Efficient Kernels for Real-Time Position Decoding from In Vivo Calcium Images

Zhe Chen*[†], Jim Zhou*[†], Garrett J. Blair[‡], Hugh T. Blair[‡] and Jason Cong[†]
[†]*Computer Science Department*, [‡]*Department of Psychology*
UCLA, Los Angeles, U.S.

***Abstract*—Recent studies have found that the position of mice or rats can be decoded from calcium imaging of brain activity offline. However, given the complex analysis pipeline, real-time position decoding remains a challenging task, especially considering strict requirements on hardware usage and energy cost for closed-loop feedback applications. In this paper, we propose two neural network based methods and corresponding hardware designs for real-time position decoding from calcium images. Our methods are based on: 1) convolutional neural network (CNN), 2) spiking neural network (SNN) converted from the CNN. We implemented quantized CNN and SNN models on FPGA. Evaluation results show that the CNN and the SNN methods achieve 56.3%/83.1% and 56.0%/82.8% *Hit-1*/*Hit-3* accuracy for the position decoding across different rats, respectively. We also observed an accuracy-latency tradeoff of the SNN method in decoding positions under various time steps. Finally, we present our SNN implementation on the neuromorphic chip Loihi.**

***Index Terms*—calcium image, decoding, neural network**

I. Introduction

The miniaturized calcium imaging microscope (e.g. Miniscope) is an emerging device that can be implanted at certain brain region of a mouse or rat for monitoring a large population of cell activity while the animal is performing various behavioral tasks [1], [2]. Most existing calcium image processing pipelines focus on real-time trace extraction [3]–[6], whereas the real-time calcium image decoding remains a challenge, especially considering the strict hardware usage and energy cost requirements for closed-loop feedback applications.

The calcium image decoding has gained popularity in recent literatures. [7] employed Laplacian Eigenmaps to decode behavior states from a reduced dimensional space of neural activity. [8] used the ResNet18 model to detect the forelimb reaching activity of a mouse from mean calcium images. [9] proposed supervised and unsupervised methods for decoding mouse's position from the surrogate of spikes. [10] aimed at accomplishing real-time mouse movement detection with a support vector machine based decoder running on the CPU.

In this work, we target at real-time position decoding from calcium images recorded from rat running on a linear track, as Fig. 1 shows. We first propose the decoding methods based on a convolutional neural network (CNN) and a converted spiking neural network (SNN), respectively. Then we implement accelerator kernels for these methods on an FPGA device and benchmark the accuracy, hardware cost, and latency of these implementations. Finally, we further deploy the SNN based decoder on the neuromorphic chip Loihi [11] and report its performance on the latency and energy efficiency.

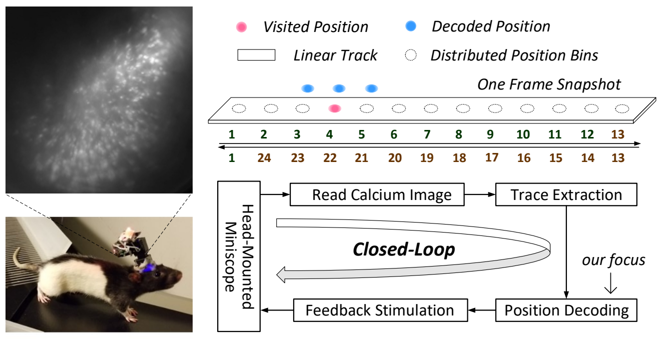

Fig. 1. Real-time position decoding of a rat's position on a linear track from the calcium imaging.

II. Background

A. Real-Time Calcium Image Processing

Conventional calcium image processing aims at extracting cell activity from the calcium image recording and usually consists of following consecutive steps [3]: The motion correction, the image enhancement and the trace extraction, as Fig. 2 illustrates. The motion correction eliminates the motion artifacts caused by the movement of the brain tissue during the recording. The image enhancement estimates the background fluorescence and gets rid of it to improve the signal-to-noise ratio. The trace extraction obtains fluorescence traces reflecting the cell firing activities.

We can conduct calcium image decoding in two ways: 1) First pre-process raw calcium images to extract fluorescence traces reflecting cell activities [6], and then apply a decoder to infer the position from the traces. 2) Bypass the pre-processing and decode positions from the raw calcium images directly. Compared to 2), the first approach greatly reduces the input dimension of the decoder and saves the computation time and hardware cost for both training and inference. In this paper, we aim at realizing real-time calcium image decoding based on 1). Fig. 2 shows our proposed calcium image processing pipeline. The data acquisition board (DAQ) [12] receives calcium images from the Miniscope and sends them to the pre-processing pipeline implemented on the FPGA. The decoder takes the extracted traces from the pre-processing and makes inference for the decoded position. We implement the decoder on both the FPGA and the Loihi.

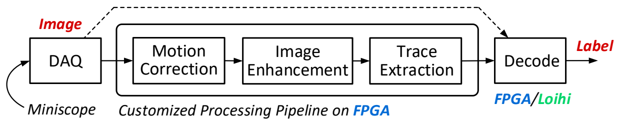

Fig. 2. Calcium image processing flow for the real-time position decoding.

B. Position Decoding Task

Fig. 1 illustrates the position-decoding task. During the experiments, the rat ran back and forth on a linear track, whereas an implanted Miniscope device [2] captured calcium images with 512×512 spatial resolution and 22.8 fps rate. A separate behavioral camera kept track of the rat's position on the linear track. We divided the track's positions into 24 evenly distributed bins by length. From left to right and right to left, we labelled the rat's positions with labels 1 to 12 and 13 to 24, respectively. We recorded data from 8 different rats from R1 to R8. For each rat, we collected 8000 frames of calcium images with aligned position labels. We then separated the collected calcium image data and position labels into training and test sets for the position decoding task.

III. Proposed Method

A. Accuracy Evaluation Metrics

We aim at a frame-based position decoding from in vivo calcium images. Fig. 3(a) presents a plot showing ground-truth and decoded positions of a rat running in an experiment, with positions labeled into 24 bins. The blue and red dots represent the ground-truth and decoded positions, respectively. We come up with a *Hit-N* metric to evaluate the decoding accuracy, which reflects the percentage of correct decoding among all trials. N represents the number of bins that we count as correct decoded positions around the ground-truth position. For example, if the ground-truth position label is 6, then we count all decoded position labels falling into the ± 1 neighborhood of 6, *i.e.*, 5, 6, and 7, as correct under the *Hit-3* metric. Fig. 3(b) visualizes the *Hit-1* to *Hit-9* metrics in a pie chart. Fig. 3(c) gives an example of accuracy evaluation on the decoding in Fig. 3(a) under different metrics.

B. CNN-Based Method

We first propose a CNN-based position decoding method. We explored two CNN models: the ResNet20, and a simple CNN model. The ResNet20 takes 32×32 image as input. Each image consists of 8×8 non-overlapping tiles, and each tile contains 4×4 pixels. We assigned a same trace value corresponding to a specific cell to all 16 pixels within a tile. In contrast, the simple CNN takes an $N \times N$ image (N=8 for comparison here) as input, and each pixel represents the trace value computed from a single cell. The simple CNN model is made up of one convolution (CONV) layer and one fully connected (FC) layer. The CONV layer contains 6 convolution kernels in 3×3 size, and the FC layer has all-to-all connection between (N-2)$\times$(N-2) hidden nodes and 24 output nodes. We

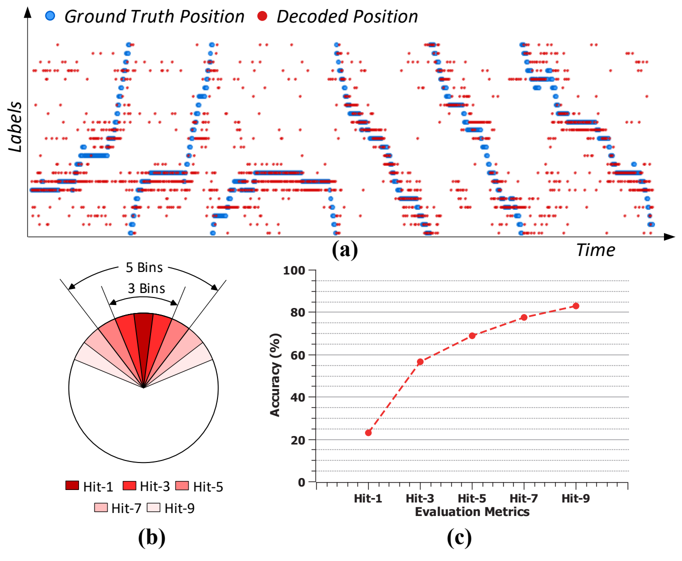

Fig. 3. (a) The ground-truth and decoded position labels throughout a recording session. (b) Proposed accuracy evaluation metrics for the position decoding task. (c) Accuracy evaluation on the decoding results shown in (a).

TABLE I
Comparison between the ResNet20 and the simple CNN Models

Model	# Layers	# Weights	Size	*Hit-3*
ResNet20	19 CONV + 1 FC	269,722	1.03 MB	56.0%
Simple CNN	1 CONV + 1 FC	5,250	5.13 KB	56.2%

trained these two CNN models and evaluated their accuracy using the same decoding data set collected from a rat. Table I shows comparison results on the network structure, memory footprint and decoding accuracy between these two models.

The simple CNN model achieves similar decoding accuracy as the ResNet20, whereas it reduces the model size and the parameter number significantly. This can be explained by a lack of hierarchical features in the calcium image, in which place cells fire as the rat passes specific locations on the linear track [9]. Considering the strict requirements on hardware cost and latency for closed-loop applications, we chose the simple CNN model for the position decoding. Further evaluation shows that 8-bit quantization does not cause significant accuracy loss for the simple CNN model.

C. SNN-Based Method

The second method we propose for the position decoding relies on a rate-based SNN model. We converted the SNN from a well-trained simple CNN model introduced in Section III-B using the SNN Toolbox [13]. Fig. 4 shows the architecture of the converted SNN model. We only converted the FC layer of the CNN. We first applied rate-based encoding at the input of the FC layer by converting hidden node values into spike sequences over a certain number of time steps based on the integrate-and-fire (IAF) model [14]. Then we fed the spike sequences to the output nodes in a fully connected fashion. The output nodes operated as the IAF neurons as well. We describe the converted SNN model [15] as follows:

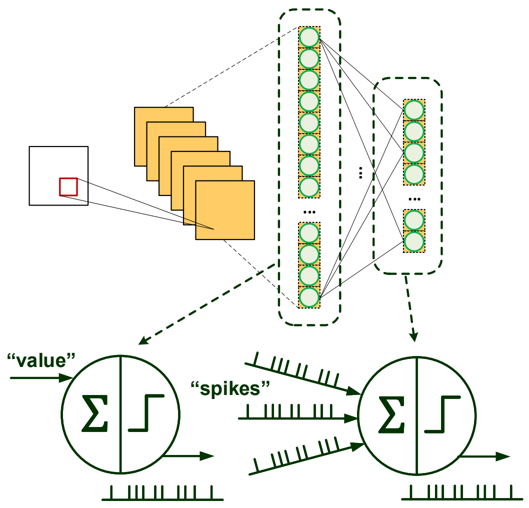

Fig. 4. Converted rate-based SNN model for the position decoding from calcium images.

We normalize each element of SNN input between 0 and 1, as conductance p_i. Suppose the membrane voltage is V_t, the SNN input current z_i^0 at time step t is computed as:

$$z_i^0(t) = V_t \times p_i, \;\; i \in [1, N_0] \tag{1}$$

For the layer L containing N_L nodes, weights and biases are w_{ij}^L and b_i^L. The output spike from layer $L-1$ at the time step t is $s_j^{L-1}(t)$. The current is then given by:

$$z_i^L(t) = V_t \left(\sum_{j=1}^{N_{L-1}} w_{ij}^L s_j^{L-1}(t) + b_i^L \right), \;\; i \in [1, N_L] \tag{2}$$

At each time step, each SNN neuron integrates its input spikes into current (and charge), which is added to its membrane potential $V_i^L(t)$. When the $V_i^L(t)$ reaches V_t, the neuron fires a spike. Subsequently, the neuron's membrane potential gets reset by subtracting V_t from itself. This allows the excess charge to be preserved for further spike generation, and improves the firing rate approximation of the SNN. The equations describing the membrane potential dynamics are:

$$\begin{aligned} V_i^L(t) &= V_i^L(t-1) + z_i^L(t) - s_i^{L-1}(t)V_t \\ s_i^L(t) &= \begin{cases} 1, & V_i^L(t) \geq V_t \\ 0, & V_i^L(t) < V_t \end{cases}, \;\; i \in [1, N_L] \end{aligned} \tag{3}$$

D. Accuracy Evaluation

We evaluated the decoding accuracy of a baseline floating-point CNN and a converted 8-bit SNN across datasets from 8 different rats on linear tracks (different from the dataset in Fig. 3). Table II summarizes the cell number and the input dimensions of the CNN and the SNN for these rats. Fig. 5 shows the *Hit-1* and *Hit-3* accuracy evaluation across test sets. The CNN/SNN achieves 56.3%/56.0% and 83.1%/82.8% *Hit-1* and *Hit-3* accuracy on average, respectively.

TABLE II
THE NUMBER OF DETECTED CELLS AND THE INPUT SIZE FOR THE CNN AND SNN MODELS AMONG RATS.

Rat	R1	R2	R3	R4
# Cells	153	194	296	309
Input Size	12×12	13×13	16×16	17×17
Rat	R5	R6	R7	R8
# Cells	317	322	643	760
Input Size	17×17	17×17	25×25	27×27

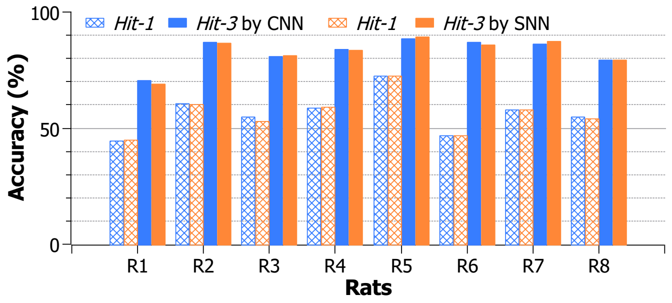

Fig. 5. *Hit-1* and *Hit-3* accuracy by CNN and SNN models across test sets.

We observed that reducing the SNN inference time steps shortens the latency but slightly degrades the decoding accuracy. Fig. 6 shows the averaged latency and accuracy achieved by the CNN and SNN decoders across the datasets. The latency increases linearly with the time steps, whereas the decoding accuracy gradually approaches that of the CNN decoder as the employed time step increases. This latency/accuracy tradeoff of the SNN model makes it flexible in adapting itself to applications with various inference speed requirements.

IV. Implementation

A. FPGA Accelerator Design

We designed accelerators for the CNN and SNN models using the Vitis HLS, targeting the low-power Ultra96 FPGA under 300 MHz clock frequency. We implemented the matrix

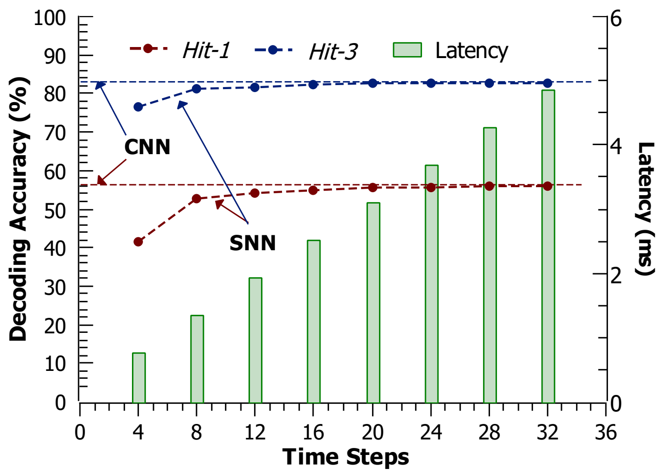

Fig. 6. Tradeoff on accuracy and latency for SNN converted from CNN.

vector multiplication in the CNN/SNN model with a multi-layer nested loop in the HLS design, and we applied the loop reordering and the loop pipelining to reduce the latency cycle count while keeping the hardware cost as low as possible. Our implementation achieves the initiation interval (II) [6] of 1 for all nested loops in the CNN and the SNN accelerator designs.

We quantized the weights of the CNN and SNN models into 8-bit fixed-point representation, and we verified that this quantization does not cause accuracy loss compared to the floating-point CNN design. We implemented the quantized CNN and SNN models with 8-bit multiplications and additions instead of expensive floating-point operations to save the hardware usage.

One limitation of a converted SNN is that the neuron's firing rates have to be kept proportional to the original activations in the CNN model in order to maintain the accuracy. This requires the input current of a neuron to be normalized between 0 and V_t. However, the normalization is computationally expensive because it involves division and must operate for each neuron at every time step. We applied a data-based normalization method [14], which takes average of the maximum current of each layer over the total analyzed time steps T, and makes use of it to rescale the weights and biases offline. It reduces the computation cost by keeping the V_t to be 1 and avoiding the normalization computation step during the online SNN inference.

For the layer L with input current $z_i^L(t)$, weights and biases are normalized as follows:

$$\begin{aligned} \bar{w}_{ij}^L &= w_{ij}^L/\beta^L \quad , \ \bar{b}_i^L = b_i^L/\beta^L \\ \beta^L &= \left(\sum_{t=1}^{T} \max\left(z_i^L(t) \middle| \, i \in [1, N_L]\right)\right)/T \end{aligned} \tag{4}$$

B. Deployment on Neuromorphic Hardware

To further improve the energy efficiency and latency of our proposed SNN model, we implemented it on Intel's Loihi [11], which is a customized programmable neuromorphic processor optimized for asynchronous SNN computation. We used the SNN Toolbox [13], which converted our SNN model to a Loihi-compatible form using the NxTF compiler [16]. The NxTF performs partitioning of the model across neurocores followed by a mapping of the model to the Loihi.

V. Results

A. Resource Utilization

We summarize the comparison of the resource utilization for the CNN and SNN based decoders in Table III. We report the resource utilization of the decoders for R1 and R8 rats, as these two rats have the minimum and maximum numbers of cells detected from their calcium image recordings, and correspond to the minimum and maximum hardware utilization on the decoder designs, respectively. Compared to its CNN based decoder, the SNN decoder saves on average 37.6%, 38.7%, and 80.0% LUT, FF and DSP resources.

TABLE III
Resource utilization of implemented decoders on the Ultra96

Models	CNN		SNN		Reduction Rate		
Rats	R1	R8	R1	R8	R1	R8	Average
LUT	2787	2526	1615	1612	42.1%	36.2%	37.6%
FF	2463	2492	1443	1555	41.4%	37.6%	38.7%
DSP	7	16	0	6	100%	62.5%	80.0%
BRAM	8	54	10	53	-	1.9%	-

TABLE IV
Latency and accuracy comparison among decoder implementations

Models	CNN		SNN		SNN	
Time Steps	1		4		32	
Rats	R1	R8	R1	R8	R1	R8
Latency (ms)	0.112	0.799	0.270	1.727	1.736	10.830
Hit-1 (%)	44.3	**54.6**	38.2	43.6	**44.8**	54.1
Hit-3 (%)	**70.6**	**79.4**	66.3	70.6	69.0	79.3

B. Latency and Accuracy

Table IV summarizes the comparison results on the latency and the *Hit-1* and *Hit-3* accuracy achieved by the CNN and SNN decoders implemented on the FPGA. The SNN decoder can operate with different time steps. Under 32 time steps, the SNN decoder achieves similar accuracy as the CNN decoder, though it suffers a longer latency. In this case, the main benefit of the SNN decoder is the less hardware usage. Under 4 time steps, the SNN decoder largely reduces the latency, but it experiences slight accuracy degradation on the decoding. For applications that have strict requirements on hardware usage and latency but can tolerate small accuracy loss, the SNN decoder operating with short time steps is a more appropriate.

C. Performance and Energy Efficiency on Loihi

For the position decoding on the R1 and the R8 rats, the SNN models consumed 4 (3.1%) and 15 (11.7%) neurocores on the Loihi and took 3.02 ms and 6.94 ms for each inference, with 94.8 µJ/inf and 258 µJ/inf energy efficiency, respectively. Compared to the FPGA based SNN accelerator, the SNN implemented on the Loihi has 1.56x speedup on the position decoding for the R8 rat, with the same decoding accuracy.

VI. Conclusion

In this paper, we proposed the methods of combining the CNN and the SNN inference with the calcium image preprocessing pipeline for realizing the real-time position decoding. The SNN decoder consumes less hardware cost while maintaining similar accuracy compared to its CNN counterpart, and it offers a unique accuracy-latency tradeoff feature. Given its capability to be mapped onto the neuromorphic hardware, the SNN decoder remains promising in achieving real-time calcium image based position decoding with high energy efficiency for future closed-loop feedback applications.

Acknowledgment

This work is supported by NSF Awards DBI-1707408, CCF-1723773 (REU supplement), the SRC Education Alliance for Undergraduate Student Support, and NIH Grant MH122800.

References

[1] K. K. Ghosh, L. D. Burns, E. D. Cocker, A. Nimmerjahn, Y. Ziv, A. El Gamal and et al., “Miniaturized integration of a fluorescence microscope,” Nature Methods, vol. 8(10), pp. 871–878, 2011.

[2] D. Aharoni, B. S. Khakh, A. J. Silva and P. Golshani, “All the light that we can see: a new era in miniaturized microscopy,” Nature Methods, vol. 16(1), pp. 11–13, 2019.

[3] J. Friedrich, A. Giovannucci and E. A. Pnevmatikakis, “Online analysis of microendoscopic 1-photon calcium imaging data streams,” PLoS Comput. Biology, vol. 17(1), e1008565, 2021.

[4] Y. Lee, J. Xie, E. Lee, S. Sudarsanan, D. T. Lin, R. Chen and et al., “Real-time neuron detection and neural signal extraction platform for miniature calcium imaging,” Front. Comput. Neurosci., vol. 14(43), 2020.

[5] Z. Chen, A. Howe, H. T. Blair and J. Cong, “BLINK: Bit-sparse LSTM inference kernel enabling efficient calcium trace extraction for neurofeedback devices” Proc. Int. Symp. Low Power Electron. Des. (ISLPED), 2020.

[6] Z. Chen, G. J. Blair, H. T. Blair and J. Cong, “Fast calcium trace extraction for large-field-of-view miniscope” IEEE Biomed. Circuits Syst. Conf. (BioCAS), 2021.

[7] A. Rubin, L. Sheintuch, N. Brande-Eilat, O. Pinchasof, Y. Rechavi, N. Geva and et al., “Revealing neural correlates of behavior without behavioral measurements,” Nature Commun., vol. 10(1), 4745, 2019.

[8] C. Li, D. C. W. Chan, X. Yang, Y. Ke and W.-H. Yung, “Prediction of forelimb reach results from motor cortex activities based on calcium imaging and deep learning,” Front. Cellular Neurosci., vol. 13(88), 2019.

[9] M. Tu, R. Zhao, A. Adler, W.-B. Gan and Z. S. Chen, “Efficient position decoding methods based on fluorescence calcium imaging in the mouse hippocampus.” Neural Comput., vol. 32(6), pp. 1144–1167, 2020.

[10] K. Lee, Y. Lee, D.-T. Lin, S. S. Bhattacharyya and R. Chen, “Real-time calcium imaging based neural decoding with a support vector machine,” IEEE Biomed. Circuits Syst. Conf. (BioCAS), 2019.

[11] M. Davies, N. Srinivasa, T. H. Lin, G. Chinya, Y. Cao, S. H. Choday and et al., “Loihi: A neuromorphic manycore processor with on-chip learning,” IEEE Micro, vol. 38(1), pp. 82–99, 2018.

[12] Data Acquisition Box. [Online]. Available: http://miniscope.org/

[13] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer and S.-C. Liu, “Conversion of continuous-valued deep networks to efficient event-driven networks for image classification,” Front. Neurosci., vol. 11, 682, 2017.

[14] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu and M. Pfeiffer, “Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing.” Int. Joint Conf. on Neural Networks (IJCNN), 2015.

[15] Y. Cao, Y. Chen, D. Khosla, “Spiking deep convolutional neural networks for energy-efficient object recognition” Int. J. Compt. Vision, vol. 113(1), pp. 54–66, 2015.

[16] B. Rueckauer, C. Bybee, R. Goettsche, Y. Singh, J. Mishra and A. Wild, “NxTF: An API and compiler for deep spiking neural networks on Intel Loihi.” ArXiv:2101.04261.