Exploring Convolutional Spiking Neural Networks for Image Classification

AI3610 Brain-Inspired Intelligence Homework

Xiangyuan Xue

November 12, 2023

Shanghai Jiao Tong University

Method

Spiking Neuron

Leaky Integrate-and-Fire (LIF) Model

- The membrane potential $v(t)$ is given by the following differential equation, where τ_m is the time constant, v_{rest} is the resting potential and $I(t)$ is the input current.[1]

$$\tau_m \frac{\mathrm{d}v}{\mathrm{d}t} = -[v(t) - v_{\mathrm{rest}}] + RI(t)$$

- For the sake of computer simulation, we use difference equation to approximate the differential equation.

$$V[t] - V[t-1] = -\frac{1}{\tau}(RI[t] + V_{\mathrm{rest}} - V[t-1])$$

Spiking Neuron

Leaky Integrate-and-Fire (LIF) Model

- The charging process is just as described above.

$$H[t] = V[t-1] + \frac{1}{\tau}(X[t] - (V[t-1] - V_{\text{reset}}))$$

- If $H[t]$ reaches $V_{\text{threshold}}$, the neuron will fire and reset.

$$S[t] = \Theta(H[t] - V_{\text{threshold}})$$
$$V[t] = H[t](1 - S[t]) + V_{\text{reset}}S[t]$$

- Note that $\Theta(x)$ is the Heaviside step function.

Spiking Neuron

Surrogate Gradient

- The derivative of the Heaviside step function is the Dirac delta function, which cannot be implemented.

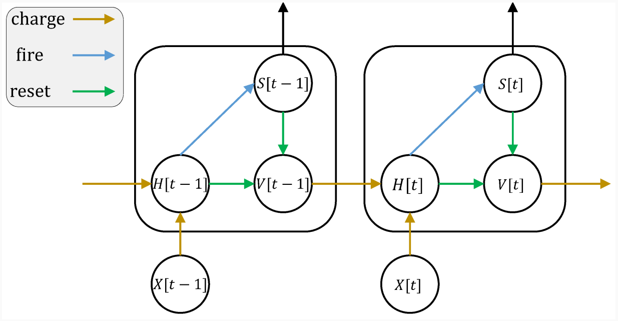
$$\Theta(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad \delta(x) = \begin{cases} +\infty, & x = 0 \\ 0, & x \neq 0 \end{cases}$$

- Therefore, we only use $\Theta(x)$ in forward propagation and use $\sigma'(x)$ as the surrogate gradient in backward propagation, where $\sigma(x)$ is the sigmoid function.

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad \sigma'(x) = \sigma(x)(1-\sigma(x))$$

Spiking Neuron

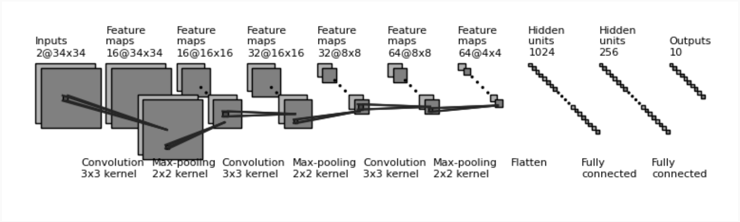
Neuron Dynamics



- The arrows describe the process of forward propagation. Backward propagation goes in the opposite direction.

Network Architecture

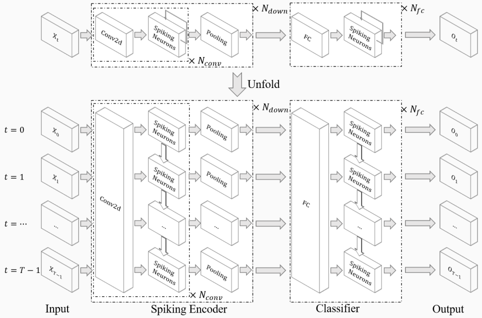
Convolutional Spiking Neural Network



- To achieve better performance in image classification tasks, we apply convolutional layers in the network.[3]
- Our convolutional spiking neural network contains 3 convolutional layers and 2 fully connected layers.

Network Architecture

Convolutional Spiking Neural Network



- No bias should be added to the convolutional layers.
- Activation functions are replaced by spiking neurons.

Data Processing

Neuromorphic Datasets

- In neuromorphic datasets, images are represented by event sequences $\{e_i = (x_i, y_i, t_i, p_i)\}_{i=1}^{n}$.
- Equally divide n events into m frames. Each frame is generated by integrating events in a time window.

$$f(k, p, x, y) = \sum_{i=k_l}^{k_r} \mathcal{I}_{p,x,y}(p_i, x_i, y_i)$$

- Note that $\mathcal{I}$ is the indicator function, which takes 1 if and only if $(p, x, y) = (p_i, x_i, y_i)$. Otherwise it takes 0.

Data Processing

Tensor Flow

- The images are converted into spike trains with shape (T, N, C, H, W), where T is the frame numbers, N is the batch size, C is the channel numbers, H and W are the image size.
- Through the convolutional spiking neural network, the outputs are spike trains with shape (T, N, M), where M is the number of classes.
- Take the average of the spike trains along the time axis, we get the firing rate of the output neurons. The one with the highest firing rate represents the predicted class.

Optimization Algorithm

Loss Function

- An ideal case is that only the correct neuron fires at the highest frequency. At the same time, other neurons remain as silent as possible.
- Here we choose mean squared error loss as the loss function, which tends to achieve better performance.

$$\mathcal{L}(y, \hat{y}) = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} (y_{ij} - \hat{y}_{ij})^2$$

- Note that y represents the firing rate of the output neurons and $\hat{y}$ represents the one-hot encoded label.

Optimization Algorithm

Adam Optimizer

- Adam is an effective and efficient optimization algorithm, which is widely used in deep learning.[2]
- Based on the gradient descent algorithm, Adam optimizer uses adaptive learning rates for each parameter.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \quad \hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

Experiments

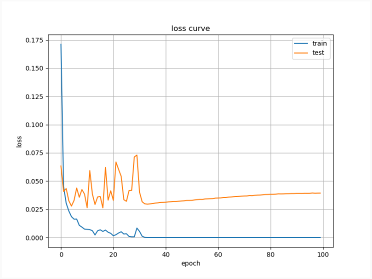
Expriment Setup

Parameter Setting

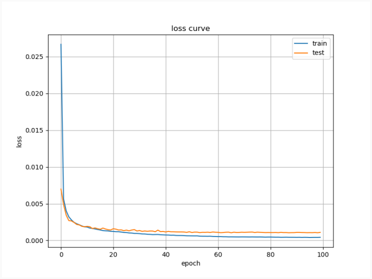
- We conduct experiments on the N-MNIST dataset.
- A convolutional artificial neural network with exactly the same architecture is implemented as the baseline, which uses the MNIST dataset and cross entropy loss.
- Both networks are trained with Adam optimizer with cosine annealing scheduler for 100 epochs on a single NVIDIA GeForce RTX 3090 GPU. The batch size is 256 and the initial learning rate is 10^{-3}.
- Loss, accuracy and time will be recorded every epoch during training for comparison and analysis.

Classification Precision

Loss Curve



CNN

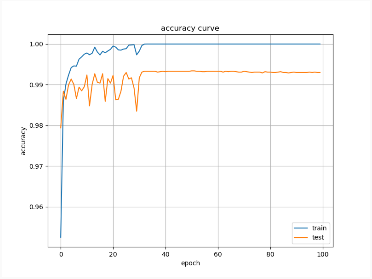


SNN

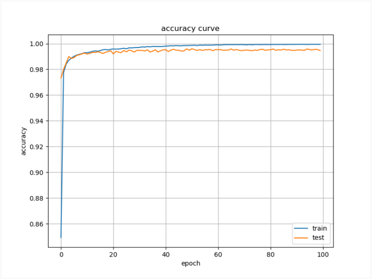
- SNN has higher loss on the training set, but achieves lower loss when generalizing to the test set.

Classification Precision

Accuracy Curve



CNN



SNN

- CNN overfits the training set with 100% accuracy.
- SNN generalizes better to the test set with higher accuracy.

Classification Precision

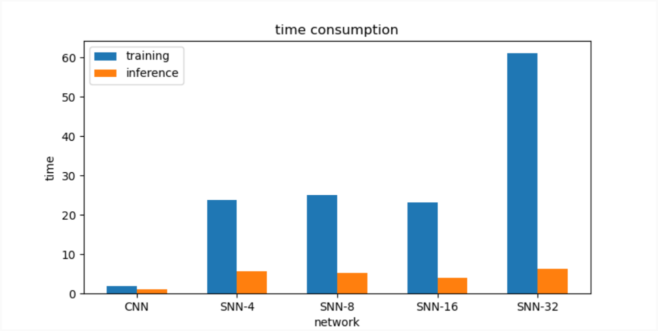
Detailed Comparison

	Evaluation Metrics			
	$\mathcal{L}_{\text{train}}$	$\mathcal{L}_{\text{test}}$	$\boldsymbol{\tau}_{\text{train}}$	$\boldsymbol{\tau}_{\text{test}}$
CNN	0.0000	0.0329	100.00%	99.34%
SNN	0.0006	0.0011	99.86%	99.60%

- $\mathcal{L}$ and $\boldsymbol{\tau}$ represents the loss and accuracy respectively.
- CNN shows stronger fitting ability, but tends to overfit the training set. SNN shows better generalization ability and achieves higher accuracy on the test set.

Computational Efficiency

Time Consumption



- CNN is much faster in both training and inference.
- With larger time steps, SNN consumes more time.

Computational Efficiency

Detailed Comparison

	Time Consumption (s)				
	CNN	SNN-4	SNN-8	SNN-16	SNN-32
Training	1.98	23.76	25.12	23.24	61.19
Inference	1.13	5.64	5.30	4.00	6.23
Total	3.11	29.40	30.42	27.25	67.42

- CNN is approximately 10 times faster than SNN, which makes it successful with limited resources.
- Larger time steps bring more time consumption and slightly better performance. Therefore, we choose 16 as a balance between performance and efficiency.

References

Wulfram Gerstner, Werner M Kistler, et al.
Neuronal dynamics: From single neurons to networks and models of cognition.
Cambridge University Press, 2014.

Diederik P Kingma and Jimmy Ba.
Adam: A method for stochastic optimization.
arXiv preprint arXiv:1412.6980, 2014.

Ruthvik Vaila, John Chiasson, et al.
Deep convolutional spiking neural networks for image classification.
arXiv preprint arXiv:1903.12272, 2019.

Thank You!