

1
2
3
4
5
6
7
8

Supplementary Information

Neuromorphic Intermediate Representation: A Unified Instruction Set for Interoperable Brain-Inspired Computing

9
10
11
12
13
14
15
16
17
18
19
20
21
22

Jens E. Pedersen ^{*†1}, Steven Abreu^{†2,3}, Matthias Jobst^{4,5}, Gregor Lenz⁶, Vittorio Fra⁷, Felix Christian Bauer⁸, Dylan Richard Muir⁸, Peng Zhou⁹, Bernhard Vogginger⁴, Kade Heckel¹⁰, Gianvito Urgese⁷, Sadasivan Shankar^{11,12}, Terrence C. Stewart¹³, Sadique Sheik⁸, and Jason K. Eshraghian¹⁴

¹KTH Royal Institute of Technology, Sweden

²CogniGron Center, University of Groningen, Netherlands

³Bernoulli Institute, University of Groningen, Netherlands

⁴Technische Universität Dresden, Germany

⁵Centre for Tactile Internet with Human-in-the-Loop, Dresden, Germany

⁶Neurobus, Toulouse, France

⁷Politecnico di Torino, Italy

⁸SynSense, Zurich, Switzerland

⁹LuxiTech Co. Ltd., Shenzhen, China

¹⁰University of Cambridge, UK

¹¹Stanford University, USA

¹²SLAC National Laboratory, Menlo Park, CA, USA

¹³National Research Council Canada

¹⁴University of California, Santa Cruz, USA

23
24
25
26
27
28

July 2024

A Neuron model parameterizations

We now turn to the implementation details of NIR equations across the different platforms, with a particular focus on leaky integrate-and-fire dynamics from experiments 1 and 3 in the main paper. We note that the idealized reset can be discretized by either a hard reset to θ_{thr} or a “soft” subtractive reset that retains the residual from a possible overshoot:

$$\text{Hard reset} \quad v_{t+1} = \begin{cases} v_t & v_t < \theta_{\text{thr}} \\ 0 & v_t \geq \theta_{\text{thr}} \end{cases} \quad (1)$$

$$\text{Subtractive reset} \quad v_{t+1} = \begin{cases} v_t & v_t < \theta_{\text{thr}} \\ v_t - \theta_{\text{reset}} & v_t \geq \theta_{\text{thr}} \end{cases} \quad (2)$$

*Corresponding author: jeped@kth.se

†equal contribution

29 The differences in the discretization of spiking neuron dynamics are detailed below; equations
 30 for Spx are not listed, as they follow the same implementation as `snnTorch`.

31 **A.1 LIF**

32 **NIR** The leaky integrate-and-fire model in NIR is given by:

$$\tau \dot{v} = v_{\text{leak}} - v + R i \quad (3)$$

33 where τ is the time constant, v_{leak} is the membrane leak, R is the resistance and $i(t)$ is the input
 34 current. Discretized using explicit Euler's method, this turns into:

$$v(t+1) = \left(1 - \frac{dt}{\tau}\right) v(t) + \frac{dt}{\tau} v_{\text{leak}} + \frac{dt}{\tau} R i(t) \quad (4)$$

35 where dt is the discretization timestep.

36 **Exact solution for constant input current** The course of the membrane potential v can be
 37 exactly calculated for a constant input current $i = i_0 = \text{const.}$ and initial condition $v(0) = v_0$ [1, eq.
 38 1.7]:

$$v(t) = v_{\text{leak}} + R i_0 \left(1 - \exp\left(-\frac{t}{\tau}\right)\right) + v_0 \exp\left(-\frac{t}{\tau}\right) \quad (5)$$

39 **Lava-dl** In `lava-dl`, we used the `CuBa-LIF` model as:

$$u(t) = (1 - \alpha_u) u(t-1) + i(t) \quad (6)$$

$$v(t) = (1 - \alpha_v) v(t-1) + u(t) \quad (7)$$

40 with $\alpha_u = 1$ to effectively turn it into a LIF model:

$$v(t) = (1 - \alpha_v) v(t-1) + i(t) \quad (8)$$

41 with

$$\alpha_v = \frac{dt}{\tau_{\text{mem}}} \quad (9)$$

42 With the constraints

$$R \frac{dt}{\tau_{\text{mem}}} = 1 \quad (10)$$

$$v_{\text{leak}} \frac{dt}{\tau_{\text{mem}}} = 0 \quad (11)$$

43 One may implement the first constraint (R) by hard-coding it into a linear layer, or into the voltage
 44 threshold. One may implement v_{leak} through a bias term, but this was not done in our experiments.

45 **Norse** The LIF model in `Norse` lacks the resistivity term, which we compensate for by scaling the
 46 input

$$v(t+1) = v(t) + \frac{dt}{\tau} (v_{\text{leak}} - v(t) + i) \quad (12)$$

47 **snnTorch** The LIF model in snnTorch is given by:

$$v(t+1) = \beta v(t) + i(t+1) \quad (13)$$

48 where β is the membrane potential decay rate. We have:

$$\beta = 1 - \frac{dt}{\tau} \quad (14)$$

49 And the constraints are:

$$R \frac{dt}{\tau} = 1 \quad (15)$$

$$v_{\text{leak}} \frac{dt}{\tau} = 0 \quad (16)$$

50 However, we can use a linear layer before the LIF in simulation to effectively implement the
51 resistance term. The weight matrix of this linear layer is given by:

$$W_{ij} = \begin{cases} R \frac{dt}{\tau} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \quad (17)$$

52 **SpiNNaker2** In py-spinnaker2 [2] the LIF model is given by:

$$v(t) = \alpha_{\text{decay}} v(t-1) + i(t) + i_{\text{offset}} \quad (18)$$

53 Two modes for the translation between NIR and SpiNNaker2 are supported:

54 In Exponential-Euler mode, the decay factor α_{decay} is calculated such that in case without input
55 ($i = 0$ and $i_{\text{offset}} = 0$) and with $v_{\text{leak}} = 0$ the voltage exactly matches the analytical expression for
56 continuous-time dynamics:

$$v(t+dt) = \alpha_{\text{decay}} v(t) \stackrel{!}{=} v(t) \exp\left(-\frac{dt}{\tau}\right) \quad (19)$$

$$\Rightarrow \alpha_{\text{decay}} = \exp\left(-\frac{dt}{\tau}\right) \quad (20)$$

57 The SpiNNaker2 threshold Θ is scaled to cope for the non-existing resistance variable:

$$\Theta = \frac{\theta_{\text{thr}}}{\left(1 - \exp\left(-\frac{dt}{\tau}\right)\right) R} \quad (21)$$

58 In Forward-Euler mode we match the discrete Euler update of the LIF model (Eq. 4) so that
59 decay factor and threshold become:

$$\alpha_{\text{decay}} = 1 - \frac{dt}{\tau} \quad (22)$$

$$\Theta = \theta_{\text{thr}} \frac{\tau}{dt R} \quad (23)$$

60 In both modes, the offset current i_{offset} is used to consider the parameter v_{leak} and the bias i_{bias}
61 from incoming **Affine** or **Conv2D** nodes:

$$i_{\text{offset}} = \frac{v_{\text{leak}}}{R} + i_{\text{bias}} \quad (24)$$

62 **Nengo** The default LIF model in Nengo’s CPU implementation is identical to the Exponential-
63 Euler model for SpiNNaker2 described above, except that it has a fixed threshold Θ . To compensate
64 for this we adjust the gain on each neuron (a multiplicative factor applied to all inputs).

65 **Rockpool** The LIF model in Rockpool is a time-discretized neuron membrane, with difference or
 66 update equations

$$\begin{aligned} V_{mem}[n] &= \alpha V_{mem}[n-1] \\ V_{mem}[n] &= V_{mem}[n] + S_{in}[n] + b + \sigma \zeta[n] \\ V_{mem}[n] &> \theta_{thr} \rightarrow S_{out}[n] = 1 \\ V_{mem}[n] &= V_{mem}[n] - \theta \end{aligned}$$

67 where $\alpha = \exp(-dt/\tau_{mem})$. Here V_{mem} is the floating-point membrane potential state; S_{in} and
 68 S_{out} are the incident and output spike trains respectively; $\sigma \zeta[n]$ is a scaled white noise process with
 69 mean 0 and std. dev. σ ; b is a bias current and θ is the firing threshold. We have neglected neuron
 70 indices in the above equations; all parameters are independently defined per neuron in Rockpool.

71 A.2 CuBa-LIF

72 **NIR** The current-based LIF model in NIR is given by:

$$\tau_{syn} \dot{u} = -u + w_{in} i \quad (25)$$

$$\tau_{mem} \dot{v} = v_{leak} - v + Ru \quad (26)$$

73 where $u(t)$ is the synaptic current, τ_{mem} the membrane time constant, τ_{syn} the synaptic time con-
 74 stant, and w_{in} the scaling factor of the current input $i(t)$. Discretized:

$$u(t+1) = \left(1 - \frac{dt}{\tau_{syn}}\right) u(t) + w_{in} \frac{dt}{\tau_{syn}} i(t+1) \quad (27)$$

$$v(t+1) = \left(1 - \frac{dt}{\tau_{mem}}\right) v(t) + \frac{dt}{\tau_{mem}} v_{leak} + R \frac{dt}{\tau_{mem}} u(t) \quad (28)$$

75 **Lava-dl** In lava-dl, the model is:

$$u(t) = (1 - \alpha_u) u(t-1) + i(t) \quad (29)$$

$$v(t) = (1 - \alpha_v) v(t-1) + u(t) \quad (30)$$

76 Such that:

$$\alpha_u = \frac{dt}{\tau_{syn}} \quad (31)$$

$$\alpha_v = \frac{dt}{\tau_{mem}} \quad (32)$$

77 With the constraints

$$w_{in} \frac{dt}{\tau_{syn}} = 1 \quad (33)$$

$$R \frac{dt}{\tau_{mem}} = 1 \quad (34)$$

$$v_{leak} \frac{dt}{\tau_{mem}} = 0 \quad (35)$$

78 Where the first constraint (w_{in}) can be hard-coded into a linear layer, or into the voltage threshold.

79 One may also implement v_{leak} through a bias term, but this was not done in our experiments.

80 **Norse** Norse discretizes the CuBa-LIF model as follows:

$$u(t+1) = u(t) - \frac{dt}{\tau_{syn}} i(t) \quad (36)$$

$$v(t+1) = v(t) + \frac{dt}{\tau_{mem}} (v_{leak} - v + u) \quad (37)$$

81 This deviates from the canonical formulation by lacking a weight and resistance term. The weight
82 term is introduced by scaling the input, while the resistance term can be introduced by adding a
83 weight factor between the synaptic (u) and somatic (v) parts.

snnTorch

$$u(t+1) = \alpha u(t) + i(t+1) \quad (38)$$

$$v(t+1) = \beta v(t) + u(t+1) \quad (39)$$

84 Leading to:

$$\alpha = 1 - \frac{dt}{\tau_{syn}} \quad (40)$$

$$\beta = 1 - \frac{dt}{\tau_{mem}} \quad (41)$$

85 With constraints:

$$w_{in} \frac{dt}{\tau_{syn}} = 1 \quad (42)$$

$$R \frac{dt}{\tau_{mem}} = 1 \quad (43)$$

$$v_{leak} \frac{dt}{\tau_{mem}} = 0 \quad (44)$$

86 Where a linear layer can be used with:

$$W_{ij} = \begin{cases} w_{in} \frac{dt}{\tau_{syn}} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \quad (45)$$

87 to replace the first constraint.

88 snnTorch also offers a tunable membrane potential threshold which is shared across a population
89 of neurons. As such, if W is a multiple of the identity matrix, the first constraint can also be satisfied
90 by simply scaling the membrane potential threshold accordingly.

91 **SpiNNaker2** In py-spinnaker2 [2] the CuBa-LIF model is given by:

$$I_{syn}(t) = \alpha_{syn} I_{syn}(t-1) + i(t) \quad (46)$$

$$v(t) = \alpha_{mem} v(t-1) + I_{syn}(t) + i_{offset} \quad (47)$$

92 As for LIF, two modes for the translation between NIR and SpiNNaker2 are supported:

93 In Exponential-Euler mode, the decay factors, the threshold and i_{offset} are calculated as follows:

$$\alpha_{mem} = \exp\left(-\frac{dt}{\tau_{mem}}\right) \quad (48)$$

$$\alpha_{syn} = \exp\left(-\frac{dt}{\tau_{syn}}\right) \quad (49)$$

$$\Theta = \theta_{thr} \left(\left(\left(1 - \exp\left(-\frac{dt}{\tau_{mem}}\right) \right) R \left(1 - \exp\left(-\frac{dt}{\tau_{syn}}\right) \right) w_{in} \right)^{-1} \right) \quad (50)$$

$$i_{offset} = \frac{v_{leak}}{R} \left(\left(\left(1 - \exp\left(-\frac{dt}{\tau_{syn}}\right) \right) w_{in} \right)^{-1} + i_{bias} \left(1 - \exp\left(-\frac{dt}{\tau_{syn}}\right) \right)^{-1} \right) \quad (51)$$

94 In Forward-Euler mode we match the discrete CuBa-LIF equations (36-37) so that SpiNNaker2
 95 parameters become:

$$\alpha_{\text{mem}} = 1 - \frac{dt}{\tau_{\text{mem}}} \quad (52)$$

$$\alpha_{\text{syn}} = 1 - \frac{dt}{\tau_{\text{syn}}} \quad (53)$$

$$\Theta = \theta_{\text{thr}} \frac{\tau_{\text{mem}}}{dtR} \frac{\tau_{\text{syn}}}{dtw_{\text{in}}} \quad (54)$$

$$i_{\text{offset}} = \frac{v_{\text{leak}}}{R} \frac{\tau_{\text{syn}}}{dtw_{\text{in}}} + i_{\text{bias}} \frac{dt}{\tau_{\text{syn}}} \quad (55)$$

96 We note that in both modes the integration of the bias into i_{offset} is not exact. Instead, the
 97 conversion is chosen such that for bias-only input the membrane voltage converges to the same value
 98 as in the NIR model.

99 **Rockpool** The Rockpool CuBa-LIF neuron is a time-discretized simulation of exponential LIF
 100 dynamics.

$$\begin{aligned} I_{\text{syn}}[n] &= I_{\text{syn}}[n-1] + S_{\text{in}}[n] + W_{\text{rec}}\mathbf{S}_{\text{out}}[n-1] \\ I_{\text{syn}}[n] &= \beta I_{\text{syn}}[n] \\ V_{\text{mem}}[n] &= \alpha V_{\text{mem}}[n-1] \\ V_{\text{mem}}[n] &= V_{\text{mem}}[n] + I_{\text{syn}}[n] + b_j + \sigma\zeta[n] \\ V_{\text{mem}}[n] > V_{\text{thr}} &\rightarrow S_{\text{out}}[n] = 1 \\ V_{\text{mem}}[n] &= V_{\text{mem}}[n] - V_{\text{thr}} \end{aligned}$$

101 where $\alpha =$ and $\beta = \exp(-dt/\tau_{\text{syn}})$. Here W_{rec} is the recurrent weight matrix for the layer of
 102 neurons; $\mathbf{S}_{\text{out}}[n]$ is the vector of spikes produced by the layer at timestep n ; and other symbols
 103 are as in the Rockpool LIF equations above. We have neglected neuron indices for simplicity; all
 104 neurons have independent state variables and parameters.

105 **Xylo** The Xylo CuBa-LIF neuron is a time-discretized low-bit-depth integer logic neuron, simu-
 106 lating an approximation of exponential LIF dynamics using integer bit-shift decay circuitry. In the
 107 following, the state variables I_{syn} and V_{mem} are 16-bit signed integers; W_{rec} are 8-bit signed integers;
 108 d_{syn} and d_{mem} are unsigned integer bit-shift decay parameters approximating $d \approx \log_2(\tau/dt)$; V_{thr}
 109 and b are 16-bit signed integers; S_{in} and S_{out} are unsigned integer number of events per dt .

$$I_{\text{syn}}[n] = I_{\text{syn}}[n-1] + S_{\text{in}}[n] + W_{\text{rec}}\mathbf{S}_{\text{out}}[n-1]$$

$$I_{\text{syn}}[n] = \begin{cases} I_{\text{syn}}[n] - I_{\text{syn}}[n] \gg d_{\text{syn}} & \text{if } I_{\text{syn}}[n] \gg d_{\text{syn}} > 0 \\ I_{\text{syn}}[n] - 1 & \text{otherwise} \end{cases} \quad (56)$$

$$V_{\text{mem}}[n] = \begin{cases} V_{\text{mem}}[n] - V_{\text{mem}}[n] \gg d_{\text{mem}} & \text{if } V_{\text{mem}}[n] \gg d_{\text{mem}} > 0 \\ V_{\text{mem}}[n] - 1 & \text{otherwise} \end{cases} \quad (57)$$

$$\begin{aligned} V_{\text{mem}}[n] &= V_{\text{mem}}[n] + I_{\text{syn}}[n] + b + \sigma\zeta[n] \\ V_{\text{mem}}[n] &> \theta_{\text{thr}} \rightarrow S_{\text{out}}[n] = 1 \\ V_{\text{mem}}[n] &= V_{\text{mem}}[n] - t\theta_{\text{thr}} \end{aligned}$$

110 where $x \gg n$ is the right-shift operator, shifting x by n bits.

111 The bit-shift decay operation in Eqs. 56 and 57 approximates exponential decay for V_{mem} and
 112 I_{syn} state variables. When the integer quantised decay parameters d_* obtained from τ_* are close to
 113 the corresponding floating-point values for d_* , this approximation causes only small deviations in
 114 dynamics. For values of V_{mem} and I_{syn} close to zero, where the bit-shift decay operation would not
 115 result in a decrease of the state variables, we include a unit decay per dt . We have neglected neuron
 116 indices in the above equations; all neurons have independent state variables and parameters.

117 B Relation to other intermediate representation and compi- 118 lation frameworks

119 To clarify the relation of NIR to other compilers, code generators, optimizers, and network modelling
 120 tools, we provide the following comparison table. Most of the frameworks in the table solve different
 121 problems. As such, the table provides high-level and independent descriptions of each framework to
 122 situate NIR in the literature.

Supplementary Table 1: Comparison of ONNX, MLIR, LLVM, PyNN, NeuroML.

Feature / Framework	ONNX (Open Neural Network Exchange)	MLIR (Multi-Level Intermediate Representation)	LLVM	PyNN	NeuroML	NIR
Scope	AI model formats and runtimes	Compiler infrastructure targeting CPUs, GPUs, and more	Compiler for the optimization and generation of machine code	Abstraction layer for neural network descriptions, focusing on spiking models	Models neural connectivity and dynamics, focusing on biological realism	Declaration of discrete- or continuous-time neural primitives
Language	Not applicable (format)	Based on LLVM, supports many frontends (e.g., C++, Python)	Primarily C++	Python	XML-based description	Not applicable (format)
Supported Platforms / Frameworks	TensorFlow, PyTorch, and other major AI frameworks	Integrates with TensorFlow, PyTorch, and others through LLVM	Wide range of hardware from desktops to embedded devices	NEURON, Brian, NEST	NEURON, Brian, jLEMS	Lava, Loihi, Nengo, Norse, Rockpool, Sinabs, snnTorch, SpiNNaker2, Speck, Spyx, Xylo
Typical Use Cases	Exchanging models between different AI frameworks	Optimizing performance of machine learning models at compile time	Low-level optimization of applications in various programming languages	Research and educational use in computational neuroscience	Detailed simulations of neuronal behavior and neural circuitry for research	Application-dependent models and also able to exchange models between platforms

123 **References**

- 124 [1] Wulfram Gerstner, Werner M. Kistler, Richard Naud, and Liam Paninski. Neuronal Dynamics:
125 From Single Neurons to Networks and Models of Cognition. Cambridge University Press, Cam-
126 bridge, 2014.
- 127 [2] Bernhard Vogginger, Florian Kelber, Matthias Jobst, Yexin Yan, Pascal Gerhards, Martin Weih,
128 and Mahmoud Akl. py-spinnaker2, November 2023. <https://doi.org/10.5281/zenodo.10202110>.