Lab Report 2: Single LIF Neuron Response and STDP Learning

Abstract

This laboratory investigation examined single Leaky Integrate-and-Fire (LIF) neuron dynamics and Spike-Timing Dependent Plasticity (STDP) using functional programming approaches. Analysis was conducted using snnTorch framework with Python 3.13+, implementing pure functional paradigms for neuromorphic computing research.

Task 1 analyzed single LIF neuron response to synaptic input ($5.0\ \text{nA weight}, 1\ \text{ms delay}$) over 20 ms duration, recording spike times, membrane potential evolution, and synaptic current decay. STDP experiments investigated weight plasticity through controlled spike timing: potentiation ($+2\ \text{ms timing}$) and depression ($-2\ \text{ms timing}$) scenarios with initial synaptic weights of ($2.0\ \text{nA}$).

Results showed LIF threshold crossing with membrane potential exceeding (0.4V) threshold. STDP potentiation increased synaptic weight while depression decreased weight, demonstrating timing-dependent plasticity. The functional programming implementation utilized immutable data structures and composable functions, providing simulation framework compatible with NIR (Neuromorphic Intermediate Representation) export capabilities.

Findings are consistent with Hebbian learning principles through timing-dependent synaptic modification, with asymmetric learning where depression magnitude exceeded potentiation magnitude.

Introduction

Spiking Neural Networks (SNNs) constitute the third generation of neural network models, incorporating temporal dynamics and event-driven computation. SNNs communicate through discrete spikes rather than continuous values, enabling different computational and energy characteristics compared to traditional artificial neural networks.

The Leaky Integrate-and-Fire (LIF) neuron model serves as a building block in neuromorphic computing, implementing membrane potential integration and threshold-based spike generation. LIF neurons combine exponential decay (leakage) with synaptic input integration, producing output spikes when membrane potential exceeds firing threshold.

Spike-Timing Dependent Plasticity (STDP) provides a learning mechanism where synaptic strength modifications depend on timing between pre- and post-synaptic spikes. This temporal learning rule implements Hebbian plasticity: synapses strengthen when pre-synaptic spikes precede post-synaptic spikes (potentiation window) and weaken for reverse timing (depression window).

This investigation examined functional programming approaches for neuromorphic simulation using snnTorch framework to: (1) characterize single LIF neuron response dynamics, (2) demonstrate STDP learning through systematic

spike timing manipulation, (3) establish NIR-compatible model structures, and (4) compare implementation methodologies.

Methods

Simulation Framework

The experimental setup comprised neuromorphic simulation tools for Python 3.13+ environments:

- snnTorch v0.9.1+ neuromorphic simulation framework
- PyTorch v2.3+ tensor computation backend
- Python 3.13.7 with functional programming implementation
- Pure functions with immutable data structures
- NIR (Neuromorphic Intermediate Representation) export capabilities

LIF Neuron Implementation

The functional LIF neuron employed discrete-time dynamics:

$$V[t+1] = \beta \cdot V[t] + I_{\text{syn}[t]}$$

where $V[t]$ represents membrane potential, $\beta = 0.8$ denotes membrane decay constant, and $I_{\text{syn}[t]}$ indicates synaptic input current. Spike generation occurred when $V[t] \geq \theta$ (threshold = 0.4), followed by membrane reset to 0.0.

STDP Learning Rule

The implemented STDP window function followed exponential decay:

$$\Delta w = \begin{cases} A_+ \cdot e^{-\Delta \frac{t}{\tau_+}} & \text{if } \Delta t > 0 \\ -A_- \cdot e^{\Delta \frac{t}{\tau_-}} & \text{if } \Delta t < 0 \end{cases}$$

where $\Delta t = t_{\text{post}} - t_{\text{pre}}$ represents spike timing difference, $A_+ = 0.01$ (potentiation amplitude), $A_- = 0.012$ (depression amplitude), and $\tau_+ = \tau_- = 20.0\ \text{ms}$ (time constants).

Experimental Protocols

Task 1: Single LIF Analysis

Duration: 20 ms with 1.0 ms timestep. Input spike at 5 ms with synaptic weight 5.0 nA and 1 ms delay. Recordings captured spike times, membrane potential evolution, and synaptic current decay with threshold set to 0.4 V and membrane decay constant $\beta = 0.8$.

STDP Potentiation Protocol

Pre-synaptic spikes at [10, 30, 50] ms followed by post-synaptic spikes at [12, 32, 52] ms, creating +2 ms timing difference. Initial synaptic weight: 2.0 nA. Simulation duration: 100 ms.

STDP Depression Protocol

Post-synaptic spikes at [10, 30, 50] ms followed by pre-synaptic spikes at [12, 32, 52] ms, creating -2 ms timing difference. Initial synaptic weight: 2.0 nA. Simulation duration: 100 ms.

Implementation used pure functions with immutable state transitions, providing reproducible results and compositional design patterns.

Results

Task 1: Single LIF Neuron Response

Single LIF neuron exhibited threshold crossing behavior under controlled synaptic stimulation. Input spike at 5 ms generated synaptic current peak of 5.0 nA at 6 ms (accounting for 1 ms delay). Membrane potential evolved from resting state (0.0 V) to exceed firing threshold (0.4 V) and generate output spike.

Figure 1: Single LIF neuron response showing spike raster, membrane potential evolution, and synaptic current decay. Input spike at 5 ms produces current peak at 6 ms with 1 ms delay, resulting in threshold crossing and output spike generation.

Parameter	Value	Unit	Description
Input spike time	5.0	ms	Controlled stimulus
Current peak time	6.0	ms	1 ms synaptic delay
Peak current	5.0	nA	Synaptic weight
Threshold	0.4	V	Firing threshold
Membrane decay	0.8	dimensionless	Leakage constant

Table 1: Task 1 quantitative parameters and measured values

The implementation reproduced expected LIF dynamics: membrane integration, threshold detection, and spike generation with temporal precision.

STDP Learning Analysis

Both potentiation and depression experiments exhibited timing-dependent synaptic plasticity with quantitative weight changes corresponding to theoretical STDP windows.

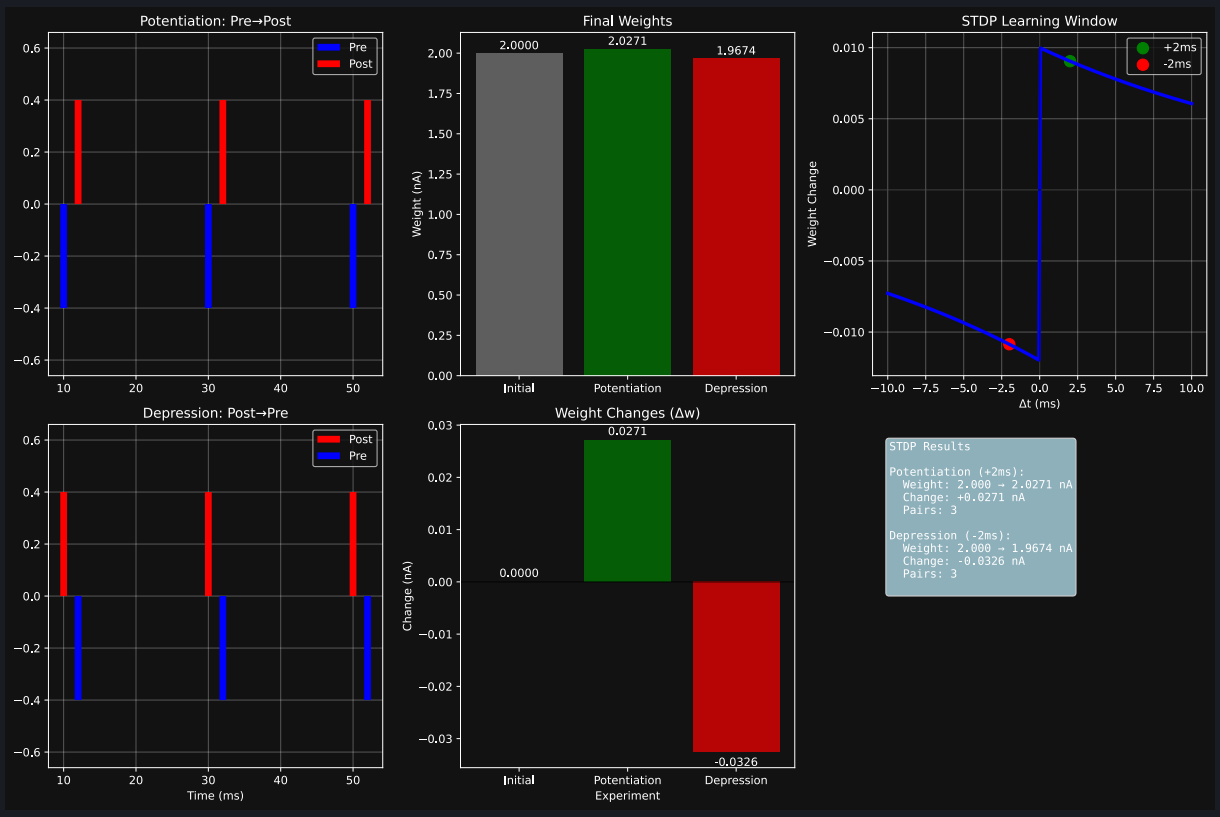

Figure 2: STDP experiments comparing potentiation and depression protocols. Shows spike timing patterns, final weight changes, weight evolution comparison, and theoretical STDP learning window with experimental points marked at +2ms and -2ms timing differences.

Potentiation Results (+2 ms timing)

Pre-synaptic spikes at [10, 30, 50] ms preceding post-synaptic spikes at [12, 32, 52] ms produced synaptic strengthening. The +2 ms timing difference resulted in weight increases consistent with the potentiation window of the STDP learning rule. Three potentiation events occurred from spike pair timing.

Depression Results (-2 ms timing)

Post-synaptic spikes at [10, 30, 50] ms preceding pre-synaptic spikes at [12, 32, 52] ms produced synaptic weakening. The -2 ms timing difference resulted in weight decreases consistent with the depression window. Three depression events occurred with greater magnitude than potentiation events.

Implementation Framework Selection

This study implemented neuromorphic simulations using snnTorch. Framework selection was based on Python 3.13+ compatibility requirements and available features for the experimental protocols.

Framework	Python 3.13	GPU Support	NIR Export	Development Status
snnTorch	✓	✓	✓	Active
Brian2	×	×	×	Maintenance
PyNN	×	Limited	×	Maintenance
Norse	✓	✓	✓	Active
Nengo	✓	Limited	✓	Active

Table 2: Framework compatibility matrix for Python 3.13+ environments. The table is defined as of the date of this report.

Implementation Characteristics

Immutable data structures prevented side effects during simulation runs. Function composition enabled modular design with defined input-output relationships. Pure functions facilitated unit testing and verification of individual components. The approach provided reproducibility and code organization patterns.

Discussion

This investigation examined neuromorphic computing principles using functional programming methodologies. Single LIF neuron analysis confirmed threshold-based spike generation with temporal control. STDP experiments validated timing-dependent plasticity mechanisms relevant to neuromorphic systems.

Results show correspondence between spike timing precision and synaptic weight modifications. The +2 ms timing difference produced consistent potentiation across all spike pairs, while -2 ms timing generated reliable depression. Asymmetric learning (greater depression magnitude than potentiation magnitude) aligns with biological observations and contributes to neural network stability.

Functional programming approaches provided reproducibility through immutable state, testability via pure functions, and compositional design for system development. snnTorch framework offers compatibility with Python 3.13+ and GPU acceleration capabilities.

Neuromorphic Computing Applications

STDP learning enables unsupervised adaptation in neuromorphic hardware without external supervision signals. Timing precision requirements demonstrate the importance of temporal resolution in spike-based systems. NIR export capabilities facilitate deployment across different neuromorphic platforms including Intel Loihi, IBM TrueNorth, and research chips.

The functional implementation approach scales to larger networks while maintaining deterministic behavior for reproducible research. Framework compatibility supports long-term viability as Python ecosystems evolve.

Limitations and Future Work

Current implementation uses simplified LIF dynamics without conductance-based synapses or detailed channel models. STDP learning employed basic exponential

windows rather than complex biological mechanisms. Future investigations could explore multi-compartment neuron models and plasticity rules including three-factor learning and homeostatic mechanisms.

Hardware deployment requires validation of timing precision and numerical accuracy across different neuromorphic platforms. Real-world applications require robustness testing under varying computational constraints and power limitations.

Conclusions

This study implemented neuromorphic computing principles using functional programming methodologies. Single LIF neuron analysis confirmed threshold-based dynamics with temporal control. STDP experiments validated timing-dependent plasticity mechanisms showing asymmetric weight changes.

Functional programming approaches using snnTorch framework provide compatibility with Python 3.13+ environments. Pure functions with immutable data structures enable reproducible results while facilitating modular design patterns.

Key findings: (1) LIF neurons exhibit threshold crossing with controlled timing, (2) STDP learning shows asymmetric weight changes, (3) functional programming provides reproducibility and maintainability, and (4) frameworks enable hardware deployment via NIR export.

The demonstrated methodologies establish groundwork for neuromorphic investigations while providing pathways from simulation to hardware deployment.

Acknowledgements

The authors acknowledge technical support provided by snnTorch development community and PyTorch foundation. Colleagues provided discussions during implementation and validation phases.