Ternarized TCN for µJ/Inference Gesture Recognition from DVS Event Frames

Funding acknowledgement:
877056 - A Cognitive Fractal and Secure EDGE based on an unique Open-Safe-Reliable-Low Power Hardware Platform Node (EC)
180625 - Heterogeneous Computing Systems with Customized Accelerators (SNF)

Ternarized TCN for µJ/Inference Gesture Recognition from DVS Event Frames

Georg Rutishauser*, Moritz Scherer*, Tim Fischer*, Luca Benini*†
*Departement Informationstechnologie und Elektrotechnik, ETH Zürich, Zürich, Switzerland
†Dipartimento di Ingegneria dell'Energia Elettrica e dell'Informazione, Università di Bologna, Bologna, Italy

***Abstract*—Dynamic Vision Sensors (DVS) offer the opportunity to scale the energy consumption in image acquisition proportionally to the activity in the captured scene by only transmitting data when the captured image changes. Their potential for energy-proportional sensing makes them highly attractive for severely energy-constrained sensing nodes at the edge. Most approaches to the processing of DVS data employ Spiking Neural Networks to classify the input from the sensor. In this paper, we propose an alternative, event frame-based approach to the classification of DVS video data. We assemble ternary video frames from the event stream and process them with a fully ternarized Temporal Convolutional Network which can be mapped to CUTIE, a highly energy-efficient Ternary Neural Network accelerator. The network mapped to the accelerator achieves a classification accuracy of $94.5\,\%$, matching the state of the art for embedded implementations. We implement the processing pipeline in a modern $22\,\mathrm{nm}$ FDX technology and perform post-synthesis power simulation of the network running on the system, achieving an inference energy of $1.7\,\mu\mathrm{J}$, which is $647\times$ lower than previously reported results based on Spiking Neural Networks.**

***Index Terms*—Machine learning, Edge computing, Classification algorithms**

I. Introduction

With the increasing ubiquity of embedded smart devices, research into methods of interaction with devices without traditional control interfaces has intensified tremendously. One of the most natural ways for people to interact with their devices is by hand gestures. As with many other fields, works on this and other human-computer interface tasks have recently been dominated by deep-learning based approaches. A trend is to push the processing to the edge devices which capture the data for various reasons, e.g., latency minimization, privacy requirements, and avoiding the energy and component costs of high-bandwidth transmission of raw data to a cloud processing center. Performing this processing on small, battery-powered edge devices requires an efficient approach to the sensing and processing approaches employed. For embedded microcontrollers, this is traditionally done by quantizing networks and leveraging Single Instruction Multiple Data (SIMD) instructions [1], [2].

To enable the highest degree of energy efficiency for embedded devices, all contributors to power consumption must be minimized. Typically, the largest drivers in sensor-based edge applications are the sensors and the processing platform, which can be optimized by employing a hardware accelerator. One such hardware accelerator is CUTIE [3], which enables processing of Ternarized Neural Networks, neural networks that are quantized to use ternary operands. To push the full system's power envelope even further, several researchers have proposed optimizing the energy consumption of the sensor using event-based techniques like Dynamic Vision Sensors (DVS) [4], [5]. DVS cameras operate by transmitting per-pixel events in their field of view whenever the change in a pixel's brightness exceeds a defined threshold. Transmitted events only report the direction of change, i.e., increase or decrease in intensity, resulting in binary data. This makes them unique in that their power consumption is coupled to the movement in their field of view, allowing for ultra-low power consumption during periods of low activity. While DVS camera data are commonly processed using Spiking Neural Networks (SNN), we show that frame-based processing offers a competitive alternative for ultra-low-energy gesture recognition from DVS data at state of the art (SoA) accuracy.

In this paper, we present a ternary Temporal Convolutional Network (TCN) which we train to recognize and classify gestures from the 11-class DVS128 gesture dataset [6], achieving a classification accuracy of $94.5\,\%$. We further describe the deployment of this network to an extended version of CUTIE, showing a decrease in inference energy of $647\times$ with respect to the state-of-the-art.

The contributions of this paper are as follows:

- We present a frame-based DVS gesture recognition algorithm based on the first fully ternarized TCN (to our knowledge) reported in the literature, achieving SoA validation accuracy of $94.5\,\%$.
- We present a modular, end-to-end hardware data acquisition pipeline that allows capturing DVS data in an autonomous, energy-efficient manner, and configurably concatenates them to event frames, enabling further processing with frame-based algorithms.
- We map the proposed network on a highly efficient TNN accelerator (CUTIE) and present post-layout power simulation results in an advanced $22\,\mathrm{nm}$ technology, achieving a classification energy of $1.7\,\mu\mathrm{J}$, a factor of $647\times$ better than the previous state of the art.

The authors would like to thank *armasuisse Science & Technology* for funding this research. Furthermore, we gratefully acknowledge support by the EU H2020 Fractal project funded by ECSEL-JU grant agreement #877056 and the HCSCA project #180625 funded by the Croatian-Swiss Research Programme.

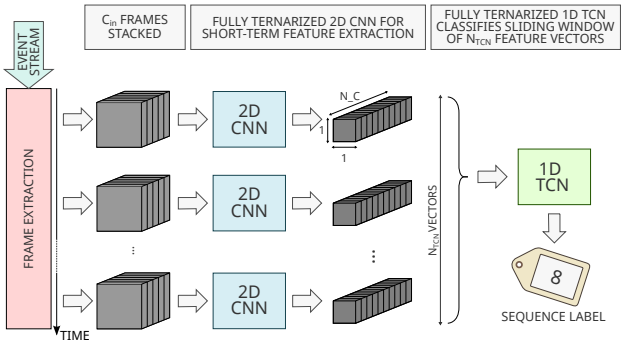

Fig. 1. Overview of the proposed processing pipeline.

II. Related Work

With the recent surge in increasingly sensor-driven smart devices, the domain of gesture recognition has received a lot of attention.

1) Embedded Gesture Recognition Approaches: Correctly identifying gestures in an embedded setting is a problem that has been tackled using several different sensors. One popular approach is based on equipping subjects with wearable sensors like accelerometers or gyrometers [7]–[9]. These approaches typically result in high classification accuracy in the range of $90\,\%$ or more. However, even commercial wearable sensors are often hard to adapt to practical deployment scenarios at their reported accuracies [10], [11], due to inter-person variability. Furthermore, the requirement of being physically connected to the sensor makes these approaches less versatile than those based on contactless sensing.

2) Gesture Recognition with DVS Cameras: Gesture Recognition is one of the popular use-cases for event-based vision systems. Recent works using SNN-based approaches include [6], [12], which deploy their networks onto the IBM TrueNorth and Intel Loihi platforms, respectively. Other works mainly study SNNs on the DVS128 dataset without explicitly deploying them to SNN accelerators, achieving remarkable accuracies in the range of $90\,\%$ - $97.6\,\%$ [13]–[16].

In this work, we add to the growing body of research on Gesture Recognition applications using DVS camera data and demonstrate an alternative approach to SNN-based processing of event data, based on the accumulation and processing of ternary event frames. We further deploy our network to a state-of-the-art embedded accelerator platform and present detailed power simulation results.

III. Ternarized TCN for DVS Gesture Recognition

In this section, we describe our proposed processing pipeline for DVS-based gesture recognition.

A. Overview

The class of cameras we target produce events consisting of three values each: Two spatial coordinates (x_e, y_e) describing the location of the event on the sensor grid, and a polarity $p_e \in \{-1, 1\}$ to indicate a decrease or increase in brightness, respectively. The first stage of our proposed pipeline performs data preparation, aggregating events detected by the DVS camera into 2-dimensional frames. The resulting frames are natively ternary: Pixels where at least one event occurred during a frame interval take the value of the most recent event's polarity, while pixels with no activity take the value 0.

The prepared data is then processed by a two-stage hybrid Convolutional Neural Network (CNN), inspired by [17]. In the first processing stage, C_{in} sequential frames are fed into a fully ternarized 2D CNN. By processing multiple frames in a single inference, the 2D CNN can analyze short-term temporal dependencies, which are encoded by the last layer into 1-dimensional ternary feature vectors.

In the third and final stage, a fully ternarized TCN analyzes the longer-term temporal dependencies by performing inference on a sliding window covering N_{TCN} feature vectors and produces a vector of class scores $V_p \in \mathbb{Z}^{N_c}$, where N_c is the number of classes. For the DVS128 dataset, $N_c = 11$. Finally, the class label of the sequence is given as $Cls = \arg\max_i (V_p)$.

Figure 1 shows a diagram of the proposed processing pipeline. The frame extraction process is shown in Figure 3 and described in more detail in Section III-D.

TABLE I
Topology of the proposed hybrid CNN architecture, consisting of a 2D CNN for short-term feature extraction and a 1D TCN to capture longer-term temporal dependencies. **(C)S/V** indicate (causal) same/valid padding, respectively.

	Layer	Out Chan.	Out Res.	Pad	Dilation
2D CNN	Input	C_{in} (8)	64×64	N/A	N/A
	Conv2D (3×3)	32	64×64	S	1
	MaxPool (2×2)	32	32×32	V	1
	Conv2D (3×3)	N_{ch} (96)	32×32	S	1
	MaxPool (2×2)	N_{ch} (96)	16×16	V	1
	Conv2D (3×3)	N_{ch} (96)	16×16	S	1
	MaxPool (2×2)	N_{ch} (96)	8×8	V	1
	Conv2D (3×3)	N_{ch} (96)	8×8	S	1
	MaxPool (2×2)	N_{ch} (96)	4×4	V	1
	Conv2D (3×3)	N_{ch} (96)	2×2	V	1
	MaxPool (2×2)	N_{ch} (96)	1×1	V	1
1D TCN	Input	N_{ch} (96)	N_{TCN} (5)	N/A	N/A
	Conv1D (2)	N_{ch} (96)	N_{TCN} (5)	CS	1
	Conv1D (2)	N_{ch} (96)	N_{TCN} (5)	CS	2
	Conv1D (2)	N_{ch} (96)	N_{TCN} (5)	CS	4
	Conv1D (5)	11	1	V	1

B. Network Design

The ternarized hybrid CNN/TCN adopts a fully feed-forward architecture, i.e., there are no residual branches in the network. All layers but the first have an identical number N_{ch} of output channels. The two networks' topologies are listed in Table I. The final, fully ternarized network consists only of convolutional layers with no bias, ternary acivations 2 and pooling layers. Consider a layer stack consisting of a convolutional layer with N_i/N_o input/output channels respectively and an optional pooling layer and denote the ternary values as $\mathcal{T} \triangleq \{-1, 0, 1\}$. The layer stack takes as input a tensor $\mathbf{X} \in \mathcal{T}^{N_i \times H_X \times W_X}$, where H_X/W_X are the spatial dimensions. $\mathbf{X}$ is convolved with the convolutional weights

$\mathbf{W} \in \mathcal{T}^{N_o \times N_i \times k \times k}$ and pooled, where k is the kernel size, to yield pre-activations $\mathbf{Z} \in \mathbb{Z}^{N_o \times H_Y \times W_Y}$, shown in Equation 1. $\mathbf{Z}$ is then mapped to ternary activations $\mathbf{Y} \in \mathcal{T}^{N_o \times H_Y \times W_X}$ by channel-wise thresholding as shown in Equation 2 with $\mathbf{t}^{lo}$ and $\mathbf{t}^{hi}$, $\mathbf{t}^{lo}, \mathbf{t}^{hi} \in \mathbb{Z}^{N_o}$.

$$\mathbf{Z} = pool\left(\mathbf{X} * \mathbf{W}\right) \tag{1}$$

$$y_{i,x,y} = \begin{cases} -1, & z_{i,x,y} < t_i^{lo} \\ 0, & t_i^{lo} \leq z_{i,x,y} < t_i^{hi} \\ 1, & z_{i,x,y} \geq t_i^{hi} \end{cases} \tag{2}$$

C. Ternarized Network Training

As a fully ternarized network is not differentiable and thus can not be trained directly using gradient descent-based methods, we train a *fake-quantized* version of the network using the Incremental Network Quantization (INQ) algorithm [18] to ternarize the weights. The training is done in three steps:

1) Training the full-precision network to convergence,
2) quantizing activations, and
3) incremental weight quantization.

During all training stages, Batch Normalization (BN) layers are present after convolutional layers to allow for channel-wise scaling. During the full-precision training phase, we use the hard hyperbolic tangent (Htanh) activation function. After the full-precision network has converged, the Htanh activations are quantized using the parameter-free thresholding function (3), and the network is retrained for a few epochs.

$$thresh(x) = \begin{cases} -1, & x < -0.5 \\ 0, & -0.5 \geq x < 0.5 \\ 1, & x \geq 0.5 \end{cases} \tag{3}$$

During backpropagation, we apply the Straight-Through-Estimator (STE) [19] to the quantizer (3) to propagate the gradient. Finally, the weights are incrementally quantized to the nearest ternary value as detailed in [18]. We quantize the weights in order of decreasing magnitude. The detailed training parameters are listed in Section IV-A.

For deployment, the converged, fake-quantized network is converted into a fully ternarized network without any floating-point parameters. This process consists of folding the BN parameters into the threshold vectors of the following activation layer. For a BN layer following a convolutional layer with N_o channels, bias tensor $\mathbf{B}$ and weight tensor $\mathbf{W}$, the convolution bias and the BN parameters are first folded into a single, channel-wise affine transform with scale factors $\hat{\boldsymbol{\gamma}}$ and biases $\hat{\boldsymbol{\beta}}$ as in Equation 4. Two vectors of lower and upper thresholds $\hat{\mathbf{t}}^{lo}$, $\hat{\mathbf{t}}^{hi} \in \mathbb{R}^{N_o}$ are then computed as in Equation 5. Since negative weights in the BN layer's weights $\boldsymbol{\gamma}$ flip the inequalities used in thresholding, we fold their sign into the convolutional weights and invert the thresholds for all channels where $\boldsymbol{\gamma}$ is negative. Finally, the thresholds are rounded to the next greater integer number (Equation 7). In the final, fully ternarized network, the convolutional layer is replaced with a bias-free version with the modified weights $\tilde{\mathbf{W}}$, while the BN and thresholding layers are replaced with a single thresholding activation using the computed channel-wise thresholds $\mathbf{t}^{lo}$ and $\mathbf{t}^{hi}$, respectively.

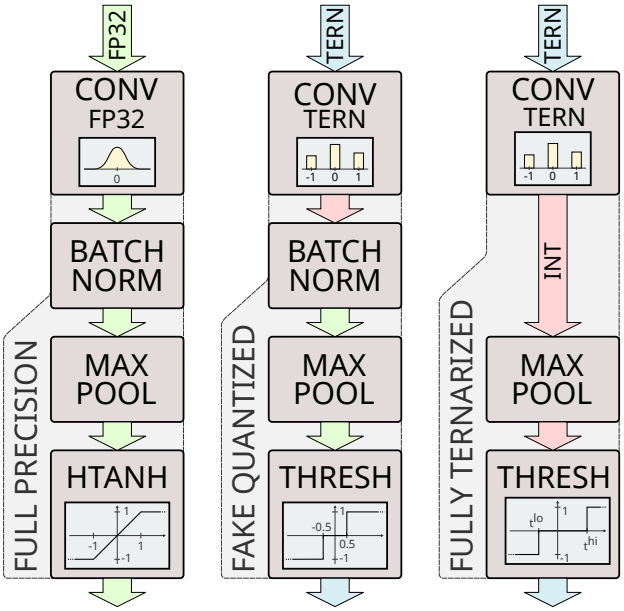

Fig. 2. The three stages of the TNN training and deployment flow: The full-precision network is first trained to convergence, then fake-quantized and retrained. For deployment, the fake-quantized network is integerized by folding the BN parameters into a channel-wise thresholding function.

$$\hat{\boldsymbol{\beta}} = \boldsymbol{\gamma}\frac{(\mathbf{B} - \boldsymbol{\mu})}{\boldsymbol{\sigma}} + \boldsymbol{\beta}, \qquad \hat{\boldsymbol{\gamma}} = \frac{\boldsymbol{\gamma}}{\boldsymbol{\sigma}} \tag{4}$$

$$\hat{\mathbf{t}}^{lo} = \frac{-0.5 - \hat{\boldsymbol{\beta}}}{\hat{\boldsymbol{\gamma}}}, \qquad \hat{\mathbf{t}}^{hi} = \frac{0.5 - \hat{\boldsymbol{\beta}}}{\hat{\boldsymbol{\gamma}}} \tag{5}$$

$$\tilde{\mathbf{W}}_c = \mathbf{W}_c - 2\mathbf{W}_c \mathbb{1}_{\gamma_c < 0} \qquad \forall\ 0 \leq c < N_o \tag{6}$$

$$t_c^{lo,hi} = \left\lceil \hat{t}_c^{lo,hi} - 2\hat{t}_c^{lo,hi} \mathbb{1}_{\gamma_c < 0} \right\rceil \qquad \forall\ 0 \leq c < N_o \tag{7}$$

The resulting network contains only ternary convolutions, integer pooling, and integer thresholding operations.

D. Data Preparation

Unlike previous works [6], [12], which employ fully event-based processing schemes and rely on SNNs to classify event data directly, our proposed approach first aggregates the event stream into ordinary 2D images. While this incurs overhead for data preparation (frames need to be assembled and buffered) and in theory decouples the processing energy from the density of the event stream, we argue that it also offers major advantages which more than compensate for these factors:

Crucially, the generated frame data is natively ternary - the full information content of the event stream is directly encoded in a format that can be processed by the 2D TNN of the first processing stage. CUTIE, the targeted accelerator architecture, exploits the regular nature of CNN and TCN inference together with highly energy-efficient ternary multiply-accumulate (MAC) units to achieve very high energy efficiency. Furthermore, lower event density results in sparser

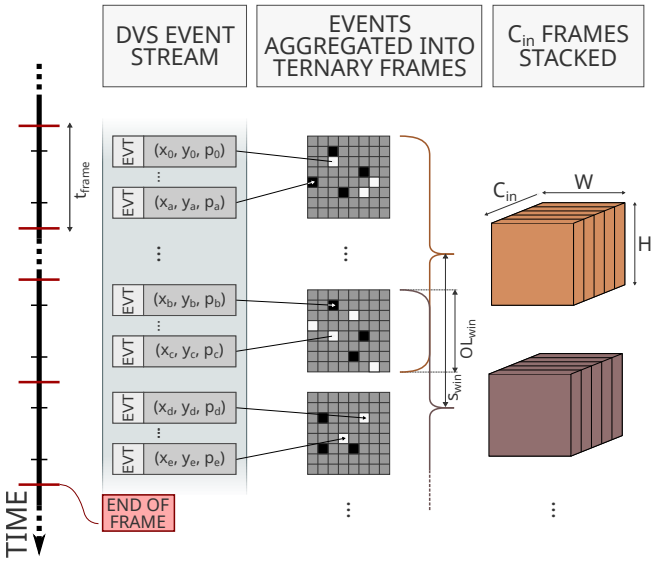

Fig. 3. Frame aggregation for processing by the 2D CNN. All events occurring during a time window of length t_{frame} are assigned to the same frame. The 2D CNN takes C_{in} frames as input, and two successive stacks of C_{in} frames overlap by OL_{win} frames

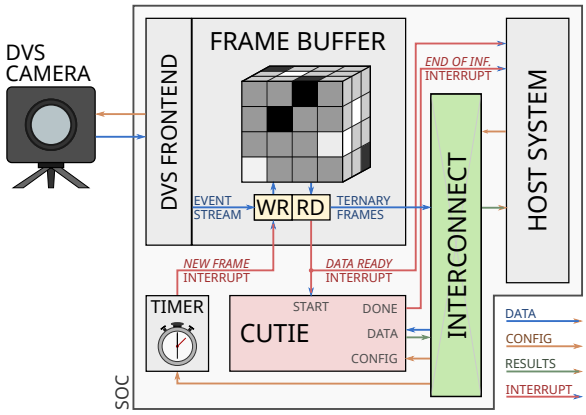

Fig. 4. Proposed system for acquisition and TNN-based classification of DVS data. Event data from the DVS camera is assembled into frames and written into a frame buffer. The end of a frame interval is signaled to the frame buffer by a timer interrupt. Every s_{win} frames, the frame buffer autonomously writes C_{in} frames to the data memory of the CUTIE TNN accelerator and triggers the start of inference. When CUTIE has finished processing the data, the host processor can read the results from CUTIE's register file.

frame data. As detailed in [3], CUTIE performs more efficiently on sparse data.

a) Parameters in data generation: The process of generating frame data from the event stream, illustrated in Figure 3 has several degrees of freedom:

- Frame rate FPS, or, equivalently, frame time $t_{frame} = \frac{1}{FPS}$
- CNN input window size C_{in}; this parameter also dictates the number of input channels to the CNN's first layer.
- Temporal CNN input window stride s_{win}, which gives rise to the input window overlap $OL_{win} = C_{in} - s_{win}$
- Downsampling factor D - from our experimental observations, we choose $D = 2$, i.e., the height and width of the original frame are both halved and the resulting frame has $1/4$ the resolution of the raw camera data.

While all of these parameters must be fixed before training a network on the generated data, the hardware which implements the frame aggregation can be designed to leave them configurable at runtime with negligible complexity overhead. The receptive time interval t_p of the full hybrid network is given by the number of input vectors to the TCN N_{TCN}, the number of frames encoded in a single vector C_{in} and the stride of the CNN input windows s_{win} as $t_p = C_{in} + (N_{TCN} - 1)s_{win}$. Once running, the latency l of the system to react is determined by the window stride and the frame rate as well as the processing time t_{inf}: $l = s_{win}/FPS + t_{inf} \approx s_{win}/FPS$, where t_{inf} can be neglected due to CUTIE's very high throughput compared to the frame time.

E. Mapping to Hardware

We map the ternarized network to a modified version of the CUTIE accelerator [3], which we integrated with a System-on-Chip (SoC) based on the Pulpissimo SoC [20]. In the proposed system, shown in Figure 4, event data is received from an attached DVS camera and aggregated in a frame buffer, which is fully configurable in C_{in} and s_{win}. The start of a new frame can be signaled by a configurable on-chip timer or manually triggered by the host core via a configuration register. When C_{in} frames have been assembled, the frame buffer autonomously writes them to a configurable address and raises an interrupt, which is connected to the system interrupt controller as well as to CUTIE. By configuring the frame buffer to write directly into CUTIE's feature map memory and triggering the inference directly with the interrupt, the system can (after initial configuration) operate without intervention from the host core, apart from the readout of the results. The CUTIE accelerator supports convolutional layers with kernel sizes up to 3×3 and configurable strides, max- and average-pooling layers with configurable kernel sizes and thresholding activations in the form of Equation 2. The modifications to CUTIE [3] consist of additional memory to buffer a sliding window of TCN inputs, added support for dilated TCN kernels and causal 1D convolutions, and a change in the maximum number of channels from 128 to 96. Furthermore, a register file was added into which the unquantized integer outputs of a classification layer can be saved and which is accessible through CUTIE's peripheral interface. With these changes, the complete network proposed in Section III-B can be executed on the accelerator.

IV. Experiments and Results

A. Training Setup and Model Parameters

We trained our model in the QuantLab framework [1] using the Adam optimizer [26], with a starting Learning Rate (LR) of 0.01. A full-precision model was first trained for 100 epochs using cosine learning rate decay to a minimum LR

[1] https://github.com/pulp-platform/quantlab

TABLE II
COMPARISON OF THE RESULTS WITH STATE OF THE ART EMBEDDED AND SPIKING NEURAL NETWORK GESTURE RECOGNITION IMPLEMENTATIONS

	[21]	[22]	[6]	[12]	[23]	[24]	**This work**
Dataset	Short Range Radar	DVS & sEMG	DVS128	DVS128	DVS128	DVS128	DVS128
Input format	Dense features	Events	Events	Events	Event	Event Frames	Event frames
Neural network architecture	CNN & Transformer	SNN	SNN	SNN	3D CNN	Transformer	CNN/TCN
Compute platform	GAP 8	Loihi	TrueNorth	Loihi	GPU	GTX 1080	CUTIE
Weight bitwidth	8 bit	9 bit	ternary	9 bit	8 bit	32 bit	ternary
Activation bitwidth	8 bit	9 bit	1 bit	9 bit	32 bit	32 bit	ternary
Leakage power	-	29 mW [25]	134.4 mW	29 mW [25]	-	-	0.9 mW
Dynamic power	-	137 mW	44.5 mW	-	up to 180 W	up to 180 W	4.4 mW
Receptive window	500 ms	200 ms	105 ms	300 ms	500 ms	750 ms	666.7 ms
Processing latency	9 ms	7 ms	-	11 ms	-	13.1 ms	**0.3 ms**
Energy per Inference	0.47 mJ	1.1 mJ	18.8 mJ	-	>100 mJ	>100 mJ	**1.7 µJ**
Normalized Inf. E. @10 inf/s	-	3.55 mJ	17.9 mJ	-	>100 mJ	>100 mJ	**92 µJ (48 µJ)**[a]
Statistical accuracy	77.15 %	96.0 %	94.6 %	90.5 %	99.6 %	96 %	94.5 %

[a] Energy figure in parentheses assumes power-gated activation memories and combinational logic in CUTIE.

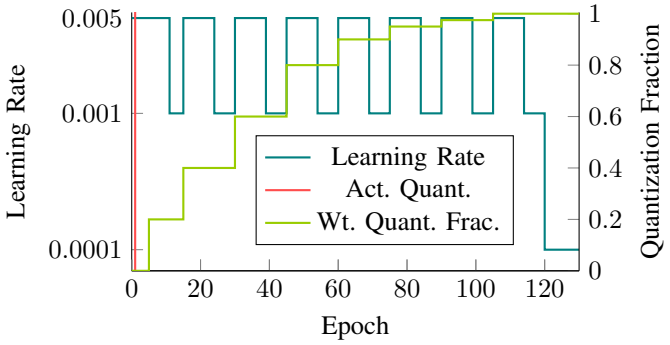

Fig. 5. Quantization and learning rate schedules used to ternarize the full-precision network. The left y-axis measures the learning rate, the right y-axis measures the fraction of weights quantized.

of 0.00001, reaching a classification accuracy of 96.1%. The learning rate and quantization schedules for the quantized training are plotted in Figure 5. The DVS128 dataset was split by subject into a training and a validation set, with subjects 1-23 comprising the training set and the remaining users used as the validation set. The training data was augmented by randomly shifting the input frames by up to 10% of their height/width into each direction and by randomly flipping each pixel's polarity with a probability of $p_{flip} = 0.1$. FPS, C_{in}, N_{TCN} and s_{win} were fixed at 30, 8, 1 and 5 respectively for training, resulting in a receptive time interval of $t_p = 400\,\mathrm{ms}$. For validation (and for the deployed model), s_{win} was set to 3, yielding $t_p = 666.7\,\mathrm{ms}$.

B. Power Consumption

The basis of our evaluations is a full-featured SoC based on the open-source PULP architecture featuring a DVS peripheral,[2] frame buffer and the modified CUTIE accelerator.The system was synthesized and implemented in the GlobalFoundries 22 nm FDX process. As the full system contains many components unrelated to the problem of frame-based DVS data classification and the proposed infrastructure can be implemented in any edge processing system, we isolate the power measurements of the DVS interface, frame buffer, and CUTIE. Power was simulated using Synopsys PrimeTime using the synthesized netlist, with switching activity extracted for the entire processing pipeline of frame acquisition, data transfer, and inference. CUTIE was clocked at 17.6 MHz and the rest of the system, including the DVS interface and frame buffer, was clocked at 50 MHz. Due to library availability, a power analysis was run for the typical corner with all supply voltages at 0.8 V, and we scale our power numbers to supply voltages of 0.65 V. We report two numbers for classification energy: The first is the isolated inference energy, comprised of the energy required for receiving the 3 DVS frames, transferring the content of the frame buffer to CUTIE, and running inference. To obtain a real-life energy requirement estimate, we also calculate a normalized total energy per classification, which includes the idle power consumed between frame acquisitions and inferences at our implementation's rate of 10 classifications per second. Where idle power is reported, we calculate this quantity for the works we compare to. For our system, this normalized classification energy is dominated by the leakage power drawn in the idle state - this is exacerbated by the very short time required to run inference on CUTIE when compared to the idle time between inferences. This leakage energy could be reduced by power-gating CUTIE's internal activation memories and combinational cells (weight memories cannot be power-gated as the model's weights must be persistent). We model the effect of this approach on the normalized classication energy using a simplified model by subtracting the leakage power of the activation memories and combinational logic from the idle power and include the modified figure in Table II in parentheses.

C. Discussion

The model we chose to map to CUTIE achieves 94.5% classification accuracy. Among models targeted at embedded deployment, our approach matches or outperforms the state of the art. However, it is outperformed by approaches that

[2]Designed for the IniVation DVS132S camera

do not place any restrictions on model complexity or size, most notably [23], which achieves a near-perfect classification accuracy of 99.6%. A comparison between different state-of-the-art gesture recognition systems is shown in Table II. The proposed system outperforms even the most efficient implementations on SNN accelerators in terms of energy per inference by a factor of $647\times$ at equivalent or higher statistical accuracy. When considering total classification energy normalized to 10 inferences per second, the gap is reduced by the relatively high proportion of leakage power consumed by CUTIE's internal memories, but remains substantial at a factor of $39\times$. By power-gating CUTIE's activation memories as well as CUTIE's combinational cells, the normalized classification energy can be reduced by 48%, improving our approach's advantage over SNN-based works to a factor of $75\times$. Our results show that frame-based processing of DVS data using a ternarized hybrid CNN/TCN network running on a highly efficient TNN outperforms the state of the art of SNN-based approaches by several orders of magnitude in terms of energy efficiency, while maintaining equivalent or better accuracy.

V. Conclusion

In this work, we have presented a highly accurate TCN-based neural network running on an embedded sensor node platform, which together are capable of acquiring, processing, and classifying event-based camera data from a DVS sensor. We further demonstrated a novel embedded frame-based approach for classifying DVS data using a TCN network. The resulting validation accuracy of 94.5% is on par with state-of-the-art embedded solutions. Finally, we presented accurate gate-level power simulation data, showing an inference energy consumption of $1.7\,\mu\text{J}$, which is $647\times$ lower than the previous state of the art in DVS-based processing. When considering the normalized classification energy including idle power between inferences, leakage power during idle phases can be minimized by power-gating the non-state holding parts of the design, resulting in a normalized energy consumption of $48\,\mu\text{J}$ per classification at 10 classifications per second, an improvement of $75\times$ over the previous, SNN-based state of the art. The results show the effectiveness of both the frame-based and TCN-based approaches to DVS processing.

References

[1] L. Lai, N. Suda, and V. Chandra, "Cmsis-nn: Efficient neural network kernels for arm cortex-m cpus," 2018.

[2] A. Garofalo, M. Rusci, F. Conti, D. Rossi, and L. Benini, "PULP-NN: Accelerating quantized neural networks on parallel ultra-low-power RISC-V processors," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 378, no. 2164, 2020.

[3] M. Scherer, G. Rutishauser, L. Cavigelli, and L. Benini, "CUTIE: Beyond PetaOp/s/W Ternary DNN Inference Acceleration with Better-than-Binary Energy Efficiency," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–14, 2021.

[4] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128×128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[5] M. A. Mahovald and C. Mead, "The Silicon Retina," *Scientific American*, vol. 264, pp. 76–83, 1991.

[6] A. Amir *et al.*, "A Low Power, Fully Event-Based Gesture Recognition System," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 27, no. 36. IEEE, jul 2017, pp. 7388–7397.

[7] J. Wu and R. Jafari, "Orientation independent activity/gesture recognition using wearable motion sensors," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1427–1437, 2019.

[8] R. Yao, G. Lin, Q. Shi, and D. C. Ranasinghe, "Efficient dense labelling of human activity sequences from wearables using fully convolutional networks," *Pattern Recognition*, vol. 78, pp. 252–266, 2018.

[9] F.-T. Liu, Y.-T. Wang, and H.-P. Ma, "Gesture recognition with wearable 9-axis sensors," in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1–6.

[10] B. Bent, B. A. Goldstein, W. A. Kibbe, and J. P. Dunn, "Investigating sources of inaccuracy in wearable optical heart rate sensors," *npj Digital Medicine*, vol. 3, no. 1, p. 18, Feb 2020.

[11] J. D. Stone *et al.*, "Assessing the accuracy of popular commercial technologies that measure resting heart rate and heart rate variability," *Frontiers in Sports and Active Living*, vol. 3, p. 37, 2021.

[12] R. Massa, A. Marchisio, M. Martina, and M. Shafique, "An Efficient Spiking Neural Network for Recognizing Gestures with a DVS Camera on the Loihi Neuromorphic Processor," *Proceedings of the International Joint Conference on Neural Networks*, no. July, pp. 1–9, 2020.

[13] S. B. Shrestha and G. Orchard, "SLAYER: Spike layer error reassignment in time," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 1419–1428.

[14] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, and Y. Tian, "Incorporating learnable membrane time constant to enhance learning of spiking neural networks," 2021.

[15] Y. Xing, G. Caterina, and J. Soraghan, "A new spiking convolutional recurrent neural network (scrnn) with applications to event-based hand gesture recognition," *Frontiers in Neuroscience*, vol. 14, p. 590164, 11 2020.

[16] L. Cordone, B. Miramond, and S. Ferrante, "Learning from event cameras with sparse spiking convolutional neural networks," 2021.

[17] M. Scherer, M. Magno, J. Erb, P. Mayer, M. Eggimann, and L. Benini, "TinyRadarNN: Combining Spatial and Temporal Convolutional Neural Networks for Embedded Gesture Recognition with Short Range Radars," *IEEE Internet of Things Journal*, pp. 1–11, 2021.

[18] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental Network Quantization: Towards Lossless CNNs with Low-Precision Weights," *Proceedings - 2017 International Conference on Learning Representations*, pp. 1–14, feb 2017.

[19] Y. Bengio, N. Léonard, and A. Courville, "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation," pp. 1–12, 2013.

[20] M. Gautschi *et al.*, "Near-threshold risc-v core with dsp extensions for scalable iot endpoint devices," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 10, pp. 2700–2713, 2017.

[21] A. Burrello, M. Scherer, M. Zanghieri, F. Conti, and L. Benini, "A microcontroller is all you need: Enabling transformer execution on low-power iot endnodes," in *2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS)*, 2021, pp. 1–6.

[22] E. Ceolini *et al.*, "Hand-gesture recognition based on emg and event-based camera sensor fusion: A benchmark in neuromorphic computing," *Frontiers in Neuroscience*, vol. 14, p. 637, 2020.

[23] S. U. Innocenti, F. Becattini, F. Pernici, and A. Del Bimbo, "Temporal binary representation for event-based action recognition," *Proceedings - International Conference on Pattern Recognition*, pp. 10 426–10 432, 2020.

[24] J. Yang *et al.*, "Modeling point clouds with self-attention and gumbel subset sampling," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 3318–3327, 2019.

[25] P. Blouw, X. Choo, E. Hunsberger, and C. Eliasmith, "Benchmarking keyword spotting efficiency on neuromorphic hardware," *ACM International Conference Proceeding Series*, 2019.

[26] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," pp. 1–15, 2014.