A Low Power, Fully Event-Based Gesture Recognition System

Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza†, Jeff Kusnitz, Michael Debole, Steve Esser, Tobi Delbruck‡, Myron Flickner, and Dharmendra Modha

IBM Research
{arnon, dmodha}@us.ibm.com

†UC San Diego

‡UZH-ETH Zurich
& iniLabs GmbH
tobi@ini.uzh.ch

Abstract

We present the first gesture recognition system implemented end-to-end on event-based hardware, using a TrueNorth neurosynaptic processor to recognize hand gestures in real-time at low power from events streamed live by a Dynamic Vision Sensor (DVS). The biologically inspired DVS transmits data only when a pixel detects a change, unlike traditional frame-based cameras which sample every pixel at a fixed frame rate. This sparse, asynchronous data representation lets event-based cameras operate at much lower power than frame-based cameras. However, much of the energy efficiency is lost if, as in previous work, the event stream is interpreted by conventional synchronous processors. Here, for the first time, we process a live DVS event stream using TrueNorth, a natively event-based processor with 1 million spiking neurons. Configured here as a convolutional neural network (CNN), the TrueNorth chip identifies the onset of a gesture with a latency of $105\ \mathrm{ms}$ *while consuming less than* $200mW$*. The CNN achieves* 96.5% *out-of-sample accuracy on a newly collected DVS dataset (DvsGesture) comprising 11 hand gesture categories from 29 subjects under 3 illumination conditions.*

1. Introduction

Event-based computation, used by the eye and the brain, is a biologically inspired paradigm for using sparse, asynchronous events to represent data more efficiently than the dense, synchronous frames that are captured and processed by most modern sensors and computing devices. Remarkably, despite the slower speed of biological neurons and synapses compared to silicon transistors, biological systems still solve complex vision problems faster and at lower power than conventional computers, leveraging parallel, distributed, event-based computation to operate efficiently in real-time, resource-constrained environments.

The prime example of a frame-based computing device is a digital camera, which repeatedly scans out its entire array of pixels at a predetermined frame rate, independent of any activity actually observed in the scene. Frame-based cameras have two major drawbacks. First, the camera's reaction speed is limited to its frame rate, typically 30 frames per second. Second, consecutive video frames are highly redundant, and acquisition of redundant data wastes considerable resources, both in the camera itself and in any downstream devices, since energy and bandwidth requirements are driven by the rate of data transfer [36].

Event-based cameras mimic the biological retina by sending an asynchronous event whenever a pixel detects a change in brightness, eliminating redundant data transmission [2, 36, 22]. The data transfer rate can vary from very few events when observing a static scene, to many events when a large fraction of the scene changes, allowing energy and bandwidth consumption to scale dynamically with actual demand. The high temporal resolution is an effective sampling rate that can only be matched by a high-speed frame-based camera with dramatically higher energy and bandwidth requirements. Event-based cameras also have very high dynamic range relative to standard cameras.

The advantages of event-based sensors are diluted if their event streams must be cast back into synchronous frames for the benefit of conventional processors downstream. Conventional processors, like CPUs and GPUs, are efficient in processing dense, synchronously delivered data structures, not sparse, asynchronous event streams. Throughput is maintained by keeping the instruction and data pipelines as full as possible, even at the cost of executing redundant computations on unchanging data [7].

Recently, a new generation of natively event-based neuromorphic processors has appeared that can operate on sensor event streams directly [11]. These many-core systems instantiate large populations of spiking neurons in massively parallel, low-power hardware, and inherit all of the advantages of event-based sensors, such as efficient, data-

1063-6919/17 $31.00 © 2017 IEEE
DOI 10.1109/CVPR.2017.781

driven resource consumption. Processing latency can be as fast as the event propagation time through the longest chain of neurons, so a neural network running on these systems can react to a stimulus in tens of milliseconds, fast enough to identify quick motions like hand gestures in real-time.

Real-time hand-gesture recognition is a practical problem well-suited to event-based computation. Hand gestures are ubiquitous in visual cognition, pervading body language in all ages and cultures and tightly integrated with verbal communication [16]. Hand gestures are actively used in human-computer interaction in applications spanning sign-language recognition, virtual manipulation, daily assistance, gaming, and human-robot interaction [6]. Low latency is an important factor in gesture-recognition systems as perceptually smooth interactions require systems to respond within $100-200$ ms [3, 25], allowing 30 fps camera systems only a few frames to operate. Conventional cameras can suffer from various motion-related artifacts (motion blur, rolling shutter, etc.) which may impact performance for rapid gestures. Challenging lighting conditions constitute another confounding variable, and are typically addressed by fusing data from different sensing modalities [32], increasing power consumption. The system presented in this paper offers a solution to these problems.

This paper describes a low-power, real-time, event-based hand gesture recognition system. Events from a Dynamic Vision Sensor (DVS) [22] are passed to a deep convolutional neural network (CNN) running on TrueNorth – an asynchronous, low-power, event-based neuromorphic processor [24] which achieves significant power and speed advantages over mobile GPUs [29]. TrueNorth output events indicate the recognized gesture, producing 1000 classifications per second with an average 105 ms latency from the start of a gesture.

The contribution of this paper is twofold. First, a gesture recognition system is implemented on event-based hardware that operates on live event streams in real-time. Second, a new hand-gesture dataset is collected with an event-based camera.

2. Related work

Real-time gesture recognition systems are varied in the hardware and algorithms used for gesture classification and localization. [35] provides a recent review of the algorithms for RGB and RGB-D based hand gesture recognition.

Non-event-based gesture and action recognition systems rely on either handcrafted features or learned features. Handcrafted feature extraction consists of a feature detection stage followed by a feature description stage [46]. Spatio-temporal feature detectors such as Harris3D [19], Cuboid [8] and Hessian3D [45] are typically used to localize interesting video keypoints, from which feature descriptors are extracted [38, 17, 43, 34]. However, it is generally acknowledged that there is no single optimal handcrafted feature [44]. Convolutional neural networks (CNNs) have been applied successfully to hand gesture recognition and localization [27, 32, 18, 40, 14], as have recurrent neural networks [28]. Multimodal systems [46, 31, 33, 15] have also been used to improve performance.

Figure 1: Picture of the setup. The DVS128 camera is connected to the NS1e board using a USB cable. The board exhibits a power cord and an Ethernet cable.

Work on low-power gesture recognition systems centers on developing either energy-efficient cameras or "smart" systems that save energy by optimizing power usage with traditional hardware. [5] achieved a real-time low-power gesture recognition system by reducing the frame rates and putting the sensor in standby mode to reduce the power requirements. Such efforts focus on reducing energy for frame-based gesture recognition, but these systems face a latency-power tradeoff wherein the low latencies required for gesture interaction can only be achieved by high sample rates that continuously burn higher power.

While there is a large body of work on hand gesture recognition, including many real-time systems [26, 47, 30], and a growing number of publications on event-based sensors like the DAVIS [2], ATIS [36], and DVS [22], there is very little work in the intersection of the two. [20] were the first to show an event-based gesture recognition system with a Dynamic Vision Sensor (DVS) and to show a postprocessing step with leaky-integrate-and-fire neurons (LIF), although this system can be further improved in performance and scalability. The event-based SpiNNaker processor has been used to run a 5-layer CNN on prerecorded DVS events to recognize playing cards [39], but we know of no prior work that combines an event-based sensor with an event-based processor to perform gesture recognition in real-time.

3. Event-based hardware

Our event-based gesture recognition system uses two devices: a *DVS128 camera* to generate input events, and a *TrueNorth processor* to inspect the input event stream

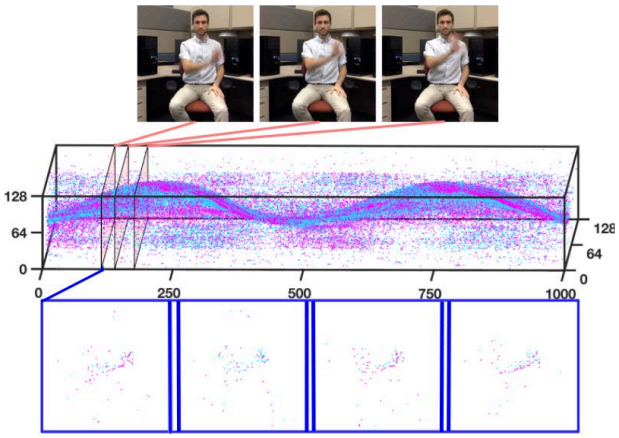

Figure 2: The system described in this paper operates on the data shown in the final row. Frame-based and event-based camera output for a large circle gesture. **Top:** 24-bit RGB video frames taken at 30 fps. Note the motion blur. **Middle:** Positive (magenta) and negative (cyan) DVS events over time. Red shaded planes indicate the times at which frames were sampled. Note the dense sampling of the DVS events compared to the RGB frames. **Bottom:** 4 ms of DVS event data displayed as 1 ms image slices. The first image is aligned to the first RGB video frame.

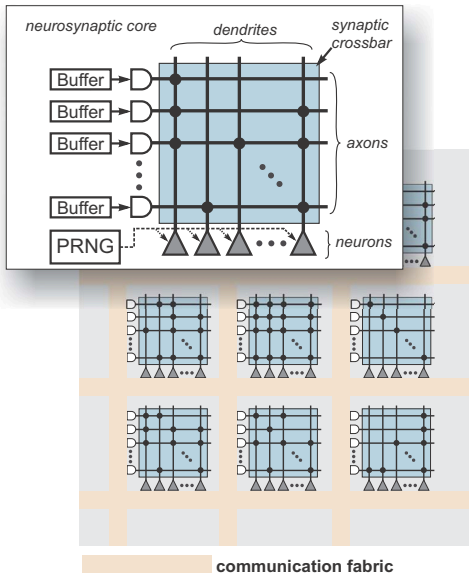

Figure 3: TrueNorth architecture. Neurosynaptic cores connect inputs to neurons using a synaptic crossbar. An asynchronous communication fabric routes events between cores.

for recognized gestures. Here, the TrueNorth processor is hosted on an *NS1e development board* that receives input events from the DVS128 via USB 2.0, and sends output events via Ethernet to a laptop for visualization (Fig. 1).

3.1. DVS128 camera

The iniLabs DVS128 camera is a 128×128-pixel Dynamic Vision Sensor that generates events only when a pixel value changes magnitude by a user-tunable threshold [21, 22] (this work used the default settings provided by the cAER configuration software [23]). Each event encodes the spatial coordinates of a pixel reporting a change and a timestamp indicating when that change happened. The device offers a high dynamic range (120 dB) and a typical and maximum event rate of 100K and 1M events/sec.

3.2. TrueNorth processor

The IBM TrueNorth chip is a reconfigurable, non-von Neumann processor containing 1 million spiking neurons and 256 million synapses distributed across 4096 parallel, event-driven, neurosynaptic cores [24]. Cores are tiled in a 64×64 array, embedded in a fully asynchronous network-on-chip. Under normal workloads, the chip consumes 70 mW when operating at a 1 ms computation tick.

Each neurosynaptic core connects 256 inputs to 256 neurons using a crossbar of 256×256 synapses with about 2 bits of weight precision. A neuron state variable called a membrane potential integrates synaptically weighted input events with an optional leak decay. Neurons can be configured to either generate an output event deterministically, whenever the membrane potential exceeds a threshold; or stochastically, with a pseudorandom probability related to the difference between the membrane potential and its threshold [4].

Each neuron can send output events to exactly one core in the array, where they can be delivered to any or all of the core's neurons via the synaptic crossbar. Every core input has a delay buffer that can cache incoming events for up to 15 ticks before releasing them into the crossbar.

Algorithms for TrueNorth must satisfy the constraints imposed by the architecture — neurons output binary events, not continuous values; synapses have low precision, not high precision; connectivity is core-to-core, not all-to-all. TrueNorth programs are written in the Corelet Programming Language, a hierarchical, compositional, object-oriented language implemented in MATLAB [1].

3.3. NS1e development board

The NS1e development board is a modular and compact platform for mobile embedded application development using a single TrueNorth processor. The TrueNorth chip is responsible for about 6% of board power consumption, which

is typically around 2 or 3 W. The board has an area of 125×69 mm^2 and weighs 98 g.

A Xilinx Zynq Z-7020 system-on-chip provides the interface between the TrueNorth chip and an assortment of onboard sensors and connectors. Two ARM Cortex-A9 cores run the Linux operating system and the software stack for streaming events between TrueNorth and standard interfaces, including USB and Ethernet. An FPGA fabric handles data conversion and address translation. Although not used in this work, the NS1e includes connectors that support direct, pin-to-pin access to the TrueNorth chip, which permits an event-based sensor like the DVS to stream events directly into TrueNorth, bypassing the Zynq SoC entirely.

4. Gesture recognition on TrueNorth

Our gesture recognition algorithm runs entirely on TrueNorth, and has four major components (Fig. 4). First, a *temporal filter cascade* captures a sequence of snapshots of the DVS event stream. Second, the concatenated snapshots are presented as input features to a stack of *convolution layers* trained offline using GPU acceleration. Third, a *winner-take-all decoder* identifies the gesture with the highest response from the final convolution layer. Finally, the resulting stream of instantaneous gesture classifications is cleaned up by a *sliding window filter*.

The entire software workflow described in this section, including source code, has been released as a reference example included in the TrueNorth developer toolkit.

4.1. Temporal filter cascade

To capture the sequence information required to encode gesture identity, a cascade of K delayed temporal filters collects a sequence of DVS events as they enter TrueNorth. The first filter outputs a stream of events delayed by one tick, and creates a second copy of its input which is passed to the next filter in the cascade. Each subsequent filter caches its incoming events in the delay buffers that are attached to each of a core's 256 input axons, accumulating 16 ticks of delay per stage, for a total of $1+16(K-1)$ ms. The output event streams of all K filters are concatenated to form the input features for the first convolution layer. The neurons in these filters are configured to generate events stochastically, using a constant leak to decay the membrane potential linearly with time. Using a stochastic decay makes the rate (or probability) of filter output events proportional to the time since the corresponding event was received at the filter input (after delay). This "short term memory" gradually forgets its input events as time elapses.

The temporal filter cascade may be compared to stacking frames to create a spatio-temporal input to CNN, such as in [14], or the temporal channel in [10], although with frames the temporal history usually does not decay stochastically.

4.2. Convolution layers

A convolutional neural network (CNN) is a multilayer feedforward network whose layers are neurons that collectively perform a convolutional filtering of the input or a prior layer (Fig. 5). Neurons within a layer are arranged in two spatial dimensions, corresponding to shifts in the convolution filter, and one feature dimension, corresponding to different filters.

CNNs are mapped to TrueNorth using the *Energy-efficient deep networks* (Eedn) algorithm [9], implemented in MATLAB using the MatConvNet library to enable GPU-accelerated training [42]. The Eedn algorithm satisfies the TrueNorth architecture constraints by restricting network precision to binary neuron output and trinary weights $\{-1, 0, 1\}$, and by limiting neuron fan-in and fan-out.

1. *Binary neuron output*: Instead of the standard rectified linear unit (ReLU) activation function, which produces multivalued neuron output, Eedn uses a binary step function that can be implemented by TrueNorth neurons operating as threshold logic units with integer biases. The derivative of a step function is a delta function, which is not finite as required by backpropagation, so it is approximated with a triangle function $\frac{\partial y}{\partial r} \approx \max(0, 1-|r|)$ where r is the filter response and y is the neuron output. Filter output is computed using batch normalization [13] during training, with the batch normalization parameters rolled into the neuron threshold for deployment.

2. *Trinary weights*: Trinary weights w are trained offline using an adaptation of the standard backpropagation algorithm. The trinary weights are used during the forward and backpropagation passes. However, the resulting weight updates are applied to a shadow network of high-precision proxy weights w_h. Trinary weights are updated by rounding with hysteresis their corresponding shadow weights:

$$w(t) = \begin{cases} -1 & \text{if } w_h(t) \le -0.5 - h, \\ 0 & \text{if } |w_h(t)| \le 0.5 - h \\ 1 & \text{if } w_h(t) \ge 0.5 + h, \\ w(t-1) & \text{otherwise,} \end{cases}$$

   where h is a hysteresis parameter set to 0.1.

3. *Neuron fan-in*: Group constraints are used to accommodate the limited fan-in constraint of 128 inputs per neuron (while TrueNorth allows 256 inputs per neuron, two inputs are used per synapse to allow trinary weights). Specifically, in a layer which uses G groups, each neuron receives connections from only N/G features from the N features in the source layer. For example, a convolution layer with a 1×1 kernel receiving

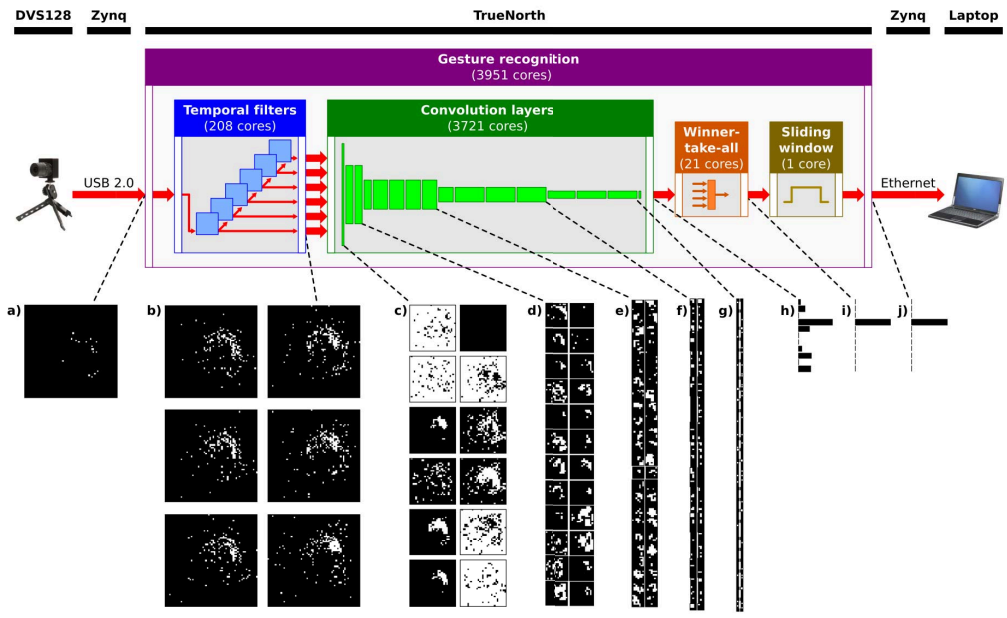

Figure 4: System block diagram showing how many of the 4096 TrueNorth cores are allocated to each component (for one of the experiments); examples of instantaneous event maps output by a) the DVS, b) the temporal filter cascade, and c–g) layers 1, 3, 8, 12, and 15 of the CNN described in Table 1; and histograms of events corresponding to the 11 gesture categories after h) the final convolution layer, i) the winner-take-all decoder, and j) the sliding window filter. All events are taken from the same 1 ms tick.

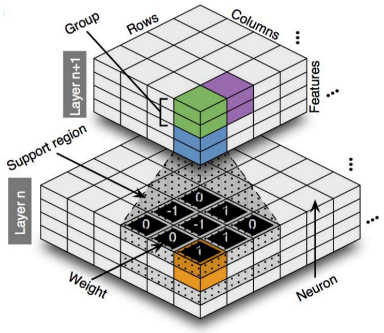

Figure 5: Two layers of a convolutional network, where each layer is a rows × columns × features collection of outputs from filters applied to the prior layer. Each output neuron has a topographically aligned filter support region in its source layer. Adjacent features have their receptive field shifted by the stride in the source layer. A layer can be divided into multiple groups along the feature dimension, where each group has a filter support region that covers a different set of features in the source layer. Two groups are highlighted (green, blue).

input from 256 features in the source layer must use 2 groups in order to reduce the fan-in to $1 \times 1 \times 256/2$ = 128 inputs per neuron. (The blue and green neurons in Fig. 5 are in two different groups.)

4. *Neuron fan-out*: TrueNorth neurons can target a single core. Therefore neuron copies are used to provide multiple neuron outputs for the weight representation scheme, and where filter overlap necessitates targeting multiple cores. A neuron is copied by replicating its parameters using free neurons on its own core, or by using splitter neurons on additional cores.

Each group of neurons in a layer at a given row and column in the feature map (i.e. boxes of the same color in Layer $n + 1$ in Fig. 5) is mapped to a separate TrueNorth core. Each core can compute up to 256 features from one location in a given feature map.

Despite these constraints, Eedn networks running on TrueNorth approached state-of-art classification accuracy on 8 image and audio datasets while achieving a peak throughput of 1200–2600 frames per second and consuming only 25–275 mW [9].

4.3. Winner-take-all decoder

A winner-take-all (WTA) decoder creates a single gesture prediction based on the population code that is output by the final convolution layer, which contains groups of output neurons for each class. The strength of a class is determined by the number of spikes output by its neurons in a tick. The decoder is a preconfigured neural network in which the largest-valued class neurons inhibit all the smaller-valued inputs and only the strongest class survives to be output as the winner.

4.4. Sliding window filter

The sliding window filter smooths the stream of instantaneous gesture classifications that are produced every millisecond by the winner-take-all decoder. Using a window of 80 ms, the sliding window filter improves out-of-sample classification accuracy from 91.77% coming out of the decoder to 94.59%. The filter effect is shown in Fig 6. For each class, independently, the filter counts the number of classification events observed within the window (up to 80), and spikes if the count exceeds a user-defined threshold. By using a threshold equal to 50% the window size, at most one class can spike. Such a filter is implemented using 8 neurons per class (one core for all classes). The sliding window filter adds 40 ms to the system latency, but preserves the 1 ms time resolution in determining the beginning and end time of the gesture. Shorter windows yield even lower latency, but might impair classification accuracy.

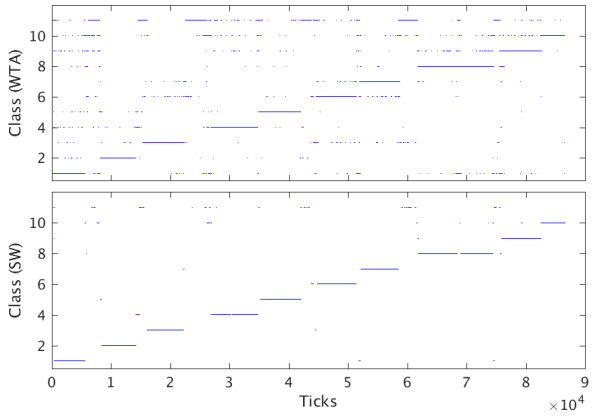

Figure 6: The winner-take-all output (WTA) and sliding-window filter (SW) showing an out-of-sample classification sequence of 11 gestures detected over 87214 ticks (milliseconds). The WTA and SWF can both produce up to one classification per tick, and in this case generate similar numbers of classifications, 75535 and 73133, respectively. Notably, the SWF output is about 2-4% more accurate.

5. Experiments

5.1. Dataset collection

A plethora of hand gesture datasets have been created in recent years, as thoroughly reviewed in [37]. Most of these datasets place subjects at a fixed distance from a single sensor – such as a Kinect, Wiimote, stereo camera, or regular color camera – that is typically frame-based. As a result, Hu et al. [12] reported an urgent need for DVS datasets in order to advance research into event-based computer vision. They converted four frame-based datasets to event-based representation by recording the output of a DVS camera pointed at a screen displaying the data. However, as noted in [41], the microsecond temporal resolution produced by a DVS camera pointed directly at the scene cannot be reproduced from the tens-of-milliseconds frame, and the converted data also contain additional undesired artifacts. Since the only extant DVS dataset for gesture recognition is not large enough to reliably train a CNN [20], we used the DVS128 to create a new hand gesture dataset that includes timestamped DVS128 event files, RGB videos from a webcam mounted next to the DVS, and ground-truth files with gesture labels and start and stop times.

The DvsGesture dataset comprises 1,342 instances of a set of 11 hand and arm gestures (Fig. 7), grouped in 122 trials collected from 29 subjects under 3 different lighting conditions. During each trial one subject stood against a stationary background and performed all 11 gestures sequentially under the same lighting condition. The gestures include hand waving (both arms), large straight arm rotations (both arms, clockwise and counterclockwise), forearm rolling (forward and backward), air guitar, air drums, and an "Other" gesture invented by the subject. The 3 lighting conditions are combinations of natural light, fluorescent light, and LED light, which were selected to control the effect of shadows and fluorescent light flicker on the DVS128. Each gesture lasts about 6 seconds.

To evaluate classifier performance, 23 subjects are designated as the training set, and the remaining 6 subjects are reserved for out-of-sample validation. The dataset is available at `http://research.ibm.com/dvsgesture/`.

5.2. Classifier training

In order to train the convolution layers on the appropriate signals, we first preprocessed the dataset by feeding the raw DVS events through the same temporal filter cascade that shapes the input to the CNN during runtime. The filter output events were written into a Lightning Memory-Mapped Database (LMDB), a high-performance database format preferred by many deep learning frameworks.

We trained a CNN on the preprocessed data using a GPU-accelerated implementation of the Eedn algorithm. The network structure (Table 1) was modeled after the best

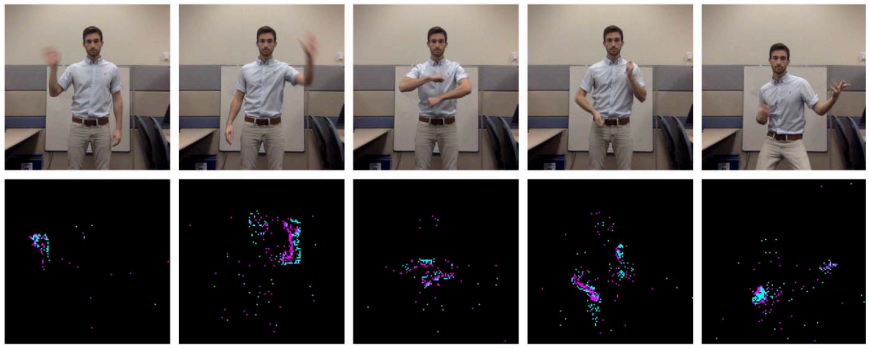

Figure 7: **Top:** 24-bit RGB video frames of the gestures used in this experiment. From left to right the gestures are hand-wave, large-circle, hand-role, air-guitar, and air-drums. For compactness, only one hand is shown for hand-wave and large-circle, and only one direction for large-circle. **Bottom:** DVS 'frames' created by superimposing spikes over a 5 ms window aligned to the onset of each RGB video frame.

	map size	feat -ures	kernel	stri-de	pad	groups
	64×64	6				
1	31×31	12	3×3	2	0	1
2	14×14	252	4×4	2	0	2
3	14×14	256	1×1	1	0	2
4	7×7	256	2×2	2	0	2
5	7×7	512	3×3	1	1	32
6	7×7	512	1×1	1	0	4
7	7×7	512	1×1	1	0	4
8	7×7	512	1×1	1	0	4
9	3×3	512	2×2	2	0	16
10	3×3	1024	3×3	1	1	64
11	3×3	024	1×1	1	0	8
12	3×3	1024	1×1	1	0	8
13	1×1	1024	2×2	2	0	32
14	1×1	1024	1×1	1	0	8
15	1×1	968	1×1	1	0	8
16	1×1	2640	1×1	1	0	8

Table 1: CNN specification for the gesture recognition system. All network layers are convolutional layers.

single-chip network in [9]. We trained the network for 250,000 iterations with a batch size of 256, using standard heuristics like momentum (0.9), weight decay (10^{-7}), and decreasing learning rate (the initial rate of 20 is divided by 10 twice during training at 200,000 and 225,000 iterations). We used a value of 0.0001 for an Eedn parameter called spike sparsity, which acts as a regularizer that encourages the trained network to produce fewer output spikes, conserving energy. To improve accuracy, we augmented the training data by randomly cropping a 64×64 region from larger images created by zero-padding the images with an 8-pixel-wide border region.

5.3. Power measurement

To measure the power consumed by the gesture recognition network running on TrueNorth, we run the network on a separate NS1t test and characterization board, which has dedicated circuitry to measure the power consumed by a TrueNorth chip. Power measurements are sampled at 62.5kHz, and averaged over a three-second DVS event sequence containing a full gesture. Boards operate at a supply voltage of $1.0\ V$. Total network power is the sum of leakage power, computed by scaling the measured idle power by the fraction of the chip's cores that are in use, and active power, computed by subtracting idle power from total power measured when the system is operating on input events.

5.4. Accuracy results

The results from 9 different training experiments are shown in Table 2. We took the CNN structure and all training metaparameters without modification from [9], and explored the impact of dataset preprocessing, varying the input image sampling (32×32, 42×42, and 64×64), and lengthening the temporal footprint of each sample by adding stages to the temporal filter cascade or increasing the decay period in each stage. We also compare the effect of augmenting the dataset with spatial translations up to one-eighth of the image size. For each experiment, we recorded the training and test accuracy of the CNN, the test accuracy of the system with and without the 'Other' category, the number of TrueNorth cores used, and the measured TrueNorth chip power.

The system from experiment E1 had the highest accuracy, a 10-category test score of 96.49%, while consuming only 178.8 mW. The sliding window filter improves the score by almost 5% over the raw winner-take-all output. Augmenting the dataset by spatially translating the frames improves classifier accuracy by about 1.5% (E1 vs E2).

In experiment E4 we decreased the number of time-delayed channels in the input compared with experiment E1. Six delayed channels gave better results for a 15 ms

Experiment	Input $W \times H$	Cascade filters	Filter duration (ticks)	Augment (pixels)	CNN train acc.	CNN test acc.	System acc. 11 cat.	System acc. 10 cat.	Neuro-synaptic cores	TN leak mW	TN active mW	TN total mW
E1	64×64	6	32	8	93.06	**91.77**	**94.59**	**96.49**	3838	134.3	44.5	178.8
E2	64×64	6	32	0	98.53	90.29	92.23	93.53	3838	134.3	44.5	178.8
E3	32×32	6	32	4	86.16	88.07	90.78	92.59	3459	122.5	46.5	169.0
E4	64×64	4	32	8	89.98	90.69	93.44	95.25	3625	127.0	45.0	172.1
E5	42×42	6	32	5	90.01	84.59	90.59	91.46	3968	140.2	55.6	195.8
E6	64×64	6	64	8	94.58	88.29	90.57	92.17	3951	138.7	44.2	182.9
E7	42×42	6	64	5	85.16	83.74	86.49	87.62	1936	68.8	19.7	88.5
E8	64×64	6	85	5	92.95	88.49	90.76	92.24	3951	139.0	37.2	176.2
E9	64×64	6	128	8	94.55	91.10	93.15	94.81	3951	142.8	49.7	192.5

Table 2: Classifier parameter exploration. Accuracy is reported in percents. Each network fits on a single TrueNorth chip (4096 cores).

delay between successive channels used here.

We experimented with downsampling to see how spatial resolution affects performance, while keeping the network sizes approximately the same by reducing the stride to 1 in the first convolutional layer (E3) or the 4th (E5). The best results were obtained with the largest image size, 64×64. 32×32 images resulted in a drop of 4% with approximately the same sized network. The 42×42 input in E5 gave the lowest accuracy, possibly due to the difference in network architecture. E7 had a stride of 1 in layers 1 and 9, reducing the network size and energy by half, but had the lowest accuracy of all the experiments.

Finally, experiments with the filter duration indicate that a decay of 32 ticks (E1) produced the best accuracy among the values explored, with 2-4% degradation for E6, E8, and E9 having 64, 85, and 128 ticks decay, respectively.

5.5. Latency measurement

Latency was measured on system E1. For this measurement, we aligned annotation and spike test data by removing all unlabeled events between gestures, as if user motion only occurred during a gesture. The latency to detect the start of a gesture is defined as $T_{start}(i) = t_{det}(i) - t_{start}(i)$, where $t_{start}(i)$ is the first tick in which events from the i^{th} gesture appear in the input stream and $t_{det}(i)$ is the first tick in which the gesture class is correctly recognized in the output stream. The search is limited to a narrow time window, from $t_{start}(i) + 1$ to $t_{start}(i) + 217$ ticks, representing the longest period an event can "travel" through the network, which is the sum of all maximal delays; stochastic decays; and CNN, WTA, and sliding window latencies. If there is no correct detection within this window, the gesture start is considered missed. A similar process applies to the gesture end. The average latency to detect the start and end of a gesture, over all 10 gestures in all 25 test sequences, is 104.6 ms (with 14 misses) and 120.6 ms (with 13 misses), respectively. The 16 ms difference might be due to the filter's stochastic decay, or "fading memory" after the gesture has ended, and is equal to half the decay time for E1.

5.6. User experience with the live system

The system was deployed with a live video feed of the recognized gesture on a computer screen. The system is very responsive, immediately detecting the gesture in most cases. A video of the gesture recognition system operating live can be found in Supplementary Information.

6. Discussion

The work reported in this paper unites a DVS camera with a TrueNorth processor to create an end-to-end event-based gesture recognition system. Leveraging recent advances in deep convolutional neural networks, it processes and classifies learnt gestures at 1000 Hz with high accuracy (96.49%), low latency (105 ms), and low power (178.8 mW), using a spiking neural network with low-precision weights running entirely on a TrueNorth processor. This homogeneous computational neuromorphic substrate stands in contrast to much of the other work in gesture recognition, which often utilizes frame-based sensors and multiple stages of processing – such as filtering, segmentation, tracking, 3D models, and more – prior to the classifier (CNN-based or otherwise). This observation suggests that such recognition tasks might require much less data and processing than traditional systems suggest.

7. Acknowledgments

The authors thank everyone who participated in the DvsGesture dataset collection. This research was sponsored by DARPA under contract No. HR0011-09-C-0002. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressly or implied, of DARPA or the U.S. Government.

References

[1] A. Amir, P. Datta, W. P. Risk, A. S. Cassidy, J. A. Kusnitz, S. K. Esser, A. Andreopoulos, T. M. Wong, M. Flickner, R. Alvarez-Icaza, et al. Cognitive computing programming paradigm: a corelet language for composing networks of neurosynaptic cores. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–10. IEEE, 2013.

[2] 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 Journal of Solid-State Circuits*, 49(10):2333–2341, 2014.

[3] S. K. Card, G. G. Robertson, and J. D. Mackinlay. The information visualizer, an information workspace. In *Proceedings of the SIGCHI Conference on Human factors in computing systems*, pages 181–186. ACM, 1991.

[4] A. S. Cassidy, P. Merolla, J. V. Arthur, S. K. Esser, B. Jackson, R. Alvarez-Icaza, P. Datta, J. Sawada, T. M. Wong, V. Feldman, et al. Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–10. IEEE, 2013.

[5] M. Chandra and B. Lall. Low power gesture recognition system using skin statistics for consumer applications. In *2016 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, pages 1–2. IEEE, 2016.

[6] H. Cheng, L. Yang, and Z. Liu. Survey on 3d hand gesture recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(9):1659–1673, Sept 2016.

[7] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014.

[8] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 65–72. IEEE, 2005.

[9] S. K. Esser, P. A. Merolla, J. V. Arthur, A. S. Cassidy, R. Appuswamy, A. Andreopoulos, D. J. Berg, J. L. McKinstry, T. Melano, D. R. Barch, C. di Nolfo, P. Datta, A. Amir, B. Taba, M. D. Flickner, and D. S. Modha. Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the National Academy of Sciences*, 113(41):11441–11446, 2016.

[10] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[11] S. Furber. Large-scale neuromorphic computing systems. *Journal of Neural Engineering*, 13(5):051001, 2016.

[12] Y. Hu, H. Liu, M. Pfeiffer, and T. Delbruck. Dvs benchmark datasets for object tracking, action recognition, and object recognition. *Frontiers in Neuroscience*, 10, 2016.

[13] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 448–456, 2015.

[14] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013.

[15] S. E. Kahou, C. Pal, X. Bouthillier, P. Froumenty, Ç. Gülçehre, R. Memisevic, P. Vincent, A. Courville, Y. Bengio, R. C. Ferrari, et al. Combining modality specific deep neural networks for emotion recognition in video. In *Proceedings of the 15th ACM on International conference on multimodal interaction*, pages 543–550. ACM, 2013.

[16] S. D. Kelly, S. M. Manning, and S. Rodak. Gesture gives a hand to language and learning: Perspectives from cognitive neuroscience, developmental psychology and education. *Language and Linguistics Compass*, 2(4):569–588, 2008.

[17] A. Klaser, M. Marszałek, and C. Schmid. A spatio-temporal descriptor based on 3d-gradients. In *BMVC 2008-19th British Machine Vision Conference*, pages 275–1. British Machine Vision Association, 2008.

[18] O. Koller, H. Ney, and R. Bowden. Deep hand: How to train a cnn on 1 million hand images when your data is continuous and weakly labelled. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3793–3802. Rheinisch-Westfalische Technische Hochschule Aachen, Aachen, Germany, 2016.

[19] I. Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2-3):107–123, 2005.

[20] J. H. Lee, T. Delbruck, M. Pfeiffer, P. K. Park, C.-W. Shin, H. Ryu, and B. C. Kang. Real-time gesture interface based on event-driven processing from stereo silicon retinas. *IEEE transactions on neural networks and learning systems*, 25(12):2250–2263, 2014.

[21] P. Lichtsteiner, C. Posch, and T. Delbruck. A 128 x 128 120db 30mw asynchronous vision sensor that responds to relative intensity change. In *2006 IEEE International Solid State Circuits Conference - Digest of Technical Papers*, pages 2060–2069, Feb 2006.

[22] P. Lichtsteiner, C. Posch, and T. Delbruck. A 128 x 128 120db 15μs latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566–576, Feb 2008.

[23] L. Longinotti and C. Brändli. *CAER: a Framework for Event-based Processing on Embedded Systems: Bachelor Thesis*. Institut für Informatik, 2014.

[24] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

[25] R. B. Miller. Response time in man-computer conversational transactions. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, pages 267–277. ACM, 1968.

[26] S. Mitra and T. Acharya. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(3):311–324, 2007.

[27] P. Molchanov, S. Gupta, K. Kim, and J. Kautz. Hand gesture recognition with 3d convolutional neural networks. In *Pro-*

ceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 1–7, 2015.

[28] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz. Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4207–4215, 2016.

[29] W. Murphy, M. Renz, and Q. Wu. Binary image classification using a neurosynaptic processor: A trade-off analysis. In *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on*, pages 1342–1345. IEEE, 2016.

[30] J. Nagi, F. Ducatelle, G. A. Di Caro, D. Cireşan, U. Meier, A. Giusti, F. Nagi, J. Schmidhuber, and L. M. Gambardella. Max-pooling convolutional neural networks for vision-based hand gesture recognition. In *Signal and Image Processing Applications (ICSIPA), 2011 IEEE International Conference on*, pages 342–347. IEEE, 2011.

[31] N. Neverova, C. Wolf, G. Taylor, and F. Nebout. Moddrop: adaptive multi-modal gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(8):1692–1706, 2016.

[32] N. Neverova, C. Wolf, G. W. Taylor, and F. Nebout. Multi-scale deep learning for gesture detection and localization. In *Workshop at the European Conference on Computer Vision*, pages 474–490. Springer, 2014.

[33] E. Ohn-Bar and M. M. Trivedi. Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations. *IEEE Transactions on Intelligent Transportation Systems*, 15(6):2368–2377, 2014.

[34] O. Oreifej and Z. Liu. Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 716–723, 2013.

[35] P. K. Pisharady and M. Saerbeck. Recent methods and databases in vision-based hand gesture recognition: A review. *Computer Vision and Image Understanding*, 141:152–165, 2015.

[36] 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 Journal of Solid-State Circuits*, 46(1):259–275, 2011.

[37] S. Ruffieux, D. Lalanne, E. Mugellini, and O. Abou Khaled. A survey of datasets for human gesture recognition. In M. Kurosu, editor, *Human-Computer Interaction. Advanced Interaction Modalities and Techniques: 16th International Conference, HCI International 2014, Heraklion, Crete, Greece, June 22-27, 2014, Proceedings, Part II*, pages 337–348. Springer International Publishing, Cham, 2014.

[38] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 357–360. ACM, 2007.

[39] T. Serrano-Gotarredona, B. Linares-Barranco, F. Galluppi, L. Plana, and S. Furber. Convnets experiments on spinnaker. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2405–2408. IEEE, 2015.

[40] A. Sinha, C. Choi, and K. Ramani. Deephand: robust hand pose estimation by completing a matrix imputed with deep features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4150–4158, 2016.

[41] C. Tan, S. Lallee, and G. Orchard. Benchmarking neuromorphic vision: lessons learnt from computer vision. *Frontiers in neuroscience*, 9, 2015.

[42] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. In *Proceeding of the ACM Int. Conf. on Multimedia*, 2015.

[43] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for action recognition. *International journal of computer vision*, 103(1):60–79, 2013.

[44] H. Wang, M. M. Ullah, A. Klaser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. In *BMVC 2009-British Machine Vision Conference*, pages 124–1. BMVA Press, 2009.

[45] G. Willems, T. Tuytelaars, and L. Van Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *European conference on computer vision*, pages 650–663. Springer, 2008.

[46] D. Wu, L. Pigou, P. Kindermans, N. LE, L. Shao, J. Dambre, and J. Odobez. Deep dynamic neural networks for multimodal gesture segmentation and recognition. *IEEE transactions on pattern analysis and machine intelligence*, 2016.

[47] X. Zabulis, H. Baltzakis, and A. Argyros. Vision-based hand gesture recognition for human-computer interaction. *The Universal Access Handbook. LEA*, pages 34–1, 2009.