Accelerated Spiking Convolutional Neural Networks for Scalable Population Genomics

Federico Corradi
Eindhoven University of Technology
Eindhoven, The Netherlands, The Netherlands

Zhanbo Shen
Eindhoven University of Technology
Eindhoven, The Netherlands, The Netherlands

Hanqing Zhao
University of Twente
Enschede, The Netherlands, The Netherlands

Nikolaos Alachiotis
University of Twente
Enschede, The Netherlands, The Netherlands

ABSTRACT

Population genomics studies the genetic composition of populations to explain the evolutionary mechanisms that have contributed to species' adaptation to their environment. With continuous advances in DNA sequencing technologies, dataset sizes have grown, leading to an increased demand for processing power. Typical acceleration efforts for bioinformatics map established analytical models, including Convolutional Neural Networks (CNNs), to accelerator hardware like Graphical Processing Units (GPUs) and Field-Programmable Gate Arrays (FPGAs). However, these methods are not explicitly designed for high throughput or efficient mapping to these platforms, limiting their performance potential. Here, we address two major computational problems in population genomics: detecting selective sweeps and recombination hotspots. These evolutionary processes, crucial for species adaptation and survival, find applications in plant breeding, human genetics, and drug design. We propose a scalable solution that anticipates larger sample sizes, and, for the first time, we introduce a co-design approach that explores SCNNs for scalable population genomics and their implementation on FPGA technology. We choose SCNNs because these can be efficiently implemented on FPGA hardware due to their massive parallelism and binary communication, resulting in lower resource utilization and high-throughput performance. Our findings show that when using SCNNs, it is only necessary to process a portion of the sample size while maintaining a classification accuracy comparable to conventional CNNs. We demonstrate the performance of our system in FPGA hardware for several genomic tasks, achieving up to 3X throughput and 100X less power consumption. This paves the way for a high-performance, future-proof acceleration solution that addresses the computational challenges of increasing sample sizes and longer chromosome lengths.

CCS CONCEPTS

• **Computer systems organization** → **Neural networks**.

HEART '24, June 19–21, 2024, Porto, Portugal

ACM ISBN 979-8-4007-1727-7/24/06
https://doi.org/10.1145/3665283.3665285

KEYWORDS

Population Genomics, Spiking Neural Networks, Field-Programmable Gate-Arrays, High-Level Synthesis, Hardware Accelerators

ACM Reference Format:
Federico Corradi, Zhanbo Shen, Hanqing Zhao, and Nikolaos Alachiotis. 2024. Accelerated Spiking Convolutional Neural Networks for Scalable Population Genomics. In *14th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies (HEART'24)) (HEART '24), June 19–21, 2024, Porto, Portugal.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3665283.3665285

1 INTRODUCTION

Bioinformatics provides life and health science investigators with essential tools to process molecular data, addressing pivotal biological questions. These range from disentangling the evolutionary history of organisms [5] to identifying clinically relevant genes crucial for cancer risk assessment, diagnosis, prognosis, and treatment [42]. Recent years have witnessed advancements in DNA sequencing technologies, resulting in increased throughput and reduced costs. This has led to a substantial increase in the accumulation of whole genomes within databases [8, 43]. Consequently, studies typically incorporate an increasing number of organisms in an ongoing effort to improve statistical power, ultimately improving the accuracy and reliability of the results [10]. This evolution has naturally transformed bioinformatics into a computational discipline, necessitating scalable algorithms and high-performance processing systems.

The need for high-performance bioinformatic solutions in population genomics has been further exacerbated in recent years due to the introduction of numerous approaches and methods that rely on computational neural networks [29]. Population genomics analyzes the genetic variation of populations to detect and understand evolutionary phenomena such as positive selection, epistasis, recombination, linkage disequilibrium, and genetic drift, among others. Under certain evolutionary assumptions, such as the infinite-sites model [28], genetic variation at a genomic location is represented as a binary vector that encodes allele differences resulting from mutations across individuals within a population. Because physically close genes tend to be inherited together more frequently than expected by chance, statistical methods that analyze population genomics data typically examine genomic windows instead of individual locations independently, resulting in 2D matrices that can be easily represented as images. This has steered method development in the direction of casting challenging problems in

population genomics, such as the detection of selective sweeps and recombination hotspots, as an image classification task that exploits CNNs [18, 34, 56]. A selective sweep describes the phenomenon of observing reduced genetic diversity in the region surrounding a fixed (already spread to the entire population) beneficial mutation, while a recombination hotspot refers to a specific subgenomic region where the frequency of genetic recombination events is higher than the surrounding areas.

CNNs deliver high classification accuracy when processing subgenomic regions. Yet, they require prohibitively long execution times to thoroughly process full genomes. Based on a partial scan of the human chromosome 22 using SweepNet [56] on a personal computer with an Intel Core i7 CPU running at 2.60GHz and including all 2,504 samples sequenced and made available by the 1000Genomes project [12], we estimate that scanning the entire human genome would take approximately 4,652 CPU hours, i.e., a single CPU core running for longer than half a year. The problem of long execution times and the need for high compute power is only expected to worsen as sample sizes increase (e.g., the 100,000 Genomes project [46]), thereby discovering new genetic variants and consequently leading to ever larger images to be classified. To this end, we explore a co-design approach that introduces a novel SCNN architecture for the classification of adaptive genomics regions and we present a custom hardware acceleration architecture implemented on FPGA that delivers unprecedented performance.

SCNNs offer a computational paradigm inspired by how biological neural networks process information [51]. SCNNs handle temporal data more efficiently than traditional CNNs. Unlike CNNs, which require the entire input (such as an image or a matrix representing genomic data) to be loaded and processed as one step, SCNNs operate on an event-driven basis. They process input in the forms of binary pulses (e.g., spikes) that are transmitted across neurons in the network instead of real-valued neural activations as in CNNs. This characteristic drastically reduces the computational requirements when classifying adaptive genomic regions, as only the parts of the input data containing relevant information trigger processing (i.e., the part of the binary input being positive). Moreover, SCNNs naturally capture the temporal dynamics of input data, which is a significant advantage that enables early detection when analyzing large-scale genomic data. Studies typically analyze data sets with hundreds to thousands of sequences from the same species/population to increase accuracy and confidence, and SCNNs treat the input as unfolding over different time scales by processing sequences individually.

Precisely, this study demonstrates how to exploit SCNNs to classify "temporal" patterns of genomic segments without processing the entire sample size at once. Moreover, we propose a hardware accelerator for SCNNs in FPGAs. We select FPGAs because these platforms can be easily configured to handle the sparse, massively parallel, and event-driven computations typical of SCNN models. We demonstrate that FPGA implementations lead to a high-performance genomic processing system, as the parallelism inherent in SCNNs helps reduce bandwidth and memory requirements.

In summary, the main contributions of this work are:

- We deliver a new SCNN architecture with comparable classification accuracy to leading-edge CNNs. This is achieved while only processing a subset of the total samples (image rows) for classifying selective-sweep and recombination-hotspot genomic regions. The new SCNN incorporates a flexible early-stopping mechanism that allows the inference process to halt as soon as a classification result is obtained. Our experiments demonstrate that, on average, by processing only 59.9% of the samples, the SCNN reaches 97.6% of the classification accuracy of a CNN for the same problem. To the best of our knowledge, this is the first study that explores the potential of SCNNs in processing cross-sequence segments as if they were temporally dependent. This approach addresses the computational challenge posed by increasing sample sizes in population genomics.
- We demonstrate that the combination of spike-based processing and the ability to exploit temporal patterns of the proposed SCNN model achieves high classification accuracy with less computational resources than GPUs and full-frame CNN-based approaches. This efficiency reduces memory requirements and bandwidth, as the network processes only pertinent events rather than the entire dataset. Consequently, our SCNN-based, FPGA-accelerated system delivers over $3x$ higher throughput than a GPU while being over $100x$ more power efficient in scanning large-scale population genomics datasets for selective sweeps and recombination hotspots.

2 BACKGROUND

2.1 Selective sweeps

Positive selection is an evolutionary process in which advantageous genetic traits that enhance an organism's fitness become more prevalent in a population over time, thereby playing a critical role in shaping the evolution of species. A selective sweep occurs when a specific genetic variant, typically a beneficial mutation, quickly rises in frequency in a population due to positive selection. Selective sweeps create genomic segments in a population that are characterized by fewer mutations than neighboring regions while exhibiting specific patterns in the Site Frequency Spectrum (SFS) [17] and in the levels of Linkage Disequilibrium (LD) [27]. The SFS is a representation of the distribution of allele frequencies at polymorphic sites in a population, while LD is a term that describes the non-random association of alleles at different loci (gene locations) on a chromosome. A polymorphic site is referred to as a Single-Nucleotide Polymorphism (SNP) and describes genomic locations where at least one mutation has occurred. The most commonly used measure of LD is the squared Pearson's correlation coefficient, calculated as follows:

$$LD = r^2 = \frac{(p_{AB} - p_A \times p_B)^2}{p_A \times p_B \times (1 - p_A) \times (1 - p_B)},$$

where p_A and p_B are the derived allele frequencies at loci A and B, respectively, while p_{AB} is the frequency of mutations occurring together at loci A and B in the same sequence.

Figure 1 shows an example of a selective sweep. An advantageous mutation, shown as a black circle, appears in an individual (Figure 1a) and spreads in the population over generations (Figure 1b) until all individuals carry this beneficial mutation (Figure 1c). This

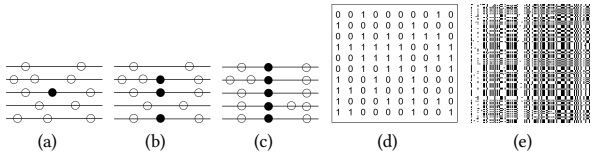

Figure 1: A selective sweep and gray-scale image conversion. a) A beneficial mutation (black circle) emerges within the population. b) The frequency of the chromosome harboring the advantageous mutation increases. c) The beneficial mutation becomes fixed (all individuals have it) and the frequency of neutral mutations (white circles) changes. d) Simulated SNP data under the Infinite-Sites Model (10 SNPs/columns and 10 samples/rows). The numbers '0' and '1' represent the ancestral (no mutation with respect to a reference genome) and derived (mutation) states, respectively. e) A gray-scale image representing SNP data (128 SNPs and 128 samples). The white pixels indicate the ancestral state, while the black pixels represent the derived state.

means that the frequency of the advantageous mutation reaches 1.0. As other evolutionary forces act on the genomes as well (e.g., genetic hitchhiking [17]). The frequency of neutral mutations, shown as white circles, linked to the selected allele also changes. A pattern of high LD is observed on the same side of the selected locus, while a low LD pattern is observed across the different sides.

Figures 1e and 1f show how the simulated genetic data are represented as gray-scale images under the Infinite-Sites Model (ISM) [28]. The ISM assumes an infinite number of sites. Thus, the probability that more than one mutation occurs at the same site is zero, which requires only two states, 0/1, for data representation. In the raw SNP data (Figure 1e), state '0' indicates the ancestral state, while state '1' represents the derived state. Both states are defined with respect to a reference/ancestral genome; therefore, a derived state implies that it results from a mutation. Subsequently, these binary data are transformed into gray-scale images (Figure 1f), where '0's appear as white pixels and '1's appear as black pixels. Gray-scale images are typically used as input to Convolutional Neural Networks (CNN) for image classification in population genomics. One of the main reasons for the shift of the development of methods in population genomics toward the use of deep learning methods is the lack of theoretical knowledge to explain other evolutionary factors that generate similar patterns as selective sweeps [18, 57], thus confounding population genomics inference. For instance, population bottlenecks in the form of sharp reductions in the size of a population, results in a significant loss of genetic diversity, and migration between adjacent populations can generate patterns similar to those expected by a selective sweep.

2.2 Recombination hotspots

A recombination hotspot is a specific region within the genome where genetic recombination occurs, the process by which DNA is exchanged between two homologous chromosomes during cell division, leading to the creation of new combinations of genetic information at an elevated rate with respect to the rest of the genome. Typically, recombination hotspots are narrow regions spanning 1 to 2 kilo-bases (kb). Recombination hotspots play a crucial role in shaping the patterns of genetic diversity in populations. They create a distinct pattern of LD, a block-like structure characterized by islands of high LD interspersed with regions experiencing rapid LD breakdown [24], providing signatures to identify and locate such regions. Unlike population bottlenecks and migration, which affect genetic diversity and allele frequencies, recombination hotspots primarily impact recombination rates and the genomic patterns associated with LD, an essential measure for identifying selective sweeps. Therefore, recombination hotspots confound selective sweep detection by affecting LD levels.

The spatial distribution and varying intensities of recombination hotspots can influence haplotype structures in the genome and complicate the identification of selective sweeps. Thus, accurately detecting recombination hotspots and quantifying their intensity is of paramount importance in population genomics.

2.3 Spiking convolutional neural networks

SCNNs are a class of neural networks where layers of spiking neurons are connected, just as traditional CNNs. This includes convolutional layers for extracting features and fully connected layers for classification. The unique feature of SCNNs is the use of spiking neurons. Figure 2 shows a schematic representation of a spiking convolutional neural network architecture.

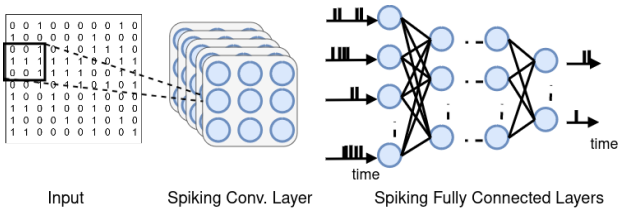

Figure 2: Spiking convolutional neural network, convolutional layers followed by fully connected layers of spiking neurons communicating through binary spikes.

Unlike traditional CNNs, communication between layers in a spiking neural network occurs through binary sparse pulses. SCNNs are efficient due to the time-dependent and infrequent occurrence of spikes, which allows neurons to process and transmit binary data only when required. This is in stark contrast with the constant computational requirements of conventional CNNs, which necessitate the exchange of real values at regular intervals.

Several neuron models for Spiking Neural Networks (SNNs) have been implemented on FPGA, such as Hodgkin-Huxley [50], Izhikevich [40], Wilson [22], Fitzhugh-Nagumo [35], Morris-Lecar [32], and Leaky Integrate-and-Fire (LI&F) models [26], providing a spectrum of complexity and biological realism. The LI&F model stands out for its ability to balance computational efficiency and reasonable biological plausibility. The LI&F neuron model is described by the neuron's cell membrane in terms of leak conductance and capacitance. Weighted synaptic spikes from presynaptic neurons represent input stimuli. When the membrane potential, indicated

as $u(t)$, reaches a predefined threshold level (V_{th}), the neuron activates, resulting in a voltage spike. Subsequent to this activation, the potential is reset to the resting potential level (u_{rest}), allowing the cycle to restart after a refractory period ($t_{refract}$). Figure 3 shows a schematic representation of the behavior of a LI&F neuron. The sub-threshold behavior of the LI&F neuron is defined by a differential equation as follows:

$$\tau_m \frac{du}{dt} = -(u(t) - u_{rest}) + \sum_{i=1}^{n^l} w_i \cdot \sum_{j=1}^{n} \delta(t - t_j), \qquad (1)$$

where $u(t)$ represents the membrane potential of the neuron at time t, τ_m is the time constant of the membrane, n^l is the count of presynaptic weights, and n denotes the number of presynaptic spikes in the sequence. The term w_i represents the weight of the synaptic connection from the presynaptic neuron i to the postsynaptic neuron, and δ represents the Dirac delta function, indicating the times when spikes occur.

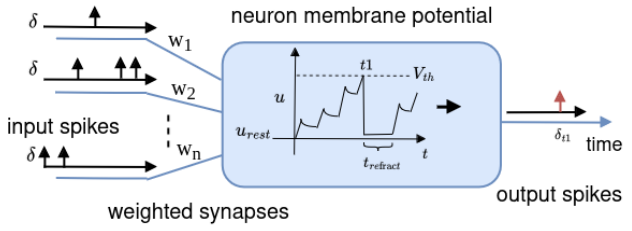

Figure 3: Schematic representation of the Leaky-integrate-and-fire neuron model; the neuron accumulates incoming spikes weighted by the synaptic weights w_i; if the membrane potential u reaches the threshold V_{th} the neuron emits a spike, and the membrane potential falls to the resting potential. During the refractory period $t_{refract}$ after a spike, the membrane potential stays at the resting value, and the neuron is inactive.

3 RELATED WORK

3.1 Population genomics

Various problems in population genomics have been previously accelerated using hardware accelerators. Binder et al. [9] introduce a flexible framework for GPU-accelerated LD calculations. Tests on Nvidia TITAN V, GeForce GTX 980, and AMD Radeon Vega GPUs demonstrate speeds up to 7.8× faster than a BLIS-based CPU approach on an Intel Xeon E5-2620v2 CPU. Theodoris et al. [45] released quickLD, an optimized software for LD statistics on CPUs and GPUs using OpenCL. Using a server node with two Intel Xeon E5-2660v3 CPUs and an Nvidia Tesla K40 GPU, the authors report up to 29× faster processing than PLINK1.9 [11] (20 threads) for datasets of up to 100,000 samples and 10,000 SNPs. Wienbrandt et al. [48] present an FPGA-accelerated GWAS epistasis detection tool. The proposed solution is implemented on the RIVYERA [39] system that features 128 Xilinx Spartan 6-LX150 FPGAs and two Intel Xeon E5-2620 CPUs as host processors, achieving up to 285× faster processing than the iLOCi software executed on two Intel Xeon quad-core 2.4 GHz CPUs. Yung et al. [54] introduce GBOOST, a CUDA-based GPU-accelerated version of BOOST [47]. Tested on an Nvidia GeForce GTX 285, it achieves a 40× speedup over BOOST, running on a single CPU running at 3 GHz. Alachiotis et al. [3] propose a reconfigurable architecture based on accelerator tiles, dubbed Accelerator-Tile-Architecture (ATA), and implemented it for OmegaPlus [2], a sweep detection method. ATA is mapped on the Zynq 7045 MPSoC and achieves up to 6.42× speedup compared to OmegaPlus-G (software benchmark) tested on a workstation with an Intel Xeon E5-2630 6-core Sandy Bridge processor. Alachiotis et al. [4] present RAiSD-X, an FPGA accelerator for the μ statistic [1] for detecting positive selection. RAiSD-X is up to 124× and 40× faster than the parallel tools OmegaPlus [2] and SweeD [38], respectively, when 40 threads are launched on 20 CPU cores, and up to 1,755× and 75× times faster than the sequential tools SweepFinder2 [14] and RAiSD [1], respectively. Acceleration efforts in detecting recombination hotspots are limited in the literature. Guo et al. [19] utilize a Markov chain Monte Carlo convergence diagnostic algorithms to optimize LDhat (an extensively used statistical method) for estimating recombination rates, reporting an order of magnitude speedups on multi-core CPU systems.

3.2 Spiking convolutional neural networks

SCNNs have attracted significant attention for their ability to deliver comparable performance to traditional CNNs in vision, audio, and temporal signal processing benchmarks. In 2018, Rueckauer et al. [41] introduced a methodology to convert standard CNNs into SCNNs, facilitating a nearly lossless transition and achieving good performance on benchmarks such as MNIST and CIFAR10. Furthermore, recent advancements in training methods for SCNNs have incorporated surrogate training methods [15, 52]. These methods exploit a straight-through estimator that disregards the derivative of the threshold function and employs functions as tanh and Gaussian to address the discontinuity of spikes. These advancements, supported by tools such as PyTorch and GPU hardware, have expanded the applications of spiking neural networks to complex tasks, including object detection and classification [53], spatiotemporal feature extraction, radar signal processing [7], medical image analysis [59], camera and radar images classification [13], and lidar point-cloud classification [60], among others.

Recent literature highlights the potential of mapping SCNNs onto FPGAs for applications requiring fast processing and high energy efficiency. Wu et al. [49] describe an FPGA-based SCNN using adaptive channel-wise logarithmic quantization and optimizing hardware resource efficiency, achieving up to 99.26% accuracy on the MNIST dataset. Aung et al. [6] present an FPGA architecture that accommodates large neural network layers, demonstrating the high throughput of 1,500 frames per second in the ImageNet recognition tasks, highlighting its suitability for large-scale, energy-efficient image processing. Panchapakesan et al. [36] demonstrate on multiple Xilinx ARM-FPGA SoC boards scalable spiking neural networks architectures with similar accuracy but better throughput performance compared to traditional CNNs models on three image classification benchmarks: MNIST, CIFAR-10, and SVHN datasets. Fang et al. [16] demonstrate a temporal encoding-based SNN inference on the Xilinx ZCU102 SoC board with 99.2% accuracy and

2,124 FPS at 125 MHz for MNIST classification. Irmak et al. [23] propose an FPGA-based architecture capable of switching between traditional CNNs and SNNs using dynamic partial reconfiguration without compromising throughput or accuracy and demonstrating that the implemented designs result in low-latency, highly flexible, and resource-efficient accelerator. Khodamoradi et al. [25] demonstrate a steaming SNN for radio frequency applications mapped into an FPGA. They achieve more than three-fold improvements in memory utilization concerning standard computational models. Leone et al. [30] introduce an FPGA-based neural decoding system employing an SNN, emphasizing an efficient spike detection methodology for continuous decoding tasks, achieving accuracy comparable to conventional decoders with fewer parameters while leveraging signal sparsity for reduced computational demand. Furthermore, combining SCNNs with binary input data, especially in sensor-processing from event-based cameras and delta-coded sensory systems, has shown favorable trade-offs among energy and latency performance [31].

The aforementioned results highlight the potential of Spiking Convolutional Neural Networks (SCNNs) implemented on FPGAs and demonstrate the flexibility and efficiency of SNN systems through various architectures and applications.

For gene sequence analysis, research so far has focused on the software-level exploration of SNNs [33] and SCNNs [58], outperforming traditional deep neural network models and positioning SNNs as a potent instrument for genomic studies. In addition to exploring for the first time, to the best of the authors' knowledge, SCNNs to address the sample-size scalability problem in population genomics, this work further advances the current state of the art with an efficient SCNN accelerator architecture for large-scale population genomics inference.

4 METHOD AND SYSTEM DESIGN

4.1 Spiking neural network architecture

Our SCNN model includes a convolutional layer and two fully connected layers, all designed and verified in High-Level Synthesis (HLS). The SCNN architecture is generated using the Vivado block design tool by linking the single-layer IPs generated and verified in HLS. The SCNN architecture is shown in Figure 4.

At each time step, five convolutional kernels work on three sequential input image rows. The output, 126 elements from each kernel, is integrated into LI&F neurons, with the leakage parameter set at zero. The output spikes from the convolutional layer are combined and passed to the fully connected layers of dimensions $630x128$ and $128x2$, and with each time increment, a new sequence of outputs is generated. After each fully connected layer, an additional LI&F neuron layer is implemented, emulating binary neural transmission. We have structured our SCNN model to boost hardware throughput by organizing computations in independent steps akin to pipeline integration in a processing datapath. This allows the fully connected layers to process outputs as they are produced, enabling continuous operation across all layers and enhancing computational efficiency.

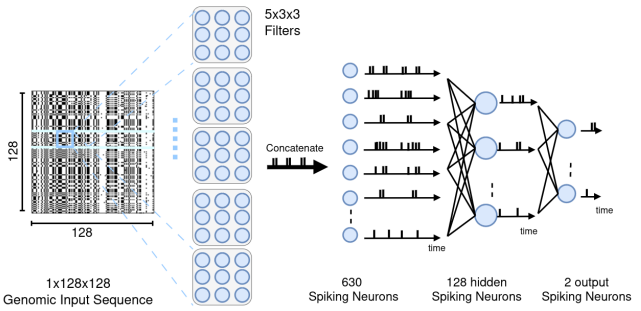

Figure 4: SCNN architecture tailored for processing input genomic sequences. The genomic sequences ($1x128x128$) are fed through a sequence of spiking convolutional filters ($5x3x3$). This produces outputs from 630 spiking neurons whenever their membrane potential surpasses the threshold. Following, the neurons are fully connected to a second hidden layer comprising 128 spiking neurons. This layer feeds into two final output neurons, whose activity constitutes the classification result.

4.2 SCNN training and model optimization

We used an NVIDIA A100 GPU and the PyTorch framework [37] to train SCNN models. We developed a custom model exploration code to emulate the main SCNN hardware functionalities while optimizing the SCNN architecture for hardware acceleration, focusing on quantized bit-precision, membrane potential constraints, and discrete neuronal timing. To enhance efficiency and reduce the memory requirements of our accelerator, we implemented weight quantization and, after testing, selected an 8-bit weight resolution in the integer range of $[-128, 127]$ due to the inability of a 4-bit weighted network to converge on a series of reference genomic datasets. Finally, we have set the activation threshold for all neurons at 256. As a result, the SCNN relies solely on integer operations, streamlining its deployment on FPGA devices.

Figure 6 shows the activity of the whole SCNN network during an inference. Each black dot in Figure 6 represents a neuronal activation, i.e., a spike. The genomic input sequence is streamed into the network in 128-time steps. The hidden neurons show sparse activations, and the output neuron zero is the most active, showing the correct classification of the genomic input sequence. Figure 5 shows the evolution of neuron spikes over the time steps and denotes the early-decision point for the input genomic window after 37 time steps. The PyTorch code and the accelerator architecture described using High-Level Synthesis (HLS) are available for download at: https://github.com/federicohyo/genomicsnn

4.3 Hardware accelerator architecture

The accelerator architecture supports fully connected and convolutional spiking neural networks. Its key features include pipelined processing for streamlined data handling, parallel weight fetching for efficient data retrieval, reconfigurability, and direct memory mapping of weights for quick read-write access.

Figure 7 illustrates the accelerator design. The accelerator architecture is implemented in the Programmable Logic (PL) within a

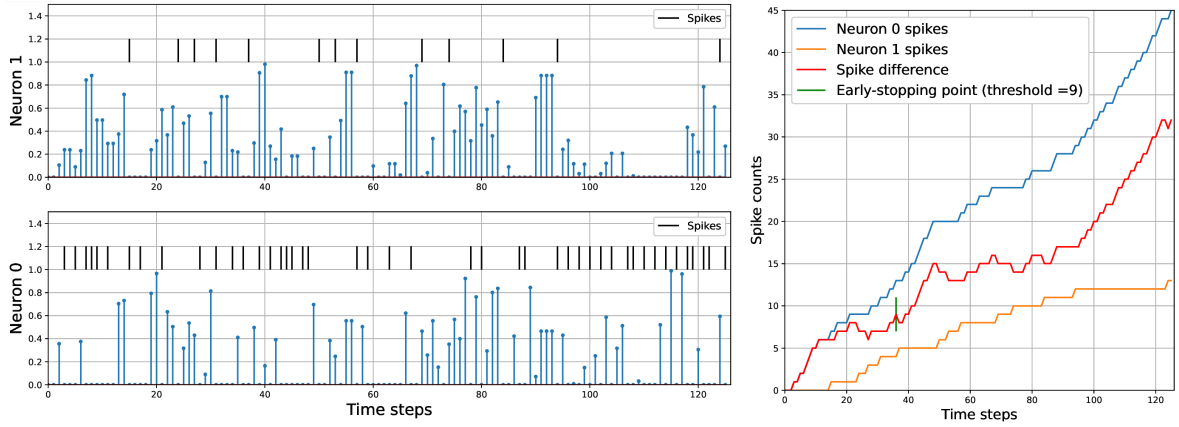

Figure 5: SCNN decision. The left plots show sampled values of the membrane potential of the output neurons during a classification inference and their output spikes. The right plot shows the output spike count for both output neurons and their spike count difference. The classification output (class zero) is correctly identified at time step 37 when the decision threshold is reached.

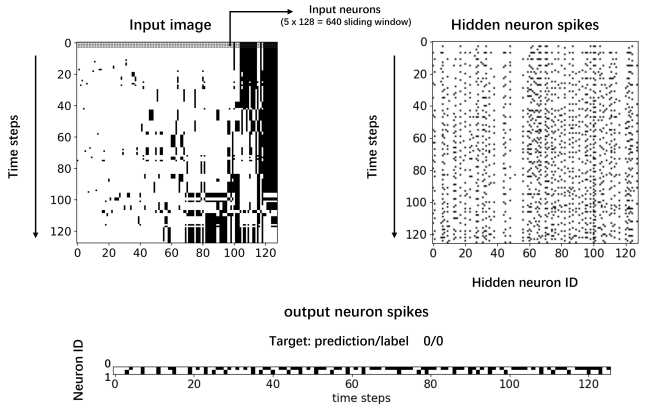

Figure 6: SCNN activity, the black dots represent a spike. The top right plots show the binary genomic input sequence, presented in 128 time steps with a sliding window of size 5x128. The top left plot shows the hidden neuron's spikes. The bottom plot shows the spikes from the two output neurons. The most active neuron indicates the output classification.

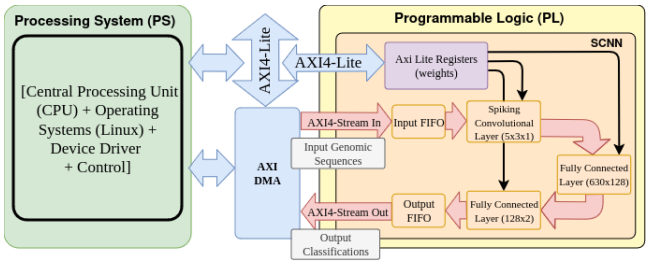

Figure 7: Hardware accelerator architecture. The entire setup consists of a Multiprocessor System-on-Chip (MPSoC), which includes an ARM processing system (PS), programmable logic (PL), and external DDR3 memory. Within the PL, the accelerator features a modular and pipelined design across its modules. The modules are connected via AXI4 Stream, and parameters are uploaded to the PL via the AXI Lite interface.

Multiprocessor System-on-Chip (MPSoc). It operates by receiving data from the Processing System (PS), processing this data, and then notifying the PS upon completion.

The SCNN architecture implementation offers customization options, including the ability to adjust the number and type of layers, choose different neuron models and parameters, select the resolution of synaptic weights, and configure the network topology to suit diverse requirements. We use memory-mapped AXI interfaces for data/parameters, AXI Lite for the weights, and AXI4 Streams for inter-module communication. Genomic sequences are retrieved from memory using Direct Memory Access (DMA) and placed into an input FIFO buffer. When an input batch is ready, and the system is idle, these sequences are sequentially processed through successive layers in a pipeline fashion, with new data rows introduced to previous layers as results advance. We employed the TUSER signal within the AXI4 Stream protocol to signal the beginning of a new genomic sequence, resetting each neuron's potential in the accelerator to zero for the initial row. Output data is gathered in an output FIFO and returned to memory via DMA when a batch is complete, facilitating the determination of genomic sequence classification outcomes based on the highest values in the output array.

5 EVALUATION

5.1 Experimental setup

To evaluate the performance of the SCNN accelerator, we generated simulated datasets with classic selective sweeps under three distinct demographic models: population bottleneck, migration, and

Table 1: Simulated datasets for evaluation and accuracy comparison per dataset between a 32-bit floating-point SweepNet and an 8-bit quantized integer-based SCNN. "s", "bt", "d", "jt" and "ri" represent severity, begin time, duration, population join time and recombination intensity, respectively.

Dataset	Test case	Confounding factor	Simulation software	Model parameters	SweepNet Acc.	SCNN Acc.
D1	classic selective sweep	Mild bottleneck	ms and mssel	s = 0.5, bt = 0.1, d = 0.01	100	100
D2	classic selective sweep	Severe bottleneck	ms and mssel	s = 0.005, bt = 0.1, d = 0.004	92	90.7
D3	classic selective sweep	Recent migration	ms and mssel	jt = 0.003	99	99.3
D4	classic selective sweep	Old migration	ms and mssel	jt = 3	87.2	89
D5	classic selective sweep	Low intensity recomb.	msHOT and mbs	ri = 2	89	84.4
D6	classic selective sweep	High intensity recomb.	msHOT and mbs	ri = 20	99	94.4
D7	Low intens. recomb. hotspot	-	msHOT	ri = 2	79.5	74.5
D8	High intens. recomb. hotspot	-	msHOT	ri = 20	99	98.6

recombination hotspots. These confounding factors challenge the accurate detection of selective sweeps. We additionally simulated recombination hotspots to evaluate the capability of SCNN in accurately classifying regions with recombination events of varying intensities. We used ms [21] and mssel (provided by R. R. Hudson) to generate input data for genetic population bottleneck and migration scenarios. For recombination events, we used msHOT [20] and mbs [44] to generate recombination hotspots without and with a selective sweep, respectively. For the simulations, we assumed that the present-day population size is 50,000 haploid genomes, and each population has 128 individuals. We encoded each population into a 128×128 image, i.e., 128 SNPs are extracted. Each dataset consists of a training set, a validation set, and a test set comprising 800, 200, and 1000 images, respectively. We preprocessed all images by arranging sequences based on their Hamming distance while sorting SNPs according to derived allele frequency [55] before feeding them to our SCNN. Table 1 provides simulation details for the dataset.

5.2 Classification accuracy using the entire sample size

We use a full-precision (i.e., 32-bit floating point) SweepNet [56] as the baseline to evaluate classification accuracy. Table 1 provides test accuracies of the SCNN and SweepNet for the different datasets. The SCNN attains comparable performance with SweepNet and outperforms it for some datasets. The SCNN practically achieves the same testing accuracy as SweepNet for mild bottlenecks (D1), recent migration (D3), and high-intensity recombination hotspots (D8). At the same time, a performance drop (between 1.3% and 5%) is observed for the more challenging scenarios such as severe bottlenecks (D2), old migration (D4), recombination hotspots as a confounding factor (D5 and D6), and low-intensity recombination hotspots (D7). Overall, the accuracy of the SCNN shows a slight decrease compared to SweepNet, which is an acceptable trade-off between accuracy and low-precision weights. Notice that SweepNet uses 32-bit single-precision floating-point weights, whereas the SCNN relies on 8-bit quantized integer weights.

5.3 Classification accuracy using early-stopping inference

The proposed SCNN model processes input images three rows at a time, treating their processing as a distinct time step. This sequential approach enables the model to perform classification based on the data gathered over multiple time steps, introducing the concept of early-stopping inference. This technique allows for the use of fewer steps to perform a classification. The criteria for early stopping is determined by the difference in the firing counts of two output neurons, referred to as the early-stopping threshold. When the firing count of one neuron surpasses that of the other by a margin exceeding this threshold, the classification is deemed to have reached sufficient confidence, and the corresponding class of the more active neuron is selected as the prediction outcome.

As depicted in Figure 8a, there is a notable increase in accuracy as the early-stopping threshold is adjusted from 1 to 9. This increase is particularly significant in the threshold range from 1 to just below 5, where accuracy swiftly escalates from lower levels to match the accuracy achieved by considering the entire image for classification. In particular, our model reaches maximum accuracy on Dataset 1 and Dataset 3 with an early-stopping threshold of 6. For all of the other datasets, the early-stopping accuracy saturates with an early-stopping threshold between 2 and 9.

Figure 8a indicates that most datasets achieve maximum accuracy when the entire genomic sequence is processed. Additionally, Figure 8b shows that with an early-stopping threshold of 9, the system needs only a fraction of the input data, averaging less than 100 rows. Specifically, datasets D1 and D3 are easier, requiring just 46.2 and 49.12 average time steps, respectively, to reach maximum accuracy with an early-stopping threshold of 9, as shown in Figure 8c.

5.4 FPGA accelerator throughput and energy consumption

We mapped our FPGA-based SCNN accelerator on a Xilinx Pynq7020 SoC and a KRIA KV260 SoM. Table 3 provides a detailed performance comparison with the throughput in images per second. In the SCNN accelerator, the 630x128 linear layer module significantly influences both latency and resource utilization due to its high requirement for memory accesses and arithmetic additions. The update of the membrane potential in this linear layer is not completely pipelined because of the limited LUT resources on the target FPGAs. Table 2 shows the resource utilization for the SCNN accelerator.

We use batch storage for both inputs and outputs to enhance the utilization rate during inference. Specifically, we process inputs in batches of 100 images, which are queued in the input FIFO,

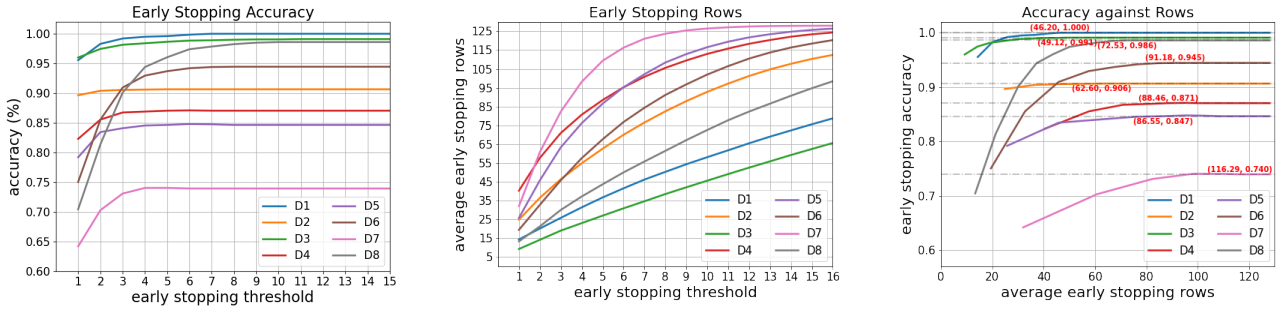

Figure 8: Classification using the early-stopping method. (a) Early-stopping accuracy for different threshold values: accuracy saturates with an early-stopping threshold of value 9. (b) The amount of processed rows increases with increased early-stopping threshold values. When the early-stopping threshold is 9 (i.e., at saturated accuracy), the average number of rows processed varies depending on the datasets, with a mean of about 86 processed rows. (c) Accuracy when using an early-stopping threshold equal to 9 versus the average number of rows processed for different datasets.

Table 2: FPGA Resource Utilization for the PYNQ-Z2 (xc7z020clg400) and KRIA-KV260 (xck26-c) boards. In percentage, we list the resource utilization versus the total board's resources (per type).

FPGA	LUT (%)	LUTRAM (%)	FF (%)	BRAM (%)	URAM (%)	DSP (%)	BUFG (%)
PYNQ-Z2	31685 (59.56%)	7970 (45.80%)	31196 (29.32%)	134.5 (96.07%)	-	4 (1.82%)	1 (3.13%)
KRIA-KV260	32572 (27.81%)	4316 (7.49%)	29678 (12.67%)	43.5 (32.21%)	20 (31.25%)	4 (0.32%)	6 (1.7%)

Table 3: Performance comparison. FPGA power consumption is the sum of processing system (PS) and programmable logic (PL) powers.

Model	Platform	Power (W)	Bandwidth	Accuracy	Throughput	Throughput (early-stopping)
SweepNet	AMD EPYC 7402P 24Cores CPU @ 2.80GHz	180	1.02 MB/s	1-0.79	503 img/sec	-
SweepNet	RTX 3080 Ti GPU	350	1.10 MB/s	1-0.79	540 img/sec	-
SCNN	A100 GPU	380	1.31 MB/s	1-0.74	648 img/sec	-
SCNN	AMD EPYC 7402P 24Cores CPU @ 2.80GHz	180	0.82 MB/s	1-0.74	405 img/sec	-
SCNN	Pynq 7020	2.0	0.49 MB/s	1-0.74	242 img/sec	358 img/sec
SCNN	Kria KV260	3.1	2.94 MB/s	1-0.74	1,452 img/sec	2,161 img/sec

while the accelerator's output is directed to an output FIFO. On the Pynq7020 board, our design achieves a throughput of 242 images per second. Moreover, the Kria KV260 board can accommodate three pipelines at twice the clock frequency (i.e., 100Mhz) of the Pynq7020 board, which runs at 50MHz, delivering an expected six-fold increase in throughput, up to 1452 images per second. This represents a 2.69-fold improvement over SweepNet's throughput of 540 images per second on a GeForce RTX 3080 Ti GPU. Regarding power efficiency, our design consumes 2W on the Pynq board and 2.958W on the Kria KV260 board, in stark contrast to the 3080Ti's consumption of 350W. Table 3 reports the throughput based on full-frame processing and with early-stopping. Performance with early stopping is estimated with an early stopping threshold equal to 9, with an average increase in speed of 1.48X.

6 CONCLUSION AND FUTURE WORK

We presented a novel SCNN architecture for classifying evolutionary genomic segments, including selective sweeps and recombination hotspots. Our model achieves accurate predictions by employing an early-stopping inference mechanism, allowing it to process only a fraction of the input data, thus eliminating the need to parse the entire sample size. We mapped the proposed SCNN on modern FPGA devices, achieving 3x higher throughput than a high-end GPU and over 100x higher power efficiency. Finally, we released our code and digital hardware architecture to the public. This work addresses genomic segment classification; however, we plan to employ the FPGA-based SCNN accelerator system for detecting selective sweeps and recombination hotspots across entire genomes.

ACKNOWLEDGMENTS

This work was Funded by the Volkswagen Foundation.

REFERENCES

[1] Nikolaos Alachiotis and Pavlos Pavlidis. 2018. RAiSD detects positive selection based on multiple signatures of a selective sweep and SNP vectors. *Communications biology* 1, 1 (2018), 79.

[2] Nikolaos Alachiotis, Alexandros Stamatakis, and Pavlos Pavlidis. 2012. OmegaPlus: a scalable tool for rapid detection of selective sweeps in whole-genome datasets. *Bioinformatics* 28, 17 (2012), 2274–2275.

[3] Nikolaos Alachiotis, Dimitris Theodoropoulos, and Dionisios Pnevmatikatos. 2017. Versatile deployment of FPGA accelerators in disaggregated data centers: A bioinformatics case study. In *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 1–4.

[4] Nikolaos Alachiotis, Charalampos Vatsolakis, Grigorios Chrysos, and Dionisios Pnevmatikatos. 2019. Raisd-x: A fast and accurate fpga system for the detection of positive selection in thousands of genomes. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 13, 1 (2019), 1–30.

[5] Ryan M Ames, Daniel Money, Vikramsinh P Ghatge, Simon Whelan, and Simon C Lovell. 2012. Determining the evolutionary history of gene families. *Bioinformatics* 28, 1 (2012), 48–55.

[6] Myat Thu Linn Aung, Daniel Gerlinghoff, Chuping Qu, Liwei Yang, Tian Huang, Rick Siow Mong Goh, Tao Luo, and Weng-Fai Wong. 2023. DeepFire2: A Convolutional Spiking Neural Network Accelerator on FPGAs. *IEEE Trans. Comput.* 99 (2023), 1–11.

[7] Dighanchal Banerjee, Smriti Rani, Arun M George, Arijit Chowdhury, Sounak Dey, Arijit Mukherjee, Tapas Chakravarty, and Arpan Pal. 2020. Application of spiking neural networks for action recognition from radar data. In *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–10.

[8] Dennis A Benson, Mark Cavanaugh, Karen Clark, Ilene Karsch-Mizrachi, David J Lipman, James Ostell, and Eric W Sayers. 2012. GenBank. *Nucleic acids research* 41, D1 (2012), D36–D42.

[9] Elliott Binder, Tze Meng Low, and Doru Thom Popovici. 2019. A portable gpu framework for snp comparisons. In *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 199–208.

[10] Katherine S Button, John PA Ioannidis, Claire Mokrysz, Brian A Nosek, Jonathan Flint, Emma SJ Robinson, and Marcus R Munafò. 2013. Power failure: why small sample size undermines the reliability of neuroscience. *Nature reviews neuroscience* 14, 5 (2013), 365–376.

[11] Christopher C Chang, Carson C Chow, Laurent CAM Tellier, Shashaank Vattikuti, Shaun M Purcell, and James J Lee. 2015. Second-generation PLINK: rising to the challenge of larger and richer datasets. *Gigascience* 4, 1 (2015), s13742–015.

[12] 1000 Genomes Project Consortium et al. 2015. A global reference for human genetic variation. *Nature* 526, 7571 (2015), 68.

[13] Federico Corradi, Guido Adriaans, and Sander Stuijk. 2021. Gyro: A digital spiking neural network architecture for multi-sensory data analytics. In *Proceedings of the 2021 Drone Systems Engineering and Rapid Simulation and Performance Evaluation: Methods and Tools Proceedings*. 9–15.

[14] Michael DeGiorgio, Christian D Huber, Melissa J Hubisz, Ines Hellmann, and Rasmus Nielsen. 2016. SweepFinder2: increased sensitivity, robustness and flexibility. *Bioinformatics* 32, 12 (2016), 1895–1897.

[15] Jason K Eshraghian, Max Ward, Emre O Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D Lu. 2023. Training spiking neural networks using lessons from deep learning. *Proc. IEEE* (2023).

[16] Haowen Fang, Zaidao Mei, Amar Shrestha, Ziyi Zhao, Yilan Li, and Qinru Qiu. 2020. Encoding, model, and architecture: Systematic optimization for spiking neural network in FPGAs. In *Proceedings of the 39th International Conference on Computer-Aided Design*. 1–9.

[17] Justin C Fay and Chung-I Wu. 2000. Hitchhiking under positive Darwinian selection. *Genetics* 155, 3 (2000), 1405–1413.

[18] Lex Flagel, Yaniv Brandvain, and Daniel R Schrider. 2019. The unreasonable effectiveness of convolutional neural networks in population genetic inference. *Molecular biology and evolution* 36, 2 (2019), 220–238.

[19] Jing Guo, Ritika Jain, Peng Yang, Rui Fan, Chee Keong Kwoh, and Jie Zheng. 2014. Reliable and Fast Estimation of Recombination Rates by Convergence Diagnosis and Parallel Markov Chain Monte Carlo. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 11, 1 (2014), 63–72. https://doi.org/10.1109/TCBB.2013.133

[20] Garrett Hellenthal and Matthew Stephens. 2007. msHOT: modifying Hudson's ms simulator to incorporate crossover and gene conversion hotspots. *Bioinformatics* 23, 4 (2007).

[21] Richard R Hudson. 2002. Generating samples under a Wright–Fisher neutral model of genetic variation. *Bioinformatics* 18, 2 (2002), 337–338.

[22] Mohammad Amin Imani, Arash Ahmadi, Mazdak RadMalekshahi, and Saeed Haghiri. 2018. Digital multiplierless realization of coupled Wilson neuron model. *IEEE Transactions on Biomedical Circuits and Systems* 12, 6 (2018), 1431–1439.

[23] Hasan Irmak, Federico Corradi, Paul Detterer, Nikolaos Alachiotis, and Daniel Ziener. 2021. A dynamic reconfigurable architecture for hybrid spiking and convolutional fpga-based neural network designs. *Journal of Low Power Electronics and Applications* 11, 3 (2021), 32.

[24] Alec J Jeffreys, Liisa Kauppi, and Rita Neumann. 2001. Intensely punctate meiotic recombination in the class II region of the major histocompatibility complex. *Nature genetics* 29, 2 (2001), 217–222.

[25] Alireza Khodamoradi, Kristof Denolf, and Ryan Kastner. 2021. S2n2: A fpga accelerator for streaming spiking neural networks. In *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. 194–205.

[26] Jeeson Kim, Vladmir Kornijcuk, Changmin Ye, and Doo Seok Jeong. 2020. Hardware-efficient emulation of leaky integrate-and-fire model using template-scaling-based exponential function approximation. *IEEE Transactions on Circuits and Systems I: Regular Papers* 68, 1 (2020), 350–362.

[27] Yuseob Kim and Rasmus Nielsen. 2004. Linkage disequilibrium as a signature of selective sweeps. *Genetics* 167, 3 (2004), 1513–1524.

[28] Motoo Kimura. 1969. The number of heterozygous nucleotide sites maintained in a finite population due to steady flux of mutations. *Genetics* 61, 4 (1969), 893.

[29] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[30] Gianluca Leone, Luigi Raffo, and Paolo Meloni. 2023. On-FPGA Spiking Neural Networks for End-to-End Neural Decoding. *IEEE Access* (2023).

[31] Alejandro Linares-Barranco, Fernando Perez-Peña, Diederik Paul Moeys, Francisco Gomez-Rodriguez, Gabriel Jimenez-Moreno, Shih-Chii Liu, and Tobi Delbruck. 2019. Low latency event-based filtering and feature extraction for dynamic vision sensors in real-time FPGA applications. *IEEE Access* 7 (2019), 134926–134942.

[32] Idir Mellal, David Crompton, Milos Popovic, and Milad Lankarany. 2021. A Flexible FPGA Implementation of Morris-Lecar Neuron for Reproducing Different Neuronal Behaviors. In *International Conference on Neuromorphic Systems 2021*. 1–5.

[33] Durgesh Nandini, Elisa Capecci, Lucien Koefoed, Ibai Laña, Gautam Kishore Shahi, and Nikola Kasabov. 2018. Modelling and analysis of temporal gene expression data using spiking neural networks. In *Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part I 25*. Springer, 571–581.

[34] Arnaud Nguembang Fadja, Fabrizio Riguzzi, Giorgio Bertorelle, and Emiliano Trucchi. 2021. Identification of natural selection in genomic data with deep convolutional neural network. *BioData Mining* 14, 1 (2021), 1–18.

[35] Moslem Nouri, Gh R Karimi, Arash Ahmadi, and Derek Abbott. 2015. Digital multiplierless implementation of the biological FitzHugh–Nagumo model. *Neurocomputing* 165 (2015), 468–476.

[36] Sathish Panchapakesan, Zhenman Fang, and Jian Li. 2022. SyncNN: Evaluating and accelerating spiking neural networks on FPGAs. *ACM Transactions on Reconfigurable Technology and Systems* 15, 4 (2022), 1–27.

[37] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[38] Pavlos Pavlidis, Daniel Živković, Alexandros Stamatakis, and Nikolaos Alachiotis. 2013. SweeD: likelihood-based detection of selective sweeps in thousands of genomes. *Molecular biology and evolution* 30, 9 (2013), 2224–2234.

[39] Gerd Pfeiffer, Stefan Baumgart, Jan Schröder, and Manfred Schimmler. 2009. A massively parallel architecture for bioinformatics. In *Computational Science–ICCS 2009: 9th International Conference Baton Rouge, LA, USA, May 25-27, 2009 Proceedings, Part I 9*. Springer, 994–1003.

[40] Kenneth L Rice, Mohammad A Bhuiyan, Tarek M Taha, Christopher N Vutsinas, and Melissa C Smith. 2009. FPGA implementation of Izhikevich spiking neural networks for character recognition. In *2009 International Conference on Reconfigurable Computing and FPGAs*. IEEE, 451–456.

[41] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, and Michael Pfeiffer. 2016. Theory and tools for the conversion of analog to spiking convolutional neural networks. *arXiv preprint arXiv:1612.04052* (2016).

[42] Arlina Shen, Han Fu, Kevin He, and Hui Jiang. 2019. False discovery rate control in cancer biomarker selection using knockoffs. *Cancers* 11, 6 (2019), 744.

[43] Yuelong Shu and John McCauley. 2017. GISAID: Global initiative on sharing all influenza data–from vision to reality. *Eurosurveillance* 22, 13 (2017), 30494.

[44] Kosuke M Teshima and Hideki Innan. 2009. mbs: modifying Hudson's ms software to generate samples of DNA sequences with a biallelic site under selection. *BMC bioinformatics* 10 (2009), 1–4.

[45] Charalampos Theodoris, Tze Meng Low, Pavlos Pavlidis, and Nikolaos Alachiotis. 2021. quickLD: An efficient software for linkage disequilibrium analyses. *Molecular Ecology Resources* 21, 7 (2021), 2580–2587.

[46] Clare Turnbull, Richard H Scott, Ellen Thomas, Louise Jones, Nirupa Murugaesu, Freya Boardman Pretty, Dina Halai, Emma Baple, Clare Craig, Angela Hamblin, et al. 2018. The 100 000 Genomes Project: bringing whole genome sequencing to the NHS. *Bmj* 361 (2018).

[47] Yue Wang, Guimei Liu, Mengling Feng, and Limsoon Wong. 2011. An empirical comparison of several recent epistatic interaction detection methods. *Bioinformatics* 27, 21 (2011), 2936–2943.

[48] Lars Wienbrandt, Jan Christian Kässens, Jorge González-Domínguez, Bertil Schmidt, David Ellinghaus, and Manfred Schimmler. 2014. FPGA-based acceleration of detecting statistical epistasis in GWAS. *Procedia Computer Science* 29 (2014), 220–230.

[49] Jiadong Wu, Yinan Wang, Lun Lu, Changlin Chen, and Zhiwei Li. 2023. A High-speed and Low-power FPGA Implementation of Spiking Convolutional Neural Network Using Logarithmic Quantization. In *2023 19th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*. IEEE, 1–8.

[50] Safa Yaghini Bonabi, Hassan Asgharian, Saeed Safari, and Majid Nili Ahmadabadi. 2014. FPGA implementation of a biological neural network based on the Hodgkin-Huxley neuron model. *Frontiers in neuroscience* 8 (2014), 379.

[51] 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.

[52] Bojian Yin, Federico Corradi, and Sander M Bohté. 2021. Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks. *Nature Machine Intelligence* 3, 10 (2021), 905–913.

[53] Bojian Yin, Federico Corradi, and Sander M Bohté. 2023. Accurate online training of dynamical spiking neural networks through Forward Propagation Through Time. *Nature Machine Intelligence* (2023), 1–10.

[54] Ling Sing Yung, Can Yang, Xiang Wan, and Weichuan Yu. 2011. GBOOST: a GPU-based tool for detecting gene–gene interactions in genome–wide case control studies. *Bioinformatics* 27, 9 (2011), 1309–1310.

[55] Hanqing Zhao and Nikolaos Alachiotis. 2023. Effective Data Preprocessing Techniques for CNN-based Selective Sweep Detection. In *2023 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 793–800.

[56] Hanqing Zhao, Pavlos Pavlidis, and Nikolaos Alachiotis. 2023. SweepNet: A Lightweight CNN Architecture for the Classification of Adaptive Genomic Regions. In *Proceedings of the Platform for Advanced Scientific Computing Conference*. 1–10.

[57] Hanqing Zhao, Matthijs Souilljee, Pavlos Pavlidis, and Nikolaos Alachiotis. 2023. Genome-wide scans for selective sweeps using convolutional neural networks. *Bioinformatics* 39, Supplement_1 (2023), i194–i203.

[58] Qian Zhou, Saibing Qi, and Cong Ren. 2021. Gene essentiality prediction based on chaos game representation and spiking neural networks. *Chaos, Solitons & Fractals* 144 (2021), 110649.

[59] Qian Zhou, Yan Shi, Zhenghua Xu, Ruowei Qu, and Guizhi Xu. 2020. Classifying melanoma skin lesions using convolutional spiking neural networks with unsupervised stdp learning rule. *IEEE Access* 8 (2020), 101309–101319.

[60] Shibo Zhou, Ying Chen, Xiaohua Li, and Arindam Sanyal. 2020. Deep scnn-based real-time object detection for self-driving vehicles using lidar temporal data. *IEEE Access* 8 (2020), 76903–76912.