White Paper

Neuro-Inspired Error Propagation (NIEP): Refractory Period-Based Learning for Stable, Incremental AI Systems

David N. Halenta

Picyboo Cybernetics Inc., Research Lab (Canada)

Subject: Neural and Cognitive Architectures, Continual Learning, Gradient Dynamics, Energy-Efficient AI, Federated Learning, Computational Neuroscience

Contact: desk@davidhalenta.de / contact@picyboo.com

CSR: https://CognitiveScienceResearch.com/CSR4812

ORCID: https://orcid.org/0009-0003-3994-7912

Version 1.3, October 27, 2025.

DOI: https://doi.org/10.5281/zenodo.17455451

GitHub: https://github.com/Picyboo-Cybernetics/picyboo-public-niep

CSR-WP04-111025

Abstract

Standard backpropagation updates weights simultaneously and unconditionally, leading to unstable updates, catastrophic interference, and excessive energy consumption in continual or distributed learning scenarios. Biological neural networks, in contrast, employ refractory periods and locally bounded learning windows that temporally dampen and distribute error signals.

This paper presents Neuro-Inspired Error Propagation (NIEP), a learning scheme that temporally gates, locally constrains, and safely commits gradients. Core elements include refractory gating per parameter or neuron, eligibility traces for delayed updates, error budgeting with adaptive dampening, a safe-commit protocol using shadow weights, and asynchronous update scheduling. The objective is to achieve higher stability, reduced interference, improved energy and memory profiles, with maintained or enhanced generalization performance.

Keywords: Continual Learning, Catastrophic Interference, Catastrophic Forgetting, Refractory Period, Eligibility Traces, Error Gating, Safe Commit, Asynchronous Training, Stability–Plasticity Balance, Federated Learning, Energy Efficiency

1. Introduction

Deep learning systems have achieved remarkable success across diverse domains, yet they face fundamental challenges when learning must occur continuously, incrementally, or in resource-constrained environments. Standard backpropagation operates by computing gradients across the entire network and applying weight updates simultaneously and unconditionally. While this approach works well in batch training scenarios with stationary data distributions, it encounters severe difficulties in more realistic settings where data arrives sequentially, tasks change over time, or computational resources are limited.

The simultaneous nature of standard gradient updates creates several interrelated problems. When all parameters are updated at once in response to new training data, previously learned representations can be catastrophically overwritten. This phenomenon, known as catastrophic interference or catastrophic forgetting, represents a fundamental barrier to continual learning. Additionally, simultaneous updates can create gradient resonances where error signals from different parts of the network constructively interfere in unintended ways, leading to training instability and poor convergence properties.

From an energy perspective, standard backpropagation is inherently inefficient for incremental learning scenarios. Every training iteration triggers memory accesses and arithmetic operations across all parameters, regardless of whether individual weights actually require updating. This results in redundant computation and energy expenditure that scales poorly to edge devices, mobile platforms, or large-scale distributed systems.

Biological neural systems offer an alternative computational paradigm that addresses these challenges through fundamentally different mechanisms. Neurons in biological networks do not update their connection strengths simultaneously. Instead, they employ refractory periods during which their responsiveness to new inputs is temporarily reduced following recent activity. Learning signals are not applied instantaneously but rather accumulate through biochemical processes that create eligibility traces, allowing synaptic modifications to occur only when appropriate temporal and contextual conditions are met.

These biological principles suggest a path toward more stable and efficient artificial learning systems. NIEP translates three core neurobiological mechanisms into the framework of deep learning: refractory periods that temporarily reduce parameter update sensitivity following recent changes, eligibility traces that accumulate learning signals for delayed application, and safe-commit protocols that verify the benefit of proposed weight changes before permanently applying them. By incorporating these mechanisms, NIEP aims to preserve the plasticity necessary for learning while prioritizing the stability required for reliable incremental knowledge acquisition.

2. Problem Statement

The challenges addressed by NIEP arise from fundamental properties of standard gradient-based learning in non-stationary or resource-constrained scenarios.

2.1 Gradient Resonance and Simultaneous Updates

When all parameters in a neural network are updated simultaneously in response to each training batch, the resulting gradient flow can create resonance effects where error signals from different network regions constructively interfere. This phenomenon is particularly problematic in deep architectures where gradients must propagate through many layers. Small errors in early processing stages can be amplified through subsequent layers, creating instabilities that manifest as training divergence or oscillation around optimal solutions. The lack of temporal structure in standard backpropagation means that these resonances cannot be naturally dampened through phase-shifted or asynchronous updates.

2.2 Catastrophic Interference in Sequential Learning

Perhaps the most severe limitation of standard backpropagation for continual learning is catastrophic interference. When a network trained on one task is subsequently trained on a different task, the weight updates required for the new task often directly overwrite the representations learned for the previous task. This occurs because there is no mechanism in standard backpropagation to protect previously learned information from modification. The network has no concept of which weights encode important knowledge from past tasks versus which weights can be safely repurposed. As a result, performance on earlier tasks can degrade dramatically—often dropping to near-random levels—after training on just a few examples from a new task.

2.3 Energy Inefficiency Through Redundant Updates

Standard backpropagation computes gradients and applies updates to all parameters on every training iteration, regardless of whether individual parameters actually require modification. In many learning scenarios, particularly in later stages of training or when learning subtle refinements, only a small subset of parameters needs adjustment. Nevertheless, the entire network undergoes memory reads, arithmetic operations, and memory writes on each iteration. This creates significant energy overhead, particularly problematic for deployment on battery-powered devices, edge computing platforms, or large-scale data centers where energy costs are substantial.

2.4 Challenges in Distributed and Federated Settings

Distributed and federated learning scenarios introduce additional complications. In federated learning, multiple devices train local models on private data, then aggregate their updates to improve a global model. Standard synchronous aggregation requires all devices to complete their local training and communicate their updates simultaneously, creating bottlenecks when devices have varying computational capabilities or network connectivity. Moreover, without local stability mechanisms, the aggregated updates from different devices can conflict, leading to model divergence or oscillation rather than convergence toward a globally optimal solution.

2.5 Design Goal

The overarching objective of NIEP is to create a learning mechanism that maintains the plasticity necessary for acquiring new knowledge while prioritizing the stability required for retaining existing knowledge and ensuring reliable performance. This is achieved by introducing temporal structure into the learning process: dosing error signals over time rather than applying them all at once, localizing updates to parameters that are not in refractory states, and confirming that proposed changes genuinely improve performance before committing them permanently to the model.

3. Core Principles

NIEP builds upon three fundamental principles inspired by neurobiological learning mechanisms, translating them into mathematically precise operations compatible with deep learning frameworks.

3.1 Refractory Periods

In biological neurons, the refractory period is a brief interval following an action potential during which the neuron's threshold for firing is elevated, making it temporarily less responsive to new inputs. This mechanism prevents excessive firing rates and allows the neuron to recover its ionic balance before responding to subsequent stimuli.

NIEP adapts this concept to weight updates in artificial neural networks. Each parameter maintains a refractory state that increases following both high activation levels and large gradient magnitudes. When a parameter's refractory state is elevated, its susceptibility to updates is reduced through a gating mechanism. This creates temporal structure in the learning process where parameters that have recently undergone significant changes or contributed strongly to network computations are temporarily protected from further modification. The refractory state decays over time through a leaky integration process, allowing parameters to gradually regain full update sensitivity.

This mechanism serves multiple purposes. It prevents rapid, oscillatory changes to individual weights that can destabilize training. It naturally implements a form of temporal credit assignment where parameters that have recently been active or updated are given time to consolidate their new values before being modified again. It also provides a substrate for implementing parameter-specific learning rates without requiring explicit meta-learning or manual tuning.

3.2 Eligibility Traces

Biological synaptic plasticity does not occur instantaneously upon the arrival of a learning signal. Instead, synapses maintain biochemical traces of recent pre- and post-synaptic activity, and modification occurs only when an appropriate neuromodulatory signal arrives while these traces are active. This creates a temporal credit assignment mechanism that can bridge delays between actions and their consequences.

NIEP implements eligibility traces as running accumulators of gated gradient information for each parameter. Rather than immediately applying computed gradients to weights, gradients

are first modulated by the refractory gating function, then accumulated into the eligibility trace through a leaky integration process. The eligibility trace thus represents a temporally smoothed, refractory-gated summary of recent gradient information. Updates are applied based on the eligibility trace rather than raw gradients, introducing temporal filtering that reduces noise and prevents premature commitment to update directions based on single training examples.

The eligibility trace mechanism also enables more sophisticated credit assignment in reinforcement learning settings, where rewards may arrive long after the actions that caused them. By maintaining a decaying trace of recent gradient information, NIEP can appropriately assign credit to parameters that contributed to successful outcomes, even when feedback is delayed.

3.3 Safe Commit Protocol

Perhaps the most distinctive feature of NIEP is its two-stage update mechanism inspired by database transaction protocols. Rather than directly modifying working weights based on accumulated gradients, NIEP first applies proposed updates to shadow weights—a parallel set of parameters that represents the tentative post-update model. The system then evaluates the shadow weights against stability and performance criteria, typically by measuring validation loss, calibration quality, or other metrics on held-out data.

Only if the shadow weights pass this evaluation—demonstrating that the proposed update genuinely improves performance without catastrophically degrading previously learned capabilities—are they committed to the working weights. The commit operation itself is soft, applying a fractional blend between current and shadow weights rather than an all-or-nothing replacement. This creates a conservative update scheme that prioritizes stability: proposed changes are only accepted if they are demonstrably beneficial, and even then, they are applied gradually.

This safe-commit mechanism provides strong guarantees against catastrophic interference. Even if the gradient computation or eligibility accumulation suggests a particular update direction, the update will not be permanently applied if it would harm overall model performance. This makes NIEP particularly suitable for safety-critical applications where model degradation must be avoided, and for continual learning scenarios where preserving performance on previous tasks is paramount.

4. Mathematical Framework

We now formalize the NIEP algorithm through precise mathematical definitions of its core components.

4.1 Notation

We consider a neural network with parameters denoted by weights w_i for each parameter index i. Standard backpropagation computes gradients $g_i(t)$ at time step t, and updates are controlled by a learning rate η.

NIEP extends this framework with additional per-parameter state variables:

Refractory state $r_i(t) \in [0,1]$: Higher values indicate the parameter is in a refractory period with reduced update sensitivity.

Eligibility trace $e_i(t) \in \mathbb{R}$: Accumulated, gated gradient information.

Gating function $G_i(t) \in [0,1]$: Determines the fraction of current gradient that contributes to the eligibility trace.

Error budget $B_i(t) \geq 0$: Maximum permissible update magnitude for parameter i at time t.

Shadow weight $\tilde{w}_i \in \mathbb{R}$: Tentative parameter value used for safe-commit evaluation.

Additional notation includes parameter activations $a_i(t)$, the decay factors α, λ, and ρ for various leaky integrators, and hyperparameters β, γ, δ, ξ, κ, T, χ, and τ that control the dynamics of each component.

4.2 Refractory State Dynamics

The refractory state for each parameter evolves according to a leaky integrator that accumulates evidence of recent activity and gradient magnitudes:

$$r_i(t+1) = \alpha \cdot r_i(t) + \beta \cdot |a_i(t)| + \gamma \cdot |g_i(t)|$$

Here, $\alpha < 1$ is a decay factor that causes refractory states to gradually return to zero in the absence of activity or gradients. The terms $\beta \cdot |a_i(t)|$ and $\gamma \cdot |g_i(t)|$ couple the refractory state to recent activation magnitudes and gradient magnitudes, respectively. Parameters that have been strongly activated or have received large gradient signals enter elevated refractory states, temporarily reducing their update sensitivity.

The hyperparameters β and γ control the relative influence of activation-based versus gradient-based contributions to refractoriness. In feedforward networks, gradient magnitude may be the primary driver, while in recurrent networks or during inference-time adaptation, activation magnitude becomes more relevant.

4.3 Gating Function

The gating function determines what fraction of the current gradient contributes to the eligibility trace based on the current refractory state. We define:

$$G_i(t) = \sigma((\kappa - r_i(t))/T)$$

where σ is the sigmoid function, κ is a threshold parameter, and T controls the temperature or sharpness of the gating transition. When $r_i(t)$ is well below κ, the gating function approaches 1, allowing full gradient contribution. When $r_i(t)$ exceeds κ, the gating function approaches 0, effectively blocking gradient accumulation.

An alternative formulation uses a hard threshold:

$$G_i(t) = 1[r_i(t) \leq \kappa]$$

where $1[\cdot]$ is the indicator function. This creates a binary gate that is fully open when refractoriness is below threshold and fully closed otherwise. The soft sigmoid version is typically preferred for gradient-based optimization as it maintains differentiability.

4.4 Eligibility Trace Dynamics

The eligibility trace accumulates gated gradients through leaky integration:

$$e_i(t+1) = \lambda \cdot e_i(t) + G_i(t) \cdot g_i(t)$$

The decay factor $\lambda < 1$ causes old gradient information to fade over time, ensuring that the eligibility trace represents recent rather than ancient learning signals. The term $G_i(t) \cdot g_i(t)$ adds the current gradient modulated by the refractory gate, allowing gradient accumulation only when the parameter is not in a refractory state.

This formulation creates a temporally smoothed gradient estimate that filters out high-frequency noise while respecting refractory constraints. In the limit where $\lambda \to 1$ and $G_i(t) \equiv 1$, the eligibility trace becomes a running sum of all past gradients. In practice, moderate values of λ (e.g., 0.9) and time-varying gating create a selective memory that emphasizes recent, refractory-permitted gradient information.

4.5 Error Budget

The error budget constrains the maximum permissible update magnitude for each parameter, adapting based on gradient statistics and recent eligibility trace magnitudes. The budget evolves as:

$$B_i(t+1) = \max(0, \rho \cdot B_i(t) + \delta \cdot \mu/\sqrt{(\mathrm{Var}[g_i] + \varepsilon)} - \xi \cdot |e_i(t+1)|)$$

This equation incorporates several components. The term $\rho \cdot B_i(t)$ with $\rho < 1$ implements leaky decay, gradually increasing budget availability over time if no updates are applied. The term $\delta \cdot \mu/\sqrt{(\mathrm{Var}[g_i] + \varepsilon)}$ adds budget based on gradient statistics: μ is a target mean update size, and the denominator normalizes by gradient variance to provide larger budgets when gradients are stable and smaller budgets when gradients are noisy. The term $\xi \cdot |e_i(t+1)|$ subtracts from the budget proportional to the magnitude of the accumulated eligibility trace, effectively consuming budget as updates are prepared.

The $\max(0, \cdot)$ operation ensures budgets never become negative. The hyperparameter ε is a small constant preventing division by zero. The parameter μ represents a target mean update magnitude, typically set to match the expected gradient magnitude $E[|g|]$ or chosen as a constant scaling factor to produce budgets in a desired numerical range. This adaptive budgeting mechanism provides variance-aware update constraints that automatically adjust to the statistical properties of gradients for each parameter.

4.6 Pre-Commit Update

Proposed updates are first applied to shadow weights rather than working weights:

$$\tilde{w}_i \leftarrow \tilde{w}_i - \eta \cdot \mathrm{clip}(e_i(t+1), B_i(t+1))$$

The clip operation constrains the eligibility trace magnitude to respect the error budget:

$$\mathrm{clip}(e, B) = e \cdot \min(1, B/|e|)$$

This ensures that even if the eligibility trace suggests a large update, the actual pre-commit modification respects the local error budget. The learning rate η scales the overall update magnitude, and this can be further modulated by standard optimizer transformations (e.g., Adam's adaptive learning rates).

4.7 Safe-Commit Protocol

The final step evaluates whether the shadow weights should be committed to the working weights. This decision is based on validation metrics, typically measured on a held-out dataset. The commit operation is:

$w_i \leftarrow w_i + \chi \cdot (\tilde{w}_i - w_i)$ if $\Delta L_{val} \leq \tau \wedge$ calibration improves

Here, $\chi \in (0,1]$ is a commit rate that controls how aggressively shadow weights are blended into working weights. The condition $\Delta L_{val} \leq \tau$ requires that the validation loss change is within an acceptable tolerance τ. The additional calibration condition can be formalized through metrics such as Expected Calibration Error (ECE), requiring that the shadow weights produce better-calibrated predictions than the current working weights.

If the commit condition is not satisfied, the shadow weights remain separate, and the system defers commitment until future iterations when conditions improve. This creates a conservative update policy where stability is prioritized over rapid adaptation.

5. The NIEP Algorithm

We now present the complete NIEP algorithm as an integrated procedure suitable for implementation in modern deep learning frameworks.

5.1 Algorithm Structure

The NIEP training loop operates on mini-batches of data, computing gradients through standard backpropagation but applying updates through the refractory-gated, eligibility-traced, safe-commit mechanism. The algorithm accepts as inputs a mini-batch X, a loss function L, a base optimizer (e.g., Adam or SGD), and the hyperparameters α, β, γ, λ, ρ, δ, ξ, κ, T, η, χ, and τ.

Each training iteration proceeds through the following steps:

Step 1: Forward Pass. Compute activations a and loss L from the current mini-batch using the working weights w.

Step 2: Backward Pass. Compute gradients g through standard backpropagation.

Step 3: Refractory Update. Update each parameter's refractory state according to $r \leftarrow f(r, a, g)$ where f implements the leaky integration defined in Section 4.2.

Step 4: Gating. Compute gating values $G \leftarrow \text{gate}(r)$ based on current refractory states and threshold parameters.

Step 5: Eligibility Update. Update eligibility traces according to $e \leftarrow \lambda \cdot e + G \odot g$, where $\odot$ denotes element-wise multiplication.

Step 6: Budget Update. Update error budgets based on $B \leftarrow \text{budget}(B, e, \text{Var}[g])$, implementing the variance-aware accumulation defined in Section 4.5.

Step 7: Pre-Commit. Apply clipped eligibility traces to shadow weights: $\tilde{w} \leftarrow \tilde{w} - \eta \cdot \text{clip}(e, B)$.

Step 8: Safe-Commit Evaluation. Evaluate shadow weights on validation data and compute metrics ΔL_val and calibration quality.

Step 9: Conditional Commit. If commit conditions are satisfied, blend shadow weights into working weights: $w \leftarrow w + \chi \cdot (\tilde{w} - w)$. Otherwise, retain current working weights and defer commitment.

Step 10: Periodic Relaxation. At regular intervals, reduce all refractory states by the global decay factor α to prevent permanent parameter freezing.

5.2 Integration with Standard Optimizers

NIEP is designed to be compatible with existing optimization algorithms such as Adam, RMSprop, or SGD with momentum. The integration occurs at Step 7, where the clipped eligibility trace clip(e, B) serves as the effective gradient input to the optimizer. The optimizer then applies its own transformation—adaptive learning rates, momentum terms, etc.—to produce the final update applied to shadow weights. This allows NIEP to leverage the benefits of advanced optimization techniques while adding refractory gating, eligibility traces, and safe-commit on top of those optimizations.

5.3 Computational Considerations

The additional state variables (r, e, B, $\tilde{w}$) increase memory requirements by a factor of approximately 4× compared to standard training that stores only w and optimizer states. However, these states can be stored in lower precision (e.g., FP16 or even INT8 for r and B) to reduce overhead. The additional computations—refractory updates, gating, budget calculations—are element-wise operations that add minimal computational cost compared to the forward and backward passes themselves.

The safe-commit evaluation requires periodic validation passes, which do add computational cost. However, this evaluation need not occur on every iteration; evaluating every k iterations (e.g., k=10 or k=100) provides a practical balance between safety and efficiency. The validation dataset can also be a small subset rather than the full validation set, further reducing overhead.

6. Architectural and System Implications

Beyond the core algorithmic components, NIEP has broader implications for neural network architecture design and system-level implementation.

6.1 Asynchronous Updates Across Layers

Standard backpropagation updates all layers simultaneously after gradient computation. NIEP's refractory gating naturally enables asynchronous updates where different layers or network blocks learn at different rates based on their individual refractory states. Layers that have recently undergone significant updates or have high activation variance may have elevated refractory states, causing them to update more slowly than layers with lower refractory states.

This asynchronous update pattern reduces synchronization costs in distributed implementations, as different computational nodes can update their assigned parameters independently without waiting for global synchronization barriers. It also more closely mimics biological neural networks, where different brain regions exhibit different learning rates and timescales.

6.2 Local Memory Pools

The per-parameter state variables (r, e, B, $\tilde{w}$) should ideally be stored close to the parameters themselves in memory hierarchy. In GPU implementations, this means organizing memory layouts such that a parameter's weight, refractory state, eligibility trace, budget, and shadow weight are stored contiguously or in the same memory bank. This locality reduces memory access latency and improves cache utilization, as operations that read a parameter's weight typically also need to access its associated states.

In neuromorphic or specialized hardware implementations, these local memory pools map naturally to on-chip SRAM or register files co-located with arithmetic units, enabling high-bandwidth, low-latency access to all parameter-associated data.

6.3 Event-Driven Computation

The gating mechanism creates opportunities for event-driven computation. When $G_i(t)$ is near zero due to high refractory state, parameter i does not require gradient accumulation or update operations. Hardware or software systems can exploit this by skipping computations for gated-off parameters, reducing both energy consumption and processing time.

In sparse networks or after pruning, this event-driven approach becomes particularly valuable. If a parameter has been pruned (set to zero and frozen), its refractory state can be set to maximum, permanently gating off all updates and allowing complete omission of that parameter from computation graphs.

6.4 GPU Implementation

Modern GPU implementations of NIEP leverage parallel processing for element-wise operations on state tensors. Refractory updates, gating, eligibility accumulation, and budget updates are all highly parallelizable. Masking operations for gating can be implemented through element-wise multiplication of gradient tensors with gate value tensors. Leaky integration (for r, e, and B) maps naturally to AXPY-style operations ($\alpha \cdot x + y$) available in optimized BLAS libraries.

The safe-commit operation requires a specialized kernel that conditionally blends shadow weights into working weights based on validation metrics. This can be implemented as a fused kernel that reads validation results, computes per-parameter or per-layer blend coefficients, and performs the blending in a single pass to minimize memory traffic.

6.5 Neuromorphic Compatibility

NIEP's mechanisms align well with neuromorphic hardware architectures. Leaky integrators for refractory states and eligibility traces map directly to capacitor-based or digital accumulator circuits common in neuromorphic chips. The gating function can be implemented through comparator circuits or lookup tables. The event-driven nature of gated updates aligns with the spike-based or event-driven paradigms of neuromorphic processors, potentially enabling very low-power implementations of NIEP on specialized hardware.

7. Theoretical Properties

While formal convergence proofs and error bounds remain subjects for future theoretical work, we can outline the expected theoretical properties of NIEP based on its design principles.

7.1 Stability-Plasticity Trade-off

NIEP addresses the stability-plasticity dilemma through its refractory and safe-commit mechanisms. The refractory gating temporarily reduces plasticity for parameters that have recently changed, providing a window for those changes to consolidate without immediate further modification. The safe-commit protocol ensures that new learning does not catastrophically overwrite stable, useful representations by requiring validation before permanent updates.

This creates a dynamic balance where the system remains plastic enough to learn new information when refractory states are low and validation metrics support updates, but stable enough to resist overwriting useful knowledge when refractory states are high or validation metrics indicate degradation.

7.2 Interference Reduction

Catastrophic interference in standard neural networks occurs when gradient updates for new tasks directly overwrite weights that encode information about previous tasks. NIEP reduces this interference through multiple mechanisms. Refractory gating prevents large, simultaneous updates across the entire network, distributing changes over time. Error budgets constrain update magnitudes based on gradient statistics, preventing extreme weight swings. Most importantly, safe-commit refuses to apply updates that degrade validation performance, providing a direct guard against catastrophic interference.

7.3 Convergence Sketch

Consider a scenario with a stationary data distribution and fixed hyperparameters. The refractory states, governed by the leaky integration $r_i(t+1) = \alpha \cdot r_i(t) + \beta \cdot |a_i(t)| + \gamma \cdot |g_i(t)|$ with $\alpha < 1$, will converge to a bounded set determined by the typical magnitudes of activations and gradients. Similarly, eligibility traces with $\lambda < 1$ will converge to bounded values representing temporally smoothed gradients.

In this setting, the safe-commit protocol enforces that working weights are only updated when validation loss improves within tolerance τ. This creates a monotonic non-worsening

guarantee on validation performance within validation tolerance τ, similar to early stopping or validation-based model selection. While this does not guarantee convergence to a global optimum—standard backpropagation itself lacks such guarantees—it does ensure that NIEP will not diverge or catastrophically degrade in performance.

A complete convergence analysis would require specifying assumptions about the loss landscape, gradient noise properties, and hyperparameter choices, and deriving bounds on the rate of convergence. This remains an important direction for future theoretical work.

7.4 Energy Profile

NIEP's energy efficiency comes from reducing the effective number of parameter updates through refractory gating. When G_i(t) is small or zero, gradient accumulation into eligibility traces has minimal impact, and if updates are skipped entirely for gated-off parameters, the system avoids memory reads and writes for those parameters.

In the limit where a large fraction of parameters are in refractory states at any given time, the number of active updates per iteration can be substantially reduced compared to standard backpropagation. This translates directly to reduced energy consumption, as memory access (particularly off-chip DRAM access) is typically the dominant energy cost in neural network training. Quantifying the exact energy savings requires empirical measurement on specific hardware and workloads, but preliminary analysis suggests potential reductions of 30-70% in memory-access-related energy for continual learning tasks.

8. Applications

NIEP is designed for learning scenarios where stability, incremental adaptation, or resource efficiency are critical requirements.

8.1 Continual and Online Learning

In continual learning, a model must learn a sequence of tasks without forgetting previously learned tasks. This is the primary application domain for NIEP. The safe-commit protocol directly addresses catastrophic forgetting by refusing updates that degrade performance on validation data drawn from previous tasks. The refractory mechanism further protects consolidated knowledge by reducing update sensitivity for parameters that have recently learned stable representations. NIEP enables networks to accumulate knowledge over extended task sequences, maintaining performance on early tasks while acquiring new capabilities.

8.2 Edge and On-Device Learning

Mobile devices, IoT sensors, and edge computing platforms have strict energy and memory constraints. NIEP's reduced update frequency and event-driven computation lower energy consumption compared to standard continuous learning approaches. The ability to perform incremental fine-tuning with stability guarantees makes NIEP suitable for personalization scenarios where a pre-trained model adapts to individual user data without catastrophic degradation. The safe-commit protocol provides additional safety for autonomous systems that must guarantee they do not degrade below acceptable performance thresholds during adaptation.

8.3 Safety-Critical Systems

In domains such as medical diagnosis, autonomous vehicles, or industrial control, model updates carry risk. A naive update that improves performance on new data but degrades performance on safety-critical edge cases could have severe consequences. NIEP's safe-commit protocol with validation-based acceptance criteria provides a mechanism to ensure that updates genuinely improve overall model capability without introducing regressions. This makes NIEP particularly suitable for scenarios where model reliability and safety are paramount, and where controlled, verified adaptation is preferred over rapid but risky learning.

8.4 Federated Learning

Federated learning involves multiple devices training local models on private data, then aggregating updates to improve a shared global model. Standard federated averaging can suffer from high variance in local updates and conflicts between updates from different devices. NIEP provides local stability mechanisms—refractory gating, eligibility smoothing, and safe-commit—that reduce the variance of local updates before aggregation. This leads to more stable global models and reduces the risk of divergence in federated settings.

Moreover, NIEP's asynchronous nature aligns well with realistic federated scenarios where devices have varying computational capabilities and intermittent connectivity. Devices can perform local NIEP training independently, committing updates only when their local validation metrics support it, then asynchronously contributing their committed changes to the global aggregation when connectivity is available.

8.5 Reinforcement Learning

Reinforcement learning in non-stationary environments presents challenges similar to continual learning: the agent must adapt to changing reward structures or environment dynamics without forgetting useful behaviors. NIEP's update throttling and safe-commit mechanisms enable more stable policy improvement in RL scenarios. By gating rapid policy changes through refractory periods and validating policy updates against performance metrics (e.g., average return over multiple episodes), NIEP can prevent the destructive policy oscillations that sometimes occur in standard RL training.

9. Evaluation Framework and Metrics

Assessing NIEP's effectiveness requires metrics that capture both traditional learning performance and the specific goals of stability, reduced interference, and efficiency.

9.1 Catastrophic Forgetting Metrics

Standard continual learning metrics quantify forgetting and transfer:

Average Forgetting (F): After training sequentially on tasks T_1, T_2, ..., T_n, forgetting for task T_i is defined as the difference between the maximum accuracy achieved on T_i during training and the final accuracy after training on all subsequent tasks. Average forgetting is the mean over all tasks except the last.

Backward Transfer (BWT): Measures how learning new tasks influences performance on previous tasks, computed as the average difference in accuracy on previous tasks after versus before learning new tasks.

Forward Transfer (FWT): Measures how learning previous tasks influences performance on new tasks, computed by comparing initial accuracy on a new task (before training on it) against a baseline trained from scratch.

9.2 Stability-Plasticity Index

To quantify the balance between stability and plasticity, we define:

$$SPI = \Delta Gen / (\Delta Forget + \varepsilon)$$

where ΔGen is the improvement in generalization (e.g., validation accuracy gain) from updates, ΔForget is the increase in forgetting, and ε is a small constant preventing division by zero. Higher SPI values indicate better balance: significant generalization gains with minimal forgetting.

9.3 Calibration Quality

The safe-commit protocol in NIEP explicitly considers model calibration, making calibration metrics central to evaluation. Expected Calibration Error (ECE) measures the difference between predicted confidence and actual accuracy across binned predictions. A well-calibrated model's confidence scores accurately reflect the true probability of correctness.

We measure ECE before and after commit operations to verify that NIEP's safe-commit protocol genuinely improves calibration. Additionally, Negative Log-Likelihood (NLL) provides a proper scoring rule for probabilistic predictions, penalizing both inaccurate predictions and miscalibrated confidence. NIEP should demonstrate improved or maintained calibration compared to standard backpropagation, particularly in continual learning scenarios where calibration often degrades as new tasks are learned.

9.4 Energy and Resource Metrics

To validate NIEP's efficiency claims, we measure:

Training Energy (Joules per sample): Total energy consumed during training divided by the number of training samples processed. This can be measured directly through power monitoring hardware or estimated through operation counts and hardware specifications.

Memory Accesses per Update: The number of memory read and write operations required per training iteration. Reduced memory traffic directly translates to energy savings and improved training throughput on memory-bound systems.

Effective Update Frequency: The fraction of parameters that receive non-negligible updates per iteration, accounting for refractory gating. Lower effective update frequency indicates more selective, efficient learning.

Wall-Clock Time and Throughput: Training time per epoch and throughput (samples processed per second) under various hardware configurations, including edge devices and distributed systems.

9.5 Latency and Asynchronous Performance

In distributed and asynchronous settings, we measure:

Convergence Time: Wall-clock time to reach target performance levels under asynchronous update scheduling.

Communication Overhead: In federated settings, the volume and frequency of model updates exchanged between devices and central servers.

Straggler Resilience: Performance degradation when some devices or workers experience delays, measuring NIEP's ability to make progress despite asynchronous updates.

10. Ablation Studies and Component Analysis

Understanding the contribution of each NIEP component requires systematic ablation studies where individual mechanisms are disabled or modified.

10.1 Without Refractory Gating ($r \equiv 0$)

Setting all refractory states to zero effectively disables the gating mechanism, allowing all gradients to flow directly into eligibility traces. This ablation isolates the contribution of temporal gating versus eligibility smoothing and safe-commit alone. We expect this variant to show reduced stability compared to full NIEP, as parameters lack protection against rapid successive modifications.

10.2 Without Eligibility Traces ($e \equiv g$)

Bypassing eligibility traces by directly using raw gradients eliminates temporal smoothing and credit assignment mechanisms. This ablation tests whether the leaky integration of gradients provides benefits beyond refractory gating. We expect increased sensitivity to gradient noise and reduced performance in reinforcement learning scenarios where temporal credit assignment is crucial.

10.3 Without Error Budgets ($B \equiv \infty$)

Removing error budgets by setting them to infinity allows arbitrarily large updates regardless of gradient statistics or recent update history. This ablation isolates the contribution of variance-aware update constraints. We expect this variant to show increased training instability, particularly in later stages of training or when gradient variance is high.

10.4 Without Safe-Commit (Direct Updates)

Applying updates directly to working weights without the safe-commit evaluation removes the validation-based acceptance criterion. This ablation measures the specific contribution of safe-commit to preventing catastrophic interference. We expect significantly increased forgetting in continual learning scenarios, as updates are no longer required to preserve validation performance.

10.5 Varying Commit Frequency

Evaluating commit conditions every k iterations for different values of k (e.g., k=1, 10, 100, 1000) explores the trade-off between safety and computational overhead. Very frequent evaluation provides maximum safety but incurs substantial validation costs, while infrequent evaluation reduces overhead but may allow longer periods of degraded performance before detection.

10.6 Component Interactions

Beyond individual ablations, studying interactions between components reveals synergies. For example, the combination of refractory gating and eligibility traces may provide greater benefits than either mechanism alone, as refractory periods allow eligibility traces to accumulate meaningful temporal structure. Similarly, error budgets may be most effective when combined with safe-commit, as budgets prevent extreme pre-commit updates while safe-commit prevents acceptance of harmful but budget-respecting updates.

11. Hyperparameter Guidelines

NIEP introduces multiple hyperparameters governing the dynamics of refractory states, eligibility traces, budgets, and commit operations. While optimal values are task-dependent, we provide initial recommendations based on theoretical considerations and preliminary experiments.

11.1 Refractory Parameters

$\alpha = 0.9$: Decay factor for refractory states. This value provides a half-life of approximately 7 iterations, allowing refractory effects to influence learning over meaningful timescales while preventing permanent parameter freezing.

$\beta = 0.05$, $\gamma = 0.05$: Coupling strengths for activation and gradient contributions to refractoriness. These moderate values ensure that typical activation and gradient magnitudes produce refractory states in the 0.1-0.3 range, providing gentle gating rather than complete blocking.

$\kappa = 0.3$: Gating threshold. Parameters with refractory states below 0.3 receive near-full gradient contribution, while those above 0.3 experience increasing gating.

$T = 0.05$: Gating temperature. This controls the sharpness of the sigmoid transition in the gating function, with smaller values producing sharper, more threshold-like behavior.

11.2 Eligibility Parameters

$\lambda = 0.9$: Decay factor for eligibility traces. Matching the refractory decay factor creates aligned timescales for gating and trace accumulation. This value emphasizes recent gradients while maintaining memory of gradients from approximately the last 10 iterations.

11.3 Budget Parameters

$\rho = 0.9$: Decay factor for error budgets, allowing gradual budget recovery.

$\delta = 1.0$: Scaling factor for variance-based budget replenishment. This provides unit-scale budgets when gradients have unit variance.

$\xi = 0.1$: Budget consumption rate per unit of eligibility trace magnitude. This moderate value allows multiple moderate updates before budget exhaustion while preventing extreme update sequences.

11.4 Commit Parameters

η: Base learning rate should initially match the learning rate used for the baseline optimizer (e.g., 0.001 for Adam).

χ = 0.25: Commit blending rate. This conservative value applies only 25% of the shadow-working weight difference per commit, providing smooth, gradual convergence even when multiple consecutive commits occur.

τ: Validation tolerance depends on the task and acceptable performance degradation. For safety-critical applications, τ might be set to 0 (requiring strict validation improvement), while for less critical scenarios, small positive values (e.g., 0.01) allow commits that cause minor validation loss increases.

11.5 Adaptation and Tuning

These initial values provide a reasonable starting point, but optimal hyperparameters vary with network architecture, task characteristics, and performance requirements. Automated hyperparameter tuning through techniques such as Bayesian optimization or population-based training can discover task-specific configurations. Key tuning axes include:

- Refractory sensitivity (β, γ): Increased values create stronger gating and longer refractory periods.
- Temporal scale (α, λ): Lower values create shorter memory, higher values create longer temporal integration.
- Commit conservatism (χ, τ): Lower χ and stricter τ prioritize stability over adaptation speed.

12. Security, Robustness, and Safety Considerations

NIEP's mechanisms have implications for adversarial robustness, security in distributed settings, and safe deployment.

12.1 Protection Against Commit Exploitation

The safe-commit protocol could potentially be exploited if an adversary can manipulate validation data or metrics to cause acceptance of harmful updates. To mitigate this risk, NIEP implementations should enforce minimum observation windows, requiring multiple validation batches from diverse data sources before committing. Multi-criteria commit conditions—simultaneously checking validation loss, calibration, and task-specific metrics—provide redundancy against single-metric attacks.

In federated or distributed settings, cryptographic verification of validation metrics and secure aggregation protocols prevent individual nodes from falsely reporting validation improvements to force acceptance of malicious updates.

12.2 Adversarial Robustness

Refractory gating and error budgets provide a degree of natural robustness against gradient-based adversarial attacks during training. When an adversary attempts to inject malicious gradients through poisoned training examples, refractory states limit the immediate impact of these gradients. Error budgets constrain update magnitudes based on typical gradient statistics, causing extreme adversarial gradients to be clipped or rejected.

The safe-commit protocol offers additional protection: even if adversarial gradients pass through refractory gating and eligibility accumulation, the resulting shadow weights will exhibit degraded validation performance, causing the commit condition to fail and preventing permanent model corruption. This creates defense-in-depth where multiple mechanisms must be compromised simultaneously for a successful attack.

12.3 Distributed Consensus and Conflict Resolution

In distributed training scenarios, different nodes may propose conflicting updates based on their local data. NIEP's safe-commit protocol can be extended to distributed consensus mechanisms where multiple nodes vote on proposed updates based on their local validation metrics. A commit is accepted only when a sufficient fraction of nodes report validation improvement.

When conflicts occur—some nodes support a commit while others reject it—conflict resolution strategies include weighted voting based on validation confidence, hierarchical decision-making where certain nodes have priority, or gradient-based negotiation where conflicting updates are blended proportionally to their validation support.

12.4 Safety Monitoring and Guarantees

For safety-critical applications, NIEP can be augmented with formal safety monitors that verify proposed updates against explicit safety constraints beyond validation metrics. For example, in autonomous vehicle applications, safety monitors might check that updated policies maintain minimum distances from obstacles or respect speed limits across a diverse set of test scenarios. Commits are accepted only when both validation metrics and safety constraints are satisfied.

Runtime monitoring of refractory states and eligibility trace magnitudes can detect anomalous learning behavior, such as sudden widespread parameter changes or accumulation of extremely large eligibility traces, triggering alerts or automatic training suspension before catastrophic failures occur.

13. Limitations and Trade-offs

While NIEP offers substantial benefits for stability and efficiency in specific learning scenarios, it also introduces limitations and trade-offs that must be considered.

13.1 Increased Memory Requirements

NIEP requires maintaining additional state variables (r, e, B, $\tilde{w}$) alongside standard network parameters and optimizer states. This approximately quadruples per-parameter memory consumption compared to baseline training with only parameters and first-order optimizer states (e.g., momentum). For very large models approaching hardware memory limits, this overhead may be prohibitive.

Mitigation strategies include storing NIEP states in lower precision (FP16 or INT8), selectively applying NIEP only to critical layers while using standard updates for others, or implementing compressed state representations that exploit redundancy in refractory states and eligibility traces.

13.2 Hyperparameter Complexity

NIEP introduces approximately a dozen hyperparameters governing refractory dynamics, eligibility traces, budgets, and commits. While we provide initial guidelines, optimal values are task-dependent and may require tuning. This complexity increases the barrier to adoption and may deter practitioners accustomed to simpler training procedures.

Future work should focus on reducing effective hyperparameter dimensionality through principled defaults, automatic adaptation schemes that adjust hyperparameters during training based on observed dynamics, or meta-learning approaches that discover optimal hyperparameters across task distributions.

13.3 Reduced Benefits for Stationary, Large-Batch Scenarios

NIEP's mechanisms are most beneficial in continual, online, or small-batch learning scenarios where stability and selective updating provide clear advantages. In traditional large-batch, stationary-distribution training—where all data is available upfront and gradient estimates are low-variance—the benefits of refractory gating and eligibility smoothing may be minimal. The computational overhead of safe-commit evaluations may outweigh benefits in such scenarios.

NIEP is therefore best viewed as a specialized technique for challenging learning regimes rather than a universal replacement for standard backpropagation. Practitioners should assess whether their application exhibits characteristics (sequential tasks, non-stationary distributions, resource constraints, safety requirements) that align with NIEP's strengths.

13.4 Validation Data Requirements

The safe-commit protocol requires access to validation data that is representative of all tasks or data distributions the model should preserve. In some applications, obtaining and maintaining such validation data may be challenging. If validation data is biased or unrepresentative, safe-commit may fail to prevent catastrophic interference on underrepresented tasks.

Strategies to address this include maintaining diverse, continually updated validation sets, using uncertainty-based sampling to ensure validation coverage of edge cases, or employing generative models to synthesize validation data for previously learned tasks.

14. Future Research Directions

NIEP establishes a foundation for temporally structured, biologically inspired learning in artificial neural networks, but numerous opportunities for extension and refinement remain.

14.1 Theoretical Analysis

Rigorous mathematical analysis of NIEP's convergence properties, stability guarantees, and optimality remains an important open problem. Key theoretical questions include:

- Under what conditions on loss landscapes, gradient noise, and hyperparameters does NIEP converge, and at what rate?
- Can we derive formal bounds on forgetting in continual learning scenarios as a function of NIEP hyperparameters?
- What is the information-theoretic capacity of NIEP's eligibility trace mechanism for temporal credit assignment?
- How do refractory periods and error budgets interact with the loss landscape curvature and conditioning?

Developing this theoretical framework would provide principled guidance for hyperparameter selection and enable formal verification of NIEP's properties for safety-critical applications.

14.2 Efficient Implementation

Optimized implementations of NIEP for modern hardware platforms remain an important practical direction. This includes developing CUDA kernels for fused NIEP operations that minimize memory traffic, exploring low-precision implementations where NIEP states are stored in INT8 or even binary formats, and designing specialized hardware accelerators that provide native support for refractory gating and eligibility accumulation.

Neuromorphic implementations are particularly promising, as NIEP's leaky integrators and event-driven updates align naturally with neuromorphic computing paradigms. Demonstrating efficient NIEP implementations on neuromorphic platforms could enable ultra-low-power continual learning for edge applications.

14.3 Combination with Complementary Techniques

NIEP can be combined with other continual learning approaches for potentially synergistic effects. Orthogonality-constrained methods such as Orthogonal Weights Modification (OWM) prevent updates from interfering with previously learned subspaces. Combining OWM's geometric constraints with NIEP's temporal gating and safe-commit could provide both spatial and temporal protection mechanisms.

Task-specific parameter isolation techniques that dedicate subsets of parameters to individual tasks could be enhanced by NIEP's adaptive gating, allowing the system to automatically discover which parameters should be task-specific versus shared. Replay-based methods that maintain example buffers from previous tasks could use NIEP's safe-commit protocol to determine when replay is necessary.

14.4 Benchmark Development

Comprehensive evaluation of NIEP requires standardized benchmarks spanning diverse continual learning scenarios. Proposed benchmarks include:

Split-MNIST: Sequential learning of digit classification tasks formed by splitting MNIST classes into groups.

Permuted-CIFAR: Learning sequences of image classification tasks where each task involves differently permuted pixel orderings of CIFAR-10 or CIFAR-100.

Continual NLP: Sequential learning of natural language processing tasks such as sentiment analysis, named entity recognition, and question answering on diverse text corpora.

RL Continuous Control: Reinforcement learning scenarios where reward functions or environment dynamics change over time, requiring adaptation while maintaining stable policies.

Federated Multi-Domain: Distributed learning across heterogeneous data sources (e.g., medical imaging from multiple hospitals) where each source has different data distributions and privacy constraints.

Establishing these benchmarks with standardized evaluation protocols, baseline comparisons, and public leaderboards would accelerate NIEP research and enable rigorous comparison with alternative approaches.

14.5 Adaptive and Meta-Learning Extensions

Current NIEP formulations use fixed hyperparameters throughout training. Adaptive extensions could adjust hyperparameters based on observed learning dynamics, such as increasing refractory sensitivity when catastrophic interference is detected or relaxing budgets when learning plateaus. Meta-learning approaches could learn hyperparameter schedules or even parameter-specific hyperparameters across distributions of continual learning tasks, enabling NIEP to automatically configure itself for new application domains.

15. Conclusion

Neuro-Inspired Error Propagation represents a paradigm shift in how artificial neural networks learn, moving from unconditional simultaneous updates toward temporally structured, locally gated, and validation-verified learning. By incorporating refractory periods, eligibility traces, error budgets, and safe-commit protocols, NIEP addresses fundamental challenges in continual learning, resource-constrained adaptation, and safe deployment of learning systems.

The core insight of NIEP is that biological neural networks' temporal and spatial structure in learning is not merely an implementation detail but a fundamental design principle that provides stability, efficiency, and robustness. Artificial systems can benefit from similar mechanisms, translated into computationally precise and practically implementable forms.

NIEP's multi-component architecture provides defense in depth against catastrophic interference. Refractory gating prevents rapid successive modifications to recently updated parameters. Eligibility traces smooth gradient noise and enable temporal credit assignment. Error budgets constrain update magnitudes based on gradient statistics. Safe-commit ensures that proposed updates genuinely improve performance before permanent application. Each mechanism independently contributes to stability, and their combination provides synergistic effects exceeding the sum of individual contributions.

The practical implications of NIEP extend across multiple domains. In continual learning, NIEP enables accumulation of knowledge over extended task sequences without catastrophic forgetting. In edge computing, reduced update frequency and event-driven computation lower energy consumption. In safety-critical systems, validation-based commits provide formal guarantees against performance degradation. In federated learning, local stability mechanisms reduce variance and conflicts in distributed updates.

While NIEP introduces additional complexity—increased memory requirements, multiple hyperparameters, validation data dependencies—these costs are justified in scenarios where stability, safety, and efficiency are paramount. NIEP is not intended as a universal replacement for standard backpropagation but rather as a specialized technique for challenging learning regimes where traditional approaches fail.

The path forward involves rigorous theoretical analysis to formalize convergence properties and stability guarantees, efficient implementations leveraging modern hardware and neuromorphic platforms, comprehensive empirical evaluation across diverse benchmarks, and exploration of combinations with complementary continual learning techniques.

Through these efforts, the vision of stable, efficient, and safe incremental learning in artificial systems moves closer to practical realization.

NIEP demonstrates that insights from neuroscience, translated carefully into computational form, can address pressing challenges in modern machine learning. As neural networks are deployed in increasingly dynamic, resource-constrained, and safety-critical environments, biologically inspired learning mechanisms such as NIEP will become not merely interesting research directions but essential components of practical AI systems.

References

This conceptual paper builds upon established frameworks in gradient-based optimization, continual learning, and computational neuroscience. Core references include foundational work on backpropagation and stochastic gradient descent, catastrophic interference and stability-plasticity trade-offs in neural networks, eligibility traces and temporal credit assignment in reinforcement learning, leaky integrators and refractory periods in biological neural models, safe learning and validation-based model selection, and federated and asynchronous distributed training. No proprietary or unpublished data were used in the development of this theoretical framework.

Acknowledgments

The author thanks the cognitive neuroscience and artificial intelligence research communities for decades of foundational work that makes conceptual synthesis possible. Special acknowledgment to the pioneering work on complementary learning systems, meta-cognition, and conflict monitoring that inspired this architectural framework.

Disclosure Statements

Conflict of Interest Statement: The author declares no conflicts of interest.

Funding: This research received no external funding.

Data Availability Statement: No new data were created or analyzed in this study.

Context of Publication

This document was prepared by Picyboo Cybernetics Inc. (CA) as part of its ongoing research and development activities in foundational computational architectures integrating neural-cognitive mechanisms for next-generation systems. It reflects theoretical and technical considerations derived from internal experimentation and design work related to Picyboo technologies and affiliated products. While not exhaustive, it represents a portion of the company's broader research efforts and is shared publicly to contribute to open scientific and technological discourse.

License and Authorship Notice

Appendix: Pseudocode Implementation

The following pseudocode provides a concrete implementation template for NIEP, suitable for adaptation to specific deep learning frameworks.

```
# Initialize NIEP state variables
r = zeros_like(parameters)  # refractory states
e = zeros_like(parameters)  # eligibility traces
B = ones_like(parameters) * initial_budget  # error budgets
w_tilde = copy(parameters)  # shadow weights

# Hyperparameters
alpha, beta, gamma = 0.9, 0.05, 0.05  # refractory dynamics
lambda_decay = 0.9  # eligibility decay
kappa, T = 0.3, 0.05  # gating threshold and temperature
rho, delta, xi = 0.9, 1.0, 0.1  # budget dynamics
eta = 0.001  # learning rate
chi = 0.25  # commit rate
tau = 0.01  # validation tolerance

for batch in training_data:
  # Standard forward and backward pass
  activations = forward(batch, parameters)
  loss = compute_loss(activations, batch.targets)
  gradients = backward(loss)

  # Update refractory states
  r = alpha * r + beta * abs(activations) + gamma * abs(gradients)

  # Compute gating values
  G = sigmoid((kappa - r) / T)

  # Update eligibility traces
  e = lambda_decay * e + G * gradients

  # Update error budgets (variance-aware)
  grad_var = variance(gradients)
  budget_increment = delta / sqrt(grad_var + epsilon)
  B = maximum(0, rho * B + budget_increment - xi * abs(e))

  # Clip eligibility by budget
  e_clipped = clip_by_budget(e, B)

  # Apply to shadow weights (can integrate with optimizer)
  w_tilde = w_tilde - eta * optimizer_transform(e_clipped)
```

```
    # Periodic safe-commit evaluation
    if iteration % commit_frequency == 0:
        val_loss_current = evaluate(parameters, validation_data)
        val_loss_shadow = evaluate(w_tilde, validation_data)

        delta_loss = val_loss_shadow - val_loss_current
        calibration_improved = check_calibration(w_tilde, parameters)

        if delta_loss <= tau and calibration_improved:
            # Commit shadow weights to working weights
            parameters = parameters + chi * (w_tilde - parameters)
        # else: defer commit, shadow weights continue accumulating

    # Periodic refractory relaxation (global decay)
    if iteration % relaxation_frequency == 0:
        r *= alpha  # Apply decay to all refractory states

def clip_by_budget(eligibility, budget):
    """Clip eligibility trace magnitude to respect error budget"""
    magnitude = abs(eligibility)
    scale = minimum(1.0, budget / (magnitude + epsilon))
    return eligibility * scale
```

This implementation provides the core NIEP loop, showing how refractory gating, eligibility accumulation, budget management, and safe-commit integrate into a cohesive training procedure. Framework-specific optimizations, such as vectorized operations, fused kernels, and efficient masking, can be applied to this template for production deployments.