

SuperSNN: A Hardware-Aware Framework for Physically Realizable, High-Performance Superconducting Spiking Neural Network Chips

Changxu Song, Arda Caliskan, Beyza Zeynep Ucpinar, Yasemin Kopur, Mustafa Altay Karamuftuoglu, Sasan Razmkhah, Shahin Nazarian, Massoud Pedram
 University of Southern California, Los Angeles, USA
 {changxus, acaliska, ucpinar, karamuft, razmkhah, shahin.nazarian, pedram}@usc.edu

Abstract—Despite numerous proposed designs for superconducting neural networks (SNNs), most have overlooked practical fabrication constraints, leading to implementations limited to only a few neurons or synapses. Current superconducting technologies, such as MIT LL SFQ5ee, impose severe limitations on chip area, routing, and input/output pin counts (e.g., $5 \times 5 \text{ mm}^2$ chip with 40 pins), drastically restricting network size and complexity. These hardware constraints necessitate a comprehensive framework to tailor network designs for physical realizability while minimizing accuracy loss. This paper introduces SuperSNN, a comprehensive framework for the implementation of full superconducting SNNs on a chip within these constraints. The key technical contributions include: (1) A hardware-aware training methodology for SNNs, utilizing off-chip pruning and weight quantization for energy-efficient superconducting implementations. (2) Design and layout of an inference SNN chip that incorporates novel high fan-in neurons and custom superconducting cells. (3) An optimized locally synchronous, globally synchronous (LAGS) clock distribution scheme for robust circuit implementation and management of data transfer delays in SFQ SNNs. The main results and findings demonstrate the effectiveness of the framework: (1) The complete network achieved 96.47% accuracy on the full MNIST dataset after quantization and pruning. (2) The fabricated SuperSNN chip successfully classified a reduced set of digits (2, 3, and 4) with 80.07% accuracy, reaching a maximum of 86.2% accuracy for digits 0, 1, and 2. (3) The chip operates at an ultra-high 3.02 GHz clock frequency. (4) It occupies a compact area of $3.4 \times 3.9 \text{ mm}^2$, incorporates 5,822 Josephson Junctions, consumes 2.15 mW static power, and has an exceptionally low energy cost of 6.55 fJ (or $1.31 \times 10^{-6} \text{ nJ}$) per inference.

Index Terms—Superconductor electronics, Spiking neural network, Network quantization and pruning, Design framework

1. INTRODUCTION

Neuromorphic computing seeks to fundamentally transform data processing by mimicking the operation of biological nervous systems. Traditional computers rely on the Von Neumann architecture, which suffers from significant inefficiencies due to the physical separation of memory and processing units, necessitating continuous data transfer. Although alternative

This work is supported by a grant from the National Science Foundation (NSF) under the project Expedition: Discover (Design and Integration of Superconducting Computation for Ventures beyond Exascale Realization), grant number 2124453, and by the Department of Energy (DOE) AI4Science program with project name Flexbrain.

neural network architectures have been proposed to enhance performance, conventional approaches such as Artificial Neural Networks (ANNs), which depend heavily on computationally intensive convolution and matrix multiplication, face exponential increases in computational demands and power consumption as network size grows.

To bridge this energy gap, brain-inspired Spiking Neural Networks (SNNs) have been proposed [1]. SNN emulates the efficiency of the human brain, offering superior energy performance compared to ANN. This efficiency comes from their event-driven processing paradigm and binary spike activations. Unlike ANNs that continuously process data streams, SNNs are triggered only by input events and remain inactive otherwise; their binary spikes require only simple integer accumulations instead of multiply-accumulate operations, reducing latency and power consumption. In addition, low-precision temporal coding and the inherent sparsity of spike trains further minimize memory traffic, enhancing both speed and energy efficiency. When implemented on specialized neuromorphic hardware, this event-driven sparsity significantly reduces data movement between on-chip and off-chip resources.

Despite the many benefits of SNNs, exploring suitable alternative hardware architectures for them remains a challenge. Although neuromorphic optical computing offers low power and latency [2], reconfigurability and integration challenges make complex optical systems less practical compared to CMOS-based solutions. Memristor-based devices provide a CMOS-compatible option and are well suited for synaptic emulation due to their inherent variability [3]. However, they are analog devices requiring high precision, making noise unavoidable, and their dissipative nature during read/write operations presents a serious heating problem as systems scale.

Superconducting neuromorphic architectures offer a scalable solution to these issues, providing significantly improved energy efficiency compared to conventional electronic systems. Operating at cryogenic temperatures, they enable ultrahigh-speed signal propagation with near-zero resistive losses, ensuring zero data degradation [4]. Digital Single-Flux-Quantum (SFQ) circuits, leveraging superconducting technology, are asynchronous and spike-based, operating at tens of GHz with ultra-fast switching characteristics. They transmit data as

quantized pulses at high speeds. These characteristics make SFQ circuits a promising candidate for emulating the event-based nature and energy efficiency of SNN processing.

A broad spectrum of superconducting approaches has been explored to realize the advantages of SNNs using SFQ or similar superconductor logic families. Early work demonstrated mimicking spiking neuron behavior with Josephson Junctions (JJs), including leaky integrate-and-fire models simulated via SFQ pulse generation and thresholding [5]. Further investigations showed synchronization and coupling dynamics between JJ-based neurons on the picosecond time scale [6]. Developments focused on biologically plausible dynamics, with circuits fabricated that emulate action potentials, refractory periods, and thresholds, demonstrating high-speed and energy-efficient behavior [7]. The hybrid integration of photonics and superconductors emphasized the scaling potential [8]. Magnetic JJs (MJJs) facilitated tunable and stochastic synapses with sub-attojoule operation and analog weight updates [9], [10]. Various neuron implementations, such as nanowire-based designs [11] and quantum phase-slip junction (QPSJ) designs [12], [13], offered low-power switching and high fan-in. The hybrid RSFQ-QFP circuits introduced dynamic weight programming [14]. Architecture-level advancements included multilayer spike rate coded networks using purely SFQ logic, preserving asynchronous, ultralow-power operation [15]. The bioSFQ framework bridged the analog and digital domains, enabling SFQ-based analog operations [16], [17]. The learning mechanisms were incorporated through local STDP rules [18], [19], trainable neurons [20], and unsupervised networks [21]. Integration efficiency techniques such as ternary weights and temporal encoding reduce the area and the count of components [22], [23]. Processors such as SUSHI [24] and an ultra-high-speed processor demonstrated RSFQ / SFQ-based platforms with on-chip learning support [25].

Superconductor logic still faces significant challenges in physical circuit implementation. Although in-memory computing architectures such as SNNs partially address the lack of dense memory, a key challenge remains: implementing neurons with high fan-in and fan-out. Splitter circuits handle fan-out in SFQ. However, high fan-in is more complex. [26] introduced a superconductor neuron design using mutual coupling branches for the dynamics of somatic operation, which was used to design a SNN feedforward inference. The proposed network achieved 95.13% accuracy after pruning and quantization.

Despite this progress, a fully fabricated and inference-ready feedforward SNN implementation has not yet been realized. Given the potential for tens of GHz classification speeds, ultra-low power consumption, and a minimal footprint, developing such a chip remains a valuable goal. Although software implementations can achieve high accuracy on small datasets such as MNIST, the fabrication process imposes significant constraints. Current technologies, such as the MIT LL SFQ5ee process [27], feature a limited number of routing metal layers, a restricted chip area of $5 \times 5\text{mm}^2$, and a small number of input/output pads (40 pads placed uniformly around

the periphery of the chip), severely limiting the size of the networks that hardware can support. Furthermore, data transfer delays require careful synchronization design. These hardware limitations mean that the network must be tailored to physical constraints, often through compaction techniques with minimal loss of accuracy. Consequently, designing an SNN chip based on SFQ logic within these constraints presents substantial challenges.

In this work, we present a comprehensive framework for developing on-chip inference SNNs that comply with hardware constraints of superconductor electronics and successfully realize the chip layout. We utilize our newly designed SFQ standard cell libraries for manual layout, routing, and optimization. We introduce SuperSNN, an end-to-end framework for implementing fully superconducting SNNs on a chip using the MIT LL SFQ5ee fabrication process. Our framework encompasses network training, pruning to limit fan-in and fan-out, and weight quantization to ensure physical realizability. To demonstrate the effectiveness of our approach, we design, train, and implement an SNN for MNIST classification on a single $5 \times 5\text{ mm}^2$ chip. We employ a novel high fan-in neuron design optimized for scalability and utilize hybrid SFQ wiring cells for synaptic connections. The proposed architecture leverages the event-driven nature of SNNs and the ultra-high-speed, low-power advantages of SFQ circuits to overcome the inefficiencies present in conventional systems.

The main contributions of this paper are:

- Developing a hardware-aware training methodology for SNNs optimized for energy-efficient superconducting implementations. This framework enables off-chip training with pruning and weight quantization, which is crucial for physical realizability.
- Design of an inference SNN chip that incorporates high-fan-in neurons and custom superconducting cells. This resulting network is designed to operate at a clock frequency of 3.02 GHz and occupies a compact area of $3.4 \times 3.9\text{ mm}^2$.
- Implementation of a locally asynchronous, globally synchronous (LAGS) clock distribution scheme to ensure an optimized and robust circuit implementation of the SFQ SNN. This scheme helps address the challenge of data transfer delays in superconducting logic.

2. BACKGROUND

2.1. Leaky Integrate-and-Fire Neuron Model

Neurons are surrounded by a lipid membrane that electrically isolates the cell while allowing ion flow through gated channels. This membrane behaves like a capacitor, storing and integrating charge. The leaky integrate-and-fire (LIF) model captures this dynamic by abstracting neuronal behavior as temporal integration of inputs, rather than applying a direct activation function. The LIF neuron models membrane integration as an RC-like circuit, accumulating inputs until the membrane potential exceeds a threshold and triggers a spike. Rather than modeling the spike waveform, the LIF

model treats it as a discrete event, striking a balance between biological realism and computational simplicity. We adopt a simplified LIF model from `snnTorch` with a hard reset mechanism. This reset period effectively serves as a refractory phase, allowing for temporal filtering and contributing to spike-based computation. The spiking output is defined by the Heaviside function in Eq. 1:

$$S[t] = \begin{cases} 1, & \text{if } U[t] > U_{\text{thr}} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

where $S[t]$ is the binary spike output at time t , $U[t]$ is the membrane potential, and U_{thr} is the threshold. Each time step t becomes discrete, and at most one spike may occur. The membrane potential update rule is given in Eq. 2:

$$U[t+1] = \beta U[t] + W X[t+1] - S[t] U_{\text{thr}} \quad (2)$$

Here, β is a decay constant ($0 < \beta < 1$) that models the exponential leakage of the potential in the absence of input. W is the synaptic weight and $X[t+1]$ is the input spike in the next time step. The subtraction term $S[t] U_{\text{thr}}$ resets the membrane after a spike, enforcing the hard reset behavior. This formulation compactly models both integration and decay, closely resembling the dynamics of an RC circuit with time constant $\tau = RC$.

2.2. Training Methodology of The Network

When training neural networks composed of discrete, recurrent spiking neurons, these networks may be approximated as Recurrent Neural Networks (RNNs). Specifically, the computational graph of the spiking neurons over multiple time steps is first unrolled, as shown in Figure 1, followed by the application of backpropagation through time (BPTT).

At each time step, the loss gradient is computed with respect to the shared weights, and these gradients are summed across all time steps to obtain the total gradient for parameter updates. The gradient with respect to the weight parameter W can be expressed as:

$$\frac{\partial \mathcal{L}}{\partial W} = \sum_t \sum_{s \leq t} \frac{\partial \mathcal{L}[t]}{\partial W[s]}, \quad (3)$$

Here, $\mathcal{L}[t]$ denotes the loss in time step t , and $W[s]$ refers to the shared synaptic weight in time s . During backpropagation over time, a key step is computing the loss gradient at time t with respect to the weight at the previous step, $W[t-1]$. Assuming that the loss depends directly on spikes and is differentiable, this gradient is given by Eq. 4:

$$\begin{aligned} \frac{\partial \mathcal{L}[t]}{\partial W[t-1]} &= \frac{\partial \mathcal{L}[t]}{\partial S[t]} \cdot \frac{\partial \tilde{S}[t]}{\partial U[t]} \\ &\cdot \frac{\partial U[t]}{\partial U[t-1]} \cdot \frac{\partial U[t-1]}{\partial I[t-1]} \cdot \frac{\partial I[t-1]}{\partial W[t-1]} \end{aligned} \quad (4)$$

where $\tilde{S}[t]$ is a surrogate for the nondifferentiable spike output $S[t]$, allowing gradient-based optimization. $U[t]$ is the membrane potential and $I[t-1]$ is the input current at time

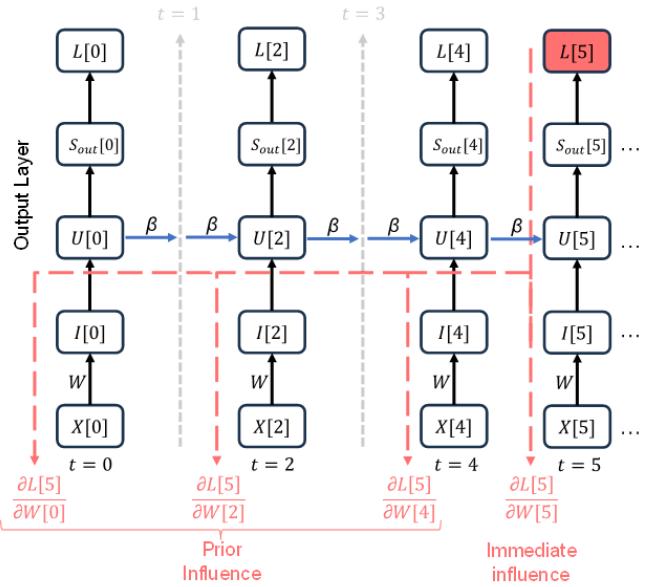


Fig. 1: Unrolling of a spiking neural network over multiple time steps.

$t-1$, depending on the weight $W[t-1]$. These quantities are linked via the LIF update rule in Eq. 2.

The partial derivatives in Eq. 4 arise directly from the membrane update rule and basic calculus. Their values are as follows:

- $\partial U[t]/\partial U[t-1] = \beta$, the decay factor, representing the influence of the previous membrane potential.
- $\partial U[t-1]/\partial I[t-1] = 1$, from the direct relationship between the input current and the membrane potential.
- $\partial I[t-1]/\partial W[t-1] = X[t-1]$, where $X[t-1]$ is the pre-synaptic input spike at time $t-1$.

Applying the chain rule, the immediate effect of the weight $W[t-1]$ on the membrane potential $U[t]$ is $\beta \cdot X[t-1]$. When spikes are used for training, $S[t]$ is typically defined by the Heaviside function, which is nondifferentiable at $x = 0$ and has zero derivative elsewhere, resulting in undefined or vanishing gradients. To avoid this, frameworks like `snnTorch` apply surrogate gradients, replacing the Heaviside function with an approximation during backpropagation. For instance, the arctangent surrogate yields $\partial \tilde{S}/\partial U = 1/[\pi(1+(U\pi)^2)]$, allowing stable, nonzero gradients.

Alternatively, if the loss is defined over the membrane potential instead of spikes, gradients can bypass $S[t]$ entirely, as shown in Eq. 5:

$$\frac{\partial \mathcal{L}[t]}{\partial W[t-1]} = \frac{\partial \mathcal{L}[t]}{\partial U[t]} \cdot \frac{\partial U[t]}{\partial U[t-1]} \cdot \frac{\partial U[t-1]}{\partial I[t-1]} \cdot \frac{\partial I[t-1]}{\partial W[t-1]} \quad (5)$$

In supervised multiclass classification, the predicted label corresponds to the output neuron with the highest spike count. To guide the network toward this behavior during training, we encourage the correct class neuron to spike more frequently by increasing its membrane potential beyond the threshold

$(U > U_{\text{thr}})$, while suppressing the potentials of incorrect class neurons $(U < U_{\text{thr}})$. This mechanism is implemented by applying a cross-entropy loss to the membrane potentials of the output layer, referred to as *mem loss*. As shown in Eq. 6, we first apply a softmax function to the membrane potentials:

$$p_i[t] = \frac{e^{U_i[t]}}{\sum_{j=0}^C e^{U_j[t]}} \quad (6)$$

where C denotes the number of classes and $U_i[t]$ is the membrane potential of the i th output neuron at time step t . We compute the softmax probabilities $p_i[t]$ over membrane potentials, and define the target label y_i as a one-hot encoded vector (Eq. 7). The cross-entropy loss is then computed between $p_i[t]$ and y_i , as shown in Eq. 8:

$$y_i = \begin{cases} 1, & i = \text{target class}, \\ 0, & i \neq \text{target class} \end{cases} \quad (7)$$

$$\mathcal{L}_{CE}[t] = - \sum_{i=0}^C y_i \log(p_i[t]) \quad (8)$$

This loss is applied at each time step, resulting in a total loss that is the sum over all time steps, as shown in Eq. 9:

$$\mathcal{L}_{CE} = \sum_t \mathcal{L}_{CE}[t] \quad (9)$$

The *mem loss* encourages a higher membrane potential for the correct class and suppresses it for incorrect classes, thus promoting spiking activity in the target neuron while inhibiting others. This loss is typically sufficient; however, under extreme constraints, such as reducing the network to a single forward pass and fewer than 25 neurons for deployment on specialized SNN hardware, *mem loss* alone may cause under-fitting.

To address this, we introduce an additional *spike loss*, forming a hybrid objective aligned with Multi-task Learning [28], [29]. Like *mem loss*, it applies cross-entropy, but to spike outputs $S[t]$. We compute a softmax over spike counts in the output layer and use the same one-hot target comparison, thereby encouraging the correct neuron to spike more often while suppressing spiking in others. The formulation mirrors that of *mem loss*, with the only difference being the substitution of membrane potentials $U_i[t]$ with spike outputs $S_i[t]$ in Eq. 6, resulting in a parallel cross-entropy loss computed over binary spiking activity.

2.3. Circuit Components

1) *Transmission Lines*: A Josephson Transmission Line (JTL) consists of a series of grounded, shunted, and biased JJs that are connected by inductors, which are configurable for different design requirements, such as transmission latency or peak SFQ pulse amplitude [30]. To simplify both the design and analysis, the bias currents and inductance are chosen to be identical in this design.

In addition to providing a significant margin, JTLs have drawbacks, such as occupying a large area and having limited wiring distance. Although the switching power is minimal for a

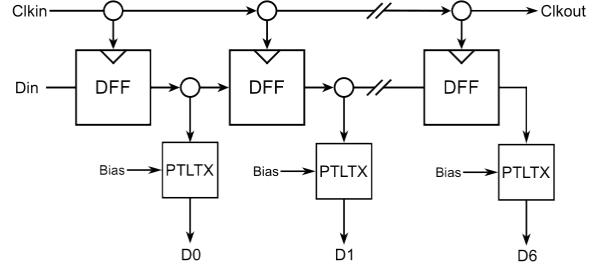


Fig. 2: Input shift register circuit. The clock follows the data path and propagates the data through the DFFs.

JTL, the cumulative power consumption becomes considerable for long-distance connections operating at frequencies greater than 10 GHz. Consequently, Passive Transmission Lines (PTL) that allow SFQ pulses to propagate at speeds approaching that of light with minimal signal attenuation are often used instead for designs that involve long distances. A PTL typically consists of a stripline connecting a driver and a receiver.

2) *D Flip-Flop (DFF)*: A destructive readout D flip-flop (DFF) typically consists of three Josephson junctions. To store a single flux quantum in the loop, the inductance L must satisfy the condition $L > \Phi_0/I_C$, which can be maintained indefinitely. A clock pulse is applied at the input terminal to read the state of the flip-flop, to redirect the stored current to the output, and generate an SFQ pulse.

3) *Input Shift Register*: To meet the 40-pin constraint of the chip, we implemented a shift register (SR) input scheme to minimize pin usage. The input layer consists of seven shift registers, each with seven DFFs in series, resulting in 49 total DFFs. The input SR layer allows a downsampled 7×7 image to be loaded bit-by-bit over seven data pins in seven clock cycles. Each register shares a common serial clock, implemented by chaining local clocks, enabling all registers to be driven from a single clock pin. As shown in Fig. 2, each DFF output is split—one path connects to the next DFF, and the other feeds the next layer. Biasing for the next layer is disabled until all bits are loaded, preventing premature activations.

The clock tree is implemented serially to reduce area overhead. This design introduces a 433 ps delay across the entire shift register path, limiting the system's throughput to just over 1 GHz. By contrast, an H-tree-based clock structure would shift the bottleneck to neurons and increase maximum throughput to 8.9 GHz.

4) *Fan-out Circuit*: SNNs can require large fan-outs while propagating data throughout the network. We created fan-out circuits built using cascaded splitters (SPL) cells of fan-out two to facilitate this requirement. The maximum fan-out is seen to be nine, which corresponds to eight cascaded SPL cells. In our library, each SPL cell introduces 3.1 ps of delay, resulting in approximately 24.8 ps of delay from the fan-out circuit.

5) *High Fan-in Neural Cell*: Each neuron receives input current via mutual inductive coupling through multiple den-

drites, each weighted as $+1$, -1 , or 0 . When the combined loop current and bias exceed the JJ's critical current, the neuron fires an SFQ pulse, which is transmitted to downstream neurons. For output neurons, the SFQ signal is converted to a DC level and routed to an output pin. For data synchronization, each input path includes a DFF, avoiding the need for manual delay balancing. A JTL is inserted on the positive path to introduce latency, ensuring that the negative pulse arrives first [26]. Fig. 4 shows the layout of an eight-input neuron with two dendritic branches, six positive and two negative. Synchronized input pulses propagate through the coupling network before reaching the neuron.

In the original neuron peripheral design, JTLs were used to interface with both fan-in and fan-out nodes before connecting to PTLs. In our implementation, the PTL receivers are directly integrated at the fan-in nodes, allowing the neuron and PTL interface to be optimized together as a single unit. To improve parametric yield, a wider timing margin is allocated for incoming pulses, enabling greater tolerance to variations in signal arrival times. However, this increased timing margin requires a lower decay rate in the leaky loop of the neuron, which ultimately reduces the maximum operating frequency.

6) Network Architecture: Using the described circuit components, we constructed a feedforward SNN composed of SPL cells, PTLs, and neuron cells. The input layer consists of 49 DFFs, followed by 18 and three neurons in the hidden and output layers, respectively. Inputs are loaded through a shift register and routed through SPL trees and PTLs, expanding 49 inputs into 97 outputs. These are received by 18 neuron cells, which use internal clock trees and DFFs for synchronization before coupling to the soma to trigger activation. Each neuron outputs a single bit, transmitted via PTL links to a second fan-out block that converts 18 inputs into 23 outputs. These are routed to the final three output neurons, whose spikes are converted to DC and routed off-chip to represent the classified digit.

3. SOFTWARE FRAMEWORK

3.1. Applying the Training Methodology on the SNN Network

The proposed training methodology demonstrates exceptional versatility and effectiveness, achieving high accuracy across both conventional and highly constrained superconductor electronics hardware and tapeout capabilities. In this study, we validate the approach using two variants of the inference network. The first, known as the complete network, performs classification on digits 0–9. Each neural cell supports up to 64 inputs in this configuration, without requiring uniformity across cells. There are no constraints on space or pin count, and output spikes from each neuron in the final layer are counted to determine the classification result. The second, called the chip network, is designed to operate under real superconducting hardware constraints. These include a chip with only 40 pins (7 input and three output pins) and limited space accommodating just 25 neurons. As a result, classification is restricted to digits 2, 3, and 4. In addition, the chip cannot count the total number of output spikes. Therefore, the

chip network must simulate the behavior of an inference SNN within all these hardware constraints.

As we will demonstrate, both networks achieve excellent accuracy despite their respective constraints, validating the robustness of our training approach.

1) Training the Complete Network: The complete network consists of five layers: an input layer, three hidden layers, and an output layer. The input layer comprises 28×28 registers, while the hidden layers contain 128, 96, and 96 neurons, respectively. The output layer consists of 10 neurons. For data processing, each 28×28 pixel training sample is flattened and presented to the network repeatedly at each time step (with a simulation duration of 25 time steps). During inference, input data is quantized to 1 bit to emulate the presence or absence of an SFQ pulse. The weights are limited to values of $+1$, -1 , and 0 by enforcing discrete coupling coefficients $K_i \in \{+1, -1, 0\}$ between each primary inductor L_i and its magnetically coupled secondary LB_i . As shown in Fig. 3., so that the induced current in every active secondary coil LB_i , which flows into the JJ, has identical magnitude, differing only in sign. Each neuron's fan-in is limited to 64, without specific constraints on the distribution between positive- and negative-weight connections. For output classification, we employ spike count decoding [31], [32], where network prediction is determined by counting the spikes emitted by each output neuron throughout the simulation window. The neuron with the highest spike count indicates the predicted class.

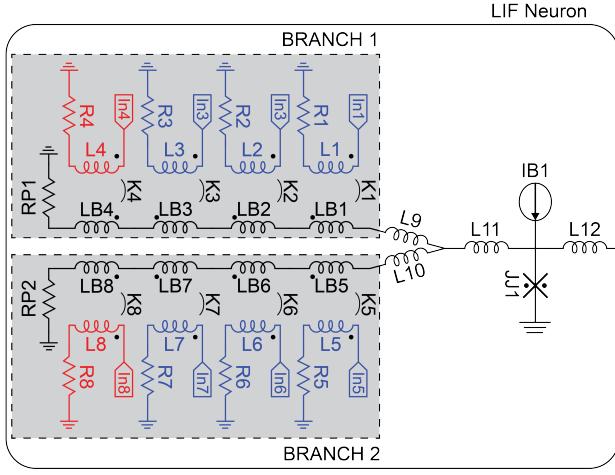
The entire training process for this network is divided into five stages. Each stage runs for a predetermined number of epochs, with the model yielding the highest accuracy on the full test dataset being preserved.

a) Unrestricted training: In this stage, the network is allowed to train freely using a membrane potential-based loss (mem loss) with the AdamW optimizer on 32-bit floating point data. The pre-binarized 28×28 input vector is fed directly into the network, and each vector is propagated over 25 time steps, yielding 25 prediction results whose output-layer membrane potentials are aggregated to compute the overall mem loss, which is then backpropagated to update the weights.

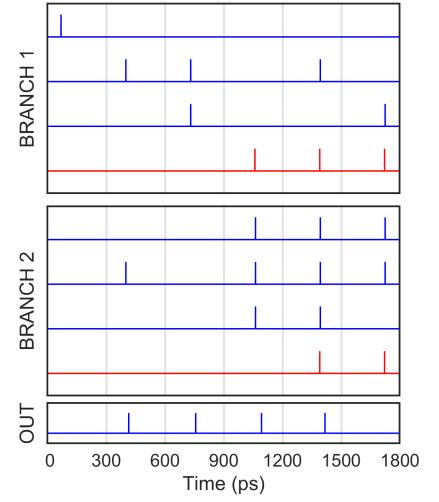
b) Input binarization training: Starting from the network obtained in stage (a), the inputs are binarized to 0 or 1 before being fed into the network.

c) Layer-by-layer quantization-aware training: In this stage, beginning with the network from stage (b), we sequentially apply quantization-aware training to each layer by constraining weights to the discrete values of $+1$, -1 , and 0 .

d) Layer-by-layer gradual pruning and quantization-aware training: In this stage, starting from the network of stage (c), we apply gradual pruning while maintaining quantization-aware training on all layers. Ideally, each neuron would retain only the 64 weights with the highest absolute values after each mini-batch, with remaining weights masked to zero in subsequent forward passes. However, directly pruning from 784 to 64 connections, particularly in the first layer, would cause severe underfitting. To mitigate this, we imple-



(a) Schematic of high fanin neuron ($L_1 = L_2 = L_3 = L_4 = L_5 = L_6 = L_7 = L_8 = 11.52 \text{ pH}$, $RP_1 = RP_2 = 0.55 \Omega$, $R_1 = R_2 = R_3 = R_5 = R_6 = R_7 = 0.56 \Omega$, $R_4 = R_8 = 0.40 \Omega$, $K_1 = K_2 = K_3 = K_5 = K_6 = K_7 = 0.6$, $K_4 = K_8 = -0.6$, $L_9 = L_{10} = 1.10 \text{ pH}$, $L_{11} = 0.93 \text{ pH}$, $I_{JJ1c} = 0.1 \text{ mA}$, $L_{12} = 0.5 \text{ pH}$, $ROUT = 4.32 \Omega$, $I_{JJ1c} = 0.15 \text{ mA}$, $L_{13} = 1.90 \text{ pH}$, $I_{JJ2c} = 0.17 \text{ mA}$)



(b) Simulation result of the network showing maximum operating at a clk frequency of 3.02 GHz with peripheral JJ pulse observations.

Fig. 3: High fan-in neuron design. Each neuron receives weighted inputs via two dendritic bundles—BRANCH 1 (RP_1 , synapses L_1-L_4) and BRANCH 2 (RP_2 , synapses L_5-L_8)—and fires an SFQ pulse once its combined current surpasses the JJ’s critical current; in the schematic, excitatory (positive-weight) inputs are shown in blue, while inhibitory (negative-weight) inputs are shown in red.

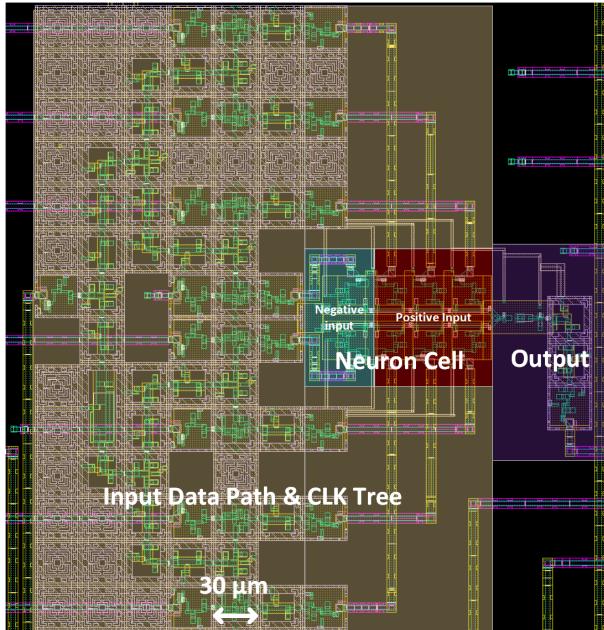


Fig. 4: NeuronLayout

ment a progressive pruning strategy according to Equation 10:

$$C_t = C_i - \frac{(C_i - C_f) \times (s + 1)}{S} \quad (10)$$

where C_t represents the target connections in the current step, C_i denotes the initial number of connections per neuron in the fully connected state, C_f is the final target number of

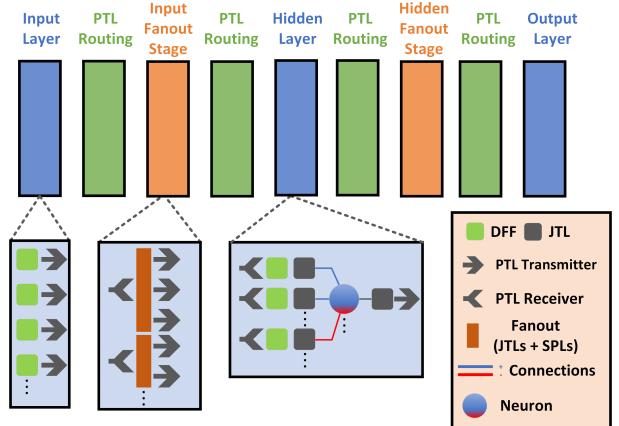


Fig. 5: Network Architecture. Inputs spread through a fanout stage into the hidden layer; their single-bit outputs fan-out again using PTL/JTL links, finally converging on the output layer, with DFFs coordinating timing throughout.

connections per neuron after pruning, s indicates the current pruning step (starting from 0), and S is the total number of pruning steps. For example, with $S = 60$ pruning steps for the first layer, the connections of each neuron gradually decrease from $C_i = 784$ to $C_f = 64$. This gradual approach preserves critical information while enabling the network to adapt to increased sparsity. Through network pruning, inactive neurons and their corresponding synaptic connections are eliminated, reducing computational cost and enhancing energy efficiency.

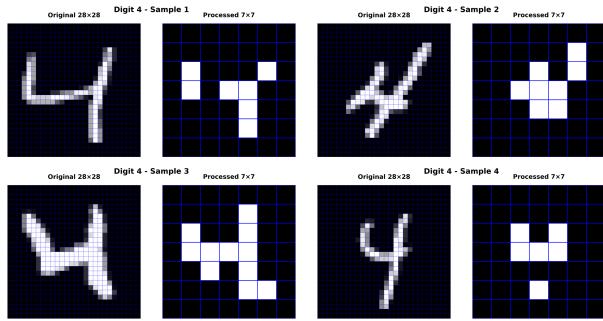


Fig. 6: Preprocessing for digit 4. Original 28×28 MNIST images are downsampled to 7×7 through block averaging with a threshold of 0.3, followed by binarization.

e) Validation: Finally, the model is evaluated on the entire test dataset.

2) Training the Chip Network: The chip network consists of three layers: an input layer, a hidden layer, and an output layer. The input layer consists of 49 DFFs. The hidden layers contain 24 neurons before pruning, respectively, while the output layer has three neurons. Original 28×28 images are downsampled to 7×7 dimension by block averaging with a threshold of 0.3 and binarization as a preprocessing step, as illustrated in Fig. 6. The weights will still be quantized to $+1$, -1 , and 0 . This time, our goal is to keep each neuron structurally uniform to utilize standard cells, thereby reducing the burden of layout and routing. After completing the quantization-aware training phase for this mini-network using our previous methodology, we observed that each neuron ultimately retained approximately six positive and two negative weights, a distribution extracted from our highest-accuracy model during training. Based on this observation, we designed each neuron to have six positive and two negative connections, creating a standardized architecture that optimizes computational efficiency and hardware implementation.

During both training and inference, we employ a single forward pass per input sample. At the end of this forward pass, a prediction is deemed correct only if exactly one neuron in the output layer, namely the one corresponding to the correct class, emits a single spike. All other output neurons must remain silent.

The training methodology for the chip network follows a similar approach with four key differences: First, for the loss function, we employ a combination of membrane potential-based loss (\mathcal{L}_{mem}) and spike-based loss ($\mathcal{L}_{\text{spike}}$) with the total loss ($\mathcal{L}_{\text{total}}$) given by

$$\mathcal{L}_{\text{total}} = 0.6 \mathcal{L}_{\text{spike}} + 0.4 \mathcal{L}_{\text{mem}} \quad (11)$$

Second, we introduce a weight clamping stage after input binarization training, where weights are restricted to $[-1, 1]$. Third, we changed the training sequence by pruning the network first, performing quantization-aware training for the chip network next, and pruning last. Fourth, during the gradual layer-by-layer pruning process, we specifically target a final

structure with a maximum of six positive weights and two negative weights per neuron.

4. EXPERIMENTAL RESULTS

4.1. Training Results

After quantization and pruning, the complete network achieves an accuracy of 96.47% on the complete MNIST dataset. Input data samples are binarized, with each neuron restricted to a maximum fan-in of 64 and weights limited to values of $+1$, -1 , and 0 . The time step parameter is set to 25. Table I presents the weight matrix statistics for the entire network, while Table II summarizes its architectural parameters.

TABLE I: Weight Matrix Statistics for the Complete Network

| Statistic | Hidden Layer 0 | Hidden Layer 1 | Hidden Layer 2 | Output Layer |
|----------------|----------------|----------------|----------------|--------------|
| Size | 100352 | 12288 | 9216 | 960 |
| Active Neurons | 128/128 | 96/96 | 96/96 | 10/10 |
| 1s | 3957 (3.94%) | 2935 (23.89%) | 2104 (22.83%) | 70 (7.29%) |
| -1s | 3697 (3.68%) | 2654 (21.60%) | 3410 (37.00%) | 273 (28.44%) |
| 0s | 92698 (92.37%) | 6699 (54.52%) | 3702 (40.17%) | 617 (64.27%) |

TABLE II: Complete Network Parameters

| Parameter (Complete Network) | Value |
|------------------------------|-------|
| Input pins | 784 |
| Output pins | 10 |
| Layers | 4 |
| Neurons | 330 |
| Max Fanout Hidden Layer0 | 128 |
| Max Fanout Hidden Layer1 | 96 |
| Max Fanout Hidden Layer2 | 96 |
| Max Fanout Hidden Layer3 | 10 |

For the chip network, after quantization and pruning, we achieve the precision 80.07% when classifying the digits 2, 3, and 4. The input data are downsampled from 28×28 to 7×7 and binarized, with each neuron having eight fan-in connections (two negative and six positive). The network operates with a single forward pass. Furthermore, we also evaluated various combinations of digits, with digits 0, 1, and 2 producing the highest precision of 86.2%. These results demonstrate that our Superconductor SNN Chip can effectively process multiple dataset configurations based on hardware configurations.

Finally, we selected digits 2, 3, and 4 for our chip design. The statistics of the weight matrix are presented in Table III, with the values in parentheses representing the percentage of each weight value.

TABLE III: Chip Network Weight Matrix Statistics

| Statistic | Hidden Layer | Output Layer |
|----------------|---------------|--------------|
| Total elements | 1176 | 72 |
| Active Neurons | 18/24 | 3/3 |
| Number of 1s | 76 (6.46%) | 18 (25.00%) |
| Number of -1s | 21 (1.79%) | 5 (6.94%) |
| Number of 0s | 1079 (91.75%) | 49 (68.06%) |

4.2. Chip Network Implementation

Our superconducting SNN chip has dimensions of 3.4×3.9 mm² with 40 pins and accommodates a maximum of 25 neurons. Pin allocation includes clock and bias current pins, leaving only seven pins for input data and three pins for output data. Based on hardware constraints, we trained the chip network to fit within the chip's capacity. The overall parameters for the chip network are shown in Table IV.

The hidden layer consists of 18 neurons, reduced from the original 24. This reduction results from two factors: first, the inherent sparsity of the downsampled MNIST dataset, where boundary pixels contain no information (zero values); second, the quantization process, which forces many weight elements to zero. These factors eliminate excitatory connections for certain neurons, rendering them inactive. To accommodate this, we implement a hierarchical structure on our chip. Input data are preprocessed and down-sampled to a 7×7 format before being fed into the chip. The input layer comprises 49 input shift registers. Due to the limited number of seven input pins, data is shifted in over seven cycles until the entire sample is loaded. Once all 49 shift registers contain the 1-bit data representing a complete sample, individual bias currents are applied to the output fanout circuits, which split and transmit the pulses through PTLs to the fan-in circuits of the hidden layer. After data synchronization, the 18-neuron hidden layer is activated, generating pulses that propagate through their respective fan-out circuits to the fan-in of the output layer (three neurons). The output layer then produces pulses representing the information used for classification. Due to negative clock routing constraints, two cycles are needed to propagate the inputs after loading a complete sample, followed by an additional cycle to retrieve the outputs. This process enables the chip to produce a prediction every ten cycles.

The superconducting SNN chip layout is shown in Fig. 5.

TABLE IV: Chip network parameters for 3 classes of the MNIST dataset.

| Parameter (Chip Network) | Value |
|--------------------------|-------|
| Input pins | 7 |
| Output pins | 3 |
| Layers | 3 |
| Neurons | 18+3 |
| Max Fanout Hidden Layer | 9 |
| Max Fanout Output Layer | 3 |

4.3. Power and Accuracy Results

We evaluated the performance of our superconducting SNN chip through comprehensive and detailed benchmarking. Table VI presents the accuracy results for various digit combinations. Variations in accuracy in different combinations can be attributed to inherent visual similarity between digits after downsampling and binarization. As shown in Figure 8, digits '3', '4', and '5' become visually indistinguishable following the 7×7 downsampling and binarization process, leading to a decrease in the classification accuracy. In contrast, digits '0', '1', and '2' retain more distinctive visual features even

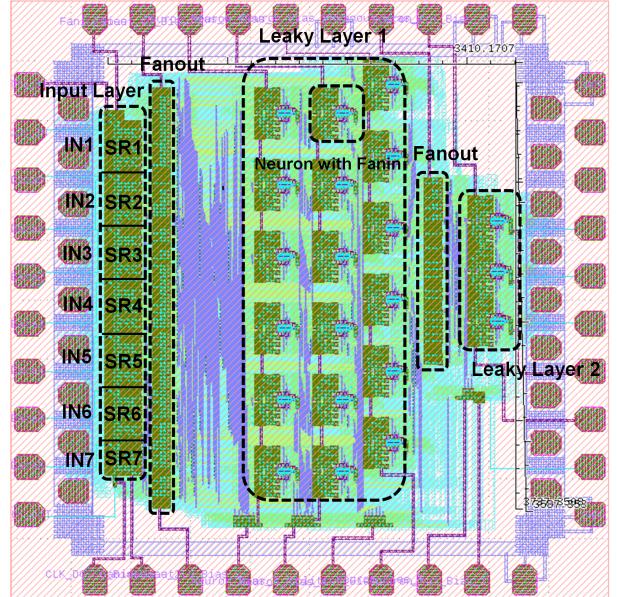


Fig. 7: Superconducting SNN Chip

after preprocessing, as shown in Figure 9, resulting in higher accuracy. The overall performance could be further enhanced by using higher resolution inputs (e.g. 14×14) or incorporating convolutional layers for more advanced data preprocessing.

TABLE V: Resource and power comparison for two of our recent works. The energy for inference is the dynamic energy of JJ switchings.

| Metric | karamuft et al [26] | This work |
|---------------------------|---------------------|-----------------------|
| Predictable digits | 0–9 | 2, 3, 4 |
| Accuracy (%) | 96.1 | 86.2 |
| JJs count | 196,972 | 5,822 |
| Static power (mW) | 44.6 | 2.15 |
| Energy per inference (nJ) | 1.5 | 1.31×10^{-6} |

As summarized in Table V, this work reports a fabricated superconducting SNN prototype that, due to stringent hardware constraints outlined in Section 4.2, focuses on the digit subset 2, 3, and 4, achieving an inference accuracy of 86.2%. In contrast, the network by Karamuftuglu et al. [26], which utilizes high-fan-in neurons and is evaluated in software, classifies all ten MNIST digits with 96.1% accuracy but requires a significantly larger design (containing 196,972 JJs) and has higher power consumption (44.6 mW static power and 1.5 nJ per inference). Our prototype incorporates 5,822 JJs, consumes 2.15 mW of static power, and has an energy cost of approximately 1.31×10^{-6} nJ per inference, highlighting the trade-offs involved when transitioning from simulation to a physically realizable superconducting SNN implementation.

5. CONCLUSION

This paper presented a novel design methodology for a superconducting SNN chip based on our innovative superconducting neuron cell. The proposed architecture leverages the event-driven operation of SNNs and the ultra-high-speed,

TABLE VI: Network accuracy for different digit combinations after downsampling and quantization of the inputs

| Digits | Accuracy (%) |
|---------|--------------|
| 0, 1, 2 | 86.20 |
| 2, 3, 4 | 80.07 |
| 3, 4, 5 | 72.34 |
| 5, 6, 7 | 75.07 |

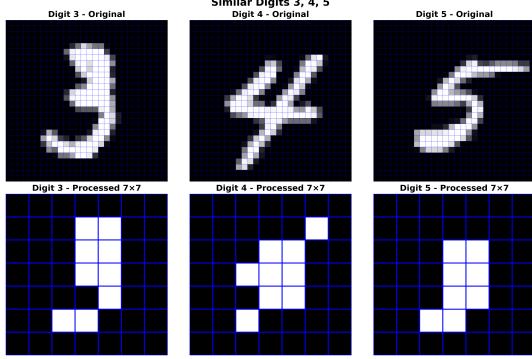


Fig. 8: Visual similarity of digits 3, 4, and 5 after 7×7 downsampling and binarization.

low-power capabilities of SFQ digital circuits to address the inefficiencies inherent in conventional digital systems. We developed a training framework using `snnTorch`, which incorporates a hybrid loss function that combines membrane potential and spike losses, allowing the network to achieve a classification accuracy of 96.5% on the entire MNIST dataset after quantification and pruning. To meet the strict hardware constraints of superconducting platforms, such as limited chip area and pin availability, we optimized the network to classify a reduced set of digits, achieving a maximum inference accuracy of 86.2% for digits 0, 1, and 2 within a chip area of $3.4 \times 3.9 \text{ mm}^2$. The total JJ count for our chip is 5,822, with a static power consumption of 2.15 mW, and an estimated switching energy of 6.55 fJ per inference. This work demonstrates the potential of superconducting SNNs for high-speed, energy-efficient neuromorphic computing and lays a promising foundation for future developments in scalable

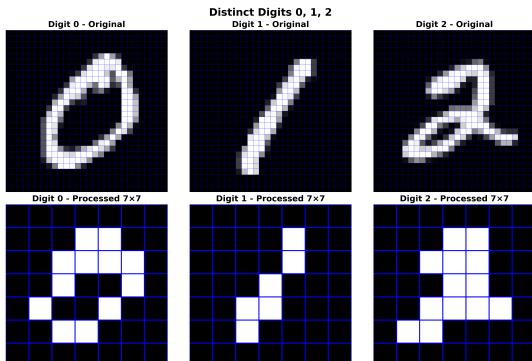


Fig. 9: Distinct visual features of digits 0, 1, and 2 after 7×7 downsampling and binarization.

neural-hardware systems.

REFERENCES

- [1] J. Han, Z. Li, W. Zheng, and Y. Zhang, “Hardware implementation of spiking neural networks on fpga,” *Tsinghua Science and Technology*, vol. 25, no. 4, pp. 479–486, 2020.
- [2] T. Fu, J. Zhang, R. Sun, Y. Huang, W. Xu, S. Yang, Z. Zhu, and H. Chen, “Optical neural networks: progress and challenges,” *Light: Science & Applications*, vol. 13, no. 1, p. 263, Sep. 2024. [Online]. Available: <https://www.nature.com/articles/s41377-024-01590-3>
- [3] F. Cai, S. Kumar, T. Vaerenbergh, X. Sheng, R. Liu, C. Li, Z. Liu, M. Foltin, S. Yu, Q. Xia, J. J. Yang, R. Beausoleil, W. Lu, and J. P. Strachan, “Power-efficient combinatorial optimization using intrinsic noise in memristor hopfield neural networks,” *Nature Electronics*, vol. 3, pp. 1–10, 07 2020.
- [4] A. Cresti, *Beyond-CMOS: State of the Art and Trends*, ser. ISTE Consignment. Wiley, 2023. [Online]. Available: <https://books.google.com/books?id=PpnMEAAAQBAJ>
- [5] T. Hirose, T. Asai, and Y. Amemiya, “Spiking neuron devices consisting of single-flux-quantum circuits,” *Physica C Superconductivity*, vol. 445–448, 10 2006.
- [6] K. Segall, M. LeGro, S. Kaplan, O. Svitelskiy, S. Khadka, P. Crotty, and D. Schult, “Synchronization dynamics on the picosecond time scale in coupled Josephson junction neurons,” *Phys. Rev. E*, vol. 95, p. 032220, Mar 2017. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevE.95.032220>
- [7] P. Crotty, D. Schult, and K. Segall, “Josephson junction simulation of neurons,” *Phys. Rev. E*, vol. 82, p. 011914, Jul 2010. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevE.82.011914>
- [8] J. M. Shainline, B. A. Primavera, and R. O’Loughlin, “Relating superconducting optoelectronic networks to classical neurodynamics,” 2024. [Online]. Available: <https://arxiv.org/abs/2409.18016>
- [9] S. E. Russek, C. A. Donnelly, M. L. Schneider, B. Baek, M. R. Pufall, W. H. Rippard, P. F. Hopkins, P. D. Dresselhaus, and S. P. Benz, “Stochastic single flux quantum neuromorphic computing using magnetically tunable josephson junctions,” in *2016 IEEE International Conference on Rebooting Computing (ICRC)*, 2016, pp. 1–5.
- [10] M. L. Schneider, C. A. Donnelly, S. E. Russek, B. Baek, M. R. Pufall, P. F. Hopkins, P. D. Dresselhaus, S. P. Benz, and W. H. Rippard, “Ultralow power artificial synapses using nanotextured magnetic josephson junctions,” *Science Advances*, vol. 4, no. 1, p. e1701329, 2018. [Online]. Available: <https://www.science.org/doi/abs/10.1126/sciadv.1701329>
- [11] E. Toomey, K. Segall, and K. K. Berggren, “Design of a power efficient artificial neuron using superconducting nanowires,” *Frontiers in neuroscience*, vol. 13, p. 933, 2019.
- [12] R. Cheng, U. S. Goteti, and M. C. Hamilton, “Spiking neuron circuits using superconducting quantum phase-slip junctions,” *Journal of Applied Physics*, vol. 124, no. 15, p. 152126, 10 2018. [Online]. Available: <https://doi.org/10.1063/1.5042421>
- [13] ———, “High-speed and low-power superconducting neuromorphic circuits based on quantum phase-slip junctions,” *IEEE Transactions on Applied Superconductivity*, vol. 31, no. 5, pp. 1–8, 2021.
- [14] M. A. Jardine and C. J. Fourie, “Hybrid rsfq-qfp superconducting neuron,” *IEEE Transactions on Applied Superconductivity*, vol. 33, no. 4, pp. 1–9, 2023.
- [15] A. J. Edwards, G. Krylov, J. S. Friedman, and E. G. Friedman, “Harnessing stochasticity for superconductive multi-layer spike-rate-coded neuromorphic networks,” *Neuromorphic Computing and Engineering*, vol. 4, no. 1, p. 014005, feb 2024. [Online]. Available: <https://dx.doi.org/10.1088/2634-4386/ad207a>
- [16] V. K. Semenov, E. B. Golden, and S. K. Tolpygo, “Biosfq circuit family for neuromorphic computing: Bridging digital and analog domains of superconductor technologies,” *IEEE Transactions on Applied Superconductivity*, vol. 33, no. 5, pp. 1–8, 2023.
- [17] E. B. Golden, V. K. Semenov, and S. K. Tolpygo, “Development of a neuromorphic network using biosfq circuits,” *IEEE Transactions on Applied Superconductivity*, vol. 35, no. 5, pp. 1–7, 2025.
- [18] K. Segall, C. Purmessur, A. D’Addario, and D. Schult, “A superconducting synapse exhibiting spike-timing dependent plasticity,” *Applied Physics Letters*, vol. 122, no. 24, p. 242601, 06 2023. [Online]. Available: <https://doi.org/10.1063/5.0150687>

- [19] K. Segall, L. Nichols, W. Friend, and S. B. Kaplan, "Learning dynamics on the picosecond timescale in a superconducting synapse structure," 2025. [Online]. Available: <https://arxiv.org/abs/2504.02754>
- [20] B. Z. Ucpinar, M. A. Karamuftuoglu, S. Razmkhah, and M. Pedram, "An on-chip trainable neuron circuit for sfq-based spiking neural networks," *IEEE Transactions on Applied Superconductivity*, vol. 34, no. 3, pp. 1–6, 2024.
- [21] M. A. Karamuftuoglu, B. Z. Ucpinar, S. Razmkhah, M. Kamal, and M. Pedram, "Unsupervised sfq-based spiking neural network," *IEEE Transactions on Applied Superconductivity*, vol. 34, no. 3, pp. 1–8, 2024.
- [22] Z. Han, Z. Li, N. Yoshikawa, and Y. Yamanashi, "High area efficiency binary neural processing elements with time-domain signals using sfq circuits," *Superconductor Science and Technology*, vol. 37, no. 4, p. 045014, mar 2024. [Online]. Available: <https://dx.doi.org/10.1088/1361-6668/ad2fd9>
- [23] Z. Li, H. Shen, N. Yoshikawa, and Y. Yamanashi, "A binary neural computing unit with programmable gate using sfq and cmos hybrid circuit," *Superconductor Science and Technology*, vol. 37, no. 6, p. 065012, may 2024. [Online]. Available: <https://dx.doi.org/10.1088/1361-6668/ad44e2>
- [24] Z. Liu, S. Chen, P. Qu, H. Liu, M. Niu, L. Ying, J. Ren, G. Tang, and H. You, "Sushi: Ultra-high-speed and ultra-low-power neuromorphic chip using superconducting single-flux-quantum circuits," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 614–627. [Online]. Available: <https://doi.org/10.1145/3613424.3623787>
- [25] A. Bozbey, M. A. Karamuftuoglu, S. Razmkhah, and M. Ozbayoglu, "Single flux quantum based ultrahigh speed spiking neuromorphic processor architecture," 2020. [Online]. Available: <https://arxiv.org/abs/1812.10354>
- [26] M. A. Karamuftuoglu, B. Z. Ucpinar, A. Fayyazi, S. Razmkhah, M. Kamal, and M. Pedram, "Scalable superconductor neuron with ternary synaptic connections for ultra-fast snn hardware," *Superconductor Science and Technology*, vol. 38, no. 2, p. 025014, jan 2025. [Online]. Available: <https://dx.doi.org/10.1088/1361-6668/adaaa9>
- [27] S. Tolpygo, V. Bolkhovsky, T. Weir, A. Wynn, D. Oates, L. Johnson, and M. Gouker, "Advanced fabrication processes for superconducting very large scale integrated circuits," *IEEE Transactions on Applied Superconductivity*, p. 1–1, 2016. [Online]. Available: <http://dx.doi.org/10.1109/TASC.2016.2519388>
- [28] R. Cipolla, Y. Gal, and A. Kendall, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7482–7491.
- [29] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," 2019. [Online]. Available: <https://arxiv.org/abs/1810.04650>
- [30] K. Likharev and V. Semenov, "Rsfq logic/memory family: a new josephson-junction technology for sub-terahertz-clock-frequency digital systems," *IEEE Transactions on Applied Superconductivity*, vol. 1, no. 1, pp. 3–28, 1991.
- [31] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in Computational Neuroscience*, vol. 9, 2015. [Online]. Available: <https://www.frontiersin.org/journals/computational-neuroscience/articles/10.3389/fncom.2015.00099>
- [32] M. A. Karamuftuoglu, B. Z. Ucpinar, S. Razmkhah, M. Kamal, and M. Pedram, "Unsupervised sfq-based spiking neural network," *IEEE Transactions on Applied Superconductivity*, vol. 34, no. 3, pp. 1–8, 2024.