Converting Data for Spiking Neural Network Training

Erik Sadovsky, Maros Jakubec, Roman Jarina
Department of Multimedia and Information-Communication Technologies-FEIT,
University of Zilina, Zilina, Slovak Republic

***Abstract*—The application of spiking neural networks (SNNs) for processing visual and auditory data necessitate the conversion of traditional neural network datasets into a format suitable for spike-based computations. Existing datasets designed for conventional neural networks are incompatible with SNNs due to their reliance on spike timing and specific preprocessing requirements. This paper introduces a comprehensive pipeline that enables the conversion of common datasets into rate-coded spikes, meeting processing demands of SNNs. The proposed solution is evaluated on Spike-CNN trained on Time-to-First-Spike encoded MNIST and compared with the similar system trained on the neuromorphic dataset (N-MNIST). Both systems have comparative precision; however the proposed solution is more energy efficient than the system based on neuromorphic computing. Since, the proposed solution is not limited to any specific data form and can be applied to various types of audio/visual content. By providing a means to adapt existing datasets, this research facilitates the exploration and advancement of SNNs across different domains.**

Keywords*—SNN; rate coding; spike timing; data conversion; MNIST

I. Introduction

Spiking Neural Networks (SNNs) have emerged as a highly promising research direction, bridging the gap between neuroscience and machine learning. By emulating the behaviour of biological neurons and their asynchronous communication through discrete spikes, SNNs offer a compelling computational framework for modelling and understanding neural processes [1], [2]. However, a significant obstacle in fully realizing the potential of SNNs lies in the lack of dedicated databases specifically designed for their training and evaluation.

In contrast to conventional neural networks that are trained using readily and widely available datasets such as MNIST or ImageNet (which are widely used in computer vision and pattern recognition fields among many others), SNNs require special data representations that capture the temporal dynamics of neural processing in a form of spikes. The precise timing of the spikes becomes crucial for encoding and processing information, necessitating a departure from traditional data formats [3]. Consequently, substantial research efforts are currently focused on developing comprehensive databases tailored explicitly for SNN training and evaluation.

Despite notable progress in the field of SNNs, the development of specialized databases for training and testing remains an ongoing research challenge. Currently, only a limited number of publicly available datasets, such as the Spiking Neural Network Architecture (SNA), N-MNIST [4], DVS Gesture [5], and N-TIDIGITS [6], have been specifically designed for SNNs. These datasets enable training and testing of SNNs across various tasks, including decoding neural activity, image classification, gesture recognition, and speech recognition.

The availability of dedicated datasets is crucial for advancing the field of SNNs, as they serve as the foundation for training and evaluating network performance. However, existing datasets for SNNs (usually recorded using a specialized neuromorphic device) are limited in size and diversity, hindering the exploration of SNN capabilities across different domains and applications. This limitation underscores the pressing need to develop methodologies for converting and adapting conventional datasets into formats suitable for SNN training.

Our objective is to explain a processing pipeline for converting amplitude-based data into time/rate encoded spikes and compare a performance of SNN trained on such data with the performance of the SNN trained on specialized neuromorphic dataset. Such conversion process involves the transformation of input data into spike-based representations that preserve the temporal information necessary for accurate neural computation. This conversion necessitates careful consideration of various factors, including spike encoding schemes, spike rates, and the representation of spike timing. Moreover, it is essential to ensure that the converted data maintains the underlying structure and statistical properties of the original data to guarantee meaningful and reliable training of SNNs. By bridging the gap between conventional data formats and SNNs, our work aims to empower researchers and practitioners to overcome the limitations imposed by the scarcity of SNN-specific databases.

In this paper, we propose specifically an approach to converting image data (conventionally represented by pixel intensity values in matrix form) into spike-based representations incorporating temporal encoding techniques. We study and evaluate the proposed approach on MNIST dataset. To prove our concept, we compare the performance of developed SNN architectures trained on the spike-converted MNIST database with ones trained using the N-MNIST database, created by capturing MNIST images using neuromorphic Dynamic Vision Sensor (DVS) camera [7]. N-MNIST is a widely used benchmark dataset for evaluating the performance of SNN models in various tasks.

This paper is organized as follows. Section II introduces Spiking Neural Networks (SNNs) and emphasizes the need for new databases. Section III provides an overview of SNNs applications developed on the N-MNIST database. Section IV outlines the proposed methodology, including data conversion, database description, and network settings. Section V presents the results, demonstrating the effectiveness of SNNs trained on the newly acquired database. Finally, Section VI concludes the paper by summarizing the findings, discussing research implications and limitations, and providing recommendations for future studies.

II. Spiking Neural Network – Theoretical Background

Despite the enormous efforts of scientists and evidently great progress in information and cognitive science, the human brain, with its billions of interconnected neurons, remains an enigmatic engine capable of complex cognitive processes. In recent years, there has been a surge of interest in developing computational models that emulate the functionality of biological neural networks. While traditional Artificial Neural Networks (ANNs) have been successful in numerous applications in diverse domains, they fall short in capturing the temporal dynamics and binary nature of spiking neurons observed in biological systems.

Spiking neural networks present a promising alternative to ANNs, providing a more biologically realistic approach to modelling neural computation. By communicating through discrete binary events known as spikes, SNNs mimic the action potentials observed in real neurons. This temporal coding scheme enables SNNs to capture the dynamics and synchronization observed in biological neural systems, opening new avenues for understanding brain function and developing advanced cognitive computing systems.

A. Neuronal Dynamics in SNNs

The core of a SNN lies in the dynamics of its constituent spiking neurons. Unlike traditional ANNs, which operate using real-valued activations, SNNs leverage the binary nature of spiking neurons to encode and process information through time.

1) Integrate and fire model: The Integrate and Fire (IF) [8] model represents a fundamental building block of SNNs. In this simplified model, the membrane potential of a neuron, denoted as V, integrates the input spike trains it receives. Once the membrane potential surpasses a threshold voltage, the neuron generates an output spike. The dynamics of the membrane potential in the IF model can be described as:

$$\frac{du}{dt} = R \cdot I(t), \qquad u < V_{th} \tag{1}$$

where u denotes the membrane potential, the derivative du/dt represents the rate of change of the membrane potential, R is the membrane resistance, $I(t)$ represent the input spike train, and V_{th} is the threshold voltage.

2) Leaky integrate and fire model: The Leaky Integrate and Fire (LIF) model builds upon the IF model by incorporating the concept of leakage. In biological neurons, the membrane potential gradually decays towards a resting potential due to ion leakage. The LIF model accounts for this phenomenon by including a leakage term in the dynamics of the membrane potential. The LIF model can be expressed as:

$$\tau_m \frac{du}{dt} = -(u - u_{rest}) + R.I(t), \qquad u < V_{th} \tag{2}$$

where τ_m represents the membrane time constant, u_{rest} is the resting potential. Fig. 1 provides a visual representation of the key elements and parameters characterizing a LIF neuron.

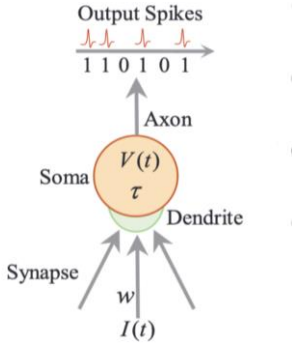

Fig. 1. LIF neuron characterized by membrane potential *V*, membrane time constant *τ*, input *I(t)*, and synaptic weight *w*.

B. Impulse Coding

Impulse coding is a fundamental aspect of SNNs, as it involves the transformation of data into an impulse-based format that enables efficient processing within these networks. The objective of impulse coding is to preserve relevant information while generating a stream of spikes. However, determining the importance of specific information and developing a unified approach to impulse coding remains a complex and context-dependent challenge. The following subsections briefly discuss three mechanisms of impulse coding in SNNs, namely rate encoding, temporal encoding, and population coding.

1) Rate encoding: Rate encoding, also known as rate coding, is a widely studied and utilized method of encoding information in SNNs. It's based on the assumption that the average firing rate of neurons over a specific time interval carries the desired information. By modulating the firing rate of neurons, different stimuli can be represented. The rate encoding approach offers a straightforward and intuitive method for representing information using spikes.

The strength of the stimulus representation is believed to increase with the firing rate. However, the precise mapping between firing rate and stimulus intensity can vary depending on the neural population and the specific encoding scheme employed. The rate encoding method provides a reliable means of representing information in SNNs and has been successfully applied in various applications.

2) Temporal encoding: Temporal encoding is another mechanism employed in impulse coding, which focuses on the precise timing of spikes to represent information. Instead of relying solely on the firing rate, temporal encoding

emphasizes the temporal order and precise timing of individual spikes. The relative timing of spikes across multiple neurons can convey specific features or patterns of stimuli.

In temporal encoding, the timing of spikes within a spike train carries the information, such as the duration between spikes or the occurrence of specific spike patterns. The brain has the remarkable ability to decode and interpret these temporal patterns to extract meaningful information. Temporal coding offers a rich representation that captures fine-grained details of stimuli and enables precise temporal processing in neural networks.

3) Population encoding: In addition to rate encoding and temporal encoding, population encoding has been introduced as a third category of impulse coding. Population encoding involves the joint activity of multiple neurons to encode information. Rather than relying on the individual firing rates or precise timing of spikes, population encoding considers the collective behaviour of a group of neurons.

The underlying principle of population coding is that the combined activity of a population of neurons carries information that cannot be represented by individual neurons alone. By analysing the distributed patterns of activity across the population, specific features or stimuli can be decoded. Population coding provides a powerful mechanism for encoding complex information and has been observed in various biological sensory systems.

III. Related Work

In recent years, several studies have investigated a variety of SNN approaches, using the N-MNIST database, a widely used benchmark dataset for performance evaluation (see Section IV for details on N-MNIST). He et al. [9] compared the performance of the feedforward SNNs and recurrent neural networks (RNNs). The authors modified the N-MNIST database by compressing individual spiking events along the temporal axis and utilized the leaky integrate and fire (LIF) model as the spiking neuron model. Their findings indicated that SNNs generally outperformed conventional RNNs in terms of accuracy. However, with the adaptation of loss functions and the incorporation of Long Short-Term Memory (LSTM) networks, RNNs achieved competitive accuracy with SNNs. This study highlighted the advantage of SNNs in processing features represented by a sparse set of spikes. Another approach by Cohen et al. [10] introduced the method of inverse synaptic kernels for training SNNs on N-MNIST. The authors constructed a spiking neural network with a hidden layer comprising up to 10,000 neurons and achieved a high classification rate of 92.87% on the N-MNIST test subset. This work demonstrated the potential of leveraging biologically inspired principles to further enhance the performance of SNNs.

Wu et al. [11] proposed a novel architecture by incorporating a population of neurons in the output layer of a convolutional SNN. The output spike sequence from this layer represented population coding, which improved the discriminative capabilities of the network. The experiments conducted on the N-MNIST and DVS-CIFAR10 databases showed remarkable accuracies of 99.53% and 60.5%, respectively. This study highlighted the effectiveness of population coding in visual recognition tasks using SNNs.

For event-based features as (SNN input data), Ramesh et al. [12] introduced the Event-Based structural Descriptor (EBD) that captures a spatio-temporal structure using a log-polar grid and applied it to various computer vision problems, including N-MNIST classification. They proved the efficacy of event-based representations in capturing spatiotemporal information and leveraging it for robust classification in SNN frameworks. Their classifier achieved a high accuracy of 97.95% on the N-MNIST test subset.

Addressing the challenges associated with SNN training, Shrestha et al. [13] explored the non-differentiability of the spike generation function and proposed a solution for converting existing databases into spike-based representations. They introduced the SLAYER algorithm, inspired by backpropagation, enabling the training of both feedforward and convolutional SNNs. The N-MNIST database was used to demonstrate the algorithm's effectiveness and the conversion process from image-based to spike-based representations.

Table I summarizes the accuracy achieved by various approaches on the N-MNIST dataset, providing a comprehensive performance comparison of the state-of-the-art SNN-based methods.

TABLE I. Performance Comparison of the State-of-the-Art Methods on the N-MNIST Dataset

Authors	Method	Accuracy (%)
Sironi et al., 2018 [14]	HATS	99.10
Lee et al., 2020 [15]	Spike based supervised gradient descent	99.09
Bi et al., 2019 [16]	Graph based object classification	99.00
Jin et al., 2019 [17]	HM2-BP	98.84
Yousefzadeh et al., 2018 [18]	Active perception with DVS	98.80
Wu et al., 2018 [19]	Spatiotemporal backpropagation	98.78
Lee et al., 2016 [20]	Training SNN using backpropagation	98.74
Ramesh et al., 2017 [12]	Event-Based Descriptor	97.95
Liu et al., 2020 [21]	Segmented probability-maximization	96.30
Kaiser et al., 2020 [22]	DECOLLE	96.00
Cohen et al., 2016 [10]	Inverse synaptic kernels for training SNN	92.87

The Spiking Heidelberg Digits (SHD) and the Spiking Speech Command (SSC) [23] datasets are both audio-based classification datasets that provide input spikes and output labels for different spoken digits and commands. Both of these datasets were created using a software conversion that was based on mathematical models of inner auditory system.

The IBM gestures dataset [24] contains spike trains of gesture movement recordings under different illumination conditions. It is one of the most popular real world scenario datasets for training SNNs.

IV. METHODS

In this section, we present a comprehensive methodology that enables the conversion of various datasets into temporal-encoded spikes suitable for SNN processing. The proposed approach combines techniques of data preprocessing, feature extraction, and network configuration to ensure the compatibility of the converted spikes with the SNN architecture.

A. Datasets

1) MNIST: MNIST is a standard database consisting of grayscale images of handwritten digits. It contains 60,000 training images and 10,000 test images, with each image having a size of 28×28 pixels. One of the advantages of using this database is that it requires minimal data preprocessing since most machine learning libraries have built-in support for MNIST. Although MNIST is not encoded in spikes, it can be utilized for SNN development in such a way that classical ANNs are first trained on MNIST and then converted to SNNs. It should be noted that MNIST does not include a separate validation set. If needed, a few samples from the training set have to be separated for such a purpose.

2) N-MNIST: Neuromorphic MNIST (N-MNIST) [4] is a spike-based representation of the MNIST database. It captures the dynamics of handwritten digits using a Dynamic Vision Sensor (DVS) camera. N-MNIST consists of the same number of samples as MNIST. Each sample is encoded as a binary file, storing pixel index (x, y), event type (ON or OFF), and the event timestamp. The events represent changes in light intensity. N-MNIST provides dynamic sequences with a duration of 300 ms and a resolution of 34×34 pixels. The motion in N-MNIST is inspired by the saccadic eye movement, featuring rapid movements in three directions lasting 100 ms each. This database enables the exploration of SNNs and their performance on tasks involving temporal information, serving as an alternative to static image-based databases.

B. Conversion Procedure: Transforming Data for Effective SNN Training

The first step for enabling SNN training on the MNIST dataset is to encode the data samples into time-distributed spike sequences.

The conversion pipeline consists of the following steps:

- Loading the image data
- Preprocessing the image data
- Scaling the image data
- Converting the image data into spike sequences
- Saving the converted spike sequences for later use in training

To implement this spike encoding procedure, several libraries written in Python have been used: *Pytorch* and *Pytorch-vision* for MNIST loading, *snnTorch* library for temporal encoding and *h5py* library for saving the converted spike sequences.

The next step was to preprocess the image data to be compatible with the *snnTorch* library. One main preprocessing step was to scale the image values between 0 and 1. This step was mandatory as it was a requirement from the *snnTorch* library. One can do multiple preprocessing operations step such as resizing, rotating, etc, at this step.

The scaled values of the data are then used as an input for the spike generation module of the *snnTorch* library (this module enables to use several encoding schemes, either of rate or temporal types). The simulation time of the temporal encoded samples may be specified to a desired value.

The output from the spike generation module is an array of discrete values 0 and 1, in which value of 1 represents a spike. The array is a two-dimensional array of [T, U] size, in which the first dimension represents the simulation time index and the second one corresponds to the feature number, T is the total number of simulation time steps, and U represents the number of features (which matches SNN input layer size).

The final step is to save the generated spike data for the later use in SNN training. A common practice in neuromorphic datasets is to save the data in the form of arrays that correspond to: coordinates (x,y), spike times, and labels. Each event (spike) has a coordinate that corresponds to the index of the input neuron firing a specific spike, and a timestamp. In addition, a label is assigned to the whole sample. We have followed this common practice and reformulated the encoded spike data into corresponding arrays as mentioned above.

In our experiments, we have chosen a temporal encoding scheme, because of lower number of spikes needed to carry the information. This encoding scheme belongs to the group of temporal encoding schemes. As opposed to the rate encoding schemes, temporal encoding schemes contain a smaller number of spikes. In rate encoding schemes the average number of spikes represents the information. However, in temporal encoding schemes, the precise timing of a single spike carries the information. There are several temporal encoding schemes available. [25]. In this experiment, we applied the Time-to-First-Spike encoding procedure (T2FS) [26]. In the case of T2FS, the information is carried in the time of the first spike from the beginning of the simulation. It was experimentally proven, that tactile systems (e.g., at the fingertips) use a similar scheme to encode and transmit information about touch. Also, it has been suggested that the first spike carries twice as much information compared to rate encoding [27].

Lower number of spikes has a positive impact on overall hardware requirements, especially on the size of the batch in GPU. Also, this lower number of spikes reflects lower requirements for energy if a SNN processing this dataset would be implemented on hardware. The simulation time of samples was set to 30 ms as opposed to 300 ms simulation time of N-MNIST samples. The shorter sample time was selected to investigate the ability of SNNs to process samples with short sample time.

The encoding procedure of the *spikegen* module returns a multi-dimensional array with values 0 and 1. Note, due to

differences between input structure of the library used for SNN training with converted samples and the output structure of the *spikegen* module, the structures need to be reorganized to fit each other. The proposed pipeline block diagram is shown in Fig. 2.

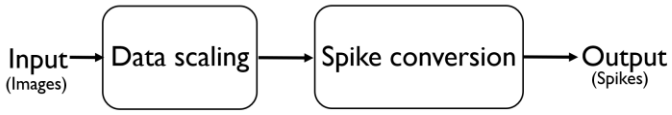

Fig. 2. Proposed pipeline block diagram.

C. Baseline System

The Spiking Convolutional Neural Network (SCNN), that is similar to the structure published in [13] was chosen as a baseline architecture. The architecture of the baseline system is shown in Table II.

The baseline model was trained on N-MNIST. The output layer of the model uses the rate encoding method, where the neuron with a higher spike rate is selected as the neuron representing the class. The neurons were trained to spike 60 times for the representing (true) class and 10 times otherwise (false class). The simulation time of SNN was 300 ms. The trained SNN was able to classify samples with processing delay of 150 ms. Although the process was more biologically plausible, its computational cost may be a disadvantage.

TABLE II. THE ARCHITECTURE OF THE BASELINE SYSTEM

Layer	Parameters
Input	34×34×2
Conv1	12 kernels (5×5)
Delay1	-
Pool1	2×2
Delay2	-
Conv2	64 kernels (5x5)
Delay3	-
Pool2	2×2
Delay4	-
Fc1	10

D. The Proposed System

As an alternative we propose an approach using a software conversion of the MNIST dataset that may be widely available and without the need of a specialized hardware (in contrast to N-MNIST).

The proposed architecture is again a SCNN similar to the baseline, except the input layer. The size of the input layer corresponds with the size and format of the MNIST image data. The structure of the model consists of convolutional layers followed by pooling. After each layer also a time delay layer is applied. The delay layer is used as a special layer in SNNs. The SCNN output layer is a spiking fully connected layer with 10 neurons corresponding with the number of classes to be recognized. The whole architecture is shown in Table III.

The proposed system was trained from scratch for 80 epochs. The time of the whole training process was around 2.5 hours (using NVIDIA RTX 2060TI with 6GB of GPU memory). We used the Adam optimizer with starting learning rate of 0.001. The learning rate parameter was modified by the *ReduceLROnPlateau* learning rate scheduler, which modified the value of the learning rate based on criteria. The batch size of both train and test subsets was set to 32.

To implement and train the proposed SCNN on the converted MNIST dataset we applied the *Pytorch* library along with the *slayerPytorch* library that includes an implementation of SNN training using *Pytorch*. SNNs implemented using this library consists of Spike Response Model spiking neurons. The architecture is trained using the Spike SLAYER algorithm. This algorithm uses a surrogate gradient approach to overcome difficulties with training SNNs. This library contained all building blocks to create a SNN. This includes special layers that were made of spiking neurons, spike processing and the surrogate gradient method. There are more parameters that were needed to be configured. Most of these parameters were related to the *slayerPytorch* library and were used to control the simulation of SNN. We used similar parameters as the baseline, except that our samples had different simulation length. Note, our proposed model was trained for temporal encoding, where a lower number of spikes were needed for the model to classify a sample than in the case of the rate encoding that was used in the baseline.

TABLE III. ARCHITECTURE OF THE MODEL

Layer	Parameters
Input	28×28
Conv1	12 kernels (5×5)
Delay1	-
Pool1	2×2
Delay2	-
Conv2	64 kernels (5x5)
Delay3	-
Pool2	2×2
Delay4	-
Fc1	10

V. EXPERIMENTAL RESULTS AND DISCUSSION

The trained architecture was able to achieve accuracy of 98.79% on the MNIST test set. The changes in accuracy and loss during the training process are shown in Fig. 3 and Fig. 4.

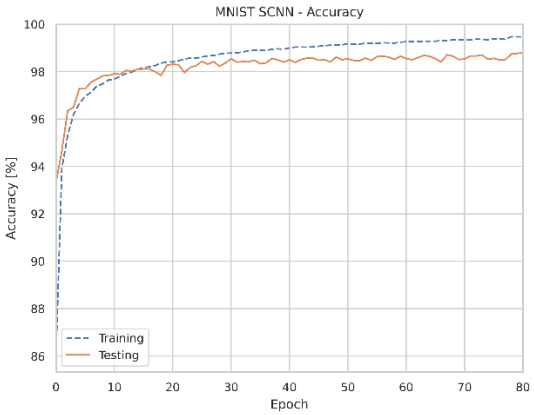

Fig. 3. Accuracy on subsets during the training process.

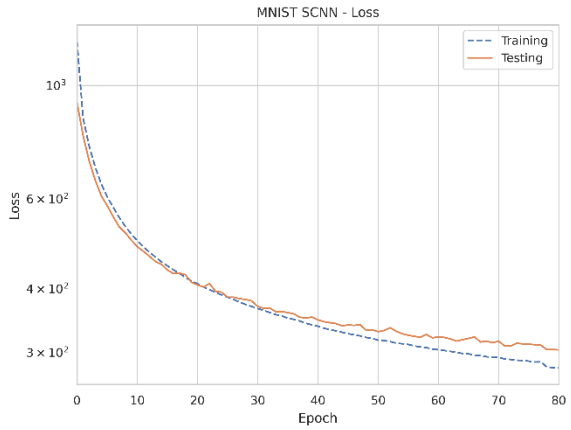

Fig. 4. Loss of the model during the training process.

The accuracy of the baseline system on the N-MNIST test set was 99.2%. The performance of the baseline system is comparable with other systems trained on N-MNIST and published. Although, the accuracy of our trained model was not higher than the baseline system, it is still a competitive number.

The improvement of our model is in spike efficiency. Our trained model was able to classify MNIST samples with only average of 27 spikes on the output layer during the simulation time. The baseline system for N-MNIST used 150 spikes on average during the simulation time. Our proposed system uses a more energy-efficient temporal encoding method, while the baseline system used rate encoding. Also, the number of spikes that are on the input of the SNN is lower. Baseline system for N-MNIST used around 4100 spikes on average, while our system used only 784 spikes. When encoding an image, the spikes are first occurring on the indexes of pixels with higher brightness intensity, while spikes on indexes of pixels with lower brightness intensity occur later. Note, each represented pixel contains only one spike. This means, that for MNIST images with a resolution of 28×28 pixels only 784 spikes occur for each image. Such a number of spikes are much smaller than in the case of the N-MNIST dataset, in which each sample around 4100 spikes on average.

Our proposed model is more energy efficient not only on the input spikes, but also on the output spikes. The comparison of the baseline system and our proposed system is summarized in Table IV.

TABLE IV. COMPARISON OF THE BASELINE SYSTEM AND OUR PROPOSED SYSTEM

	Baseline system	**Proposed system**
Dataset	N-MNIST	MNIST (converted with the proposed pipeline)
Encoding	Rate encoding	Temporal encoding
Number of spikes on input (average)	4100	784
Number of spikes on output	150	27
Accuracy	99.2%	98.79%

Although our experiments are carried out on MNIST benchmark dataset, the methodology we present is versatile and applicable to other image datasets as well as to diverse data modalities, including audio or biological signals (e.g. in the form of spectrograms).

VI. CONCLUSION

In this paper we have presented a software conversion process that uses an energy efficient temporal encoding method to convert static image data into a format of spikes distributed in time. The proposed method was compared with a baseline method that used a specialized hardware for converting the same dataset. The baseline system used rate encoding. The functionality of the proposed method was examined on SNN that used a similar architecture to the baseline system. The results of the proposed solution in terms of accuracy were competitive with the baseline. However, the SNN trained using the proposed temporal encoding needs a significantly lower number of spikes in both input and output spike trains to correctly classify the dataset.

We envise that the proposed pipeline will not only facilitate improved training and testing of SNNs but also inspire the development of larger datasets that cover a broader application domain. Through these efforts, we attempt to unlock the immense potential of SNNs and neuromorphic computing while advancing our understanding of brain-inspired computation.

The future improvements for the proposed pipeline may be in experiments with more datasets to be converted into spike trains and then used for training SNNs. The length of samples in time may play a role in the accuracy of the trained model and would need to be further investigated.

ACKNOWLEDGMENT

This work was supported by the Slovak Grant Agency KEGA under contract no. KEGA 008ZU-4/2021 and also by the ERDF project of Operational Programme Integrated Infrastructure, ITMS2014+ code 313011ASK8.

REFERENCES

[1] S. Ghosh-Dastidar and H. Adeli, ‘Spiking neural networks’, Int. J. Neural Syst., vol. 19, no. 04, pp. 295–308, Aug. 2009, doi: 10.1142/S0129065709002002.

[2] S. Thorpe, A. Delorme, and R. Van Rullen, 'Spike-based strategies for rapid processing', Neural Netw., vol. 14, no. 6, pp. 715–725, Jul. 2001, doi: 10.1016/S0893-6080(01)00083-1.

[3] T. Zhang, M. R. Azghadi, C. Lammie, A. Amirsoleimani, and R. Genov, 'Spike sorting algorithms and their efficient hardware implementation: a comprehensive survey', J. Neural Eng., vol. 20, no. 2, p. 021001, Apr. 2023, doi: 10.1088/1741-2552/acc7cc.

[4] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, 'Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades', Front. Neurosci., vol. 9, 2015, Accessed: Feb. 07, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2015.00437

[5] 'DVS128 Gesture Dataset - IBM Research'. https://research.ibm.com/interactive/dvsgesture/ (accessed Feb. 07, 2023).

[6] 'TIDIGIT Spikes Dataset', Google Docs. https://docs.google.com/document/d/1Uxe7GsKKXcy6SlDUX4hoJVAC0-UkH-8kr5UXp0Ndi1M/edit?usp=embed_facebook (accessed Feb. 07, 2023).

[7] P. Lichtsteiner, C. Posch, and T. Delbruck, 'A 128\times 128 120 dB 15 µs Latency Asynchronous Temporal Contrast Vision Sensor', IEEE J. Solid-State Circuits, vol. 43, no. 2, pp. 566–576, Feb. 2008, doi: 10.1109/JSSC.2007.914337.

[8] J. Feng and David Brown, 'Integrate-and-fire Models with Nonlinear Leakage', Bull. Math. Biol., vol. 62, no. 3, pp. 467–481, May 2000, doi: 10.1006/bulm.1999.0162.

[9] W. He et al., 'Comparing SNNs and RNNs on Neuromorphic Vision Datasets: Similarities and Differences'. arXiv, May 02, 2020. Accessed: May 19, 2023. [Online]. Available: http://arxiv.org/abs/2005.02183

[10] G. K. Cohen, G. Orchard, S.-H. Leng, J. Tapson, R. B. Benosman, and A. van Schaik, 'Skimming Digits: Neuromorphic Classification of Spike-Encoded Images', Front. Neurosci., vol. 10, 2016, Accessed: May 19, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2016.00184

[11] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, and L. Shi, 'Direct Training for Spiking Neural Networks: Faster, Larger, Better', Proc. AAAI Conf. Artif. Intell., vol. 33, no. 01, Art. no. 01, Jul. 2019, doi: 10.1609/aaai.v33i01.33011311.

[12] B. Ramesh, H. Yang, G. Orchard, N. A. Le Thi, S. Zhang, and C. Xiang, 'DART: Distribution Aware Retinal Transform for Event-Based Cameras', IEEE Trans. Pattern Anal. Mach. Intell., vol. 42, no. 11, pp. 2767–2780, Nov. 2020, doi: 10.1109/TPAMI.2019.2919301.

[13] S. B. Shrestha and G. Orchard, 'SLAYER: Spike Layer Error Reassignment in Time', in Advances in Neural Information Processing Systems, Curran Associates, Inc., 2018. Accessed: Feb. 17, 2023. [Online]. Available: https://proceedings.neurips.cc/paper/2018/hash/82f2b308c3b01637c607ce05f52a2fed-Abstract.html

[14] A. Sironi, M. Brambilla, N. Bourdis, X. Lagorce, and R. Benosman, 'HATS: Histograms of Averaged Time Surfaces for Robust Event-Based Object Classification', in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Jun. 2018, pp. 1731–1740. doi: 10.1109/CVPR.2018.00186.

[15] C. Lee, S. S. Sarwar, P. Panda, G. Srinivasan, and K. Roy, 'Enabling Spike-Based Backpropagation for Training Deep Neural Network Architectures', Front. Neurosci., vol. 14, 2020, Accessed: Jul. 28, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2020.00119

[16] Y. Bi, A. Chadha, A. Abbas, E. Bourtsoulatze, and Y. Andreopoulos, 'Graph-Based Object Classification for Neuromorphic Vision Sensing', in 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Oct. 2019, pp. 491–501. doi: 10.1109/ICCV.2019.00058.

[17] Y. Jin, W. Zhang, and P. Li, 'Hybrid Macro/Micro Level Backpropagation for Training Deep Spiking Neural Networks', in Advances in Neural Information Processing Systems, Curran Associates, Inc., 2018. Accessed: Jul. 28, 2023. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/hash/3fb04953d95a94367bb133f862402bce-Abstract.html

[18] A. Yousefzadeh, G. Orchard, T. Serrano-Gotarredona, and B. Linares-Barranco, 'Active Perception With Dynamic Vision Sensors. Minimum Saccades With Optimum Recognition', IEEE Trans. Biomed. Circuits Syst., vol. 12, no. 4, pp. 927–939, Aug. 2018, doi: 10.1109/TBCAS.2018.2834428.

[19] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, 'Spatio-Temporal Backpropagation for Training High-Performance Spiking Neural Networks', Front. Neurosci., vol. 12, 2018, Accessed: May 29, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2018.00331

[20] J. H. Lee, T. Delbruck, and M. Pfeiffer, 'Training Deep Spiking Neural Networks Using Backpropagation', Front. Neurosci., vol. 10, 2016, Accessed: Jul. 28, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2016.00508

[21] Q. Liu, H. Ruan, D. Xing, H. Tang, and G. Pan, 'Effective AER Object Classification Using Segmented Probability-Maximization Learning in Spiking Neural Networks', Proc. AAAI Conf. Artif. Intell., vol. 34, no. 02, Art. no. 02, Apr. 2020, doi: 10.1609/aaai.v34i02.5486.

[22] J. Kaiser, H. Mostafa, and E. Neftci, 'Synaptic Plasticity Dynamics for Deep Continuous Local Learning (DECOLLE)', Front. Neurosci., vol. 14, 2020, Accessed: Jul. 28, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2020.00424

[23] B. Cramer, Y. Stradmann, J. Schemmel, and F. Zenke, 'The Heidelberg Spiking Data Sets for the Systematic Evaluation of Spiking Neural Networks', IEEE Trans. Neural Netw. Learn. Syst., vol. 33, no. 7, pp. 2744–2757, Jul. 2022, doi: 10.1109/TNNLS.2020.3044364.

[24] A. Amir et al., 'A Low Power, Fully Event-Based Gesture Recognition System', in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jul. 2017, pp. 7388–7397. doi: 10.1109/CVPR.2017.781.

[25] J. K. Eshraghian et al., 'Training Spiking Neural Networks Using Lessons From Deep Learning'. arXiv, May 15, 2023. Accessed: May 29, 2023. [Online]. Available: http://arxiv.org/abs/2109.12894

[26] B. Rueckauer and S.-C. Liu, 'Conversion of analog to spiking neural networks using sparse temporal coding', in 2018 IEEE International Symposium on Circuits and Systems (ISCAS), May 2018, pp. 1–5. doi: 10.1109/ISCAS.2018.8351295.

[27] H. P. Saal, S. Vijayakumar, and R. S. Johansson, 'Information about Complex Fingertip Parameters in Individual Human Tactile Afferent Neurons', J. Neurosci., vol. 29, no. 25, pp. 8022–8031, Jun. 2009, doi: 10.1523/JNEUROSCI.0665-09.2009.