Synsense Design Assignment

Hajar Asgari

October 31, 2024

Question 1: Handling flicker noise

Flicker noise caused by artificial lightning sources can degrade the image quality and event based cameras are more vulnerable to this type of artefact. This is one of the challenges in building a robust application with these sensors and severity of the effect can vary from negligible to very destructive, depending on the deployment environment and the efficiency of artefact removal algorithms.

You are asked to design and build a gesture recognition application with SynSense Speck (DVS camera chained with a SNN processor). You are supposed to collect the training dataset with DVS camera and make your model as robust as possible in presence of various artificial lights.

1. What would be your strategies at data collection, data engineering and model training level to minimize the effect of flicker noise?

Dataset: The training dataset should be collected with different illumination conditions. I would start with the available standard dataset such as DVS128 Gesture Dataset [1] collected by IBM. It comprises 11 hand gesture categories including circulating hand clockwise and counter-clockwise, clapping, and playing drum from 29 subjects under 3 illumination conditions. The 3 lighting conditions are combinations of natural light, fluorescent light, and LED light, which were selected to control the effect of shadows and fluorescent light flicker on the DVS128. I would use 80% of the dataset for training and keep 20% to evaluate the trained network.

Model training: The Speck chip is fully configurable for implementing different spiking convolutional neural network architectures [2]. The available functions on Speck are 2D convolutional and Pooling layers. So I would start my design with a Spiking Convolutional Neural Network, pairs of convolution and Pooling layers, and a winner-take-all layer in the last layer, then modify the network architecture and hyperparameters to enrich the network's performance. We might need to modify the dimension of the input vector to the network based on the training dataset using a flattened layer. The training of sCNN networks for Speck is supported by the rich, open source, high-level framework Sinabs based on PyTorch and a full development solution called Samna

2. Is there any remedy to minimize the flicker noise effect at deployment time?

The flicker noise effect is a well-known problem in the community and there are several approaches to minimize it. The following approaches can help with it:

- Low-Pass Filters can be an effective approach to reduce flicker noise in data from a DVS. Flicker noise, often caused by lighting variations or sensor imperfections, tends to produce high-frequency, transient events that don't align with meaningful motion in the scene. Low-pass filters are designed to allow only low-frequency signals to pass.
- Accumulating events over short time windows can help average out noise events, reducing their impact. Accumulating events can be a simple yet effective method for filtering out flicker noise in DVS data. By accumulating events over short time windows or spatial regions, we can differentiate between meaningful activity and random noise.
- Adaptive thresholding is an effective technique for filtering flicker noise in data from a DVS. By using adaptive thresholds, we can dynamically adjust the filter's sensitivity to distinguish between meaningful events and noise.

- Regular Calibration: Periodically calibrate the DVS camera under controlled lighting conditions to help the system distinguish between noise and true events. Calibration helps align the DVS to current conditions and reduces noise by ensuring that the sensor's responses are consistent, accurate, and optimized for the scene's characteristics.
- Uses the newly designed DVS camera such as SciDVS [3] which is specifically designed to deal with dim lighting.

3. Can you suggest an adaptive algorithm (to be implemented at camera or SNN HW or fully at SW level) to reduce this effect?
As I mentioned in our meeting, we designed an event-driven saliency-based selective attention mechanism presented in BioCAS 2022 [4]. In that project, by each event arrival, we update the state of the pixel (calculate the firing rate of the pixel) and based on that select the most salience region. We define the pixel state as:

$$S[t] = 1 + S(t_{old}).exp[(t_{old} - t)/\tau] \tag{1}$$

Where S is defined as the state or firing rate, τ is the time constant representing the forgetting ability of the system, t and t_{old} are the current time and the timestamp of the last event, respectively.
It's possible to use the same strategy for filtering out the flicker noise as the firing rates of the noisy pixels are extremely high compared to the normal pixels.

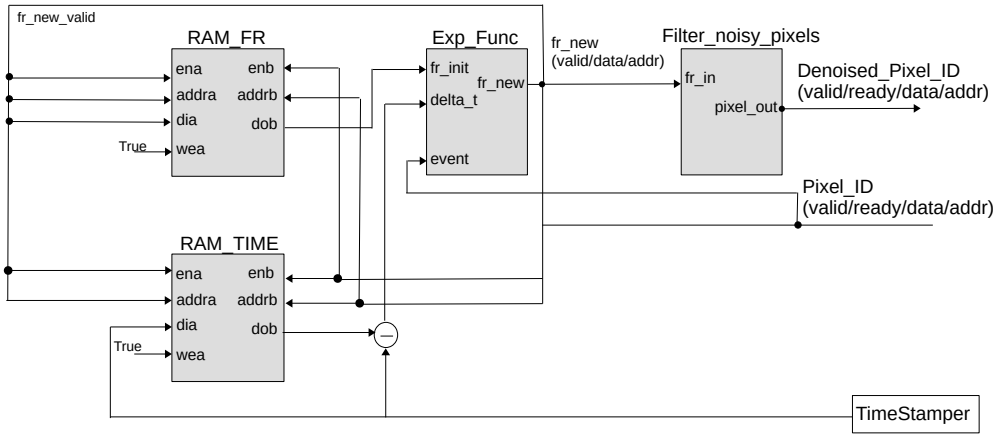

Figure 1

Figure 1 presents a possible algorithm for filtering the flicker noise. This algorithm can be implemented on FPGA, ASIC, or at the software level. Each event arrival updates the pixel's firing rate using equation 1 and extracting the pixel's information from the RAMS. Filter_noisy_pixels block filters out the pixels with unusually high firing rates. The boundary of this filter can be calculated based on the average of all the firing rates over a time window to make the algorithm adaptive to the changes. The main resource utilization of this algorithm would be the Block RAMs to save the pixels' firing rates and timing information of the latest event while the computational part is very light.
For hardware-based solutions, an SNN filter might also be ideal, as it aligns well with DVS principles and has already been used in other research for filtering the flicker noise of the event-based images [5] or denoising [6].

Question 2: Training a SNN with temporal data

Sequence and Temporal learning is one of the strengths of Spiking neural networks. SNNs use time dimension of the data and capture order of events. However with insufficient training data or lack of complementary techniques this feature of SNNs can remain underutilized.

In the application described above you want to classify 4 gestures:

- circulating hand clockwise (CW)
- circulating hand counter clockwise (CCW)
- clapping
- playing drum

You train and deploy your model. With your flicker removal strategies it works well under various indoor lightning conditions. However your model seems to perform poorly for the two first classes and it confuses CW and CCW movements. How do you approach this problem?

Answer

Clockwise and counter-clockwise rotations have closer Sequence and Temporal features while the other two gestures are farther apart. Here's how I would approach diagnosing and addressing this issue:

- Augment the dataset with additional samples of CW and CCW motions, ideally with different lighting conditions and various angles. Since rotation is sensitive to small changes, manufactured data can help the model learn rotational differences better.
- Refine the model architecture for better temporal sensitivity: for instance, use residual connections in the SCNN architecture which helps with overcoming the vanishing gradient problem and the possibility of increasing the network depth. The other refinement would be using a recurrent spiking neural network in the system (Of course if it can be available on hardware as well).
- Enhance the preprocessing to improve directional features
- Visualize and evaluate model activations to ensure the models extract relevant features while training.
- Use learning rules adapted for temporal data: in addition to traditional backpropagation, SNNs often benefit from bio-inspired learning algorithms, like Spike-Timing Dependent Plasticity (STDP) approaches. These rules allow SNNs to capture dependencies and relationships based on the precise timing of spikes.

References

[1] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza, J. Kusnitz, M. Debole, S. Esser, T. Delbruck, M. Flickner, and D. Modha, "A low power, fully event-based gesture recognition system," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7388–7397.

[2] O. Richter, Y. Xing, M. De Marchi, C. Nielsen, M. Katsimpris, R. Cattaneo, Y. Ren, Y. Hu, Q. Liu, S. Sheik, T. Demirci, and N. Qiao, "Speck: A smart event-based vision sensor with a low latency 327k neuron convolutional neuronal network processing pipeline," 2024.

[3] R. Graca, B. Zhou, Sheng snd McReynolds, and T. Delbruck, "Scidvs: A scientific event camera with 1.7% temporal contrast sensitivity at 0.7 lux," 2024.

[4] H. Asgari, N. Risi, and G. Indiveri, "Fpga implementation of an event-driven saliency-based selective attention model," in *2022 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 2022, pp. 307–311.

[5] K. Xiao, X. Cui, K. Liu, X. Cui, and X. Wang, "An snn-based and neuromorphic-hardware-implementable noise filter with self-adaptive time window for event-based vision sensor," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8.

[6] A. Rios-Navarro, S. Guo, G. Abarajithan, K. Vijayakumar, A. Linares-Barranco, T. Aarrestad, R. Kastner, and T. Delbruck, "Within-camera multilayer perceptron dvs denoising," 2023. [Online]. Available: https://arxiv.org/abs/2304.07543