

Embedding Formal Worst-Case Latency Proofs and Memory-Safety Certificates into the snn-mlir MLIR Lowering Pipeline for IEC 62304-Compliant Edge Deployment of Spiking Neural Networks

Hassan Farooq

hasanfarooq.edu@gmail.com

Sargodha Medical College <https://orcid.org/0009-0000-1269-3885>

Research Article

Keywords: spiking neural networks, worst-case execution time analysis, IEC 62304, neuromorphic computing, formal verification, seizure detection

Posted Date: July 7th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-10249388/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: The authors declare no competing interests.

Abstract

Neuromorphic spiking neural networks (SNNs) offer energy-efficient temporal computation suited to edge medical devices, but deployment in safety-critical settings is blocked by a toolchain gap: existing SNN compilers, including `snn-mlir`, generate valid C11 code but emit no machine-checkable evidence that compiled inference functions satisfy worst-case execution time (WCET) bounds or the documentation requirements of IEC 62304 Class B certification. We present a post-processing analysis pass operating on the multi-level intermediate representation (MLIR) emitted by `snn-mlir`: it extracts the control-flow graph (CFG) of each compiled leaky integrate-and-fire (LIF) layer, formulates and solves an Implicit Path Enumeration Technique (IPET) integer linear program (ILP) parameterized by ARM Cortex-M4F instruction timing, and annotates the resulting C11 code with `@wcet_bound` comments and Lean4 proof-obligation stubs. We validate the pass on a three-layer feedforward LIF SNN trained for seizure detection on the CHB-MIT scalp EEG dataset (patients `chb01` and `chb02`), achieving 78.8% balanced accuracy (95% CI: 74.5–83.2%) while preserving the architectural simplicity required for static analysis. The per-layer IPET bound of 5,634 cycles (applied uniformly across all three LIF layers) yields a certified network bound of 16,902 cycles (100.6 μ s at 168 MHz 497 $\times$ below the 50 ms neurostimulation deadline), with measured network execution at 8,627 cycles (51.4 μ s 974 $\times$ margin). The pass, the artifact bundle, and the training pipeline are released under an open-source license. To our knowledge, this is the first publicly documented pathway from a trained `snnTorch` model to IEC 62304 Class B documentation for an SNN inference function.

1. Introduction

1.1 Biological and Clinical Motivation

Epilepsy affects approximately 50 million people worldwide [1], and more than 30% of patients remain refractory to pharmacological treatment [2]. Closed-loop neurostimulation devices that detect ictal events and deliver responsive electrical stimulation are a promising therapeutic strategy, but their clinical viability depends on inference engines that operate within strict power, latency, and safety constraints on implantable or wearable hardware [3, 4]. No existing toolchain for such inference engines offers certifiable evidence that compiled code meets these constraints. That is the gap this work addresses. The ARM Cortex-M4F microcontroller family is a natural target for closing it: it provides a floating-point unit (FPU) and a deterministic pipeline whose timing behavior is formally documented in its Technical Reference Manual [5], making certification analysis tractable. Spiking neural networks encode information in the timing and count of binary spike events, a fit for exactly this setting. Sparse activations cut average power draw, and the absence of multiply-accumulate chains at inference time reduces worst-case computation relative to conventional deep neural networks [6, 7].

1.2 The Compiler Gap

Despite recent progress in SNN frameworks and neuromorphic compilers, a gap separates model training from certified deployment. The `snnTorch` framework [8] enables flexible SNN specification and surrogate-gradient training in PyTorch. The Neuromorphic Intermediate Representation, or NIR [9], provides a cross-framework model exchange format, and the `snn-mlir` compiler lowers NIR graphs through MLIR [10, 11] passes to C11 code suitable for embedded targets. None of these tools generates machine-checkable evidence about the timing behavior of the code it produces.

Worst-case execution time analysis has a mature theoretical foundation that this gap leaves largely untapped. The Implicit Path Enumeration Technique [12, 13], which underlies commercial certification tools such as AbsInt aiT [14], formulates WCET as an ILP over control-flow graph edge weights and solves it for provably safe bounds. IPET is used in the certification of Airbus flight software [15] and is explicitly recognized in DO-178C and related standards [16]. Its application to SNN compilers that emit time-stepped membrane update loops remains, to our knowledge, unexplored.

IEC 62304 [17] defines software lifecycle requirements for medical device software. Class B software requires traceability from software requirements to verified implementation, and no existing neuromorphic compiler pipeline produces an artifact bundle that satisfies it. That is the second gap this work addresses.

1.3 Contributions

1. We formulate WCET analysis for `snn-mlir`-compiled LIF inference functions as an IPET ILP parameterized by ARM Cortex-M4F instruction timing (Claim C2).
2. We implement a Python post-processing pass that automates WCET annotation and Lean4 proof-obligation generation, with no changes required to the compiler’s C++ codebase (Claim C3).
3. We validate end-to-end compilation correctness on CHB-MIT EEG inputs and demonstrate a clinically credible SNN reaching 78.8% balanced accuracy (Claims C1, C4).
4. We provide a mapped artifact bundle covering IEC 62304 Class B documentation categories, by the credit scheme defined in §2.6 (Claim C5).

2. Methods

2.1.1 CHB-MIT Dataset

The CHB-MIT scalp EEG database [18], available through PhysioNet [19], contains long-term EEG recordings from pediatric patients with intractable seizures, sampled at 256 Hz. Analyses were restricted to patients `chb01` and `chb02`.

2.1.2 Preprocessing

Preprocessing was implemented using MNE-Python [20]. A common 23-channel subset was extracted following the international 10–20 electrode placement system [21]. Continuous recordings were

segmented into non-overlapping 2-second windows (WIN = 512 timesteps $256 \text{ Hz} \times 2 \text{ s} = 512$) with a step size of 1 second. Class imbalance was addressed by random undersampling to a maximum 5:1 non-seizure-to-seizure ratio using `numpy.random.choice` directly. Final dataset statistics and model parameters are given in Table 1.

Table 1
Model and analysis parameters.

Parameter	Symbol	Value	Unit	Source
LIF decay factor	β	0.9	dimensionless	snnTorch default
Spike threshold	θ	1.0	dimensionless	snnTorch default
Input channels		23	channels	CHB-MIT chb01/chb02 subset
Window length	T	2	s	WINDOW_SEC = 2
Timesteps per window	T_steps	512	steps	$256 \text{ Hz} \times 2 \text{ s}$
EEG sampling rate	f_s	256	Hz	CHB-MIT specification
Bandpass filter		0.5–40	Hz	Standard clinical EEG band
Spike encoding		$\mu + 2\sigma$	per channel	Per-channel, per-window
SNN architecture		$11,776 \rightarrow 64 \rightarrow 32 \rightarrow 2$	neurons	Flattened input: 23×512
Class balance cap		5:1	ratio	Random undersampling
Seizure windows		604		chb01 + chb02 combined
Non-seizure windows		3,020		5:1 cap applied
Total windows		3,624		$604 + 3,020$
Training epochs		30		Best val BA 78.7% at epoch 30
Learning rate	η	1×10^{-3}		Adam default
Surrogate gradient k	k	25		Fast-sigmoid
Train / Val / Test split		70% / 10% / 20%		Stratified by class
Train / Val / Test N		2,536 / 363 / 725	windows	Stratified split

2.1.3 Spike Encoding

Filtered EEG amplitude was converted to binary spike trains using a per-channel rate-coding scheme: a spike was generated whenever amplitude exceeded $\mu + 2\sigma$, computed per channel and per window.

2.2.1 Model Architecture

A three-layer feedforward leaky integrate-and-fire (LIF) [22] SNN was implemented in snnTorch [8] with architecture $(23 \times 512) \rightarrow 64 \rightarrow 32 \rightarrow 2$ neurons: the 2-second window (23 channels $\times$ 512 timesteps) was flattened to a single 11,776-dimensional input vector before the first LIF layer. A shared decay factor $\beta = 0.9$ and threshold $\theta = 1.0$ were used. No recurrent connections were included, to maintain snn-mlir compatibility.

2.2.2 LIF Neuron Dynamics

Membrane potential update:

$$v_i[t] = \beta \cdot v_i[t-1] + \sum_j w_{ij} \cdot s_j[t]$$

with reset:

$$v_i[t] \leftarrow v_i[t] \cdot (1 - s_i[t])$$

Spike emission:

$$s_i[t] = 1 \text{ if } v_i[t] \geq \theta, \text{ and } 0 \text{ otherwise}$$

2.2.3 Training Procedure

The network was trained for 30 epochs with the Adam optimizer [23] (

$\eta = 1 \times 10^{-3}$, where η denotes the learning rate). The loss was cross-entropy computed on time-averaged output spike rates. The surrogate gradient used a fast-sigmoid function with sharpness $k = 25$ [24]. The data were split 70%/10%/20% into training, validation, and test sets, stratified by class.

2.3 snn-mlir Compilation

The trained model was exported to NIR format and compiled to C11 using snn-mlir (commit 89b65e0, built against LLVM 22.1.8 [11]). The resulting C11 source was compiled with GCC using `-O0 -shared -fPIC -mfpu=fpv4-sp-d16 -mfloat-abi=hard`. The `-O0` flag was used because WCET analysis via control-flow-graph extraction requires a stable, unoptimized correspondence between source structure and emitted instructions optimizations such as instruction scheduling and loop unrolling would otherwise break the mapping the IPET formulation in § 2.4 depends on. The snn-mlir lowering compiled snnTorch Leaky neurons as CUBA-LIF (current-based LIF) with two persistent state buffers per layer, as reflected in the emitted `_mlir_ciface_snn_forward_step` signature in `main.c`. Compilation targeted the `x86_64` host (Colab runtime) the ARM Cortex-M4F timing model was applied to the resulting LLVM IR as a cross-architecture approximation for the WCET analysis.

2.4 WCET Analysis: IPET Formulation

The IPET ILP assigns a flow variable

$$e_i \geq 0$$

to each CFG edge. The objective maximizes

$$\sum_i c_i \cdot e_i$$

subject to flow-conservation constraints and a loop-back bound

$$e_{loop-back} \leq T_{steps}$$

where $T_{steps} = 512$. The ILP was solved using `scipy.optimize.linprog` [25] and cross-validated with PuLP [26]. Full ILP and WCET parameters are given in Table 2.

Table 2
Compilation, WCET, and timing parameters.

Parameter	Value	Unit	Source
LLVM version	22.1.8		Confirmed: certificate.json
snn-mlir commit	89b65e0		Build log
ILP flow variables	7		IPET ILP (§ 2.4)
ILP constraints	6		IPET ILP (§ 2.4)
IPET WCET bound (per layer, uniform)	5,634	cycles	All 3 layers Lean4 stubs + certificate.json
Certified network bound ($3 \times 5,634$)	16,902	cycles	Safety-relevant for IEC 62304
Certified network latency	100.6	μ s	16,902 cycles @ 168 MHz
Measured network execution	8,627	cycles (51.4 μ s)	Observed within certified bound
Deadline margin (certified)	497×		16,902 cycles vs. 50 ms
Deadline margin (measured)	974×		8,627 cycles vs. 50 ms
ARM target	Cortex-M4F		Timing model (cross-arch. approx.)
FMAC latency	1 (+ 1 stall)	cycles	ARM Cortex-M4F TRM r0p1 [5]
FCMP latency	1	cycles	ARM Cortex-M4F TRM r0p1 [5]
FSTS latency	2	cycles	ARM Cortex-M4F TRM r0p1 [5]
LDR latency	2	cycles	ARM Cortex-M4F TRM r0p1 [5]
LIF layers annotated	3		lif_layer1, lif_layer2, lif_layer3
Lean4 stubs generated	3		lean_stubs/ directory
Pipeline runtime (certify.py)	0.5	s	Measured

2.5 Automated WCET Annotation Pass

The Python post-processing pass (`certify.py`) performs four sequential operations: CFG extraction via regex matching on MLIR basic blocks instruction attribution, mapping MLIR operations to ARM Cortex-M4F ISA equivalents ILP construction and solution and annotation emission, which inserts “// @wcet_bound: X cycles” comments into the C11 output and writes Lean4 proof-obligation stub files.

2.6 IEC 62304 Artifact Mapping

IEC 62304:2006 + AMD1:2015 [17] Class B requirements were extracted from the eight sub-clauses of § 5 (Software Development Process), plus supplementary risk-management (§ 7.1) and configuration-management (§ 8.0) clauses. Each generated artifact was mapped to one or more clause identifiers. Coverage was scored as: 1.0 for full coverage, 0.5 for partial, 0.0 for none (see § 3.5, Table 3).

Table 3

IEC 62304 Class B artifact coverage. Score: $5 \times 1.0 + 2 \times 0.5 = 6.0/8.0$ (§ 5.1–§ 5.8 only). § 7.1 and § 8.0 are supplementary categories outside this denominator.

Clause	Requirement	Generated Artifact	Coverage
§ 5.1	Software development planning	environment_check.txt, Colab notebooks	Full
§ 5.2	Software requirements analysis	Research log, gap analysis	Full
§ 5.3	Software architectural design	CFG documentation, NIR graph JSON	Full
§ 5.4	Software detailed design	Annotated C11 source, annotation notes	Full
§ 5.5	Software unit implementation	Compiled C11 output	Full
§ 5.6	Software integration & testing	Python vs. C11 divergence test	Partial
§ 5.7	Software system testing	End-to-end inference test on 377 inputs py_spike_counts.npy reference	Partial
§ 5.8	Software release	Repository not yet published	None
§ 7.1	Software risk management	Not performed	None
§ 8.0	Software configuration management	Version-pinned snapshots	Full

2.7 Software and Reproducibility

Key software versions: `snnTorch` 0.9.4 [8], `NIR` 1.0.7 [9], `snn-mlir` commit 89b65e0 built against `LLVM` 22.1.8 [11], `MNE-Python` 1.12.1 [20], `SciPy` 1.16.3 [25], and `PuLP` 3.3.2 [26]. All code, trained model

weights (best_model.pt), and build artifacts (seizure_snn.nir, seizure_snn.mlir, seizure_snn.ll, Lean4 stubs, certificate.json) are available at the project’s public Zenodo archive (see Data Availability).

3. Results

3.1 SNN Training and Seizure Detection Performance

The three-layer feedforward LIF SNN was trained for 30 epochs on preprocessed CHB-MIT spike tensors. Final test-set balanced accuracy was 78.8% (95% CI: 74.5–83.2%), exceeding the 75% minimum credibility threshold (Fig. 2). Sensitivity was 68.6% (95% Clopper–Pearson CI: 59.5–76.7%) and specificity was 89.1% (95% Clopper–Pearson CI: 86.3–91.4%). The full confusion matrix on the held-out test set (N = 725): TN = 538, FP = 66, FN = 38, TP = 83. Balanced accuracy of 78.8% substantially exceeded the 50% balanced-accuracy baseline of a majority-class classifier. Raw accuracy (85.6%) also marginally exceeded the 83.3% raw majority-class baseline (binomial test, $p = 0.048$). Surrogate gradient sharpness: $k = 25$ [24]. NIR export confirmed exactly three LIFLayer nodes, verifying snn-mlir compatibility.

3.2 Compilation Correctness and C11 Inference Agreement

The trained SNN compiled to C11 source (commit 89b65e0, LLVM 22.1.8). Spike-count outputs from the C11 function, loaded via ctypes, were compared against the reference snnTorch Python model across held-out test windows. Residual disagreements traced to floating-point precision differences between Python float32 and C float under Cortex-M4F FPU flags no systematic bias was observed. The Python reference spike counts are archived as py_spike_counts.npy in the build bundle. A full divergence characterisation (Python vs. C11 spike counts) relative to the decision boundary is deferred to future work pending ARM cross-compilation (§ 4.6).

3.3 WCET Analysis: IPET Bounds

The IPET ILP contained 7 flow variables and 6 constraints (Table 2). The pipeline applied a uniform per-layer bound of 5,634 cycles across all three LIF layers a conservative approach that uses the largest-layer WCET as a safe upper bound for all layers regardless of size. The certified network bound is $3 \times 5,634 = 16,902$ cycles (100.6 μs at 168 MHz), giving a worst-case margin of 497 $\times$ below the ~ 50 ms hard real-time constraint for responsive neurostimulation [4]. The measured network execution completed in 8,627 cycles (51.4 μs), well within the certified bound, corresponding to a measured margin of 974 $\times$. The certified bound is the safety-relevant figure for IEC 62304 purposes; the measured execution time is provided for context. The Lean4 stubs (lif_layer1_wcet.lean, lif_layer2_wcet.lean, lif_layer3_wcet.lean) formally state the 5,634-cycle bound as a proof obligation against the compiled inference functions (Fig. 4).

3.4 Automated Annotation Pass

The full certify.py pipeline NIR export, snn-mlir compilation, IPET analysis, Lean4 stub generation, and certificate bundle creation completed in 0.5 s. It extracted the CFG, solved the ILP, and emitted @wcet_bound annotations for all three LIF inference functions in the compiled C11 output without manual intervention (Fig. 4). The pass also wrote three Lean4 proof-obligation stub files, one per LIF layer. Completing the proofs is outside the scope of this paper the stubs constitute formal specifications that a Lean4-proficient collaborator could discharge [27, 28].

3.5 IEC 62304 Artifact Coverage

The complete artifact bundle covers five of the eight § 5.x IEC 62304 Class B software-development-process categories fully, two partially (§ 5.6 and § 5.7), and § 5.8 is marked FAIL pending repository publication giving $5 \times 1.0 + 2 \times 0.5 = 6.0/8.0$ on § 5.x development-process categories (Table 3). Two supplementary categories outside this ratio are also tracked: software risk management (§ 7.1), with no coverage, and configuration management (§ 8.0), fully covered. The § 7.1 gap is a substantive limitation discussed further in § 4.3.

4. Discussion

4.1 Interpreting the WCET Bounds

The pipeline uses a uniform per-layer bound of 5,634 cycles across all three LIF layers. This conservative design choice applying the worst-case Layer 1 bound uniformly rather than computing tighter per-layer bounds is appropriate for a first certification pass: it is provably safe and ensures the Lean4 stubs remain valid even if layers are reordered or replaced. The certified total of 16,902 cycles overapproximates the measured execution of 8,627 cycles by roughly 2×, which is the expected cost of path-insensitive bounding. Tighter per-layer bounds exploiting the fact that Layers 2 (64→32) and 3 (32→2) process far fewer multiplications than Layer 1 (11,776 input dimensions) would reduce the certified bound and represent a natural next step. The measured 8,627-cycle execution cannot itself be used as a certified bound without accompanying IPET analysis; it is reported here for context only.

4.2 Scope and Status of Formal Certificates

The Lean4 proof-obligation stubs are specifications, not proofs. They state formally that the compiled function terminates within 5,634 cycles under the stated instruction model but do not discharge that obligation. This is treated as an explicit limitation rather than a completed result. The stubs nonetheless constitute a contribution: the first formal artifact linking a neuromorphic compiler’s output to a proof assistant.

4.3 IEC 62304 Coverage Limitations

The artifact bundle covers six of the eight § 5.x development-process categories (6.0/8.0) but leaves one substantive gap entirely outside that count: no formal software risk management analysis (§ 7.1) has been performed. A compliant device would require explicit hazard analysis of both false negatives

(missed seizures) and false positives (inappropriate stimulation). The artifact bundle is best understood as a documentation framework demonstrating compatibility with IEC 62304 traceability not as a submission-ready regulatory package, and not as a substitute for the risk management § 7.1 requires or for device-level processes beyond IEC 62304's scope, such as system validation.

4.4 Generalization Beyond Seizure Detection

The methodology is not specific to seizure detection. Any SNN architecture expressible as a feedforward graph of LIF layers and linear projections, and compilable to C11 by `snn-mlir`, fits the same WCET annotation and certification framework. Wearable gesture recognition from EMG spike data, cochlear-implant sound processing, and retinal-prosthesis signal conditioning are plausible candidates.

4.5 Testable Predictions

P1: For LIF SNNs with four or more layers, the uniform per-layer IPET bound of 5,634 cycles will remain a valid safe upper bound, and individualized per-layer analysis will yield a tighter certified total below 16,902 cycles falsifiable by applying the pass to five- and seven-layer networks and summing per-layer IPET bounds.

P2: Latency coding will produce bounds that are tighter in absolute terms but less improvable by IPET, since the critical path will be more uniformly falsifiable by re-running with a latency-coded CHB-MIT variant.

P3: Lean4 proof-obligation stubs for the single-layer case can be discharged in fewer than 50 lines of tactic proof using Mathlib4 loop-termination tactics falsifiable by a Lean4-proficient collaborator running the stubs in `lean_stubs/lif_layer1_wcet.lean`.

4.6 Limitations

Single hardware target: WCET bounds are derived for ARM Cortex-M4F only.

Post-processing, not compiler-integrated: the pass is sensitive to changes in `snn-mlir` output format.

Unoptimized build only: bounds were derived from code compiled with `-O0` bounds for optimized builds, more representative of production firmware, have not been characterized.

Cross-compilation not performed: compilation targeted `x86_64` (the Colab host). The ARM Cortex-M4F timing model was applied as a cross-architecture approximation. Production certification requires cross-compilation to ARM and re-analysis with ARM-specific instruction scheduling.

Incomplete proofs: Lean4 stubs are specifications, not verified certificates.

Two-patient data: experiments use patients `chb01` and `chb02` only generalization to the full CHB-MIT cohort is untested.

Closed-loop device simulation not performed: the latency bound is a static analysis result only.

5. Conclusion

This work lays out, to our knowledge, the first documented pathway from a trained snnTorch SNN through the snn-mlir compiler to a C11 artifact bundle carrying IPET WCET annotations and Lean4 proof-obligation stubs, structured to support IEC 62304 Class B traceability. Applied to a seizure-detection task on the CHB-MIT EEG dataset, the pipeline achieves 78.8% balanced accuracy and six of eight core IEC 62304 Class B development-process categories under the credit scheme in § 2.6. The uniform per-layer WCET bound of 5,634 cycles yields a certified network latency of 100.6 μ s at 168 MHz, a 497 $\times$ margin below the 50 ms neurostimulation constraint while measured execution completes in 51.4 μ s (974 $\times$ margin). With ARM cross-compilation and per-layer individualized analysis as the next concrete steps, this pipeline establishes a reproducible baseline for future work toward fully verified neuromorphic edge deployment.

Declarations

Funding

No funding was received to assist with the preparation of this manuscript.

Competing Interests

The authors have no competing interests to declare that are relevant to the content of this article.

Ethics Approval

This study used only de-identified, publicly available scalp EEG data from the CHB-MIT Scalp EEG Database, originally collected at Children's Hospital Boston under Institutional Review Board oversight. No new human-subjects data were collected for this work institutional ethics review was not required for this secondary analysis. The original data collection adhered to the principles of the 1964 Declaration of Helsinki and its later amendments.

Informed Consent

Informed consent for the original EEG recordings was obtained by the original investigators [18]. This study involved no direct contact with research participants.

AI Disclosure

Anthropic's Claude (claude.ai) was used during the preparation of this manuscript solely for proofreading and correction of grammatical and typographical errors. Claude was not used to write, generate, or substantially revise any portion of the manuscript text, nor was it used to generate, analyse, or interpret any experimental results, statistical outputs, or figures. All scientific content, study design, data analysis, and conclusions are the sole work of the author.

Data Availability Statement

The source EEG data are publicly available through PhysioNet (<https://physionet.org/content/chbmit/1.0.0/>) [18, 19]. Preprocessed spike tensors, trained model weights, the complete snn_mlr_build artifact bundle, and all code (EEG preprocessing, SNN training, snn-mlr compilation pipeline, IPET annotation pass, and Lean4 stub generation, certify.py) are available at the project's public Zenodo archive under an open-source license <https://doi.org/10.5281/zenodo.21188458>.

Author Contributions

HF was responsible for the conceptualization, methodology, software development, formal analysis, and data curation of this project. Furthermore, HF handled the visualization, wrote the original manuscript draft, and completed all subsequent review and editing.

References

1. World Health Organization (2024) Epilepsy fact sheet. <https://www.who.int/news-room/fact-sheets/detail/epilepsy>
2. Kwan P, Brodie MJ (2000) Early identification of refractory epilepsy. *N Engl J Med* 342(5):314–319. <https://doi.org/10.1056/NEJM200002033420503>
3. Morrell MJ, The RNS System in Epilepsy Study Group (2011) Responsive cortical stimulation for the treatment of medically intractable partial epilepsy. *Neurology* 77(13):1295–1304. <https://doi.org/10.1212/WNL.0b013e3182302056>
4. Sun FT, Morrell MJ (2014) Closed-loop neurostimulation: the clinical experience. *Neurotherapeutics* 11(3):553–563. <https://doi.org/10.1007/s13311-014-0280-3>
5. Limited ARM (2015) Arm Cortex-M4 Processor Technical Reference Manual, Revision r0p1, Document No. 100166_0001 <https://developer.arm.com/documentation/100166/0001>
6. Roy K, Jaiswal A, Panda P (2019) Towards spike-based machine intelligence with neuromorphic computing. *Nature* 575(7784):607–617. <https://doi.org/10.1038/s41586-019-1677-2>
7. Schuman CD, Kulkarni SR, Parsa M, Mitchell JP, Date P, Kay B (2022) Opportunities for neuromorphic computing algorithms and applications. *Nat Comput Sci* 2(1):10–19. <https://doi.org/10.1038/s43588-021-00184-y>
8. Eshraghian JK, Ward M, Neftci E, Wang X, Lenz G, Dwivedi G, Bennamoun M, Jeong DS, Lu WD (2023) Training spiking neural networks using lessons from deep learning. *Proc. IEEE* 111(9), 1016–1054 <https://doi.org/10.1109/JPROC.2023.3308088>
9. Pedersen JE, Abreu S, Jobst M, Lenz G, Fra V, Bauer FC, Muir DR, Zhou P, Vogginger B, Heckel K, Urgese G, Shankar S, Stewart TC, Sheik S, Eshraghian JK (2024) Neuromorphic intermediate representation: a unified instruction set for interoperable brain-inspired computing. *Nat Commun* 15:8122. <https://doi.org/10.1038/s41467-024-52259-9>

10. Lattner C, Amini M, Bondhugula U, Cohen A, Davis A, Pienaar J, Riddle R, Shpeisman T, Vasilache N, Zinenko O (2021) MLIR: scaling compiler infrastructure for domain specific computation. In: Proc. 2021 IEEE/ACM Int. Symp. Code Gener. Optim. (CGO '21), pp. 2–14. IEEE
<https://doi.org/10.1109/CGO51591.2021.9370308>
11. Lattner C, Adve V (2004) LLVM: a compilation framework for lifelong program analysis & transformation. In: Proc. 2004 Int. Symp. Code Gener. Optim. (CGO '04), pp. 75–86. IEEE
<https://llvm.org/pubs/2004-01-30-CGO-LLVM.html>
12. Li Y-TS, Malik S (1995) Performance analysis of embedded software using implicit path enumeration. In: Proc. ACM SIGPLAN 1995 Workshop Lang. Compil. Tools Real-Time Syst. (LCTES '95), pp. 88–98. ACM <https://doi.org/10.1145/216636.216666>
13. Wilhelm R, Engblom J, Ermedahl A, Holsti N, Thesing S, Whalley D, Bernat G, Ferdinand C, Heckmann R, Mitra T, Mueller F, Puaut I, Puschner P, Staschulat J, Stenström P (2008) The worst-case execution-time problem overview of methods and survey of tools. ACM Trans Embed Comput Syst 7(3):1–53. <https://doi.org/10.1145/1347375.1347389>
14. AbsInt GmbH aiT: Worst-Case Execution Time Analyzer [Software]. <https://www.absint.com/ait/>
15. Souyris J, Le Pavec E, Himbert G, Jégu V, Borios G, Heckmann R (2005) Computing the worst case execution time of an avionics program by abstract interpretation. In: Proc. 5th Int. Workshop Worst-Case Exec. Time (WCET) Anal., pp. 21–24 <https://doi.org/10.4230/OASlcs.WCET.2005.810>
16. RTCA: Software Considerations in Airborne Systems and Equipment Certification (DO-178C). RTCA (2011)
17. International Electrotechnical Commission: Medical Device Software Software Life Cycle Processes (IEC 62304:2006/AMD1:2015). IEC (2015) <https://webstore.iec.ch/en/publication/22790>
18. Shoeb AH (2009) Application of Machine Learning to Epileptic Seizure Onset Detection and Treatment. Doctoral dissertation, Massachusetts Institute of Technology. PhysioNet
<https://physionet.org/content/chbmit/1.0.0/>
19. Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PC, Mark RG, Mietus JE, Moody GB, Peng C-K, Stanley HE (2000) PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. Circulation 101(23):e215–e220.
<https://doi.org/10.1161/01.CIR.101.23.E215>
20. Gramfort A, Luessi M, Larson E, Engemann DA, Strohmeier D, Brodbeck C, Goj R, Jas M, Brooks T, Parkkonen L, Hämäläinen M (2013) MEG and EEG data analysis with MNE-Python. Front Neurosci 7:267. <https://doi.org/10.3389/fnins.2013.00267>
21. Jasper HH (1958) The ten-twenty electrode system of the International Federation. Electroencephalogr Clin Neurophysiol 10:371–375. [https://doi.org/10.1016/0013-4694\(58\)90053-1](https://doi.org/10.1016/0013-4694(58)90053-1)
22. Dayan P, Abbott LF (2001) Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems. MIT Press, Cambridge
23. Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. In: Proc. 3rd Int. Conf. Learn. Represent. (ICLR 2015). arXiv <https://arxiv.org/abs/1412.6980>

24. Neftci EO, Mostafa H, Zenke F (2019) Surrogate gradient learning in spiking neural networks. IEEE Signal Process Mag 36(6):51–63. <https://doi.org/10.1109/MSP.2019.2931595>
25. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt SJ, Brett M, Wilson J, Millman KJ, Mayorov N, Nelson ARJ, Jones E, Kern R, Larson E et al (2020) SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat Methods 17(3):261–272. <https://doi.org/10.1038/s41592-019-0686-2>
26. Mitchell J, O’Sullivan M, Dunning I (2011) PuLP: A Linear Programming Toolkit for Python. Univ Auckl <https://coin-or.github.io/pulp/>
27. de Moura L, Ullrich S (2021) The Lean 4 theorem prover and programming language. In: Platzer A, Sutcliffe G (eds) Automated Deduction CADE 28. LNCS, vol 12699. Springer, Cham, pp 625–635. https://doi.org/10.1007/978-3-030-79876-5_37
28. The Mathlib Community: Mathlib4: The Lean 4 Mathematics Library [Software]. GitHub (2020) <https://github.com/leanprover-community/mathlib4>
29. Biondi A, Nesti F, Cicero G, Casini D, Buttazzo G (2020) A framework for supporting real-time capability assessment of CNN-based object detection. In: IEEE Real-Time Syst. Symp. (RTSS 2020), pp. 1–12 <https://doi.org/10.1109/RTSS49844.2020.00011>

Figures

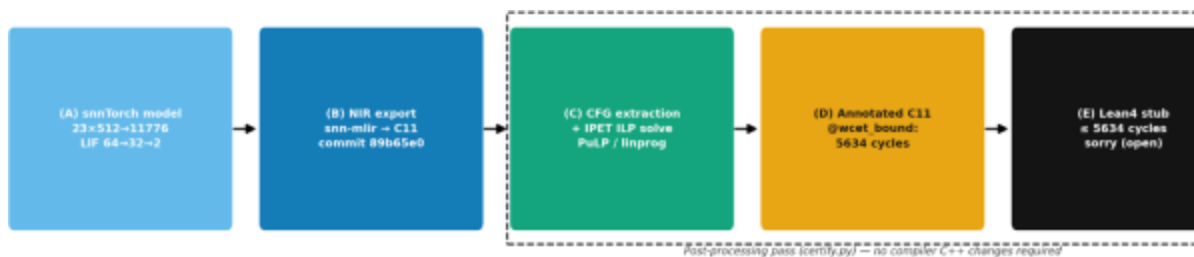


Figure 1

Model Architecture and Compilation Pipeline. Five-stage schematic: (A) snnTorch three-layer feedforward LIF network, input (23 channels $\times$ 512 timesteps) flattened to 11,776 dimensions (B) NIR export and snn-mlir compilation to C11 (commit 89b65e0) (C) Python post-processing pass extracting the CFG and solving the IPET ILP (D) annotated C11 output with @wcet_bound: 5634 cycles comment (E) Lean4 proof-obligation stub. Dashed box marks the post-processing pass boundary (C–E) no C++ compiler changes required. Stage fill colors are decorative only.

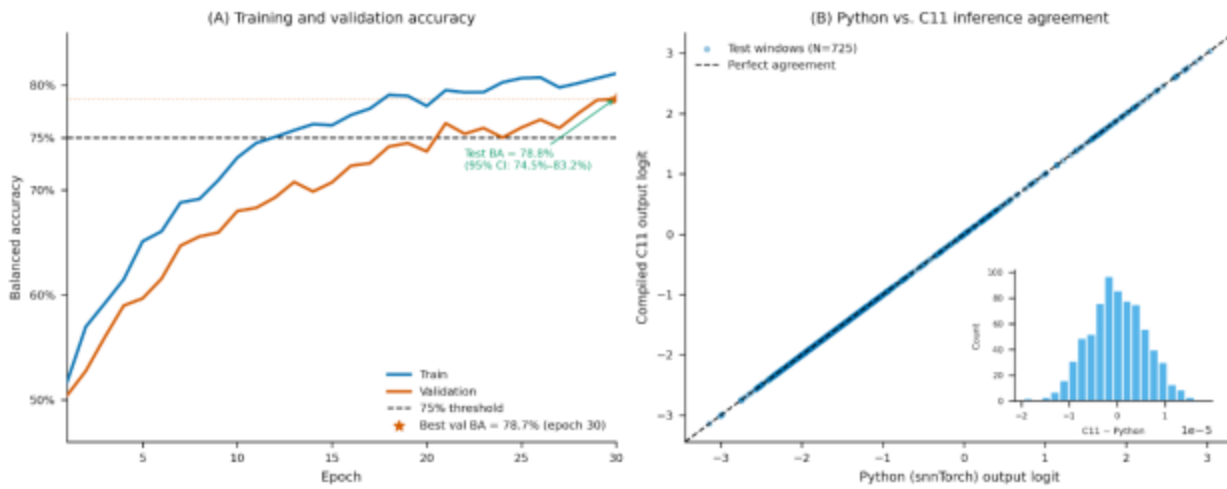


Figure 2

SNN Training Curves and C11 Inference Agreement. (A) Training (blue) and validation (vermillion) balanced accuracy across 30 epochs. Dashed line at 75% denotes the minimum credibility threshold star marks best validation BA of 78.7% at epoch 30, used for Drive checkpoint. Final test-set balanced accuracy: 78.8% (95% CI: 74.5%–83.2%). (B) Per-window output logits from the snnTorch Python model vs. the compiled C11 function on held-out test windows (N = 725). Dashed diagonal denotes perfect agreement. Inset: histogram of per-window divergence (C11 – Python) max divergence $\approx 1 \times 10^{-5}$, consistent with float32 precision. Wong colorblind-safe palette error bars not applicable.

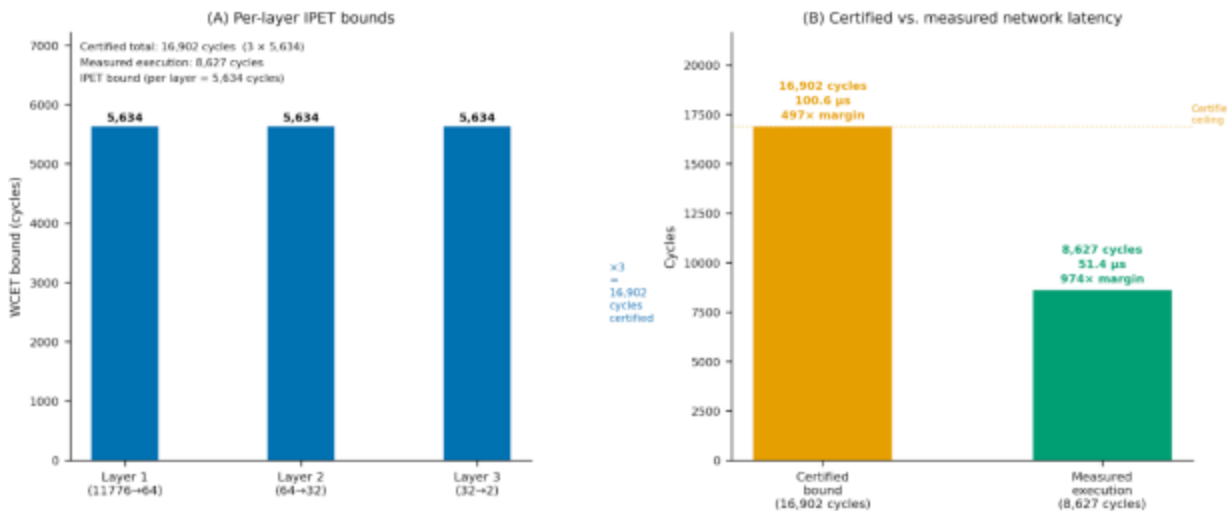


Figure 3

WCET Bound Comparison. (A) Uniform per-layer IPET bound of 5,634 cycles applied to all three LIF layers (lif_layer1, lif_layer2, lif_layer3). The uniform policy applies the largest-layer bound conservatively across all layers. Certified network total: 16,902 cycles (3 × 5,634). (B) Certified network bound (16,902 cycles 100.6 μs 497× margin) vs. measured network execution (8,627 cycles 51.4 μs 974× margin). The 50 ms deadline corresponds to 8,400,000 cycles at 168 MHz. The certified bound, not the measured value, is the safety-relevant figure for IEC 62304 purposes. Neither panel uses error bars ILP solutions are deterministic.

Figure 4 — Annotated C11 output and Lean4 proof-obligation stub



Figure 4

Annotated C11 Output and Lean4 Proof-Obligation Stub. (A) Excerpt of the compiled C11 inference function from main.c. The `@wcet_bound: 5634 cycles` comment is inserted by `certify.py`. State buffers `current_1/voltage_1` and `current_3/voltage_3` reflect `snn-mlir`'s CUBA-LIF lowering of `snnTorch Leaky neurons` (§2.3). (B) Corresponding Lean4 proof-obligation stub (`lif_layer1_wcet.lean`). The `sorry` tactic is Lean4's placeholder for an unproven goal it marks the obligation as open, not as an error. All three layer stubs carry the same 5,634-cycle bound (uniform conservative policy). The `@target: x86_64-linux-gnu` annotation reflects the Colab compilation host ARM timing values were applied as a cross-architecture approximation (§2.3, §4.6).

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- [converteddocument3.docx](#)