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. 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.

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.
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 × 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) × (N-2) hidden nodes and 24 output nodes. We 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:
We normalize each element of SNN input between 0 and 1, as conductance \( p_i \). Suppose the membrane voltage is \( V_i \), the SNN input current \( z_i^0 \) at time step \( t \) is computed as:

\[
z_i^0(t) = V_i \times p_i, \quad i \in [1, N_0]
\]

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

\[
z_i^L(t) = V_i \left( \sum_{j=1}^{N_{L-1}} w_{ij}^L s_j^{L-1}(t) + b_j^L \right), \quad i \in [1, N_L]
\]

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 \( V_i \), the neuron fires a spike. Subsequently, the neuron’s membrane potential gets reset by subtracting \( V_i \) 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:

\[
V_i^L(t) = V_i^L(t - 1) + z_i^L(t) - s_i^{L-1}(t) V_i
\]

\[
s_i^L(t) = \begin{cases} 1, & V_i^L(t) \geq V_i \\ 0, & V_i^L(t) < V_i \end{cases}, \quad i \in [1, N_L]
\]

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

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

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

Table II: The number of detected cells and the input size for the CNN and SNN models among rats.

<table>
<thead>
<tr>
<th>Rat</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
</tr>
</thead>
<tbody>
<tr>
<td># Cells</td>
<td>153</td>
<td>194</td>
<td>296</td>
<td>309</td>
</tr>
<tr>
<td>Input Size</td>
<td>12×12</td>
<td>13×13</td>
<td>16×16</td>
<td>17×17</td>
</tr>
<tr>
<td>Rat</td>
<td>R5</td>
<td>R6</td>
<td>R7</td>
<td>R8</td>
</tr>
<tr>
<td># Cells</td>
<td>317</td>
<td>322</td>
<td>643</td>
<td>760</td>
</tr>
<tr>
<td>Input Size</td>
<td>17×17</td>
<td>17×17</td>
<td>25×25</td>
<td>27×27</td>
</tr>
</tbody>
</table>
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 \( z^L_t(t) \), weights and biases are normalized as follows:

\[
\bar{w}_{ij}^L = \frac{w_{ij}^L}{\beta^L}, \quad \bar{b}_i^L = \frac{b_i^L}{\beta^L}
\]

\[
\beta^L = \left( \sum_{i=1}^{N_L} \max\left(z^L_{it}(t) \mid i \in [1, N_L]\right) \right)/T \tag{4}
\]

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.

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


