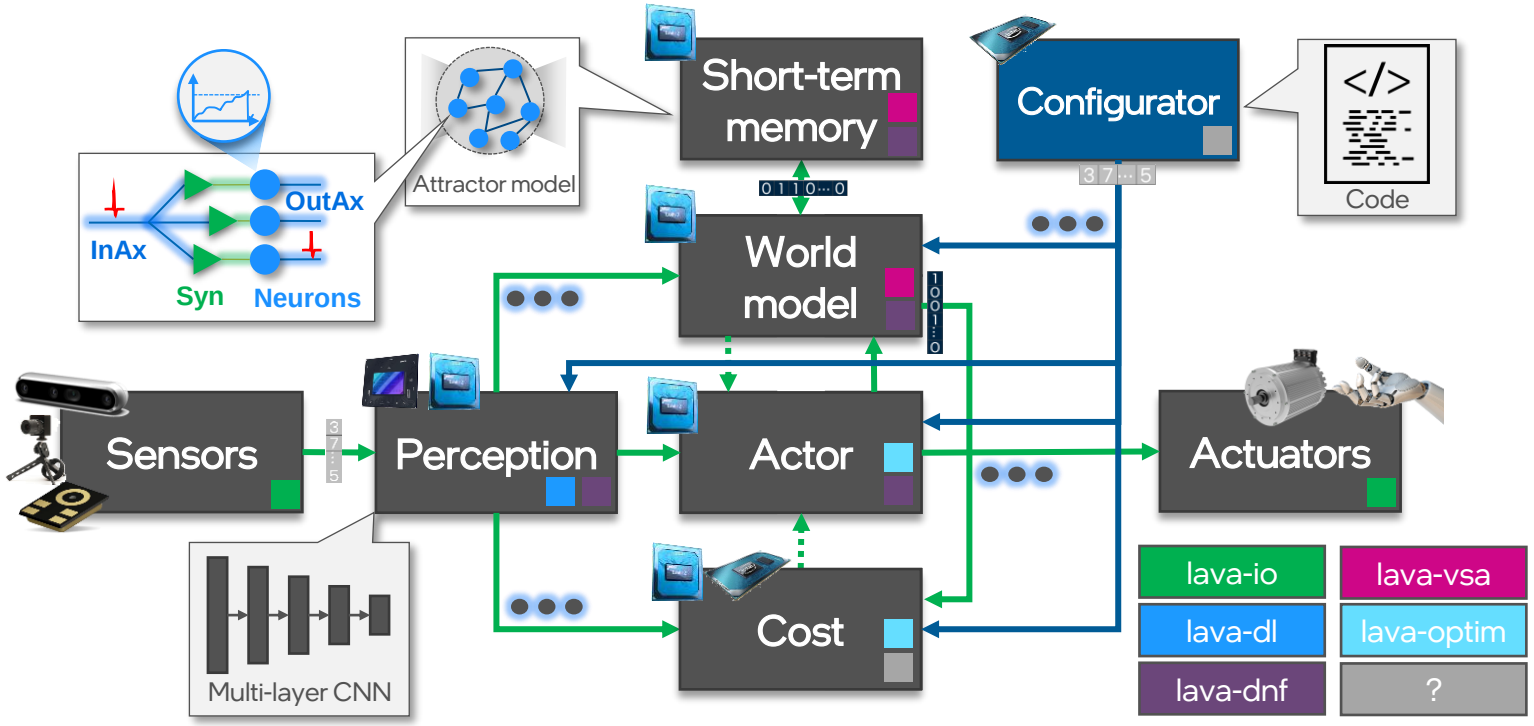
```python
# Run for 100 seconds
import time
time.sleep(100)
act.pause()

# Inspect & modify internal process variables
print(vfe.layers[3].voltage.get())
vfe.layers[4].current.set([1, 2, ..])
```

Resume/Stop

```python
# Resume execution for another 100 steps
from lava.magma.core.run_conditions import Runs
act.run(condition=RunSteps(num_steps=100))

# Terminate
act.stop()
```



Learn how to build behavioral models in Lava Deep Dive session!

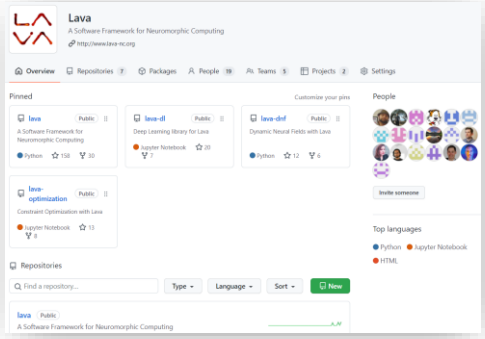
How to Contribute to Lava

Join the Lava working group for further discussion!

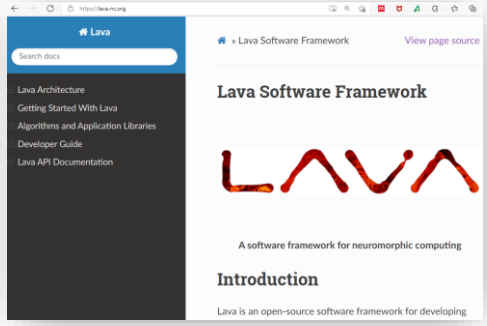
You can help just by using Lava and sharing your code.

- Build models with Lava and share your code
- Port common models and algorithms to Lava
- Connect Lava to other frameworks and tools
- Help us support more neuromorphic devices
- Extend lava-dl training with additional methods
- Work on new Lava libraries (e.g. lava-evolve, lava-robotics)
- Or help optimize and extend Lava's compiler & runtime infrastructure…

www.github.com/lava-nc



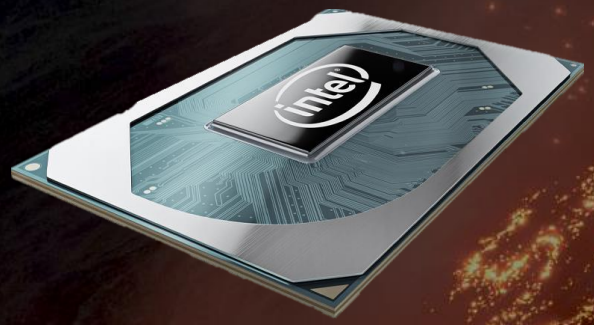
www.lava-nc.org



Lava roadmap

Announcement: Lava soon available on Loihi 2!

LAVA



October 2021

May 2022

Upcoming v0.5.0 features:

- Extends support from CPU to Loihi 1 & 2
- Program NeuroCores (Py) & embedded CPUs (C)
- Extended standard process library
- NeuroCore compiler now fully automated
- First end-to-end Deep Learning app examples

Join *Lava Deep Dive* tomorrow to learn more!

Lava Roadmap for 2022*

First Quarter

- ✓ Execute on CPUs and GPUs
- ✓ Provide access to Loihi 2 Kapoho Point systems
- ✓ lava-dl: Train deep SNNs with SLAYER

Second Quarter

- Execute on Loihi 1 & 2
- On-chip learning
- Real-time sensor input from RGB and DVS
- lava-vsa: Enable building VSAs in Lava
- lava-optim: Add CSP, QUBO & QP solver
- lava-dnf: Add visual search

Third Quarter

- Execute on embedded systems
- Compiler memory & performance optimizations
- Cross-platform power & performance Profiler
- lava-dl: Improve support for recurrent SNNs
- lava-vsa: Add resonator for factorization

Fourth Quarter

- Execute on multi-node systems
- lava-vsa: Add continual learning & attention

* Lava is an agile project. Features are tentative and subject to change.

Lava-focused workshop sessions

- Leverage Lava libraries for app development
- Develop your own processes
- Lava architecture
- Emerging community projects

Lava Community Working Group:
- Ensure Lava solves your problems
- Lava Community launch
- Discussion

- Getting started with Lava
- Running elementary models

Time	April 19	April 20	April 21	April 22	
06:00 PDT 09:00 EDT 15:00 CET		Loihi 2 Deep Dive Option 2 Engaged INRC Members	Lava Basics Tutorial Option 2	Loihi for Robotics	
07:00 PDT 10:00 EDT 16:00 CET		Signal Processing	Lava Deep SNN Tutorial Option 1		
08:00 PDT 11:00 EDT 17:00 CET	New Tools for a New Era of Neuromorphic Computing	Lava Deep Dive	Continual Learning	Working Groups #2	Aerospace Apps & Tech
09:00 PDT 12:00 EDT 18:00 CET			Offline Training	Working Groups #3	
10:00 PDT 13:00 EDT 19:00 CET	Q&A / Break Featured Community Results	Application Frontiers	Optimization	Ecosystem Development	
11:00 PDT 14:00 EDT 20:00 CET		Working Groups #1	Vector Symbolic Architectures	Conclusion	
12:00 PDT 15:00 EDT 21:00 CET					
17:00 PDT 20:00 EDT 02:00 CET	Loihi 2 Deep Dive Option 1 Engaged INRC Members	Lava Basics Tutorial Option 1	Lava Deep SNN Tutorial Option 2		

- Getting started with lava-dl using DNN example

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.