RUTGERS UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

Gesture Recognition using SNNs

Anmol Sharma (as3593)

May 04 2022
BRAIN INSPIRED COMPUTING
PROF KONSTANTINOS MICHIMIZOS

Contents

1 Introduction

For Gesture Recognition, there are two types used of problem techniques. Static and Dynamic. Our focus is more on the ways to handle Dynamic Gesture Recognition. For that, we are using Quantized Spiking Neural Networks(SNN) [12] [9] and scheduling of the Learning Rate to bring better accuracy

We are using the approach described in the paper "Navigating Local Minima in Quantized Spiking Neural Networks" by Eshrengian J. et al. 2022[4] as our reference.

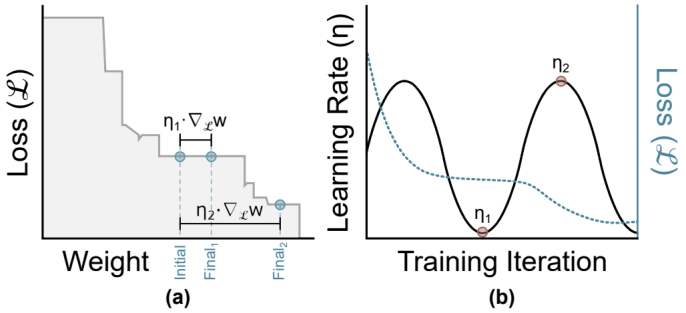


Figure 1: Cosine Annealing- Scheduling Technique for Learning rate to avoid plateaus and local minimas

In this paper the Author proposes improvements on traditional spiking neural networks by quantizing the weights and scheduling the learning rate. We choose Spiking Neural Architecture [1] as it has been proved to be effective in recognizing gestures in various studies. The weight quantization is a standard approach that is usually applied to reduce the solution space of weights like done in [6] and [3]. In our datasets, we use one neuromorphic gesture dataset- DVS gesture [2] and one image recognition dataset - FMNIST.

The quantization of weights to 4/8 bits helps train the model faster without much loss in accuracy and as seen in Fig 1., the scheduling of Learning Rate using Cosine Annealing helps us get out of local minimas.By periodically boosting the LR at repeated intervals in Cosine Annealing can be thought of as providing the network with another chance to search for a more optimal solution with potentially improved initial conditions. In this phase of our project, we experiment with the learning rate schedulers on the two datasets and try and reproduce results similar to the ones reported by the author.

2 Background

A neuron in our brains may be connected to 1,000 to 10,000 additional neurons. If one neuron spikes, it may be felt by all neurons downhill. But what factors influence whether or not a neuron spikes? Experiments over the last century have shown that if a neuron receives enough stimuli at its input, it can become stimulated and fire its own spike. A pair of neurons are connected by a synapse. A stronger synaptic connection will cause the post-synaptic neuron to "feel" output signals emitted by the pre-synaptic neuron more strongly. Input spike can either excite or inhibit a neuron's ability to fire. If a neuron is sufficiently excited, it will emit its own spike down its axon to downstream neurons.

$$u_{t+1}^{j} = \beta u_t^j + \sum_i w^{ij} z_t^i - z_t^j \theta$$

$$z_t^j = \begin{cases} 1, & \text{if } u_t^j > \theta \\ 0, & \text{otherwise} \end{cases}$$

We have used cosine annealing for minimizing loss in learning.The gradient descent method [10]is an iterative algorithm that solves the objective function's gradient vector, updates the parameter value in the negative gradient direction, and solves the objective function's minimum until convergence. As a result, in order to minimize the loss function, each network parameter must be updated along the negative gradient direction.

The spiking neuron model used is a single-state leaky integrate-and-fire neuron governed by the discrete-time dynamics: which is derived using the forward Euler method to solve the continuous-time representation which is derived using the forward Euler method to solve the continuous-time representation [5]. Here, u_t^j is the hidden state (membrane potential) of neuron j at time t; β is the membrane potential decay rate; w^{ij} is the synaptic weight between neurons i and j; and the final term of (1) resets the state by subtracting the threshold θ each time an output spike $z_t^j \in \{0, 1\}$ is generated.
During training, QSNNs are especially prone to becoming stuck at suboptimal barriers and local minima that are far from the Bayes optimal error. A large LR causes large weight changes at the risk of jumping over ideal solutions, but it may also be necessary to traverse across flat loss surfaces later in training.

3 Experiment Settings

We reproduce the approach and architecture presented in the paper. The two main Ideas of the paper are Quantization of weights which helps reduce the memory requirements while also reducing the solution space and Scheduling of Learning rate which helps us avoid local minimas. We refer to the author's Github repository for code reference and try to reproduce the results on our local. The Datasets are used directly without any augmentation. We also attach an ipython notebook example which can be directly executed on colab/jupyter. We also experiment with the learning rate and show comparison of ConsineAnnealing with other LRSchedulers like MultistepLR and Exponential LR.
As the spike encoding is non-differentiable, to train the model using back-propagation [7], we use a threshold-shifted fast sigmoid surrogate gradient.

3.1 Dataset: DVS Gesture

We perform our experiment using Quantized SNNs on two datasets, first one being Fashion MNIST and the second being DVS-Gesture. Our main objective going forward would be to include more neuromorphic datasets like DVS gesture becuase they are inherently more suitable for Spiking Neural Networks.

DVS-Gesture is a neuromorphic dataset that is made publicly available by IBM. It contains video recording of people doing twelve types of gestures as shown in the figure. The dataset is captured using Dynamic Visual Sensing Camera that only capture the moving pixels making this dataset suitable for Spiking Neural Nets in the way that it mimics the input to our neurons in the brain. The author suggests to apply spatial downsampling (32×32), and integrating events over a temporal resolution of 5 ms per input at a given sequence step, such that training samples were fit to a duration of 1 s each, which corresponds to a sequence length of 100 time steps.

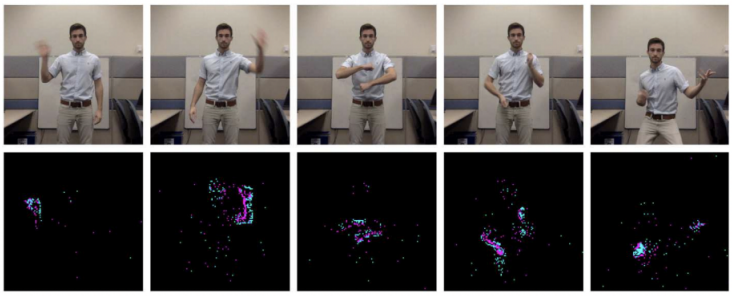


Figure 2: DVS Gesture Dataset

3.2 Dataset: FMNIST

We use Fashion MNIST as a more difficult version of MNIST and to better identify the performance of our experiments. It has the same size and resolution as MNIST, but it has ten different classes of clothing accessories. We feed 100 timesteps of raw input into the network without encoding and then evaluate our Learning Rates' performance.

3.3 Model Architecture

We used the architecture proposed by the author and use mainly a combination of Convolutional layers, Leaky Integrate and Fire layers(SNN), Fully-connected layers and a final dropout layer [11].

Adam Optimizer [8] is used because it has been shown to produce good results on Spiking Neural Networks. Each

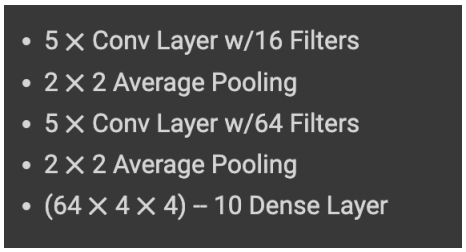


Figure 3: Model Architecture

model employs two convolutional layers followed by a dense layer. Following each convolution operation, average-pooling was applied to the membrane potential rather than spikes, because pooling sparse outputs with many zero elements would arbitrarily reduce the influence of spikes.

3.3.1 Pooling Layer

By summarizing the existence of features in portions of the feature map, pooling layers provide a method for down sampling feature maps. Average pooling and max pooling are two common pooling algorithms that summarize a feature's average presence and most activated presence, respectively. We calculate the average value for each patch on the feature map using Average Pooling. Calculating the average for each patch of the feature map is what average pooling entails. This means that each of the feature map's 22 squares gets down sampled to its average value.

3.3.2 Leaky Integrate and Fire Layers

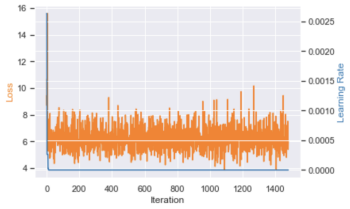
It, like the artificial neuron, takes the sum of weighted inputs. However, rather than passing it directly to an activation function, it will integrate the input with a leakage over time, similar to an RC circuit. If the integrated value exceeds a certain threshold, the LIF neuron will produce a voltage spike. The shape and profile of the output spike are abstracted away by the LIF neuron; it is simply treated as a discrete event. As a result, information is stored in the timing (or frequency) of spikes rather than the spike itself.

3.3.3 Fully connected layer

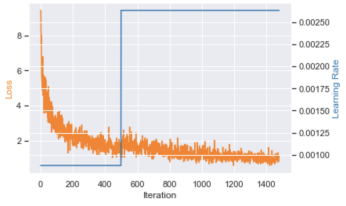
The neuron in fully connected layers applies a linear transformation to the input vector via a weights matrix. The product is then subjected to a non-linear transformation via a non-linear activation function f. In this case, we're taking the dot product of the weights matrix W and the input vector x. Inside the non-linear function, the bias term (W0) can be added.

3.3.4 Dropout Layer

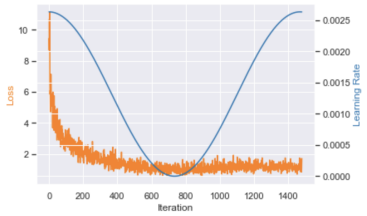
Dropout refers to data or noise that is purposefully removed from a neural network in order to improve processing and time to results. A neural network is a piece of software that attempts to mimic the actions of the human brain. The human brain is made up of billions of neurons that communicate with one another via electrical and chemical signals in order to coordinate thoughts and life functions. A neural network employs a software equivalent of these neurons, referred to as units. Each unit receives signals from other units and computes an output, which it then sends to other neuron/units, or nodes, in the network.



(a) Multi- Step Learning Rate

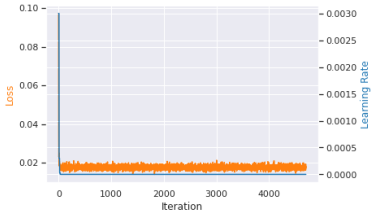


(b) Linear Learning Rate

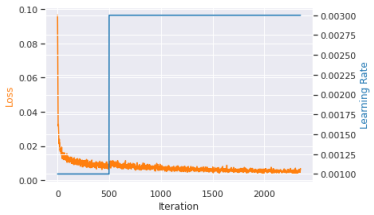


(c) Cosine Learning Rate

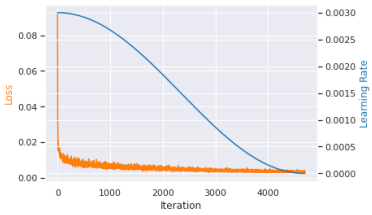
Figure 4: Loss and Learning Rate Comparison for DVS Gesture Dataset



(a) Multi- Step Learning Rate



(b) Linear Learning Rate



(c) Cosine Learning Rate

Figure 5: Loss and Learning Rate Comparison for Fashion MNIST

4 Results

We get the following accuracy table for results of our experimental settings.

Dataset	MultiLR	LinearLR	CosAnnealLR
FMNIST	69.7	84.9	86.8
DVS-Gesture	32.6	85.76	88.7

Accuracy Table

Apart from that we plot the losses and learning rate with the iterations for our experiment as follows:

4.1 Loss and Learning Rate comparisons

We can observe the various shapes and corresponding losses for the datasets and the learning rate rules. Observe, that cosine rule is the only one that is periodic. This property helps it avoid the local minimas and look again over missed optimas as it is boosted in its periodic cycle. We can see in Figure 4 and Figure 5 that loss decreases consistently for the Cosine Learning Rule.

4.2 Weight Quantization

We experiment with the effect of quantization of weight with model training parameters and results. We use 3 types of weight values i.e. float (normal weights) , 4-bit weight values and 8 bit weight values. We can see that

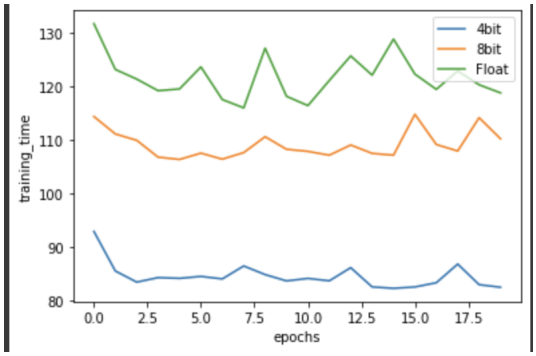


Figure 6: training time comparison for 4 bits, 8. bits and Floating point

As the model architecture is same, the accuracy is for 4 bits, 8 bits and Floating point is almost the same.

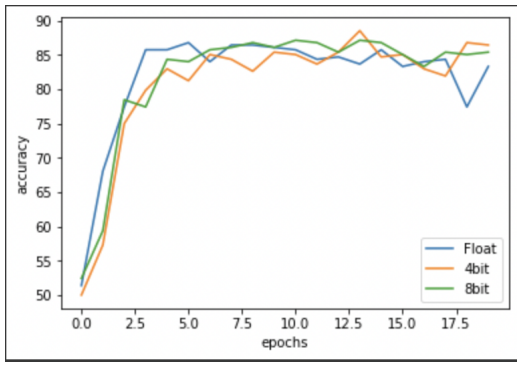


Figure 7: Accuracy comparison for 4 bits, 8. bits and Floating point

The loss reduction rate is also same for 4 bits, 8 bits and floating points.

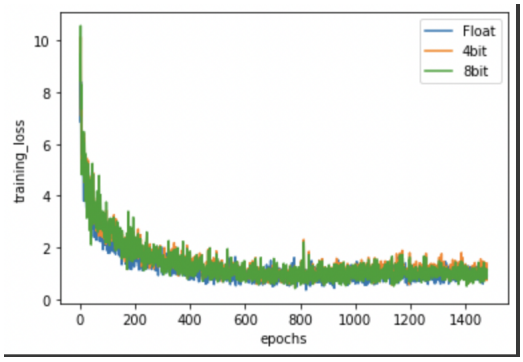


Figure 8: training loss comparison for 4 bits, 8. bits and Floating point

5 Conclusion

In this project we proposed a spiking neural network model for classification for video as well as image data for DVS and FMNIST datasets respectively.

1. We perform analysis based on Learning Rate Schedulers like, linear LR, Multi- step LR and Cosine LR
2. We also perform comparative analysis based on weight quantization comparing floating points, 8 bits and 4 bits weight values

We conclude that Cosine LR is the best scheduler for the model and having 4 bit quantization takes the least time in training the model without any decrease in performance.

References

[1] Rahib H. Abiyev, Okyay Kaynak, and Yesim Oniz. Spiking neural networks for identification and control of dynamic plants. In *2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1030–1035, 2012.

[2] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, Jeff Kusnitz, Michael Debole, Steve Esser, Tobi Delbruck, Myron Flickner, and Dharmendra Modha. A low power, fully event-based gesture recognition system. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7388–7397, 2017.

[3] Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2285–2294, Lille, France, 07–09 Jul 2015. PMLR.

[4] Jason K. Eshraghian, Corey Lammie, Mostafa Rahimi Azghadi, and Wei D. Lu. Navigating local minima in quantized spiking neural networks, 2022.

[5] Jason K. Eshraghian, Max Ward, Emre Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D. Lu. Training spiking neural networks using lessons from deep learning, 2021.

[6] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization, 2014.

[7] Hecht-Nielsen. Theory of the backpropagation neural network. In *International 1989 Joint Conference on Neural Networks*, pages 593–605 vol.1, 1989.

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[9] Nitin Rathi and Kaushik Roy. Diet-snn: Direct input encoding with leakage and threshold optimization in deep spiking neural networks, 2020.

[10] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2016.

[11] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[12] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural Networks*, 111:47–63, mar 2019.