Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

uSense: Unary-Computing-based Stochastic Edge Neuromorphic Sensing

Mohsen Rakhshan

mohsen.rakhshan@ucf.edu

University of Central Florida

Zubaidah Al-Mashhadani

University of Central florida

Di Wu

University of Central Florida https://orcid.org/0000-0001-9775-8026

Article

Keywords:

Posted Date: March 11th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-8961215/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: There is **NO** Competing Interest.

uSense: Unary-Computing-based Stochastic Edge Neuromorphic Sensing

Zubaidah Al-Mashhadani[1], Mohsen Rakhshan[1,2*], Di Wu[1*]

[1*]Department of Electrical and Computer Engineering, University of Central Florida, 4000 Central Florida Blvd., Orlando, 32816, FL, USA.
[2]Disability, Aging, and Technology Faculty Cluster, University of Central Florida, 4000 Central Florida Blvd., Orlando, 32816, FL, USA.

*Corresponding author(s). E-mail(s): mohsen.rakhshan@ucf.edu; di.wu@ucf.edu;

Abstract

Tactile sensing on the edge is a new frontier of embodied artificial intelligence, advancing novel applications like robotics, prostheses, and immersive virtual reality. However, the high-dimensional data generated from tactile arrays requires heavy computation and communication for real-time responses. To address this problem, we propose unary-computing-based stochastic edge neuromorphic sensing (uSense), a fast Fourier transform (FFT) framework based on unary computing for near-sensor tactile sensing. uSense encodes raw signals from the tactile sensors into rate-coded probabilistic bitstreams and performs stochastic FFT with extremely simple digital logic, alleviating memory, energy, bandwidth, and latency bottlenecks. Such benefits result from three key techniques: dual-domain (i.e., unary and binary) dynamic scaling, stage-wise bitwidth optimization, and computational graph pruning. First, *dual-domain dynamic scaling* ensures that our unary computing approach minimizes the numerical accuracy loss throughout the FFT computation. Second, *stage-wise bitwidth optimization* progressively tunes the data precision at each FFT stage, achieving one order of magnitude data compression with minimized accuracy loss. Third, *computational graph pruning* considers the task sensitivity of frequency components and removes redundant computation and memory access by 50%, achieving another order of magnitude data compression with minimized accuracy loss. Our experiments demonstrate that the uSense framework achieves 99.18% data compression, preserves 97.64% texture recognition accuracy across 21 textures, only 1.35% accuracy loss compared to floating-point baselines. Our analysis reveals an overlooked fact: *high numerical inaccuracy does not necessarily translate to low*

decoding inaccuracy, as long as the structure of the feature latent space is preserved. Further analytical modeling on FFT hardware shows uSense can reduce the computation and memory footprint of FFT by more than 50%, and enable real-time responses of FFT in less than 100 nanoseconds.

1 Introduction

Touch is the primary sense for understanding and interacting with the physical world. Mechanoreceptors within our skin transduce forces and vibrations into neural spike trains that propagate from the peripheral to the central nervous system via afferent pathways, enabling fine perception and control[1–4]. It is shown that sensory feedback is essential for dexterous object manipulation, safe interaction with the environment, embodiment, motor coordination, and reducing the cognitive load[1,4–7]. Without tactile information, and even in the presence of visual feedback, simple tasks such as lighting a match become challenging[8]. Therefore, research on tactile sensing has become a key frontier of next-generation embodied artificial intelligence (AI) for broader applications, such as intelligent robotics, industrial manipulation, and virtual reality[1,4,5,5,7,9–13].

1.1 Limitations in Current Tactile Sensing Systems

Despite of the recent promising progress towards sensory-feedback augmentation[6,14], translating human-like tactile sensing and sensory feedback into artificial systems remains a significant challenge in robotics and neuroscience[6,15–20]. Varying algorithms have been used to digitize touch through texture recognition, ranging from deep neural networks (DNNs)[21–23] to classical machine learning[24–27]. DNNs can learn hierarchical features from the raw data directly, however, at the cost of significant computation and memory footprint. On the other hand, classical machine learning models, such as support vector machine (SVM) and k-means, achieve competitive decoding accuracy with fewer hardware overheads, but require complicated yet costly pre-processing. These facts indicate clear gaps in addressing both efficiency and accuracy simultaneously in real time.

There are two fundamental limitations in current tactile sensing systems that hinder broader adoption. The first limitation is that tactile sensors generate a huge volume of sensory data that are too complex and costly to process and transmit at scale. During active exploration, tactile sensors generate high-bandwidth, high-dimensional spatiotemporal signals, which impose task-dependent overfitting risks[28], creating a mismatch between the high-throughput tactile signals and existing systems. Existing systems usually require a sampling rate between 1kHz and 100kHz per channel[29–34] with up to 4096 channels[35,36] in a single tactile sensor. Such a huge data volume exceeds the capability of low-power wireless communication protocols, such as Bluetooth[37] and Zigbee[38], leading to information loss or prolonged latency. As a result, tactile sensing systems require task-specific feature extraction and data compression to be performed directly at or near the sensing interface, under low-power and area

constraints, to support real-time and scalable artificial touch. Near-sensor computing has therefore emerged as a promising paradigm, pushing signal processing directly to the sensing interface to address the bandwidth and latency bottlenecks of edge sensing applications, reducing data transmission and memory overhead[39,40].

The second limitation is that the high-precision, 32-bit floating-point (FP32) binary data used in conventional tactile sensing systems enforce expensive binary circuits, imposing substantial area and energy overheads[41,42]. Even with near-sensor computing adopted, the high-precision for each data point incurs substantial hardware costs due to parallel multi-bit processing and exponentially increasing wire congestion[43] and costly analog-to-digital conversion[40], which increases with signal dimensionality. Such a wide data format further enlarges the need of memory bandwidth and capacity, limiting the adoption of large-scale tactile sensing systems for complex tasks. To reduce the data bitwidth, spike-based neuromorphic computing has been explored as a promising alternative for tactile sensing and texture recognition[44–46]. These systems aim to mimic biological transduction by converting continuous sensor outputs into event-based spike representations inspired by mechanoreceptor physiology[47,48]. Despite these advantages, neuromorphic tactile systems rely on specialized circuits or devices that limit their integration into existing edge systems. Furthermore, their DNN-alike nature demands a large number of weights that have to be stored in memory, which amortizes the benefit from reducing data bitwidth.

Together, these limitations highlight *a persistent gap between algorithmic advances and practical on-device implementation*, given the complicated trade-off among decoding accuracy, hardware efficiency, and real-time response. Motivated by the limitations of conventional binary computing and the memory constraints of neuromorphic computing, we consider unary computing[39,43,49] for near-sensor tactile sensing. Unary computing adopts an unconventional numerical representation based on serial bitstreams rather than parallel bits, and operates on these bitstreams with extremely simple hardware, making it suitable for massively-parallel and low-power processing.

1.2 Challenges in Unary Tactile Sensing

Despite numerous advantages of unary computing, it has not achieved broad adoption in real-world applications due to inherent representational constraints. Unary representations encode numerical values in either unipolar or bipolar format, bounded by the ranges $[0, 1]$ and $[-1, 1]$, respectively[43]. The numerical values are expressed through the probability of 1s in a stochastic bitstream[50].

Beyond the limited range of unary values, a fundamental obstacle in deploying unary computing for real-world applications lies in cumulative numerical distortion. If not taken care of properly, cascaded arithmetic operations in deep computational pipelines can introduce significant degradation in numerical accuracy, thus potential information loss, in varying forms, thereby jeopardizing the end-to-end decoding accuracy. For example, in unary computing, cascaded scaled additions drive the probability values toward zero, causing the signal-to-noise ratio (SNR) to deteriorate sharply, a phenomenon commonly referred to as the scaling effect[51].

Importantly, existing research on unary computing has primarily focused on isolated arithmetic kernels or synthetic data[50,52,53], rather than large-scale, real-world data. We summarize the principal challenges associated with unary computing, especially for tactile sensing:

1. **Numerical inaccuracy ties to low latency.** Numerical precision in unary representations is proportional to bitstream length, which grows exponentially with bitwidth. Increasing bitwidth improves probabilistic resolution but substantially increases latency, thus the computational costs[43,54]. In tactile sensing, where real-time response is favored, low latency directly ties to *numerical inaccuracy*, imposing a trade-off between latency and accuracy.
2. **Numerical saturation drops information.** Unary computing operates within a strictly bounded numerical range for both unipolar and bipolar formats. Unary values that exceed the representable bounds will be clipped to the corresponding bound, i.e., *numerical saturation*. Numerical saturation inevitably results in irreversible information loss that propagates through subsequent computations[40,43,55]. Though numerical saturation is not a major concern for isolated operations, it is pervasive in algorithms with cascaded operations like the fast Fourier transform (FFT), where additions will be performed at every FFT stage.
3. **Numerical vanishing degrades SNR.** To mitigate excessive numerical saturation that leads to information loss, unary computing can use scaled additions, which divide the sum of all inputs by a scaling factor, to keep values within the legal range. However, frequent scaled additions will progressively scale down the output, causing the unary values to approach zero, i.e., vanishing[51,56]. Ultimately, the small unary values from excessive scaled additions, together with the persistent, uncontrollable stochastic noises, effectively lower the SNR and incur information loss.

Collectively, these constraints outline the bottlenecks that prevent unary computing from research beyond simple, isolated kernels, despite unary computing offering unparalleled efficiency. Addressing these challenges is essential to transitioning unary computing from a theoretical computing paradigm to a viable backbone for end-to-end, real-world applications such as tactile sensing.

1.3 Proposed Solution to Unary Tactile Sensing

Given the practical limitations of tactile sensing and the specific challenges of unary tactile sensing, we propose unary computing-based stochastic edge neuromorphic sensing (uSense), a near-sensor tactile sensing framework that performs FFT in the unary domain for texture recognition decoding. To the best of our knowledge, this work presents the first unary-based tactile sensing for real-world applications. The adoption of unary computing in uSense draws inspiration from the biological efficiency of the human somatosensory system. Tactile information from thousands of mechanoreceptors is encoded using event-driven spikes, thus neuromorphic, and transmitted with minimal energy and bandwidth[14]. Leveraging unary computing, uSense is lightweight enough to serve as a near-sensor, in-situ computing engine for tactile sensing. The

advantages of uSense stem from addressing the challenges posed by unary computing through a set of optimizations designed to maximize decoding accuracy and data compression, while minimizing computation complexity and latency simultaneously.

First, to address the challenges of numerical saturation and numerical vanishing that lead to information loss and SNR degradation, we introduce a *dual-domain dynamic scaling* strategy in both binary and unary domains. In the binary domain between FFT stages, binary input values are adaptively scaled before they are converted to unary bitstreams, in order to fully utilize the unary data range to avoid numerical saturation and vanishing. In the unary domain within each FFT stage, unary scaling dynamically determines when and how to scale down the addition results to minimize the numerical saturation and vanishing.

Second, to mitigate the challenge of low latency tied to numerical inaccuracy, we propose *stage-wise bitwidth optimization* to choose the minimum bitwidth at each FFT stage without compromising decoding accuracy. We first minimize the bitwidth of the final FFT stage with no accuracy drop in texture recognition to maximally compress the output of the FFT, so that the following communication cost can be minimized. Then we progressively optimize the bitwidth of other FFT stages, one after another, to compress the data in all FFT stages, again without accuracy loss in decoding textures. The resulting unary FFT, with dual-domain dynamic scaling and stage-wise bitwidth optimization applied, shows one order of magnitude data reduction in both final and full FFT stages, with negligible loss in decoding accuracy.

Finally, to further reduce the computation cost of FFT and the communication cost of FFT outputs, we propose *computational graph pruning* to focus on the critical frequency components that contribute the most to decoding accuracy. Rivera et al.[27] implemented FFT-based piezoresistive tactile feature extraction to decode 12 textures, achieving 91.4% accuracy with a k-means classifier. Their results revealed that discriminative information mainly resides in a narrow low-frequency band. This observation motivates us to selectively process a few important frequency components to achieve task-aware tactile sensing to maximize the hardware efficiency. We further designed optimized stochastic butterfly units (BUs) that reduce the number of compute elements.

Together, these optimizations form a unified stochastic neuromorphic FFT pipeline optimized for real-time tactile sensing, offering orders-of-magnitude compression and efficiency improvement while maintaining high accuracy and significantly reducing hardware energy and area. Our major contributions in this paper are summarized as follows:

- We design a unary-computing-based stochastic edge neuromorphic sensing framework (uSense) to perform FFT for near-sensor texture recognition.
- We develop a dual-domain dynamic scaling strategy that mitigates numerical saturation and numerical vanishing challenges in unary computing.
- We develop a stage-wise bitwidth optimization method to reduce the processing latency while minimizing the potential loss in decoding accuracy.
- We develop a computational graph pruning approach to focus on important frequency components to maximize computation and communication efficiency.

- We reveal that high numerical inaccuracy does not necessarily translate to low decoding inaccuracy, as long as the structure of the feature latent space is preserved.
- We demonstrate that uSense achieves two orders of magnitude data compression (99.18%) with high decoding accuracy (97.64%) across 21 textures, with more than 50% projected reduction in computation and memory footprint in less than 100 nanoseconds.

The remainder of this paper is organized as follows. First, Section 2 reviews unary computing. Section 3 presents our dataset for tactile sensing, and details our proposed design of the stochastic FFT. Next, Section 4 evaluates our design and presents the quantitative results. Finally, Section 5 and 6 discuss and conclude the work.

2 Background

Unary computing represents data with serial bitstreams[43], and unary computing with rate coding is also known as stochastic computing[50].[1] Instead of operating on bit-parallel binary numbers, unary computing performs arithmetic directly on serial stochastic bitstreams (spike trains) using simple and efficient hardware elements. Unary computing is promising for in-situ processing while improving area and power efficiency compared to conventional binary designs[39,57].

2.1 Unary Data Representation

Unary data with rate coding, i.e., a stochastic bitstream, encodes a value based on the probability of 1s it contains[56]. Furthermore, a bw-bit binary number can be represented as a stochastic bitstream of length 2^{bw} with identical resolution[58]. The most widely used encoding representations of stochastic numbers are the unipolar and bipolar formats. In a unipolar format, a value in the range [0,1] is encoded as the probability of a bit being 1 in the bitstream, as shown in Fig. 1, while the bipolar format extends the value range to [-1,1], supporting negative numbers. Therefore, any real number can be scaled and mapped into a rate-coded bitstream[43].

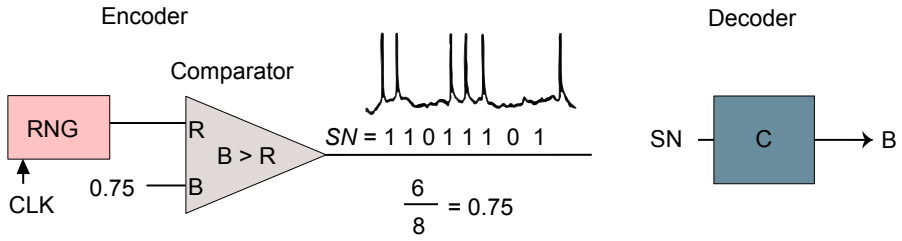


Fig. 1 Encoder (left) and decoder (right) for unary computing with rate coding. A binary number B is encoded into a unary stochastic number SN, by comparison with a random value R from a random number generator RNG at each clock cycle CLK. The resulting bitstream SN resembles a train of action potentials in the nervous system. Decoding is performed by counting ones in a SN with a counter (C).

[1] We will use unary and stochastic interchangeably in this paper.

Generating and retrieving stochastic bitstreams can be done via an encoder and decoder, as shown in Fig. 1. The encoder compares a sequence of random numbers between 0 and 1 with the source binary number. A bit 1 is generated whenever the source binary number is larger than the random number. For example, a binary number 0.75 is compared to a sequence of 8 random numbers, and the output will have 6 out of 8 total bits to be 1. The decoder converts a bitstream back to a binary number by counting the number of 1s.

2.2 Unary Computing Unit

Unary computing units can be realized using simple logic gates. For unipolar bitstreams, the product of two input bitstreams is obtained through an AND gate. For bipolar bitstreams, multiplication is implemented using an XNOR gate[50]. However, both unipolar and bipolar formats can use the same adder circuits. A conventional two-input unary adder is implemented with a multiplexer (MUX) whose select signal is a bitstream with a probability of 0.5[50].

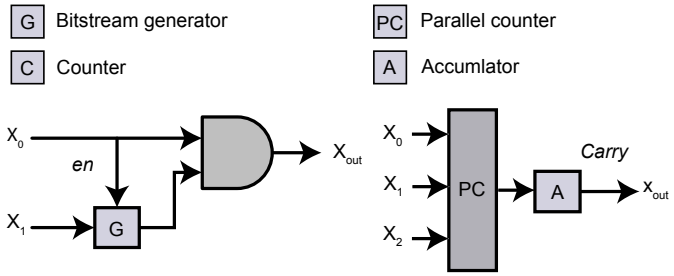


Fig. 2 Multiplier (left) and adder (right) in unary computing.

However, such simple circuit implementations are built on the assumption that bitstreams are statistically uncorrelated, where 1s and 0s are independently distributed. Correlation between bitstreams can happen if the positional distributions of 1s are similar[59]. When input bitstreams are correlated, the computed probability deviates from the ideal output, introducing uncontrollable computational errors[60]. Prior work has explored various strategies to address this correlation problem, including using a deterministic approach or scrambling the bitstream at the expense of longer latency and lower accuracy[59–63]. Recent work has innovated more efficient solutions to this challenge. Wu et al.[43] proposed a unary General Matrix Multiplication (uGEMM) architecture that relaxes the constraints on bitstream length and correlation while maintaining high accuracy and fast convergence[54], making uGEMM suitable for resource-constrained systems. In this work, we follow the uGEMM formulation to ensure high accuracy and low latency, as shown in Fig. 2. On the left and right are the multiplier and adder for the unipolar format. For the multiplier, the top input is a bitstream, while the bottom input is a binary number. This organization naturally matches the multiplication in FFT, an arbitrary input multiplying a fixed twiddle factor. The enable signal (en) ensures correlation-free multiplication for high accuracy[43].

The adder utilizes a parallel counter followed by an accumulator with carry overflow to alleviate the correlation issue in conventional MUX-based implementations[43,56].

Unary computing reduces hardware area and power consumption by large margins over binary computing[39,49,64], hinting at its potential for near-sensor tactile sensing. Furthermore, unary computing aligns naturally with the physical structure of tactile sensing systems. Tactile arrays generate concurrent multi-channel signals, where each taxel's bitstream can be processed in parallel using simple unary computing units. Moreover, unary computing is particularly well-suited for sensor streaming. Outputs of analog sensors can be converted to bitstreams through lightweight analog-to-stochastic converters, which are then processed directly without being stored into memory[40,43,65]. Therefore, unary computing is a perfect fit for near-sensor tactile sensing that supports massive parallelism and lowers the energy demands from data conversion and data storage, which bottleneck conventional systems.

3 Methodology

Our proposed work is inspired by biological tactile perception, in which tactile information is encoded as probabilistic spike trains rather than high-precision numerical values. The probabilistic data representation in unary computing aligns well with this perspective, making unary computing well-suited for spike-based tactile sensing at the edge. Fig. 3 illustrates this biological inspiration and its abstraction into an artificial tactile sensing system based on unary computing, mimicking an active human touch.

Translating such biologically inspired representations into practical applications confronts persistent challenges that are inherent to unary computing, including numerical inaccuracy, saturation, and vanishing. This work presents uSense, an optimized unary-computing-based stochastic edge neuromorphic sensing framework for FFT-based near-sensor tactile sensing. uSense features a stochastic FFT architecture, and exhibits enhanced processing latency, decoding accuracy, and data compression via three techniques, including dual-domain dynamic scaling, stage-wise bitwidth Optimization, and computational graph pruning. We design controlled experiments to rigorously evaluate the decoding accuracy of the proposed uSense framework.

The following subsections detail the uSense framework and evaluation. We first describe the tactile dataset and data acquisition protocol (Section 3.1), followed by signal preprocessing (Section 3.2). We then introduce our stochastic FFT architecture (Section 3.3), dual-domain dynamic scaling (Section 3.4), and compression strategies based on stage-wise bitwidth optimization (Section 3.5) and computational graph pruning (Section 3.6), and conclude with the evaluation methodology (Section 3.7).

3.1 Dataset

We collected a tactile dataset using a 9-taxel (3×3) piezoresistive array mounted on a 3D-printed artificial finger. The sensor was placed on the fingertip, replacing a 3D printer filament nozzle, as shown in Fig. 4. During data collection, the finger swiped horizontally, parallel to the texture plane at controlled speeds and pseudo-forces. Each taxel measured voltage variations corresponding to pressure, while a load cell beneath the texture recorded the applied normal force. Trials followed a consistent sequence

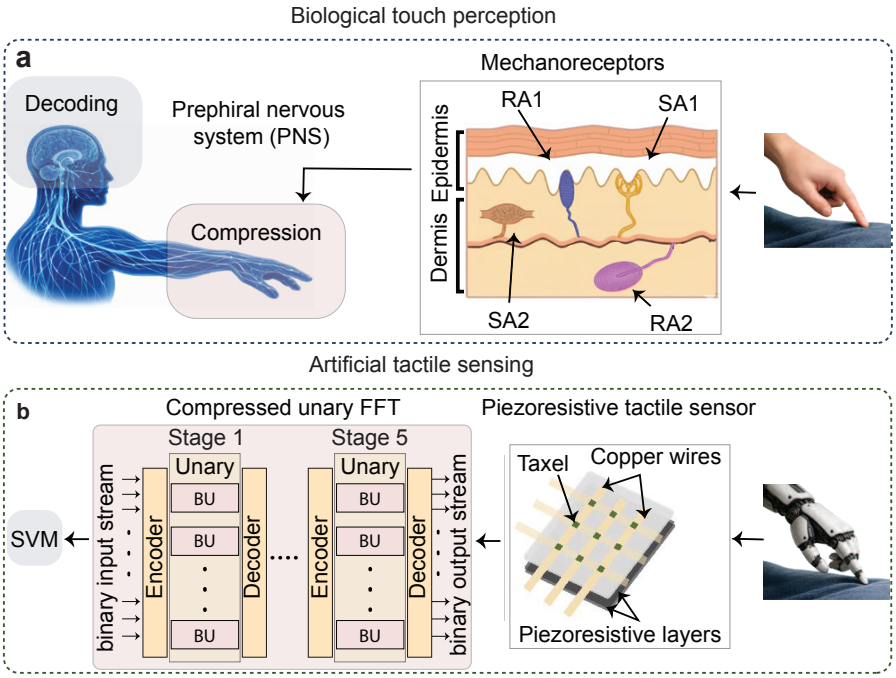


Fig. 3 An artificial robotic sensory system mimicking human tactile perception for texture recognition. **a** Biological tactile sensing in humans. Interactions with textures are transduced by cutaneous mechanoreceptors in the epidermis and dermis and encoded through the peripheral nervous system for downstream neural decoding. **b** Artificial tactile sensing system proposed in this work, comprising a piezoresistive tactile sensor array and an ultra-compressed unary FFT processing pipeline. Tactile signals are encoded into bitstreams, processed using a stochastic Cooley–Tukey algorithm based on butterfly units (BUs), and decoded into spectral features for texture recognition. The proposed pipeline performs feature extraction and compression at the sensing interface for efficient tactile sensing under resource constraints.

of phases, including free-air motion, stationary contact, active texture interaction, and contact-free motion. Data were recorded at a sampling rate of approximately 338 Hz, and the temporal structure of each trial varied with the swiping speed. The recorded data included tactile signals from the 9-taxel array, load-cell force readings, and timestamps.

A total of 21 distinct textures were explored, including a controlled flat texture. Each texture combined one of the four bump geometries (flat, ridge, hemispherical, and conical) with one of five bump densities (2–6 rows within a constant distance). Each texture was tested at five swiping speeds (1200, 2400, 3600, 4800, 6000 mm/min) and four pseudo-force levels (0.0, 0.5, 1.0, and 1.5 mm; created by lowering the finger into the texture), resulting in 400 unique texture-force-speed combinations. For each combination, 100 trials were collected, resulting in a total of 42,000 trials.

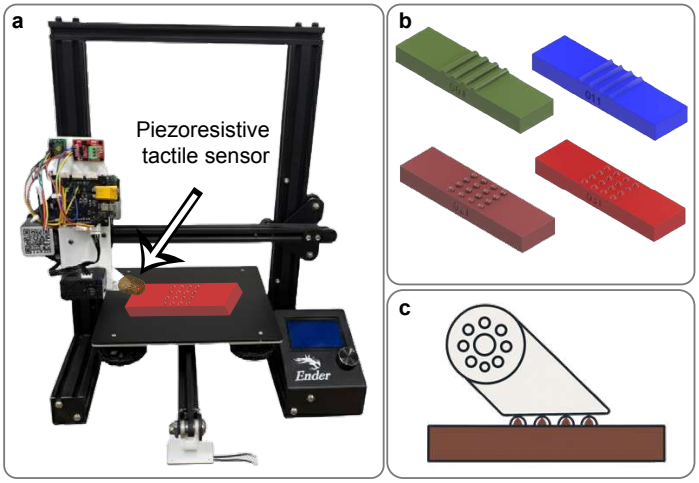


Fig. 4 Experimental setup for tactile data collection. **a** Modified 3D printer to swipe the piezoresistive tactile sensor over a texture sample. **b** Examples of 3D-printed textured surfaces. **c** Schematic of the piezoresistive sensor interaction with a textured surface.

3.2 Preprocessing

Outlier removal was first applied based on the load-cell readings, with samples exceeding 250,000 discarded as measurement error. Baseline correction was then applied by subtracting the mean of the initial 10 seconds of free-air motion prior to texture contact. To maintain consistency across trials with varying scanning speeds, a one-second segment was extracted from each trial during the active texture–contact phase. Therefore, each trial was of dimension $X \in \mathbb{R}^{T \times D}$, where $T = 338$ samples, and $D = 9$ taxels. Moreover, to support real-time processing of texture discriminative spectral features, we implemented a 32-point FFT. Therefore, each processed trial was divided into 10 non-overlapping windows of length $N = 32$ samples (320 samples in total). The last 18 samples of each trial were discarded to avoid FFT computation on heavily zero-padded windows. Finally, prior to spectral analysis, the direct current (DC) component of each window was removed by subtracting the mean, centering the windowed data before the downstream FFT computations.

3.3 Proposed Stochastic FFT Architecture

We designed our architecture based on radix-2 Cooley−Tukey FFT, a multistage algorithm that computes frequency components through BUs described by:

$$X = x + W_N^k \, y, \qquad Y = x - W_N^k \, y, \tag{1}$$

$$W_N^k = e^{-j \frac{2\pi}{N} k}. \tag{2}$$

where x and y are the complex inputs to the BU, X and Y are the complex outputs of the BU, the twiddle coefficient W_N^k is determined by the stage and BU index, and N is the FFT length. The number of FFT stages is given by $log_2(N)$.

A conventional radix-2 FFT BU is illustrated in Fig. 5a. In its standard binary formulation, each BU performs one complex multiplication followed by two complex additions, which decompose into four scalar multiplications and six scalar additions, causing substantial hardware overhead as the FFT size increases. We first implemented the classical FFT BU in the unary domain (Fig. 5b). This stochastic mapping exploits the intrinsic simplicity of unary arithmetic, and the resulting BU structure is substantially simpler compared to its binary equivalent.

However, cascaded unary additions suffer from SNR degradation due to the numerical vanishing effect, i.e., values become smaller and closer to zero with more additions. To address this challenge, we optimized the BU by using one 3-input parallel counter (PC) for addition instead of two 2-input additions (Fig. 5d). This design improves SNR by reducing the number of numerical vanishing. In this work, we implemented a 32-point FFT, which requires 480 additions across all BUs if not optimized. Our optimized BU reduces the number of additions from 480 to 320, i.e., a 33.3% reduction.

3.4 Dual-Domain Dynamic Scaling

Due to the cascaded additions across stochastic FFT stages, the bitstream values can suffer from numerical saturation (values exceed legal bounds) and numerical vanishing (values are close to zero), leading to information loss and SNR degradation. More specifically, when using non-scaled additions, the sum can rapidly exceed the unary value bounds, causing data clipping and irreversible loss of numerical information. When using scaled additions, the sum will be scaled down by a fixed scale, resulting in a poor SNR due to the numerical vanishing effect.

Though we have designed optimized BUs to alleviate the numerical saturation and vanishing challenges (Section 3.3), these problems could persist at the FFT level due to the chain of cross-BU additions. To solve these problems, we introduced adaptive scaling that applies scaling selectively across BUs rather than uniformly. Specifically, we proposed a dual-domain dynamic scaling strategy in both binary and unary domains. Binary scaling is applied to the input of each FFT stage to ensure the values can fully utilize the entire legal unary value range, which benefits both numerical saturation and numerical vanishing. Unary scaling is applied to prevent numerical saturation and vanishing due to cascaded additions within each FFT stage. Together, *two scaling strategies preserve effective use of the unary dynamic range and sustain SNR throughout FFT stages*, as illustrated in Fig. 6.

1. **Binary scaling:** The objective of binary scaling is (1) to avoid excessive numerical saturation at the unary bounds that causes information loss at the FFT input, and (2) to prevent excessive numerical vanishing within an FFT stage, where unary bitstreams are so close to zero that SNR is degraded. The FFT input is adaptively scaled to the bipolar unary range $[-1, 1]$, allowing down-scaling, up-scaling, or no scaling depending on the input distribution. In contrast, inputs to subsequent FFT stages (non-first stage) are inherently bounded within the unary range and therefore require either up-scaling or no scaling to maximize effective utilization of the representable interval. More specifically, to keep a high SNR, we perform binary scaling to keep the input values of each stage close to the bipolar boundaries, rather than close to zero.

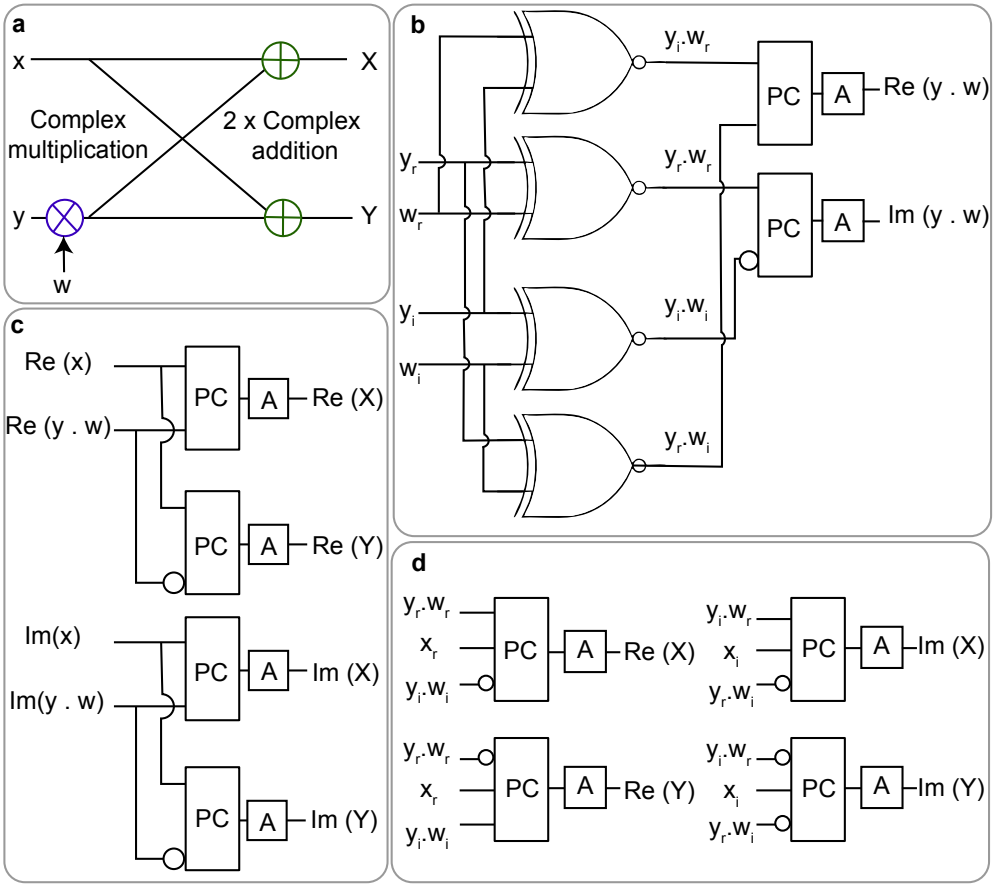


Fig. 5 Unary BU architecture in uSense. **a** Conventional complex BU block diagram. **b** Stochastic complex multiplication. **c** Stochastic complex addition/subtraction. **d** Optimized BU arithmetic operations in the uSense framework.

Fig. 6a illustrates a case where the input distribution (yellow) exceeds the unary bounds too much (beyond a user-acceptable threshold), down-scaling becomes necessary to prevent saturation and irreversible information loss. In contrast, when the binary values distribution underutilizes the unary range (Fig. 6b), applying up-scaling, subject to a controlled clipping threshold, improves representational resolution with negligible precision loss (red), which is confined to infrequent outlier samples. A direct approach is to determine the scaling factor from the maximum observed value. However, this strategy is highly sensitive to sparse outliers, leading to unnecessary down-scaling of informative samples. These unnecessary down-scalings scale the majority of informative samples toward zero, distorting spectral structure and degrading the input SNR of subsequent stages. To address this limitation, we adopt a quantile-guided dynamic scaling strategy. As detailed in Alg. 1

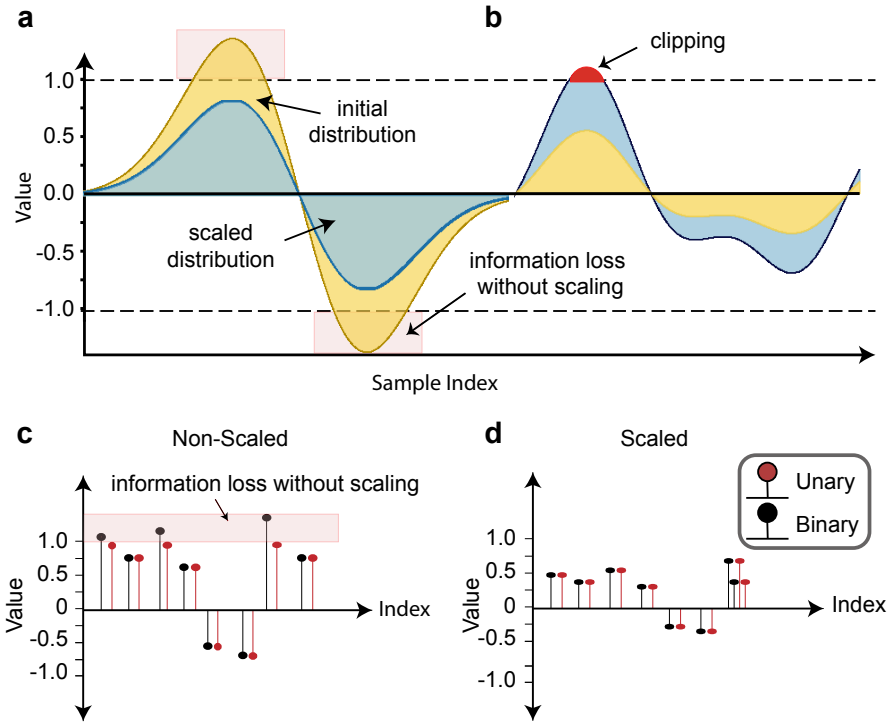


Fig. 6 Dual-domain dynamic scaling benefits both numerical saturation and vanishing. **a** Binary scaling for numerical saturation: initial distribution of the FFT binary input samples (yellow) exceeding the unary bounds too much (pink) is scaled down to be properly represented within the valid range, avoiding information loss. **b** Binary scaling for numerical vanishing: binary samples that underutilize the unary range is scaled up (blue), improving representational resolution with negligible information loss (red). **c** Without unary scaling, portions of the binary input values that exceed the unary limits lead to saturation of the encoded unary values and irreversible information loss. **d** With appropriate unary scaling, accumulated intermediate values are constrained within the unary boundaries, preserving information.

(lines 3–4), the binary scaling factor is derived from the ρ-quantile of the input distribution. Experiments indicate that the 95^{th} percentile as the clipping threshold effectively prevents extreme outliers from dominating the scaling decision B_{scale}.

2. **Unary scaling:** The objective of unary scaling is to avoid excessive numerical saturation and vanishing during unary computing within each FFT stage. A fixed scaling factor in unary addition is ill-posed for a multi-stage unary FFT pipeline. Applying a constant scaling factor $s < 1$ to all N stages with each including one level of addition, the signal amplitude decays rapidly ($\propto s^N$), collapsing toward zero (numerical vanishing). In contrast, $s = 1$ (no scaling) may cause the amplitude to accumulate rapidly over additions, eventually causing numerical saturation. As a result, neither "always down-scale" nor "never scale" preserves the optimal dynamic range across stages, i.e., fully utilizing the legal unary value range. Fig. 6c illustrates the case without using a unary scaling factor, the sum over cascaded additions exceeding unary bounds causes numerical saturation and information loss. Fig. 6d

Algorithm 1 Dual-Domain Dynamic Scaling

1: **Input:** $\mathcal{D}$: input binary distribution; ρ: quantile value; (B_o, U_o): binary and unary outputs; ε: per-sample error threshold; τ: MAE threshold.
2: **Output:** $B_{scale},\ U_{scale}$
3: $Q_\rho \leftarrow \text{Quantile}(\mathcal{D}, \rho)$
4: $B_{scale} \leftarrow 2^{-\lceil \log_2(Q_\rho) \rceil}$ ▷ Binary scale
5: $E \leftarrow \text{MAE}\left(\{(B_o^i, U_o^i) \mid |B_o^i - U_o^i| > \varepsilon\}\right)$ ▷ MAE
6: Initialize $U_{scale} \leftarrow 1$
7: **while** $(E > \tau)$ **and** $(\ \exists i,\ B_o^i > 1)$ **do** ▷ i: index
8: $\quad U_{scale} \leftarrow U_{scale} + 1$ ▷ Unary scale
9: **end while**
10: **return** $\{B_{scale}, U_{scale}\}$

shows that using appropriate adaptive unary scaling preserves values within the unary range and prevents information loss.

The unary dynamic scaling is detailed in Alg. 1 (lines 5-9). We start each FFT stage with a unary scaling factor ($U_{scale} = 1$). Next, we compute the mean absolute error (MAE) of the FFT window only over samples that have binary-unary absolute error $\varepsilon > 0.05$. A scale-down update is then triggered when MAE > 0.2, thereby preventing adjustments driven by trivial fluctuations. The unary scaling factor is selected from a bounded set that avoids excessive search of the scaling factor at each stage. Importantly, deciding the bounded set is not part of the scaling algorithm itself, but rather task dependent. To obtain the bounded set, we first conduct a pilot analysis to identify suitable scaling factors on a representative subset of the full dataset acquired from our tactile sensors for texture recognition. We then use unrestricted scaling factors (1-8) for all stages and record the frequency of each factor per stage on the subset. The analysis shows that stages 1-4 predominantly utilize scaling factors ≤ 2, whereas stage 5 exhibits a non-negligible use of larger scaling factors. Finally, we restrict the candidate scaling set to $\{1, 2\}$ for stages 1-4 and $\{1, 2, 3, 4\}$ for stage 5. This set stabilizes the unary dynamic range, reduces the search space for scaling factors for each input, and bounds the information loss and SNR degradation due to numerical saturation and vanishing.

3.5 Stage-Wise Bitwidth Optimization

To mitigate the challenge of low latency tied to numerical inaccuracy, we introduce stage-wise bitwidth optimization to reduce the bitwidth of the outputs at each FFT stage, while keeping the decoding accuracy loss minimal. The benefits are two-fold: First, lowering output bitwidth reduces the unary bitstream length, effectively early terminating the computation to achieve lower latency[43,54,66]. Early termination is achieved by observing the progressive precision (PP) and stability properties of bitstreams[43,66]. PP, introduced by Alaghi et al.[66], defines the numerical precision at a certain bitstream length, i.e., indicating a trade-off between accuracy and latency. Furthermore, stability, introduced by Wu et al.[43,54], measures how early a bitstream PP converges to a user-acceptable threshold, so that early termination can be done

in a predictive manner. Second, lowering the output bitwidth compresses the data effectively, and reduces the pressure on memory and communication bandwidth. With bitwidth reduced to k bits, the data is compressed by $k/32$ compared to FP32 format in conventional systems. This leads to a proportional reduction in memory footprint and communication bandwidth.

To perform the optimization, the key is to understand that numerical accuracy does not necessarily equal decoding accuracy. *As long as the feature space with reduced bitwidth is numerically separable by the classifier, decoding accuracy can be maintained.* Since the last stage interacts directly with the classifier (Fig. 3), prioritizing its bitwidth reduction ensures robust decoding before propagating bitwidth reduction into earlier stages. We determines the lowest bitwidth k of the final FFT stage without decoding accuracy loss, leading to k/32 data compression. Next, we conduct intermediate-stage bitwidth optimization. With the final stage bitwidth fixed to k, we exhaustively evaluated all bitwidth combinations $\{5, 4, 3, 2\}$ across the remaining four stages. We select the lowest bitwidth for each stage without degrading accuracy, resulting in substantial latency and efficiency improvements.

3.6 Computational Graph Pruning

In addition to bitwidth optimization, which compresses the data and reduces computation cost, we further introduce a second level of reduction derived from the spectral importance of the tactile signals[27]. Specifically, we determine a task-relevant frequency band of interest and retain only the FFT frequency bins corresponding to this range. Given a sampling rate F_s and an N-point FFT, the frequency resolution is defined as $\Delta f = F_s/N$. The center frequency of bin k is given by:

$$f_k = k \cdot \Delta f \tag{3}$$

We define the target set of bins Ω by mapping the frequency band of interest to the FFT discrete bin indices as follows:

$$\Omega = \{\, k \in \{0, \ldots, N-1\} \mid f_{\min} \leq f_k \leq f_{\max} \,\} \tag{4}$$

We prune the radix-2 FFT by representing it as a directed acyclic graph (DAG), where each node corresponds to an intermediate FFT value indexed by stage and frequency index. Specifically, a node (s, k) represents the *k-th* output at stage s of the FFT, with stages $s = 1, \ldots, L$ and FFT of length $N = 2^L$. Directed edges represent the BU computational dependencies between stages, such that each node at stage s depends only on two nodes at stage $s - 1$. This construction results in a DAG that captures the dataflow of the radix-2 Cooley–Tukey FFT. Alg. 2 details the pruning strategy, by tracking the parent nodes of Ω through backward propagation within the DAG. Starting from the nodes (L, k) for $k \in \Omega$, only the edges in the resulting active subgraph are computed. Fig. 7 illustrates FFT pruning, where the nodes and edges highlighted in red correspond to the exact set of computations required to obtain the targeted output frequencies.

As a result, the number of active nodes scales linearly with $|\Omega|$ across $L = \log_2 N$ stages, reducing the computational complexity from $\mathcal{O}(N \log N)$ to $\mathcal{O}(|\Omega| \log N)$, where $|\Omega|$ denotes the cardinality of the set Ω. In addition, pruning reduces data transmission and storage requirements and simplifies the downstream classifier due to fewer inputs, further leaning the tactile sensing system.

Algorithm 2 Computational Graph Pruning for Radix-2 FFT

Input: $L = \log_2(N)$, target bins Ω
Output: $Y[k]$ for all $k \in \Omega$ ▷ FFT output frequency bin at index k
$A_L \leftarrow \{(L, k) \mid k \in \Omega\}$ ▷ Initialize active set
for $s = L, \ldots, 1$ **do**
    $A_{s-1} \leftarrow \text{parent}(A_s)$ ▷ Track parent nodes of active nodes
end for
$A \leftarrow \bigcup_{s=0}^{L} A_s$ ▷ Identify all active nodes
Execute the computational graph A
return $Y[k]$ for $k \in \Omega$

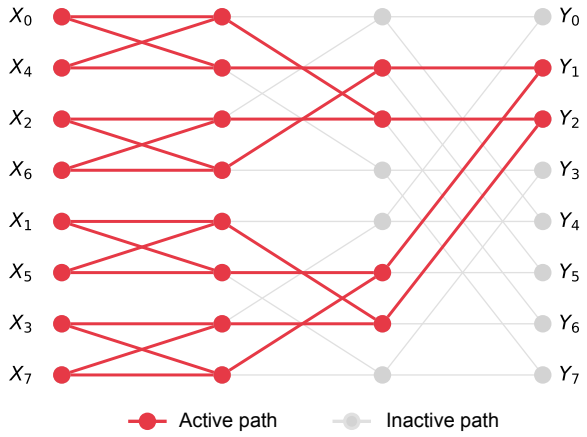


Fig. 7 Pruned 8-point FFT example. Grey nodes indicate memory saved by pruning, and grey edges represent the corresponding saved computations.

3.7 Evaluation Methodology

3.7.1 Setup

We evaluated our uSense framework using an edge-optimized tactile classification pipeline. For each windowed 32-sample (Section 3.2), we computed a pruned stochastic radix-2 FFT (Section 3.6), and the resulting magnitude spectrum served as the feature set for the classifier. An SVM with a Radial Basis Function (RBF) kernel is used as the

decoder, and the model performance is evaluated using Group K-Fold cross-validation with 5 folds. Classification accuracy is reported as mean ± inter-fold variability over Group 5-fold cross-validation across unseen speed–force conditions. The choice of the evaluation is to prevent data leakage by ensuring that all segments from a given trial are either in the training or test set. This setup provides a more realistic estimate of model generalization and validates the decoder's speed–force invariance.

3.7.2 Metrics for Data Compression

We define compression in terms of both numerical precision and pruning reductions. Let b_i denote the numerical precision (in bits) used at processing stage i, and let N be the total number of stages. The effective precision of a compressed configuration, with all stages considered, is defined as

$$B_{\text{comp}} = \sum_{i=1}^{N} b_i. \tag{5}$$

Given a reference configuration with effective precision B_{ref}, the precision-based compression rate is defined as

$$\text{CR}_{\text{prec}} = \frac{B_{\text{ref}}}{B_{\text{comp}}} \tag{6}$$

and the corresponding compression efficiency (CE) is

$$\text{CE}_{\text{prec}} = \left(1 - \frac{B_{\text{comp}}}{B_{\text{ref}}}\right) \times 100\ \%. \tag{7}$$

In addition to precision reduction, structural compression is achieved by retaining only a subset of output coefficients. Let K_{ref} denote the number of output coefficients in the reference configuration, and let K_{comp} denote the number of retained coefficients after pruning. The total compression efficiency, accounting for both precision reduction and coefficient pruning, is defined as

$$\text{CE}_{\text{total}} = \left(1 - \frac{B_{\text{comp}}\, K_{\text{comp}}}{B_{\text{ref}}\, K_{\text{ref}}}\right) \times 100\ \%. \tag{8}$$

All compression ratios and efficiencies reported throughout this manuscript are computed using these definitions.

4 Experimental Results

We evaluated uSense to answer the following four key questions:

- What is the optimal design point that achieves the best trade-off between data compression and decoding accuracy?

- How each of the three optimization techniques constructs the full design space and contributes to the final optimality?
- How well the aggressive compression framework preserves the classifier decision boundary and class separability?
- What are the projected gains for area and energy at the hardware level after all these optimizations are applied?

4.1 Optimal Data Compression and Decoding Accuracy

After all optimizations are properly configured, we obtain the optimal design point of uSense as using the bitwidth of $(2, 3, 3, 3, 3)$ for stages $(1 \rightarrow 5)$ and using the first three non-zero frequency components as the band of interest. With this, the data compression rate is 99.18%, compared to the FP32 binary baseline. Furthermore, uSense achieves 97.64%$\pm$0.61 decoding accuracy, corresponding to only a 1.35% reduction relative to the FP32 binary baseline (98.99%$\pm$0.35). Furthermore, both the computation and memory footprint are reduced by 51.9%.

4.2 Analysis of Three Optimizations

This section explains how we obtain the optimal design point, as well as diving deeper into how sensitive compression and accuracy are to each optimization.

4.2.1 Dual-Domain Dynamic Scaling

Dual-domain dynamic scaling retains information and SNR by avoiding excessive numerical saturation and vanishing via both binary and unary scaling. To evaluate the impact of each scaling, we conduct an ablation study using the classifier configuration described in Section 3.7.1 and the optimized bitwidth setting. We evaluated all four possible configurations of the two scaling: both applied, neither applied, and either applied (binary-only and unary-only), with the decoding accuracy shown in Table 1.

Table 1 Ablation study of decoding accuracy for dual-domain dynamic scaling.

Scaling method	**Binary + Unary**	**No scaling**	**Binary-only**	**Unary-only**
Accuracy (mean $\pm$ std)	97.64 ± 0.61	90.50 ± 0.57	86.51 ± 1.34	94.23 ± 2.33

With both scaling applied, the classifier achieves the best accuracy among all. With both scaling removed, the decoding accuracy is 7.14% lower than the case with both scaling applied. Removing unary scaling (binary-only) degrades the accuracy by 11.13% reduction from the optimal accuracy, hinting at the criticality of unary scaling for preserving the discriminative structure of signals. In contrast, removing binary scaling (unary-only) results in a 3.41 accuracy drop. Since raw tactile signals typically occupy a wider numerical range than the legal unary range [-1,1], binary scaling can properly adjust the numbers towards [-1,1], suppressing the likelihood of numerical saturation and vanishing. These results reveal that two scaling methods collaboratively

and synergistically contribute to uSense's superior decoding accuracy, as well as the decoding stability (lower standard deviation).

In addition to the decoding accuracy, we also want to understand how the numerical accuracy is impacted by two scaling methods. We randomly sampled 32-point inputs from two uniform distributions: $U(-2, 2)$ and $U(-0.05, 0.05)$. The former represents values exceeding unary range (numerical saturation), and the latter represents low-magnitude values within unary range (numerical vanishing). For each range, we examine the stage-1 output of an 8-bit unary FFT under four configurations, identical to the ablation study of decoding accuracy: both applied, neither applied, and either applied (binary-only and unary-only), with results given in Fig. 8.

When inputs exceed the unary range $U(-2, 2)$, unary-only scaling results in clipping (Fig. 8 a), whereas binary-only scaling and no scaling lead to overflow during addition (Fig. 8 e,f). In contrast, applying both scaling preserves dynamic range and closely matches FP32 (Fig. 8 b). For low-magnitude inputs $U(-0.05, 0.05)$, unary-only scaling maintains bounded values but exhibits reduced numerical accuracy (Fig. 8 c). Binary-only scaling properly adjusts the input ranges but causes overflow due to missing unary scaling (Fig. 8 g). Although applying no scaling incurs only slight numerical accuracy loss in this scenario (Fig. 8 h), it lacks adaptive range control, which potentially degrades the SNR. In contrast, applying both scaling consistently preserves numerical accuracy (Fig. 8 d). These results demonstrate that binary and unary scaling address complementary numerical limitations independently, and together they maintain high representability during FFT.

Overall, even at high bitwidth, applying no scaling introduces significant numerical inaccuracy; with our dual-domain dynamic scaling, even at low bitwidth, uSense achieves high decoding accuracy.

4.2.2 Stage-Wise Bitwidth Optimization

Stage-wise bitwidth optimization reduces the bitwidth of all FFT stages to improve the processing latency, data compression, while preserving decoding accuracy. We perform multi-step analysis to understand how bitwidth impacts decoding accuracy and converge to the final choice in Section 4.1.

First, we aim to narrow down the scope of bitwidth choices by uniformly reducing the bitwidth of all FFT stages from 32 bits. We show the accuracy by sweeping bitwidth from 2 to 8 bits in Fig. 9. It is observed that decoding accuracy increases rapidly between 2 and 4 bits and then saturates. At 2 bits, aggressive bitwidth reduction leads to a sharp accuracy drop. Whereas at 6 bits, uSense achieves $99.19 \pm 0.71\%$, slightly exceeding the binary baseline ($98.99 \pm 0.35\%$), while offering an 81.25% compression rate relative to the FP32 baseline. We therefore adopt 6-bit precision as a *reference* for the following analyses.

Next, we analyze the stage-wise sensitivity to bitwidth. Here, the sensitivity is evaluated progressively across stages by selecting one target stage at a time and varying its bitwidth $\{6, 5, 4, 3, 2\}$ while keeping the remaining stages fixed at 6 bits. The result is given in Fig. 10, and we observe a clear non-uniform sensitivity profile. In particular, the output FFT stage exhibits a higher sensitivity to bitwidth reduction, whose decoding accuracy drops significantly at 2 bits. In contrast, the remaining stages

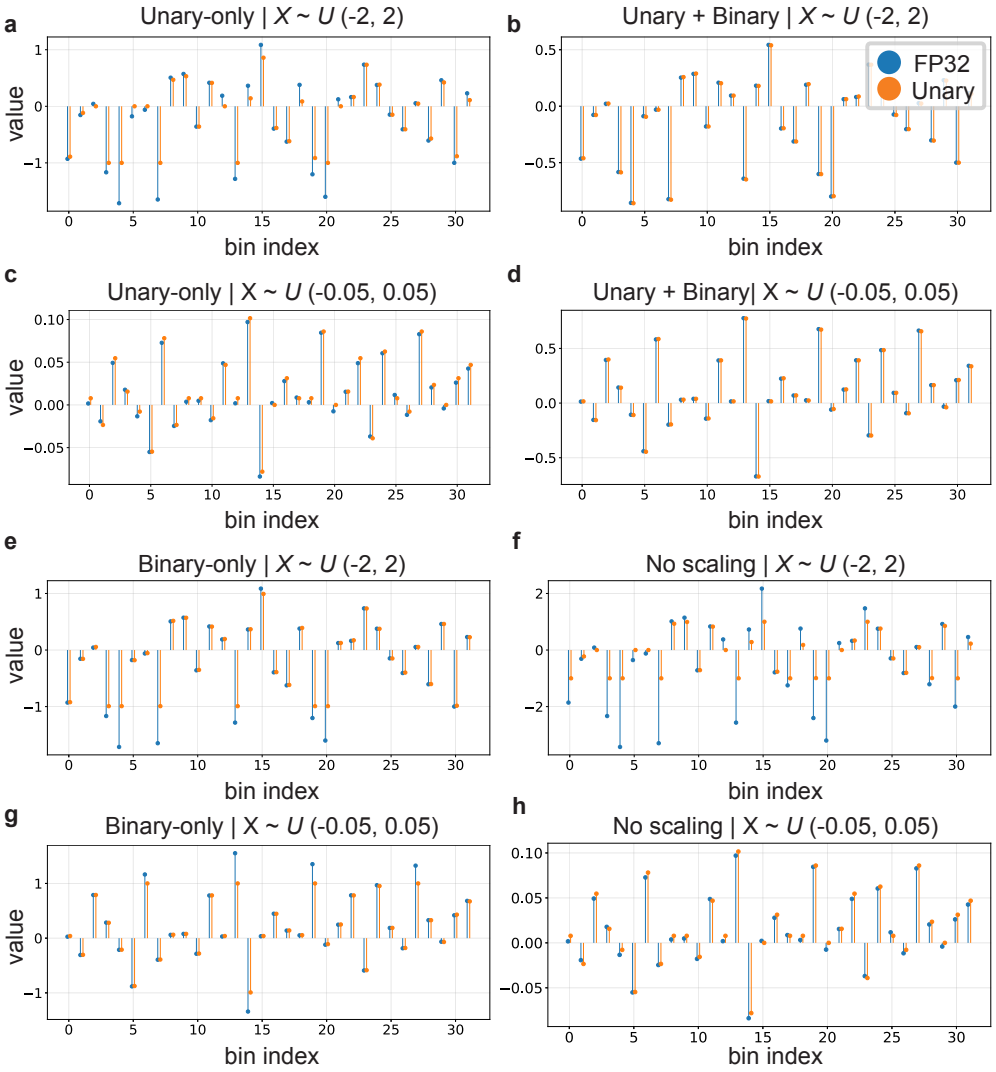


Fig. 8 Binary and unary scaling preserves the numerical accuracy of intermediate FFT data with different input ranges. The input X is randomly sampled from a uniform distribution. Results correspond to 8-bit unary computation and FP32 binary computation, demonstrating the impact of scaling on the stage-1 output of a 32-point FFT. (a,b,e,f) shows that inputs drawn from $U(-2, 2)$ exceed unary bounds. Unary-only scaling (a) causes clipping at $[-1, 1]$. Binary-only scaling (e) and no scaling (f) cause overflow during accumulation. The combined scaling (b) preserves numerical accuracy. (c,d,g,h) shows that inputs drawn from $U(-0.05, 0.05)$ lie within unary bounds. Unary-only scaling (c) shows a slight reduction in precision at small magnitudes. Binary-only scaling (g) ensures data representation in unary bounds yet induces overflow after accumulation. Without scaling (h), outputs remain bounded but show minimal reduction in precision, similar to (c). The combined scaling (d) preserves numerical accuracy.

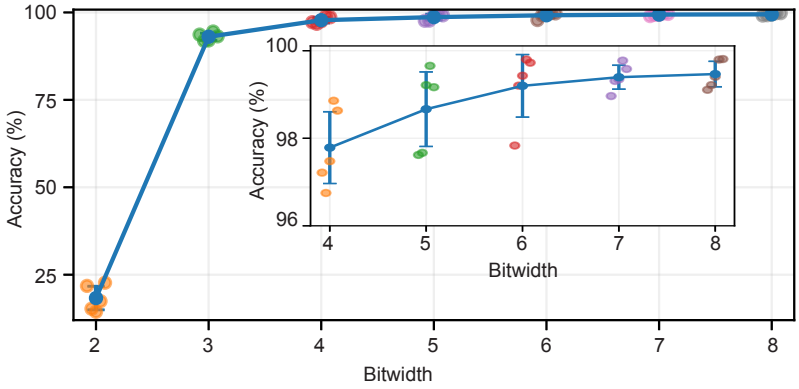


Fig. 9 Impact of uniform bitwidth on decoding accuracy. A 6-bit unary representation achieves lossless accuracy with over 80% data compression relative to the FP32 binary baseline. Decoding accuracy increases rapidly from 2 to 4 bits and saturates beyond 6 bits, with 6-bit precision slightly outperforming the binary baseline. Individual markers indicate accuracies of a 5-fold group cross-validation, and the blue curve shows the mean accuracy with error bars indicating inter-fold variability.

demonstrated higher tolerance to reduced precision while maintaining high accuracy even at lower bitwidths.

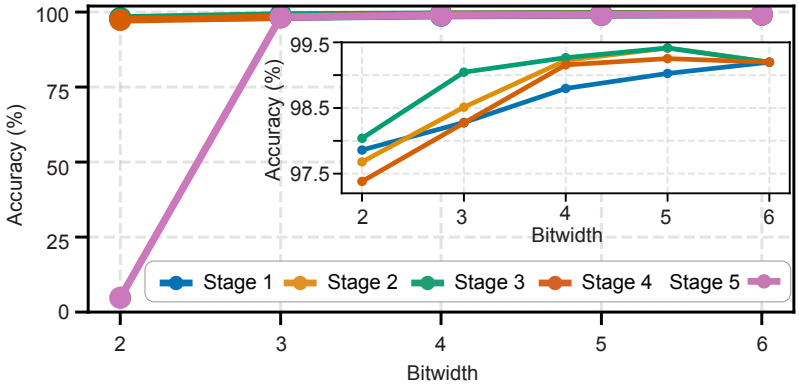


Fig. 10 Impact of stage-wise bitwidth to decoding accuracy. Each data point represents the average accuracy of 5-fold group cross-validation when the bitwidth for a single FFT stage is reduced while all other stages are fixed at 6 bits.

Then we jointly optimize the first and final FFT stages for both their distinct roles in information flow and tolerance to bitwidth reduction. The first stage operates directly on the raw tactile signal, and the impact of bitwidth reduction will propagate through all subsequent stages. Moreover, the bitwidth of the first stage will impact the cost of sensor resolution, which is often one of the most costly hardware components. The final stage outputs the compressed spectral features that serve as the classifier input, directly impacting the decoding accuracy. We must minimize its bitwidth to lower the communication cost between FFT and SVM, while maintaining the accuracy. The result is shown in Fig. 11. The first stage 1 exhibits a smooth trend that stabilizes

near 2 bits, whereas the final stage 5 requires at least 3 bits to maintain high decoding accuracy. The joint bitwidth reduction corroborates the findings in Fig. 10. Setting the first stage to 2 bits and the final stage to 3 bits results in only a 1.35% reduction in accuracy relative to the FP32 reference, demonstrating that aggressive precision reduction can be applied selectively without compromising decoding accuracy.

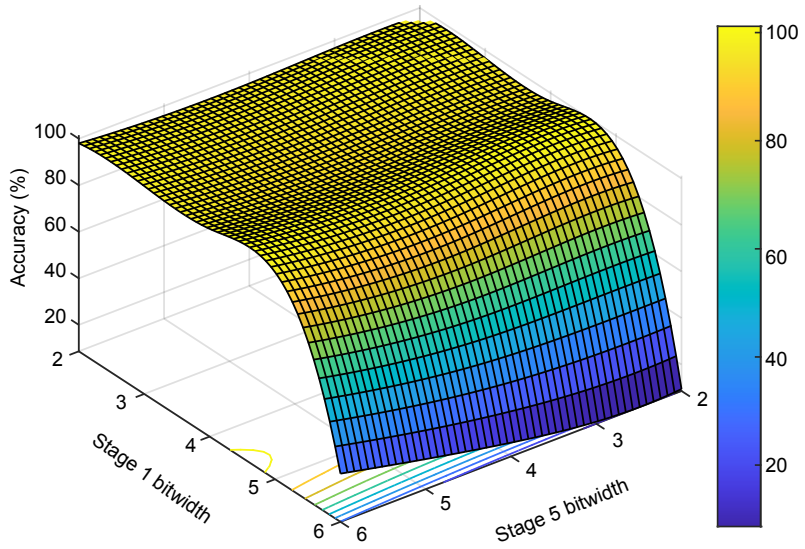


Fig. 11 Impact of bitwidth for stages 1 and 5. The surface shows classification accuracy as a function of the unary bitwidth assigned to each stage, with all other stages assigned 6 bits. Stage 5 exhibits strong sensitivity, with accuracy dropping steeply below 3 bits, whereas stage 1 shows a smoother trend that stabilizes around 2 bits.

Finally, we tune the intermediate stages, with results shown in Fig. 12. The orange paths are of particular interest due to their high decoding accuracy, rather than the blue and gray paths. Among all orange paths, a higher bitwidth in stage 1, e.g., 4 or 5 bits, commonly yields better decoding accuracy. However, this comes at the cost of longer processing latency and higher cost in sensor hardware. We observe that stages 2-4 can be reduced to 3 bits with negligible loss in decoding accuracy, i.e., larger than 97.5%. Based on this analysis, we set an optimized configuration of $(2, 3, 3, 3, 3)$ across the five stages, maximizing data compression (91.25% over FP32 baseline) and hardware efficiency while preserving high decoding accuracy ($97.64\% \pm 0.61$).

4.2.3 Computational Graph Pruning

Computational graph pruning will further reduce the computation and improve data compression while preserving decoding accuracy, due to the fact that only a subset of frequency components contribute to texture discrimination. Fig. 13 illustrates a representative taxel signal in time and frequency domains, where discriminative components reside in the low-frequency band. We observe that the majority of tactile signal energy during texture contact is concentrated below 35 Hz, which corresponds to only

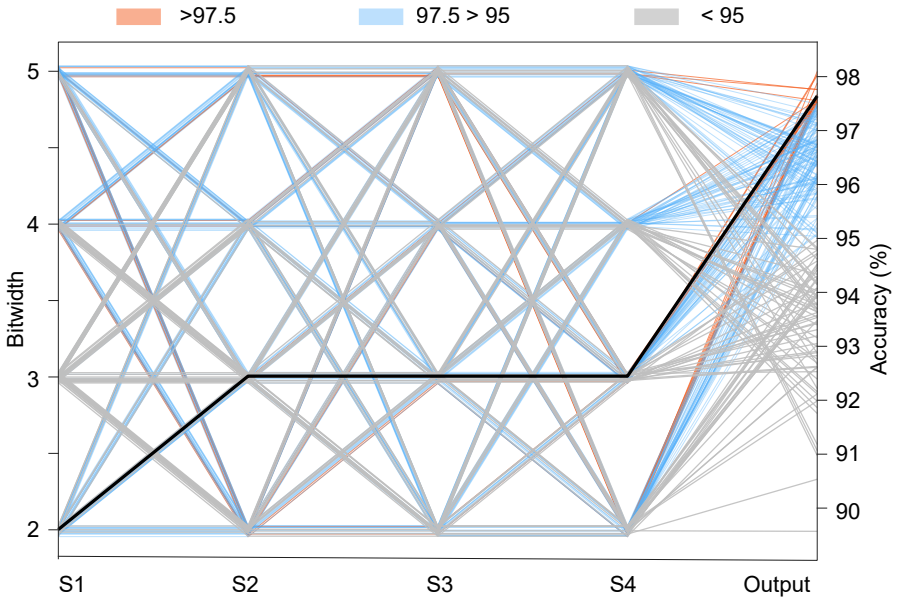


Fig. 12 Impact of bitwidth for intermediate stages 2-4. Based on the sensitivity analysis of stages 1 and 5, which indicated stable performance at 2 and 3 bits, respectively 11. Intermediate stages are tuned to reduce latency while maintaining accuracy. Each configuration path represents bitwidth assignment across the first four stages. The black path marks the final bitwidth configuration considered as optimal.

three frequency bins in a 32-point FFT output. According to Eq. 4 in Section. 3.6, we set $f_{min} = 1$ and $f_{max} = 35$, defining the target discrete frequency set $\Omega \in \{1, 2, 3\}$ excluding the DC component.

Then applying our DAG-based pruning strategy (Alg. 2) to Ω shows that only 77 of 160 FFT nodes (memory) and 154 of 320 edges (computation) are required for texture recognition. Overall, pruning resulted in a compression rate of 90.6% of the calculated frequency bins, which is on top of the data compression from bitwidth reduction. Also, pruning allows a 51.9% reduction in both computation and memory footprint. Furthermore, only 12 of the 32 twiddle factors (defined in Section 3.3) are retained, resulting in an additional 61.3% reduction in twiddle factor computation and memory.

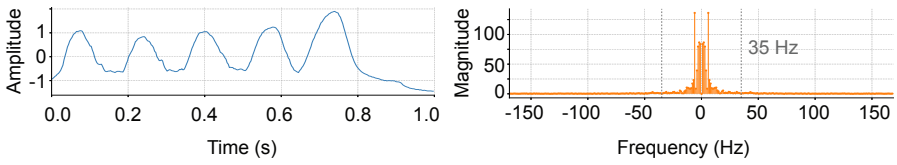


Fig. 13 Texture energy is concentrated below 35 Hz. (left) Tactile signal in the time domain obtained from a single taxel for a hemispherical texture, recorded at a sliding speed of 2400 mm/s and a force of 1.5. (right) Magnitude spectrum of the signal, highlighting discriminative frequency content within the low-frequency band.

4.3 Decision Boundary and Latency Space

Such aggressive data compression brings benefits of 99.18% data compression, two orders of magnitude fewer data in uSense compared to the FP32 baseline. However, it was still unclear whether:

- the numerical accuracy is preserved after full FFT computation;
- the decision boundary is preserved to ensure high class separability;

Therefore, in following we examine these two questions.

4.3.1 Mismatch in Numerical Accuracy and Decoding Accuracy

The numerical accuracy of different FFT implementations is given in Fig. 14. We observed that 8-bit unary representation can successfully track the numerical accuracy of the binary baseline for all real, imaginary and magnitude components. However, we found that a 3-bit representation cannot track the numerical accuracy of the binary baseline. This observation indicates a mismatch between numerical accuracy and decoding accuracy. In the conventional view where numerical accuracy matters a lot, this observation is counterintuitive as how it is possible to have high decoding accuracy if the numerical accuracy is reduced.

4.3.2 Well-Preserved Decision Boundary and Closely-Aligned Latency Space

To unveil the reason for the mismatch between numerical accuracy and decoding accuracy, we visualized the decision boundary in the form of a confusion matrix, as shown in Fig. 15. In both cases, a clear diagonal structure is observed, indicating consistent class separation despite the severe bitwidth reduction in uSense. This suggests that task-relevant discriminative information is preserved even when signal-level reconstructability is compromised (Fig. 14).

In addition, to show the robustness in the decision boundary, we further examined the actual latent space of spectral features of the binary baseline and uSense. More specifically, we tested whether compressed unary features preserve the class-discriminative structure of binary FFT features in a shared latent space. Therefore, we align the feature representations using Multiset Canonical Correlation Analysis (MCCA) and visualize the aligned features in the shared latent space using t-SNE. Fig. 16 shows t-SNE visualizations of the feature distributions for each texture class obtained from binary and uSense outputs. Despite substantial differences in the feature space, the projected, aligned features exhibit strong class-wise correspondence and a similar cluster organization across both domains. This indicates that uSense preserves the task-relevant latent features structure required for texture recognition.

We further quantified representational compatibility by cross-domain generalization. We found that an SVM trained on binary features achieves 98.99% accuracy when evaluated on unary features in the shared space, indicating that the decision boundary learned in the binary domain transfers to the compressed unary domain. These results explain why uSense maintains high decoding accuracy under aggressive data compression: while signal fidelity (numerical accuracy) is reduced, the task-relevant

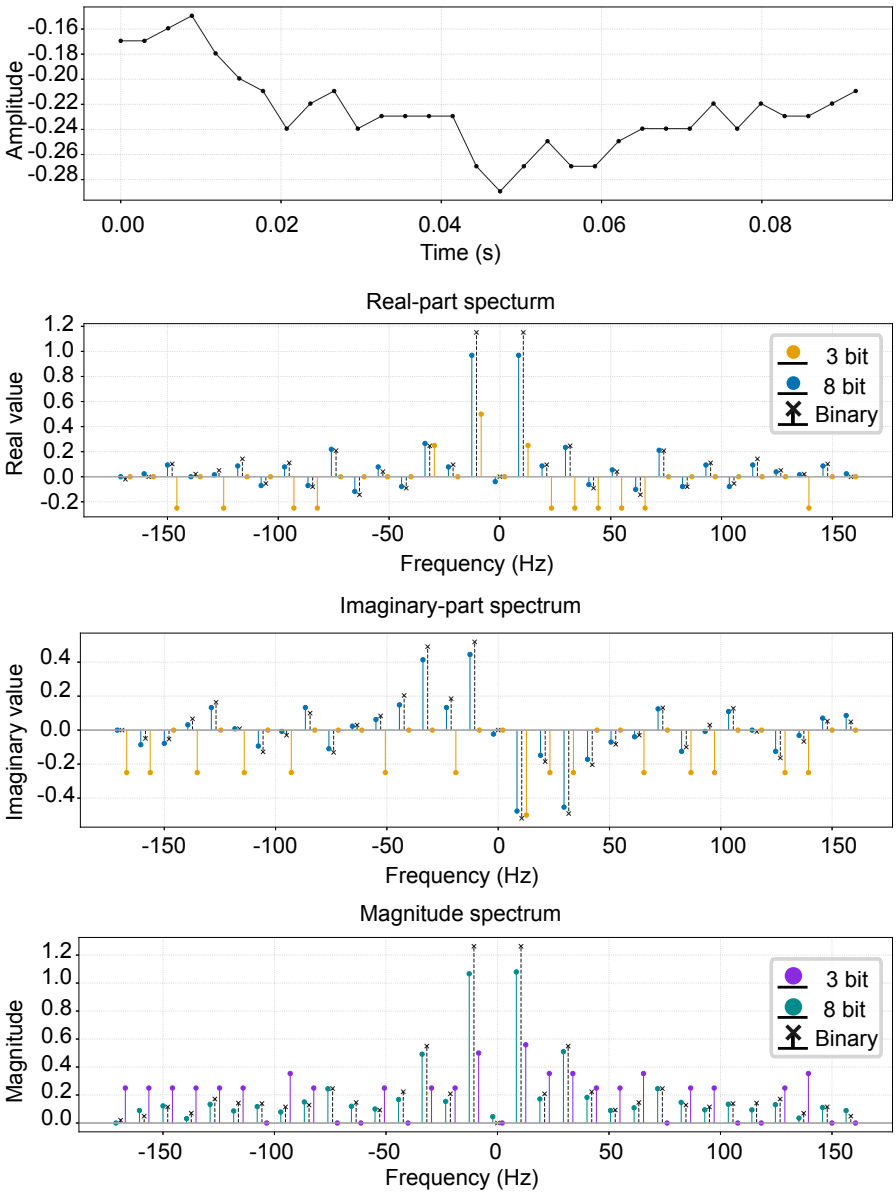


Fig. 14 Preservation of numerical accuracy in real, imaginary, and magnitude components, compared to FP32 binary baseline. 8-bit unary representation faithfully preserves the relative spectral fidelity of the binary format, while the 3-bit representation fails.

latent structure (decoding accuracy) is well preserved. More generally, *high numerical inaccuracy does not necessarily translate to low decoding inaccuracy, as long as the structure of the feature latent space is preserved.* This fact will unlock aggressive optimizations of algorithms and hardware in edge AI systems.

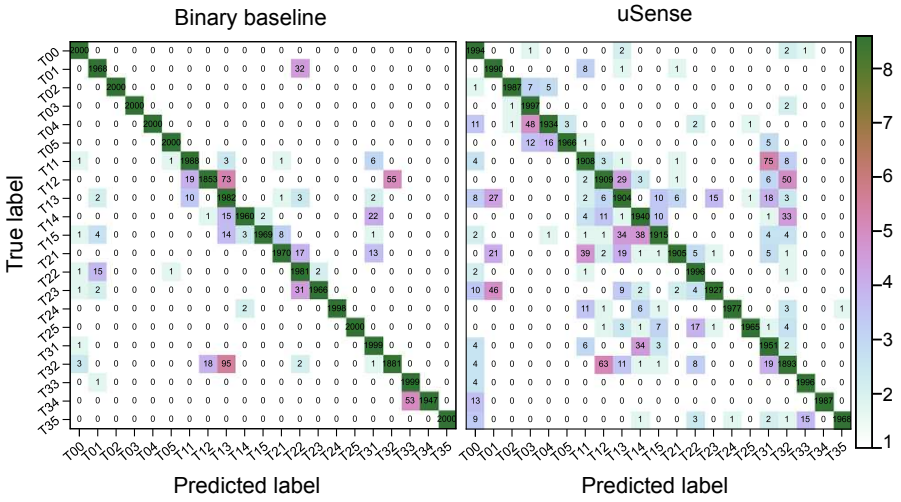


Fig. 15 Visualization of decision boundary in the form of a confusion matrix. Results are averaged over 5-fold cross-validation across unseen speed–force combinations. Confusion matrix for SVM trained on binary data (left). Confusion matrix for SVM trained on compressed data using uSense (right). Texture labels T_{XY} denote texture class, where X indicates the texture type and Y denotes the bump density. Color intensity corresponds to $\ln(1+n)$, where n denotes the prediction count.

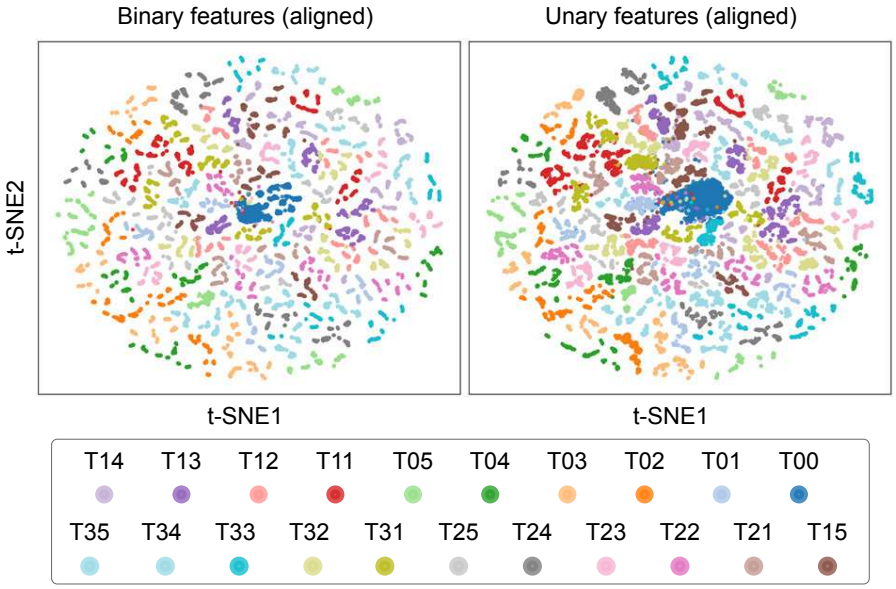


Fig. 16 Ultra-compressed unary representations share a latent space that preserves texture-discriminative information, with that of 32-bit binary representations. t-SNE plots for binary FFT-extracted features (left) and uSense FFT outputs (right) show similar feature distributions across 21 textures.

4.4 Projected Area and Energy

Though this paper primarily focused on algorithmic innovation, we additionally conducted back-of-the-envelope analysis on the cost of hardware implementation and project the potential gains in area and energy of uSense, both of which are dominant constraints for wearable, embedded, and near-sensor edge AI systems. We compare uSense against a conventional binary radix-2 implementation of a 32-point FFT. Both binary and unary implementations are assumed to be fully flattened (that is, each BU has a dedicated physical module) to isolate the impact of varying hardware optimizations. Hardware area, power, and energy are estimated using a component-level analytical cost model derived from published post-synthesis hardware characterizations under 32nm CMOS technology and clock frequency $f_{\text{clk}} = 400$ MHz[67–69].

4.4.1 Area

We first evaluated silicon area to quantify the hardware footprint of uSense. Total area is computed as the sum of the areas of all instantiated arithmetic units. For the binary FFT, this yields

$$A_{\text{binary}} = 480\, A_{\text{adder}} + 320\, A_{\text{multiplier}} = 6.627 \text{ mm}^2. \tag{9}$$

In contrast, the uSense FFT replaces multi-bit arithmetic with unary accumulators and XNOR-based multipliers, resulting in

$$A_{uSense} = 154\, A_{\text{adder}} + 212\, A_{\text{multiplier}} = 0.017 \text{ mm}^2. \tag{10}$$

The binary design occupies 6.627 mm^2, corresponding to 480 adders (0.007779 mm^2 each) and 320 multipliers (0.009039 mm^2 each). The uSense implementation, by comparison, requires only 0.017 mm^2, comprising 154 unary accumulators (9.75×10^{-4} mm^2 each) and 212 unary multipliers (4.45×10^{-6} mm^2 each)[67–69]. This corresponds to a 99.74% reduction in total area compared to the binary baseline, highlighting the substantial circuit simplification of uSense.

4.4.2 Energy

While area reduction is critical, energy efficiency ultimately controls deployability in battery-powered near-sensor and wearable technologies. Therefore, we quantify the energy consumed per FFT execution. Total power is computed as the sum of dynamic and leakage power across all active arithmetic units, and energy is obtained by accounting for the latency in clock cycles:

$$E_{\text{FFT}} = \frac{\sum_i N_i \left(P_i^{\text{dyn}} + P_i^{\text{leak}}\right) \cdot \text{cycles}}{f_{\text{clk}}} \tag{11}$$

where N_i denotes the number of instances of arithmetic unit i. The binary FFT completes in five clock cycles (number of FFT stages), corresponding to the number of stages. In contrast, the unary FFT incurs additional latency due to the temporal

accumulation of bitstreams, with the latency being 2^{bitwidth}. Under the optimized stage-wise bitwidth configuration $(2, 3, 3, 3, 3)$, the unary FFT latency is 36 cycles.

Despite this increase in latency, the unary design exhibits substantially lower power per cycle, resulting in a total energy of

$$E_{\text{binary}} = 3.754 \times 10^{-6}\ \mu\text{J}, \qquad E_{uSense} = 0.139 \times 10^{-6}\ \mu\text{J}. \tag{12}$$

This corresponds to a 96.3%, or 27× reduction in energy per 32-point FFT.

4.4.3 System-Level Implications

We next examined how our optimizations translate into end-to-end system-level trade-offs. Fig. 17 summarizes the system-level comparison between binary FFT and uSense across accuracy, throughput, area efficiency, and energy efficiency. Area and energy efficiencies are defined as the inverse of total area and energy per FFT, respectively, while throughput is defined as the inverse of latency. The radar plot highlights uSense 's pronounced advantages in area and energy efficiency while maintaining near-lossless classification accuracy relative to the binary baseline. Although this efficiency gain is accompanied by a moderate throughput reduction due to unary temporal accumulation, this trade-off is well aligned with ultra-low-power and resource-constrained edge applications.

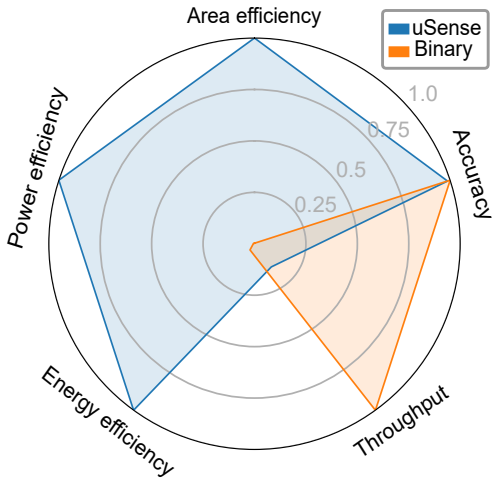


Fig. 17 Performance–efficiency trade-off between binary FFT and uSense. System-level comparison between the proposed uSense FFT and a conventional binary FFT. All metrics are normalized such that larger values indicate better performance. Area efficiency is computed as the inverse of total hardware area, energy efficiency as the inverse of energy per 32-point FFT, and throughput as the inverse of latency in cycles. Accuracy is reported as the average 5-fold group cross-validation accuracy over unseen speed–force conditions for texture recognition.

A quantitative summary of area, energy, latency, and accuracy is reported in Table 2. Despite operating at a 7.2× higher latency, the absolute number of cycles

remains small and does not violate real-time constraints (90ns latency for uSense). Though not studied in this work, the simplicity of unary circuits allows higher achievable clock frequencies, which can partially compensate for the increased cycle count. Overall, uSense achieves an order-of-magnitude reduction in energy and nearly two orders of magnitude reduction in area, while preserving high decoding accuracy, making it a compelling spectral processing pipeline for lightweight tactile sensing and broadly low-power edge AI systems.

Table 2 Comparison of hardware costs and efficiency for a 32-point FFT. The gain is comparing uSense to FP32.

Method	**Area** (mm^2)	**Power** (μW)	**Energy** (μJ)	**Latency** (ns)	**Cycles**	**# Adders**
Binary (FP32)	6.627	300.283	3.754×10^{-6}	12.50	5	480
Opt. BUs	0.034	3.124	0.281	90.00	36	320
Unary (uSense)	0.017	1.541	0.139×10^{-6}	90.00	36	154
Gain	390× ↓	195× ↓	27× ↓	7.2× ↑	7.2× ↑	67.92% ↓

5 Discussion

Tactile sensing has advanced with the development of high-resolution artificial skins[6,44,70,71], but it suffers from computational bottlenecks. Current systems rely on processors decoupled from the sensors, requiring continuous transmission of dense raw signals to centralized computing units[72], increasing bandwidth, latency, and energy consumption. Subsequently, limiting deployment on edge platforms for real-time closed-loop tasks such as dexterous manipulation[73]. Crucially, this limitation reflects a mismatch between the spatiotemporal structure of tactile signals and conventional near-sensor computational frameworks under strict resource constraints. We discuss uSense's performance with existing tactile sensing approaches based on (i) classical binary and (ii) conventional neuromorphic approaches.

Classical binary approaches

Deep learning models have shown strong performance in texture recognition. Xiong et al.[45] introduced a dictionary-reconstruction architecture that fused multimodal features and achieved 97.7% and 89.4% accuracy on the LMT-108 (9 textures) and SpectroVision datasets (8 textures), respectively. Aggarwal et al.[74] used a CNN to classify 16 texture classes and reached 96.36% accuracy. However, these models impose significant overheads in memory, computation, and energy, which limit their practicality for wearable or embedded tactile systems operating under tight resource constraints. In contrast, our results show that almost-lossless texture recognition can be achieved without such computational overhead by constraining spectral analysis to a low-frequency range and computing directly in the unary domain.

Frequency-based pipelines combined with lightweight classifiers offer a simpler alternative. Rivera et al.[27] introduced a hardware-oriented feature extractor on an FPGA with a k-means classifier that achieved 89.17% accuracy compared with 91.4% using FFT features across 12 textures. Their evaluation, however, is restricted to constant speed and a single sensing channel, underscoring the hardware challenges of deploying FFT-based processing. Moreover, conventional FFT computation depends on 32-bit floating-point arithmetic whose latency and memory overheads scale poorly with signal dimensionality. Our DAG-pruned stochastic FFT addresses these limitations directly by eliminating unnecessary computations and reducing bitwidth, lowering both memory footprint and computational complexity while preserving exact outputs within the selected frequency band.

Conventional neuromorphic approaches

Motivated by the advantage of spike-based processing, many recent tactile studies have begun using neuromorphic computing to develop decoders that remain agnostic to natural variations during active touch. Iskarous et al.[75] introduced a speed–force invariant neuromorphic representation by streaming tactile and motion data to a microcontroller that generated spiking activity through an Izhikevich neuron model and applied speed–force scaling processing. The resulting spike-train features are then transmitted to a computer for texture classification using linear discriminant analysis. Texture recognition is evaluated across 16 textures and 15 speed–force conditions, achieving 83.15% accuracy. However, decoding occurs only after the full 2 to 6 second trial, thus not in real-time. Moreover, this setup depends on a dedicated neuromorphic hardware, explicit normalization steps to overcome variances in force-speed combinations, and dense raw data transmission. In contrast, our uSense framework is evaluated across 20 textures and 20 speed–force combinations, inspired by Isakrouss et al.[75], but with considerably higher hardware efficiency. uSense performs computation directly on the incoming data stream and reduces transmitted data by 95% without additional processing, supporting real-time near-sensor operation. Moreover, uSense achieved 97.68% accuracy using an SVM decoder, outperforming prior neuromorphic systems while operating under extremely strict computational and bandwidth constraints.

Other neuromorphic encoders rely on mixed-mode subthreshold circuits[76,77] or oscillator-based spike generation[78]. These designs optimize power efficiency but typically require mechanoreceptor-specific transducers, which complicate mechanical interfacing and fixed-frequency spike timing, limiting their ability to capture the adaptive firing behaviors of biological mechanoreceptors[79]. In contrast, uSense is based on a unary representation that aligns with the biological tactile encoding, which can be easily implemented under tangible CMOS technologies, facilitating large-scale adoption.

Future directions.

Future work will investigate how uSense performs across a broader range of tactile interactions beyond controlled sliding contact. In particular, extending uSense to multimodal tactile sensing represents a promising direction, including integration of temperature or proprioceptive cues to examine how near-sensor unary computing

interacts with different sensory inputs. Finally, while the present study demonstrates strong classification performance in simulation, implementing uSense on physical hardware will be essential to validate latency, energy consumption, and performance robustness in real-world applications.

6 Conclusion

In this article, we introduce uSense, a unary computing-based stochastic edge neuromorphic sensing framework for tactile sensing using FFT. uSense addresses the gap between high decoding accuracy and the constraints of real-time operation on resource-constrained edge devices via biologically inspired encoding and spectral computations directly at the sensing interface. Our framework demonstrates significant reductions in data transmission, computation cost, hardware area, and energy consumption compared with conventional edge systems, while preserving high decoding accuracy across force–speed conditions, underscoring its potential for efficient and scalable tactile intelligence. We demonstrate that uSense can compress the data by two orders of magnitude with negligible accuracy loss. The energy consumption is also projected to be lowered by one order of magnitude compared to the FP32 baseline FFT, while maintaining real-time responses.

7 Data availability

Representative tactile data used in this study are publicly available at https://github.com/LIMB-UCF/uSense-tactile-data. The complete dataset is available from the corresponding author upon reasonable request.

8 Code availability

The uSense stochastic FFT implementation is publicly available at https://github.com/UnaryLab/napl/blob/fft/src/napl/algorithm/fft/fft.py. The end-to-end uSense feature extraction, processing, and texture classification pipeline is available at https://github.com/UnaryLab/napl/tree/fft/zoo/uSense.

References

[1] Ding, K., Rakhshan, M., Paredes-Acuña, N., Cheng, G. & Thakor, N. V. Sensory integration for neuroprostheses: from functional benefits to neural correlates. *Medical & Biological Engineering & Computing* **62**, 2939–2960 (2024).

[2] Handler, A. & Ginty, D. D. The mechanosensory neurons of touch and their mechanisms of activation. *Nature Reviews Neuroscience* **22**, 521–537 (2021).

[3] Jenkins, B. A. & Lumpkin, E. A. Developing a sense of touch. *Development* **144**, 4078–4090 (2017).

[4] Wei, Y. *et al.* Human tactile sensing and sensorimotor mechanism: from afferent tactile signals to efferent motor control. *Nature Communications* **15**, 6857 (2024).

[5] Bensmaia, S. J., Tyler, D. J. & Micera, S. Restoration of sensory information via bionic hands. *Nature Biomedical Engineering* **7**, 443–455 (2023).

[6] Bai, N. *et al.* A robotic sensory system with high spatiotemporal resolution for texture recognition. *Nature Communications* **14**, 7121 (2023).

[7] Qiu, Y. *et al.* Nondestructive identification of softness via bioinspired multisensory electronic skins integrated on a robotic hand. *npj Flexible Electronics* **6**, 45 (2022).

[8] Johansson, R. S. & Flanagan, J. R. Coding and use of tactile signals from the fingertips in object manipulation tasks. *Nature Reviews Neuroscience* **10**, 345–359 (2009).

[9] He, K. *et al.* An artificial somatic reflex arc. *Advanced Materials* **32**, 1905399 (2020).

[10] Mao, Q., Liao, Z., Yuan, J. & Zhu, R. Multimodal tactile sensing fused with vision for dexterous robotic housekeeping. *Nature Communications* **15**, 6871 (2024).

[11] Zheng, C. *et al.* Design of multi-modal feedback channel of human–robot cognitive interface for teleoperation in manufacturing. *Journal of Intelligent Manufacturing* **36**, 4283–4303 (2025).

[12] Pang, G., Yang, G. & Pang, Z. Review of robot skin: A potential enabler for safe collaboration, immersive teleoperation, and affective interaction of future collaborative robots. *IEEE Transactions on Medical Robotics and Bionics* **3**, 681–700 (2021).

[13] He, L. *et al.* Robotic simulators for tissue examination training with multimodal sensory feedback. *IEEE Reviews in Biomedical Engineering* **16**, 514–529 (2023).

[14] Chen, L. *et al.* Spike timing–based coding in neuromimetic tactile system enables dynamic object classification. *Science* **384**, 660–665 (2024).

[15] Dahiya, R. S., Metta, G., Valle, M. & Sandini, G. Tactile sensing—from humans to humanoids. *IEEE transactions on robotics* **26**, 1–20 (2009).

[16] Jones, L. *Haptics* (MIT press, 2018).

[17] Zollo, L. *et al.* Restoring tactile sensations via neural interfaces for real-time force-and-slippage closed-loop control of bionic hands. *Science robotics* **4**, eaau9924 (2019).

[18] Raspopovic, S., Valle, G. & Petrini, F. M. Sensory feedback for limb prostheses in amputees. *Nature Materials* **20**, 925–939 (2021).

[19] Zhao, Z. *et al.* Embedding high-resolution touch across robotic hands enables adaptive human-like grasping. *Nature Machine Intelligence* 1–12 (2025).

[20] Iskandar, M., Albu-Schäffer, A. & Dietrich, A. Intrinsic sense of touch for intuitive physical human-robot interaction. *Science Robotics* **9**, eadn4008 (2024).

[21] Cimpoi, M., Maji, S. & Vedaldi, A. *Deep filter banks for texture recognition and segmentation*, 3828–3836 (2015).

[22] Chen, Z., Li, F., Quan, Y., Xu, Y. & Ji, H. *Deep texture recognition via exploiting cross-layer statistical self-similarity*, 5231–5240 (2021).

[23] Böhm, A. *et al. What matters for active texture recognition with vision-based tactile sensors*, 15099–15105 (2024).

[24] Guo, Q., Al, G. A. & Martinez-Hernandez, U. Vibrotact: Soft piezo vibration fingertip sensor for recognition of texture roughness via robotic sliding exploratory procedures. *IEEE Sensors Letters* **8**, 1–4 (2024).

[25] Kaboli, M., Walker, R., Cheng, G. *et al. In-hand object recognition via texture properties with robotic hands, artificial skin, and novel tactile descriptors*, 1155–1160 (IEEE, 2015).

[26] Markert, T., Matich, S., Hoerner, E., Theissler, A. & Atzmueller, M. *Fingertip 6-axis force/torque sensing for texture recognition in robotic manipulation*, 1–8 (IEEE, 2021).

[27] Lora-Rivera, R., Oballe-Peinado, Ó. & Vidal-Verdú, F. Texture detection with feature extraction on embedded fpga. *IEEE Sensors Journal* **23**, 12093–12104 (2023).

[28] Kroemer, O., Lampert, C. H. & Peters, J. Learning dynamic tactile sensing with robust vision-based training. *IEEE Transactions on Robotics* **27**, 545–557 (2011).

[29] Hou, C. *et al.* A piezoresistive-based 3-axial mems tactile sensor and integrated surgical forceps for gastrointestinal endoscopic minimally invasive surgery. *Microsystems & Nanoengineering* **10**, 141 (2024).

[30] Taunyazov, T. *et al. Extended tactile perception: Vibration sensing through tools and grasped objects*, 1755–1762 (2021).

[31] Yoshimoto, S., Sakamoto, K., Takeda, R. & Yamamoto, A. Design of a high-performance tomographic tactile sensor by manipulating the detector conductivity. *IEEE Transactions on Industrial Electronics* **71**, 16783–16791 (2024).

[32] Murayama, Y. & Lindahl, O. A. Sensitivity improvements of a resonance-based tactile sensor. *Journal of medical engineering & technology* **41**, 131–140 (2017).

[33] Chen, H. *et al.* A large-area flexible tactile sensor for multi-touch and force detection using electrical impedance tomography. *IEEE Sensors Journal* **22**, 7119–7129 (2022).

[34] Koul, M., Manivannan, M. & Saha, S. K. Effect of dual-rate sampling on the stability of a haptic interface. *Journal of Intelligent & Robotic Systems* **91**, 479–491 (2018).

[35] Ren, M., Wu, Q. & Huang, X. Flexible tactile sensors inspired by bio-mechanoreceptors. *Biosensors and Bioelectronics* **267**, 116828 (2025).

[36] Zhao, Z. *et al.* Large-scale integrated flexible tactile sensor array for sensitive smart robotic touch. *ACS nano* **16**, 16784–16795 (2022).

[37] Dian, F. J., Yousefi, A. & Lim, S. *A practical study on bluetooth low energy (ble) throughput*, 768–771 (2018).

[38] Craig, W. C. Zigbee: Wireless control that simply works. *Program Manager Wireless Communications ZMD America, Inc* (2004).

[39] Najafi, M. H., Faraji, S. R., Bazargan, K. & Lilja, D. *Energy-efficient pulse-based convolution for near-sensor processing*, 1–5 (IEEE, 2020).

[40] Wu, D., Li, J., Pan, Z., Kim, Y. & Miguel, J. S. *ubrain: a unary brain computer interface*, ISCA '22, 468–481 (Association for Computing Machinery, New York, NY, USA, 2022). URL https://doi.org/10.1145/3470496.3527401.

[41] Faraji, S. R. & Bazargan, K. *Hybrid binary-unary hardware accelerator*, 210–215 (2019).

[42] Li, H. & Chen, Y. Hybrid logic computing of binary and stochastic. *IEEE Embedded Systems Letters* **14**, 171–174 (2022).

[43] Wu, D. *et al.* *Ugemm: Unary computing architecture for gemm applications*, 377–390 (IEEE, 2020).

[44] Rostamian, B. *et al.* Texture recognition based on multi-sensory integration of proprioceptive and tactile signals. *Scientific reports* **12**, 21690 (2022).

[45] Xiong, P., Zhang, K., Shi, Z., Zhou, M. & Song, A. Deep dictionary learning with reconstruction for texture recognition. *Scientific Reports* **15**, 31164 (2025).

[46] Ibrahim Awan, M. & Jeon, S. Estimating perceptual attributes of haptic textures using visuo-tactile data. *IEEE Access* **13**, 109931–109945 (2025).

[47] Zheng, Y., Yu, Z., Wang, S. & Huang, T. Spike-based motion estimation for object tracking through bio-inspired unsupervised learning. *IEEE Transactions on Image Processing* **32**, 335–349 (2022).

[48] Zhu, Z. *et al.* Recent advances in spike-based neural coding for tactile perception. *Microsystems & Nanoengineering* **11**, 212 (2025).

[49] Lee, V. T., Alaghi, A., Hayes, J. P., Sathe, V. & Ceze, L. *Energy-efficient hybrid stochastic-binary neural networks for near-sensor computing*, 13–18 (IEEE, 2017).

[50] Gaines, B. R. Stochastic computing systems. *Advances in Information Systems Science: Volume 2* 37–172 (1969).

[51] Gonzalez-Guerrero, P., Guo, X. & Stan, M. R. *Asc-fft: Area-efficient low-latency fft design based on asynchronous stochastic computing*, 117–120 (2019).

[52] Wu, D. & Miguel, J. S. *In-Stream Stochastic Division and Square Root via Correlation* (2019).

[53] Wu, D., Yin, R. & Miguel, J. S. In-Stream Correlation-Based Division and Bit-Inserting Square Root in Stochastic Computing. *IEEE Design & Test* **38**, 53–59 (2021).

[54] Wu, D., Yin, R. & Miguel, J. S. *Normalized Stability: A Cross-Level Design Metric for Early Termination in Stochastic Computing* (2021).

[55] Wu, D. *et al.* uGEMM: Unary Computing for GEMM Applications. *IEEE Micro* **41**, 50–56 (2021).

[56] Wu, D. & Miguel, J. S. *usystolic: Byte-crawling unary systolic array*, 12–24 (2022).

[57] Lee, V. T., Alaghi, A., Hayes, J. P., Sathe, V. & Ceze, L. *Energy-efficient hybrid stochastic-binary neural networks for near-sensor computing*, 13–18 (2017).

[58] Alaghi, A., Qian, W. & Hayes, J. P. The promise and challenge of stochastic computing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **37**, 1515–1531 (2017).

[59] Alaghi, A. & Hayes, J. P. *Exploiting correlation in stochastic circuit design*, 39–46 (IEEE, 2013).

[60] Lee, V. T., Alaghi, A. & Ceze, L. *Correlation manipulating circuits for stochastic computing*, 1417–1422 (IEEE, 2018).

[61] Jenson, D. & Riedel, M. *A deterministic approach to stochastic computation*, 1–8 (IEEE, 2016).

[62] Liu, S. & Han, J. *Energy efficient stochastic computing with sobol sequences*, 650–653 (IEEE, 2017).

[63] Najafi, M. H., Jenson, D., Lilja, D. J. & Riedel, M. D. Performing stochastic computation deterministically. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **27**, 2925–2938 (2019).

[64] Ardakani, A., Leduc-Primeau, F., Onizawa, N., Hanyu, T. & Gross, W. J. Vlsi implementation of deep neural network using integral stochastic computing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **25**, 2688–2699 (2017).

[65] Alaghi, A., Li, C. & Hayes, J. P. *Stochastic circuits for real-time image-processing applications*, 1–6 (2013).

[66] Alaghi, A. & Hayes, J. P. *Fast and accurate computation using stochastic circuits*, 1–4 (IEEE, 2014).

[67] Price, D., Vellaisamy, P., Shen, J. & Wu, D. Mugi: Value level parallelism for efficient llms. *arXiv preprint arXiv:2601.10823* (2026).

[68] https://github.com/UnaryLab/archx/tree/f18425ee1503b6607dca20302b65be85a94f4bec/src/archx (2026). Accessed: Jan. 2026.

[69] Price, D. *et al.* A-graph: A unified graph representation for at-will simulation across system stacks. *arXiv preprint arXiv:2602.04847* (2026).

[70] Li, S. *et al.* Bioinspired robot skin with mechanically gated electron channels for sliding tactile perception. *Science advances* **8**, eade0720 (2022).

[71] Gupta, A. K., Nakagawa-Silva, A., Lepora, N. F. & Thakor, N. V. Spatio-temporal encoding improves neuromorphic tactile texture classification. *IEEE Sensors Journal* **21**, 19038–19046 (2021).

[72] Wang, M. *et al.* Artificial skin perception. *Advanced Materials* **33**, 2003014 (2021).

[73] Zhou, F. & Chai, Y. Near-sensor and in-sensor computing. nat. electron. 3, 664–671 (2020).

[74] Aggarwal, A. & Kumar, M. Image surface texture analysis and classification using deep learning. *Multimedia Tools and Applications* **80**, 1289–1309 (2021).

[75] Iskarous, M. M. *et al.* Invariant neuromorphic representations of tactile stimuli improve robustness of a real-time texture classification system. *Advanced Intelligent Systems* **7**, 2401078 (2025).

[76] Donati, E. & Valle, G. Neuromorphic hardware for somatosensory neuroprostheses. *Nature Communications* **15**, 556 (2024).

[77] Bartolozzi, C., Indiveri, G. & Donati, E. Embodied neuromorphic intelligence. *Nature communications* **13**, 1024 (2022).

[78] Subbulakshmi Radhakrishnan, S., Sebastian, A., Oberoi, A., Das, S. & Das, S. A biomimetic neural encoder for spiking neural network. *Nature communications* **12**, 2143 (2021).

[79] Liu, F. *et al.* Neuro-inspired electronic skin for robots. *Science robotics* **7**, eabl7344 (2022).