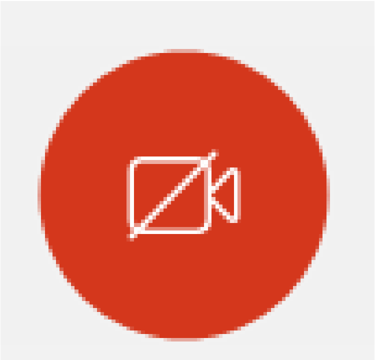
Today’s Session Is Being Recorded

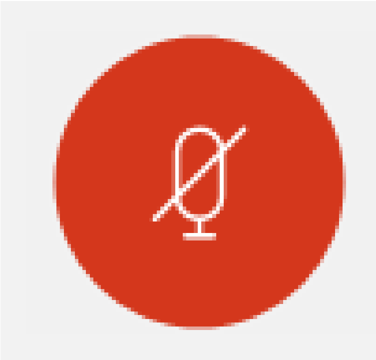
This session is being recorded and will be made available for viewing by others by invitation.

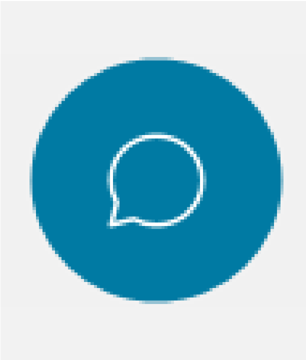
If you do not wish for your voice to be recorded, please place your phone on mute for the duration of the session.

This session is non-confidential, so if you wish to ask a question or make a comment, please do not disclose any information that is confidential to you, your employer, or any third party.

The views expressed in this session are those of the contributors and do not necessarily reflect the views of Intel Corporation.







RECORDING
IN PROGRESS

Latest DNN results on Loihi

Garrick Orchard | Andreas Wild

Feb 8, 2021

INRC Winter Workshop 2021

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.

Agenda

- Why neuromorphic deep learning
- Training deep SNNs for Loihi
- Performance of deep SNNs on Loihi
- Conclusion

Why neuromorphic deep learning?

Loihi SNNs can benefit from Deep Learning

- Deep Learning (backpropagation) is a powerful tool
  - There are mature theories and tool frameworks available
  - Achieves state of the art results on many tasks

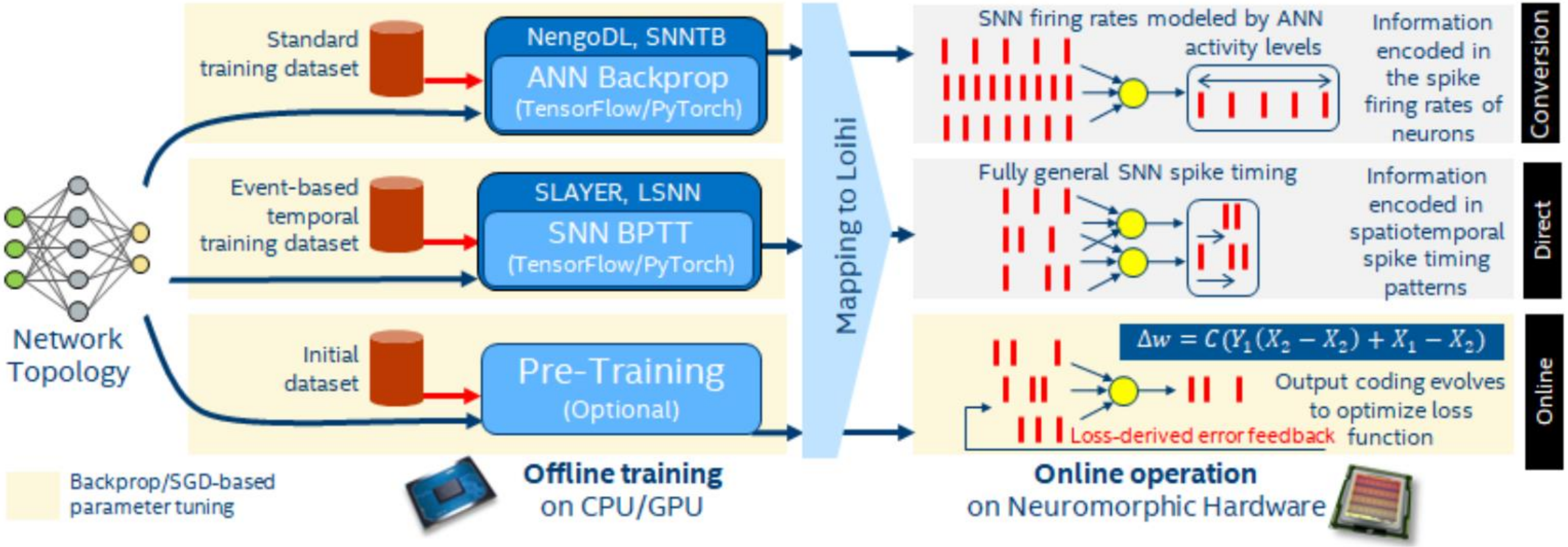
- Deep Learning has a lot to offer to SNNs
  - Provides a way to configure large networks of neurons
  - Provides ideas for network architectures and their suitability for different tasks

Loihi is not a Deep Learning accelerator

	Loihi	Feedforward DNNs
Memory	Limited on-chip memory	Require large memory to store many parameters
Neuron update	Depends on previous state	Stateless
Synaptic op	Accumulate	Multiply-accumulate
Updates	Optimized for sparse irregular synaptic updates	Requires dense predictable computation

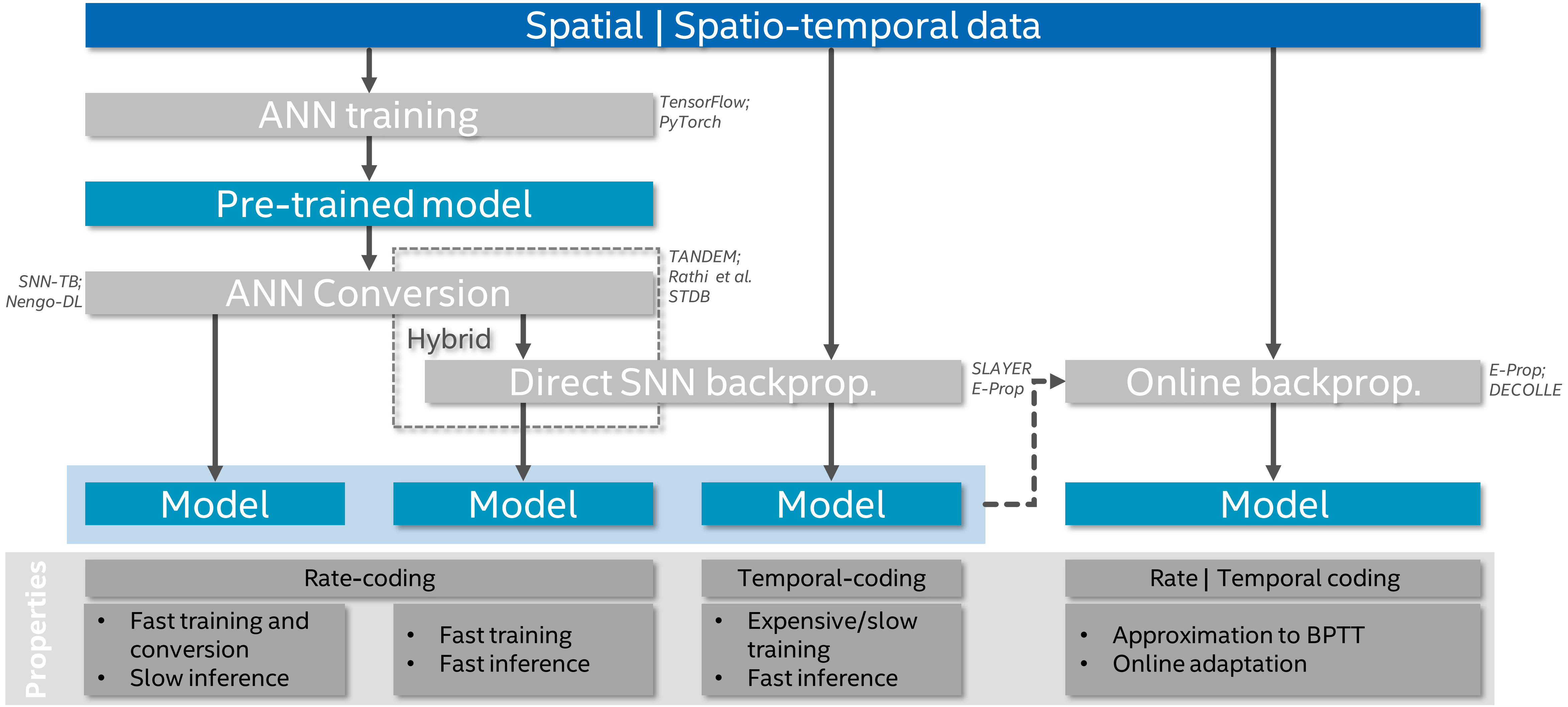
- It is naïve to expect Loihi to beat DNN accelerators on feedforward DNN workloads
- Loihi *can* run DNNs, but it can also run many other networks

Deep learning on Loihi today



Training deep SNNs for Loihi

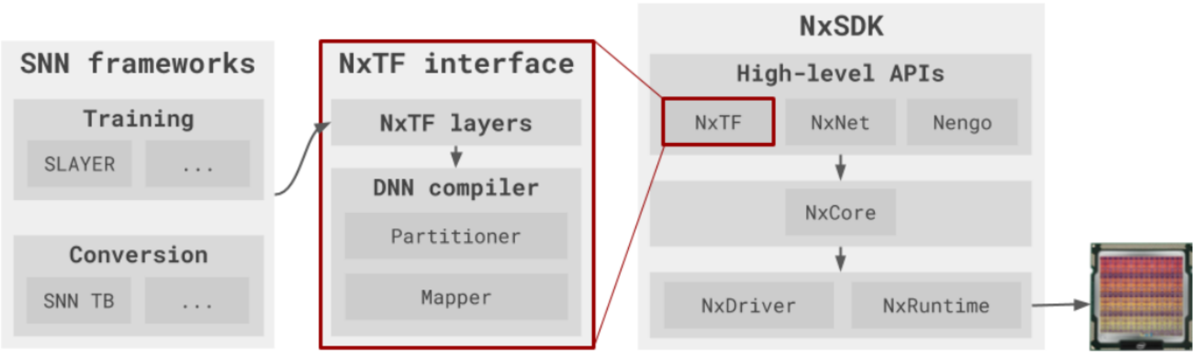
Training approaches



Offline SNN conversion

- Nengo-DL*:
  - ANNs trained with TensorFlow
  - Conversion to rate-coded SNN
  - Deep SNNs interoperable with rest of Nengo framework

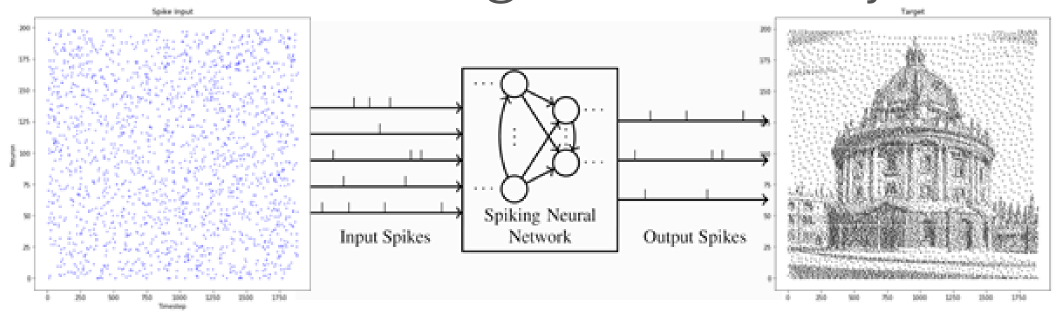
- NxTF**:
  - Part of NxSDK family
  - Keras API
  - Most resource efficient CNN compiler for Loihi
  - Relies to external training frameworks (SLAYER | SNN-TB)
  - *To be merged into Lava*

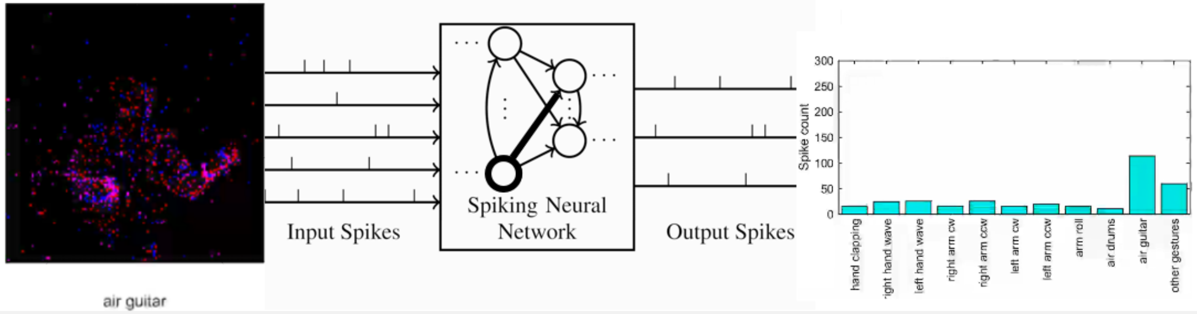


*Rasmussen, D. NengoDL: Combining Deep Learning and Neuromorphic Modelling Methods. Neuroinform 17, 611–628 (2019)
**Rueckauer et al., NxTF: An API and Compiler for Deep Spiking Neural Networks on Intel Loihi, arXiv: 2101.04261 (2021)

Offline direct SNN training

- SLAYER[1]: See recorded talk!
  - Custom PyTorch implementation
  - Automatic deployment to Loihi through NxSDK
  - Learns weights and delays





- Other approaches:
  - Spike Time Dependent Backprop (STDB)[2]
  - Hybrids: Hybrid-STDB[3], TANDEM[4]
  - TU Graz's BPTT[5]
  - SpyTorch[6]

[1] S. B. Shrestha and G. Orchard, NeurIPS (2018)
[2] Y. Wu et al., Front. Neurosci., Vol. 12, p. 331, (2018)
[3] N. Rathi et al., *arXiv:2005.0180* (2020)
[4] J. Wu et al., arXiv:2007.01204 (2020)
[5] G. Bellec et al., NIPS, pp. 787–797 (2018)
[6] https://github.com/fzenke/spytorch

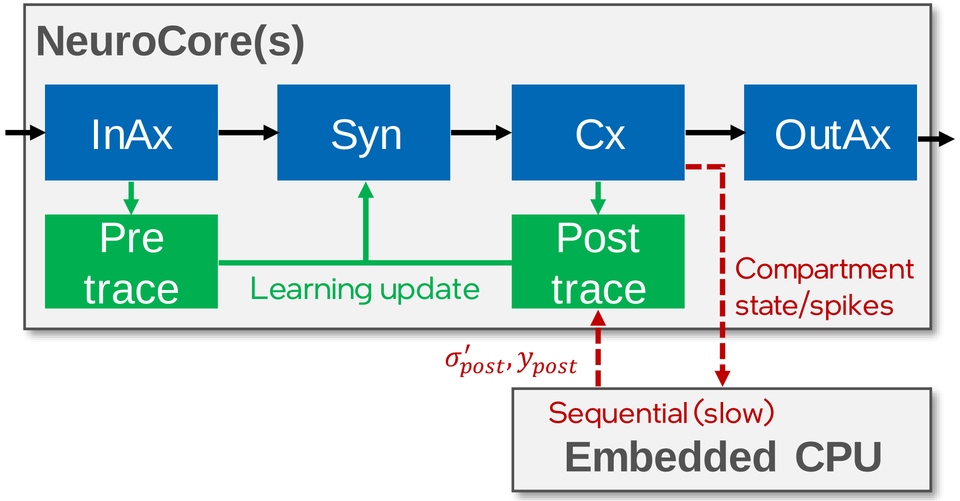
Online SNN backpropagation

- Several approximative backprop methods under development in INRC
  (Maass, Neftci , Qiu, Senn)
- Single layer 3-factor "backprop" demonstrated on Loihi:

  $$\Delta w = x_{pre} \times \left(\sigma'_{post} \odot y_{post,err}\right)^T$$

- Examples:
  - PES
  - SEOL/DECOLLE

- Problem:
  - σ'_{post}, y_{post} not directly accessible by synapses
  - Workaround via embedded CPU inefficient



Performance of deep SNNs on Loihi

Models and tasks

- Rigorous benchmarking of various DNN tasks
- Tasks differ by:
  - Training approach & tool
  - Reference platforms
  - Size (cores) / complexity
- Comparable accuracies
- Benchmarking metrics: latency, energy, EDP

	Task / Dataset | Simulator	Training / Tool	Platform	Cores	Batch size	Acc (%)	Latency (ms)	Latency ratio	Energy (mJ) idle	Energy (mJ) dyn	Energy ratio	EDP ratio
1	Keyword spotting [S1] / Custom	Conversion / NengoDL	Loihi:	6	1	93.8	8.9	1.0x	0.3	0.7	1.0x	1.0x
			NCS:		1	92.7	6.2	0.7x	1.3	2.8	4.2x	2.9x
			Loihi:	26	1		8.9	1.0x	0.3	0.8	1.0x	1.0x
			NCS:		1		16.7	1.9x	3.5	9.8	12.0x	22.3x
2	Image retrieval [S2] / Fashion MNIST	Conversion / SNN-TB	Loihi:	17	1	85.1	3.0	1.0x	2.8	0.1	1.0x*	1.0x*
			i7-8750H		1	90.1	0.3	0.1x	5.8	0.5	3.6x*	0.3x*
			V100		1	90.1	1.3	0.4x	31.1	21.3	162.3x*	70.5x*
			i7-8750H		128	90.1	1.7	0.6x	0.3	0.1	0.6x*	0.3x*
			V100		4096	90.1	30.7	10.3x	0.2	0.2	1.2x*	12.1x*
3	Image segmentation [S3] / ISBI 2D EM	Conversion / NengoDL	Loihi:	256	1	90.6	162.1	1.000x	914.1	6.5	1.0x*	1.00x*
			E5-1650:		1	91.0	2.2	0.014x	29.9	142.3	22.0x*	0.30x*
			RT 2080:		1	91.0	0.6	0.004x	4.0	33.9	5.2x*	0.02x*
4	Image classification / CIFAR 10	Conversion / SNN-TB	Loihi[1]:	861	1	91.6	339.7	1.00x	218.7	88.3	1.0x	1.00x
			RTX 2070[2]:		1	92.6	5.2	0.02x	62.0	146.5	0.7x	0.01x
			i7-9700K[2]:		1	92.6	4.0	0.01x	8.8	121.7	0.4x	0.01x
5	Gesture recog. / DVS gestures	Direct / SLAYER	Loihi[1]:	84	1	98.1	12.7	1.0x	0.0	0.0	1.0x	1.0x
			True North:		1	94.6	104.6	8.2x	0.1	0.0	6.2x	50.8x
6	Robot navigation [S4] / Gazebo sim.	Direct / STDB	Loihi(T=5):	1	1	93.0	2.2	1.0x	2.4	0.0	1.0x*	1.0x*
			Loihi(T=50):	1	1	98.0	8.0	3.6x	8.7	0.1	3.6x*	13.1x*
			Jetson TX2:		1	90.5	2.6	1.2x	3.5	1.2	43.3x*	50.3x*
			E5-1660:		1	90.5	0.2	0.1x	2.0	8.9	329.6x*	22.6x*
			Tesla K40:		1	90.5	0.3	0.1x	7.9	15.3	564.1x*	83.7x*
7	SeqMNIST classification / MNIST	Direct / BPTT	Loihi[3]:	1	1	96.0	12.7	1.0x	0.2	0.0	1x	1x
			i5-7440HQ[4]:		1	98.5	83.2	6.6x	1740		6108x	40016x
			Tesla P100[5]:		1	98.5	76.0	6.0x	2759		9685x	57959x
			Tesla P100[5]:		64	98.5	111.4	8.8x	74		260x	2279x
8	Relational reasoning / bAbI	Direct / BPTT	Loihi[1]	2320	1	98.5	6.5	1.0x	12.4	10.3	1.0x	1.0x
			RTX 2070[2]:		1	98.5	2.5	0.4x	84.1	14.8	4.4x	1.7x
			RTX 2070[2]:		50	98.5	4.4	0.7x	3.0	6.5	0.4x	0.3x
			Loihi[1]:	124	1	98.5	3.3	1.0x	0.6	5.0	1.0x	1.0x
			RTX 2070[2]:		1	98.5	2.4	0.7x	80.2	12.0	16.4x	12.1x
			RTX 2070[2]:		50	98.5	2.9	0.9x	1.9	2.4	0.8x	0.7x
9	Adaptive robotic ctrl. [S5] / Mujoco sim.	Online / Nengo PES	Loihi:	4	1	73.3	3.1	1.0x	3080		1.0x**	1.0x**
			i7-6700K		1	62.4	3.1	1.0x	14398		4.7x**	4.8x**
			GTX 1070		1	63.8	4.4	1.4x	189216		61.4x**	87.4x**

Davies et al. (2021), (in review)
System test configuration details of all tasks are provided in the backup.
* Report includes excessive inactive power
** No idle power reported but dominated by inactive power
Performance results are based on testing as of Dec. 2020 and may not reflect all publicly available security updates. Results may vary.

Benchmarking methodology

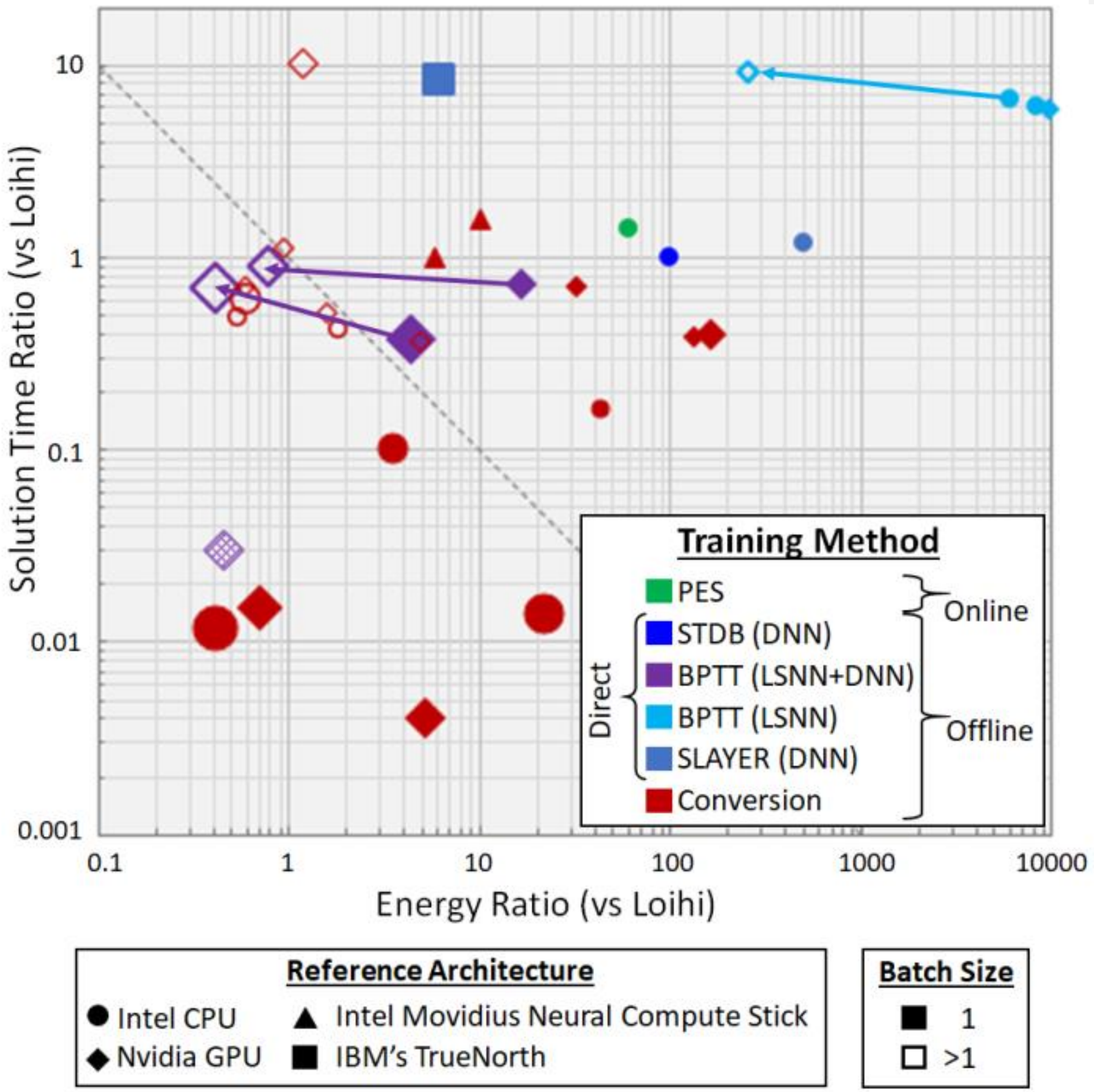
- Benchmarking metric: Energy-Delay Product (EDP)
  - Energy & time consumed to solution for each sample of task
  - Lower is better!

- Characterization tools for Energy and Delay:
  - Loihi: NxSDK Energy and Execution-Time probes automate collection of metrics
  - CPU: Intel SocWatch
  - GPU: nvidia-smi

- For more details see …
  - Whitepaper: Power and energy benchmarking of SNNs on Loihi
  - Pre-recorded presentation: Performance characterization on Loihi

Key lessons

- Energy:
  - Loihi often substantially more efficient
  - Batching improves efficiency of reference arch.
- Latency:
  - Loihi faster/on-par for small workloads
  - Loihi faster for directly trained workloads
  - Loihi slower for large (rate-coded & multi-chip) networks
- Cause for Loihi's latency degradation:
  - Need for increased runtime for deeper rate-coded nets to mitigate error accumulation
  - Rate/time-coding accompanied by high/low traffic
  - Spike mesh congestion due to high traffic and slow inter-chip links

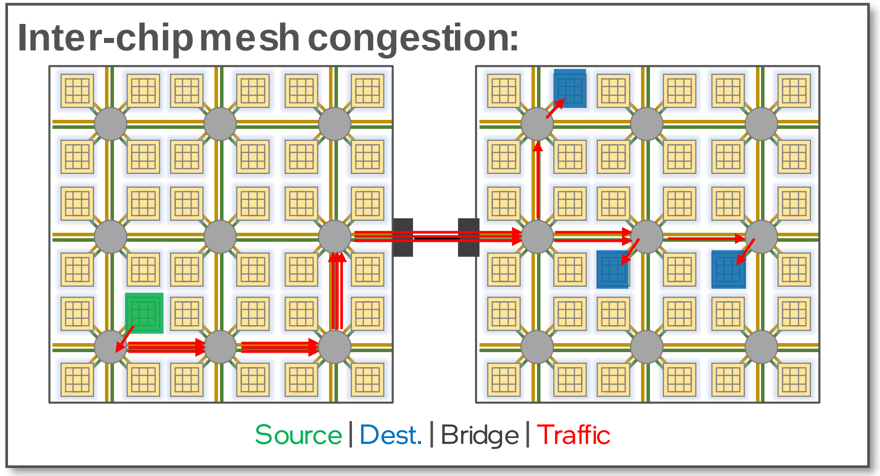


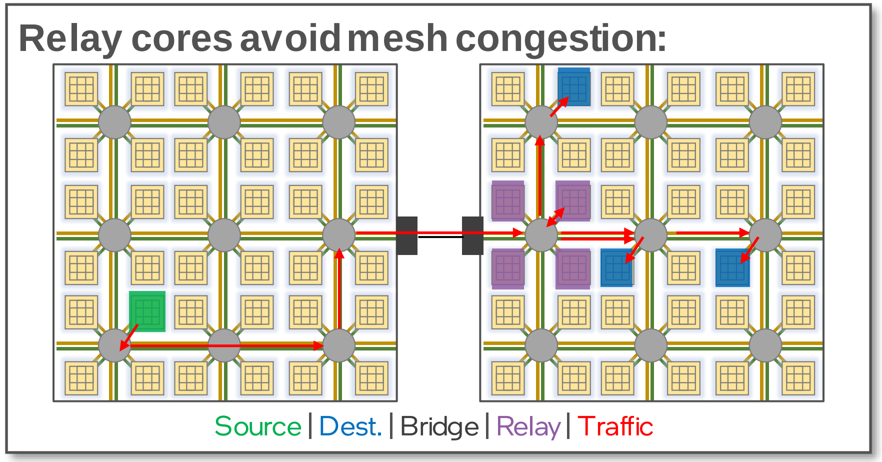
Davies et al. (2021), (in review)
System test configuration details of all tasks are provided in the backup.
Performance results are based on testing as of Dec. 2020 and may not reflect all publicly available security updates. Results may vary.

Mitigating spike congestion

- Issues:
  - Rate-coded SNNs often cause dense traffic
  - Inter-chip links have $\approx 30\times$ lower bandwidth than intra-chip links
- Consequence:
  - Spike congestion at inter-chip links
  - Dramatic slowdown up to $100\times$ possible
- Mitigation:
  - Software: Introduce relay cores* (not ideal)
  - Hardware: Motivates architectural improvements in Loihi 2

*For details, see pre-recorded talk on "Relational Reasoning" DNN





Conclusions and future challenges

- Deep SNNs on Loihi vs. CPU/GPU…
  - … are up to $1000 \times$ more energy efficient
  - … often suffer from $1 - 100 \times$ longer latency especially for multi-chip networks
  - Future HW must/will fix congestion
  - Loihi only advantageous if …
    - batch=1 is a must (real-time) and latency due to batching is not tolerable
    - model size does not require off-chip DRAM → motivates network pruning techniques
- Coding:
  - Old news: Avoid inefficient rate coding, in particular of static data in feed-forward models
  - Prefer directly trained models with temporal input and output code

- Training:
  - Offline training can work:
    - BPTT works well for small SNNs
    - But how to scale up? → expensive
    - Hybrid conversion/training or different coding strategies may help
  - Online training is challenging:
    - Loihi currently only supports 3F-learning with workarounds
    - Batch=1 training inefficient; but the only choice for real-time learning
    - Real-time learning must address continual learning from incremental data to be practically useful

Session outlook

More in-depth discussion on challenges, solutions and opportunities in upcoming sessions!

Session	Type	When	Content	Audience
Loihi 2 enhancements for DNN	Live	Wed. Feb. 10th 11:15am (PST)	Deep dive into upcoming Loihi 2 hardware features relevant for DNN inference and online learning.	Engaged INRC members
How to further improve support for neuromorphic DNNs	Live & Pre-recorded	Wed. Feb. 10th 11:45am (PST)	INRC members share their latest results and analysis what future neuromorphic systems should support. Watch pre-recorded material now!	INRC members
Solving the challenges of deep learning on neuromorphic hardware	Live	Thu. Feb. 11th 8:00am (PST)	Panel participants assess the general challenges and opportunities of deep learning on neuromorphic systems to identify what questions the community should address and how neuromorphic systems must evolve.	Public

intel®

Thank You!

References and System Test Configuration Details

[Task 1] P Blouw et al, 2018. arXiv:1812.01739

[Task 2] TY Liu et al, 2020, arXiv:2008.01380

[Task 3] KP Patel et al, "A spiking neural network for image segmentation," *submitted, in review*, Aug 2020.

[Task 4] **Loihi**: Nahuku system running NxSDK 0.95. CIFAR-10 image recognition network trained using the SNN-Toolbox (code available at https://snntoolbox.readthedocs.io/en/latest). **CPU**: Core i7-9700K with 32GB RAM, **GPU**: Nvidia RTX 2070 with 8GB RAM. OS: Ubuntu 16.04.6 LTS, Python: 3.5.5, TensorFlow: 1.13.1. Performance results are based on testing as of July 2020 and may not reflect all publicly available security updates.

[Task 5] **Loihi**: Nahuku system running NxSDK 0.95. Gesture recognition network trained using the SLAYER tool (code available at https://github.com/bamsumit/slayerPytorch). Performance results are based on testing as of July 2020 and may not reflect all publicly available security updates. **TrueNorth:** Results and DVS Gesture dataset from A. Amir et al, "A low power, fully event-based gesture recognition system," in IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2017.

[Task 6] T. Taunyazov et al, 2020. RSS 2020

[Task 7] Bellec et al, 2018. arXiv:1803.09574. **Loihi**: Loihi: Wolf Mountain system running NxSDK 0.85. **CPU**: Intel Core i5-7440HQ, with 16GB running Windows 10 (build 18362), Python: 3.6.7, TensorFlow: 1.14.1. **GPU**: Nvidia Telsa P100 with 16GB RAM. Performance results are based on testing as of December 2018 and may not reflect all publicly available security updates.

[Task 8] T. DeWolf et al, "Nengo and Low-Power AI Hardware for Robust, Embedded Neurorobotics," Front. in Neurorobotics, 2020.

[Task 9] Loihi Lasso solver based on PTP Tang et al, "Sparse coding by spiking neural networks: convergence theory and computational results," arXiv:1705.05475, 2017. **Loihi**: Wolf Mountain system running NxSDK 0.75. **CPU**: Intel Core i7-4790 3.6GHz w/ 32GB RAM running Ubuntu 16.04 with HyperThreading disabled, SPAMS solver for FISTA, http://spams-devel.gforge.inria.fr/.

[Task 10] G Tang et al, 2019. arXiv:1903.02504

[Task 11] EP Frady et al, 2020. arXiv:2004.12691

[Task 12] Loihi graph search algorithm based on *Ponulak F., Hopfield J.J. Rapid, parallel path planning by propagating wavefronts of spiking neural activity. Front. Comput. Neurosci. 2013.* **Loihi**: Nahuku and Pohoiki Springs systems running NxSDK 0.97. **CPU**: Intel Xeon Gold with 384GB RAM, running SLES11, evaluated with Python 3.6.3, NetworkX library augmented with an optimized graph search implementation based on Dial's algorithm. See also http://rpg.ifi.uzh.ch/docs/CVPR19workshop/CVPRW19_Mike_Davies.pdf

[Task 13] **Loihi**: constraint solver algorithm based on *G.A. Fonseca Guerra and S.B. Furber, Using Stochastic Spiking Neural Networks on SpiNNaker to Solve Constraint Satisfaction Problems. Front. Neurosci. 2017.* Tested on the Nahuku 32-chip system running NxSDK 0.98. **CPU**: Core i7-9700K with 32GB RAM running Coin-or Branch and Cut (https://github.com/coin-or/Cbc). Performance results are based on testing as of July 2020 and may not reflect all publicly available security updates.

Results may vary.