SpikeTouch: Optimizing Spike Neural Networks for Tactile Perception

XUERONG ZHAO*, Northwest University, China
XUAN WANG*, Northwest University, Shaanxi Key Laboratory of Passive Internet of Things and Neural Computing, China
JIAN WU, Northwest University, Internet of Things Research Center, Northwest University, China
CHAO FENG†, Northwest University, Xi'an Advanced Battery-Free Sensing and Computing Technology International Science and Technology Cooperation Base, China
DINGYI FANG, Northwest University, Shaanxi International Joint Research Centre for the Battery-Free Internet of Things, China
XIAOJIANG CHEN, Northwest University, Shaanxi Key Laboratory of Passive Internet of Things and Neural Computing, China
ZHENG WANG, Leeds University, China

Tactile perception enables systems to sense and interpret physical properties such as shape, material, pressure, and texture. It is a key capability for emerging applications like robotic surgery and assistive robotics. Existing solutions to tactile perception typically rely on computation-intensive deep neural networks, which require high-performance computing resources unavailable on embedded and battery-powered devices. Spiking neural networks (SNNs) offer a promising, energy-efficient alternative, but their practical adoption remains limited due to the lack of efficient and deployable neuromorphic solutions. We present SpikeTouch, a software framework designed to reduce the computational overhead of SNNs for tactile perception on neuromorphic hardware. SpikeTouch offers three key optimizations tailed to SNN-based tactile perception: (1) a spike encoding scheme that balances precision and computational cost; (2) a systematic method for extracting multi-dimensional tactile features; and (3) a training strategy that minimizes quantization errors to improve performance and reduce memory usage. We evaluate SpikeTouch on the Tianjic neuromorphic chip using representative tactile perception workloads. Our results show that SpikeTouch achieves high recognition accuracy for 30 objects and their material stiffness , with an average accuracy of 92.92% and 92.35%, respectively. It is also highly energy and computationally efficient, operating at just 1 Watt of power - significantly lower than the hundreds or thousands of Watts typically required by a GPU - and delivering a response in under 20 ms - well below the average human reflex time of over 100 ms.

CCS Concepts: • **Human-centered computing → Ubiquitous and mobile computing**.

*Both authors contributed equally to this research.
†Corresponding author.

Authors' Contact Information: Xuerong Zhao, Northwest University, China, xrzhao@stumail.nwu.edu.cn; Xuan Wang, Northwest University, Shaanxi Key Laboratory of Passive Internet of Things and Neural Computing, China, xwang@nwu.edu.cn; Jian Wu, Northwest University, Internet of Things Research Center, Northwest University, China, 2022117131@stumail.nwu.edu.cn; Chao Feng, Northwest University, Xi'an Advanced Battery-Free Sensing and Computing Technology International Science and Technology Cooperation Base, China, chaofeng@nwu.edu.cn; Dingyi Fang, Northwest University, Shaanxi International Joint Research Centre for the Battery-Free Internet of Things, China, dyf@nwu.edu.cn; Xiaojiang Chen, Northwest University, Shaanxi Key Laboratory of Passive Internet of Things and Neural Computing, China, xjchen@nwu.edu.cn; Zheng Wang, Leeds University, China, z.wang5@leeds.ac.uk.

ACM 2474-9567/2025/12-ART242
https://doi.org/10.1145/3770658

Additional Key Words and Phrases: Tactile Perception, Spike Neural Networks, Low-power Neuromorphic Chip

ACM Reference Format:
Xuerong Zhao, Xuan Wang, Jian Wu, Chao Feng, Dingyi Fang, Xiaojiang Chen, and Zheng Wang. 2025. SpikeTouch: Optimizing Spike Neural Networks for Tactile Perception. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 9, 4, Article 242 (December 2025), 24 pages. https://doi.org/10.1145/3770658

1 Introduction

Tactile perception is the ability to sense and understand touch-related properties like texture, pressure, and movement [27]. It enables machines to detect shape, stiffness, or slipping during contact, helping them handle objects with care [29, 44], similar to how humans use their hands. Such a capability is useful in areas like surgery or assistive robots and is essential to modern robotics systems. For example, Tesla's Optimus Gen 2 robot hand uses 11 high-resolution tactile sensors [1] to handle fragile items like eggs, while the Da Vinci surgical robot uses force sensors to estimate tissue stiffness and avoid damaging it during surgery [22].

Today's tactile perception systems typically rely on deep neural networks (DNNs) to process tactile data [18, 28, 32, 42], but these models are too resource-intensive for embedded and battery-powered devices with limited computing power. Running DNNs on such systems can cause delays and reduce control accuracy, which is risky in tasks like handling fragile objects. Spiking neural networks (SNNs) offer a promising, energy-efficient alternative to traditional neural networks, inspired by how the brain processes information using short bursts of activity, or "*spikes*" [30, 38, 53]. As depicted in Fig. 1, an SNN-based tactile perception system uses neuromorphic sensors that mimic how mammals sense touch by producing spikes in response to physical contact. These spikes are then processed by spiking neurons. Because SNNs are designed to work with sparse, event-driven data and use little power, they are well-suited for real-time tactile sensing in devices with limited resources, such as robotic hands and wearable systems.

While promising, most existing SNN-based tactile perception solutions are still implemented and evaluated on conventional CPUs or GPUs [19, 37]. These approaches often assume access to powerful computing platforms - an assumption that does not hold for resource-constrained devices, where computational resources and capacity are limited. Moreover, since conventional computing architectures are not designed to exploit the spike sparsity of SNNs, prior work suffers from poor energy efficiency, which undermines the key advantages of using SNNs [52, 54]. Without concrete, quantitative evidence of improved efficiency on dedicated neuromorphic hardware, it remains difficult to justify the adoption of SNN-based tactile perception in practical applications.

This paper introduces SpikeTouch, a software framework for optimizing SNN-based tactile perception workloads. SpikeTouch integrates a set of strategies to optimize SNNs for tactile perception tasks during both model design and deployment. It addresses key limitations in existing work by tackling three aspects of energy and performance optimization for tactile perception workloads: (1) efficient spike encoding, (2) extraction of multi-dimensional tactile features, and (3) mitigation of quantization errors when trading SNN model weights for improved performance and reduced memory footprint. Furthermore, our work also contributes to the adoption of SNN-based tactile perception and further research by providing empirical evidence of tangible performance improvements on SNN-enabled neuromorphic hardware. In the remainder of this section, we describe each optimization strategy in detail, highlighting the challenges each optimization addresses and its contributions toward optimizing the performance and energy efficiency of SNN-based tactile perception applications.

Firstly, accurately encoding continuous tactile signals into spike-based representations is challenging, as it requires balancing computational efficiency against signal fidelity. High-accuracy perception requires enough spikes to capture subtle signal variations, but this increases computational overhead. Reducing the spike count improves computational efficiency but risks losing important details. Existing methods - based on fixed thresholds [40] or uniform sampling [4] are inadequate. They either generate too many spikes, increasing energy use, or

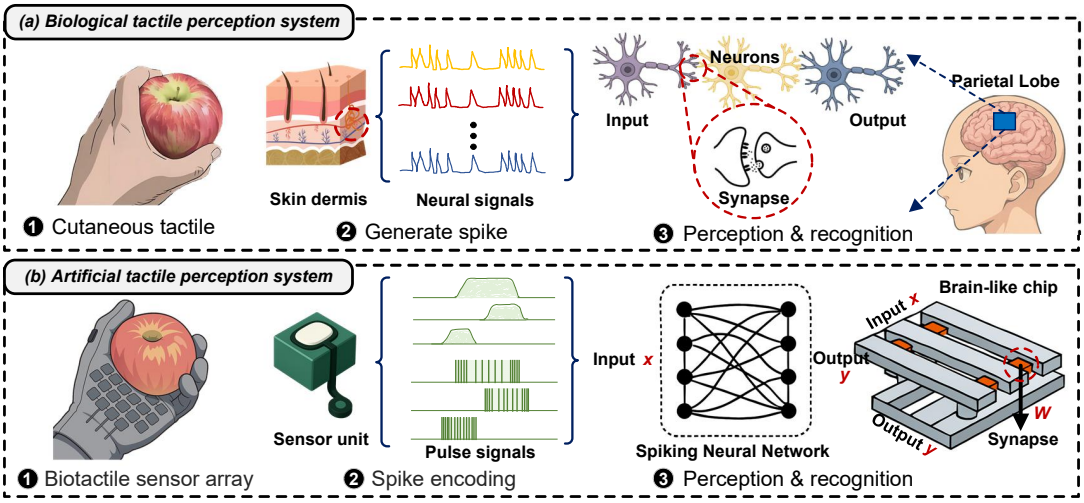


Fig. 1. Biological tactile perception systems (a) and SNN-based solutions runing on brain-like neuromorphic hardware (b)

too few, degrading precision. We address this with *SparsePulse*, a reverse-generation encoding scheme that applies adaptive thresholding to segment pressure signals and encode them using spike counts during active periods. In contrast to conventional approaches that the number of generated spikes proportionally to pressure—leading to dense spiking during stable holding phase due to sustained high pressure—our key insight is to generate dense spikes in high-entropy regions characterized by rapid changes but small pressure, while creating sparse spikes in low-entropy, stable regions with large pressure. This strategy improves efficiency while preserving perceptual accuracy.

Secondly, effective tactile perception requires extracting key features from complex, multi-dimensional signals [55]. Tactile data encodes information across time and space [34]. Instantaneous, *temporal* pressure values reveal properties like shape and material - whether soft or hard, smooth or textured - while *spatial* pressure patterns indicate contact geometry. For example, cylindrical objects like pens or bottles often yield symmetrical pressure distributions [25], whereas angular objects such as cubes or wrenches create localized, uneven zones [33]. Material stiffness also adds complexity: a rubber ball and a metal ball of the same shape generate different pressure profiles due to differences in deformation. To handle this, we propose *MT-SNN*, a lightweight, multi-task SNN architecture designed to jointly recognize object categories and infer material stiffness from spatiotemporal tactile data. Our strategy reduces the need for separate specialized models, thereby simplifying model complexity and improving inference latency using a single optimized model by avoiding model switching.

Thirdly, quantization is widely used to reduce the computational and memory overhead of trained models during deployment [8, 41, 58]. By representing model weights with low-bit values, it lowers resource requirements but also introduces precision loss - an issue especially problematic for SNNs, which rely on precise spike timing [23]. For SNNs, even small quantization errors can disrupt membrane potential dynamics, causing neurons to misfire and degrading overall performance [26]. For example, a neuron that should fire after reaching a specific input threshold might fire too early if quantized weights distort the input sum. To mitigate this, we propose *SpikeQAT*, a quantization-aware training framework that improves inference accuracy on SNN-enabled hardware. *SpikeQAT* integrates quantization and dequantization steps directly into the training phase, allowing the network to learn to compensate for quantization-induced errors via backpropagation. This is achieved by integrating quantization

and dequantization modules into the synaptic weight paths and membrane potential updates of Leaky Integrate-and-Fire (LIF) layers - the key component of SNNs, enhancing robustness and preserving performance after deployment.

We have developed a working prototype of SpikeTouch and evaluated its optimization methods on a pressure sensing array and the Tianjic brain-like neuromorphic chip on representative tactile perception tasks. Our evaluation shows that SpikeTouch can effectively recognize 30 objects and their material stiffness with an average accuracy of 92.92% and 92.35%, while consuming only 1 W of power with a response time of under 20 ms[1]. In comparison with conventional neural networks running on a NVIDIA 2080Ti GPU, SpikeTouch reduces average power consumption by over 96% and inference time by 80%.

This paper makes the following contributions:

- A new spike encoding scheme that balances precision and computational overhead, leveraging the characteristics of tactile data for energy-efficient optimization (Section 4.3);
- A systematic method for extracting multi-dimensional tactile features (Section 4.4);
- A training approach that reduces quantization errors when optimizing spiking neural networks (SNNs) for improved performance and reduced memory usage (Section 4.5);
- Empirical evidence demonstrating the benefits of optimized SNNs for tactile perception on neuromorphic hardware (Section 6).

2 Related Work

In this section, we will discuss the related studies in tactile perception and SNN.

Physics-based Models. Physics-based haptic perception systems rely on high-precision sensor signals and well-defined physical models of tactile sensation to decode haptic signals [10, 50]. For instance, J. Dargahi and S. Najarian [10] developed a neural encoding model by simulating the spatiotemporal responses of mechanoreceptors in the skin, employing frequency domain decomposition to differentiate haptic modalities. Chelsea Tymms et al. [50]employed finite element analysis (FEM) to compute the skin strain field under rigid contact. However, these methods typically rely on simplifying assumptions, making it difficult to accurately represent complex, real-world haptic interactions. This leads to poor generalization when applied to new objects or environments. Additionally, physics-based modeling is inflexible when dealing with dynamic tasks, and its real-time performance is often inadequate for fast-response requirements.

Machine Learning-based Systems. Machine learning-based tactile systems [5, 7, 36, 45], such as those developed by Barreiros et al. [5], combine Support Vector Machines (SVM) with Principal Component Analysis (PCA) to reduce the 548-channel tactile data to 8 dimensions, enabling multitask force and contact point localization. Sundaram et al. [45] utilized ResNet-18 to classify 26 types of objects, achieving an accuracy rate of 98%. Chen et al. [7] employed a Multihead-Attention Residual Network (Multihead-Attention ResNet) combined with a Bidirectional Gated Recurrent Unit (Bi-GRU) to analyze passive vibrotactile signals collected by a smartwatch, achieving classification of 20 categories of daily objects with an average accuracy of 86.4%. However, complex models used in haptic perception, such as CNN-LSTM or Transformer networks, often suffer from high computational latency due to their large parameter sizes and complex architectures. This makes them unsuitable for millisecond-level response applications, such as tissue palpation in surgical robots, force control in prosthetics, or fragile object manipulation in industrial robotics. Moreover, these models face significant compatibility issues with low-power hardware, such as embedded systems.

Spiking Neural Networks (SNN) based Methods. SNNs, inspired by brain-like computing principles, are well-suited for processing spatiotemporal tactile signals due to their biologically plausible spike-based coding

[1]For comparison, the human body's reflex response typically takes 150–200 ms - nearly an order of magnitude slower than 20 ms - suggesting SpikeTouch's potential for supporting real-time applications.

mechanisms [6, 9, 16, 17, 24, 47, 57]. This enables low-power operation at the milliwatt level while maintaining rapid response times and high-precision recognition, making them suitable for applications such as medical robots and intelligent prosthetics. For example, the pioneering works [9, 24] primarily focus on the design of human-like tactile sensors that generate neuron-like signal patterns (i.e., spikes), which are then fed into an artificial neural network (ANN) for recognition. [57] integrates pressure sensors with NbOx-based memristors to fuse multimodal analog information into a single spike sequence, enabling simultaneous pressure and temperature sensing through SNN model. Taunyazov et al. [47] encoded BioTac signals into spike sequences using Izhikevich neurons, achieving a texture classification accuracy of 94.6% by using SNN. While, the work [21] explores more advanced learning rules for spiking neural networks (SNNs), such as spike-timing-dependent plasticity (STDP), to enable unsupervised feature extraction. Although SNN-based methods show great potential for tactile perception, many are evaluated only in simulated environments rather than on real neuromorphic hardware. The energy efficiency and temporal advantages of SNNs remain underutilized, as conventional computing platforms—such as ARM-based GPUs—are not optimized for the event-driven and sparse computation that SNNs rely on.

Unlike the aforementioned works, our work presents a complete, low-cost, and efficient end-to-end neuromorphic tactile sensing system deployed on real SNN hardware. We introduce a spike encoding scheme that balances precision and efficiency, and a training strategy that mitigates quantization errors to improve performance and reduce memory usage. SpikeTouch further supports dual-task tactile perception, jointly identifying object category and physical properties (e.g., hardness) from a single static grasp.

3 Preliminary

This section introduces the bankgrounds of Spiking Neural Networks (SNNs) and the basics of neuromorphic hardware.

3.1 Spiking Neural Networks

SNNs mimic the spike-based communication patterns of biological nervous systems [12, 14, 56]. They typically consist of spiking neurons, connections known as synapses, and dynamic rules governing spike transmission. Unlike traditional neural networks, SNNs process and encode information through discrete sequences of spikes (action potentials), considering both timing and spatial positioning.

A key component of SNNs is the Leaky Integrate-and-Fire (LIF) neuron [2], as illustrated in Fig. 2. When a neuron receives input spikes, its internal membrane potential increases according to a defined integration process. At the same time, a continuous leakage effect gradually decreases the membrane potential at a constant rate, preventing it from growing without limits and maintaining stable functionality. When the membrane potential reaches a certain threshold, the neuron fires (emits a spike) and then resets its potential.

Compared to conventional artificial neural networks (ANNs) [31], SNNs provide several important benefits. First, their event-driven spike communication significantly lowers energy use, making SNNs particularly suitable for energy-efficient applications like edge computing. Second, their natural use of spike timing allows them to process temporal information effectively, avoiding the need for the complicated recurrent structures often used in ANNs. Third, the dynamic nature of SNNs and their sparse spike signals enhance their robustness against noise. Given these advantages, our work utilizes SNNs as the primary learning framework for tactile perception, aiming to achieve greater energy efficiency and reduced computational demands on embedded systems.

3.2 Neuromorphic Computing Platforms

Most existing approaches for executing SNNs rely on conventional CPU or GPU hardware. However, these platforms are not designed to take full advantage of key SNN characteristics, such as sparsity, event-driven computation, and time-based information encoding. To address these limitations, brain-inspired computing

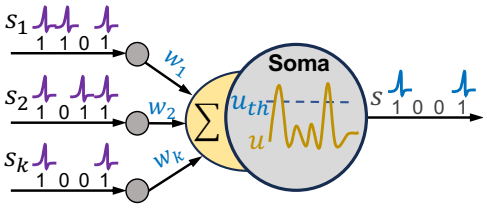


Fig. 2. Workflow of a LIF neuron.

platforms, such as Intel Loihi [11], IBM TrueNorth [35], and Tsinghua Tianjic [39], have been developed. These neuromorphic platforms mimic the structure and function of biological neural systems. By integrating computation and memory, they depart from the traditional von Neumann architecture and offer advantages such as high parallelism, low energy consumption, and distributed processing.

In this work, we evaluate our approach on the Tianjic neuromorphic chip, which supports diverse neural models, parallel task execution, dynamic learning capabilities, and flexible power control. These features make it particularly well-suited for energy-efficient and real-time tactile sensing applications. However, our software-based optimizations are equally applicable to other neuromorphic hardware architectures.

Tianjic Architecture: The Tianjic multi-core processor is designed to support the computational requirements of ANNs and SNNs [39]. Each functional core comprises five key modules: the Axon module caches spikes and activation values, the Synaptic module stores synaptic weights, the Dendritic module serves as the main computational unit, the Soma module performs nonlinear transformations, and the Routing module manages data transmission. To achieve low power consumption and reduced latency, the Tianjic chip adopts a modular hardware design with several key optimizations. These include near-memory computing to minimize data movement and energy use, input data sharing to reduce memory access frequency, row-based computation skipping to avoid unnecessary operations on zero-value inputs, column grouping enablement to power down inactive column groups, and inter-group pipeline parallelism to enhance processing efficiency. Together, these architectural innovations enable the Tianjic chip to deliver high computational throughput with energy-efficient and scalable support for a broad range of AI workloads.

Network Mapping: To run a spiking or neural network on the underlying hardware, high-level neural operations - such as convolution and activation - must first be translated into low-level hardware primitives, like the *Axon* and *ReLU* primitives on the Tianjic chip. This translation process involves generating configuration files through simulation and deploying them onto the chip. The mapping process is typically carried out in two stages. First, in the *logical mapping* stage, the neural network is broken down and assigned across functional cores to ensure functional correctness and task completion, without yet considering hardware limitations. Second, in the *physical mapping* stage, these logical cores are placed onto specific physical locations within the chip, with the goal of minimizing communication overhead by accounting for the chip's fixed two-dimensional core array. Efficient mapping is essential for making the most of available hardware resources and achieving high performance when running SNNs on the Tianjic platform.

4 System Design

4.1 Overview of SpikeTouch

SpikeTouch is a bio-inspired, low-power tactile sensing architecture that encodes continuous pressure signals into discrete spike sequences on a neuromorphic chip. By leveraging the event-driven and sparse computation characteristics of SNNs, it eliminates redundant processing commonly found in traditional approaches, thereby achieving both low power consumption and low latency. As illustrated in Fig. 3, SpikeTouch comprises four

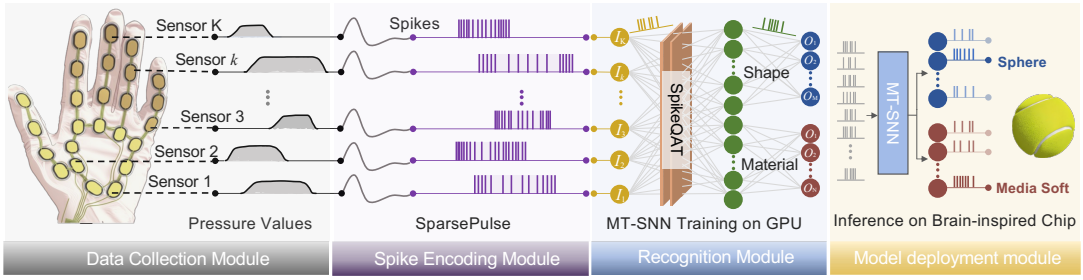


Fig. 3. Overview of the main SpikeTouch components for tactile perception.

primary modules: a tactile data acquisition module, a spike encoding module, a multi-task recognition module, and a model deployment module. The design and functionality of each module are described in detail below.

Tactile Data Acquisition and Processing Module (Sec. 4.2). To enable precise measurement of the mechanical characteristics during grasping, a flexible glove equipped with 25 pressure sensors is used. These sensors are strategically distributed across key contact regions—fingertips, finger pads, and palm—to ensure comprehensive coverage of pressure variations during hand-object interactions. The sensor array accurately captures pressure signals reflecting multiple dimensions such as object shape, stiffness, and gripping force, and transmits the ADC-converted analogue signals in real time to the processing unit.

Spike Encoding Module (Sec. 4.3). This module converts continuously varying pressure signals into discrete spike sequences for subsequent SNN-based processing. To achieve this, a reverse-generation encoding scheme, *SparsePulse*, is proposed, which employs adaptive thresholding to segment the pressure signals and encodes them using spike counts during active periods. By generating dense spikes in high-entropy regions—characterized by rapid changes but low pressure—and sparse spikes in low-entropy, stable regions with high pressure, the scheme enhances encoding efficiency while reducing power consumption.

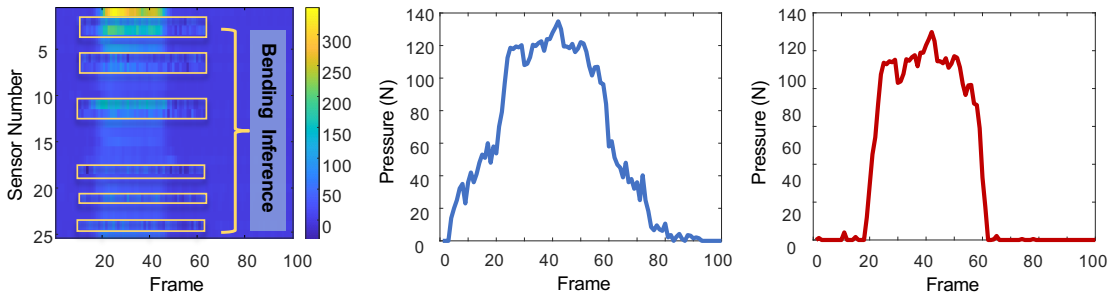
Recognition Module (Sec. 4.4 and Sec. 4.5). A collaborative multi-task SNN architecture is designed to efficiently extract spatiotemporal features from spike sequences using the LIF model, enabling simultaneous recognition of object types and their material stiffness. While training is conducted on a GPU, deployment targets neuromorphic hardware with constrained computational and memory resources. To bridge the performance gap caused by quantization, we introduce a quantization-aware training framework, SpikeQAT, which embeds quantization and dequantization operations directly into the training process. This approach enables the network to learn to compensate for quantization-induced errors through backpropagation, thereby improving robustness and preserving performance.

Model Deployment and Inference Module. The trained model is deployed into the neuromorphic chip for inference through software-based mapping. A spatiotemporal spike matrix, generated by encoding the pressure values from a set of tactile sensors, is fed into the trained MT-SNN model. After weighted computations, the output neuron with the highest spike count determines the predicted class label.

4.2 Data Preprocessing

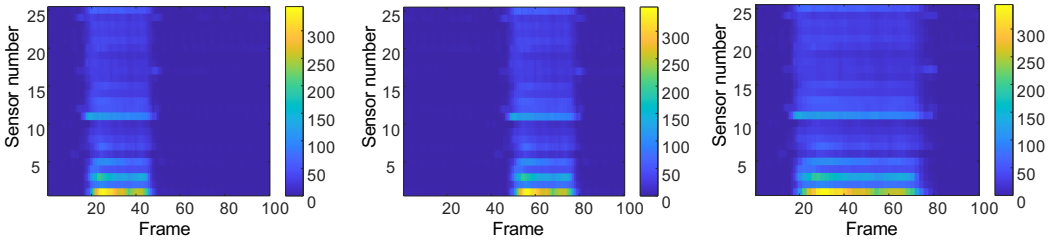
In this section, we first present a signal calibration scheme to mitigate signal distortion caused by sensor bending during data acquisition. Next, we introduce a data augmentation strategy designed to generate a large volume of diverse samples, thereby enhancing data diversity and improving model generalization.

4.2.1 Signal Calibration. During data collection, we observe that minor hand movements can cause bending of the glove's sensors, resulting in pressure artifacts. As illustrated in Fig. 4(a), sensors located at frequently bent



(a) Sensor array responses with bending interference. (b) Single-sensor pressure curve with interference. (c) Single-sensor pressure curve after interference elimination.

Fig. 4. Illustration of eliminating sensor bending interference.



(a) Initial data. (b) Translation operation. (c) Stretching operation.

Fig. 5. Temporal dimension data Augmentation.

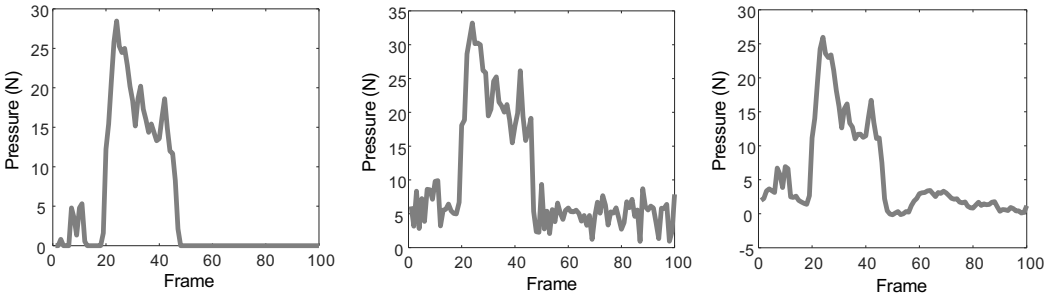
positions (e.g., sensors 2, 3, 6, etc.) show significant distortion, while others remain largely unaffected. To mitigate this issue, we build a bending error map by gradually bending the glove and recording outputs across angles. A sliding window of six data points is used to compute a composite score based on a weighted combination of mean and variance similarity, enabling the identification of regions dominated by bending-induced errors.

$$Score = w_1 \cdot \text{Mean} + w_2 \cdot \text{Var} \tag{1}$$

where $w_1 = 0.7$ and $w_2 = 0.3$. If the similarity exceeds 80%, the signal is corrected by subtracting the corresponding mapped error value derived from the bending error map. If the measured pressure exceeds 1.5 times the maximum recorded bending-induced value, it is considered real contact and corrected accordingly.

Fig. 4(b) and Fig. 4(c) illustrate the effectiveness of the denoising algorithm for a single tactile sensor. The raw pressure curve (Fig. 4(b)) exhibits significant fluctuations caused by glove deformation, especially during hand movements, where error signals are entangled with true pressure signals. In contrast, the denoised curve (Fig. 4(c)), obtained through a sliding-window matching strategy combined with a curvature-error mapping table, appears considerably smoother. The correction effectively suppresses error components and highlights the underlying true pressure signals.

4.2.2 Data Augmentation. The collected pressure data are limited in quantity and captured under relatively constrained scenarios, making it difficult to comprehensively reflect the glove's behavior across diverse usage conditions. In particular, sensor signals exhibit high nonlinearity and complexity under varying users, manipulation styles, and contact states. When employing spiking neural networks to model pressure perception, the quality and diversity of training data play a critical role in determining model performance. To address this,



(a) Initial data. (b) Time-domain noise injection. (c) Frequency-domain noise injection.

Fig. 6. Feature Perturbation Augmentation.

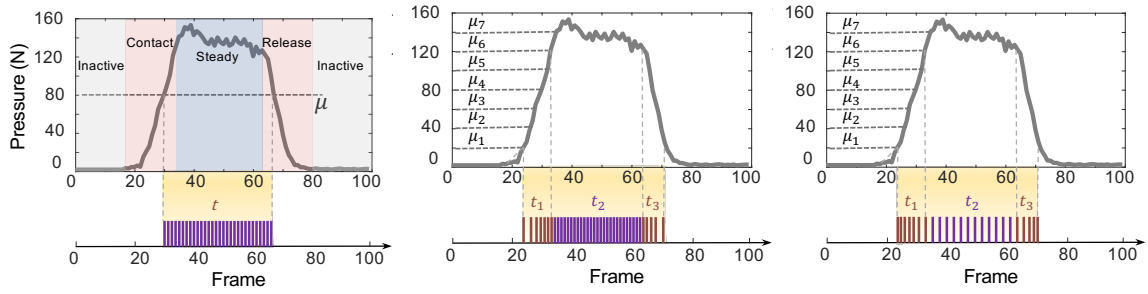
we design two types of data augmentation strategies aimed at enriching signal diversity and enhancing model generalization.

Time-Domain Augmentation: 1) *Time Shift:* Randomly offsetting the start point of the pressure signal by a value Δt to simulate variations in contact timing across different grasping instances. 2) *Time Scaling:* Applying linear interpolation to compress or stretch the time series by a factor α, enabling the model to adapt to variations in interaction speed. Fig.5 illustrates the original pressure signal (Fig.5(a)) along with its augmented versions. As shown, time shifting (Fig.5(b)) alters the onset of pressure events, while time scaling (Fig.5(c)) changes the temporal dynamics, effectively emulating different grasping speeds and force application rates.

Feature Perturbation Augmentation: Feature perturbation-based augmentation significantly enriches data diversity by introducing controlled variations into feature representations. We adopt two noise injection techniques: 1) *Time-domain Gaussian Noise Injection:* For each data point $x_{i,j}$,For each pressure data point $x_{i,j}$, Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ (e.g., $\sigma = 0.1$) is added to simulate realistic sensor fluctuations and environmental noise; 2) *Frequency-domain Modulation:* We apply the Discrete Fourier Transform (DFT) to convert signals into the frequency domain, then selectively amplify low-frequency components by scaling their magnitudes while preserving phase information. The modulated signal is then reconstructed using the Inverse DFT, yielding time-domain signals with altered spectral characteristics. Fig. 6 illustrates the effects of these augmentation methods, demonstrating how they expand the diversity of the training dataset. This two-dimensional augmentation strategy effectively addresses sample imbalance, mitigates the risk of overfitting to specific data distributions, and enhances the model's robustness under noisy conditions.

4.3 Design of Spike Encoding

In this section, we focus on converting continuously varying pressure signals into discrete spike sequences suitable for subsequent SNN-based processing. As illustrated in Fig. 7(a), a typical tactile signal during grasping exhibits several distinct temporal phases: an initial near-zero segment corresponding to the inactive state, a sharp increase indicating contact onset, a stable plateau reflecting a steady grasp, and a final drop denoting the release phase. Spatially, variations in grasp position and strength lead to complex pressure distributions, influenced by object geometry and hand posture. Traditional threshold-based encoding (Fig.7(a)), while simple and hardware-efficient, often fails to capture fine-grained variations near the threshold, leading to significant information loss. In contrast, frequency-based encoding (Fig.7(b)) assigns spike counts proportional to signal magnitude across defined value intervals. However, this approach tends to produce excessive spikes in stable, low-entropy regions—such as during steady grasps—thereby wasting computational and energy resources.



(a) Threshold coding. (b) Frequency coding. (c) *SparsePulse*.

Fig. 7. How different spike coding methods encode a grasping tactile signal. *SparsePulse* is designed to reduce the number of spikes while maintaining key information.

To address this issue, we observe that in typical pressure signals, segments with rapid temporal changes—i.e., high-entropy regions—carry more discriminative features essential for object recognition. In contrast, sustained high-pressure phases during stable grasping tend to be low-entropy and information-sparse. Building on this insight, we propose a novel spike encoding scheme, *SparsePulse*, illustrated in Fig.7(c). This method adaptively increases spike density during brief contact and release phases—where pressure is low but dynamic—capturing fine-grained temporal details. Conversely, spike generation is suppressed during prolonged, stable holding phases to reduce redundancy and conserve computational resources. This entropy-aware encoding strategy strikes a balance between precision and efficiency, significantly lowering power consumption without sacrificing critical tactile information. The complete encoding procedure of a set of timing pressure values is outlined in Algorithm 1.

ALGORITHM 1: *SparsePulse* Algorithm

Input: A set of timing pressure values PV; Threshold T; Number of intervals n; Maximum pressure P_M; Maximum spike count p_{max}; Step size Δp
Output: Encoded spike train S
Initialize: begin
  Initialize empty list: $S \leftarrow \emptyset$;
  Compute interval width: $r \leftarrow \frac{P_M - T}{n}$;
end
foreach *data point* $x \in PV$ **do**
  if $x < T$ **then**
    S.append(0);
  end
  else
    $s \leftarrow \min(\lfloor \frac{x-T}{r} \rfloor + 1, n)$;
    $p \leftarrow \max(p_{max} - (s-1) \cdot \Delta p, 0)$;
    S.append(p);
  end
end
return S;

Due to inherent noise fluctuations in the sensor's resting state—which can vary with environmental conditions and sensor aging—we design a dynamic thresholding mechanism to distinguish between noise and valid pressure

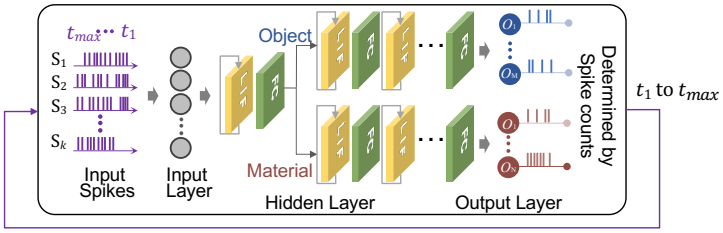


Fig. 8. Structure of *MT-SNN* .

signals. The dynamic threshold is computed in real time based on statistical properties (e.g., moving average and standard deviation) within a sliding temporal window, and is defined as $\theta = \mu + k\sigma$, where k is a tunable parameter typically set between 2 and 3. Each pressure data point x is first compared with the threshold θ. If $x < T$, it is encoded as 0, indicating that the segment contains no salient signal and does not require fine-grained encoding. When $x \geq T$, the signal is segmented for encoding. Assume the supra-threshold values are partitioned into n intervals, with corresponding boundaries $[\mu_0, \mu_1, \ldots, \mu_n]$. For each data point x, its interval index s is determined by comparing x with the boundary values.

Next, a spike generation function $p = f(s)$ is defined, where p is the number of spikes to be generated. As s increases, p decreases accordingly. A simple implementation can adopt a linearly decreasing function, such as $p = p_{\max} - (s-1) \times \Delta p$, where $p_{\max}$ is the maximum number of spikes and Δp is the step size. When pressure is low (initial contact or release), more spikes are emitted to capture transitions information. When pressure stabilizes at high levels, fewer spikes suffice to represent steady-state information, reducing overall spike count and storage costs. Overall, this strategy can improve efficiency while preserving perceptual accuracy.

4.4 *MT-SNN*: A Multi-task Tactile Spiking Neural Network

In this section, we introduce *MT-SNN*, a lightweight multi-task SNN architecture designed for simultaneous recognition of object types and material stiffness. We first present the overall structure of *MT-SNN*, followed by a detailed description of its forward and backward propagation mechanisms, which differ fundamentally from those of conventional artificial neural networks (ANNs).

4.4.1 Network Architecture. *MT-SNN* exploits the temporal dynamics of spiking neural networks, eliminating the need for complex sequential models such as LSTMs or Transformers. As illustrated in Fig. 8, the architecture follows a hierarchical spike-based processing paradigm, consisting of three key components.

Input Layer. Serving as the first stage in processing external tactile signals, the input layer of *MT-SNN* is structurally aligned with the spatio-temporal pressure distribution matrix captured by the sensor array. It comprises K neurons, each corresponding one-to-one with a tactile sensor. These sensors convert complex tactile inputs into spike sequences, which are transmitted in real time to the input neurons. These input neurons perform basic functions, primarily responsible for receiving signals and conducting preliminary transmission, resulting in output spike sequences that closely match the original sequences from the sensors.

Hidden Layer. The hidden layer serves as the core processing area of the neural network, primarily functioning to deeply integrate and extract features from the signals received from the input layer. Each hidden layer neuron receives signals from multiple input layer neurons and dynamically computes its membrane potential through complex integration operations. When the membrane potential surpasses the preset threshold, the neuron emits a spike, forming a unique spike sequence. The refined processing of signals by hidden layer neurons is crucial for the network's efficient feature extraction and accurate pattern recognition, directly influencing its ability to understand and analyze tactile signals. We employ a hybrid architecture that shares common

representations while learning task-specific features independently. A LIF layer is first used to extract shared pressure distribution patterns during object contact. Subsequently, a dual-branch architecture composed of alternating LIF and fully connected (FC) layer is introduced. The object recognition branch learns spatial features from the pressure distribution patterns to distinguish object geometries, while the material recognition branch captures the temporal dynamics of contact pressure to effectively estimate object compliance and hardness.

Output Layer. The output layer consists of M object classification neurons and N material classification neurons, where each output neuron corresponds to a specific class. They receive processed signals from the hidden layer and generate their respective output spike trains. For each output neuron, the total number of spikes emitted up to a given time t is counted, and the predicted class is determined by selecting the neuron with the highest spike count.

4.4.2 Forward Propagation Mechanism of SNNs. Unlike traditional ANNs that use continuous real-valued activations, SNNs convey information through discrete spike trains over time. Hence, we now describe the forward propagation mechanism of SNNs. Their forward propagation relies on the dynamic behavior of biological neurons, transmitting information via sparse spikes and temporal coding. This process consists of the following key steps.

Membrane Integration. Neurons integrate incoming spikes from presynaptic neurons over time, updating their membrane potential accordingly. Each spike influences the membrane potential based on its associated synaptic weight and arrival time. The membrane potential update follows the rule:

$$u[t] = \alpha \cdot u[t-1] + \sum_i w_i \cdot x_i[t] \tag{2}$$

where α is the leakage coefficient (typically determined by τ), w_i is the synaptic weight, and $x_i[t]$ is the binary spike input (0 or 1) from neuron i at time t.

Membrane Leakage. In the absence of input, the membrane potential gradually decays over time, emulating the passive leakage observed in biological neurons. This behavior is modeled using the Leaky Integrate-and-Fire (LIF) framework. The decay rate α is defined as:

$$\alpha = e^{-\Delta t/\tau} \tag{3}$$

where τ is the membrane time constant.

Spike Firing. When the membrane potential exceeds a predefined threshold u_{th}, the neuron generates an output spike. This spike generation process can be mathematically formulated as:

$$P[t] = \begin{cases} 1, & \text{if } u[t] > u_{\text{th}} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

After firing, the membrane potential is reset to a predefined value u_{rest}, and may enter a refractory period during which no spikes can be generated.

Temporal Iteration. Next, SNNs propagate information through discrete time steps, with each step iteratively performing the above processes. For example, a 100 ms input sequence may be divided into multiple short time steps (e.g., $\Delta t = 1$ ms), during which the state of each neuron is updated.

Output Decoding. At the output layer, the class label is typically determined by selecting the neuron with the highest spike count over a given time window:

$$\hat{y} = \arg\max_k \sum_{t \in T} O_m(t) \tag{5}$$

where $O_m(t)$ denotes the output of the m-th output neuron at time t.

Loss Function. A cross-entropy loss computed over the total spike count across time is used independently for both the object and material classification branches. The final loss L is obtained by averaging the two with equal weights (0.5), ensuring balanced optimization across both tasks:

$$L = 0.5 \cdot L_{\text{obj}} + 0.5 \cdot L_{\text{mat}}$$

4.4.3 Backward Propagation Mechanism of SNNs. Due to the non-differentiability of spike events, traditional backpropagation cannot be directly applied in SNNs. Two alternative paradigms have been developed: unsupervised biologically inspired learning and surrogate gradient-based supervised learning.

Spike-Timing-Dependent Plasticity (STDP) [43] is an unsupervised learning mechanism that updates synaptic weights based on the temporal correlation between pre- and post-synaptic spikes. If a pre-synaptic spike occurs before a post-synaptic spike, the weight is increased; otherwise, it is decreased. While STDP is biologically plausible and effective for capturing short-term temporal relationships, it suffers from limitations such as poor scalability to complex tasks, sensitivity to precise spike timing, and difficulty in modeling long-term dependencies due to its local and unsupervised nature.

The Surrogate Gradient method [20] approximates the non-differentiable spike firing function by constructing a differentiable surrogate function, allowing the establishment of an end-to-end supervised learning framework. For example, using a sigmoid function as a surrogate, the firing function P can be approximated by $G(u) = \sigma(\beta(u - u_{th}))$, where $G(u)$ represents the surrogate function, and β controls the slope of the function. The derivative of the surrogate function $G(u)$ is near the threshold u_{th} exhibits a sharp peak, effectively approximating the threshold jump behavior of the spike function, thus providing a feasible way for gradient calculation. Specifically, for the gradient of the loss function L with respect to the weight w, it can be expressed as:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial G} \cdot \frac{\partial G}{\partial u} \cdot \frac{\partial u}{\partial w} \approx \frac{\partial L}{\partial G} \cdot G'(u) \cdot \frac{\partial u}{\partial w} \tag{6}$$

By leveraging the derivative of a surrogate function, this method circumvents the non-differentiability of spike generation, enabling smooth gradient propagation during backpropagation. Moreover, it facilitates efficient weight updates under the supervised learning framework.

4.5 Model Quantization and deployment

The mapping of SNNs to brain-inspired hardware involves three key stages: model loading and parsing, intermediate representation generation, and graph optimization verification. Model quantization, as a crucial optimization technique throughout the mapping process, optimizes hardware resource utilization by determining weight and activation bit-widths in the intermediate representation stage, and enhances storage density by reducing weight storage space in the code generation stage, aligning with the goal of run-length encoding compression for synaptic connection matrices. However, special attention must be paid to quantization-induced accuracy loss during the optimization verification stage to ensure the quantized model maintains expected performance levels. This section analyzes the specific causes of quantization-induced accuracy degradation and introduces *SpikeQAT* (Spike quantization-aware training). By incorporating quantization awareness during the training phase, *SpikeQAT* effectively addresses precision loss issues, ensuring *MT-SNN* models operate with high accuracy and low power consumption on neuromorphic hardware.

4.5.1 Analysis of Accuracy Loss in Model Deployment. The Tianjic chip adopts a heterogeneous architecture that enables the joint execution of SNNs and ANNs. Its FCcore units incorporate 16 MAC arrays optimized for low-precision operations such as INT8. To deploy models efficiently on-chip, data representations are quantized to

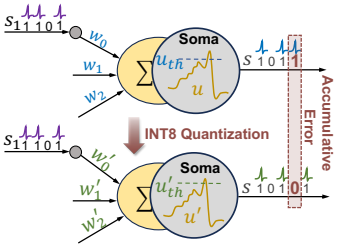


Fig. 9. Impact of quantization on spike emission.

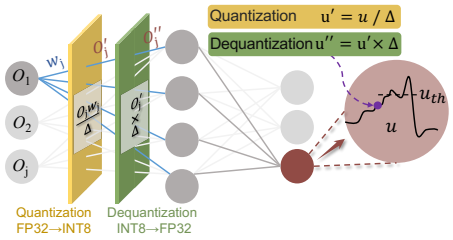


Fig. 10. Illustration of *SpikeQAT*.

reduce storage and computation costs, which introduces accuracy degradation. Quantization essentially reduces the bit-width of data representations to lower storage requirements and computational complexity, which is critical for the efficient operation of neuromorphic chips. For instance, converting from 32-bit floating-point (FP32) to 8-bit integer (INT8) substantially changes both the representational range and numerical granularity. FP32 uses a 1-bit sign, 8-bit exponent, and 23-bit mantissa, supporting a dynamic range of approximately $\pm 10^{38}$. In contrast, INT8 compresses values into a fixed range of [-128, 127] using just 8 bits. This conversion inevitably results in substantial loss of fine-grained information, which is a primary contributor to performance degradation (is proved in Fig. 18).

During quantization, continuous floating-point values are mapped to discrete integers using rounding:

$$Q = \text{round}\left(\frac{x}{\Delta}\right) \tag{7}$$

where Δ is the quantization factor and *zero_point* denotes the offset. As illustrated in Fig. 9, converting from 32-bit floating point (FP32) to low-bit integers (e.g., INT8 or INT4) causes significant information loss. Due to the temporal feature in SNNs, such errors affect not only weights but also disrupt spiking dynamics. For example, a neuron that is supposed to fire upon reaching a specific input threshold may fire prematurely if quantized weights distort the accumulated input. That is quantization errors accumulate through two critical operations: synaptic transmission and membrane potential updates and their impact increases exponentially with network depth.

4.5.2 SpikeQAT : Quantization-Aware Optimization Framework. To address this problem, inspired by simulated quantization noise techniques, we propose *SpikeQAT* , a quantization-aware training framework tailored for SNNs. As shown in Fig. 10, *SpikeQAT* inserts quantization/dequantization modules between adjacent LIF layers and within the membrane potential update path. These modules expose the network to realistic quantization effects during training, enabling it to adapt and learn to compensate for quantization-induced errors via backpropagation.

During forward propagation, quantization is introduced first at weight update using:

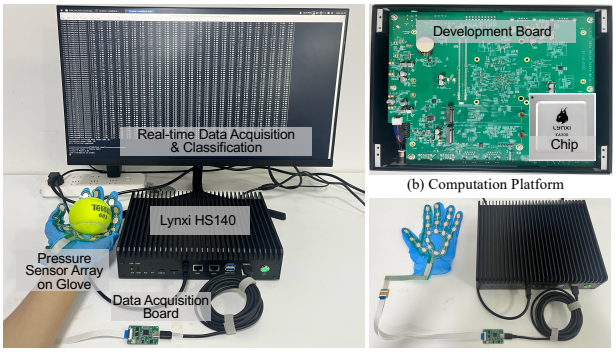
$$W_q = Q(W) = \text{round}\left(\frac{W}{\Delta}\right) \tag{8}$$

where $\Delta = \frac{W_{\max} - W_{\min}}{2^b - 1}$ is the quantization factor and b is quantified target number of digits. This process simulates quantization errors $\epsilon = W_q - W$ that occur during inference. This operation is followed by dequantization, the weight is,

$$W_q^{'} = W_q \Delta \tag{9}$$

Next, we introduce quantization operations $Q(\cdot)$ into the membrane potential u_t calculation process, to emulate quantization influence:

$$u_t^{'} = \frac{u_t}{\Delta} \tag{10}$$



(a) Overall Deployment (b) Computation Platform (c) Detailed Connection Diagram

Fig. 11. Deployment.

Fig. 12. 30 kinds of target objects.

and the membrane potential value after dequantization is:

$$u_t^{''} = u_t^{'}\Delta \tag{11}$$

This mechanism guides parameter updates toward quantization-friendly values. By jointly optimizing synaptic weights and membrane dynamics, *SpikeQAT* aligns the network's behavior during training and inference, thereby enhancing deployment performance while minimizing accuracy degradation.

5 Implementation

This section first introduces the experimental setup and deployment strategy, followed by a description of the data collection process and SNN training methodology.

5.1 Experimental Platform

Flexible Glove-Type Pressure Sensor. A commercial tactile glove RX-G0505M [49] integrating 25 flexible pressure sensors is employed for data acquisition. Each sensor can be modeled as a pressure-dependent variable resistor with an active area of 100 mm^2, providing a measurement range of 2.5 N–75 N, an accuracy of ±2.5 N, a resolution of 2.5 N, and a typical response time of <10 ms. The sensor array connects to the data acquisition circuit via FPC ribbon cables. During the data acquisition, the maximum ADC sampling frequency of 100 Hz. After analog-to-digital conversion, data is transmitted to the computing platform/PC using the USB-I2C protocol (as shown in Fig. 11(c)).

PC and Server. A PC running MATLAB and Python 3.8.10 is used for denoising, data augmentation, and spike encoding. The preprocessed samples are then uploaded to a server for model training. The server runs Ubuntu 16.04.1 (64-bit), equipped with an Intel Core i7-7820X processor (3.60 GHz), 64 GB of RAM, and an NVIDIA RTX 2080Ti GPU. Model training is conducted using the PyTorch 2.1.0 framework. Inference is executed within a Docker environment (Ubuntu 20.04.6 LTS, Linux kernel 4.15.0-142), supporting Python 3.8.10 and integrated with the LynSDK 1.20.0 software stack.

Neuromorphic Computing Platform. The trained network is deployed on the publicly available edge computing platform HS140, which integrates the KA200 neuromorphic processor (Tianjic chip) developed by Lynxi Technology [15, 48] and a dedicated ARM Cortex-A55 CPU for independent data processing. Tianjic chip adopts an innovative architecture that fuses in-memory computing, massively parallel cores, and heterogeneous integration. The HS140 host is also equipped with an 8-core Cortex-A55 processor and 16 GB RAM, running an embedded Ubuntu 18.04 OS. It supports the embedded version of LynSDK 1.20.0, enabling efficient and accurate

Table 1. 30 Objects and Their Hardness Categories

Hardness	Objects							
Hard	22 Mini Fan	13 Mouse	12 Red Bull	0 Caddy	10 Cube	19 Expansion Dock	24 Hexagonal Prism Box	23 Egg
Medium-Hard	21 Polygonal Box	8 Coca Cola	29 Medicine Box	17 Medicine Bottle	26 Candy Jar	7 Creatine Bottle	27 Hand-muscle Developer	
Medium-Soft	6 Tape	1 Orange	20 Bell Pepper	5 Cosmetic Bottle	14 Tennis Ball	28 Banana	25 Disposable Towel	/
Soft	4 Sponge	9 Bread	16 Bear Doll	11 Rubber dinosaur	3 Rubber Dog	2 Tissues	18 Bath Puff	15 Plush Toy

neural network mapping and inference. The chip is programmed using the Lyngor 1.19.2 graphical programming framework, which supports both basic and custom operators, is compatible with mainstream deep learning frameworks (e.g., PyTorch, TensorFlow), and generates instruction files for the chip via a parsing and compilation process. These files serve as the execution graphs for inference on the chip.

The complete real-time system diagram is shown in Fig. 11, which includes the pressure sensors and the neuromorphic hardware. The sensors are connected to a data acquisition module, which transmits data to the HS140 computation platform via USB and the I2C protocol. The CPU of HS140 converts the incoming pressure sequences into spike trains that serve as input to the pre-trained SNN for real-time prediction. Note that, during the SNN training phase, the acquisition module is connected to a PC to collect a large-scale dataset. The spiking neural network (SNN) is then designed and trained on the PC with GPU acceleration.

5.2 Experiment Setup

Data Collection. To ensure standardized and reproducible grasping actions, we adopted the authoritative human grasp taxonomy established by [13]. Specifically, we primarily used two representative grasp types – medium wrap and tip pinch – to collect pressure distribution data from different objects. We invited 20 healthy volunteers wearing sensor-equipped gloves to perform standardized grasping of 30 everyday objects, following reference images to ensure consistent hand postures. Each participant performed 20 grasping actions for each object, resulting in a total of 12000 samples. The data acquisition circuit sampled data at 60 Hz, collecting 100 frames of pressure data per grasp. As the pressure sensor array consists of 25 sensing points, each sample is represented as a 25×100 pressure matrix.

Test objects. A comprehensive list of these objects, along with their corresponding hardness classes, is provided in Table 1. To quantitatively evaluate the hardness classes (soft, medium-soft, medium-hard, and hard) of each object, we employ three types of Shore hardness durometers [46] to measure the hardness of all test objects. Specifically, the Shore O durometer was used for materials classified as soft to medium-soft, the Shore A durometer for medium-soft to medium-hard, and the Shore D durometer for medium-hard to hard materials. The hardness categories are defined as follows:

- **Soft:** Materials with a Shore O measurement < 50.
- **Medium-soft:** Materials with a Shore O measurement ≥ 50 and a Shore A measurement < 50.
- **Medium-hard:** Materials with a Shore A measurement ≥ 50 and a Shore D measurement < 70.
- **Hard:** Materials with a Shore D measurement ≥ 70.

5.2.1 Model Training and Hyperparameter Optimization. Training SNN is critical for effectively extracting spatiotemporal features from input data. Key training parameters are summarized in Table 2. We optimize network weights using the Adam algorithm. Early stopping is employed with a patience of 10 epochs, halting training when no improvement in validation loss is observed. To determine the optimal hyperparameters, we

Table 2. *SNN* Model Hyperparameters

Parameter	Value
Time Steps	100
Hidden Channels	512
u_{th}	0.65
Decay	0.389
Learning Rate	0.001

employ the Optuna [3] framework with Tree-structured Parzen Estimator (TPE) sampling [51], performing 30 optimization trials. The search space includes the membrane threshold u_{th}, decay factor, and learning rate. The configuration yielding the highest test accuracy is selected as the final model.

6 Evaluation

In this section, we first evaluate the overall performance of SpikeTouch, including object and material stiffness recognition accuracy, power consumption, and real-time inference capability. We then assess the effectiveness of the proposed pulse encoding module and the quantization-aware optimization strategy. Finally, we examine the system's robustness under user variability and sensor failure conditions.

6.1 Overall Performance

6.1.1 Classification Accuracy of Object and Material. All collected samples were first randomly shuffled using a fixed seed and then split into training and testing sets in an 8:2 ratio. The training set is further augmented to twice its original size (according to Sec.4.2.2). During training, we applied five-fold cross-validation: in each fold, 80% of the training set was used for model training, and the remaining 20% for validation to select the best-performing model. The results are based solely on the held-out test set, which remained completely unseen during training and validation. Confusion matrices for object and material classification on the test set are shown in Fig. 13 and Fig. 14, where rows correspond to ground truth classes and columns to predicted classes. The classification accuracies for object and material recognition reach 92.92% and 92.35%, respectively, demonstrating the excellent performance of SpikeTouch.

As shown in Fig. 13, the model achieves high classification accuracy across most object categories. Notably, "Creatine Bottle" (Class 7) and "Coca Cola" (Class 8) reach 100% accuracy, while "Rubber Dog" (Class 5), "Red Bull" (Class 12), "Mini Fan" (Class 22), and "Hand-Muscle Developer" (Class 27) all exceed 96%, indicating that their features are well captured by the model. In contrast, "Cube" (Class 10) shows a relatively lower accuracy of 78.1%, with the majority of misclassifications attributed to confusion with "Tissues" (Class 2, 4.1%). Note that some rows in the confusion matrix may not sum to exactly 100% due to rounding and minor decimal truncation in the visualization.

As shown in Fig. 14, the model achieves an average classification accuracy of 92.35% across the four material categories. The highest performance is observed in the hard category (93.9%), followed by soft (93.7%), medium-hard (91.4%), and medium-soft (90.4%). Most misclassifications occur between adjacent hardness levels, with the greatest confusion found in the medium-hard class—4.1% of its samples are misclassified as medium-soft, and 4.3% as hard. This trend suggests that the model effectively captures coarse-grained material differences (e.g., soft vs. hard), but struggles more with finer boundaries between adjacent categories. The confusion is largely due to the gradual transition in tactile feedback between medium-soft and neighboring classes, which leads to overlapping pressure patterns.

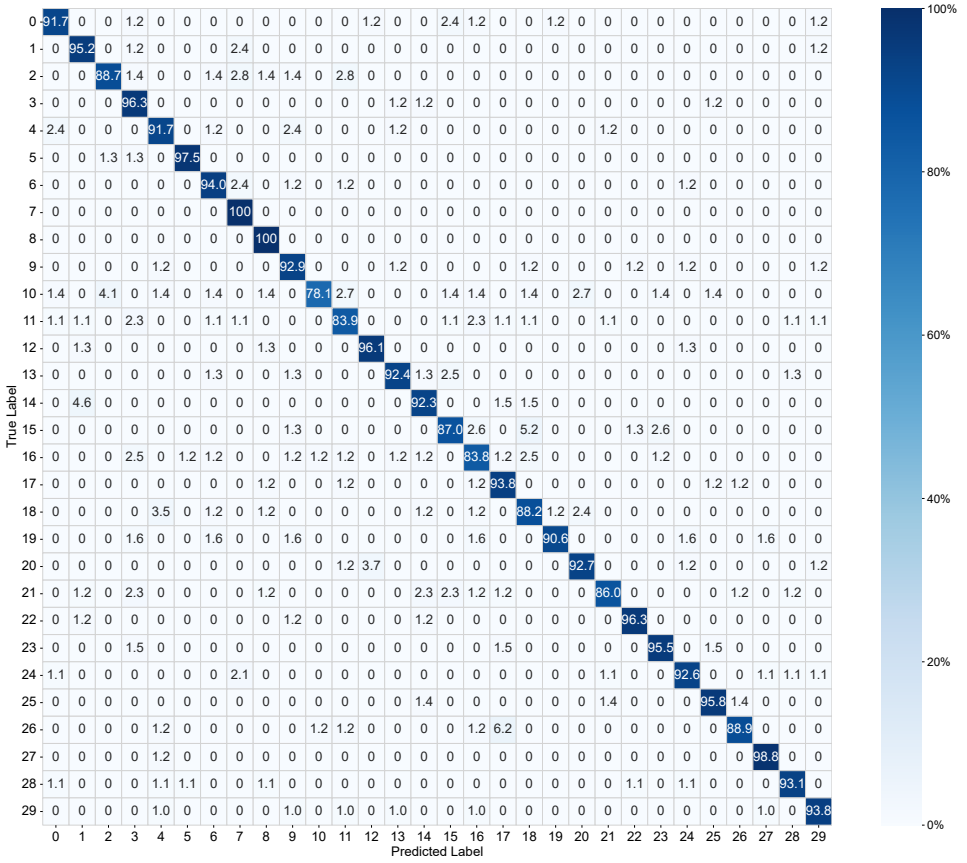


Fig. 13. The overall object classification accuracy of the SpikeTouch.

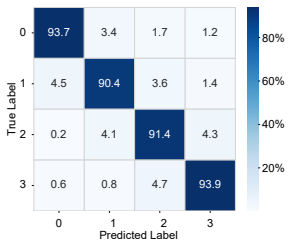


Fig. 14. Material classification accuracy of the SpikeTouch.

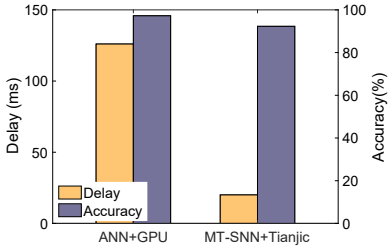


Fig. 15. Average per-sample processing latency.

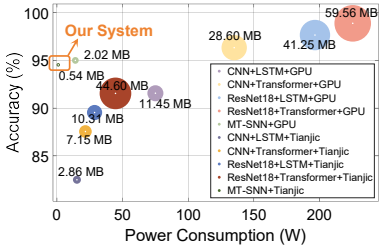


Fig. 16. Comparison across networks and platforms

6.1.2 Real-Time Performance Evaluation. To evaluate the real-time performance of the system, we conducted a comparative experiment under identical testing conditions using 2400 samples. As shown in Fig. 15, *MT-SNN* +Tianjic achieves an average per-sample processing latency of just 20 ms, which is 6.3× faster than the ANN+GPU solution (126 ms). In terms of object classification accuracy, *MT-SNN* +Tianjic reaches 92.92%, only 4.3% lower than ANN+GPU (97.22%). These results demonstrate that the proposed system substantially improves real-time responsiveness while maintaining competitive recognition performance.

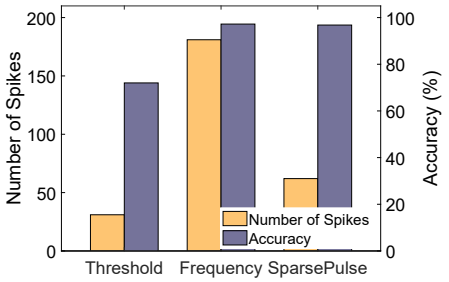


Fig. 17. Comparison of encoding methods.

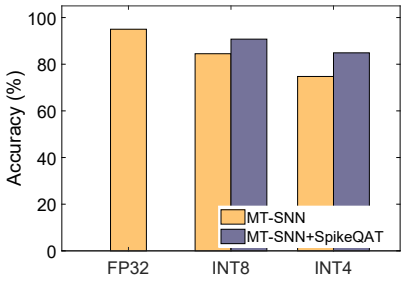


Fig. 18. Comparison of w/wo SpikeQAT.

6.1.3 Power Consumption Evaluation. To assess energy efficiency, we compare the power consumption of various neural network models on two hardware platforms: an NVIDIA GTX 2080Ti GPU and the Tianjic neuromorphic chip. This experiment involves object recognition across 20 categories using data collected from only 10 participants. The evaluation considers power consumption, model size, and object classification accuracy. Model size is measured by parameter storage size in megabytes (MB). The average power consumption per sample is calculated by subtracting the baseline power (measured with no code running) from the power consumed during test execution. Power measurements are recorded in watts (W).

As shown in Fig. 16, on the GPU platform, *MT-SNN* , achieves a power consumption of only 14 W, which is 18.67% of CNN+LSTM (75 W) and 6.21% of ResNet18+Transformer (225.3 W). Its model size (2.02 MB) is only 3.39% of the largest model while maintaining a 95% accuracy, just 2.18% lower than the best-performing model. On the Tianjic platform, *MT-SNN* further demonstrates its superiority-with only 1 W power consumption (a 92.86% - 99.5% reduction), 94.88% accuracy (3.34% - 12.38% higher than others), and the smallest model size (0.54 MB). Notably, traditional models experience up to 9.06% accuracy drop when migrated to Tianjic, while *MT-SNN* drops only 0.12%, highlighting its excellent adaptability to neuromorphic hardware platform.

6.2 Micro-benchmark

6.2.1 Validation of SparsePulse Encoding. To validate the energy and accuracy advantages of the *SparsePulse* encoding scheme, we conduct a comparative study against threshold and frequency encoding methods. For evaluation, single-channel tactile data from 200 tennis ball grasping trials collected from 10 users were used to assess spike count, while samples from 20 different objects were used to evaluate recognition accuracy. Given the event-driven nature of SNNs, power consumption is closely tied to spike density - inactive periods incur minimal static power, while spike generation and processing dominate dynamic energy consumption, spike count is adopted as a proxy for energy usage.

As shown in Fig. 17, *SparsePulse* generates an average of 62 spikes — more than threshold encoding (31 spikes), yet 65.7% fewer than frequency encoding (181 spikes), highlighting its potential for power savings. Experimental results show that *SparsePulse* achieves an object recognition accuracy of 97.22%, outperforming threshold encoding (72.00%) by 25.22%, and closely matching the performance of frequency encoding (98.60%), with only a 0.42% difference. These results demonstrate the superior energy–accuracy trade-off offered by *SparsePulse*.

6.2.2 Validation of SpikeQAT Optimization. As discussed in Sec. 4.5, quantization often leads to a reduction in model accuracy. To evaluate the effectiveness of *SpikeQAT* in mitigating quantization-induced degradation, we compare the 20 object classification accuracy of *MT-SNN* with and without applying *SpikeQAT* under three numerical precision settings: 32-bit floating point (FP32), 8-bit integer (INT8), and 4-bit integer (INT4). In the evaluation, we used tactile signals collected from 10 participants across 20 different objects. The results are shown in Fig. 18.

Table 3. Comparison of SpikeTouch with Existing SNN-Based Tactile Perception Methods

Method	Accuracy (%)	Model Size (MB)	Power (W)	Delay (ms)
TactileSGNet [16]	81.42	1.13	2.1	26
Method in [47]	90.83	0.79	2	46
DeepTactile [17]	75.15	1.58	2.3	31
SpikeTouch	**92.92**	**0.31**	**1.02**	**20**

We observe that the model running at full precision (FP32) achieves an accuracy of 95.00%. Without applying *SpikeQAT*, quantization to INT8 results in a significant drop in accuracy to 84.50% (a decrease of 10.50%), and further down to 74.75% with INT4 (a decrease of 20.25%). In contrast, when *SpikeQAT* is applied, the INT8 model achieves 90.75% accuracy, recovering 6.25% compared to the non-*SpikeQAT* case. For INT4, the accuracy improves to 84.88%, reducing the accuracy loss to just 10.12% and yielding a 10.13% improvement over the baseline. These results demonstrate that *SpikeQAT* effectively mitigates quantization degradation and enhances model robustness under low-bit settings.

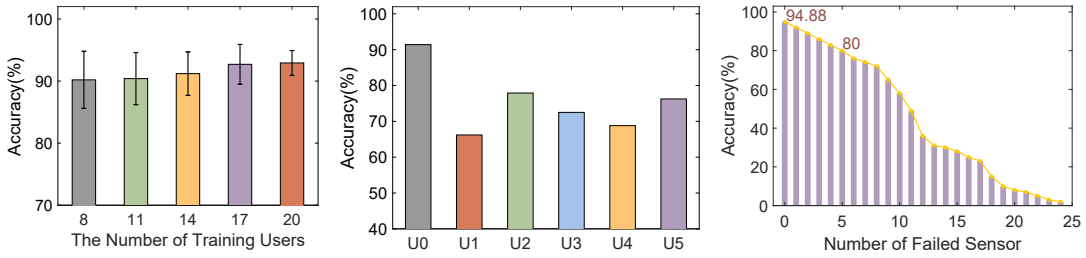
6.2.3 Comparison of State-of-the-art SNN-based Tactile Perception Methods. We reproduced three state-of-the-art SNN architectures [16, 17, 47], originally designed for tactile perception tasks, and evaluated them on our entire dataset for a fair comparison. The results of 30 object classification are summarized in Table 3. Our approach is able to deliver better accuracy to more complex convolution-based models such as those in [16, 17] with a smaller model size, faster inference time, and lower power consumption. These results demonstrate the effectiveness and efficiency of our approach, particularly for resource-constrained neuromorphic platforms.

6.3 System Robustness Evaluation

6.3.1 Different Numbers of Training Users. To investigate how the number of training users affects model performance, we gradually increased the training set size from 8 to 11, 14, 17, and 20 users. For each setting, the training data was split into training and validation subsets in an 8:2 ratio to select the best-performing model. As shown in Fig. 19(a), the average object classification accuracy consistently exceeds 90% across all training user settings, demonstrating the model's robustness to variations in the number of training users. Furthermore, increasing the number of users leads to reduced performance variance, indicating enhanced generalization capability.

6.3.2 Cross-User Robustness. To evaluate the system's generalizability, we conducted a cross-user experiment by training the model on data from 15 users and testing it on 5 previously unseen users. As shown in Fig. 19(b), the model achieved an average accuracy of 91.4% on training users, but dropped to 72.33% on unseen users U0-U5, with individual accuracies ranging from 66.2% to 77.9%. This performance gap highlights the system's limited robustness to user variability, likely caused by differences in hand shape and grasping force, which lead to varied pressure patterns despite standardized instructions. To address this, future work may incorporate domain adaptation techniques to enhance cross-user generalization.

6.3.3 Sensor Failure Robustness. To assess the system's fault tolerance, we simulate sensor failures on the 25-sensor glove. Using a pretrained model, the baseline classification accuracy reaches 95% with all sensors operational. Sensor failures are introduced by randomly deactivating the data from K sensors ($K \in [1, 25]$), following a Monte Carlo sampling scheme with 50 independent trials for each K value.



(a) Different training users count. (b) Robustness across users. (c) Robustness to sensor failures.

Fig. 19. Robustness evaluation under different conditions.

As shown in Fig. 19(c), classification accuracy gradually declines as the number of failed sensors increases. When $K \leq 5$, the accuracy remains above 80%, indicating a moderate level of fault tolerance. This robustness stems from the spatial redundancy and feature overlap across different regions of the hand—neighboring sensors often capture similar pressure patterns, enabling partial compensation for missing inputs. However, as more sensors fail, critical information about contact location and pressure dynamics is lost, leading to increased intra-class variance and misclassification.

7 Discussion

This section discusses the current limitations of SpikeTouch and proposes several potential solutions.

Limitation from Grasp Posture. SpikeTouch performs well when grasp postures closely match the standardized and stable conditions used during training. In such cases, the resulting pressure maps exhibit low intra-class variance, allowing the SNN to accurately associate inputs with learned features. However, in real-world scenarios, variations in grasp posture—such as changes in angle, contact region, or timing—can lead to significantly different pressure distributions for the same object. These variations increase intra-class variance and may be misinterpreted by the static model, resulting in false predictions under unconstrained conditions.

Limitations in Object Scalability. The system's scalability is challenged when encountering objects with high geometric similarity. While it performs well in recognizing objects with distinct shapes, the current model struggles to differentiate items that share similar base geometries—such as distinguishing a soda can from a pill bottle, both of which are cylindrical.This is because when inter-class similarity is high, the macro-level features extracted from the overall pressure profile are insufficient to capture the subtle, local details (e.g., the can's rim or the bottle's threads) that define the object's unique identity. Future work should integrate more powerful feature extraction networks or fuse multi-modal information to resolve this ambiguity.

Limitations in Fine-Grained Material Classification. While the system effectively distinguishes four coarse hardness levels, it struggles with fine-grained material differentiation (e.g., between wood and hard plastic, or silicone and rubber). The success in coarse-grained hardness classification is largely due to the overall magnitude and temporal trends of the pressure distribution can be captured by our lightweight SNN. However, fine material recognition often requires extraction of subtle features, such as surface texture-induced pressure gradients or viscoelastic responses over time. Our current network is not designed to capture such complexity, which limits its performance on detailed material classification tasks.

Limitation from User Variability. Our system exhibits limited robustness to user variability. Differences in hand shape, grasp strength, and grasping habits lead to diverse pressure patterns, even under standardized

instructions. This results in a noticeable performance gap when evaluating on unseen users. Future work may explore domain adaptation techniques to improve cross-user generalization.

8 Conclusion

We have presented the design, implementation, and evaluation of SpikeTouch, an SNN-based tactile perception system. By introducing a set of key techniques tailored for neuromorphic hardware, SpikeTouch achieves low power consumption, high efficiency, and accurate tactile sensing. Our evaluation, conducted on the Tianji neuromorphic hardware and tactile perception tasks, demonstrates the effectiveness of SpikeTouch. We hope the findings and empirical results provided in this work can contribute to further research into optimizing SNN-based solutions in resource-constrained environments.

Acknowledgments

Thanks the anonymous reviewers for their valuable comments. This work is supported in part by National Natural Science Foundation of China under Grants (62272388, 62302392), project of Shaanxi Province International Science and Technology Cooperation Programunder Grants (2024GH-YBXM-10, 2025GH-YBXM-057), and Shaanxi Science and Technology Innovation Team Program under Grant 2024RSCXTD05.

References

[1] [n. d.]. Tesla unveils its latest humanoid robot, Optimus Gen 2, in demo video — arstechnica.com. https://arstechnica.com/information-technology/2023/12/teslas-latest-humanoid-robot-optimus-gen-2-can-handle-eggs-without-cracking-them/.

[2] Larry F Abbott. 1999. Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain research bulletin* 50, 5-6 (1999), 303–304.

[3] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2623–2631.

[4] Daniel Auge, Julian Hille, Etienne Mueller, and Alois Knoll. 2021. A survey of encoding techniques for signal processing in spiking neural networks. *Neural Processing Letters* 53, 6 (2021), 4693–4710.

[5] Jose A Barreiros, Artemis Xu, Sofya Pugach, Narahari Iyengar, Graeme Troxell, Alexander Cornwell, Samantha Hong, Bart Selman, and Robert F Shepherd. 2022. Haptic perception using optoelectronic robotic flesh for embodied artificially intelligent agents. *Science Robotics* 7, 67 (2022), eabi6745.

[6] Libo Chen, Sanja Karilanova, Soumi Chaki, Chenyu Wen, Lisha Wang, Bengt Winblad, Shi-Li Zhang, Ayça Özçelikkale, and Zhi-Bin Zhang. 2024. Spike timing–based coding in neuromimetic tactile system enables dynamic object classification. *Science* 384, 6696 (2024), 660–665.

[7] Wenqiang Chen, Shupei Lin, Zhencan Peng, Farshid Salemi Parizi, Seongkook Heo, Shwetak Patel, Wojciech Matusik, Wei Zhao, and John Stankovic. 2024. ViObject: harness passive vibrations for daily object recognition with commodity smartwatches. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 8, 1 (2024), 1–26.

[8] Sayeed Shafayet Chowdhury, Isha Garg, and Kaushik Roy. 2021. Spatio-temporal pruning and quantization for low-latency spiking neural networks. In *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–9.

[9] Sungjun Chun, Jong Seok Kim, Yonghyun Yoo, Yoonho Choi, Seung Joon Jung, Dongki Jang, others, and Sung Park. 2021. An artificial neural tactile sensing system. *Nature Electronics* 4, 6 (2021), 429–438. https://doi.org/10.1038/s41928-021-00585-x

[10] Javad Dargahi and Siamak Najarian. 2004. Human tactile perception as a standard for artificial tactile sensing—a review. *The international journal of medical robotics and computer assisted surgery* 1, 1 (2004), 23–35.

[11] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. 2018. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro* 38, 1 (2018), 82–99.

[12] Lei Deng, Yujie Wu, Xing Hu, Ling Liang, Yufei Ding, Guoqi Li, Guangshe Zhao, Peng Li, and Yuan Xie. 2020. Rethinking the performance comparison between SNNS and ANNS. *Neural networks* 121 (2020), 294–307.

[13] Thomas Feix, Javier Romero, Herbert B Schmiedmayer, Aaron M Dollar, and Danica Kragic. 2015. The grasp taxonomy of human grasp types. *IEEE Transactions on Human-Machine Systems* 46, 1 (2015), 66–77. https://doi.org/10.1109/THMS.2015.2470657

[14] Samanwoy Ghosh-Dastidar and Hojjat Adeli. 2009. Spiking neural networks. *International journal of neural systems* 19, 04 (2009), 295–308.

[15] Github. [n. d.]. Edge computing host HS140. Website. https://github.com/XuanWang-kate/lynxi.git.

[16] Fuqiang Gu, Weicong Sng, Tasbolat Taunyazov, and Harold Soh. 2020. TactileSGNet: A Spiking Graph Neural Network for Event-based Tactile Object Recognition. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[17] Fangming Guo, Fangwen Yu, Mingyan Li, Chao Chen, Jinjin Yan, Yan Li, Fuqiang Gu, Xianlei Long, and Songtao Guo. 2024. Event-Driven Tactile Sensing With Dense Spiking Graph Neural Networks. *IEEE Transactions on Instrumentation and Measurement* (2024).

[18] Zhixian Hu, Lan Lin, Waner Lin, Yingtian Xu, Xuan Xia, Zhengchun Peng, Zhenglong Sun, and Ziya Wang. 2023. Machine learning for tactile perception: advancements, challenges, and opportunities. *Advanced Intelligent Systems* 5, 7 (2023), 2200371.

[19] Peng Kang, Srutarshi Banerjee, Henry Chopp, Aggelos Katsaggelos, and Oliver Cossairt. 2023. Boost event-driven tactile learning with location spiking neurons. *Frontiers in Neuroscience* 17 (2023), 1127537.

[20] Saeed Reza Kheradpisheh, Maryam Mirsadeghi, and Timothée Masquelier. 2022. Spiking neural networks trained via proxy. *IEEE Access* 10 (2022), 70769–70778.

[21] Jaehun Kim, Sungpil Kim, Jaewon Kim, Hyun Hwang, Jaeyoung Kim, Daehoon Park, and Unyong Jeong. 2020. Object shape recognition using tactile sensor arrays by a spiking neural network with unsupervised learning. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 178–183. https://doi.org/10.1109/SMC42975.2020.9283337

[22] Jin Kyong Kim, Cho Rok Lee, Sang-Wook Kang, Jong Ju Jeong, Kee-Hyun Nam, and Woong Youn Chung. 2024. Expansion of thyroid surgical territory through 10,000 cases under the da Vinci robotic knife. *Scientific Reports* 14, 1 (2024), 7555.

[23] Taeyoon Kim, Suman Hu, Jaewook Kim, Joon Young Kwak, Jongkil Park, Suyoun Lee, Inho Kim, Jong-Keuk Park, and YeonJoo Jeong. 2021. Spiking neural network (snn) with memristor synapses having non-linear weight update. *Frontiers in computational neuroscience* 15 (2021), 646125.

[24] Taehyeong Kim, Jaehun Kim, Inchan You, Jongho Oh, Sung-Phil Kim, and Unyong Jeong. 2022. Dynamic tactility by position-encoded spike spectrum. *Science Robotics* 7, 63 (2022), eabl5761. https://doi.org/10.1126/scirobotics.abl5761

[25] Espen Knoop, Moritz Bächer, and Paul Beardsley. 2017. Contact pressure distribution as an evaluation metric for human-robot hand interactions. In *HRI 2017 Workshop: Towards Reproducible HRI Experiments: Scientific Endeavors, Benchmarking and Standardization.*

[26] Chen Li, Lei Ma, and Steve Furber. 2022. Quantization framework for fast spiking neural networks. *Frontiers in Neuroscience* 16 (2022), 918793.

[27] Jing Shuang Li, Anish A Sarma, Terrence J Sejnowski, and John C Doyle. 2023. Internal feedback in the cortical perception–action loop enables fast and accurate behavior. *Proceedings of the National Academy of Sciences* 120, 39 (2023), e2300445120.

[28] Zenan Lin, Kai Chong Lei, Shilong Mu, Ziwu Song, Yuan Dai, Wenbo Ding, and Xiao-Ping Zhang. 2022. Multimodal surface sensing based on hybrid flexible triboelectric and piezoresistive sensor. In *Adjunct Proceedings of the 2022 ACM International Joint Conference on Pervasive and Ubiquitous Computing and the 2022 ACM International Symposium on Wearable Computers*. 421–426.

[29] Shan Luo, Wenxuan Mou, Kaspar Althoefer, and Hongbin Liu. 2015. Novel tactile-sift descriptor for object shape recognition. *IEEE Sensors Journal* 15, 9 (2015), 5001–5009.

[30] Changze Lv, Yansen Wang, Dongqi Han, Xiaoqing Zheng, Xuanjing Huang, and Dongsheng Li. 2024. Efficient and effective time-series forecasting with spiking neural networks. *arXiv preprint arXiv:2402.01533* (2024).

[31] Wolfgang Maass. 1997. Networks of spiking neurons: the third generation of neural network models. *Neural networks* 10, 9 (1997), 1659–1671.

[32] Luca Massari, Giulia Fransvea, Jessica D'Abbraccio, Mariangela Filosa, Giuseppe Terruso, Andrea Aliperta, Giacomo D'Alesio, Martina Zaltieri, Emiliano Schena, Eduardo Palermo, et al. 2022. Functional mimicry of Ruffini receptors with fibre Bragg gratings and deep neural networks enables a bio-inspired large-area tactile-sensitive skin. *Nature Machine Intelligence* 4, 5 (2022), 425–435.

[33] Kazuya Matsuo, Kouji Murakami, Tsutomu Hasegawa, and Ryo Kurazume. 2008. A decision method for the placement of mechanical tactile elements for grasp type recognition. In *SENSORS, 2008 IEEE*. IEEE, 1472–1475.

[34] Mahmoud Meribout, Natnael Abule Takele, Olyad Derege, Nidal Rifiki, Mohamed El Khalil, Varun Tiwari, and Jing Zhong. 2024. Tactile sensors: A review. *Measurement* (2024), 115332.

[35] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. 2014. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* 345, 6197 (2014), 668–673.

[36] Vimal Mollyn, Karan Ahuja, Dhruv Verma, Chris Harrison, and Mayank Goel. 2022. SAMoSA: Sensing activities with motion and subsampled audio. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 3 (2022), 1–19.

[37] William Navaraj and Ravinder Dahiya. 2019. Fingerprint-enhanced capacitive-piezoelectric flexible sensing skin to discriminate static and dynamic tactile stimuli. *Advanced Intelligent Systems* 1, 7 (2019), 1900051.

[38] João D Nunes, Marcelo Carvalho, Diogo Carneiro, and Jaime S Cardoso. 2022. Spiking neural networks: A survey. *IEEE access* 10 (2022), 60738–60764.

[39] Jing Pei, Lei Deng, Sen Song, Mingguo Zhao, Youhui Zhang, Shuang Wu, Guanrui Wang, Zhe Zou, Zhenzhi Wu, Wei He, et al. 2019. Towards artificial general intelligence with hybrid Tianjic chip architecture. *Nature* 572, 7767 (2019), 106–111.

[40] Balint Petro, Nikola Kasabov, and Rita M Kiss. 2019. Selection and optimization of temporal spike encoding methods for spiking neural networks. *IEEE transactions on neural networks and learning systems* 31, 2 (2019), 358–370.

[41] Clemens JS Schaefer and Siddharth Joshi. 2020. Quantizing spiking neural networks with integers. In *International Conference on Neuromorphic Systems 2020*. 1–8.
[42] Minwoo Seong, Gwangbin Kim, Jaehee Lee, Joseph DelPreto, Wojciech Matusik, Daniela Rus, and SeungJun Kim. 2024. Intelligent Seat: Tactile Signal-Based 3D Sitting Pose Inference. In *Companion of the 2024 on ACM International Joint Conference on Pervasive and Ubiquitous Computing*. 791–796.
[43] Sen Song, Kenneth D Miller, and Larry F Abbott. 2000. Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nature neuroscience* 3, 9 (2000), 919–926.
[44] Fuchun Sun, Chunfang Liu, Wenbing Huang, and Jianwei Zhang. 2016. Object classification and grasp planning using visual and tactile sensing. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 46, 7 (2016), 969–979.
[45] Subramanian Sundaram, Petr Kellnhofer, Yunzhu Li, Jun-Yan Zhu, Antonio Torralba, and Wojciech Matusik. 2019. Learning the signatures of the human grasp using a scalable tactile glove. *Nature* 569, 7758 (2019), 698–702.
[46] Syntek. 2025. German Shore Durometer for Objects Hardness Testing (Digital Type A/C/D). Taobao. July 29, 2025. https://detail.tmall.com/item.htm?from=cart&id=910656265200.
[47] Tasbolat Taunyazov, Yansong Chua, Ruihan Gao, Harold Soh, and Yan Wu. 2020. Fast texture classification using tactile neural coding and spiking neural network. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 9890–9895.
[48] Lynxi Technology. [n. d.]. Edge computing host HS140. Website. https://www.lynxi.com/lingqisupsupKA200xilie/34.html.
[49] Rouxi Technology. [n. d.]. RX-G0505M flexible glove-type force sensors. Website. http://www.roxifsr.com/productinfo/2109960.html.
[50] Chelsea Tymms, Esther P Gardner, and Denis Zorin. 2018. A quantitative perceptual model for tactile roughness. *ACM Transactions on Graphics (TOG)* 37, 5 (2018), 1–14.
[51] Shuhei Watanabe. 2023. Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance. *arXiv preprint arXiv:2304.11127* (2023).
[52] Liangshun Wu, Lisheng Xie, Jianwei Xue, Faquan Chen, Qingyang Tian, Yifan Zhou, Ziren Wu, Rendong Ying, and Peilin Liu. 2024. SPRCPl: An Efficient Tool for SNN Models Deployment on Multi-Core Neuromorphic Chips via Pilot Running. In *2024 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–5.
[53] Kashu Yamazaki, Viet-Khoa Vo-Ho, Darshan Bulsara, and Ngan Le. 2022. Spiking neural networks and their applications: A review. *Brain sciences* 12, 7 (2022), 863.
[54] Man Yao, Ole Richter, Guangshe Zhao, Ning Qiao, Yannan Xing, Dingheng Wang, Tianxiang Hu, Wei Fang, Tugba Demirci, Michele De Marchi, et al. 2024. Spike-based dynamic computing with asynchronous sensing-computing neuromorphic chip. *Nature Communications* 15, 1 (2024), 4464.
[55] Zukun Yu, Jing Yang, Qin Ran, Shaobo Li, and Zhidong Su. 2024. G2T-SNN: Fusing topological graph and Gaussian prior spiking neural networks for tactile object recognition. *IEEE Sensors Journal* (2024).
[56] Malu Zhang, Jiadong Wang, Jibin Wu, Ammar Belatreche, Burin Amornpaisannon, Zhixuan Zhang, Venkata Pavan Kumar Miriyala, Hong Qu, Yansong Chua, Trevor E Carlson, et al. 2021. Rectified linear postsynaptic potential function for backpropagation in deep spiking neural networks. *IEEE transactions on neural networks and learning systems* 33, 5 (2021), 1947–1958.
[57] Jiaxue Zhu, Xumeng Zhang, Rui Wang, Ming Wang, Pei Chen, Lingli Cheng, Zuheng Wu, Yongzhou Wang, Qi Liu, and Ming Liu. 2022. A heterogeneously integrated spiking neuron array for multimode-fused perception and object classification. *Advanced Materials* 34, 24 (2022), 2200481.
[58] Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian Reid. 2019. Structured binary neural networks for accurate image classification and semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 413–422.