Fast Retinomorphic Event-Driven Representations for Video Gameplay and Action Recognition

Huaijin Chen, Wanjia Liu, Rishab Goel, Rhonald C. Lua, Siddharth Mittal, Yuzhong Huang, Ashok Veeraraghavan, and Ankit B. Patel

***Abstract*—Good temporal representations are crucial for video understanding, and the state-of-the-art video recognition framework is based on two-stream networks. In such framework, besides the regular ConvNets responsible for RGB frame inputs, a second network is introduced to handle the temporal representation, usually the optical flow (OF). However, OF or other task-oriented flow is computationally costly, and is thus typically pre-computed. Critically, this prevents the two-stream approach from being applied to reinforcement learning (RL) applications such as video game playing, where the next state depends on current state and action choices. Inspired by the early vision systems of mammals and insects, we propose a fast event-driven representation (EDR) that models several major properties of early retinal circuits: (1) logarithmic input response, (2) multi-timescale temporal smoothing to filter noise, and (3) bipolar (ON/OFF) pathways for primitive event detection. Trading off the directional information for fast speed (>9000 fps), EDR enables fast real-time inference/learning in video applications that require interaction between an agent and the world such as game-playing, virtual robotics, and domain adaptation. In this vein, we use EDR to demonstrate performance improvements over state-of-the-art reinforcement learning algorithms for Atari games, something that has not been possible with pre-computed OF. Moreover, with UCF-101 video action recognition experiments, we show that EDR performs near state-of-the-art in accuracy while achieving a 1,500x speedup in input representation processing, as compared to optical flow.**

***Index Terms*—Smart cameras, retina, real-time systems, streaming media, cells (biology), reinforcement learning, video signal processing, video.**

Manuscript received December 24, 2018; revised March 1, 2019 and June 12, 2019; accepted June 14, 2019. Date of publication October 21, 2019; date of current version January 13, 2020. The work of H. Chen was supported in part by the NSF EAGER under Grant 1502875, in part by the NSF CAREER under Grant 1652633, and in part by the Texas Instruments Distinguished Graduate Student Fellowship. The work of W. Liu, R. C. Lua, and A. B. Patel was supported in part by IARPA via DoI/IBC under Contract D16PC00003 and in part by NSF NeuroNex under Grant DBI-1707400. The work of A. Veeraraghavan was supported in part by the NSF EAGER under Grant 1502875 and in part by the NSF CAREER under Grant 1652633. The associate editor coordinating the review of this manuscript and approving it for publication was O. Cossairt. *(Huaijin Chen and Wanjia Liu contributed equally to this work.) (Corresponding author: Ankit B. Patel.)*

H. Chen was with the Department of Electrical and Computer Engineering, Rice University, Houston, TX 77005 USA. He is now with SenseBrain Technology LLC., San Jose, CA 95131, USA (e-mail: huaijin.chen@rice.edu).

W. Liu was with the Department of Computer Science, Rice University, Houston, TX 77005, USA. He is now with Google Inc., Mountain View, CA 94043, USA (e-mail: superliuwanjia@gmail.com).

R. Goel was with the Indian Institute of Technology Delhi, Hauz Khas, New Delhi-110 016, India. He is now with Borealis AI, Montreal, QC H2S 3H1, Canada (e-mail: rgoel0112@gmail.com).

R. C. Lua and A. B. Patel are with the Department of Neuroscience, Baylor College of Medicine, Houston, TX 77030 USA (e-mail: rhonald.lua@gmail.com; abp4@rice.edu).

S. Mittal was with the Indian Institute of Technology Kanpur, Uttar Pradesh 208016, India. He is now with Quadeye, Gurgaon 122009, India (e-mail: smittal6@gmail.com).

Y. Huang was with the Olin College of Engineering, Needham, MA 02492, USA. He is now with Kensho Technologies, Cambridge, MA 02138 USA (e-mail: huangyuzhongapp@gmail.com).

A. Veeraraghavan is with the Department of Electrical and Computer Engineering, Rice University, Houston, TX 77005 USA (e-mail: vashok@rice.edu).

This article has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the authors. This includes a video, which shows the training curves of different games. This material is 41.2 MB in size.

Digital Object Identifier 10.1109/TCI.2019.2948755

I. Introduction

DEEP learning and related techniques have resulted in substantial advances in image understanding over the last decade [27], [35], [37], resulting in a new-found sense of optimism regarding possibilities in many application areas, including autonomous robots and self-driving cars. The current state of practice in video understanding tasks is either to (a) process each frame in the video sequence independently in a frame-by-frame fashion, or (b) pre-compute the temporal representations, as seen in the two-stream architecture for video recognition [63]. The latter is efficient but necessarily precludes applications like reinforcement learning (RL), where the next state depends on the current state and the actions taken by the agent. Therefore, if the two-stream approach were to be used in RL, the temporal representation will have to be computed on-the-fly. Taking the Atari Centipede gameplay as an example, reinforcement learning usually converges to the optimal strategy after 70 millions steps of trials and errors. At each step, assuming 4 frames of video frames are considered to decide an action, then under the conventional OF-based two-stream framework, there will be $70\text{ M} \times (4-1) = 210\text{ M}$ frames of OF field to be calculated across the entire training process on-the-fly. Even if we use a relatively fast OF algorithm (i.e. TV-L1 [53] which runs at 30 fps for 320×240 inputs), it will still take more than $210\text{ M}/30 = 7\text{ M}$ seconds $= 81$ days to just compute the OF. As such, there are still significant gains to be captured in speed, accuracy, bandwidth and energy, by explicitly leveraging the temporal redundancy structure in video.

Why haven't we seen such progress? We believe there are two principal reasons for the slow pace of progress in video understanding using deep networks. First, the massive size and data rates needed in video, make even the simplest feed-forward processing computationally challenging to accomplish, especially in settings like game-playing where real-time processing

is needed. A natural solution to this problem is to use event-driven cameras, such as DVS and DAVIS [8], [39], or input representations, which bring orders of magnitude reduction in sensor power consumption and data bandwidth, which are both especially helpful on energy-constrained platforms. However, existing efforts in this vein focus mostly on optimizing network architecture [29]. Second, a good input representation for temporal dynamics – one that accounts for and exploits the inherent redundancy – is crucial to good performance in video understanding. For example, optical flow has significant added value in video understanding tasks, but it is expensive to compute, making it impossible for many tasks that require real-time video input, such as reinforcement learning (Fig. 2(a) and 2(b)). As a result, despite recent breakthroughs in static image understanding [27], [35], much more research is needed in developing new fast, temporally-aware representations for video understanding.

Contributions: In this paper, we propose a retinomorphic learnable Event Driven Representation (EDR) for video. Our EDR has tunable parameters, enabling us to capture meaningful task-relevant events from the data. In the proposed EDR, we captured some of the simple properties of the biological retina and the visual cortex. Our implementation is preliminary and we only conduct first-order studies of EDR, and yet the performance gains (near state of art classification performance with potentially order of magnitude reductions in data throughput and power requirements) strongly suggest that retinomorphic event-driven representations are worthy of further study. The main contributions of our work are:

1) We propose a retinomorphic event-driven representation (EDR) for video, instantiated as an recurrent neural network (RNN) layer, that realizes three important functions of the biological retina: logarithmic transformation, ON/OFF pathways for event detection and the integration of multiple timescales (Section III).
2) We show systematic improvements in the performance (accuracy, response time, latency, throughput) and learning speed for our proposed learned EDRs as compared to Frame Driven Representations (FDRs) on the task of Atari game playing via reinforcement learning and UCF-101 action recognition (Section IV-A, IV-B).
3) On a smaller KTH datasets, we analyze the EDR's energy efficiency in terms of network activation and evaluate performance of EDR on different network designs on action recognition tasks (Section IV-C).
4) We are able to apply the EDR model to an event-driven camera hardware, showcasing a joint software/hardware event-driven motion vision system that performs high-level visual understanding tasks with order of magnitude reductions in data throughput and power per voxel compared to traditional-image-sensor based implementations (Section IV-D). With the high efficiency of the EDR, we are also able to implement it on smartphones (Fig. 2(c)).

Limitation: EDR is derived and optimized as a representation best suited to analyse and characterize high-speed events and actions in an efficient manner. Ideally, this work would lead to practical implementations of EDR cameras that realize multiple time-scales and soft-thresholding, both critical features for optimizing performance. Unfortunately, currently available event sensor hardware do not allow for either feature extraction at multiple time-scales or soft-thresholding at the sensor level. Given this limitation of current sensors, in most of our current experiments, (except for the hardware experiments, where the DVS camera was used to directly capture the video clips displayed on the monitor and had its stream fed into the network pre-trained using EDR data), EDR was mostly used as a processing step after the RGB frame is captured. Moreover, even though the EDR has learnable parameters and is differentiable, which opens up the potential for gradient descent and back-propagation, we have not fully explored that for end-to-end parameter learning due to instability in the gradients. Our preliminary experimentation with end-to-end learning of the parameters found the gradients to be unstable, resulting in vanishing/exploding gradient problems. Since EDR only has a few parameters, we did 2D grid search as our learning algorithm for the EDR parameters. The experimental results we show used the best two key EDR parameters through grid search. We are releasing the code with option to use gradient-based learning for any interested user to help us study this problem further.

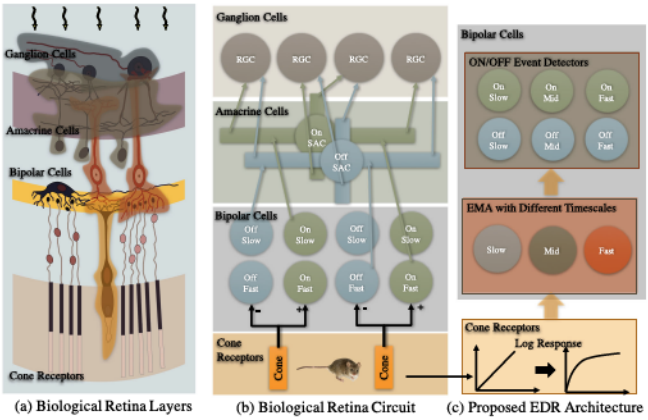
II. Related Work

A. Biological Retinas

In order to survive in a hostile environments, animals have evolved visual systems with fast reaction times, and the retina is the first stage of processing. The retina has many interesting properties both in its structure and in its variety of responses. However, there is reasonable consensus [12], [56] that at least these five major features are essential for motion-based vision (see Fig. 1(a)): (1) Photoreceptors have a logarithmic response to luminance input; (2) event detection happens in parallel ON/OFF pathways in multiple cell types (e.g. bipolar and ganglion cells) where ON/OFF events are generated based on spatiotemporal changes in luminance [22], [50], [67]; (3) Fast/slow pathways for distinguishing/integrating events across different time scales [4]. (4) Primitive motion detection for the cardinal directions (e.g. LPTC or DSGC cell types [45], [73]); and (5) 4-channel color vision cone (RGB) and rod (grayscale) cells [9]. Several recent studies propose computational neuroscience models for the retina and learning process [26], [43], but those models were not designed for practical deep learning tasks. This is by no means an exhaustive list; indeed the structure and function of the retina are active areas of ongoing research. In this paper, we focus on the event-driven nature of the retina, instantiating and exploring properties (1), (2) and (3) above, and leaving others – listed or not – for future work.

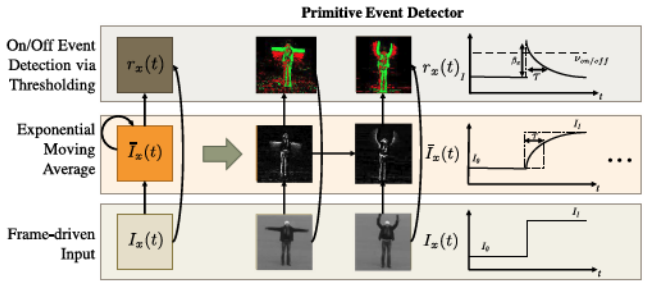
B. Retinomorphic Cameras

Research on neuromorphic image sensor and systems has developed over several decades [2], [10], [21], resulting in some impressive research prototypes [11], [55], including a sensor that reproduces all five layers of the retina [75]. Recently, retinomorphic event driven image sensors have been commercialized,



(a) Biological Retina Layers (b) Biological Retina Circuit (c) Proposed EDR Architecture

(a) **Neural circuit of biological early motion vision system vs. EDR:** We focus on three key features of the retina: (1) Logarithmic response, (2) ON/OFF event pathway, and (3) integration of multiple timescale events. The left-most column is the biological neural circuit, the middle column (adapted from [12]) is the abstracted circuit, and the right most column is our EDR implementation of the abstracted circuit.



(b) **EDR as a Retinomorphic Primitive Event Detector:** From the frame-driven inputs, we first apply an exponential moving average filter to smoothen the event estimation. We then calculate the relative changes/return in the input stream. Finally we threshold the relative changes/return and generate binary On/Off events.

Fig. 1. Overview of the proposed EDR.

such as the Dynamic Vision Sensor (DVS) [39], Dynamic and Active-pixel Vision Sensor (DAVIS) [8], [13] and Asynchronous Time-based Image Sensor [55]. Unlike conventional cameras that output grayscale or color intensities for each pixel and each frame, event cameras detect and report only significant changes in intensity at each pixel. As a result, DVS is capable of dramatically higher throughput and lower latency as compared to a conventional camera. We have seen recent applications of event-driven cameras in many areas of computer vision and robotics, such as structured light active 3D imaging [44], multi-view stereo [57], high-speed tracking [49], panoramic tracking [59], face detection and intensity reconstruction [7], visual odometry [16], [58], motion flow [6] and real-time 3D scene reconstruction [34].

Despite the amount of work on using event cameras for low-level vision tasks, there still remains a gap between event cameras (e.g. artificial retinas) and high-level semantic understanding (e.g. artificial cortex). An early exploration was carried out by [54], in which the authors attempted to map DVS camera output events back to image frames, which in turn were fed to a 5-layer ConvNet that infers motion cardinal direction. In recent work of [64], the authors proposed new representation based on the event camera inputs, and use the proposed representation to achieve better object classification.

C. Neural Network for Temporal Sequences

Recent works in video recognition using both deep learning [3], [63], [74] and traditional hand-designed features [20], [60], suggest that efficiently modeling/integrating information across time is very important for performance. On this note, there are work using temporal features extracted from video codec [31], [71], instead of computing decoded frame for efficiency. Recurrent Neural Networks (RNNs) with Long- and Short-term Memory (LSTM) [25], [28] – employed successfully for speech recognition and natural language processing – might be a promising model for such problems. Marrying ConvNets to LSTM resulted in the long-term recurrent ConvNets (LRCN), which has shown some promising but not outstanding results in video recognition [20]. LRCNs begin to integrate information across time but do so at later stages in the visual processing architecture. This is in stark contrast to the biological visual system, wherein temporal feature detection and event generation happen at the *earliest* stages yielding an architecture that is *event-driven from end to end*. Recent sigma-delta quantized networks (SDQN) [51] also investigate sparse temporal representations. However, the main goal of SDQNs is reducing computation (FLOPs) by passing quantized activation differences through time. EDR, on the other hand, is an event-driven input representation, looking for meaningful task-specific changes in the input.

D. Two-Stream Architectures for Video Inference

Recently, deep architectures have been used to obtain better performance on standard video inference benchmarks, such as UCF-101 [66] and HMBD-51 [36]. The state-of-the-art methods for those benchmarks are based on the two-stream ConvNets framework proposed by Simonyan, *et al.* [63]. Such framework uses two separate ConvNets to handle RGB frames and optical flow derived from the RGB frames, respectively. The final prediction is based on the consensus of both networks. The success of the two-stream framework demonstrated the importance of a good temporal representation, however, the temporal representation itself is much less discussed in many two-stream-based frameworks. Even the state-of-the-art two-stream frameworks, such as the Temporal Segmentation Networks (TSN) [70], Long-Term ConvNets(LTC) [68] and Inflated 3D ConvNets (I3D) [17], simply use off-the-shelf algorithms to precompute a costly but accurate optical flow (OF) field.

In short, the difference between EDR and OF is that EDR's On/Off events are based solely on a single pixel's intensity changes. Optical flow, on the other hand, infers the velocity vector (magnitude and direction) of a moving pixel based on a neighborhood of pixels. This velocity may carry extra information relevant to the task. OF is costly to compute, but may not be necessary, as suggested by recent work [23], [72], [76], which proposed that a task-driven flow can be jointly learned end-to-end as a fast alternative to conventional OF. Nevertheless, those task-driven flows can only be computed at frame

rates of up to 12–120 fps, which is still prohibitively slow for computing on-the-fly at training time, especially for applications like reinforcement learning of video gameplay.

E. Reinforcement Learning-Based Video Game Play

Deep networks have been very successful in solving reinforcement learning video game problems and demonstrated great capability in environment transition and agent behavior modeling. To name a few, Minh *et al.* [47] first introduced Deep Q-learning (DQN) by approximating the Q function using a neural network and enabled Q-learning to achieve near or over human performance on Atari video games. Minh *et al.* [46] approximates the advantage and policy vectors using a single network and have multiple agents asynchronously explore the environment, which largely improved performance over DQN. Built on top of A3C, Parallel Advantage-Actor-Critic (PAAC) [18] implements A3C on CPU and shortens A3C training time to one day. However, recent RL improvements are mostly on RL strategy, problem formulation and network structure. To our knowledge, none of the existing work utilize the two-stream framework for RL due to the slow OF computation. We believe we are the first to investigate the event-driven temporal input representation for RL, whereas all the existing RL approaches largely rely on RGB or grayscale frame-based input.

III. Event-Driven Representation

We propose a simple event detector: a *thresholded exponential moving average (tEMA)* of (relative) changes in the input. Despite its simplicity, this simple detector is widely used as a temporal event detector and descriptor in many areas including high-frequency finance [41] and the mammalian retina [52], [65]. In both cases, fast response times are critical. We now describe the structure of the tEMA which is composed of three components: (1) an exponential moving average filter, (2) a relative change computation and (3) a thresholding operation. We show the conceptual diagram of such procedure in Fig. 1(b).

A. Exponential Moving Average (EMA)

For each pixel location x, the pixel intensity $I_x(t)$ is noisy and variable. In order to smoothen the estimate, we apply an exponential moving average (EMA) filter to the sequence $I_x(t)$ to get the filtered sequence $\bar{I}_x(t;\tau_{1/2}) \equiv EMA(I_x(t);\tau_{1/2})$ where the half-life parameter $\tau_{1/2} \in \mathbb{R}_+$ controls the memory of the filter. Intuitively, a new data point affects the EMA for $\tau_{1/2}$ timesteps before decaying into half of starting amplitude. Effectively, an EMA weighs the recent past *exponentially more* than the distant past.

One computational advantage of the EMA is that it can be computed recursively as

$$\begin{aligned}\bar{I}_x(t) &= \bar{I}_x(t-1) + \alpha(I_x(t) - \bar{I}_x(t-1)) \\ &= (1-\alpha)\bar{I}_x(t-1) + \alpha I_x(t),\end{aligned} \tag{1}$$

where $\bar{I}_x(t)$ is the EMA of input $I_x(t)$ at time t and pixel location x and we have suppressed the dependence on the memory parameter $\tau_{1/2}$. In fact, the dependence on $\tau_{1/2}$ will be indirectly specified through another tunable parameter $\alpha \equiv 1 - 2^{-1/\tau_{1/2}} \in [0,1]$. A larger α (smaller $\tau_{1/2}$) places more weight on the most recent inputs and thus forgets earlier inputs $I_x(t)$ more quickly. Note that this recursive update for the EMA is linear in $I_x(t), \bar{I}_x(t)$ and so can be implemented as a *linear* recurrent neural network (RNN).

B. Relative Changes/Returns

We next need to define a way to compute changes in the input stream. One simple approach is to compute the relative change of the input with respect to past inputs i.e. a return. Given a smoothed estimate $\bar{I}_x(t)$ of the input stream, this return stream is defined as

$$R_x(t) \equiv \left(\frac{I_x(t)}{\bar{I}_x(t)}\right)^{\beta_x} \quad \text{or} \quad r_x(t) \equiv \beta_x \ln\left(\frac{I_x(t)}{\bar{I}_x(t)}\right).$$

Note that returns are dimensionless measures of changes in the input stream just like e.g. stock price returns in finance. Intuitively, when the input stream is constant $I_x(t) = I_0$ the return stream $r_x(t) \to 1$ since the EMA $\bar{I}_x(t) \to I_0$ in $O(\tau_{1/2})$ timesteps. If the input stream is an impulse, the return stream jumps quickly to its peak response and then decays with half-life $\tau_{1/2}$ back to 1 (see Fig. 1(b)). The amplitude of the peak response is controlled by $\beta_x \in \mathbb{R}$. Intuitively, increasing/decreasing β_x makes the event detector more/less sensitive to changes in the input (e.g. pixel intensity changes in the scene).

C. Event Detection via Thresholding

Given a sequence of real-valued returns, we now define a simple event detector via a thresholding operation that determines whether a change is "significant," analogous to a noise floor in a signal detection problem.

Our *soft thresholding* operation employs a *bipolar* structure, inspired by the retina, that detects two kinds of input events: ON and OFF. The output event streams are mathematically defined as

$$E_{x,ON}(t) \equiv [r_x(t) - (1+\nu_{ON})]_+ \in \mathbb{R}_+ \tag{2}$$

$$E_{x,OFF}(t) \equiv [r_x(t) - (1-\nu_{OFF})]_- \in \mathbb{R}_+, \tag{3}$$

where $[b]$ is defined as 1 if statement b is true and 0 if it is false, and $[r]_+ \equiv ReLU(r) \in \mathbb{R}_+, [r]_- \equiv ReLU(-r) \in \mathbb{R}_+$ are the positive and negative parts, respectively, of the real number $r \in \mathbb{R}$. The threshold parameters ν_{ON}, ν_{OFF} determine how large a relative change in the inputs is required for an ON/OFF event to be generated. For example, if $\nu_{ON} = +5\%$ then a 5% increase in the input $I_x(t)$ relative to its EMA $\bar{I}_x(t)$ is needed in order for an ON event to be generated. A similar relationship holds for ν_{OFF}. This mimics the retinal firing rates after the ON/OFF event is detected.

Note the similarities and differences between the bipolar events and a standard weighted ReLU layer in a recurrent ConvNet. Both can be written as recurrent weight layers with a biased ReLU. Despite this similarity, there are a few key differences inspired directly from the retina. First, our EDR possesses bipolar (ON/OFF) semantics, i.e. there are two parallel channels whose purpose is to detect significant changes in the two possible directions. Second, the weight and bias $\beta_x, \nu_{x,s}$

are *interpretable* as sensitivity parameters and detection thresholds, respectively. Third, The input into the bipolar ReLU is a hand-designed feature – the log-returns stream of the inputs – that is hand-designed to be a trend detector. The log – inspired by retinal photoreceptor responses – enables the processing of inputs with large dynamic range. (This primitive event detector is also commonly used in high-frequency finance.)

D. Multiple Timescale Events

Biological retinas have synapses/connections that combine events from fast and slow pathways to form an event stream that is sensitive to different time scales. We mimic this by providing log-return from short, medium, and long time scales $\{\alpha_S, \alpha_M, \alpha_L\}$ EDR. Note that the different timescales have different (learnable) weights $\{\beta_S, \beta_M, \beta_L\}$ associated.

$$r_{\boldsymbol{x}}(t) \equiv \{r_x(t; \alpha_j)\}, j \in \{S, M, L\} \tag{4}$$

E. Temporal Dynamics Representation Comparison

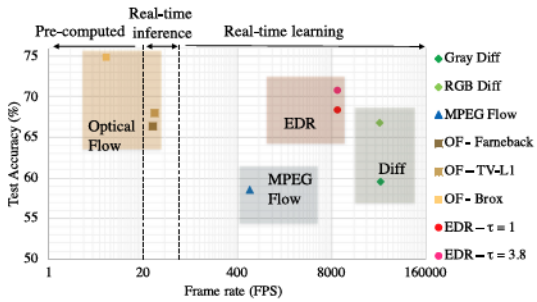
There are several input representations that aim to capture the temporal dynamics, namely optical flow [15], [24], DVS [39] camera hardware, and simple inter-frame difference. We compare the EDR with them, and summarize the differences in Table 2(b). The major features of EDR are 1) a larger temporal receptive field (RF) for capturing events of different timescales, 2) temporal smoothing to handle the noise and 3) soft thresholding that allows distinguishing events while keeping some of the texture of moving objects. We also plot out the response of different input representations across time for a random pixel location in the first "Archery" clip of the UCF-101 datasets in Fig. 3(b). The visualization of corresponding frames can be found in Fig. 3(a).

IV. Experiments and Results

To evaluate and compare the performance of our proposed retinally-inspired input representations in a variety of application scenarios, we carry out experiments on reinforcement learning of Atari game play (Section IV-A), as well as action recognition benchmarking of the UCF-101 (Section IV-B). To understand the proposed EDR's performance in detail, we also performed several ablation studies of EDR on a smaller action recognition dataset, KTH (Section IV-C). In addition, using the model trained with KTH datasets, we show hardware experiments with transfer learning in Section IV-D.

A. Atari Game Reinforcement Learning (RL)

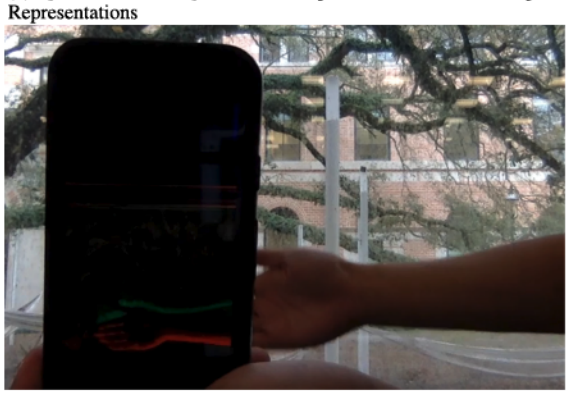
1) RL Environment and Algorithm: A natural choice for testing the proposed EDR is RL-based game play tasks, because games involve rich visual action interactions, and visual-event-action is a common reflex found in animals that make use of event-driven representations. We use a subset of six (out of 55) Atari 2600 console games in the Atari Learning Environment (ALE) [14], namely Pong, Breakout, Pacman, Centipede, Qbertand and Seaquest, ranked roughly based on game difficulty from easy to hard. Our baseline RL setup is Parallel Advantage-Actor-Critic (PAAC) [18]. PAAC is a synchronous implementation of the state-of-the art algorithm A3C [46] in GPU. In the baseline setup, game playing frames are fed into the controller network directly. For the EDR comparison experiments, the implementation details are described below.



(a) **Speed Accuracy Trade-off for Temporal Representations:** We compare the proposed EDR with several common input representations for temporal dynamics. The major features of EDR are 1) larger temporal receptive field (RF) for capturing events of different timescales (where the temporal RF is the amount of information from past frames used to compute the EDR signal at a pixel), 2) temporal smoothing to handle the noise, 3) soft thresholding that allows distinguishing events while keeping some of the texture of moving objects, and 4) Potentially learnable parameters.

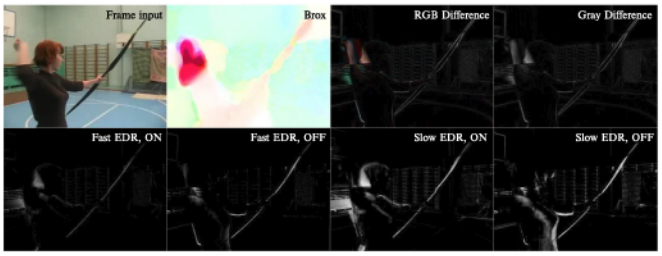
	Optical Flow	**Inter-frame Differences**	**Event-driven Camera (DVS128)**	**EDR**
Feature	Patch Displacement	Pixel intensity change	Pixel intensity change	Pixel intensity change
Computation	High	Low	Low	Low
Run-time	Pre-computed	Real-time	Real-time	Real-time
Temporal RF	2 frames	2 frames	Multiple Timesteps	τ frames
Temp. Smooth	No	No	No	Yes
Spatial RF	>10 pix	1 pix	1 pix	1 pix
Spatial Smooth	Yes	No	No	No
Thresholding	No	Hard	Hard	Soft
ON/OFF Channels	No	No	Yes	Yes
Learnable Parameters	No	No	No	Yes

(b) Quantitative and Qualitative Comparison of Different Temporal Representations

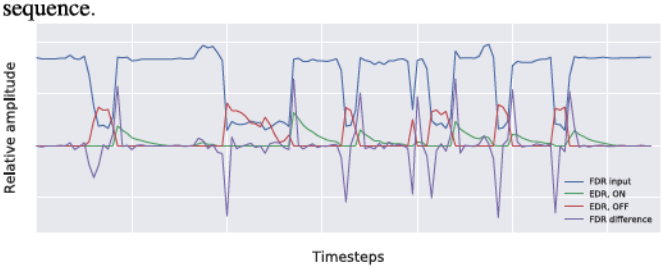


(c) Our proposed EDR is efficient enough to run real-time on regular smartphones. The EDR events are shown in red (OFF) and green (ON). EDR detects the hand motion with texture-rich background filtered out.

Fig. 2. EDR's major features compared to other temporal representations.



(a) Visualization of different input modalities of a UCF-101 frame sequence.



(b) Visualization of different input modalities for a random pixel location of a UCF-101 frame sequence.

Fig. 3. Temporal input representations comparisons on UCF-101 frames: (a) We visualize a sample frame from the UCF101 archery class. First row from left to right: RGB, Brox flow, RGBDiff, GrayDiff. Second row from left to right: Fast decay EDR ON, Fast decay EDR OFF, Slow decay EDR ON, Slow decay EDR OFF. Notice the stationary lattice pattern in the background is less emphasized in fast decay EDR than Difference, while being responsive to the archer. (b) Output values of different representations across time for a randomly chosen pixel location in the first archery clip in the UCF-101 datasets are shown.

2) Network Architecture and Training Procedure: Controller network architecture in deep reinforcement learning is important for both extracting features and mapping them into value and policies, and eventually affects agent performance. Here we use the standard ConvNets described in [48] as the baseline network, which we note as $arch_{base}$. To incorporate temporal features across time, an RNN or LSTM is commonly needed. Recurrent convolutional networks (RCN) [20] structure was proposed to perform this task by connecting the LSTM layers at the end of a ConvNet. We made some modifications to the original RCN to better track spacial features. We name it FT-RCN (explained in Section IV-C). Here, as a comparison, we add this FT-RCN structure to a baseline network, which we refer to as $arch_{FT-RCN}$. We found that our FT-RCN gave better results than the baseline. The results are shown in Table I. For the RL training policy, we experimented with A3C [46] and PAAC [18]. We found that we are able to run more experiments using PAAC because multiple instances of PAAC training can be efficiently parallelized on multiple GPUs. In contrast, in A3C, training agent instances occupy multiple threads with full load, and fill up computational resources quicker than PAAC. The reported experimental results are on PAAC. In our PAAC experiment setup, we use 32 agents and traverse 5 local steps before aggregating observations into a batch. As a result, each agent in A3C has batch size of 5 and PAAC has batch size of 80. A3C learning rate starts from 0.4226 and decreases exponentially to 0 in 80 million global steps, PAAC learning rate starts from 0.0224 and linear decay to 0 at 80 million global steps. Entropy scaling constant is 0.01 for A3C and 0.02 for PAAC. Both A3C and PAAC training clip gradient based on L2-norm: A3C gradient is clipped at 40.0 and PAAC is clipped at 3.0. All experiments use RMSprop optimizer and discount factor is set to 0.99. Pong training stops at 30 million global steps and training for remaining games stop at 80 million global steps.

3) Input Data Dimension: In the baseline setup, game playing frames are fed into the controller network directly. We concatenate EDR with ON and OFF channels and dimension $W \times H \times 2$ to the original FDR input with dimension $W \times H \times 3$ in the channel dimension. Thus, the new dimension of the input is $W \times H \times 5$ for all our EDR comparison experiments. We refer to this as FDR+EDR for all the Atari based experiments, which is similar to the two-stream framework approach used in video recognition tasks. The difference is that the OF in the conventional two-stream approach is usually pre-computed due to its complexity. Here, however, EDR is computed on-the-fly since it's an extremely efficient temporal input representation compared to OF. The intuition behind using two-stream approach in RL is similar to the motivation in the action recognition cases: we would like not only to have a good spatial representation for the context of a given time, but also a temporal representation for the context across time.

Note here we concatenate the $W \times H \times 2$ EDR input with the $W \times H \times 3$ FDR input at any given timestep, not across time steps. The 2 and 3 in each case are actually the numbers of channels at a given time step, not the number of frames across time to concatenate. EDR has two channels, ON/OFF. FDR has three channels, RGB. After the concatenation, FDR+EDR will have a $W \times H \times 5$ input at each time step, while the baseline FDR only has a $W \times H \times 3$ input at each time step.

4) RL Experiment Results: Experimental results of 6 different games are shown in Table I, where the min, max and mean total reward score of over 30 different test runs are reported. We compare the total reward scores of FDR + EDR and FDR + DIFF inputs with those of conventional FDR inputs. It is clear that a second input stream greatly improves the performance.

Meanwhile, an ablation study of $arch_{base}$ vs. $arch_{FT-RCN}$ is also included. The proposed $arch_{FT-RCN}$ has better performance compared to the baseline $arch_{base}$ in most cases in terms of average game scores regardless of the type of input given — FDR, FDR + EDR or FDR + DIFF data. For comparison of FDR + DIFF vs. FDR + EDR, overall, they perform similarly, better or worse, to each other, but in several cases, the best case scenario of FDR + EDR outperforms FDR + DIFF by a large margin. Specifically, for $arch_{base}$, FDR + DIFF out-performs FDR + EDR, but for $arch_{FT-RCN}$, FDR + EDR outperforms FDR + DIFF in best and worst game scores, but is slightly worse in average game scores compared to FDR + DIFF.

In addition, we visualize the EDR stream in Fig. 4, where we can see that EDR picks up short-term events that are relevant to winning the game. Plotting a sample training process for Pong game in Fig. 5(a), we found that both FDR and FDR+EDR networks achieve the maximum score of 21, but FDR + EDR can learn 1.7x faster (3 M vs. 5 M training episodes to reach a high score of 18) than the FDR alone. Furthermore, the strategy learned by the FDR+EDR is superior to that learned by

TABLE I
ATARI RL EXPERIMENT RESULTS: THE MIN, MAX AND MEAN TOTAL REWARD SCORE OF 6 DIFFERENT GAMES OVER 30 DIFFERENT PLAYS ARE SHOWN. COMPARISONS ARE MADE BETWEEN THE CONVENTIONAL FDR INPUTS AND PROPOSED FDR+EDR INPUTS. NOTICE THAT WORST CASE PERFORMANCE FOR FDR+EDR IS MUCH BETTER THAN THAT OF FDR ALONE

	FDR			FDR+DIFF			FDR+EDR		
	Avg.	Best	Worst	Avg.	Best	Worst	Avg.	Best	Worst
Pong	18.32	**21**	14	18.47	20	14	**19.47**	**21**	14
Breakout	422.53	494	378	**526.13**	864	**387**	446.60	**842**	365
Pacman	2828.63	6682	456	**6545.37**	**10751**	**2700**	3719.67	5030	1850
Centipede	1523.20	4243	302	**2915.03**	5895	456	2828.63	**6682**	456
Qbert	17088.80	19225	11775	**20579.17**	22950	11575	19560.00	**22800**	**19025**
Seaquest	**1691.00**	1700	1280	1638.00	**1760**	1300	1674.00	**1760**	**1360**
Avg. gain	-	-	-	+44.23%	+32.14%	+90.89%	+23.77%	+20.89%	+70.17%

(a) Experiment results of $arch_{base}$

	FDR			FDR+DIFF			FDR+EDR		
	Avg.	Best	Worst	Avg.	Best	Worst	Avg.	Best	Worst
Pong	20.90	**21**	**20**	20.73	**21**	19	**20.83**	**21**	19
Breakout	665.97	**864**	397	**706.80**	**864**	423	630.42	**864**	408
Pacman	4225.67	**8450**	1990	**5432.67**	7420	**2840**	4621.67	7630	1860
Centipede	1746.83	4483	228	1671.73	5206	206	**2450.40**	**8164**	**704**
Qbert	15662.50	16250	11525	**22970.83**	**27325**	7975	16766.67	26100	**11625**
Seaquest	2512.67	2560	2420	2479.33	2560	1960	**2532.00**	**2560**	**2480**
Avg. gain	-	-	-	+12.48%	+12.01%	-18.36%	+8.63%	+33.15%	+38.84%

(b) Experiment results of $arch_{FT-RCN}$

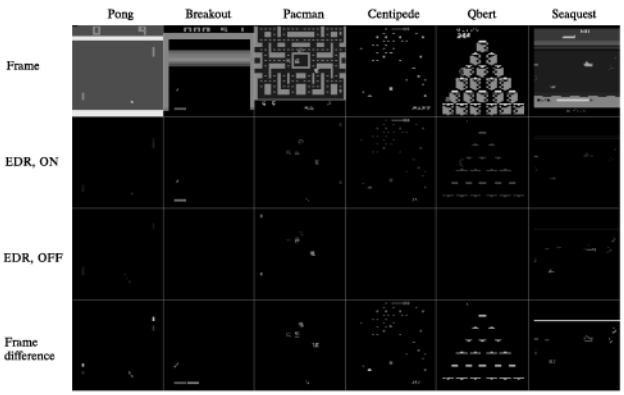


Fig. 4. Temporal input representations comparisons on RL Atari game play: Original frame, EDR and Diff representations for 6 tested Atari games are shown.

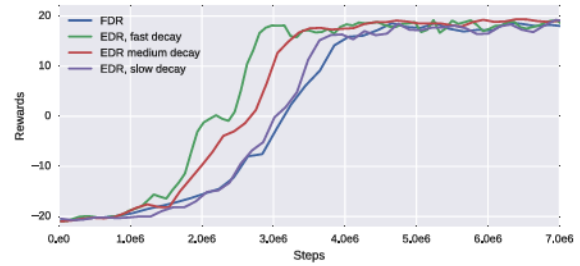
FDR, requiring only 1 shot to defeat the opponent as compared to multiple shots for FDR. Fig. 5(b) is an example training process for Centipede game. We found that the addition of the EDR representation accelerates training in PAAC. Moreover, we find that FDR+EDR based training reaches 1.8x higher reward than FDR alone and is still increasing when training terminates. FDR+EDR can reach a maximum reward of 8,300 later on in training while FDR struggles to reach 3,500. More training curves of different games can be found in the slides in supplementary material.

B. UCF-101 Action Recognition Experiments

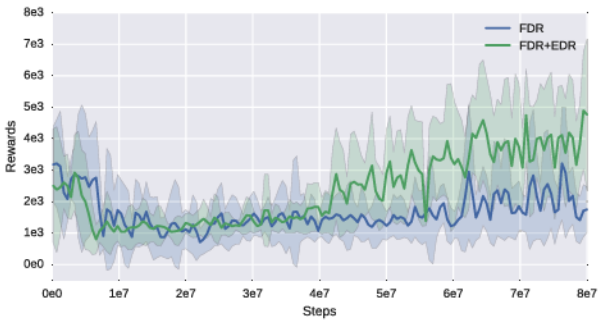
We compare EDR with temporal representation alternatives on Long Term Convolutional [68] (LTC) architecture. We would like to emphasis that the goal of this experiment is not to compete with the state-of-the-art methods in action recognition, but rather, understand how EDR with different parameters perform compared to other existing temporal input representations on state-of-the-art baseline network (i.e. LTC). We use the following representations for comparison: RGB, MPEG Flow, difference between grayscale intra-frame differences (GrayDiff), RGB intra-frame differences (RGBDiff), TV-L^1 OF [53], Farneback OF [24], Brox OF [15], fast decay soft threshold EDR ($\alpha = 0.5$) and slow decay soft threshold EDR ($\alpha = 0.166$). In addition, we test the effectiveness of the proposed EDR as a second stream on the state-of-the-art two-stream video recognition framework, Inflated 3D ConvNets (I3D) [17].

1) Datasets: UCF-101 [66] consists of 101 action categories of more than 13 k video clips in 30 fps. Each clip is in 320 $\times$ 240 resolution and the average mean clip length is 7.21 seconds. UCF-101 contains videos with rich camera motion and cluttered background, and comes with 3 pre-determined train/test splits. All splits are designed to balance out video nuisances such as actors and background, ensuring fairness between training and testing sets. We also perform standard data augmentation when loading the data, where the details can be found in *Implementation Details*. We report performance on the first splits in all our experiments.

2) Network Architecture: We use LTC to study the effectiveness of different motion features. LTC architecture has five 3D convolutional layers with 3 $\times$ 3 $\times$ 3 filter size, ReLU nonlinearity and volumetric max pooling, followed by 2 fully connected layers with 2048 units and dropout. LTC originally compares different input modalities using 60 frame architecture. LTC reaches highest 92.3% accuracy on UCF-101 when pre-trained



(a) 'Pong' Training Curve for Different EDR Time Scales



(b) 'Centipede' Training Curve

Fig. 5. RL training: (a) Rewards over game steps at training are shown for Pong. Higher is better. FDR+EDR learns 2x faster than the FDR alone. The strategy learned by the FDR+EDR is superior to that learned by FDR, requiring only 1 shot to defeat the opponent as compared to multiple shots for FDR. (b) Rewards over game steps at training are shown for Centipede. FDR+EDR reaches 1.8x higher reward than FDR alone.

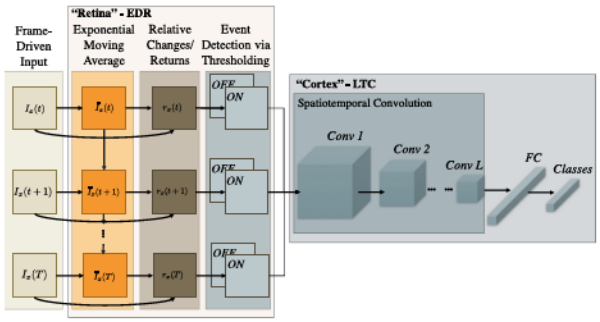
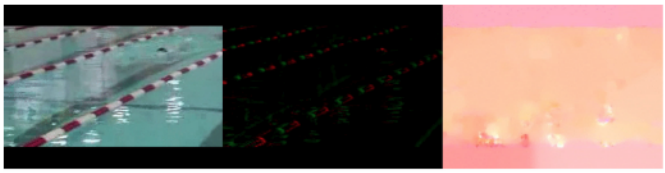


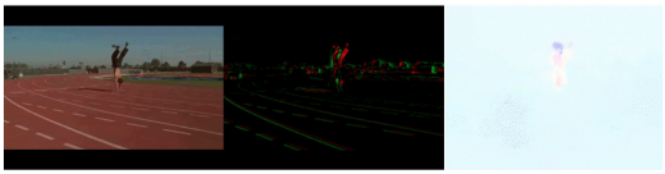
Fig. 6. System architecture diagram: There are two major components in our proposed bio-inspired architecture for video recognition. (1) A retinomorphic front-end for generating input events. (2) A back-end spatiotemporal ConvNet based on the state-of-the art LTC [68] that provides high-level semantic understanding of the events.

on Sports-1M [32], and combining Brox OF and Improved Dense Trajectories (IDT) along with the RGB input. Here we use the same setup, but without the Sprots-1 M pre-training to enable fair comparison and save computation cost. For the details about the LTC Network, we refer the reader to the original paper [68] and the implementation details section below. We used the I3D network to compare the performance of EDR (as the second stream) with the conventional OF stream. We refer the reader to [17] for the details of the I3D architecture.

3) Implementation Details: For experiments run on LTC, the architecture is shown in Fig. 6. We have two stages of input



(a)



(b)

Fig. 7. UCF-101 EDR visualization: Single frame comparison between original, EDR and Brox [15] for a video in (a) BreastStroke class and in (b) HandstandWalking class.

processing. The first stage happens before training and computes different representations of input videos in their original resolution. The second stage happens during training and involves several data augmentation techniques. First, input videos are scaled into 89 $\times$ 67 pixels. Second, we randomly sample a volume with size (height, width, 60) from the re-scaled input videos, where height and width are randomly chosen from (1, 0.875, 0.75, 0.66) multiples of scaled video size. Randomly sampled volumes are then scaled into network input dimension of 58 $\times$ 58 pixels. Each video is also randomly flipped horizontally with 0.5 chance. The standard evaluation metric is video accuracy. During test time, the first 60 frames of the test video are used as test clips. Each clip is cropped from its 4 corners and center, forming 5 cropped clips. Each cropped clip is further flipped horizontally to create a total of 10 cropped clips for each test video. Class result is computed as the maximum of averaged softmax scores of all 10 cropped clips. We use stochastic gradient descent as our training algorithm. We treat 9000 video clips as one epoch and stop training after 26 epochs. Learning rate is 1e-3 from epoch 1 to epoch 13, 1e-4 from epoch 14 to 24 and 1e-5 for last two epochs. Batch size is 15. For I3D experiments, we use the original network. Note that good performance of I3D and LTC on UCF-101 relies on pre-training with large datasets. I3D, for example, uses the Kinetics dataset [33] for pre-training, such that the authors had to utilize a 64-GPU cluster for the task [17]. Given that EDR is drastically different from RGB and OF, we can not directly use the pre-trained weights. Therefore, for fair comparison, we train the I3D and LTC networks in all our experiments from scratch with all different inputs (i.e. RGB, EDR, OF, two-stream RGB+OF and RGB+EDR, etc), thus the accuracy numbers may be worse than the best numbers presented in the original work. Nevertheless, we believe the comparison is fair and our goal here is not to beat the state-of-the-art in video recognition, but to showcase the efficiency of the proposed temporal representation, EDR, as compared to standard motion features like OF.

4) EDR Visualizations: Fig. 7(a) compares a single frame of RGB, EDR and Brox in the BreastStroke class. EDR has 60% higher accuracy than Brox and is able to retain the texture of

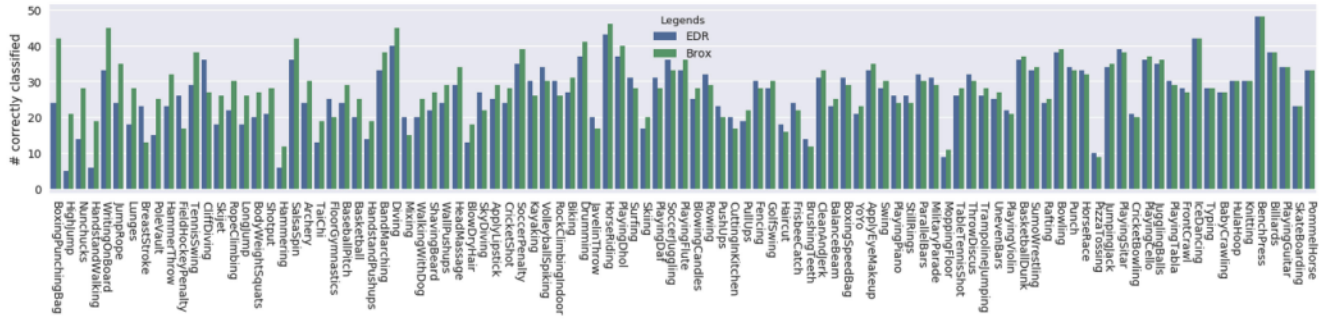


Fig. 8. UCF-101 Results - EDR vs. RGB FDR: Number of correctly classified videos per class between EDR and Brox [15]. Blue is EDR, green is Brox, sorted by the difference between the two for each class.

TABLE II
COMPARING DIFFERENT TEMPORAL REPRESENTATION ON THE UCF-101 ACTION RECOGNITION TASKS: WE SHOW THE TEST ACCURACY FOR DIFFERENT INPUT REPRESENTATIONS IN THE UCF-101 ACTION RECOGNITION TEST ON LTC. WE FOUND THAT EDR PERFORMS BETTER THAN SIMPLE INTER-FRAME DIFFERENCE AND THE MORE COMPUTATIONALLY EXPENSIVE FARNEBACK OPTICAL FLOW. HOWEVER, EDR IS LESS ACCURATE THAN THE OPTICAL FLOW METHOD PROPOSED BY BROX. NOTE THAT SPEED REFERS TO COMPUTATION OF THE MOTION FEATURES, NOT THE TOTAL EXECUTION TIME

Type	Representations	Accuracy	Speed (FPS)
Frame	RGB	57.0 [68]	-
Frame Diffs	Gray	59.5	37.6K
	RGB	66.8	36.9K
Flow	MPEG Flow [31]	58.5 [68]	591.8 [31]
	OF- Farneback [24]	66.3 [68]	27.3
	OF - TV-L^1 [53]	68.0	29.1 [53]
	OF - Brox [15]	**74.8** [68]	6.3
EDR	$\tau_{1/2} = 1$ frame	68.3	9.8K
	$\tau_{1/2} = 3.8$ frames	**70.7**	9.8K

TABLE III
EDR VS. OF IN TWO-STREAM VIDEO RECOGNITION TASK: WHEN TRAINING FROM SCRATCH USING I3D NETWORK ON TWO-STREAM INPUTS, OUR RGB + EDR IS SLIGHTLY WORSE THAN RGB + OF. HOWEVER, THE EDR IS SIGNIFICANTLY FASTER THAN THE OF TO COMPUTE

Type	2^{nd} Stream Method	Accuracy	Speed (FPS)
OF + RGB	Brox [15]	**68.33**	6.3
EDR + RGB	$\tau_{1/2} = 3.8$ frames	66.70	**9.8K**

pool lane lines, thanks to the large color gradient between lane lines (red and white) and water (green). Brox on the other hand, failed to extract a discernable pattern because moving water creates similar local pixel patches that confuses the optical flow algorithm. Fig. 7(b) compares RGB, EDR and Brox in the HandstandWalking class. Brox is 3× better than EDR and captures only body motions due to overall slow motion speed and stable local patterns. EDR on the other hand, falls short and pickup task irrelevant motion signals due to abundance of color gradient.

5) UCF-101 Experiment Results: A summary of different motion features' performance on UCF-101 on the baseline LTC network is shown in Table II. When training from scratch on UCF-101 data, among all the temporal representations, the best performing temporal representation is the Brox OF, which achieve 74.8% accuracy on UCF-101. Overall, EDR (70.7%) performs better than FDR, MPEG Flow [31], Farneback OF [24], TV-L^1 OF [53], and with a much faster computation speed measured by frame-per-second compared with all optical flow variants. Computing intra-frames difference (RGBDiff and GrayDiff) is indeed 4x faster than EDR, but the accuracy is noticeably lower. Nevertheless, 9.8 K fps on EDR is fast enough to enable real-time applications that are not practical for optical flows. We can safely conclude that EDR can trade-off accuracy slightly (and is still among the best performing temporal representations) for much better computing efficiency.

Fig. 8 compares number of samples correctly classified from EDR pre-processed network and Brox pre-processed network, sorted by the difference between the two, from large to small. Overall, EDR and Brox performs similarly: both found PizzaTossing as a difficult class and BenchPress as a simple one. However in a small portion of classes the differences are significant.

For two-stream experiments, we show the results in Table III. RGB + EDR on UCF-101, using two-stream I3D, gives accuracy of 66.7%, when training from scratch, while RGB + OF is slightly better at 68.33% under the same conditions. However, the computation of the EDR is significantly faster than the OF.

C. Analyzing EDR on KTH Action Recognition Datasets

In this section, we conduct the ablation studies of the proposed EDR. To perform various ablation experiments in a timely manner, we chose the KTH datasets [62] which are smaller than UCF-101. We analyze the performance of EDR working with different ConvNet front-end and recurrent back-end models for action recognition tasks. We also study the EDR's efficiency in terms of network activation sparsity, as well as training efficiency in terms of learning speed.

1) Network Architectures: Without the need to handle large scale datasets, we scale down our network used in the experiments as well. Our baseline network is a recurrent convolution network (RCN) [5], [20] which takes frame-driven video as input (see Fig. 10), and sends output to a deep ConvNet (DCN). We explore two different types of DCNs, Network-in-Network (NiN) [40] and LeNet [38]. We also explore two different

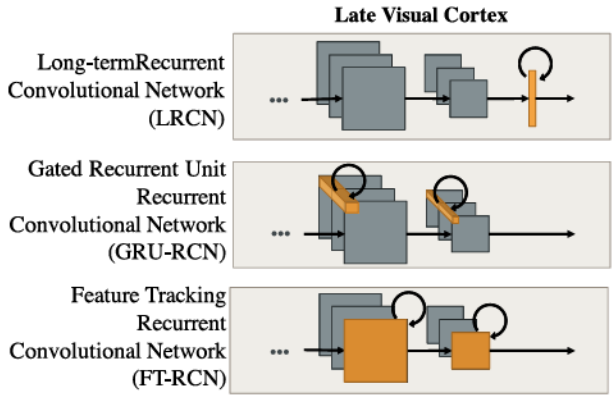


Fig. 9. Comparison of different RCNs: Late recurrent stages of the proposed FT-RCN, LRCN and GRU-RCN are shown.

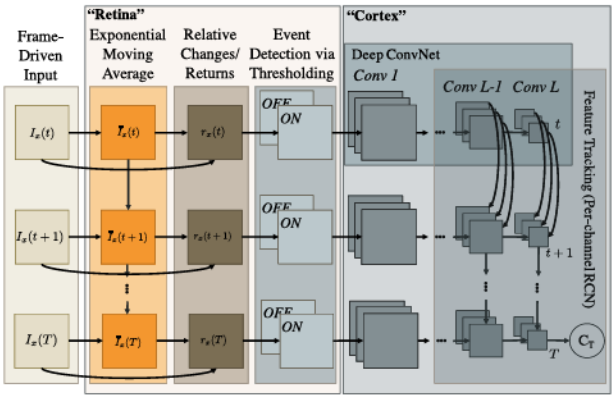


Fig. 10. System architecture diagram: The three major components in our proposed bio-inspired architecture for video recognition. (1) A retinomorphic front-end for generating input events. A neuromorphic back-end cortex that provides high-level semantic understanding of the events, including (2) early convolutional layers and (3) Late recurrent convolutional layers.

types of late-stage recurrent layers in our RCNs: LSTM-based (LRCN [20]) and GRU-based (GRU-RCN [5]).

2) Feature Tracking RCN: Since our proposed EDR extracts temporal features, our consequent recognition pipeline should track such features accordingly. As a matter of fact, a critical function of biological motion perception is feature tracking ("what target went where") [19], [42], [61] Our solution is a feature tracking recurrent convolutional neural (RCN) network to serve as the "visual cortex" for our high-level visual recognition tasks. We based the design on conventional RCNs, but introduce a simple modification to the recurrent update to address the feature tracking functionality. We name our proposed design FT-RCN. We implement this functionality by mimicking aspects of the recurrent within-channel connectivity observed in the visual cortex [30].

The diagram of Long-term RCN (LRCN), Gated Recurrent Unit (GRU) RCN and our proposed EDR feature tracking RCN (FT-RCN) are compared against each other in Fig. 9. As the figure shows, the differences lie in where the recurrence is introduced, and how recurrence is performed. LRCNs employ all-to-all recurrent connections in the last flattened fully connected layer. They are capable of noticing movement, but have lost the distinction between different features. For GRU-RCN, recurrence happens at each convolutional layer, and is for each same pixel location across all the different feature maps. It does preserve feature distinctions, but has no recurrent connections between distinct pixels, preventing it from noticing movement. In conclusion existing LRCN and GRU-RCN architectures have connectivity that makes tracking the movement of stable features difficult/impossible. In contrast, for the proposed FT-RCN, it is similar to GRU-RCN, except that the recurrence is occurring for the entire feature map. This is similar to the like-like connectivity in the visual cortex. Such design will help track high-level EDR features, as the channels in the late stage of DCNs correspond to high-level features.

We build a network that consists of three major components (see Fig. 10):

1) *Input Representation Layers:* We choose between our proposed event-driven representation and a conventional frame-driven representation of the input video.
2) *ConvNet Layers:* The ConvNet layers are used for spatial feature extraction. We are able to choose ConvNets of different structures here.
3) *Late-stage Recurrent Layers:* RCNs are used to preserve the temporal context of a dynamic scene. Common RCN architectures includes LRCN and GRU RCN. To this list we also add our proposed feature tracking RCN (FT-RCN) that accepts EDR input.

All together, we have several baseline network architectures that we compare to our EDR-based FT-RCNs. We can assess the value-add of each newly proposed component, and more generally, the value-add of event-driven representations and components.

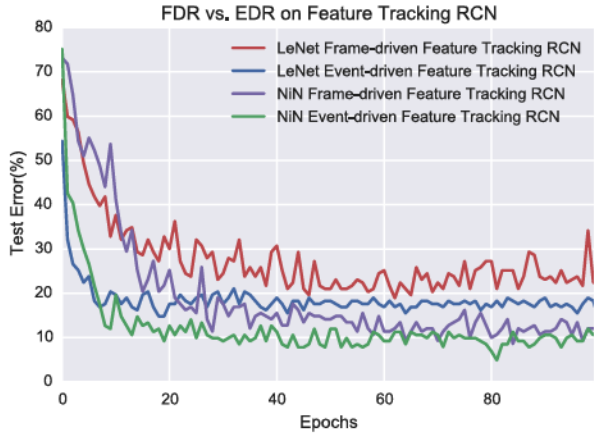
3) Datasets: The KTH dataset contains 600 120×160 grayscale videos of six action classes, where each class contains 100 videos. Given the videos have various lengths with at least 90 frames, therefore we use the first 90 frames of video for experiment. We randomly generate 75/25 train/test splits for use in learning.

4) Results and Analysis: Using the aforementioned networks, we perform end-to-end training for action recognition and compare results. Overall, the results show that with significant amount of data throughput reduction (i.e. much less activation) and potential computing energy savings (Fig. 12), we can observe noticeable improvement in both classification accuracy (Table IV) and training speed (Fig. 11), over conventional FDRs, achieving a near-state-of-the-art [1] results at 94.4% accuracy for KTH datasets. Moreover, our proposed FT-RCN "cortex" can nicely handle the event flow, and performs better overall, compared to conventional LRCN. The detailed results and analysis are discussed below.

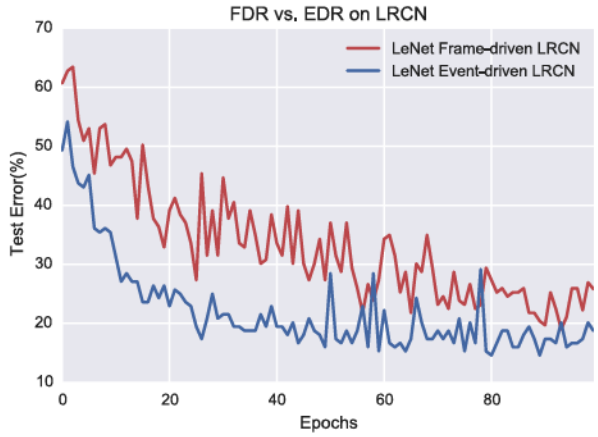
Input Representation. Primitive ON/OFF Event Detector: In this experiment, we evaluate our primitive event detector, which simulates the ON/OFF pathways between the photoreceptor and the bipolar ganglion cells in the retina. As shown in Table IV, when EDR is introduced as the input representation, we observe that the classification accuracy increases from

TABLE IV
RCN EXPERIMENT RESULTS: OVERALL, EDR PROVIDES BETTER CLASSIFICATION RESULTS THAN FDR IN RCN ON THE KTH ACTION RECOGNITION DATASETS. MEANWHILE, THE PROPOSED FT-RCN SEEMS TO BE A BETTER CORTEX FOR VIDEO RECOGNITION TASKS, AS IT IMPROVES THE OVERALL CLASSIFICATION PERFORMANCE

RCN Properties		Input Representation Properties	
Conv Model	Recurrent Structures	Frame-driven Representation	Event-driven Representation
LeNet	Long-term RCN	79.3 %	**86.7 %**
LeNet	Feature Tracking RCN	83.3 %	**88.9 %**
NiN		90 %	**94.4 %**



(a) Learning progress on feature tracking RCN



(b) Learning progress on long-term RCN

Fig. 11. Learning progress of different input representations, ConvNet type and RCN structure. One can observe that EDR results in better classification accuracy and faster convergence.

79.6%, 83.3% and 90% (FDR) to 86.7%, 88.9% and 94.4% (EDR), respectively. Moreover, the learning curve shows that learning with EDR is much faster than with the original FDR (Fig. 11).

Recurrent "Cortex" Model. Feature-Tracking RCN vs. LRCN: In this experiment, we study the effectiveness of the proposed FT-RCN in handling the EDR feature, compared to a vanilla LRCN. We fixed the input representation layers to EDR and the DCN layer to be LeNet, while setting the late-stage recurrent layers to be FT-RCN and LRCN.

We show the results of different recurrent models in Table IV. EDR-based FT-RCN achieves better performance in the action recognition test (79.3% and 86.7% vs. 86.7% and 88.9%, respectively). We believe the improvement comes from the FT-RCN's ability to better preserving high-level semantic features. As one can see, FT-RCN does a good job accommodating the EDR input, and resolves seemingly more interpretable features compared to the LRCN.

ConvNet Structure. NiN vs. LeNet: In addition, we also explore the effect of different DCN structures. Empirically speaking, deeper DCNs usually provide better end-to-end recognition results, since more layers means more nuisance disentanglement and thus better higher-level feature abstraction. In our experiment, we fix the input to be EDR, and the RCN to be FT-RCN, and then train end-to-end for the video recognition task using two DCN structures — a shallower LeNet and a deeper NiN network. We show the classification results in Table IV. Unsurprisingly, the NiN-based DCN delivers better overall classification results. Fig. 11 shows the learning curves. Interestingly, it seems that EDR provides much more benefit for the shallower LeNet, as compared to the deeper NiN. One possible explanation is that the EDR captures nuisance more directly, thus the higher-level features become more linearly separable. Therefore, even a shallower ConvNet structure will be able to perform more complex tasks well.

Sparsity, Energy Saving and Computing Efficiency: Compared to the FDR, the EDR provides higher sparsity. We plot the histogram of the activation at different layers of the neural network for both EDR and FDR input in Fig. 12. As one can observe, in general, EDR results in a much sparse activation, especially in the early layers. For the activation in the first couple layers in the cortex, EDR results in a tri-mode distribution, which is attributed to ON/OFF pathway design, while FDR's activation is further spread out. Furthermore, EDR's sparsity and binary ON/OFF pathway will significantly save the data bandwidth and computing power required.

D. Hardware Experiments

Currently, computer vision systems for semantic understanding are mostly based on a conventional camera and neural network that are designed for RGB inputs, with a focus on test accuracy, while power consumption and speed are largely under-emphasized. As a result, applications in always-on or embedded vision scenarios are greatly limited. Continuously-on ADC and data transmission regardless of scene context results in wasting 90% total power consumption [69]. As reviewed earlier, the commercially available DVS camera has a similar event-driven mechanism as our proposed EDR. Using the DVS camera could reduce the energy consumption by up to three orders of magnitude (Table V), while still capturing the essence of motion dynamics in the scene. Given that DVS camera outputs do lose some of the features of EDR, such as EMA-based smoothing, soft-thresholding and multiple timescales, here we hope to show that the deep learning model trained using our

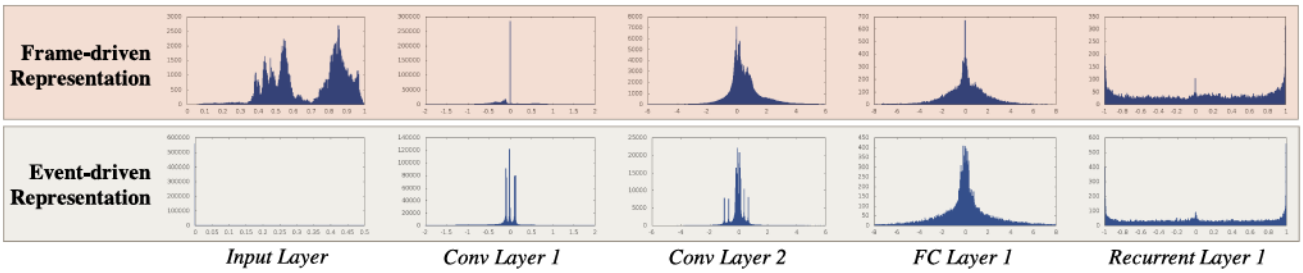


Fig. 12. Activation distribution in the artificial cortex: Histogram of activation at different layers in EDR and FDR are compared against each other. In general, EDR generates a much more sparse activation than FDR. The ON/OFF pathway in EDR produces a visible multimodal distribution in early layers.

TABLE V
DVS HARDWARE COMPARISON: COMPARISON OF DVS128 CAMERA AND A MAINSTREAM INTENSITY CAMERA (ADAPTED FROM [7]). DVS HAS SIGNIFICANT ADVANTAGES IN ENERGY, SPEED, DYNAMIC RANGE, AND DATA BANDWIDTH

Specification	Conventional Intensity Camera (Grasshopper 3)	Event-driven Camera (DVS128)
Resolution	2048 x 2048	128 x 128
Total Power	4.5 W	23 mW
Power per video voxel	11.9 nW / voxel	<0.7 nW / voxel
Max. Frame Rate	90 Hz	2 kHz (1M events/sec)
Dynamic Range	52.87 dB	120 dB
Max. Bandwidth	360 Mbps	4 Mbps

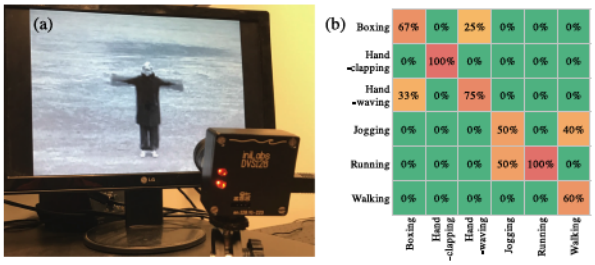


Fig. 13. DVS hardware experiment: (a) Experimental setup. (b) Confusion matrix of the classification results.

software-based EDR input can be easily transferred to the DVS hardware for real-world usage. Ultimately, if EDR is implemented in hardware, it may yield an energy-efficient imaging system for dynamic event recognition. We set up an early-stage prototype to validate this idea and showcase a hardware based event-driven action recognition pipeline.

1) Experiments: Our network for this experiment is the EDR-pre-trained FT-RCN from Section IV-C. We perform transfer learning using real-world DVS data to fine-tune the network to accommodate the DVS hardware. At testing, we send the DVS data directly to the fine-tuned FT-RCN without going through the EDR layers. We show the hardware experiment setup in Fig. 13(a), where we have the DVS128 event camera capture the actions displayed on a computer screen. The action clips are taken from the KTH datasets. We capture 10 event streams for each of the six action classes. The raw DVS event streams are integrated every 0.033 second to form an ON/OFF event image. Event inputs are split by a 70/30 ratio. 70% of the event streams are used for fine-tuning the pre-trained model, and the rest 30% are used for testing. After 100 epochs of fine-tuning on the training set, we perform testing on the testing set. We achieve an overall **72.2%** classification accuracy in this six-class classification problem. The confusion matrix of the classification results are shown in Fig. 13(b).

2) Limitations: We note here that although the DVS camera is used, it was capturing only the dynamics of the RGB frames (30 fps video) displayed *on the screen*, not the true dynamics of the scene. Thus it's not fully exploiting the full benefits of the DVS sensor that are shown in Table V. That said, we would like to emphasize that the main point of this experiment is not to demonstrate the benefit of the event sensor, nor advancing the state-of-the-art results on action recognition tasks, but rather, to show that the deep learning model trained using our software-based EDR input can be easily transferred to the DVS hardware for real-world usage. Ideally we should run the experiments on real-world-captured DVS data, however data-driven high-level visual understanding tasks require a significant amount of data to be collected (small datasets like KTH still contain 600 videos), which is outside the scope of our paper.

V. CONCLUSION

In summary, we propose EDR, an event-driven retinomorphic input representation, for extracting temporal dynamics in video. We show that EDR improves performance in Atari game playing reinforcement learning (Section IV-A) and dramatically improves the speed-accuracy tradeoffs in UCF-101 action recognition tasks, compared with alternative temporal representations (Section IV-B). We then use a smaller action recognition dataset, KTH, to analyze EDR's efficiency and performance under a different network "cortex" structure, and found EDR copes especially well with a network that can track features temporally (e.g. FT-RCN). Furthermore, as shown in Section IV-D, we build an early-stage prototype consisting of DVS event camera hardware and a light-weight FT-RCN for performing standard action recognition tasks with orders of magnitude reduction in sensing power and data transmission rates. Meanwhile, we acknowledge that our current EDR architecture is an early proof-of-concept and has some limitations. We have not yet implemented some important properties of the retina. For instance, first, we do not have a stable learnable EDR architecture. In our preliminary

experiments, end-to-end learning the EDR parameters seems to be quite unstable, probably due to the fact that EDR is formulated as an RNN, and the Back-propagation-thru-Time (BPTT) algorithm used to train it yields unstable gradients. Second, We do not include transient and sustained cell types, resulting in some loss of adaptability that is inherent in the retina. Lastly, We do not simulate any spatial processing in the retina (e.g. Horizontal and Amacrine cells), therefore our events are based solely on temporal changes, rather than spatiotemporal changes. We hope our preliminary studies on EDR will motivate more research in this direction.

REFERENCES

[1] H. A. Abdul-Azim and E. E. Hemayed, "Human action recognition using trajectory-based representation," *Egypt. Inform. J.*, vol. 16, no. 2, pp. 187–198, 2015.

[2] A. G. Andreou, R. C. Meitzler, K. Strohbehn, and K. A. Boahen, "Analog VLSI neuromorphic image acquisition and pre-processing systems," *Neural Netw.*, vol. 8, no. 7, pp. 1323–1347, 1995.

[3] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, "Sequential deep learning for human action recognition," in *Proc. Int. Workshop Human Behav. Understand.*, 2011, pp. 29–39.

[4] T. Baden, P. Berens, M. Bethge, and T. Euler, "Spikes in mammalian bipolar cells support temporal layering of the inner retina," *Current Biol.*, vol. 23, no. 1, pp. 48–52, 2013.

[5] N. Ballas, L. Yao, C. Pal, and A. Courville, "Delving deeper into convolutional networks for learning video representations," 2015, *arXiv:1511.06432*.

[6] P. Bardow, A. J. Davison, and S. Leutenegger, "Simultaneous optical flow and intensity estimation from an event camera," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 884–892.

[7] S. Barua, Y. Miyatani, and A. Veeraraghavan, "Direct face detection and video reconstruction from event cameras," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2016, pp. 1–9.

[8] R. Berner, C. Brandli, M. Yang, S.-C. Liu, and T. Delbruck, "A 240× 180 10 mW 12us latency sparse-output vision sensor for mobile applications," in *Proc. Symp. VLSI Circuits*, 2013, pp. C186–C187.

[9] H. R. Blackwell and O. M. Blackwell, "Rod and cone receptor mechanisms in typical and atypical congenital achromatopsia," *Vis. Res.*, vol. 1, no. 1, pp. 62–107, 1961.

[10] K. Boahen, "Retinomorphic vision systems," in *Proc. 5th Int. Conf. Microelectron. Neural Netw.*, 1996, pp. 2–14.

[11] K. Boahen, "Neuromorphic microchips," *Sci. Am.*, vol. 292, no. 5, pp. 56–63, 2005.

[12] A. Borst and M. Helmstaedter, "Common circuit design in fly and mammalian motion vision," *Nature Neurosci.*, vol. 18 no. 8, pp. 1067–1076, 2015.

[13] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240× 180 130 dB 3 μs latency global shutter spatiotemporal vision sensor," *IEEE J. Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, Oct. 2014.

[14] G. Brockman *et al.*, "Openai gym," 2016, *arXiv:1606.01540*.

[15] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," *Proc. Eur. Conf. Comput. Vis.*, 2004, pp. 25–36.

[16] C. Cadena *et al.*, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016.

[17] J. Carreira and A. Zisserman, "*Quo vadis*, action recognition? A new model and the kinetics dataset," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4724–4733.

[18] A. V. Clemente, H. N. Castejón, and A. Chandra, "Efficient parallel methods for deep reinforcement learning," May 2017, *arXiv preprint arXiv:1705.04862*.

[19] J. C. Culham, S. A. Brandt, P. Cavanagh, N. G. Kanwisher, A. M. Dale, and R. B. H. Tootell, "Cortical FMRI activation produced by attentive tracking of moving targets," *J. Neurophysiol.*, vol. 80, no. 5, pp. 2657–2670, 1998.

[20] J. Donahue *et al.*, "Long-term recurrent convolutional networks for visual recognition and description," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 2625–2634.

[21] R. Etienne-Cummings and J. Van Der Spiegel, "Neuromorphic vision sensors," *Sens. Actuators A, Phys.*, vol. 56, no. 1/2, pp. 19–29, 1996.

[22] E. V. Famiglietti and H. Kolb, "Structural basis for on-and off-center responses in retinal ganglion cells," *Science*, vol. 194, no. 4261, pp. 193–195, 1976.

[23] L. Fan *et al.*, "End-to-end learning of motion representation for video understanding," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2018.

[24] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Proc. Scand. Conf. Image Anal.*, 2003, pp. 363–370.

[25] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Comput.*, vol. 12, no. 10, pp. 2451–2471, 2000.

[26] J. R. Golden *et al.*, "Simulation of visual perception and learning with a retinal prosthesis," *J. Neural Eng.*, vol. 16, no. 2, 2017, Art. no. 025003.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[29] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size," 2016, *arXiv:1602.07360*.

[30] X. Jiang *et al.*, "Principles of connectivity among morphologically defined cell types in adult neocortex," *Science*, vol. 350, no. 6264, 2015, Art. no. aac9462.

[31] V. Kantorov and I. Laptev, "Efficient feature extraction, encoding and classification for action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 2593–2600.

[32] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 1725–1732.

[33] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2017, pp. 6299–6308.

[34] H. Kim, S. Leutenegger, and A. J, Davison, "Real-time 3D reconstruction and 6-DoF tracking with an event camera," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 349–364.

[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[36] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: A large video database for human motion recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 2556–2563.

[37] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[38] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[39] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 × 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.

[40] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013, *arXiv:1312.4400*.

[41] J. Loveless, S. Stoikov, and R. Waeber, "Online algorithms in high-frequency trading," *Commun. ACM*, vol. 56, no. 10, pp. 50–56, 2013.

[42] Z.-L. Lu and G. Sperling, "The functional architecture of human visual motion perception," *Vis. Res.*, vol. 35, no. 19, pp. 2697–2722, 1995.

[43] N. Maheswaranathan, S. A. Baccus, and S. Ganguli, "Inferring hidden structure in multilayered neural circuits," *PLoS Comput. Biol.*, vol. 14, no. 8, 2017, Art. no. e1006291.

[44] N. Matsuda, O. Cossairt, and M. Gupta, "MC3D: Motion contrast 3D scanning," in *Proc. IEEE Int. Conf. Comput. Photogr.*, 2015, pp. 1–10.

[45] A. S Mauss, M. Meier, E. Serbe, and A. Borst, "Optogenetic and pharmacologic dissection of feedforward inhibition in Drosophila motion vision," *J. Neurosci.*, vol. 34, no. 6, pp. 2254–2263, 2014.

[46] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.

[47] V. Mnih *et al.*, "Playing Atari with deep reinforcement learning," *NIPS Deep Learning Workshop*. 2013.

[48] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[49] E. Mueggler, B. Huber, and D. Scaramuzza, "Event-based, 6-DoF pose tracking for high-speed maneuvers," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 2761–2768.

[50] R. Nelson, E. V. Famiglietti, and H. Kolb, "Intracellular staining reveals different levels of stratification for on-and off-center ganglion cells in cat retina," *J. Neurophysiol.*, vol. 41, no. 2, pp. 472–483, 1978.

[51] P. O'Connor and M. Welling, "Sigma delta quantized networks," 2016, *arXiv preprint arXiv:1611.02024*.
[52] R. D. Penn and W. A. Hagins, "Kinetics of the photocurrent of retinal rods," *Biophys. J.*, vol. 12, no. 8, pp. 1073–1094, 1972.
[53] J. S. Pérez, E. Meinhardt-Llopis, and G. Facciolo, "TV-L1 optical flow estimation," *Image Process. On Line*, vol. 2013, pp. 137–150, 2013.
[54] J. A. Pérez-Carrasco *et al.*, "Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing—Application to feedforward ConvNets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2706–2719, Nov. 2013.
[55] C. Posch, D. Matolin, and R. Wohlgenannt, "A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS," *IEEE J. Solid-State Circuits*, vol. 46, no. 1, pp. 259–275, Jan. 2011.
[56] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, and T. Delbruck, "Retinomorphic event-based vision sensors: Bioinspired cameras with spiking output," *Proc. IEEE*, vol. 102, no. 10, pp. 1470–1484, Oct. 2014.
[57] H. Rebecq, G. Gallego, and D. Scaramuzza, "EMVS: Event-based multi-view stereo," in *Proc. Brit. Mach. Vis. Conf.*, 2016, Paper EPFL-CONF-221504.
[58] H. Rebecq, T. Horstschafer, G. Gallego, and D. Scaramuzza, "EVO: A geometric approach to event-based 6-DOF parallel tracking and mapping in real-time," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 593–600, Apr. 2016.
[59] C. Reinbacher, G. Munda, and T. Pock, "Real-time panoramic tracking for event cameras," 2017, *arXiv:1703.05161*.
[60] S. Sadanand and J. J. Corso, "Action bank: A high-level representation of activity in video," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 1234–1241.
[61] M. Saenz, G. T. Buracas, and G. M. Boynton, "Global effects of feature-based attention in human visual cortex," *Nature Neurosci.*, vol. 5, no. 7, pp. 631–632, 2002.
[62] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: A local SVM approach," in *Proc. 17th Int. Conf. Pattern Recognit.*, 2004, vol. 3, pp. 32–36.
[63] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 568–576.
[64] A. Sironi, M. Brambilla, N. Bourdis, X. Lagorce, and R. Benosman, "HATS: Histograms of averaged time surfaces for robust event-based object classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1731–1740.
[65] S. M. Smirnakis, M. J. Berry, D. K. Warland, W. Bialek, and M. Meister, "Adaptation of retinal processing to image contrast and spatial scale," *Nature*, vol. 386 no. 6620, pp. 69–73, 1997.
[66] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," CRCV-TR-12-01, Nov. 2012.
[67] W. K. Stell, A. T. Ishida, and D. O. Lightfoot, "Structural basis for on-and off-center responses in retinal bipolar cells," *Science*, vol. 198, no. 4323, pp. 1269–1271, 1977.
[68] G. Varol, I. Laptev, and C. Schmid, "Long-term temporal convolutions for action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1510–1517, Jun. 2017.
[69] A. Wang, S. Sivaramakrishnan, and A. Molnar, "A 180 nm CMOS image sensor with on-chip optoelectronic image compression," in *Proc. IEEE Custom Integr. Circuits Conf.*, 2012.
[70] L. Wang *et al.*, "Temporal segment networks: Towards good practices for deep action recognition," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 20–36.
[71] C. -Y. Wu, M. Zaheer, H. Hu, R. Manmatha, A. J. Smola, and P. Krähenbühl, "Compressed video action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6026–6035.
[72] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman, "Video enhancement with task-oriented flow," *Int. J. Comput. Vision*, vol. 127, no. 8, pp. 1106–1125, 2019.
[73] K. Yoshida, D. Watanabe, H. Ishikane, M. Tachibana, I. Pastan, and S. Nakanishi, "A key role of starburst amacrine cells in originating retinal directional selectivity and optokinetic eye movement," *Neuron*, vol. 30, no. 3, pp. 771–780, 2001.
[74] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 4694–4702.
[75] K. A. Zaghloul and K. Boahen, "A silicon retina that reproduces signals in the optic nerve," *J. Neural Eng.*, vol. 3 no. 4, pp. 257–267, 2006.
[76] Y. Zhu, Z. Lan, S. Newsam, and A. G. Hauptmann, "Hidden two-stream convolutional networks for action recognition," in *Proc. Asian Conf. Comput. Vision*, Springer, Cham, Dec. 2018, pp. 363–378.

Huaijin Chen received the B.S. degree (magna cum laude) in imaging science from the Rochester Institute of Technology, Rochester, NY, USA, in 2013, and the M.S. and Ph.D. degrees in electrical and computer engineering from Rice University, Houston, TX, USA, in 2016 and 2019, respectively, where he worked with Prof. Ashok Veeraraghavan on leveraging physics-based model for data-driven computational imaging. In 2019, he joined SenseTime Research as a Research Scientist, where he helped found the company's Computational Photography Lab in San Jose, CA, USA. He cofounded Sense.watch, a startup that provides machine-learning-based solutions to interpret wearable sensor data, and NAKAMA, a critically acclaimed whiskey lounge in Shenzhen, China. He was a recipient of the Texas Instruments Distinguished Graduate Student Fellowship and the best poster awards at the IEEE International Conference on Computational Photography 2019.

Wanjia Liu was born in Beijing, China, in 1992. He received the B.E. degree in electrical engineering and the M.S. degree in computer science from Rice University, Houston, TX, USA, in 2016 and 2018, respectively. In 2015, he joined Dr. Ankit Patel's Lab as an Intern and worked on computational neural science and deep learning research. He is a Software Engineer with Google.

Rishab Goel received M.Tech. degree in computer science and engineering from IIT Delhi, Delhi, India, in 2018. Since then, he has been a Research Engineer with Borealis AI, Montreal, QC, Canada. His broad areas of interest are applications in computer vision, relational/graph representation learning, and time series.

Rhonald C. Lua received the Ph.D. degree in physics from the University of Minnesota, MN, USA, in 2005, where he did theoretical and computational work on knots and statistical mechanics. After a brief stay at the National Institute of Standards and Technology, Washington, DC, USA, he has spent the past several years at the Texas Medical Center in Houston, TX, USA, working in the field of computational biology. More recently, he was with the Patel Lab, Baylor College of Medicine and Rice University, where he was exposed to deep learning and neuroscience, and at the Johnson Space Center's Visitor Center, where he served tours during the 50th anniversary of the Apollo 11 mission.

Siddharth Mittal received the B.Tech. degree in computer science and engineering from the Indian Institute of Technology, Kanpur, India, in 2019. Since then, he has been a Software Developer with an algorithmic trading firm, Quadeye, Gurgaon, India. His research interests include the application of deep learning to computer vision and NLP.

Yuzhong Huang received the B.S. degree in engineering from the Olin College of Engineering, Needham, MA, USA, in 2018. He is a Machine Learning Engineer with Kensho Technology, Cambridge, MA, USA. In 2016 and 2017, during his internship with the Baylor College of Medicine and Rice University in Houston, TX, USA, he conducted research in the area of computer vision, specifically video action recognition. His current work focuses mainly on topic modeling and text extraction in finance.

Ashok Veeraraghavan received the bachelor's degree in electrical engineering from the Indian Institute of Technology, Madras, Chennai, India, in 2002 and the M.S. and Ph.D. degrees from the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD, USA, in 2004 and 2008, respectively. He is currently an Associate Professor of Electrical and Computer Engineering with Rice University, Houston, TX, USA, where he directs the Computational Imaging and Vision Lab. Before joining Rice University, he spent three years as a Research Scientist with Mitsubishi Electric Research Labs, Cambridge, MA, USA. His research interests are broadly in the areas of computational imaging, computer vision, machine learning, and robotics. Dr. Veeraraghavan's thesis received the Doctoral Dissertation Award from the Department of Electrical and Computer Engineering, University of Maryland. He was the recipient of the National Science Foundation CAREER Award in 2017.

Ankit B. Patel received the undergraduate and graduate degrees in applied mathematics and computer science from Harvard University, Cambridge, MA, USA. He is currently an Assistant Professor with the Department of Neuroscience, Baylor College of Medicine and the Department of Electrical and Computer Engineering, Rice University, Houston, TX, USA. His research interests include the intersection between machine learning and computational neuroscience, two areas essential for understanding and building truly intelligent systems, with a focus on the low-level mechanisms by which learned representations work. He works with neuroscientists to build a bridge between artificial and real neuronal networks, using theories and experiments about artificial nets to help understand and make testable predictions about real brain circuits. He returned to academia after spending six years in the industry, building real-time inference systems trained on large-scale data for ballistic missile defense (MIT Lincoln Laboratory), and high-frequency trading.