algorithms

Article

Image Edge Detector with Gabor Type Filters Using a Spiking Neural Network of Biologically Inspired Neurons

Krishnamurthy V. Vemuru

Riverside Research, 2900 Crystal Dr., Arlington, VA 22202, USA; kvemuru@riversideresearch.org or kvv5cf@virginia.edu; Tel.: +1-703-908-2145

Received: 16 June 2020; Accepted: 6 July 2020; Published: 9 July 2020

Abstract: We report the design of a Spiking Neural Network (SNN) edge detector with biologically inspired neurons that has a conceptual similarity with both Hodgkin-Huxley (HH) model neurons and Leaky Integrate-and-Fire (LIF) neurons. The computation of the membrane potential, which is used to determine the occurrence or absence of spike events, at each time step, is carried out by using the analytical solution to a simplified version of the HH neuron model. We find that the SNN based edge detector detects more edge pixels in images than those obtained by a Sobel edge detector. We designed a pipeline for image classification with a low-exposure frame simulation layer, SNN edge detection layers as pre-processing layers and a Convolutional Neural Network (CNN) as a classification module. We tested this pipeline for the task of classification with the Digits dataset, which is available in MATLAB. We find that the SNN based edge detection layer increases the image classification accuracy at lower exposure times, that is, for $1 < t < T/4$, where t is the number of milliseconds in a simulated exposure frame and T is the total exposure time, with reference to a Sobel edge or Canny edge detection layer in the pipeline. These results pave the way for developing novel cognitive neuromorphic computing architectures for millisecond timescale detection and object classification applications using event or spike cameras.

Keywords: edge detection; spiking neural networks; bio-inspired image processing; computational photography; machine learning and classification

1. Introduction

Advances in computational neuroscience and statistical learning have served as an inspiration for artificial neural networks, leading to a Cambrian explosion in the field of machine learning with deep learning architectures and algorithms. However, only a limited set of neural principles have been fully explored in deep learning architectures. Novel approaches which incorporate models closer to brain circuits and mechanisms into neural networks are key to extending the state-of-the-art in deep learning [1–9]. Brain based object recognition is one key area advancing the state-of-the-art in computer vision. The goal of this work is to understand how a spike based algorithm can detect edges in scenes compared to human engineered image processing techniques and introduce those ideas into deep learning architectures. Our general approach for improving deep learning is modular. We introduce one brain-inspired function that is, spike based computing, at a time or in one layer in the network and understand how it effects the overall network. Our approach does not result in a complete brain-like object recognition in one shot, but we hope it contributes to the transition process from artificial neural networks to brain-like computing neural networks which mimic neuron units as computing blocks.

Artificial neural networks (ANNs) are based on neurons that use static and continuous valued non-linear activation functions. Biological neurons use spikes to encode, compute, and transmit information using sparse spiking processes, defined by spike timing and spike firing rates. As a result, biological neurons are mostly inactive and consume energy only when there are spike events. Spiking neural networks use neuron models that have a biologically inspired computational or functional representation and are biologically more realistic than ANNs. Spiking neural networks are event driven and are more energy-efficient than ANNs, and are thus suitable for lower Size, Weight and Power (SWaP) artificial intelligence devices based on neuromorphic computing architectures such as IBM TrueNorth [10]. The Hodgkin–Huxley neuron model is the most realistic model of the neuron [11]. While accurate, this model is computationally inefficient. The Izhikevich neuron model is a simplified model based on Hodgkin-Huxley equations. It can successfully reproduce all known neuron firing patterns with a better computational efficiency than the Hodgkin–Huxley model [12]. Hodgkin-Huxley model neurons also play an important role in the research aimed at understanding the efficient processing of images and learning visual orientations in the visual cortex [13].

We have investigated the scope for computationally efficient use of Hodgkin-Huxley neurons in a spiking neural network for low-SWaP applications in image processing, such as edge detection and machine learning. The state-of-the-art in Spiking Neural Network (SNN) architectures is inspired by Leaky-Integrate-and-Fire type neurons [14,15]. The Leaky-Integrate-and-Fire neuron model has been extensively used in research. Close variants of the Leaky-Integrate-and-Fire neuron mechanisms have been implemented in cognitive neuromorphic architectures (CNAs), for example in IBM-TrueNorth [10] and in Intel-Loihi [16] neuromorphic processors. The Leaky Integrate-and-Fire model and the Hodgkin-Huxley model are on the opposite ends of the spectrum in terms of accuracy, where the former is simple but lacks detail, and the latter more complicated but also much more accurate. Present models of neurons in CNAs solve a differential equation for the membrane potential at each time step, and it is important to explore ways to incorporate more biologically possible or accurate neuron models for cutting edge applications of artificial intelligence such as geospatial analytics, robotic delivery, transit safety and multi-modal intelligence fusion.

2. Background

2.1. Edge Detection Algorithms

Edge detection has been a popular area of research in computer vision. The edge detectors presented by Roberts [17], Canny [18] and Sobel [19] are commonly used for image processing applications across domains. An edge in an image is a significant local change in the image intensity at the pixel level, usually associated with a discontinuity in either the image intensity or the first derivative of the image intensity across pixels. An edge detector is an algorithm that produces a set of edge points or edge fragments from an image. Abrupt changes in images are either step discontinuities or line discontinuities. The smoothing introduced by most sensors filters out high-frequency components, hence sharp discontinuities rarely exist in images. One way of detecting step edges in images is to find points that have locally large gradient magnitudes. There is a trade-off between noise suppression and edge localization while detecting edges using an edge detection operator. The Canny edge detector uses the first derivative of a Gaussian as an operator to optimize the signal-to-noise ratio and edge localization.

2.2. Biologically Inspired Neuron Models

The model presented below was originally introduced by Hodgkin and Huxley to investigate the behavior of a giant squid axon [11]. It has been adopted by many researchers as the benchmark model for neural activity. Figure 1 shows the equivalent electrical circuits for the Hodgkin-Huxley (HH) model and the Leaky Integrate-and-Fire (LIF) neuron model. One can see the HH model can be reduced to the LIF model by removing the two branches of circuit representing the conductance of Na and K ion channels. In this circuit, C is the capacitance per unit area of the lipid bi-layer (membrane) separating the ions on the inside and the outside of the cell. The fixed resistance symbol indicates the current for a non-specific leak. Variable resistances indicate the voltage dependence of the conductance on the Na$^+$ and K$^+$ ions. The *emf* indicates the driving force for the ions, which is given in the model by the difference between membrane potential, $V = V_{in} - V_{out}$ and reversal potential. I is the total membrane current per unit area, G_K and G_{Na} are the potassium conductance per unit area and sodium conductance per unit area, respectively. V_K and V_{Na} are the potassium and sodium reverse potentials, respectively. G_L and V_L are the leak conductance per unit area, and leak reverse voltage, respectively. The action potential of the axon follows the equations:

$$C\frac{dV}{dt} = I - G_K n^4(V - V_K) - G_{Na}m^3h(V - V_{Na}) - G_L(V - V_L) \tag{1}$$

$$\frac{dn}{dt} = \alpha_n(V)(1 - n) - \beta_n(V)n \tag{2}$$

$$\frac{dm}{dt} = \alpha_m(V)(1 - m) - \beta_m(V)m \tag{3}$$

$$\frac{dh}{dt} = \alpha_h(V)(1 - h) - \beta_h(V)h, \tag{4}$$

where, m and h are the activation and the inactivation variables for the Na$^+$ channels, respectively; n is the K$^+$ inactivation variable; the α and β rates are functions of V at temperature T = 6.3 °C [11]. n, m and h are dimensionless quantities ranging from 0 to 1. These four partial differential equations quantitatively describe the changes in the membrane as functions of space and time. We use the conductance model to approximate the synaptic activity using the equation [20]:

$$I = g_{ex}(V - E_{ex}) + g_{in}(V - E_{in}), \tag{5}$$

where g_{ex} and g_{in} are, time-dependent excitatory and inhibitory conductances, respectively. E_{ex} and E_{in} are the reverse potentials for excitatory and inhibitory synapses, respectively.

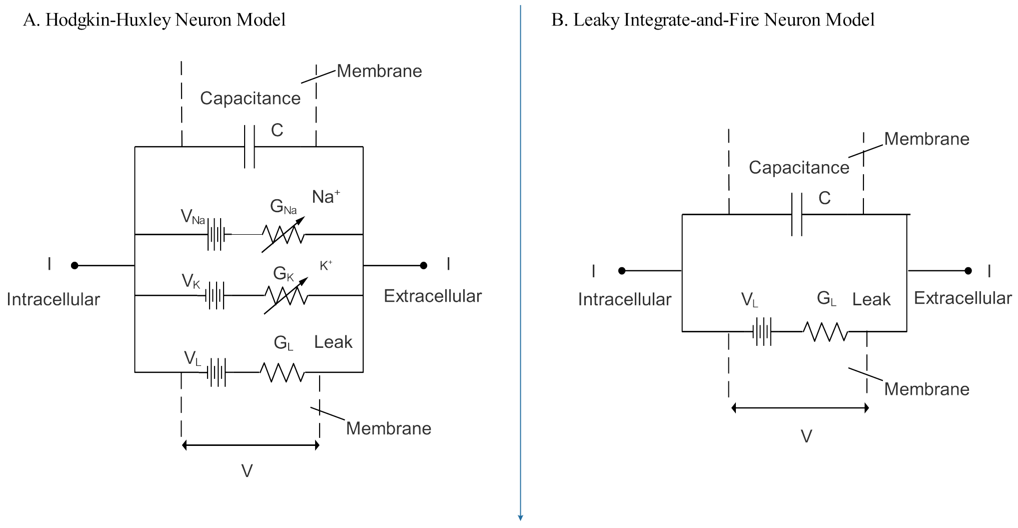

Figure 1. The electrical equivalent circuit of (**A**) Hodgkin-Huxley neuron model and (**B**) Leaky Integrate-and-Fire neuron model.

Img is the normalized intensity of the image in grayscale. The peak conductance is set equal to the image intensity: $q_{ex} = Img$, $q_{in} = Img$ and synaptic decay times for excitatory and inhibitory synapses are set to $\tau_{ex} = 4$ ms, $\tau_{in} = 10$ ms, respectively.

The time dependency of the conductances is given by two differential equations:

$$\frac{g_{ex}}{dt} = -\frac{1}{\tau_{ex}} g_{ex} + \sum_{r=-R}^{R} \sum_{f=-R}^{R} \frac{W_{ex}(r,f)}{A_{ex}} q_{ex}(x-r, y-f) \tag{6}$$

$$\frac{g_{in}}{dt} = -\frac{1}{\tau_{in}} g_{in} + \sum_{r=-R}^{R} \sum_{f=-R}^{R} \frac{W_{in}(r,f)}{A_{in}} q_{in}(x-r, y-f), \tag{7}$$

where A_{ex} = 0.0141 mV, A_{in} = 0.0281 mV. $W_{ex}(r, f)$ and $W_{in}(r, f)$ are the Gabor type filters representing synaptic weight matrices with synaptic radius R for excitatory and inhibitory synapses, respectively. We solve the above equation for the conductances in a loop over a time window T = 50 ms with the time step dt = 1 ms with updates to the total synaptic current given by the equation:

$$I_z = -g_{ex}E_{ex} - g_{in}E_{in}. \tag{8}$$

The dynamics of the neuron membrane potential is then governed by:

$$C\frac{dV}{dt} = I_z - G_K n^4(V - V_K) - G_{Na} m^3 h(V - V_{Na}) - G_L(V - V_L). \tag{9}$$

We have set the parameters of the neuron model as follows: $g_l = 0.003$, $E_l = -44.42$, $C_m = 0.01$, $E_{in} = -72.14$ mV, $E_{in} = -72.14$ mV and $E_{ex} = 55.17$ mV. The reset membrane potential, V_{reset} is set to -70 mV and threshold voltage, V_{th} as -55 mV. The time constant τ_{reset} is set to 3 ms.

3. Spiking Neural Network

3.1. Membrane Potential and Spikes

We use the analytical solution of the Hodgkin-Huxley (HH) model obtained with the assumptions of $G_{Na} = 0$ and $G_K = 0$ to solve the membrane potential defined by Equation (9) at each time step t. According to Aaby [21] and Siciliano [22], this analytical solution is given by the following equation for the membrane potential:

$$v_1 = (\frac{1}{g_l})\{(-exp(\frac{g_l t}{C_m}))(I_z + 70g_l + g_l E_l) + I_z + g_l E_l\}. \tag{10}$$

Aaby [21] and Siciliano [22] showed that the membrane potential obtained by solving the HH model with numerical methods such as Runge-Kutta and the calculation using the analytical solution are nearly the same, therefore the above assumptions are reasonable. We find that this approach is successful in generating edge images, with an added advantage in processing low-exposure images.

At each time step t and for each neuron, the membrane potential v_1 is compared with the threshold voltage. If

$$v_1 > V_{Th}, \tag{11}$$

the neuron will have a spike that is, $S_1 = 1$ is assigned and v_1 is reset to v_{reset}, else the neuron will not spike, that is, $S_1 = 0$ is assigned.

We used six Gabor type filters, each filter representing a receptor field with a radius R = 5 pixels, with two filters each for vertical, horizontal and diagonal directions, respectively. The non-zero elements in the synaptic weight matrices of these Gabor type filters are derived using the scale factors:

$$W_{ex}(r,f) = W_{ex}^{max} exp[-\frac{(r-x_c)^2}{\delta_r^2} - \frac{(f-y_c)^2}{\delta_f^2}], \tag{12}$$

$$W_{in}(r,f) = W_{in}^{max} exp[-\frac{(r-x_c)^2}{\delta_r^2} - \frac{(f-y_c)^2}{\delta_f^2}], \tag{13}$$

where (x_c,y_c) is the center of the receptor field, δ_r, δ_f are constants, and W_{ex}^{max}, W_{in}^{max} are the maximum weights for the excitatory synapses and inhibitory synapses, respectively. The magnitude and the ratio of W_{ex}^{max} and W_{in}^{max} are adjusted such that the neuron does not fire in response to a uniform image within its receptive field. The size of the receptor field can be set in the range of 2×2 to 6×6. δ_r and δ_f control the sensitiveness to edges. Larger receptor field and larger δ_r, δ_f values make the detector less sensitive to the noise, but edges become vague, so there is a trade-off between these values. The values of W_{ex}^{max}, W_{in}^{max}, δ_r and δ_f are chosen heuristically. Below, we list the weight matrices of the six filters used for feature generation along with the parameters and the condition that are used to generate these matrices.

Matrices for Filter A: W_{ex}^{max} = 0.7093 , W_{in}^{max} = 0.3455, δ_r = 2, δ_f = 6, $W_{ex}(r, f)$ = 0 for $(r - x_c) \leq 0$ and $W_{in}(r, f)$ = 0 for $(r - x_c) > 0$.

W_{ex}^A =

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0.49 & 0.54 & 0.55 & 0.54 & 0.49 \\ 0.23 & 0.25 & 0.26 & 0.25 & 0.23 \end{bmatrix}$$

W_{in}^A =

$$\begin{bmatrix} 0.11 & 0.12 & 0.13 & 0.12 & 0.11 \\ 0.24 & 0.26 & 0.27 & 0.26 & 0.24 \\ 0.31 & 0.34 & 0.35 & 0.34 & 0.31 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Matrices for Filter B: W_{ex}^{max} = 0.7093 , W_{in}^{max} = 0.3455, δ_r = 2, δ_f = 6, $W_{ex}(r, f)$ = 0 for $(r - x_c) \geq 0$ and $W_{in}(r, f)$ = 0 for $(r - x_c) < 0$.

W_{ex}^B =

$$\begin{bmatrix} 0.23 & 0.25 & 0.26 & 0.25 & 0.23 \\ 0.49 & 0.54 & 0.55 & 0.54 & 0.49 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

W_{in}^B =

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0.31 & 0.34 & 0.35 & 0.34 & 0.31 \\ 0.24 & 0.26 & 0.27 & 0.26 & 0.24 \\ 0.11 & 0.12 & 0.13 & 0.12 & 0.11 \end{bmatrix}$$

Matrices for Filter C: If $(f - y_c) = \pm 1$, then W_{ex}^{max} = W_{in}^{max} = 0.4455, else W_{ex} = W_{in} = 0.3455. δ_r = 6, δ_f = 2, $W_{ex}(r, f)$ = 0 for $(f - y_c) \leq 0$ and $W_{in}(r, f)$ = 0 for $(f - y_c) \geq 0$.

W_{ex}^C =

$$\begin{bmatrix} 0 & 0 & 0 & 0.31 & 0.11 \\ 0 & 0 & 0 & 0.34 & 0.12 \\ 0 & 0 & 0 & 0.35 & 0.13 \\ 0 & 0 & 0 & 0.34 & 0.12 \\ 0 & 0 & 0 & 0.31 & 0.11 \end{bmatrix}$$

$W_{in}^{C} =$

$$\begin{bmatrix} 0.11 & 0.31 & 0 & 0 & 0 \\ 0.12 & 0.34 & 0 & 0 & 0 \\ 0.13 & 0.35 & 0 & 0 & 0 \\ 0.12 & 0.34 & 0 & 0 & 0 \\ 0.11 & 0.31 & 0 & 0 & 0 \end{bmatrix}$$

Matrices for Filter D: $W_{ex}^{max} = 0.7093$, $W_{in}^{max} = 0.3455$, $\delta_r = 6$, $\delta_f = 2$, $W_{ex}(r, f) = 0$ for $(f - y_c) \geq 0$ and $W_{in}(r, f) = 0$ for $(f - y_c) < 0$.

$W_{ex}^{D} =$

$$\begin{bmatrix} 0.23 & 0.49 & 0 & 0 & 0 \\ 0.25 & 0.54 & 0 & 0 & 0 \\ 0.55 & 0.35 & 0 & 0 & 0 \\ 0.25 & 0.54 & 0 & 0 & 0 \\ 0.23 & 0.49 & 0 & 0 & 0 \end{bmatrix}$$

$W_{in}^{D} =$

$$\begin{bmatrix} 0 & 0 & 0.31 & 0.24 & 0.11 \\ 0 & 0 & 0.34 & 0.26 & 0.12 \\ 0 & 0 & 0.35 & 0.27 & 0.13 \\ 0 & 0 & 0.34 & 0.26 & 0.12 \\ 0 & 0 & 0.31 & 0.24 & 0.11 \end{bmatrix}$$

Matrices for Filter E: $W_{ex}^{max} = 0.3455$, $W_{in}^{max} = 0.7093$, $\delta_r = 6$, $\delta_f = 2$, $W_{ex}(r, f) = 0$ for $(r + f - y_c) \leq 2$ and $W_{in}(r, f) = 0$ for $(r + f - y_c) > 2$.

$W_{ex}^{E} =$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0.1137 \\ 0 & 0 & 0 & 0.2617 & 0.1236 \\ 0 & 0 & 0.3455 & 0.2691 & 0.1271 \\ 0 & 0.2617 & 0.3360 & 0.2617 & 0.1236 \\ 0.1137 & 0.2408 & 0.3092 & 0.2408 & 0.1137 \end{bmatrix}$$

$W_{in}^{E} =$

$$\begin{bmatrix} 0.2335 & 0.4943 & 0.6347 & 0.4943 & 0 \\ 0.2538 & 0.5373 & 0.6899 & 0 & 0 \\ 0.2609 & 0.5524 & 0 & 0 & 0 \\ 0.2538 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Matrices for Filter F: $W_{ex}^{max} = 0.7093$, $W_{in}^{max} = 0.3455$, $\delta_r = 6$, $\delta_f = 2$, $W_{ex}(r, f) = 0$ for $(r + f - y_c) > 2$ and $W_{in}(r, f) = 0$ for $(r + f - y_c) \leq 2$.

$W_{ex}^{F} =$

$$\begin{bmatrix} 0.2335 & 0.4943 & 0.6347 & 0.4943 & 0 \\ 0.2538 & 0.5373 & 0.6899 & 0 & 0 \\ 0.2609 & 0.5524 & 0 & 0 & 0 \\ 0.2538 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$W_{in}^{F} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0.1137 \\ 0 & 0 & 0 & 0.2617 & 0.1236 \\ 0 & 0 & 0.3455 & 0.2691 & 0.1271 \\ 0 & 0.2617 & 0.3360 & 0.2617 & 0.1236 \\ 0.1137 & 0.2408 & 0.3092 & 0.2408 & 0.1137 \end{bmatrix}$$

3.2. Gabor Feature Based Edge Detector

Figure 2 displays the Spiking Neural Network architecture designed for spike processing with four layers of edge detection. The first layer in the architecture is a receptor layer R with dimensions identical to the input image. The second layer is made of six sets of Gabor filers, W_{nu}^{L} (L = A to F, nu = ex or in) where a set consists of a filter each for excitatory (ex) neurons and inhibitory (in) neurons [8,14]. The third layer is an intermediate layer, consisting of six parallel layers of neurons N_i (i = 1 to 6) with dimensions identical to the image. S_A, S_B, S_C, S_D, S_E and S_F are the output spike trains from neuron layers N_1, N_2, N_3, N_4, N_5 and N_6, respectively. The fourth and final layer is the spike output layer S_{out} with dimensions identical to the input image. The output of the image is then an edge map generated by the SNN edge detector. As shown in Figure 2, the framework combines six Gabor filters to generate an edge map of the image. The synaptic conductance of the Gabor feature is calculated by combining the spikes from the six Gabor filters using the equation:

$$g_{ex1}(t) = g_{ex}(|S_A(t) - S_B(t)|+|S_B(t) - S_C(t)| + |S_C(t) - S_D(t)|+|S_D(t) - S_E(t)|+|S_E(t) - S_F(t)|) \quad (14)$$

and the synaptic current is then evaluated as:

$$I_{z1}(t) = -g_{ex1}(t)E_{ex}. \quad (15)$$

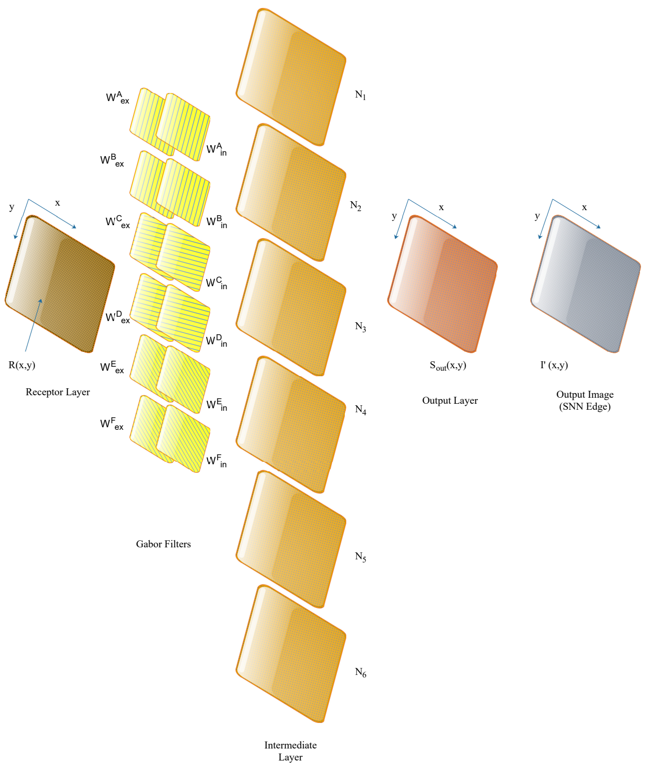

Figure 2. Spiking Neural Network Architecture showing the neuron spike processing layers of the edge detector.

The dynamics of the membrane potential is then calculated by solving Equation (9) with $g_{ex} = g_{ex1}$ and $g_{in} = 0$. The analytical solution with $V_{Na} = 0$ and $V_K = 0$ is then given by:

$$v_2 = (\frac{1}{g_l})\{(-exp(\frac{g_l t}{C_m})) \times (I_{z1} + 70g_l + g_l E_l) + I_{z1} + g_l E_l\}, \tag{16}$$

which is iteratively solved by updating the synaptic current using:

$$I_{z1} = g_{ex1}(v_2 - E_{ex}) \tag{17}$$

over the time window T to solve for the output spikes $S_2(t)$. The SNN edge map of the input image is calculated using the 1's complement of the normalized sum of spikes:

$$SNNEdge = 1 - \sum_{t=1}^{T} \frac{S_2(t)}{T}. \tag{18}$$

Figure 3 shows a flow chart with the computational steps in the MATLAB implementation of the SNN edge detector.

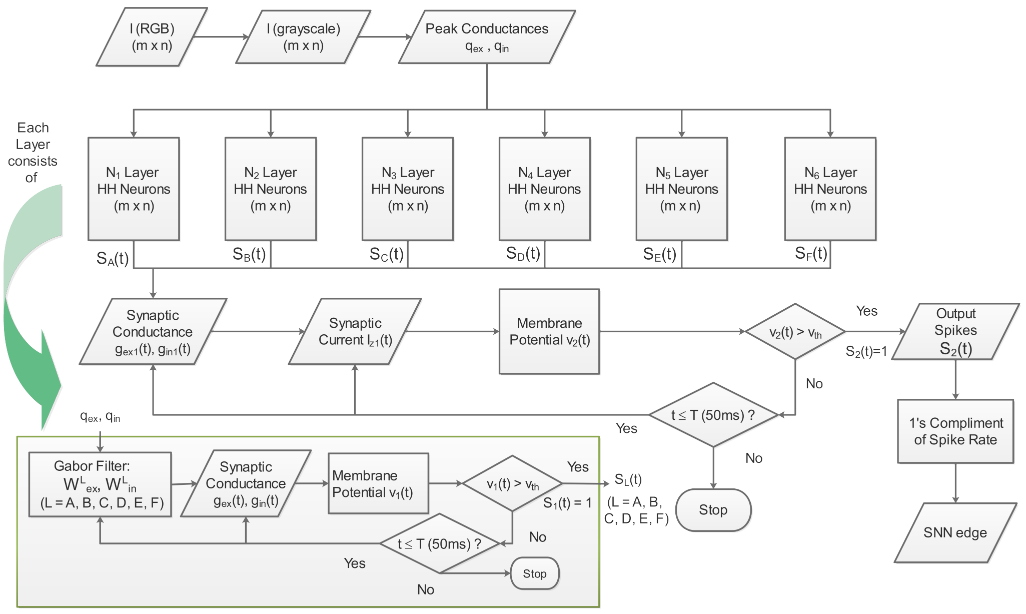
I (RGB) (m x n)
I (grayscale) (m x n)
Peak Conductances q_{ex}, q_{in}
Each Layer consists of
N1 Layer HH Neurons (m x n)
N2 Layer HH Neurons (m x n)
N3 Layer HH Neurons (m x n)
N4 Layer HH Neurons (m x n)
N5 Layer HH Neurons (m x n)
N6 Layer HH Neurons (m x n)
$S_A(t)$
$S_B(t)$
$S_C(t)$
$S_D(t)$
$S_E(t)$
$S_F(t)$
Synaptic Conductance $g_{ex1}(t)$, $g_{in1}(t)$
Synaptic Current $I_{z1}(t)$
Membrane Potential $v_2(t)$
$v_2(t) > v_{th}$
Yes
$S_2(t)=1$
Output Spikes $S_2(t)$
No
t ≤ T (50ms) ?
Yes
No
Stop
1's Compliment of Spike Rate
SNN edge
q_{ex}, q_{in}
Gabor Filter: W^L_{ex}, W^L_{in} (L = A, B, C, D, E, F)
Synaptic Conductance $g_{ex}(t)$, $g_{in}(t)$
Membrane Potential $v_1(t)$
$v_1(t) > v_{th}$
Yes
$S_L(t)$
$S_1(t) = 1$
(L = A, B, C, D, E, F)
No
t ≤ T (50ms) ?
Yes
No
Stop

Figure 3. Flow chart showing the computational steps for the implementation of the Spiking Neural Network (SNN) edge detector in MATLAB with I as the input image of $m \times n$ pixels.

4. Results and Discussion

4.1. Comparison of SNN Edges with Sobel and Canny Edge Detectors

The SNN edge detector presented in Section 3 is used to generate edges of three example images from the Berkeley Segmentation Dataset and Benchmark collection, BSDS500 [23]. Figure 4 compares these example images and their edge map based on the SNN detector with the results from the Sobel edge and Canny edge detectors. As can be seen in Figure 5, the SNN edge detector generates thicker edges that are brighter and offer higher contrast. In addition, the spatial extent of the edge features generated by the SNN edge detector is qualitatively higher compared to the edge features of Sobel edge and qualitatively lower compared to the edge features of the Canny edge. We also find that the shape of the objects is more distinctive and recognizable in the SNN edge map than in the other two edge maps.

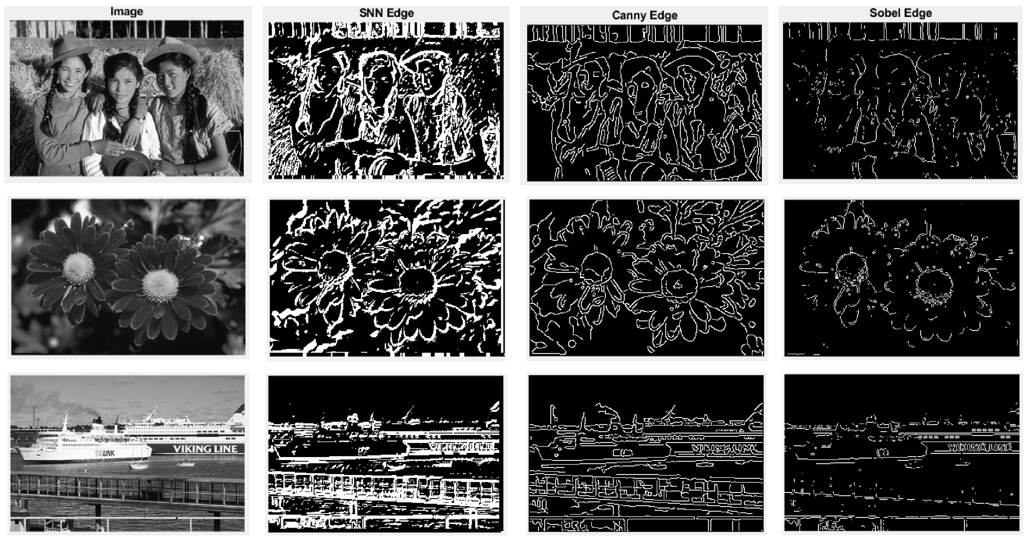

Figure 4. Examples images along with the comparison of edges generated by SNN, Canny and Sobel edge detectors.

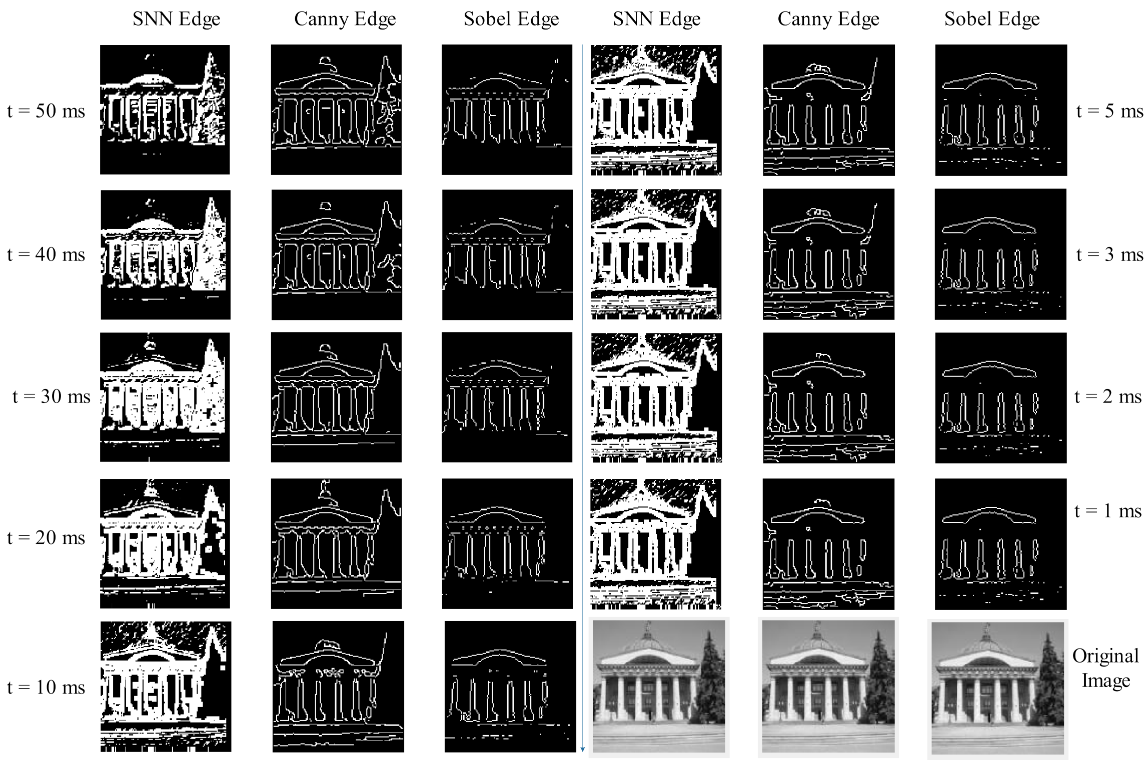

Figure 5. Comparison of edge detection methods for low exposure frames, $t = 1 - 50$ ms with an example image from MIT Sun Database.

In order to test the potential of our SNN edge detector for applications in neuromorphic computing, especially processors such as IBM-TrueNorth, we first simulated low-exposure image frames with exposure times in the range of 1 to 50 ms using a Non-linear Subtraction Range (NSR) low-exposure frame generation algorithm [24]. Then, we generated the SNN edges of the low exposure frames. Figure 5 shows the SNN edges generated for an example image from the MIT Sun Database [25] at various low exposure times in the range of 1 to 50 ms for the image. To compare the quantity of pixels that are part of the edges in the image at different exposure times, we define Edge

Pixel Metric (EPM):

$$EPM = \sum_{i,j} \delta(I'(i,j) - 1), \tag{19}$$

where I' is the edge map of the image generated by any of the three edge detectors and δ is the Dirac delta function. Figure 6 shows the exposure time dependence of the edge pixel metric, EPM, for an example image from the MIT Sun Database. As can be seen in the plot, the metric for SNN edge is significantly higher compared to the respective metric values for both the Sobel and Canny edge detectors. The higher metric for the SNN edge comes from the broader edge lines as a result of the unique architecture used for the edge detector.

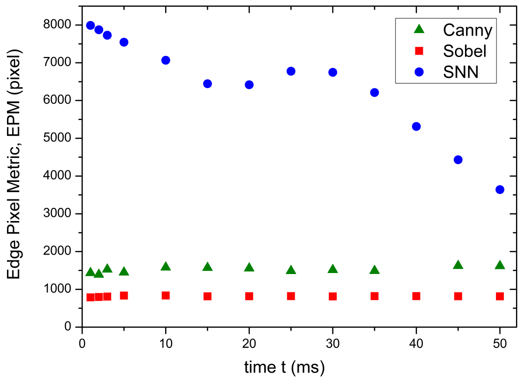

Figure 6. Total number of edge pixels in the SNN edge map of the input image as a function of simulated exposure time t.

4.2. Edge Based Classification of Digits Using a Convolutional Neural Network (CNN)

Object classification using deep learning from a series of 1 ms image frames captured with a camera is an interesting area of research for on-board signal processing and surveillance applications. In this context, we investigated the potential use of the SNN edge detector as a feature extraction layer in a CNN for object classification from low exposure image frames, each frame with an exposure time of 1 ms, simulated by the Non-linear Subtraction Range (NSR) algorithm [24]. The NSR algorithm is used to create low exposure frames with an assumed exposure time T of 40 ms for the Digits dataset, which is available in MATLAB [26]. The Digits dataset consists of 1000 images for single digit numbers, 0–9. Low exposure frames are created for all 10,000 images in the dataset. Such low exposure image versions of the Digits dataset are created with the following total exposure times; 1, 2, 3, 5, 10, 15, 20, 30 ms. We used the CNN architecture available in MATLAB [26] for digits classification with the introduction of an additional edge detection layer before the input neuron layer. We explored three cases of edge detection layers: the Canny edge detection layer, the Sobel edge detection layer and the SNN edge detection layer to compare the role and performance of the edge detection layer as a type of feature generation layer in classification. Then, each of these low exposure versions of the Digits dataset is used for both the training and the testing of the CNN for classification. Figure 7 shows the process flow diagram used for combining the 1 ms frame generation algorithm, SNN edge detection layer and a CNN for classification of digits in the Digits dataset. Therefore, our deep learning network combines concepts from both the artificial neural network and the SNN in a single network. Figure 8 illustrates the architecture of the CNN along with the details of the layers and their sizes. The CNN part of the network consists of the following layers in the order given: A 28 $\times$ 28 $\times$ 1 neuron image input layer, a convolutional layer (3 $\times$ 3 $\times$ 16 filter), a batch normalization layer, a Rectified Linear Units (ReLu) layer, a max pooling layer (pool size: 3) with a stride of 2, a convolutional layer (3 $\times$ 3 $\times$ 32 filter), a batch normalization layer, a ReLu layer, a max pooling layer (pool size of 2) with a stride of 2, a convolutional layer (3 $\times$ 3 $\times$ 64 filter), a batch normalization layer, a ReLu layer, a fully connected layer (10), a softmax layer and a classification layer. The first two numbers in the parenthesis correspond to the size of the filter in the layer and the third number defines the number of filters. Each convolution layer has a padding of 1. Figure 9 compares the example edge images for the low exposure frames after the edge layer in the network. The training and testing is performed for three epochs with a validation frequency of 30. The network is first trained with 75% of a random (but equal) number of each class of images from the low exposure dataset and tested with the remaining 25% of the low exposure dataset. The prediction accuracy of the SNN embedded CNN network as a function of the exposure time is displayed in Figure 10. The uncertainty in the prediction accuracy, displayed as error bars in the above figure, is determined from a 20-fold cross-validation experiment. We find that the network with the Canny and Sobel edge detection layers has a superior classification accuracy than the network with the SNN layer for $t > T/2$. We speculate that higher spike rates at these exposure times suppress the gradients and makes the detector less sensitive to edge features. Surprisingly, for exposure times $t < T/2$, the network with the SNN edge detection layer outperforms the networks with either the Sobel edge or Canny edge detection layer. Another noteworthy result is that the classification accuracy with the SNN edge detector layer in the network remains in the 88% to 90% range, while the accuracy with the other two edge detector layers drops to about 82% for the first 5 ms frames. The higher classification accuracy does not seem to arise from higher number of edge pixels. As can be seen in Figure 6, the number of edge pixels is the highest for the SNN edge at all the exposures times compared to the Canny edge and the Sobel edge, however its classification accuracy is lower compared to the classification accuracy of the Canny edge and the Sobel edge at higher exposure times. The edge pixel metric for the Canny edge and the Sobel edge are constant over exposure times, yet their classification accuracy drops at lower exposure times. The higher classification accuracy found for the SNN edge at low exposure times $t < T/2$ can be attributed to more discriminating features, which are retained by the broader edges, between the classes in the SNN edge images compared to Canny edge and Sobel edge images.

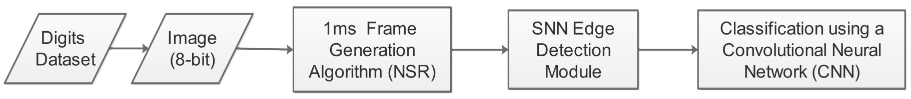
Digits Dataset
Image (8-bit)
1ms Frame Generation Algorithm (NSR)
SNN Edge Detection Module
Classification using a Convolutional Neural Network (CNN)

Figure 7. The process flow of the pipeline, which combines a 1 ms frame generator, SNN edge detector and a Convolutional Neural Network (CNN), for Digits classification.

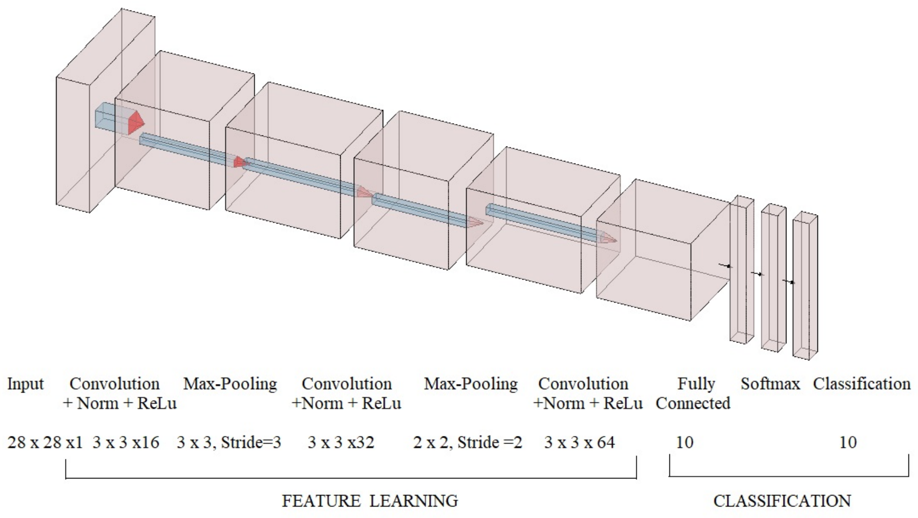

Figure 8. The architecture of the CNN used for Digits classification.

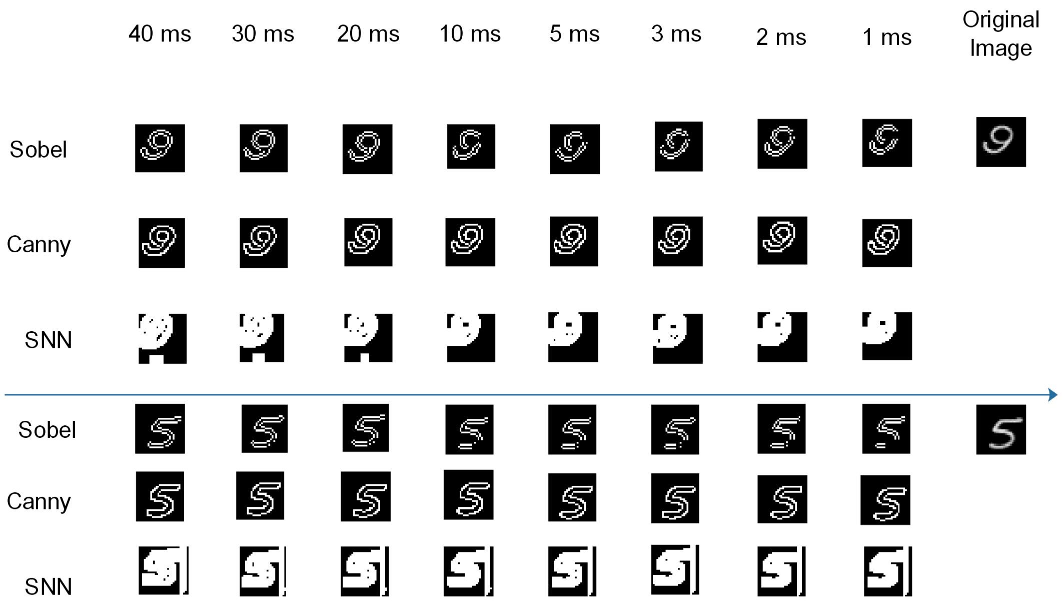

Figure 9. Comparison of edge detection methods for low exposure frames, $t = 1 - 40$ ms with an example image from Digits dataset.

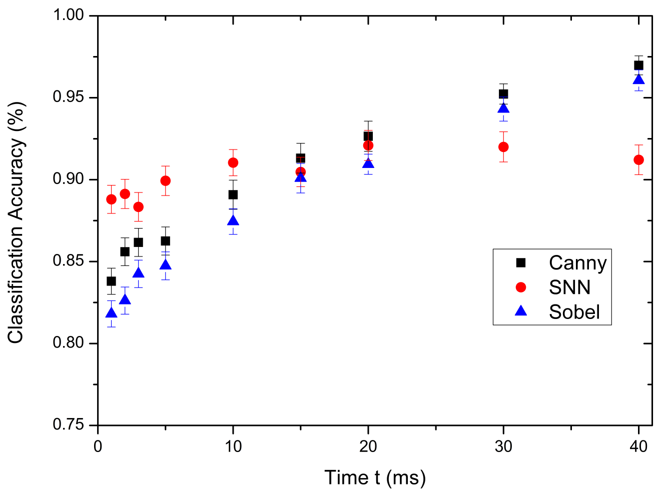

Figure 10. Classification accuracy as a function of number of 1 ms in the image used for edge detection.

5. Conclusions

We have investigated how a Spiking Neural Network (SNN) can be designed with biologically inspired neurons that closely resemble the Hodgkin-Huxley neurons for edge detection. We find that the one-step calculation of the membrane potential using the analytical expression from the exact solution of a Hodgkin-Huxley neuron model, with the assumption of zero conductance for Na and K ion channels, is computationally less burdensome compared to solving the system of ordinary differential equations for the Hodgkin-Huxley neuron model using a numerical method. The edge maps generated with the SNN based edge detector have similar structural features as edge maps created with popular edge detectors, specifically the Canny edge and Sobel edge detectors. We find that the SNN edge detector has an advantage in object detection for images generated at low exposure times on the order of a few milliseconds with 8-bit images. These results emphasize an alternative and computationally easier approach for SNN based image processing layers for machine learning in the form of combining the best of the second generation networks that is, ANNs and third generation networks, that is, SNNs. By combining an event camera like silicon retina that operates only in the visible spectrum with a neuromorphic simulator and a neural network, novel AI applications that target object detection in low-light conditions can be developed.

Funding: This work was internally funded by Riverside Research's Open Innovation Center.

Acknowledgments: The author would like to acknowledge William D. Mitchell and Jeffrey D. Clark for helpful discussions on edge detectors. The author thanks Frank Falco and Joshua Heath for critical reading and editing of the manuscript.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Demin, V.; Nekhaev, D. Recurrent Spiking Neural Network Learning Based on a Competitive Maximization of Neuronal Activity. *Front. Neuroinform.* **2018**, *12*, 79. [CrossRef]
2. Bellec, G.; Salaj, D.; Subramoney, A.; Legenstein, R.A.; Maass, W. Long short-term memory and learning-to-learn in networks of spiking neurons. In Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montreal, QC, Canada, 3–8 December 2018.
3. Florian, R.V. Reinforcement Learning Through Modulation of Spike-Timing-Dependent Synaptic Plasticity. *Neural Comput.* **2007**, *19*, 1468–1502. [CrossRef] [PubMed]
4. Gers, F.A.; Schraudolph, N.N.; Schmidhuber, J. Learning Precise Timing with LSTM Recurrent Networks. *J. Mach. Learn. Res.* **2002**, *3*, 115–143.
5. Cui, Y.; Ahmad, S.; Hawkins, J. Continuous Online Sequence Learning with an Unsupervised Neural Network Model. *Neural Comput.* **2016**, *28*, 2474–2504. [CrossRef] [PubMed]
6. Yu, Q.; Tang, H.; Tan, K.C.; Yu, H. A brain-inspired spiking neural network model with temporal encoding and learning. *Neurocomputing* **2014**, *138*, 3–13. [CrossRef]
7. Berry, M.J., II; Meister, M. Refractoriness and Neural Precision. *J. Neurosci.* **1998**, *18*, 2200–2211. [CrossRef] [PubMed]
8. Wu, Q.; McGinnity, M.; Maguire, L.; Belatreche, A.; Glackin, B. Edge Detection Based on Spiking Neural Network Model. In *Lecture Notes in Computer Science, Proceedings of the Advanced Intelligent Computing Theories and Applications, With Aspects of Artificial Intelligence, ICIC 2007, Qingdao, China, 21–24 August 2007*; Huang, D.-S., Heutte, L., Loog, M., Eds.; Springer, Berlin/Heidelberg, Germany, 2007; Volume 4682.
9. Yedjour, H.; Meftah, B.; Leźoray, O.; Benyettou, A. Edge detection based on Hodgkin–Huxley neuron model simulation. *Cogn. Process.* **2017**, *8*, 315–323. [CrossRef] [PubMed]
10. Cassidy, A.; Sawada, J.; Merolla, P.; Arthur, J.; Alvarez-lcaze, R.; Akopyan, F.; Jackson, B.L.; Modha, D. TrueNorth: A high-performance, low-power neurosynaptic processor for multi-sensory perception, action, and cognition. In Proceedings of the Government Microcircuits Applications and Critical Technology Conference, Orlando, FL, USA, 14–17 March 2016; pp. 14–17.
11. Hodgkin, A.L.; Huxley, A. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.* **1952**, *117*, 500–544. [CrossRef] [PubMed]
12. Izhikevich, E.M. Simple models of spiking neurons. *IEEE Trans. Neural Netw.* **2003**, *14*, 1569–1572. [CrossRef] [PubMed]
13. Moldakarimov, S.; Bazhenov, M.; Sejnowski, T.J. Top-down inputs enhance orientation selectivity in neurons of the primary visual cortex during perceptual learning. *PLoS Comput. Biol.* **2014**, *10*, e1003770. [CrossRef] [PubMed]
14. Tsitiridis, A.; Conde, C.; de Diego, I.M.; del Rio Saez, J.S.; Gomez, J.R.; Cabello, E. Gabor feature processing in spiking neural networks from retina-inspired data. In Proceedings of the 2015 International Joint Conference on Neural Networks (IJCNN), Killarney, Irland, 12–17 July 2015.
15. Yang, Z.; Murray, A.; Wörgötter, F.; Cameron, K.; Boonsobhak, V. A Neuromorphic Depth-From-Motion Vision Model With STDP Adaptation. *IEEE Trans. Neural Netw.* **2006**, *17*, 482–495. [CrossRef] [PubMed]
16. Davies, M.; Srinivasa, N.; Lin, T.H.; Chinya, G.; Cao, Y.; Choday, S.H.; Dimou, G.; Joshi, P.; Iman, N.; Jain, S.; et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro* **2018**, *38*, 8–99. [CrossRef]

17. Roberts, L.G. Machine perception of 3-D solids. In *Optical and Electro-Optical Information Processing*; MIT Press: Cambridge, MA, USA, 1965.
18. Canny, J. A Computational Approach to Edge Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **1986**, *6*, 679–698 [CrossRef]
19. Sobel, I. *An Isotropic* 3 × 3 *Gradient Operator, Machine Vision for Three—Dimensional Scenes*; Freeman, H., Ed.; Academic Press: New York, NY, USA, 1990; pp. 376–379.
20. Destexhe, A.; Rudolph, M.; Fellous, J.M.; Sejnowski, T. Fluctuating synaptic conductances recreate in vivo-like activity in neocortical neurons. *Neuroscience* **2001**, *107*, 13–24. [CrossRef]
21. Aaby, D. *A Comparitive Study of Numerical Methods for the Hodgkin-Huxley Model of Nerve Cell Action Potentials*; University of Dayton: Dayton, OH, USA, 2009.
22. Siciliano, R. The Hodgkin-Huxley Model, Its Extensions, Analysis and Numerics. 2012. Available online: https://www.math.mcgill.ca/gantumur/docs/reps/RyanSicilianoHH.pdf (accessed on 8 November 2019).
23. Martin, D.; Fowlkes, C.; Tal, D.; Malik, J. A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. In Proceedings of the Eighth International Conference On Computer Vision (ICCV 2001), Vancouver, BC, Canada, 7–14 July 2001.
24. Vemuru, K.V.; Clark, J.D. Low-exposure image frame generation algorithms for feature extraction and classification. In Proceedings of the SPIE, Real-Time Image Processing and Deep Learning 2019, Baltimore, MA, USA, 14 May 2019; Volume 10996, p. 109960D.
25. Xiao, J.; Hays, J.; Ehinger, K.; Oliva, A.; Torralba, A. SUN Database: Large-scale Scene Recognition from Abbey to Zoo. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 13–18 June 2010.
26. *MATLAB and Statistics Toolbox Release 2018b*; The MathWorks, Inc.: Natick, MA, USA, 2018.