Ablation-Driven Optimization of Spiking Neural Networks for Bank Account Fraud Detection

Dylan Perdigão, Francisco Antunes, Catarina Silva, and Bernardete Ribeiro

Centre for Informatics and Systems of the University of Coimbra, Department of Informatics Engineering, University of Coimbra, Coimbra, Portugal
{dgp, fnibau, catarina, bribeiro}@dei.uc.pt

Abstract. Spiking neural networks (SNNs) have emerged as the next paradigm in artificial intelligence (AI), offering potential energy efficiency for real-world deployments. However, training these networks on highly imbalanced datasets remains challenging due to their high-dimensional parameter spaces. This paper presents three ablation studies on optimization strategies for SNNs in classification tasks, analyzing membrane decay, firing thresholds, temporal window size, and surrogate gradient slopes. Our results indicate that using fixed parameter initialization while learning only the threshold, combined with smaller temporal windows and architecture-specific surrogate gradient slopes, yields the most stable performance. We evaluate these optimized SNNs on the Bank Account Fraud suite against traditional machine learning algorithms. While boosting ensembles maintain strong performance for this classification task, our optimized SNNs demonstrate competitive performance measured by recall under a 5% false positive rate, while achieving higher fairness in terms of predictive equality of sensitive attributes. Source code is available at https://github.com/DylanPerdigao/SNN-Ablation-Optimization.

Keywords: Spiking Neural Networks · Neuromorphic Computing · Neuronal Dynamics · Bayesian Optimization · Responsible AI.

1 Introduction

Spiking neural networks (SNNs), widely recognized as the third generation of artificial neural networks (ANNs) introduced by Maass [9], mark a fundamental shift from their predecessors. Unlike conventional multi-layer perceptrons and continuous deep learning models, SNNs encode and transmit information through sparse, asynchronous sequences of binary spikes. This event-driven mechanism closely mimics the neuro-biological behavior of the human brain, in which a neuron generates an action potential only when its accumulated electrical charge exceeds a critical threshold, triggering depolarization driven by ion exchange [7]. The leaky integrate-and-fire (LIF) neuron model [5] has become the main choice for neuromorphic computing to simulate their neuronal dynamics,

offering an optimal trade-off between biological plausibility and computational complexity [3]. The LIF neuron's membrane potential dynamics can be modeled continuously by the differential equation

$$\tau \ \frac{dU(t)}{dt} = -\left[U(t) - U_{rest}\right] + R \ I(t), \tag{1}$$

where t is the time, U is the membrane potential, U_{rest} the resting membrane potential, R the resistance, I the input current, and τ is the time constant of the circuit. To simulate this behavior, namely for computational implementation, a solution of the differential equation can be expressed as

$$U[t] = \beta \ U[t-1] + W \ I[t] - \vartheta_{thr} \ S[t-1], \tag{2}$$

yielding a representation in discrete time. In this formulation, W is the input weight matrix, S is the binary output spike generated by the neuron in the previous timestep, β is the membrane potential decay over time, and ϑ_{thr} is the firing threshold. Both β and ϑ_{thr} are learnable parameters specific to SNNs [1]. Furthermore, because the binary spike function is non-differentiable, standard backpropagation through time is not possible; a surrogate gradient is used to approximate the derivative of the step function during the backward pass, allowing for backpropagation and avoiding the dead neuron problem [10]. SNNs comprise more learnable parameters than traditional ANNs, namely with their membrane thresholds and decays, making them more difficult to train effectively. This motivates our study of SNN parameter optimization within the domain of bank account fraud detection. The main contributions of this paper are as follows: (i) determining the optimal strategy for LIF decay and threshold learning; (ii) identifying the optimal number of simulation timesteps for neuronal dynamics; (iii) defining the ideal surrogate gradient slope to ensure effective backpropagation; Finally, we evaluate the performance of our SNNs against other machine learning (ML) algorithms on different bank account fraud detection problems.

2 Related Works

The parameterization and architectural optimization of SNNs on financial fraud benchmarks have been the subject of recent research. However, the roles of membrane dynamics, simulation timesteps, and surrogate gradient slope remain underexplored. Previous studies explored Bayesian optimization to tune the hyperparameters of convolutional spiking neural networks (CSNNs) with an in-depth analysis of the number of LIF neurons [15] or using population coding mechanisms [12]. However, these works did not deeply investigate the isolated impact of the different parameters of SNNs. Additionally, recent frameworks, such as CalibraSNN, have been developed to estimate the energy consumption of these networks, validating their efficiency across different optimized network topologies [11]. Despite these advances with bank account fraud problems, other relevant works have explored SNNs, namely with the LitE-SNN framework

that incorporates spatial and temporal compression within the neural architecture search process [8], with a scalable framework for parallel hyperparameter optimization of SNNs [2], or with FSpiNN, an optimization framework for memory- and energy-efficient SNNs [14]. Beyond simple optimization pipelines, the SCWO-SNN framework adds federated learning, providing efficient privacy-preserving techniques into SNNs [6].

3 Methodology

Our ablation studies[1] are conducted with two distinct SNN architectures described in Table 1: a 1D convolutional spiking neural network (CSNN) and a feed-forward spiking neural network (FFSNN). The use of a convolutional architecture for tabular data is justified by the fact that features are ordered by importance, making these networks suitable for this use-case and providing an alternative to FFSNN models.

These SNNs are optimized and evaluated mainly on the `Base` variant of the Bank Account Fraud (BAF) suite of datasets [4], a real-world benchmark comprising 6 dataset variants – with 1 million instances each – of online account-opening data collected over 8 months, with a fraud prevalence of only 1%. Following the authors' recommendations [4], the first 6 months are dedicated to training, and the final 2 months to testing. Given the class imbalance, model performance is primarily evaluated using the true positive rate (TPR), with the false positive rate (FPR) constrained to remain below 5%, due to business constraints. The feature space comprises 31 variables, categorized into client-specific data (e.g., income, employment status, housing status) and system-derived telemetry (e.g., session length, device fraud count, velocity metrics). To ensure responsible artificial intelligence (AI) practices, the dataset includes three sensitive attributes – clients' age, income, and employment status – imposing fairness constraints to prevent discriminatory biases. Specifically, we evaluate model fairness regarding the clients' age using the predictive equality (PE) metric [13]. The ratio is calculated by dividing the FPR of the group with the lowest observed FPR by the FPR of the group with the highest FPR, such that

$$PE = \min\left(\frac{FPR_{\text{yng}}}{FPR_{\text{old}}}, \frac{FPR_{\text{old}}}{FPR_{\text{yng}}}\right). \tag{3}$$

A PE value closer to 1 indicates a fairer model, ensuring that legitimate client groups do not face higher account blocking rates than other groups.

To process the continuous tabular features within the spiking domain, we use a rate-coding mechanism at the input layer. Specifically, the normalized numerical features are converted into spike trains, where the probability of generating a spike at any given simulation timestep is proportional to the normalized value

[1] Source code available on GitHub at https://github.com/DylanPerdigao/SNN-Ablation-Optimization, and archived on Zenodo at https://doi.org/10.5281/zenodo.22729659.

Table 1: Architectural specifications of the evaluated SNNs. For all convolutional layers, we use a filter size $K = 3$, padding $P = 1$, and stride $S = 1$.

Convolutional SNN Layers	**Feed-Forward SNN Layers**
Conv(1, 8) → BN → LIF → MP(2)	FC(31, 64) → BN → LIF
Conv(8, 16) → BN → LIF → MP(2)	FC(64, 32) → BN → LIF
Conv(16, 32) → BN → LIF → MP(2)	FC(32, 2) → LIF
Flatten → FC(96, 2) → LIF	

Abbreviations – **Conv**(C_{in}, C_{out}): Convolutional Layer, **FC**(N_{in}, N_{out}): Fully Connected Layer, **BN**: Batch Normalization, **LIF**: Leaky Integrate-and-Fire, **MP**(K): Max Pooling.

Table 2: Structure of the ablation studies and respective configurations.

Exp.	**Configurations**		**Initialization**	**Decay (β)**	**Threshold (ϑ)**
A	Fixed Params.	A1	$\beta = 0.9, \vartheta = 1.0$	Fixed	Fixed
		A2	$\beta = 0.9, \vartheta = 1.0$	Learned	Learned
		A3	$\beta = 0.9, \vartheta = 1.0$	Learned	Fixed
		A4	$\beta = 0.9, \vartheta = 1.0$	Fixed	Learned
	Opt. Params.	A5	$\beta \in [0, 1], \vartheta \in [0, 10]$	Fixed	Fixed
		A6	$\beta \in [0, 1], \vartheta \in [0, 10]$	Learned	Learned
		A7	$\beta \in [0, 1], \vartheta \in [0, 10]$	Learned	Fixed
		A8	$\beta \in [0, 1], \vartheta \in [0, 10]$	Fixed	Learned
B	Timesteps	T	$T \in \{5, 10, 20, 30, 40, 50\}$	Fixed ($\beta = 0.9$)	Fixed ($\vartheta = 1.0$)
C	Sigmoid Slope	σ	$\sigma \in \{5, 10, 20, 30, 40, 50\}$	Fixed ($\beta = 0.9$)	Fixed ($\vartheta = 1.0$)

of the feature. The readout layer is decoded by aggregating the spike counts per class over the entire simulation window. The predicted label becomes the class with the most spikes. Additionally, we apply a softmax function to obtain normalized pseudo-probabilities, thereby enabling the construction of the receiver operating characteristic (ROC) curve. The experiments, presented in Table 2, are structured into three distinct ablation studies (`A`, `B`, `C`), corresponding to (`A`) the analysis of the best strategy for decay and threshold learning, (`B`) the ideal number of timesteps, and (`C`) the best surrogate gradient slope for backpropagation. Then, the best resulting configuration for CSNN and FFSNN models is compared across all variants of the BAF suite of datasets against different tuned models, such as AdaBoost, CatBoost, decision tree, isolation forest, LightGBM, logistic regression, naive Bayes, random forest, and XGBoost. The main objective is the maximization of the TPR at a specific point on the ROC curve, where the threshold guarantees an FPR below 5%. The training loop integrates a patience-based early stopping mechanism. The models are trained for a maximum of 100 epochs, with a patience of 5 epochs for performance improvements. The loss is updated using the Adam optimizer, with weights $(0.0106, 0.9894)$ corresponding to the inverse proportions of samples in the training set for the non-fraud and fraud classes, respectively. The optimization of the initial parameters is performed using a tree-structured Parzen estimator [16], which suggests new values for the search space for each run.

4 Results and Discussion

4.1 Experiment A: Membrane Decays and Thresholds

To determine the optimal configuration for the LIF membrane parameters, we evaluated eight distinct optimization paradigms (`A1-A8`). These configurations explore the role of parameter initialization and gradient-based learning. Specifically, configurations `A1-A4` use fixed initializations ($\beta = 0.9$, $\vartheta = 1.0$), whereas `A5-A8` select these initial parameters via Bayesian optimization. Within these two initialization strategies, we distinguish four learning directions: static parameters with no gradient updates (`A1, A5`), backpropagation for both parameters (`A2, A6`), isolated learning of the membrane decay (`A3, A7`), and isolated learning of the firing threshold (`A4, A8`).

Figures 1 and 2 reveal that dynamically defining initial parameters via optimization (`A5-A8`) introduces training instability. While the CSNN exhibits higher average TPR under `A2`, the empirical peak performance across both architectures is achieved by `A4` (fixed initialization, learning only the threshold), yielding a maximum TPR of 52.10% (CSNN) and 50.56% (FFSNN). Fixing the membrane leakage while isolating the threshold reduces the dimensionality of the optimization space. This prevents competing gradients between decay and threshold, stabilizing backpropagation and enabling more effective learning in this highly imbalanced scenario.

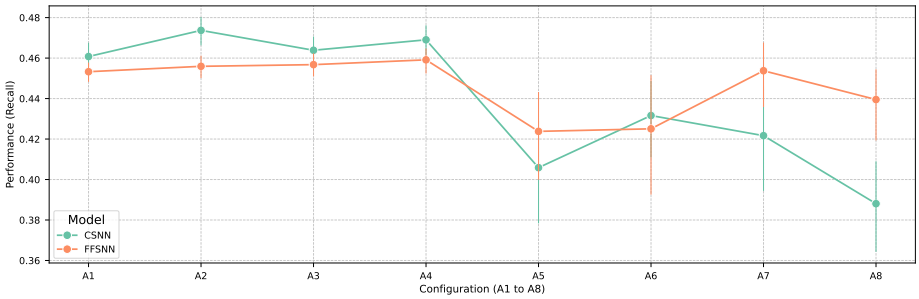


Fig. 1: Trade-off between mean performance and LIF parameters optimization strategies.

Pairwise Mann-Whitney U tests, illustrated in the heatmap of Figure 3, confirm these observations. For the CSNN architecture, the fixed-parameter configurations (`A1-A4`) show higher statistical significance than the optimized-initial parameters. While `A4` achieves its maximum recall, the statistical divergence between `A4` (threshold learning) and configurations that learn both parameters (`A2`) or exclusively the decay (`A3`) is smaller, suggesting that gradient-based learning yields greater improvements. On the other hand, for the FFSNN model, the statistical disparity between isolating a single parameter versus learning both is

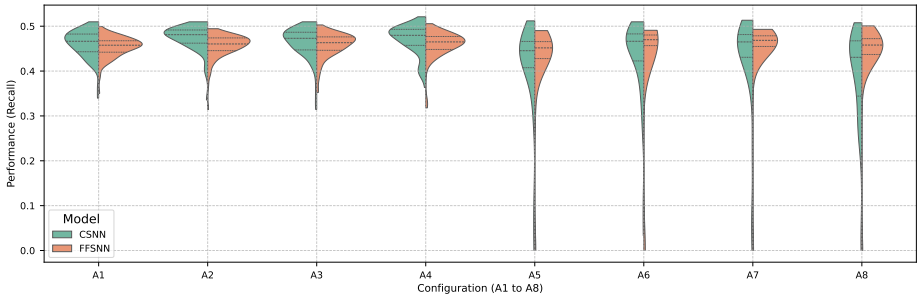


Fig. 2: Performance distribution across distinct LIF parameters optimization strategies.

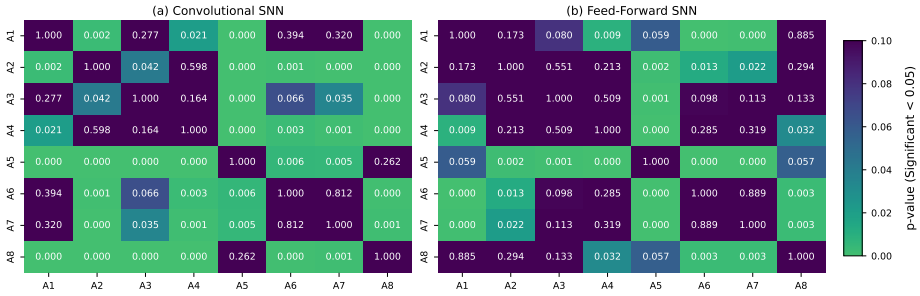


Fig. 3: Pairwise statistical significance of LIF membrane optimization strategies.

less pronounced. Nevertheless, `A4` remains an optimal choice due to its statistical significance compared to the fixed-parameter models without gradient-based learning (`A1, A5`).

4.2 Experiment B: Simulation Timesteps

In our second ablation study, we investigated the impact of the temporal simulation window, defined by the number of timesteps ($T \in \{5, 10, 20, 30, 40, 50\}$), on the TPR performance of the two previous architectures. The timestep hyperparameter determines the duration over which LIF neurons can increase their membrane potential and, consequently, fire.

Figure 4 shows optimal performance at `T=10` for both models, with performance degrading for longer windows. Interestingly, at `T=5`, the FFSNN outperforms the CSNN, as feed-forward architectures propagate spikes more rapidly without the spatial aggregation delays inherent to convolutions. The superiority of smaller temporal windows (`T=10`) is highly advantageous since it reduces the number of floating-point operations, translating into significant energy savings and lower inference latency. The empirical distributions presented in the violin plots of Figure 5 provide deeper insights into the variance and the absolute upper bounds of these configurations. While `T=10` shows lower disparity between

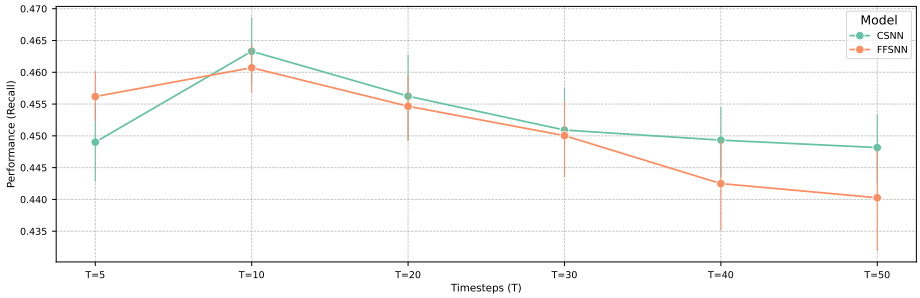


Fig. 4: Trade-off between mean performance and LIF simulation timesteps.

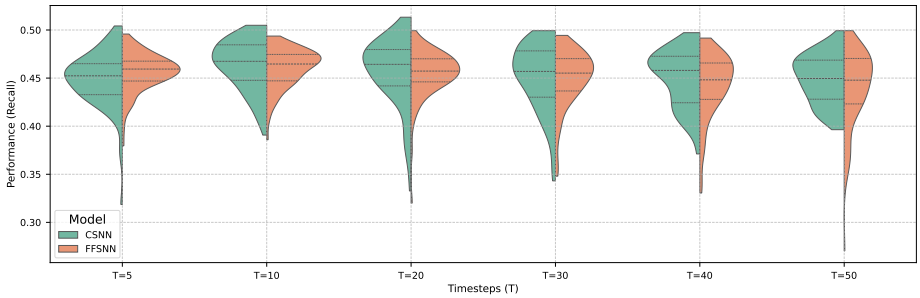


Fig. 5: Performance distribution across distinct LIF simulation timesteps.

trials, resulting in a more stable and consistent optimization performance, the maximum TPR across all trials is located at the `T=20` configuration, peaking at 52.33% TPR for the CSNN model and 49.93% TPR for the FFSNN model.

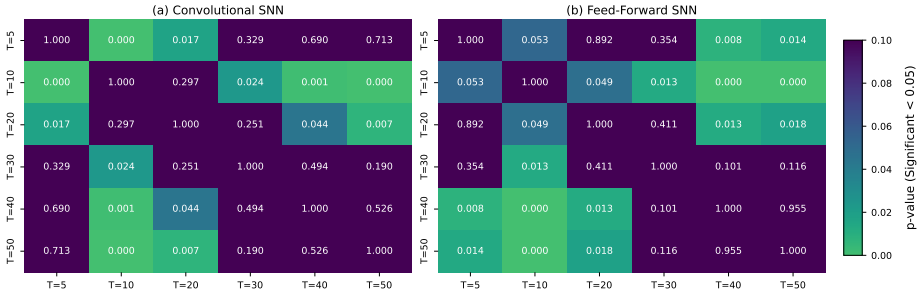


Fig. 6: Pairwise statistical significance of LIF simulation timesteps.

Additionally, we conducted pairwise Mann-Whitney U tests, as shown in the heatmap of Figure 6. The statistical analysis reveals that the `T=10` configuration

shows significant differences compared to the other temporal windows, except for `T=20` in the CSNN model. There is no statistical divergence between `T=10` and `T=20`. Furthermore, for configuration `T=20`, the difference is insignificant compared to `T=30`. On the side of the FFSNN model, the configuration `T=10` presents a significant difference compared to all other configurations, while `T=20` presents no statistical difference with `T=5` and `T=10`.

4.3 Experiment C: Surrogate Gradient Slope

The third ablation study investigates the impact of the surrogate gradient's slope ($\sigma \in \{5, 10, 20, 30, 40, 50\}$) on the network's backward pass, specifically its effect on gradient approximation with the sigmoid function affecting the learning dynamics for both the CSNN and FFSNN architectures. The surrogate gradient allows backpropagation through time since SNN spikes are non-differentiable.

Analyzing the mean TPRs in Figure 7, we observe different sensitivities to the slope of the surrogate gradient. The CSNN demonstrates superior average performance when using a relatively shallow slope (σ=5). On the other hand, the optimal performance for the FFSNN model is achieved at σ=30. While the CSNN generally outperforms the FFSNN across most configurations, the feed-forward architecture surpasses the convolutional network at σ=50. The violin plots in Figure 8 provide an overview of the trial performance distributions. At the lowest slope steepness (σ=5), the FFSNN shows more training instability, resulting in poor-performing trials. Interestingly, while the mean performance peaks differ between architectures, both models achieve their maximum recall at σ=10, with the CSNN reaching 51.61% TPR and the FFSNN attaining 50.0% TPR. Furthermore, the density distributions reveal that $\sigma \in \{20, 30, 40\}$ yield more common trial distributions for both architectures.

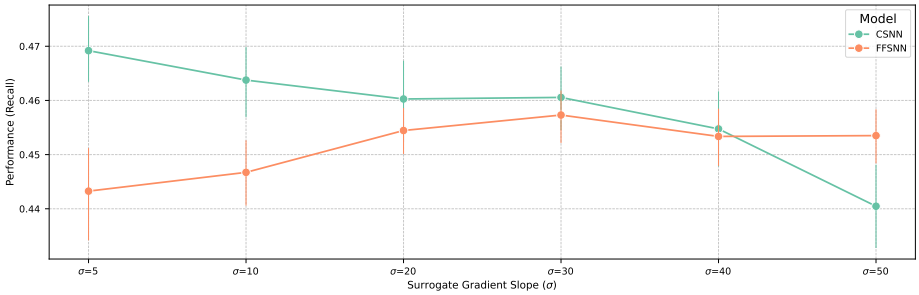


Fig. 7: Trade-off between mean performance and surrogate gradient slopes.

The pairwise Mann-Whitney U tests, detailed in the heatmap of Figure 9, show that, for the CSNN model, the shallow slope (σ=5) exhibits a statistically significant divergence relative to configurations with $\sigma \geq$30. In contrast, for

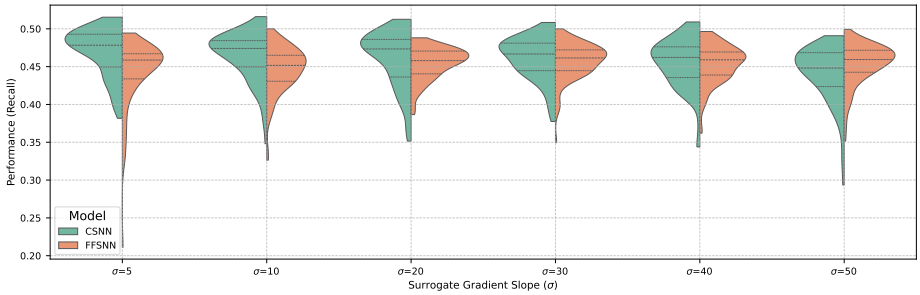


Fig. 8: Performance distribution across distinct surrogate gradient slopes.

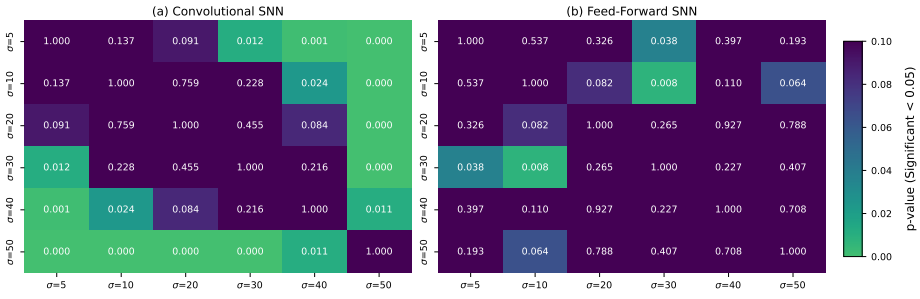


Fig. 9: Pairwise statistical significance of surrogate gradient slopes.

the FFSNN, the σ=30 shows statistical significance only when the slope values compared are strictly under 20.

4.4 Final Evaluation

Having established the optimal SNN configurations that maximize performance, we subjected these optimized models to a benchmark against traditional ML algorithms. Since tree-based ensembles historically dominate tabular dataset tasks, they serve as our primary baseline to validate the SNNs' competitiveness. We evaluated AdaBoost (ADA), CatBoost (CAT), decision tree (DT), isolation forest (IF), LightGBM (LGB), logistic regression (LR), naive Bayes (NB), random forest (RF), and XGBoost (XGB).

As illustrated in Figure 10a, the empirical results indicate a dominance of tree-based ensemble methods within traditional ML models. Specifically, the boosting algorithms (ADA, CAT, LGB, and XGB) yielded the highest recall performance, achieving up to 58% recall on the base dataset (e.g., CAT at 58.61% and XGB at 58.54%). However, maximizing recall decreases fairness metrics for the age attribute, namely measured with the PE metric. This fairness metric (Figure 10b) fluctuates across the top-performing algorithms, reaching 35.54% for ADA, 42.45% for LGB, and 52.62% for XGB, highlighting challenges with the trade-off between performance and fairness for these ensemble algorithms.

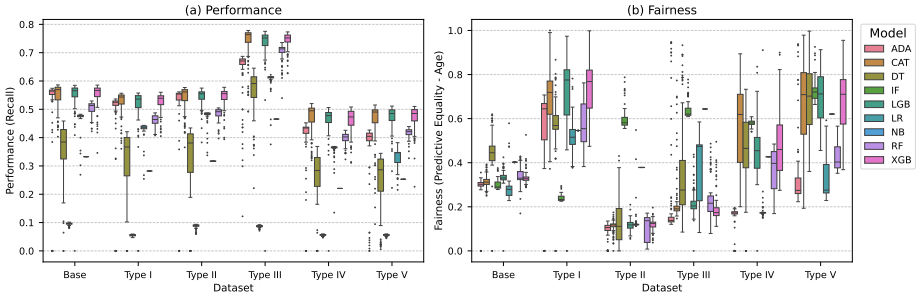


Fig. 10: Performance (a) and fairness (b) for traditional ML models across all variants of the BAF suite.

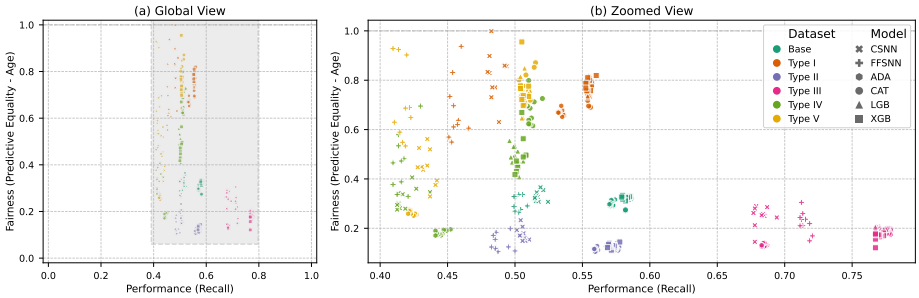


Fig. 11: Performance and fairness trade-off between the top 10 models of SNNs and boosting algorithms across all variants of the BAF suite.

Figure 11 illustrates the trade-off between performance (recall) and fairness (predictive equality) across all dataset variants. The scatter plot reveals that traditional boosting ensembles consistently dominate the Pareto front, achieving higher and more stable recall clusters than the optimized SNNs across all data bias scenarios. For instance, on the `Base` variant, boosting models cluster around 56–58% recall, whereas the optimized CSNN and FFSNN peak at 52.45% and 50.91%, respectively. Similarly, on the `Type III` dataset, ensembles reach nearly 78% recall, outpacing the SNNs which peak at 72.05%. Furthermore, the spiking architectures exhibit significant vertical variance (e.g., ranging from 0.5 to 1.0 in predictive equality on the `Type I` variant), indicating a high sensitivity to initialization conditions regarding fairness. Although the neuromorphic models do not strictly outperform boosting algorithms in pure predictive metrics on this tabular benchmark, they successfully maintain viable operational results while exhibiting higher overall fairness. This demonstrates their capacity to provide a competitive alternative for real-world deployments, where slight compromises in performance are acceptable in exchange for fairness and hardware-level energy efficiency.

5 Conclusion

This paper explored the optimization of SNNs for the highly imbalanced domain of bank account fraud detection. Through three ablation studies, we investigated LIF membrane dynamics, temporal window size, and surrogate gradient steepness. We found that isolating the learning of the firing threshold is more effective than learning the membrane decay simultaneously. Furthermore, fewer simulation timesteps improve performance, and the optimal surrogate gradient slope is architecture-dependent. Although boosting-based ensembles achieved the highest overall recall on each dataset variant, our optimized SNNs demonstrated competitive performance across different data bias patterns, while being fairer regarding sensitive attributes such as clients' age.

A main limitation of this study is the absence of direct energy and latency quantification. Future work will address this by compiling these optimized network topologies using frameworks like Sinabs for direct deployment on event-driven neuromorphic chips, such as the SynSense Speck. Validating these configurations on physical hardware will allow us to accurately quantify the theoretical energy savings and operational latency, properly benchmarking SNN efficiency against traditional ML models.

Acknowledgments. This work was supported by the Portuguese Recovery and Resilience Plan (PRR) through project PRR-C645008882-00000055, Center for Responsible AI. This work was also funded by national funds through FCT Foundation for Science and Technology, I.P., within the scope of the research unit UID/00326 Centre for Informatics and Systems of the University of Coimbra (CISUC) with DOI [10.54499/UID/00326/2025], and within the scope of the project Ref. 2024.01388.BD with DOI [10.54499/2024.01388.BD].

References

1. Eshraghian, J.K., Ward, M., Neftci, E.O., Wang, X., Lenz, G., Dwivedi, G., Bennamoun, M., Jeong, D.S., Lu, W.D.: Training Spiking Neural Networks Using Lessons From Deep Learning. Proceedings of the IEEE **111**(9), 1016–1054 (2023). https://doi.org/10.1109/JPROC.2023.3308088
2. Firmin, T., Boulet, P., Talbi, E.G.: Parallel Hyperparameter Optimization Of Spiking Neural Network. Neurocomputing **609**, 128483 (Dec 2024). https://doi.org/10.1016/j.neucom.2024.128483
3. Izhikevich, E.: Which model to use for cortical spiking neurons? IEEE Transactions on Neural Networks **15**(5), 1063–1070 (Sep 2004). https://doi.org/10.1109/TNN.2004.832719
4. Jesus, S., Pombal, J., Alves, D., Cruz, A.F., Saleiro, P., Ribeiro, R.P., Gama, J., Bizarro, P.: Turning the tables: biased, imbalanced, dynamic tabular datasets for ML evaluation. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. pp. 33563–33575. NIPS '22, Curran Associates Inc., Red Hook, NY, USA (Nov 2022)

5. Knight, B.W.: The relationship between the firing rate of a single neuron and the level of activity in a population of neurons. Experimental evidence for resonant enhancement in the population response. The Journal of General Physiology **59**(6), 767–778 (Jun 1972). https://doi.org/10.1085/jgp.59.6.767
6. Kolli, C.S., Jagannathan, V., Lanka, D.R., Rajyalakshmi, C., Radha, V.K.R., Reddy, V.V.K.: SCWO-SNN: Snow carpet weaver optimization based spiking neural network for fraud detection in UPI transactions using federated learning. Computer Standards & Interfaces **98**, 104166 (Aug 2026). https://doi.org/10.1016/j.csi.2026.104166
7. Lindsay, G.: Models of the Mind: How Physics, Engineering and Mathematics Have Shaped Our Understanding of the Brain. Bloomsbury Publishing (Mar 2021)
8. Liu, Q., Yan, J., Zhang, M., Pan, G., Li, H.: LitE-SNN: Designing Lightweight and Efficient Spiking Neural Network through Spatial-Temporal Compressive Network Search and Joint Optimization (May 2024). https://doi.org/10.48550/arXiv.2401.14652
9. Maass, W.: Networks of spiking neurons: The third generation of neural network models. Neural Networks **10**(9), 1659–1671 (1997). https://doi.org/10.1016/S0893-6080(97)00011-7
10. Neftci, E.O., Mostafa, H., Zenke, F.: Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-Based Optimization to Spiking Neural Networks. IEEE Signal Processing Magazine **36**(6), 51–63 (Nov 2019). https://doi.org/10.1109/MSP.2019.2931595
11. Perdigão, D., Antunes, F., Silva, C., Ribeiro, B.: CalibraSNN: Fair and Calibrated Convolutional Spiking Neural Network for High-Stakes Industry Applications. IEEE Access **13**, 210979–210995 (2025). https://doi.org/10.1109/ACCESS.2025.3641389
12. Perdigão, D., Antunes, F., Silva, C., Ribeiro, B.: Exploring Neural Joint Activity in Spiking Neural Networks for Fraud Detection. In: Hernández-García, R., Barrientos, R.J., Velastin, S.A. (eds.) Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. pp. 45–59. Springer Nature Switzerland, Cham (2025). https://doi.org/10.1007/978-3-031-76604-6_4
13. Pombal, J., Cruz, A.F., Bravo, J., Saleiro, P., Figueiredo, M.A.T., Bizarro, P.: Understanding Unfairness in Fraud Detection through Model and Data Bias Interactions (2022). https://doi.org/10.48550/arXiv.2207.06273
14. Putra, R.V.W., Shafique, M.: FSpiNN: An Optimization Framework for Memory-Efficient and Energy-Efficient Spiking Neural Networks. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **39**(11), 3601–3613 (Nov 2020). https://doi.org/10.1109/TCAD.2020.3013049
15. Ribeiro, B., Antunes, F., Perdigão, D., Silva, C.: Convolutional Spiking Neural Networks targeting learning and inference in highly imbalanced datasets. Pattern Recognition Letters **189**, 241–247 (Mar 2025). https://doi.org/10.1016/j.patrec.2024.08.002
16. Watanabe, S.: Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance (May 2023). https://doi.org/10.48550/arXiv.2304.11127