Event-based incremental broad learning system for object classification

Shan Gao, Guangqian Guo and C. L. Philip Chen, *Fellow, IEEE*
Northwestern Polytechnical University, China
gaoshan@nwpu.edu.cn, guogq01@163.com, philip.chen@ieee.org

Abstract

Event cameras are bio-inspired vision sensors that output pixel-level brightness changes instead of RGB values. Thousands of convolutional neural networks have emerged to process the frame-based images; however, there are few networks designed explicitly for the event-based data, which can fully take advantages of the asynchronous and high-temporal resolution data. In this paper, we propose an incremental broad learning system to learn the event-based data in a flat network structure, which consists of feature nodes and enhancement nodes in one layer. The incremental learning strategy is developed for fast adding new nodes in a broad extension, yet it is almost impossible to add a filter or layer in the CNNs without retraining from the beginning. An SVD operation is coupled with the network extension to prevent the redundancy of the network structure. In experiments, our model outperforms the state of the arts, at the same time, $15\times$ *faster than the CNNs in training. It makes event cameras easier to be the nearly online training and inference applications.*

1. Introduction

Processing the frame-based images and videos dominates computer vision for several decades, and the concept 'frame' has rooted not only in computer vision researchers but also the general public's minds. With the domination of the Convolutional Neural Networks (CNNs) [9, 23, 36, 16, 17] in recent years, people are more convinced that images and videos should exist in the convolution operation and deep structure. However, rethinking the frame-based images computer vision, it mainly faces three challenges [25]. The first challenge is that absolute pixel illumination is the main source of frame-based cameras, but illumination is not an invariant property of a scene [28]. Measuring luminance accurately is limited by the low dynamic range of conventional cameras [35]. The second challenge is that real-time operation implies a minimum of

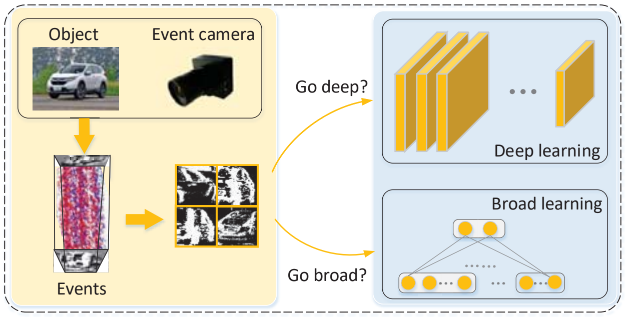


Figure 1. The motivation of this work. The deep learning methods are more applicable for the frame-based synchronous data, but which kind of networks is better for the event-based asynchronous data. In this paper, we propose using a broad learning system to handle the event data in an incremental way.

24 images per second. However, biological retinas operate at the temporal precision of 1 kHz [14]. It has been shown that there is a loss of 75 percent of valuable information leading to a poor separability between classes of objects [1] for the low temporal rates cameras (30-60 Hz). The third challenge is that the images measuring at unnatural frame-rate results in an acquisition of huge amounts of redundant data because most pixels do not change from one frame to the next. The massive redundancy in frames will be further added into the video compression.

In the meanwhile, neuromorphic cameras [2, 33, 40] are becoming more and more widespread. These devices are bio-inspired vision sensors that attempt to emulate the functioning of biological retinas, namely event cameras, such as the Asynchronous Time-based Image Sensor (ATIS) [33] and Dynamic Vision System (DVS) [10], are fundamentally different from traditional cameras that output a sequence of frames at fixed intervals. The term 'event' refers to a spike output that is characterized by a spatial location (x, y), timestamp (t) and polarity of the brightness change (p), shown in Fig. 1. Thus, the output of an event camera is a stream of asynchronous spikes that are triggered by brightness changes sensed by individual pixels. Several models have been proposed to utilize the event-based vision sensors for the object classification, *e.g.*, CNNs [19], Hierarchical

*Guangqian Guo is an intern student in Unmanned System Research Institute at Northwestern Polytechnical University.

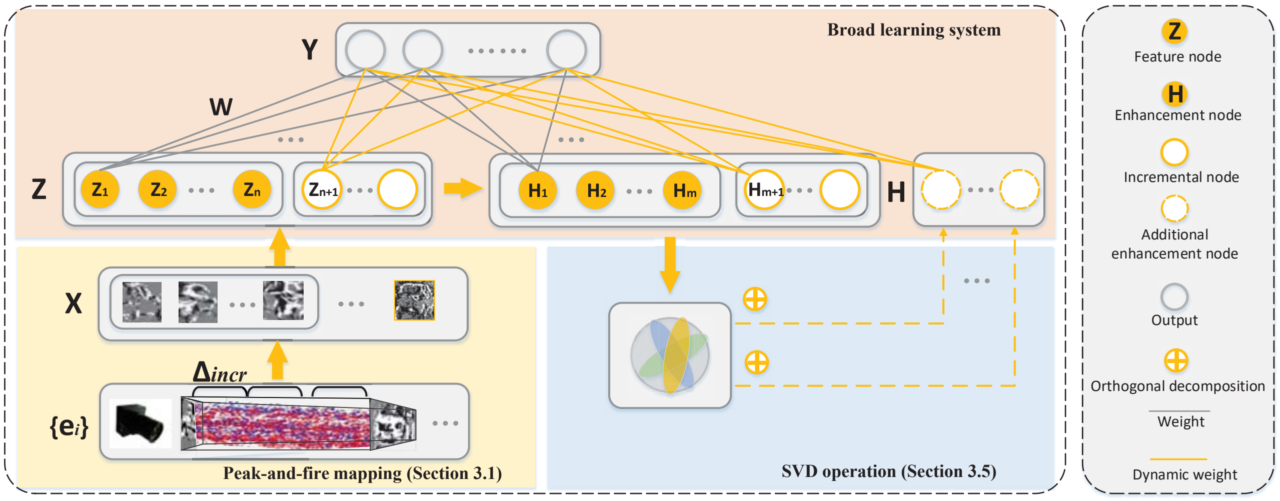


Figure 2. Overview of the proposed event-based incremental broad learning system. The network consists of feature node Z, enhancement node & additional enhancement nodes H, and output nodes Y. First, the event-based data $\{e_i\}$ are computed by the peak-and-fire mapping as input X, which then is mapped as Z. The enhancement node H is generated by the feature node and activation function. With the input data X increase, the network extends in the broad dimension by adding more feature nodes and enhancement nodes. An SVD operation is designed to decrease the redundancy during the network extension.

like models [12], and Deep Belief Networks (DBNs) [32]. Other methods start to explore the integration of dynamical information in recognition task by using motion-direction sensitive units [18] or dynamical networks (like Echo-state networks) [24]. However, the above networks origin from the frame-based methods, in which the event data is considered as a similar frame-based image in feature extracting, and the networks are trained in the old fashion, which do not take the advantages of bio-inspired event-based data.

In this paper, we design a broad learning network to deal with the event-based data for the object classification. We firstly use an asynchronous peak-and-fire mapping to depict the event-based data. Then a basic broad learning system (BLS) [7] is established in the form of a flat network, where the event-based inputs are transferred as 'feature nodes' and the structure is expanded as 'enhancement nodes'. The output layer is directly connected with the feature nodes and enhancement nodes by the weights, which is shown in Fig. 2. The broad network provides an alternative way of learning in a go-broad way, which is different from the deep C-NNs models. With the event-based input data continuously coming, the network becomes broad by adding feature nodes, enhancement nodes, and additional enhancement n-odes. In a traditional deep structure, if one needs adding layers to describe more detailed features, the whole network usually should be retrained from the beginning. It will suffer from a time-consuming training process because of a large number of connecting parameters in filters and between layers. However, our incremental BLS can be remodeled by the increment of nodes without the entire retraining.

As far as we know, we are the first to propose using the incremental broad learning way to deal with the event-based data. The key contributions of this paper are:

1. A BLS bridges the event-based data and the broad learning, which successfully integrates the asynchronous data into a flexible broad network.

2. An incremental learning strategy is easily extended in feature nodes, enhancement nodes, and the input data, which facilitates the BLS network extending from a basic network to a large one.

2. Related work

Event-based feature & Object Classification. The majority of prior work on event-based object detection focused on detecting and tracking stable features. The application includes: simultaneous localization and mapping applications [20, 34], corner detectors [43, 30], edge and line extraction [39], event-based flow [8]. [25] proposed a hierarchical representation based on the definition of time surface and clustering the time surfaces at each layer, while the last layer sent its output to a classifier. The main limitation of this method was high latency due to the increasing time window needed to compute the time surfaces and the high computational cost of the clustering algorithm. Then a compact and fast representation of [25] was proposed in [42]. However, these methods focused on extracting the accurate hand-designed feature from the event data, but the classifiers or networks were not considered in a whole framework with the asynchronous data.

Event-based networks. The networks used in event-based data origin from training artificial neural networks by reproducing or imitating the learning rules observed in biological neural networks [15, 3, 41, 29], which was similar to what was done in frame-based computer vision, try to optimize the weights of networks by minimizing a smooth error function. The most commonly used architectures for event-based cameras were Spiking Neural Networks (SNN) [6, 37, 38, 44, 31]. However, it was difficult to train an SNN with the gradient descent properly. To avoid this, [12] used predefined Gabor filters as weights in the network. Others proposed to train a CNN and then to convert the weights to an SNN [6, 37]. [5] added the attention mechanism in an event-based YOLO structure, namely YOLE, to solve the event-based classification problem. Unfortunately, when the network converted from the CNNs structure, the performance was lower than conventional CNNs on frames.

Broad learning system. BLS originated from the functional link neural network [21], which was a variant of the higher-order neural network without hidden units. BLS was developed by Chen and Liu [7], which was a fast and efficient discriminative learning method. Without stacking the layer-structure, the designed neural networks expanded the neural nodes broadly and updated the weights of the neural networks incrementally when additional nodes were needed and when the input data were entering the neural networks continuously. Therefore, the BLS structure was suitable for modeling and learning in a time-variant environment.

3. Methodology

This section consists of four parts. First, we introduce a peak-and-fire mapping, which converts the event-based data to a feature map. Second, we input the mapped feature to a broad learning network. Third, we propose extending the basic BLS by incrementally adding more nodes in a broad way. Finally, an SVD operation is introduced to decrease the network redundancy in broad extension.

3.1. Peak-and-fire Mapping

Given an event-based sensor with pixel grid size $M * N$, a stream of events is denoted by a sequence

$$e_i = [l_i, t_i, p_i]^T, \tag{1}$$

where e_i is the ith event and consists of a location ($l_i = [x_i, y_i]^T$), time (t_i) and polarity (p_i), with $p_i \in \{-1, 1\}$, where -1 and 1 represent OFF and ON events, respectively. When an object (or the camera) moves, the pixels asynchronously generate events which form a spatio-temporal point cloud representing the objects spatial distribution and dynamical behavior.

Peak-and-fire mechanism. Inspired by the SNN, we design a peak-and-fire mechanism to detect peaks of the event array and fire the peaks as outputs. First, we identify and localize the event peak as

$$P_{e_i} = \begin{cases} \sum_{e_j} exp(\alpha \cdot \Delta t) & if \;\; t_j - \Delta_{incr} \leq t_i, l_j = l_i \\ 0 & otherwise \end{cases} \tag{2}$$

where P_{e_i} provides a dynamic temporal context for the event e_i. The exponential decay expands the activity of passed events and records information about the history of the activity. $\Delta t = t_j - t_i$. α is the decay parameter. Δ_{incr} is defined as a time interval. Considering the polarity of each event, we only sum the the event in the same polarity, as $p_j = p_i$.

Because the structure of this point cloud contains information about the object and its movement, we introduce cell c_k to keep track of the activity surrounding the pixel location l_i, which is inspired by [25, 42]. The cell c_k defines an incoming event e_i as the array of most recent events in the $\{\mathbb{C} = R * R\}$ square neighborhood centered at l_i, $c_k(e_i) = \{e_j : t_j - \Delta_{incr} \leq t_i, l_j = (l_i + \mathbb{C})\}$. Then, a peak in the cell is defined as

$$P_{c_k} = \frac{1}{|N_{c_k}|} \sum_{e_i \in \mathbb{C}} P_{e_i}, \tag{3}$$

where N_{c_k} is the number of events in cell c_k. A peak in a cell is considered to be valid if its number is more than the confidence threshold in the interval Δ_{incr}. The threshold is defined as μ^t, where $\mu^t = N_{c_k}/size(c_k)$ and $size(c_k)$ is the size of the cell c_k. if $N_{e_i} > \mu^t$, the peak value in a cell c_k is fired as an output, where N_{e_i} is the number of the events in l_i. It promises the output peak value is able to eliminate the influence of noises.

Peak memory cell. When we use the peak-and-fire mechanism in the incoming time interval Δ_{incr}, every incoming event e_i need be iterated over all events in a past cell. Looping through the entire ordered event stream is extremely expensive and inefficient. We define a shared memory cell M_c for every cell c_k. The past events relevant for c_k are stored. The output of c_k is defined as

$$X_{c_k}^{t+\Delta_{incr}} = \begin{cases} P_{c_k}^t(R,p) + M_{c_i}(\Delta_{incr}) & if \; e_i \in c_k \\ P_{c_k}^t(R,p) & otherwise \end{cases} \tag{4}$$

when a new event arrives in c_k, we update Eq. (4) by only looping through M_{c_k}, which contains only the relevant past events to compute the peak memory cell. After each interval, the output is $X = [X_{c_1}, \ldots, X_{c_K}]$. Hence, we can compute a robust feature representation without a significant increase in memory requirements.

3.2. Broad learning system

To build our incremental learning framework, we start with a traditional, *i.e.*, non-incremental BLS for the object classification task. Our model is based on the BLS

Algorithm 1: Peak-and-fire Mapping

```
Input: Events $\{e_i\}$, Parameters: $\Delta_{incr}$, $R$, $\alpha$
Output: $X$
Initialize: $P_{e_i} = 0$, $|N_{c_k}| = 0$, and $M_{c_i} = 0$;
for $i = 1; i \leq I$ do
    $P_{e_i} \leftarrow$ event peak detection;
    $P_{c_k} \leftarrow$ cell peak detection;
    $\mu^t \leftarrow$ calculate fire threshold;
    $M_{c_i} \leftarrow$ memory cell update;
end
Return $X = [X_{c_1}, X_{c_2}, \ldots, X_{c_K}]$
```

[7] to provide an effective and efficient learning framework for classification and regression problems. Now let us review the BLS mathematically, given the training dataset $\{(X, Y)|X \in \mathbb{R}^{N\times K}, Y \in \mathbb{R}^{N\times C}\}$ from C classes. Here, X denotes the event data presentation in time interval Δ_{incr} described in Sec. 3.1. In a BLS, the training samples are first transformed into n random feature spaces by feature mapping ϕ_i as

$$Z_i \triangleq \phi_i(XW_{f_i} + \beta_{f_i}), i = 1, 2, \ldots, n, \tag{5}$$

where the weights W_{f_i} and the bias term β_{f_i} are generated randomly with the proper dimensions. Then we define the feature space of training samples as $Z^n \triangleq [Z_1, Z_2, ..., Z_n]$, a collection of n groups of feature nodes. The outputs of the jth group of enhancement nodes are defined by

$$H_j \triangleq \xi_j\left(Z^n W_{h_j} + \beta_{h_j}\right), j = 1, 2, \cdots, m, \tag{6}$$

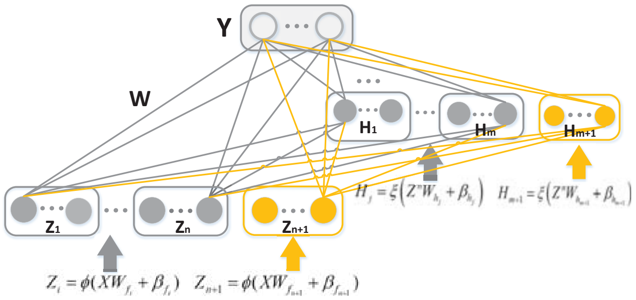
where ξ_j is a nonlinear activation function. W_{h_j} is the enhancement weights and β_{h_j} is the bias term. In practice, j and i of Z_i can be selected differently depending upon the complexity of the modeling tasks. Furthermore, ϕ_i and ξ_j can be different functions. Without loss of generality, the subscripts of the ith random mappings ϕ_i and the jth random mappings ξ_j are omitted in this paper.

We denote the outputs of the enhancement layer by $H^m \triangleq [H_1, H_2, ..., H_m]$. Therefore, the output $\hat{Y}$ of a BLS has the following form

$$\begin{aligned}\hat{Y} &= [Z_1, Z_2, \cdots, Z_n, H_1, H_2, \cdots, H_m]W \\ &= [Z^n, H^m]W \\ &= AW,\end{aligned} \tag{7}$$

where $A = [Z^n, H^m]$ denotes the transformation features, and W is the output weight connecting the feature nodes and enhancement nodes to the output layer. W should be optimized by solving the following minimization as

$$\min_W \|Y - AW\|_2^2 + \lambda \|W\|_2^2, \tag{8}$$


Y
W
H1
Hm
Hm+1
Z1
Zn
Zn+1
$H_j = \xi(Z^n W_{h_j} + \beta_{h_j})$
$H_{m+1} = \xi(Z^n W_{h_{m+1}} + \beta_{h_{m+1}})$
$Z_i = \phi(XW_{f_i} + \beta_{f_i})$
$Z_{n+1} = \phi(XW_{f_{n+1}} + \beta_{f_{n+1}})$
Figure 3. Overview of the proposed event-based incremental broad learning system.

where λ is a small trade-off regularization parameter, the first term denotes the training errors and the second term controls the complexity of network structure and improve the generality. Then by setting the derivation of Eq. (8), we can get the solution of output weight as

$$W = (A^T A + \lambda I)^{-1} A^T Y. \tag{9}$$

The calculation of weights W can always be achieved, as the matrix $(A^T A + \lambda I)$ is generally nonsingular. Specifically, we have

$$A^+ = \lim_{\lambda \to 0} (A^T A + \lambda I)^{-1} A^T. \tag{10}$$

3.3. BLS Architecture

The BLS is constructed based on the flatted functional-link networks. Fig. 3 shows the network structure. We first map the inputs to construct a set of mapped features. We use the input data X and project the data, using Eq. (5), to become the ith mapped features, Z_i, where W_{f_i} is the random weights. Z^n is the concatenation of n groups of mapping features. Similarly, the jth group of enhancement nodes H_j is calculated by Eq. (6). H^m is the concatenation of m groups of enhancement nodes. Hence, the broad model can be represented as the Eq. (7). The weights W connect the feature nodes and enhancement nodes with the output nodes and can be computed through the ridge regression approximation A^+ using Eq. (10). The pseudoinverse computation is used here, which can also be replaced with an iterative algorithm or a gradient descent approach if desired.

So far, a general framework of the BLS is presented. Theoretically, the selection of functions for the feature mapping deserves attention. Functions $\phi(\cdot)$ and $\xi(\cdot)$ have no explicit restrictions, which means that common choices such as kernel mappings, nonlinear transformations, or convolutional functions are acceptable. Specifically, if we use the convolutional functions for the feature mapping, the BLS network structure is very similar to that of classical CNN structure except that the BLS network has additional connecting links between the convolutional layers and the output layer.

3.4. Incremental Broad Learning System

When new input event-based data X coming, the basic BLS network may not be good enough for learning. It may be caused by the insufficient nodes. We propose an incremental BLS to extend the network. Here we use two strategies: the first one is adding additional enhancement nodes, and the second one is adding feature nodes and corresponding enhancement nodes.

Increment of additional enhancement nodes. First, we detail the broad expansion method for adding additional enhancement nodes. Denote $A^m = [Z^n | H^m]$ and A^{m+1} as

$$A^{m+1} = [A^m | \xi(Z^n W_{h_{m+1}} + \beta_{h_{m+1}})], \tag{11}$$

where $W_{h_{m+1}} \in \mathbb{R}^{nk \times p}$, and $\beta_{h_{m+1}} \in \mathbb{R}^p$. The connecting weights and biases from mapped features to the additional enhancement nodes are randomly generated. We could deduce the pseudoinverse of the new matrix as

$$(A^{m+1})^+ = \begin{bmatrix} (A^m)^+ - DB^T \\ B^T \end{bmatrix}, \tag{12}$$

where $D = (A^m)^+ \xi(Z^n W_{h_{m+1}} + \beta_{hm+1})$,

$$B^T = \begin{cases} (G)^+ & if\ G \neq 0 \\ (1 + D^T D)^{-1} B^T (A^m)^+ & if\ G = 0 \end{cases} \tag{13}$$

and $G = \xi(Z^n W_{h_{m+1}} + \beta_{h_{m+1}}) - A^m D$. The new updated weights are

$$W^{m+1} = \begin{bmatrix} W^m - DB^T Y \\ B^T Y \end{bmatrix}. \tag{14}$$

Notice that all the pseudoinverse of the involved matrix are calculated by the regularization in Sec. 3.2. Specifically, this algorithm only needs to compute the pseudoinverse of the additional enhancement nodes instead of computations of the entire (A^{m+1}) and thus results in fast incremental learning.

Increment of feature nodes and enhancement nodes. Now let us come to the cases that the input data keep entering. Denote X_a as the new inputs added into the network, and denote A_n^m as the n groups of feature nodes and m groups of enhancement nodes of the initial network. The respectively increment of feature nodes and enhancement nodes are formulated as follows:

$$\begin{aligned} A_x = [&\phi(X_a W_{f_1} + \beta_{f_1}), \dots, \phi(X_a W_{f_n} + \beta_{f_n}) | \\ &\xi(Z_x^n W_{h_1} + \beta_{h_1}), \dots, \xi(Z_x^n W_{h_m} + \beta_{h_m})], \end{aligned} \tag{15}$$

where $Z_x^n = \phi(X_a W_{f_1} + \beta_{f_1}), \dots, \phi(X_a W_{f_n} + \beta_{f_n})$ is the group of the incremental feature nodes updated by X_a.

The detailed proving process is in supplementary materials

Algorithm 2: Event-based Incremental BLS Learning

```
Input: training samples $X$
Output: $W$
for $i = 1; i \leq n$ do
    Random $W_{f_i}$, $\beta_{f_i}$;
    Calculate $Z_i$ by Eq. (5);
end
for $j = 1; j \leq m$ do
    Random $W_{h_j}$, $\beta_{h_j}$;
    Calculate $H_j$ by Eq. (6);
end
Set $A_n^m$ and calculate $(A_n^m)^+$ by Eq. (10);
while The training error threshold is not satisfied do
    if additional enhancement nodes are added then
        Random $W_{h_{m+1}}$, $\beta_{h_{m+1}}$;
        Calculate $H_{m+1}$;
        Update $A_n^{m+1}$;
        Calculate $(A_n^{m+1})^+$ and $W_n^{m+1}$ by Eqs. (12, 14);
        $m = m + 1$;
    end
    else
        New inputs are added as $X_a$;
        Calculate $A_x$ by Eq. (15);
        Update ${}^x A_n^m$;
        Update $({}^x A_n^m)^+$ and ${}^x W_n^m$ by Eqs. (16, 18);
    end
end
```

The W_{f_i}, β_{f_i}, β_{h_j} are randomly generated during the initial of the network. Hence, we have the updating matrix ${}^x A_n^m = [A_n^m \quad A_x^T]^T$. The associated pseudoinverse updating algorithm could be deduced as follows:

$$({}^x A_n^m)^+ = [(A_n^m)^+ - BD^T | B], \tag{16}$$

where $D^T = A_x^T A_n^{m+}$,

$$B^T = \begin{cases} (G)^+ & if\ G \neq 0 \\ (1 + D^T D)^{-1} (A_n^m)^+ D & if\ G = 0 \end{cases} \tag{17}$$

and $G = A_x^T - D^T A_n^m$. Therefore the updated weights are

$${}^x W_n^m = W_n^m + (Y_a^T - A_x^T W_n^m) B, \tag{18}$$

where Y_a is the respective labels of additional X_a. The input nodes updating algorithm is shown in Alg. 2. Again, this incremental learning saves time for only computing necessary pseudoinverse. This particular scheme is perfect for incremental learning for new incoming event-based data.

3.5. SVD Operation

After the broad expansion with added feature nodes, enhancement nodes and additional enhancement nodes, the broad structure may have a risk of being redundant due to

	H-First[12]	HOTS[25]	HATS[42]	Gabor-SNN[42]	EBLS	Ours(full)
N-MNIST	0.712	0.808	**0.991**	0.837	0.981	0.983
N-Caltech101	0.054	0.210	0.642	0.196	0.647	**0.668**
MNIST-DVS	0.595	0.803	0.984	0.824	0.980	**0.987**
CIFAR10-DVS	0.077	0.271	0.524	0.245	0.547	**0.563**
N-CARS	0.561	0.624	0.902	0.789	0.899	**0.931**

Table 1. Comparison of classification accuracy on five datasets. The highest classification rates are marked as the bold face.

poor initialization or redundancy in structure. In our incremental model, we use an SVD operation in three parts: feature nodes, enhancement nodes, and the broad expansion. First, we apply SVD to feature nodes Z_i, $i = 1, \dots, n$ as

$$\begin{aligned} Z_i &= U_{Z_i}\Sigma^P_{Z_i}V^T_{Z_i} \\ &= U_{Z_i}\cdot[\Sigma^P_{Z_i}|\Sigma^Q_{Z_i}]\cdot[V^P_{Z_i}|V^Q_{Z_i}] \\ &= U_{Z_i}\Sigma^P_{Z_i}V^{P\,T}_{Z_i} + U_{Z_i}\Sigma^Q_{Z_i}V^{Q\,T}_{Z_i} \\ &= Z^P_i + Z^Q_i, \end{aligned} \tag{19}$$

where Σ^P and Σ^Q are divided by the order of singularities, under the parameter ε_f. The idea is to compress Z_i by the principal portion, Z^P_i. The equation between Z_i and Z^P_i is derived as $Z^P_i V^P_{Z_i} = Z_i V^P_{Z_i}$. Before each iteration of SVD, we denote $A^0_n = [Z_1, \dots, Z_n]$, and the model weight is denoted as $W^0_n \triangleq [W^{\{0,n\}}_{Z_1}|\cdots|W^{\{0,n\}}_{Z_n}]^T$.

Similarly, the enhancement nodes H_m and the broad network structure could be orthogonal decomposed by the SVD operation with the threshold ε_h and ε. Generally, the number of feature nodes, enhancement nodes, and the final structure could be significantly reduced depending on the threshold values ε_f, ε_h, and ε, respectively.

4. Experiments

4.1. Network settings

We introduce the configuration of the proposed model in this section. For the feature mapping function $\phi_i(\cdot)$, we use a sparse autoencoder solved by ADMM [13]. For the enhancement nodes, the sigmoid function $\xi_i(\cdot)$ is chosen. The regularization parameter λ in Eq. (8) for ridge regression is set as 10^{-8}. The network weight W_{f_i} and the bias β_{f_i}, for $i = 1, \dots, n$ are drawn from the standard uniform distributions on the interval $[-1, 1]$. The thresholds in SVD operation are set as $\varepsilon_f = \varepsilon_h = \varepsilon = 0.8$. In the peak-and-fire mapping, the cell's length is $R = 7$ and the memory cell's length is also set as R. It makes the peak value easy to calculate in the same size. α is set as 10^{-6} in Eq. (2).

4.2. Evaluations

Datasets. We validate our approach on five different datasets: four datasets generated by converting standard frame-based datasets to events (namely, N-MNIST [11], N-Caltech101 [11], MNIST-DVS [40] and CIFAR10-DVS [27] datasets) and a novel dataset, recorded from real-world scenes, N-CARS dataset [42]. N-MNIST, N-Caltech101, MNIST-DVS, and CIFAR10-DVS are four publicly available datasets created by converting the popular frame-based MNIST [9], Caltech101 [26] and CIFAR10 [22] to an event-based representation. N-MNIST and N-Caltech101 were obtained by displaying each sample image on an LCD monitor, while an ATIS sensor was moving in front of it [11]. Similarly, the MNIST-DVS and CIFAR10-DVS datasets were created by displaying a moving image on a monitor and recorded with the ATIS camera [40]. N-Cars dataset is split in 7940 car and 7482 background training samples, and 4396 car and 4211 background testing samples. Every sample duration is 100ms. MNIST-DVS contains 10,000 samples, generated at three different resolutions, scale4, scale8, and scale16. We use 90% of the samples for training and 10% for testing in scale 4. The duration of a presentation around 2.3s. N-Caltech101 consists of 100 different object classes and a background class. The duration is approximately 300ms. In our experiments, we use two-thirds of the samples of each class for training and the rest for testing. We find that the sample duration is not the same, but in experiments, we use the $\Delta_{incr} = 100ms$ for all samples.

Baseline methods. We consider several published H-First [12], HOTS [25], HATS [42], and SNN [31, 38]. For H-First, we used the code provided by the authors online. For HOTS [25] and HATS [42], we use the results reported in [42]. They use a linear SVM for classification. Given that no code is available for the SNN, we compared our results with a two-layer SNN architecture using predefined Gabor filters [4]. The result is reported in [42]. For a fair comparison, we use our basic model, Event-based BLS (EBLS), which means we do not use the incremental learning and SVD operation in the EBLS model. Moreover, we add the CNN models as our comparisons: LeNet5 [9] and AlexNet [23]. In the experiments, we use our PFM output as an input to the LeNet5 and AlexNet models, since there is no frame-based data.

Results analysis. The results for the N-MNIST, N-Caltech101, MNIST-DVS, CIFAR10-DVS, and N-CARS datasets are given in Tab. 1. We report the results in terms of classification accuracy in different datasets compared with the ground truth. It is observed that our method has the highest classification accuracy in all the datasets. Our

Node	EBLS		Ours (Full)	
Num.	Structure	MTE	Structure	MTE
500	[100, 400]	5.63	[100,11000]→ 500	5.04
1000	[100, 900]	4.71	[100,11000]→1000	3.42
1500	[100,1400]	3.45	[100,11000]→1500	2.95
2000	[100,1900]	2.77	[100,11000]→2000	2.43
2500	[100,2400]	2.41	[100,11000]→2500	2.02
3000	[100,2900]	2.21	[100,11000]→3000	1.72

Table 2. Classification Minimal Test Error (MTE) (%) on the MNIST-DAVIS dataset with different nodes settings. The left column is the total nodes number. EBLS has a fixed structure for the one-shot test. Our full model uses the SVD operation to decrease the enhancement nodes from 11000 to specifical numbers.

Cell length	$R = 3$	$R = 7$	$R = 11$	$R = 15$
Accuracy	0.515	0.673	0.649	0.621

Table 3. Classification accuracy in different cell lengths on the N-Caltech101 dataset.

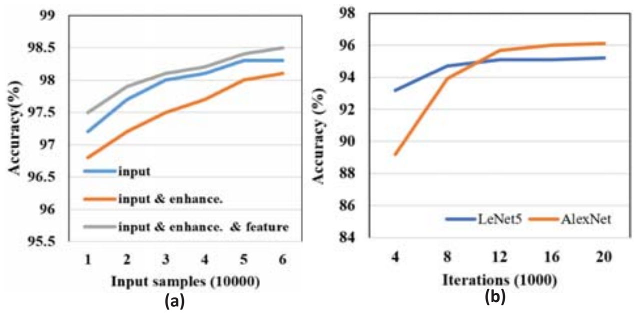


Figure 4. (a) The results of incremental learning on the N-MNIST dataset. The blue line denotes adding more training samples in a fixed network. The yellow one denotes adding input samples and enhancement nodes together. The grey one denotes adding input samples, feature nodes, and enhancement nodes together; (b) Classification accuracy of the increasing iterations of LeNet5 and AlexNet on MNIST-DAVIS dataset.

method achieves the large margin improvement in the more challenging datasets: N-Caltech101, CIFAR10-DVS, and N-CARS datasets. From the results, we can find that the large datasets, such as the N-CARS dataset, are hard for both the H-First and HOTS learning algorithms to converge to good feature representation. As an event-based feature extraction method, the HATS method performs a competitive performance, because HATS method also implements the spatio-temporal regularization. A linear SVM classifier they used limits the discrimination of the model when the class of objects and the data noise increase. However, our method has an advantage in large datasets, it is easy to broaden the network structure, and our SVD operation can push the network to show a stronger discrimination ability in orthogonal planes.

4.3. Model analysis

Incremental study. We test incremental broad learning algorithms in our model. First, we test the increment of the input samples (blue line in Fig. 4). Suppose the initial network is trained under the first 10000 training samples in the N-MNIST dataset. Then, the incremental algorithm is applied to add dynamically 10000 input samples each time until all the 60000 training samples are fed. The structure of the tested network is set as 10×10 feature nodes and 5000 enhancement nodes. Second, we test the increment of input samples and enhancement nodes together (yellow line in Fig. 4). The network is initially set to have 10×10 feature nodes and 5000 enhancement nodes. Then, the additional enhancement nodes are increased dynamically at 250 each, and the input samples are increased at 10000 each. Third, we test the incremental of input samples, feature nodes, and enhancement nodes together (grey line in Fig. 4). The initial network is set as 10×6 feature nodes and 3000 enhancement nodes. The feature nodes are dynamically increased from 60 to 100 at the step of 10 in each update, the corresponding enhancement nodes for the additional features are increased at 750 each, and the additional enhancement nodes are increased at 1250 each. The input samples are increased at 10000 each. The results of each update could be checked in Fig. 4. It is observed that the dynamic increments on both feature nodes and enhancement nodes perform the best. It may be caused by the randomness nature of the feature nodes and the enhancement nodes. This implies that the dynamic update of the model using incremental learning could present a compatible result; meanwhile, it provides the opportunities to adjust structure the system to achieve better performance.

Parameter study. We also test the influence of the length of cell R and the incremental time Δ_{incr} to the classification accuracy. The visualization of the PFM outputs X according to different parameters are shown in Fig. 5. The influence of R on the classification accuracy is shown in Tab. 3. We find that when R is low, the neighboring information could not be added. But when R is too large, large events are computed repeatedly. We also find that there are the best parameters R and Δ_{incr} for different datasets. For easily applying in experiments, we use the $R = 7$ and $\Delta_{incr} = 100ms$ for all the datasets. Next, we run experiments using SVD to simplify the structure after the network extension. The experiments are tested in MNIST-DAVIS dataset. The threshold $\varepsilon_f = \varepsilon_h = 1$ and $\varepsilon = N$ are set, which means that there is no simplification on feature nodes and enhancement nodes generation, but only to keep the first N important principle components in the final simplified network, $i.e.$, apply the SVD operation to $A_F^{\{m,n\}}$. As shown in Tab. 2, N is selected as 500, 1000, 1500, 2000, 2500, and 3000. We could obviously observe that the net-

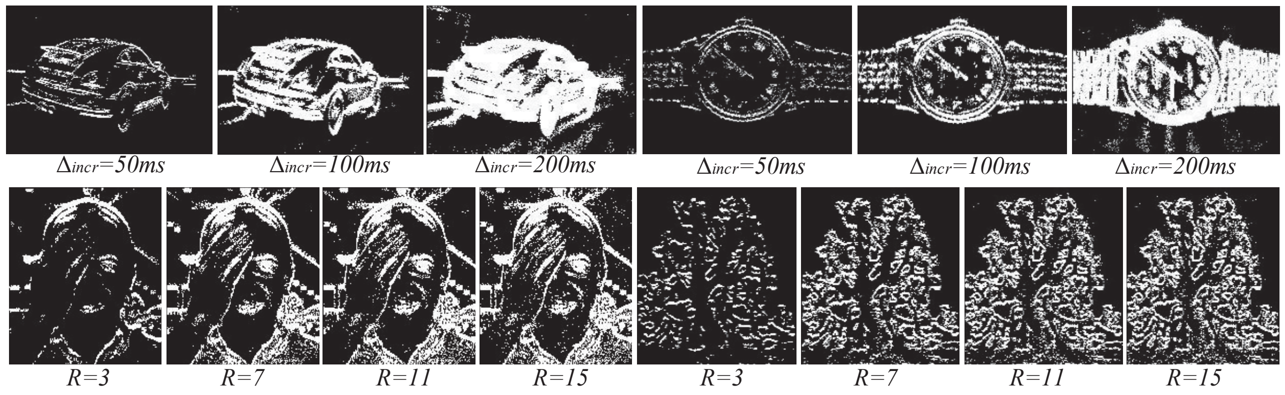


Figure 5. The influence of parameters Δ_{incr} and R. The first row is the PFM output in $R = 7$ and different Δ_{incr}, the second is the PFM output in $\Delta_{incr} = 100ms$ and different R.

works selected by the SVD improves the classification accuracy more than 0.5%. The role of the SVD is similar to the dropout rate in the CNNs. The redundant nodes could be removed by the SVD operation.

Model ablation. We systematically investigate the contribution of different components of our model by using different combinations of the components proposed in Sec. 3, including Peak Memory Cell (PMC, Sec. 3.1), Incremental strategy (Incr, Sec. 3.4), and SVD (Sec. 3.5). Tab. 4 shows the classification accuracy of model ablation results on the N-CARS dataset. The baseline method EBLS uses only PMC component. Compared to our full event-based incremental BLS model, the performance of other models decrease 1%-8% in accuracy rate. Surprisingly, the PMC component plays a significant role in our model, its performance drops from 89.9% (EBLS) to 83.4% ($1^{\#}$), meaning that the regularization brought by the local cell memory of Eq. (4) brings better accuracy. That is also certificated in the HATS method [42]. Compared with the PMC component, the SVD component usually makes sense when combining with the Incr component. The SVD + Incr model ($3^{\#}$) improves 4.3% compared with $1^{\#}$ model. If we only use the Incr component without the SVD component ($2^{\#}$), the accuracy decreases by 1.8% compared with $3^{\#}$ model.

Deep learning Vs Broad learning. Tab. 5 shows the comparison between our methods and the CNNs models: LeNet5 [9] and AlexNet [23]. The baseline method EBLS and our full model are tested on a 3.40-GHz Intel i7-6700 CPU processor PC with MATLAB platform. CNNs models are tested on the same processer with the Tensorflow framework. For the LeNet5 model, we set $batch\_size = 200$, $learning\ rate = 1e-3$. For the AlexNet model, we set $batch\_size = 64$, $learning\ rate = 1e-4$, $dropout = 0.9$. We can observe that the accuracy of our models (EBLS and full model) is higher than the CNNs model in the MNIST-DAVIS dataset, at the same time, our model is $15\times$ and $30\times$ faster than LeNet5 ($step = 20000$) and AlexNet ($step = 1500$) in training respectively, using less time in test than both CNNs models.

Our Models	PMC	Incr	SVD	Accuracy
$1^{\#}$	-	-	-	0.834
$2^{\#}$	-	✓	-	0.859
$3^{\#}$	-	✓	✓	0.877
$4^{\#}$	✓	✓	-	0.907
EBLS	✓	-	-	0.899
Full	✓	✓	✓	0.931

Table 4. Classification accuracy of the model ablation on the N-CARS dataset.

Methods	LeNet5	AlexNet	EBLS	Full
Accuracy	0.952	0.981	0.977	0.987
Train time (s)	2816.1	6174.3	79.74	172.34
Test time (s)	2.61	6.55	1.74	1.53

Table 5. The time comparison of the CNNs models and our model on the MNIST-DVS dataset.

5. Conclusion

In this work, we present an incremental Broad learning system for event-based object classification. It validates the idea that increasing the broad network by adding feature nodes and enhancement nodes is effective for the asynchronous event-based data, providing an alternative way to deal with the neuromorphic cameras. The proposed network architecture makes efficient use of the peak information in space and the memory information, extracting the peak-and-fire mapping feature as feature nodes in the broad network. The incremental learning integrated with an SVD operation promises the network in a non-redundancy feature space in both accuracy and efficiency.

6. Acknowledgement

This work is supported by NSFC 61701274.

References

[1] H. Akolkar, C. Meyer, Z. Clady, O. Marre, C. Bartolozzi, S. Panzeri, and R. Benosman. What can neuromorphic event-driven precise timing add to spike-based pattern recognition? *Neural Computation*, 27(3):561–593, 2015. 1

[2] R. Berner, C. Brandli, M. Yang, S. C. Liu, and T. Delbruck. A 240180 10mw 12us latency sparse-output vision sensor for mobile applications. In *Vlsi Circuits*, 2013. 1

[3] O. Bichler, D. Querlioz, S. J. Thorpe, J. P. Bourgoin, and C. Gamrat. Extraction of temporally correlated features from dynamic vision sensors with spike-timing-dependent plasticity. *Neural Networks*, 32(2):339–348, 2012. 3

[4] A. C. Bovik, M. Clark, and W. S. Geisler. Multichannel texture analysis using localized spatial filters. *IEEE Transactions on PAMI*, 12(1):55–73, 1990. 6

[5] M. Cannici, M. Ciccone, A. Romanoni, and M. Matteucci. Attention mechanisms for object recognition with event-based cameras. In *WACV*, 2019. 3

[6] Y. Cao, Y. Chen, and D. Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *IJCV*, 113(1):54–66, 2015. 3

[7] C. L. P. Chen, Z. Liu, and S. Feng. Universal approximation capability of broad learning system and its structural variations. *IEEE Transactions on Neural Networks and Learning Systems*, 30(4):1191–1204, 2019. 2, 3, 4

[8] X. Clady, J. M. Maro, S. Barr, and R. B. Benosman. A motion-based feature for event-based pattern recognition. *Frontiers in Neuroscience*, 10:594, 2016. 2

[9] Y. L. Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. *NIPS*, 2(2):396–404, 1990. 1, 6, 8

[10] T. Delbruck, B. Linares-Barranco, E. Culurciello, and C. Posch. Activity-driven, event-based vision sensors. In *IEEE International Symposium on Circuits and Systems*, 2010. 1

[11] O. Garrick, J. Ajinkya, C. Gregory K., and T. Nitish. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in Neuroscience*, 9(178), 2015. 6

[12] O. Garrick, M. Cedric, E. C. Ralph, P. Christoph, T. Nitish, and B. Ryad. Hfirst: A temporal approach to object recognition. *IEEE Transactions on PAMI*, 37(10):2028–2040, 2015. 2, 3, 6

[13] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk. Fast alternating direction optimization methods. *SIAM Journal on Imaging Sciences*, 7(3):1588–1623, 2014. 6

[14] T. Gollisch and M. Meister. Eye smarter than scientists believed: Neural computations in circuits of the retina. *Neuron*, 65(2):150–164, 2010. 1

[15] R. Gutig and H. Sompolinsky. The tempotron: a neuron that learns spike timingcbased decisions. 9:420–428, 2006. 3

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

[17] G. Huang, Z. Liu, L. Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 1

[18] H. Jhuang, T. Serre, L. Wolf, and T. Poggio. A biologically inspired system for action recognition. In *ICCV*, 2007. 2

[19] A. Jos, C. Prez, and 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 Transactions on PAMI*, 35(11):2706–2719, 2013. 1

[20] H. Kim, S. Leutenegger, and A. J. Davison. Real-time 3d reconstruction and 6-dof tracking with an event camera. In *ECCV*, 2016. 2

[21] M. Klassen, Y. H. Pao, and V. Chen. Characteristics of the functional-link net: A higher order delta rule net. In *IEEE International Conference on Neural Networks*, 1988. 3

[22] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009. 6

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 6, 8

[24] X. Lagorce, S. H. Ieng, X. Clady, M. Pfeiffer, and R. B. Benosman. Spatiotemporal features for asynchronous event-based data. *Frontiers in Neuroscience*, 9:46, 2015. 2

[25] X. Lagorce, G. Orchard, F. Gallupi, B. E. Shi, and R. Benosman. Hots: A hierarchy of event-based time-surfaces for pattern recognition. *IEEE Transactions on PAMI*, 39(7):1346 – 1359, 2017. 1, 2, 3, 6

[26] F. F. Li, F. Rob, and P. Pietro. One-shot learning of object categories. *IEEE Transactions on PAMI*, 28(4):594–611, 2006. 6

[27] H. Li, H. Liu, X. Ji, G. Li, and L. Shi. Cifar10-dvs: An event-stream dataset for object classification. *Frontiers in Neuroscience*, 11:309–, 2017. 6

[28] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 1

[29] D. Martł, M. Rigotti, M. Soek, and S. Fusi. Energy-efficient neuromorphic classifiers. *Neural Computation*, 28(10):1, 2016. 3

[30] E. Muggler, C. Bartolozzi, and D. Scaramuzza. Fast event-based corner detection. In *BMVC*, 2017. 2

[31] D. Neil, M. Pfeiffer, and S.-C. Liu. Phased lstm: Accelerating recurrent network training for long or event-based sequences. In *NIPS*. 3, 6

[32] O. Peter, N. Daniel, L. Shih-Chii, D. Tobi, and P. Michael. Real-time classification and sensor fusion with a spiking deep belief network. *Frontiers in Neuroscience*, 7(7):178, 2013. 2

[33] 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, 2010. 1

[34] H. Rebecq, T. Horstschaefer, G. Gallego, and D. Scaramuzza. Evo: A geometric approach to event-based 6-dof parallel tracking and mapping in real-time. *IEEE Robotics and Automation Letters*, 2(2):593–600, 2017. 2

[35] E. Reinhard, G. Ward, S. Pattanaik, and P. Debevec. *High dynamic range imaging : acquisition, display, and image-based lighting*. Princeton University Press, 2005. 1

[36] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1

[37] B. Rueckauer, I.-A. Lungu, Y. Hu, and M. Pfeiffer. Theory and tools for the conversion of analog to spiking convolutional neural networks. In *NIPS workshop*, 2016. 3

[38] A. Russell, G. Orchard, Y. Dong, S. Mihalas, E. Niebur, J. Tapson, and R. Etienne Cummings. Optimization methods for spiking neurons and networks. *IEEE Transactions on Neural Networks*, 21(12):1950–1962, 2010. 3, 6

[39] B. Z. Sajjad Seifozzakerini, Wei-Yun Yau and K. Mao. Event-based hough transform in a spiking neural network for multiple line detection and tracking using a dynamic vision sensor. In *BMVC*, 2016. 2

[40] T. Serrano-Gotarredona and B. Linares-Barranco. A 128 $\times$ 128 1.5% contrast sensitivity 0.9% fpn 3μs latency 4mw asynchronous frame-free dynamic vision sensor using transimpedance preamplifiers. *IEEE Journal of Solid-State Circuits*, 48(3):827–838, 2013. 1, 6

[41] S. Sheik, M. Pfeiffer, F. Stefanini, and G. Indiveri. Spatio-temporal spike pattern classification in neuromorphic systems. 2013. 3

[42] A. Sironi, M. Brambilla, N. Bourdis, X. Lagorce, and R. Benosman. Hats: Histograms of averaged time surfaces for robust event-based object classification. In *CVPR*, 2018. 2, 3, 6, 8

[43] R. B. Xavier Clady, Sio-Hoi Ieng. Asynchronous event-based corner detection and matching. *Neural Networks*, 66:91–106, 2015. 2

[44] B. Zhao, R. Ding, S. Chen, B. Linares-Barranco, and H. Tang. Feedforward categorization on aer motion events using cortex-like features in a spiking neural network. *IEEE Transactions on Neural Networks and Learning Systems*, 26(9):1963–1978, 2015. 3