An End-to-End Broad Learning System for Event-Based Object Classification

SHAN GAO\(^1\), GUANGQIAN GUO\(^2\), HANQIAO HUANG\(^1\), XUEMEI CHENG\(^1\), AND C. L. PHILIP CHEN\(^1\), (Fellow, IEEE)

\(^1\)Unmanned System Research Institute, Northwestern Polytechnical University, Xi’an 710072, China
\(^2\)School of Electrical and Control Engineering, Shaanxi University of Science and Technology, Xi’an 710021, China

Corresponding author: Hanqiao Huang (huanghanqiao@nwpu.edu.cn)

This work was supported in part by the NSFC under Grant 61601505, in part by the Innovative Talents Promotion Project of Shaanxi under Grant 2018KJXX-002, and in part by the Fundamental Research Funds for the Central Universities.

ABSTRACT Event cameras are bio-inspired vision sensors measuring brightness changes (referred to as an ‘event’) for each pixel independently, instead of capturing brightness images at a fixed rate using conventional cameras. Asynchronous event data mixed with noise information is challenging for event-based vision tasks. In this paper, we propose a broad learning network for object detection using the event data. The broad learning network consists of two distinct layers, a feature-node layer and an enhancement-node layer. Different to convolutional neural networks, the broad learning network can be extended by adding nodes into layers during training. We design a gradient descent algorithm to train network parameters, which creates an event-based broad learning network in an end-to-end manner. Our model outperforms state-of-the-art models, specifically, because of the small scale and increased speed displayed by our model during training. This demonstrates the superiority of event cameras towards online training and inference.

INDEX TERMS Event camera, object classification, broad learning system, incremental learning.

I. INTRODUCTION

Object classification is an important and fundamental task in image and video understanding, which has been investigated for decades and this research primarily focused on framed-based vision formats. In identifying superior performance, research has highlighted Convolutional Neural Networks (CNNs) [13], [24], [25], [31], [45] as the domination for image-based object classification task. However, frame-based imaging is not the only option for computer vision, and this format always has a number of shortcomings, presenting challenges for further development [33]. For example, the limited frame rate, high redundancy between frames, blurriness due to slow shutter adjustment under varying illumination, and high-power requirements [15] have all stifled progress within the field.

With the advent of bio-inspired vision sensors, neuro-morphic cameras [3], [43], [49] are increasingly being used to explore the aforementioned phenomena. These devices essentially attempt to emulate biological retina functioning, and particularly event cameras, such as the Asynchronous Time-based Image Sensor (ATIS) [43] and Dynamic Vision System (DVS) [16]. These technologies are fundamentally different from traditional cameras which sequence frames at fixed intervals. The term ‘event’ refers to an output spike, characterized by a specific spatial location \((x, y)\), timestamp \((t)\) and brightness change polarity \((p)\), shown in Fig. 1. This means that the output from an event camera is a stream of asynchronous spikes, triggered by brightness changes sensed by individual pixels. Thanks to their logarithmic sensitivity to illumination changes, event-based cameras also have a much larger dynamic range, exceeding 120dB [43]. The growth in popularity of these types of sensors, and the distinct advantages, have enabled researchers to implement event-based vision for a variety of applications, such as object tracking [17], [21], [37], [44], visual odometry [28], [48], [56], and optical flow estimation [2], [57].

Recently, several models have been proposed to utilize event-based data for object classification, e.g., CNNs [27], Hierarchical like models [20], and Deep Belief Networks (DBNs) [41]. Other methods have begun to explore the integration of dynamical information into recognition using motion-direction sensitive units [26] or dynamical networks such as echo-state networks [32]. However, the networks...
mentioned are inferior to their frame-based counterparts because asynchronous event-based data have different characteristics. Event-based cameras, frequently only output scene noise, while conversely sometimes generating large amounts of information. In order to address this inconsistent output problem, we adjusted the frame-based CNNs model to cope with asynchronous data. However, the accuracy of event-based object classification algorithms remains far behind frame-based technologies which may be explained by the lack of effective event representations which are used to accurately depict spatio-temporal characteristics of event-based data. The operation used to process RGB-based images in a high resolution may also be unsuitable for the asynchronous event-based data. Also, specific network information is absent for event-based object classification. Although, the network should be able to handle asynchronous event data, and could theoretically be fully trained at both the small and large numbers of samples. Therefore, it is necessary to design a flexible neural network for event-based data which will not become less accurate or less efficient.

Here we propose an end-to-end learning framework that address both problems. First, we introduce a bio-inspired mechanism, namely Peak-and-Fire Mapping (PFM), to record the peak of the stream of the asynchronous event data. The event peak encodes the dynamic information of one pixel and neighboring cell structures. Compared to previous approaches, PFM is robust to spatio-temporal noise. Second and significantly, we propose to use a broad learning network to handle PFM representation. The Broad Learning System (BLS) [9] is a novel structure established based on the random vector functional-link neural network (RVFLNN) [42] which inherits features, and can be expanded substantially. Contrary to some popular deep neural networks which suffer from a time-consuming learning for excessive parameters, BLS can provide a faster scheme with higher accuracy. These characteristics make the BLS very efficient and much less time-consuming in classification and regression problems. Nevertheless, the original BLS cannot be applied to online learning systems directly because the original learning algorithm is a one-shot scheme based on the ridge regression approximation of pseudoinverses which are not compatible with weight updates in feature or enhancement nodes. Hence, we designed an iterative learning algorithm which adjusts parameters within the nodes of all layers by a gradient descent algorithm. This step ensures the entire BLS is an end-to-end learning system, capable of adapting, unlike any of the previous models.

We designed a broad learning network to deal with event-based data for object classification. The broad network provides an alternative way of learning, distinct from the previous deep CNNs models. The key contributions of this paper are: 1. BLS integrates asynchronous data into a flexible broad network; 2. A gradient descent algorithm is proposed to train the BLS in an end-to-end manner. This was implemented to ensure our BLS model is faster and can outperform state-of-the-arts in terms of accuracy. The proposed flexible network structure is able to be fully trained in large- and small-scale training samples which demonstrates that event cameras are superior in nearly online training and inference applications.

II. RELATED WORK

In this section, we review the most relevant research into event-based object classification, including event-based feature and object classification, event-based networks, and functional link neural networks.

The majority of past research into event-based object detection focuses on detecting and tracking stable features, and the applications include: simultaneous localization and mapping applications [28], [44], corner detectors [39], [54], edge and line extraction [47], event-based flow [12]. Recently, several studies have proposed to use standard learning architectures to extract event features for object classification. Lagorce et al. [33] proposed an hierarchical representation based upon surface images, edge clustering, and cross-layer fusion. These models lack noise sensitivity and have clear dependencies according to different types of object motion within each scene. Networks used in event-based data originate from training artificial neural networks by reproducing or imitating learning rules observed in biological neural networks [4], [23], [38], [50]. These patterns are similar to processes involved in frame-based computer vision, which try to optimize network weights by minimizing smooth error functions.

The most commonly used architectures for event-based cameras are Spiking Neural Networks (SNN) [1], [7], [40], [46], [55]. SNN is a type of artificial neural networks based on set units communicating with one another through spikes and perform computation as and when needed. However, a big drawback of these models is that they are not differentiable. For example, when multiple processing layers are involved, the training procedure becomes much more complicated.
than the back-propagation algorithm used in conventional neural networks. However, it is difficult to properly train an SNN with gradient descent although to overcome this, Garrick et al. [20] used predefined Gabor filters as network weights. Others have also proposed to first train a CNN and then to convert weights and transpose these to an SNN [7], [46]. Cannici et al. [6] added the attention mechanism in an event-based YOLO structure, namely YOLE, to solve the event-based classification problem. Unfortunately, the network conversion process impacted on performance which was then lower than conventional CNNs.

The broad learning network we used belongs to the Functional Link Neural Network (FLNN), which was proposed by Klassen et al. [29]. FLNN is a variant of the higher order neural network without hidden units, and it has been developed into the random vector functional link network [9]. Different various improvements and models as well as successful applications of FLNN were developed due to its universal approximation properties. A comprehensive review of FLNN can be referred to [14]. Chen et al. [8] have presented an adaptive implementation of the FLNN architecture together with a supervised learning algorithm named rank-expansion with instant learning. The advantage of this rapid algorithm is that it can learn weights in a one-shot training phase without iteratively updating parameters. In addition, a fast learning algorithm proposed in [11] to find optimal weights of the flat neural networks (especially, the functional link network), which adopts the linear least-square method, and this algorithm makes it easier to update weights instantly for incremental input patterns and enhancement nodes.

This paper begins with an introduction of the asynchronous spatio-temporal mapping way from event camera in Section III, followed by a description of the broad learning system in Section IV, where we describe an end-to-end network training method to handle the event-based data. The experiments results and a conclusion are in Section V and Section VI.

III. ASYNCHRONOUS SPATIO-TEMPORAL MAPPING
A. EVENT STREAM
Consider an event-based sensor with pixel grid size $M \times N$, an event which can be generated when the illustration changes in the pixel location. The $i$th event is defined as

$$e_i = [x_i, y_i, t_i, p_i]^T,$$  \hspace{1cm} (1)

where $e_i$ consists of a location $(x_i, y_i)$, time $(t_i)$ and polarity $(p_i)$, $p_i \in \{-1, 1\}$, where $(x_i, y_i)$ is the spatial location, $t_i$ is the timestamp, $-1$ 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 object’s spatial distribution and dynamical behavior. The OFF event means the brightness changes lower, and the ON event means higher.

In a given time interval $\Delta_{\text{incr}}$, the event camera will trigger a number of events: $E = \{e_i\}_{i=1}^K$. Due to their asynchronous nature, events are represented as a set. To use events in combination with a neural network it is necessary to convert the event set into a grid-like representation $X$. This means, we should find a mapping $M : E \rightarrow X$ between the set $E$ and a presentation $X$. Ideally, this mapping way should maintain the structure (i.e., spatiotemporal locality) as well as event information.

B. EVENT-BASED CONVOLUTION
The event set could be defined in the view of signal spatial convolution [6], [34]. To derive a meaningful signal from the event stream, we should convolve the event data with a suitable aggregation kernel. The convolved signal thus be defined as

$$P^\alpha(x, y, t) = \sum_{e_i \in X} f(x_i, y_i, t_i) k(x - x_i, y - y_i, t - t_i),$$  \hspace{1cm} (2)

where $f(x_i, y_i, t_i)$ is the event measure, and $k(x, y, t)$ is kernel function. The prior works used several kinds of kernel. Alpha-kernel $k(x, y, t) = f(x, y) e^{\tau (t)/t}$ [34], the exponential kernel $k(x, y, t) = f(x, y) e^{\tau (t)/t}$. $\tau$ is set as the decay parameter. In fact, the exponential kernel is also used to construct the Hierarchy of Time-Surfaces [33] and Histogram of Average Time-Surfaces [52], where events are aggregated into exponential time surfaces.

After the operation of kernel convolution, we define a region of convolution to discretize the convolution form of Eq. (2) into a spatio-temporal coordinates as

$$P^\alpha(x_i, y_m, t_n) = \sum_{e_i \in \mathcal{E}} f(x_i, y_i, t_i) k(x_i - x_i, y_m - y_i, t_n - t_i),$$  \hspace{1cm} (3)

where the discrete representation is determined by a likely Voxel grid, $c_k$, in the size of $R \times R \times R$. We use $x_i \in \{0, \ldots, K - 1\}$, $y_m \in \{0, \ldots, R - 1\}$, and $t_n$ is the $n$th $\Delta_{\text{incr}}$. Typically, the structure of the events contains information about the object and it’s movement. The Voxel grid keeps track of not only the single event’s activity but also each surrounding event so that the discrete Voxel-grid convolution provides strong local support for the event description.

C. PEAK-AND-FIRE MAPPING
Generally, kernel functions are based on task-dependent heuristics with no general agreement on which is the optimal

![FIGURE 2. The original structure of BLS.](image-url)
kernel for maximizing performance. In the event-based object classification task, we design a peak-and-fire mechanism to regulate event data from discrete presentation to a robust presentation. We detect peaks of activity inside each event location and fire the peaks as outputs, as shown in Fig. 3. This additional step was inspired by the spiking recognition network proposed in [33], [52]. We adopted the exponential kernel in our classification task, because the exponential decay in the exponential kernel expands the activity of past events and provides information about activity history. The discrete representation in Eq. (3) can be defined as

\[ P_{e_i} = \begin{cases} \sum_{e_j} \exp(-\Delta t/\tau) & \text{if } t_j - \Delta_{incr} \leq t_i, \forall j \\ 0 & \text{otherwise} \end{cases} \]  

(4)

where \( P_{e_i} \) provides a dynamic temporal context at the location \( l_i \) and \( \Delta t = t_j - t_i \). \( \tau \) is set as the decay parameter. Considering the polarity of each event, we only intercalate events in the same polarity.

The peak in a Voxel grid \( e_i \), a peak in the grid is considered to be valid if its value \( P_{ck} \) is above the level of confidence in the interval, \( \Delta_{incr} \). We intercalate the components of the event peak in grid \( c_k \) region, Eq. (4) as follows;

\[ P_{ck} = \frac{1}{N_{ck}} \sum_{e_i \in c_k} P_{e_i}, \]  

(5)

where \( N_{ck} \) is the number of events in \( c_k \). A peak in a cell is considered valid only if the number is greater than the \( \mu \) where \( N \in \mathbb{Z} \).

When we use the peak-and-fire mechanism in the incoming interval \( \Delta_{incr} \), every incoming event \( e_i \) needs to 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 \( M_{c_k} \) for every cell \( c_k \), shown in Fig. 3. The past events relevant for \( c_k \) are stored. The output of \( c_k \) is defined as

\[ X_{ck}^{l+\Delta_{incr}} = \begin{cases} P_{c_k}(R,p) + M_{c_k}(\Delta_{incr}) & \text{if } e_i \in c_k \\ P_{c_k}^i(R,p) & \text{otherwise} \end{cases} \]  

(6)

When a new event arrives in \( c_k \), we update Eq. (6) 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 compute the robust feature representation without significant increasing in memory requirements.

IV. END-TO-END BROAD LEARNING SYSTEM

In this section, we input the PFM result to a broad learning network, which is in the form of a flat network. Then, we propose extending the basic broad learning network by incrementally adding more nodes in a broad way. Finally, a low-rank orthogonal approximation is introduced to decrease the network redundancy during the broad extension phase.

A. REVIEW OF BROAD LEARNING SYSTEM

To build our incremental learning framework, we start with a traditional (i.e., non-incremental) BLS for classification. Our model is based on the BLS [10] providing an effective and efficient learning framework for classification and regression problems, given the training data set \((X, Y)|X \in \mathbb{R}^{N \times M}, Y \in \mathbb{R}^{N \times C}\) from \( C \) classes. Here, \( X \) denotes the event data presentation in time interval \( \Delta_{incr} \), described in Sec. III. In the BLS, the training samples are first transformed into \( n \) random feature spaces by feature mapping \( \phi_i \) as

\[ Z_i = \phi_i(XW_{f_j} + \beta_{f_j}), \quad i = 1, 2, \ldots, n, \]  

(7)

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

\[ H_j = \xi_j(Z^nW_{h_j} + \beta_{h_j}), \quad j = 1, 2, \ldots, m, \]  

(8)

where \( \xi_j \) is a nonlinear activation function. We denote the outputs of the enhancement layer by \( H^m \triangleq [H_1, H_2, \ldots, H_m] \). Therefore, the output \( \hat{Y} \) of a BLS has the following form

\[ \hat{Y} = [Z_1, Z_2, \ldots, Z_n, H_1, H_2, \ldots, H_m]W = [Z^n, H^m]W = AW, \]  

(9)

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 minimization as follows;

\[ \min_w \|Y - AW\|_2^2 + \lambda \|W\|_2^2, \]  

(10)

where \( \lambda \) 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. (10) is zero, we can obtain the solution of output weight as

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

(11)

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

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

(12)

Once the training process is complete, the randomly generated weights $W_f$, $W_h$ and the learnt output weight $W$ should be fixed. Then, the BLS responses to testing samples $x_i$ as

$$ \hat{y}_i = x_i W. $$

(13)

B. BLS ARCHITECTURE

The BLS is constructed based on the flatted functional-link networks. Please see Fig. 4 for an outline of the entire network structure. The network consists of feature nodes, enhancement nodes, and output nodes. Initially, event-based data are computed by PFM as the input of the network, which are then connected as feature nodes $Z_i$ by the weight $W_f$ and bias $\beta_f$. The enhancement nodes $H_i$ are generated by the feature nodes and activation function according to the weight $W_h$ and bias $\beta_h$. The output nodes are connected with feature nodes and enhancement nodes directly by $W$ and $\beta$.

So far, a general network architecture of the BLS is presented, although, the selection of functions for the feature mapping requires further attention. Functions $\phi(\cdot)$ and $\xi(\cdot)$ have no explicit restrictions, which means that the common choices such as kernel mappings, nonlinear transformations, or convolutional functions are acceptable. Specifically, if we use the convolutional functions for feature mapping, the BLS network structure would be 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.

C. END-TO-END LEARNING FOR BLS

In the BLS architecture, we know that there are six parameters should be trained, including $W$, $\beta$, $W_h$, $\beta_h$, $W_f$, and $\beta_f$. In the solution of the original BLS model [9], the weights $W$ are computed through ridge regression approximation using Eq. (12). In [9], a pseudoinverse computation is used to solve the ridge regression approximation in the training. To extract sparse features from the given training data $X$, the weights $W_h$ and $W_f$ are calculated by a sparse autoencoder and solved by the Alternating Direction Method of Multipliers (ADMM) [22]. So, $W_h$ and $W_f$ are randomly generated, and not involved in the training process. In other words, only the weight $W$ is trained in the increment training and therefore, the original solution in BLS is not an end-to-end learning system and the parameter solution is fragile. Once the number of samples or categories increases, the ability of feature representation will be weakened without updating $W_f$, $\beta_f$, $W_h$ and $\beta_h$ through the iteratively training.

In contrast to previous research, we propose using the gradient-descent learning method to update all parameters iteratively, which makes the event-based BLS be an end-to-end learning system. Suppose that there are $n$ mapping groups with $K_i$ feature nodes in the $i$th group, and there is one group of $m$ enhancement nodes, given an input sample $x = (x_1, x_2, \ldots, x_M)$ and the desired output $y$, we denote $W_f = (w_{f_{k_1}}^\mu)_{M \times K_1}$, $\beta_f = (b_f^k)_{1 \times K_1}$, $W_h = (w_{h_{k_1}}^\mu)_{K_1 \times (K_1 + K_2 + \ldots + K_n) \times m}$, and $\beta_h = (b_h^k)_{1 \times m}$, where $w_{f_1}^\mu$ is the weight connecting the $i$th input $x_1$ to the $k$th feature node in $i$th mapping group, $b_f^k$ is the bias term associated to the $k$th feature node in $i$th mapping group, $w_{h_1}^\mu$ is the weight connecting the $k$th feature node of the $i$th mapping group to the $j$th enhancement node, and $b_j$ is the bias term associated to the $j$th enhancement node. Then the $q$th BLS output denoted as $\hat{y}_q$ is

$$ \hat{y}_q = \sum_{i=1}^{n} \sum_{k=1}^{K_i} w_{f_k}^i F_k^i + \sum_{j=1}^{m} w_{h_j}^q \sigma(\sum_{i=1}^{n} \sum_{k=1}^{K_i} w_{h_k}^i F_k^i + b_j), $$

(14)

where $F_k^i = \sigma((\sum_{l=1}^{M} w_{f_l}^i x_l) + b_f^k)$ denotes the output of the $k$th feature node in the $i$th mapping group, and $\sigma(\cdot)$ is the activation function. We set $\sigma(x) = \frac{1}{1 + e^{-x}}$, and it is trivial to obtain $\hat{y} = \sigma(1 - \sigma)$. After the softmax, we could obtain

$$ \hat{y} = \frac{e^{\hat{y}_q}}{\sum_{q=1}^{c} e^{\hat{y}_q}}, $$

(15)

where $c$ is the number of categories, $\hat{y}$ presents the score for a particular class. We define the weight matrix connecting the outputs of feature nodes and enhancement nodes to the output neuron (i.e., the weights in the top layer) as

$$ W = (w_f^1, \ldots, w_f^{K_1}, w_h^1, \ldots, w_h^{K_1}, w_f^q, \ldots, w_f^m), $$

(16)

where $w_{f_k}^i$ is the weight connecting the $k$th feature node in the $i$th mapping group to the $q$th output neuron, and $w_{h_j}^q$ is the
weight connecting the $j$th enhancement node to $q$th the output neuron. The error function between the actual output $y$ and the model output $\hat{y}$ is defined as follows:

$$E = \sum_{c=1}^{C} -y_c \log \hat{y}, \quad (17)$$

where $y$ is a one-hot vector. Then, it becomes easy to deduce the following derivatives for parameters $W$, $\beta$, $W_f$, $\beta_f$, $W_h$, and $\beta_h$. The parameter $W$ has two parts, one of which is $w_k^i$ and the other is $w_j^q$ as described above. The derivatives of $w_k^i$ are

$$\frac{\partial E}{\partial w_k^i} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial q} \cdot \frac{\partial q}{\partial w_k^i} = -y \cdot (1 - \hat{y}) \cdot F_k^i, \quad (18)$$

Just for simplicity, we replace $\hat{y}$ by $(1 - \hat{y})$ with $\phi$ in the follow equations. And

$$\frac{\partial E}{\partial w_j^q} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial q} \cdot \frac{\partial q}{\partial w_j^q} = \phi \cdot \sigma_j(\sum_{i=1}^{K_i} \sum_{k=1}^{K} w_{jk}^i F_k^i + b_j) \quad = \phi \cdot \Psi_j. \quad (19)$$

The next step is to deduce the derivatives of $w_{jk}^i$ and $b_j$, and by using the chain rule of derivatives, we obtain them as

$$\frac{\partial E}{\partial w_{jk}^i} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial q} \cdot \frac{\partial q}{\partial w_{jk}^i} = \phi \cdot w_{jk}^i \cdot F_k^i \cdot \sigma_j(\sum_{j=1}^{M} w_{j}^j \cdot \Psi_j + w_{jk}) \quad = \phi \cdot F_k^i \cdot \Psi_j \cdot (1 - \Psi_j), \quad (20)$$

where $\Psi_j \triangleq \sigma_j(\sum_{i=1}^{K_i} \sum_{k=1}^{K} w_{jk}^i F_k^i + b_j)$. Then,

$$\frac{\partial E}{\partial b_j} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial q} \cdot \frac{\partial q}{\partial b_j} = \phi \cdot w_{jk}^i \cdot \Psi_j \cdot (1 - \Psi_j). \quad (21)$$

Through Eq. (14) we can easily obtain

$$\frac{\partial \hat{y}_q}{\partial \hat{y}_k} = w_{jk}^i \cdot \Psi_j(1 - \Psi_j) \quad \text{and} \quad w_{jk}^i \cdot \Psi_j(1 - \Psi_j) \quad \text{and} \quad \frac{\partial \hat{y}_q}{\partial \hat{y}_k} = \phi \cdot \left[w_{jk}^i \cdot w_{jk}^i \cdot \Psi_j(1 - \Psi_j) \right] \cdot x_i \cdot \Psi_j(1 - \Psi_j), \quad (22)$$

where $\Psi_j \triangleq \sigma_j(\sum_{i=1}^{K_i} \sum_{k=1}^{K} w_{jk}^i F_k^i + b_j)$. Similarly,

$$\frac{\partial E}{\partial b_{jk}^i} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial q} \cdot \frac{\partial q}{\partial b_{jk}^i} = \phi \cdot \left[w_{jk}^i + w_{jk}^i \cdot \Psi_j(1 - \Psi_j) \right] \cdot \Psi_j(1 - \Psi_j). \quad (23)$$

The iterative formula for the parameters $w_{jk}^i$, $w_{jk}^i$, $w_{jk}^i$, $w_{jk}^i$, $b_j$ and $b_{jk}^i$ ($k = 1, 2, \ldots, K_i$, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$, $q = 1, 2, \ldots, o$) are summarized as follows,

$$w_{jk}^i(t + 1) = w_{jk}^i(t) - \eta \cdot \frac{\partial E}{\partial w_{jk}^i(t)}, \quad (24a)$$

$$w_{jk}^i(t + 1) = w_{jk}^i(t) - \eta \cdot \frac{\partial E}{\partial w_{jk}^i(t)}, \quad (24b)$$

where $\eta$ is the learning rate, and $\frac{\partial E}{\partial w_{jk}^i(t)}$, $\frac{\partial E}{\partial w_{jk}^i(t)}$, $\frac{\partial E}{\partial w_{jk}^i(t)}$, $\frac{\partial E}{\partial w_{jk}^i(t)}$, $\frac{\partial E}{\partial w_{jk}^i(t)}$, $\frac{\partial E}{\partial w_{jk}^i(t)}$ have all been deduced in Eqs. (18-23). Above all, we obtain the learning process of all the parameters in the event-based BLS.

**D. INCREMENTAL LEARNING**

In various applications, the original BLS network may be unsophisticated for learning. This may be caused by insufficient feature and enhancement nodes, rendering the system incapable of extracting enough underlying variation factors which are necessary to define the structure of the input data. In popular deep structure networks, when the existing model is incapable of task learning, the general practices are to either increase the number of the filters (or windows) or alternatively to increase the number of layers. However, these adaptive procedures suffer from more tedious, salutory learning which requires resetting parameters for each new structure. Instead, in the proposed BLS, if the increment of new feature and enhancement nodes are required, the whole structure can be easily be constructed and incremental learning can take place without the need to reset and retrain the entire network.

Here, let us consider the incremental learning for newly incremental feature and enhancement nodes. Assume that the initial structure consists of $n$ groups feature mapping nodes and $m$ groups broad enhancement nodes. Considering that the $(n + 1)$th feature group nodes are added and denoted as

$$Z_{n+1} = \phi \left(XW_{c_{n+1}} + \beta_{c_{n+1}} \right), \quad (25)$$

where $W_{c_{n+1}}$ and $\beta_{c_{n+1}}$ are new weight and bias respectively. The corresponding enhancement nodes are randomly generated as follows:

$$H_{c_{n}} = [\xi (Z_{n+1} W_{c_{1}} + \beta_{c_{1}}) \ldots \xi (Z_{n+1} W_{c_{n}} + \beta_{c_{n}})], \quad (26)$$

where $W_{c_{n}}$ and $\beta_{c_{n}}$ are randomly generated. Denote $A_{p+1} = [A_{p+1}^{m} H_{c_{n+1}}]$, which is the upgrade of new features and the corresponding enhancement nodes.
We validated our approach using five different datasets. Four datasets were generated by converting standard frame-based datasets to events (i.e., N-MNIST [19], N-Caltech101 [19], MNIST-DVS [49] and CIFAR10-DVS [36] datasets) and a novel dataset, recorded from real-world scenes, i.e. the N-CARS dataset [52]. N-MNIST, N-Caltech101, MNIST-DVS, and CIFAR10-DVS are four publicly available datasets created by converting the popular frame-based MNIST [13], Caltech101 [35] and CIFAR10 [30] 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 [19]. Similarly, the MNIST-DVS and CIFAR10-DVS datasets were created by displaying a moving image on a monitor and recorded with the ATIS camera [49]. N-Cars dataset is split in 7940 car and 7482 background training samples, and 4396 car and 4211 background testing samples. The duration of each sample is 100ms. MNIST-DVS contains 10000 samples, generated at three different resolutions, scale4, scale8, and scale16. We used 90% of the samples for training and 10% for testing in scale 4. The duration of each is approximately 2.3s. N-Caltech101 consists of 100 different object classes and a background class. Each category has between 31 and 4211 images. The duration is approximately 300ms. In our experiments, 20% of the data was randomly selected for testing and the remainder was used for training. We found that the sample duration is not identical, but we extracted a single 100ms window of events, as input to our object classification framework. We used the $\Delta_{incr} = 100ms$ for all samples.

### B. NETWORK SETTINGS

In the PFM formulation, the kernel’s length is $R = 7$ and the memory cell’s length is also set as $R$, which makes the peak value easy to calculate in the same size. $\tau$ is set as $10^6$ in Eq. (4). For the N-MNIST dataset, we set the number of feature nodes and enhancement nodes as 100 and 11000 respectively, which are denoted as (100, 11000). For N-Caltech101, MNIST-DVS, CIFAR10-DVS, and N-CARS datasets, the networks are set as (100, 11000), (100, 5000), (60, 3000), and (100, 3000), respectively. In training, we set $\text{batch\_size} = 200$. The learning rate is initialized at $10^{-3}$ and decayed by a factor of 0.1 every 10k iterations. Typically, we can also begin the BLS network from a basic structure, such as 60 feature nodes and 1000 enhancement nodes. With the increment of the input event-based sample, we can add the feature nodes and enhancement nodes by the increment learning to reach reasonable network configurations, which is investigated in Sec. V-E.

### C. COMPARISON WITH STATE-OF-THE-ARTS

We considered several state-of-the-art methods, including: H-First [20], HOTS [33], HATS [52], and SNN [1], [40]. For H-First we used the code provided by the authors online. For HOTS [33] and HATS [52], we used the results reported in [52]. These researchers used a linear SVM for classification. Given that no code was available for the SNN, we compared our results with two-layer SNN architecture using predefined Gabor filters [5] and the result is reported in [52]. For a fair comparison, we used our asynchronous spatio-temporal mapping method and the original BLS model, the results were reported in [18]. The original BLS model was solved using a ridge regression approximation and the output weights $W$ were computed through the pseudoinverse computation in the training. Our full model adopted the end-to-end learning method by the gradient descent algorithm, which is described in Sec. IV-C.

The results for the N-MNIST, N-Caltech101, MNIST-DVS, CIFAR10-DVS, and N-CARS datasets are provided in Table 1. We reported the results in terms of classification accuracy in different datasets compared with the ground truth. It was observed that our method has the highest classification accuracy in N-Caltech101, MNIST-DVS, CIFAR10-DVS, and N-CARS datasets. Particularly, our method achieves the large margin improvement in the more challenging datasets (i.e. N-Caltech101, CIFAR10-DVS, and N-CARS). From the results, we found that the large datasets, such as the N-CARS dataset, is 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 delivers a competitive performance. This may be because the HATS method also adopts an exponential kernel and implements spatio-temporal regularization. However, the linear SVM classifier used limits the capacity to discriminate when the class of objects and data noise increase. We also observed that in N-MNIST and MNIST-DVS datasets, our model and HATS provide an equivalent performance, although our method outperforms HATS by approximately 8%, 6.4%, and 3.7% in N-Caltech101, CIFAR10-DVS, and N-CARS datasets, respectively.

### D. COMPARISON WITH CNN-BASED METHODS

We also compared our method with the CNN models. In the experiments, we chose LeNet5 [13], AlexNet [31], VGG-19 [51], Inception-V4 [53] and ResNet-34 [24] as

| Dataset       | H-First [20] | HOTS [33] | HATS [52] | Gabor-SNN | BLS [18] | Ours (full) |
|---------------|--------------|-----------|-----------|-----------|---------|-------------|
| N-MNIST       | 0.712        | 0.808     | 0.991     | 0.837     | 0.983   | 0.984       |
| N-Caltech101  | 0.054        | 0.210     | 0.642     | 0.196     | 0.668   | 0.722       |
| MNIST-DVS     | 0.595        | 0.803     | 0.984     | 0.824     | 0.987   | 0.989       |
| CIFAR10-DVS   | 0.077        | 0.271     | 0.524     | 0.245     | 0.563   | 0.588       |
| N-CARS        | 0.561        | 0.624     | 0.902     | 0.789     | 0.931   | 0.939       |
the baseline CNN-based networks. Given that the format of the required input for these CNNs models is frame-based, we grouped spike events to frame form over a random time segment of 30ms. We also supplemented training with heavy data augmentation to avoid over-fitting. As such, we resized input images such that the smaller side was 256 and kept the aspect ratio, then randomly cropped, flipped and normalized 224 × 224 spatial samples of each resized frame. All CNN-based models were trained from scratch using stochastic gradient descent with momentum set to 0.9 and $L_2$ regularization set to 0.1 × 10$^{-4}$, and the learning rate was initialized at 10$^{-3}$ and decayed by a factor of 0.1 every 10k iterations.

Table 2 provides a summary of the results for the CNN-based methods. It was observed that our proposed method outperforms all the other methods. ResNet-34 model has the best performance among all the compared CNN-based methods. Though our model and ResNet-34 achieve the similar results on the N-MNIST and MNIST-DVS datasets, our model outperforms the ResNet-34 model by 3.1%, 2.5%, and 3.0% on N-Caltech, CIFAR-10, and N-Cars datasets, respectively. In particular, we saw that our representation is more suited for object classification than existing handcrafted features, such as HATS and HOTS, even when we combined these features with more complex classification models. This is likely due to HATS discarding temporal information, which, as we established, plays an important role in object classification. It is important to note, that compared to the state-of-the-art, our method can still work at high frame rates which is sufficient for high-speed applications.

**E. MODEL ANALYSIS**

1) **INCREMENTAL LEARNING**

Table 5 presents the performance of the one-shot broad learning network and three different dynamic incremental constructions for the N-Caltech dataset. Here, we tested the classification ability using different incremental strategies. The one-shot network was set as 10 × 10 feature nodes and 5000 enhancement nodes. We did not use the incremental strategy in each update, but rather used a new broad learning network trained from inception. The number of enhancement nodes was increased by 2000. The first incremental network was then set as 10 × 6 feature nodes and 5000 enhancement nodes. Then, the feature nodes were increased dynamically at 10 each. The second incremental network was also set as 10 × 10 feature nodes and 1000 enhancement nodes. Each enhancement node was increased to 1000. The third incremental network was set as 10 × 6 feature nodes and 1000 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 1000 each. The results of each update are provided in Table 5. We observed that the incremental versions have the similar performance compared to the one-shot construction. The performance of the first and the third incremental strategy was slightly lower than those of the one-shot network.

2) **PARAMETER R AND Δ incr STUDY**

We tested the influence of the length of kernel $R$ and the interval time $Δ incr$ to the classification accuracy. The visualization of the PFM outputs $X$ according to different parameters is shown in Fig. 5, which demonstrates the PFM results in different parameter $Δ incr$ and $R$ settings. It was observed that the local neighboring information could not be fully utilized when $R$ is small. The edges are obscure and dim, which consists of the dotted lines; however, the influence of noise may be amplified in the texture and background when $R$ is large. The influence of $R$ on the classification accuracy is shown in Table 3. For each sample, events within a fixed time interval were randomly extracted to input to our object classification framework. In this study, we tested under various time intervals (i.e., 50, 100, and 200 milliseconds), to see their effect on accuracy and computation. The results are shown in Fig. 5. When extracting 100ms events from one sample, the model achieves the highest accuracy. Therefore, we opted for this setting in our report. Actually, there are the best parameters $R$ and $Δ incr$ configurations for different datasets. For easy applying in experiments, we used the $R = 7$ and $Δ incr = 100ms$ for all datasets.

3) **KERNEL STUDY**

In the experiments, we found that the kernel function makes an important role in the classification system. This maybe

---

**TABLE 2. Comparison with CNN-based methods on five datasets.**

|        | LeNet [13] | AlexNet [31] | VGG-19 [51] | Inception-V4 [53] | ResNet-34 [24] | BLS [18] | Ours |
|--------|------------|--------------|-------------|-------------------|----------------|---------|-----|
| N-MNIST| 0.964      | 0.974        | 0.973       | 0.973             | 0.984          | 0.983   | 0.984|
| N-Caltech101 | 0.520      | 0.542        | 0.561       | 0.578             | 0.691          | 0.668   | 0.722|
| MNIST-DVS | 0.952      | 0.967        | 0.981       | 0.985             | 0.983          | 0.987   | 0.989|
| CIFAR10-DVS | 0.316      | 0.329        | 0.344       | 0.379             | 0.563          | 0.563   | 0.588|
| N-CARS  | 0.687      | 0.705        | 0.732       | 0.864             | 0.909          | 0.931   | 0.939|

**TABLE 3. Classification accuracy in different kernel sizes on the N-Caltech101 dataset.**

| Kernel size | $R = 3$ | $R = 7$ | $R = 11$ | $R = 15$ |
|-------------|---------|---------|---------|---------|
| Accuracy    | 0.767   | 0.792   | 0.759   | 0.742   |

**TABLE 4. Comparison of classification accuracy using different kernel functions.**

| Kernel        | Alpha | Trilinear | Exponential | Ours |
|---------------|-------|-----------|-------------|------|
| N-MNIST       | 0.972 | 0.982     | 0.978       | 0.984|
| N-Caltech101  | 0.697 | 0.709     | 0.711       | 0.722|
| MNIST-DVS     | 0.967 | 0.961     | 0.971       | 0.989|
| CIFAR10-DVS   | 0.544 | 0.573     | 0.572       | 0.588|
| N-CARS        | 0.908 | 0.915     | 0.910       | 0.939|
The influence of parameters $\Delta_{incr}$ and $R$. The first row is the ASTM output in $R = 7$ and different $\Delta_{incr}$, the second is the ASTM output in $\Delta_{incr} = 100ms$ and different $R$. The details can be observed when zoom in.

**TABLE 5.** Classification accuracy on the N-CIFAR10 dataset with three incremental strategies. Our baseline model BLS has a fixed structure for the one-shot test. The first incremental strategy is the increment of feature nodes (F. N.). The second one is the increment of enhancement node (E. N.). The third one is the increment of feature nodes and enhancement nodes at the same time (F. N. & E. N.).

| Incremental Strategy | Feature Node | Enhancement Node | Epoch | Acc. |
|----------------------|--------------|------------------|-------|------|
| One-shot             | 100          | 5000             | 50    | 0.722|
| F. N.                | $60^{100}$   | 5000             | $10 + 4 \times 10$ | 0.716|
| E. N.                | 100          | $1000 \rightarrow 5000$ | $10 + 4 \times 10$ | 0.725|
| F.N. & E.N.          | $60^{100}$   | $1000 \rightarrow 5000$ | $10 + 4 \times 10$ | 0.719|

because the kernel decides which kind of feature information is extracted from the event stream. Table 4 shows the results of different kernel setting. The compared kernel functions include Alpha, Trilinear, and Exponential, which are described in Sec. III-B. For an effective comparison, we did not add the peak-and-fire mechanism in three kernel models. Our model in Table 4 adopted the complete PFM method. Because our model is also a variation of the exponential kernel, we can conclude that the performance of our full model outperforms the exponential kernel model without the peak-and-fire mechanism. This mechanism is able to exclude the influence of noise in the event stream. The advantage became clear in the N-CARS and N-Caltech101 datasets. Also, we can see that the difference of three kernels is not large, because all three kernels encode the temporal information, which is key to the event-based classification task.

**TABLE 6.** Complexity (FLOPs) and Size (MB) of the compared networks.

| Network                | FLOPs (Giga) | Size (MB) |
|------------------------|--------------|-----------|
| LeNet5 [13]            | 0.05         | 5.41      |
| AlexNet [31]           | 0.71         | 57.04     |
| VGG-19 [51]            | 19.63        | 143.65    |
| Inception-V4 [53]      | 12.25        | 42.62     |
| ResNet34 [24]          | 3.68         | 21.29     |
| BLS (Ours)             | 0.01         | 5.35      |

4) COMPLEXITY ANALYSIS

Table 6 shows the time comparison between our method and the CNNs models, i.e., the LeNet5 [13], AlexNet [31], VGG-19 [51], Inception-V4 [53], and ResNet34 [24]. We analyzed the model complexity of the broad learning system. We used two main metrics to measure model complexity; FLOPs and Parameters. The FLOPs are the number of floating point operations, and the parameters are the number of parameters within the model. For the full connection layer, we computed FLOPs as $FLOPs = (2I - 1)O$, where $I$ is the input dimensionality and $O$ is the output dimensionality. With regards to the number of parameters, the FLOPs in fully connected layer is $(C_{in} + 1)C_{out}$. Table 6 summarizes the comparison of the network complexity. As one can see our proposed model has a smaller number of weights and requires less computation compared to deep CNN models. The main reason is that our model has only two-layers representation which is relatively compact, which in turn reduces the amount of data required because there is no convolution layer in our model, so our model requires fewer computation resource. For example, if our model is constructed by $10 \times 10$ feature nodes and 8000 enhancement nodes, the number of FLOPs is 0.01G Mac and the number of Parameters is 5.35K.

**VI. CONCLUSION**

In this study, we proposed an end-to-end broad learning network for event-based object classification. First, the peak-and-fire mechanism was adopted to map asynchronous events as input data for the broad learning network. We then designed a flexible broad learning network consisting of feature and enhancement nodes to handle asynchronous event data, thereby providing an alternative strategy for dealing with data from event-based cameras, except for CNNs and SNNs models. The incremental learning strategy of extending the broad learning network by adding feature and enhancement nodes during training is efficient and there is no need for preliminary training. In the experiments, our model outperformed state-of-the-arts. The complexity and parameters of our model are relatively small which provides distinct advantages over other models although several issues emerged...
wild event data.

proposed object classification method by using a peak-region

S. Gao

[17] G. Gallego, J. E. A. Lund, E. Mueggler, H. Rebecq, T. Delbruck, and

S. Gao, G. Guo, and C. L. P. Chen, “Event-based incremental broad

[14] S. Dehuri and S.-B. Cho, “A comprehensive survey on functional link

[12] X. Clady, J.-M. Maro, S. Barré, and R. B. Benosman, “A motion-based

[13] Y. L. Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard,

[32] X. Lagorce, S.-H. Ieng, X. Clady, M. Pfeiffer, and R. B. Benosman,

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification

[30] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from

[29] G. Orchard, C. Meyer, R. Etienne-Cummings, C. Posch, N. Thakor, and

[28] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger, “Densely

[27] J. A. Perez-Carrasco, B. Zhao, C. Serrano, B. Acha, T. Serrano-Gotarredona, S. Chen, and B. Linares-Barranco, “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.

[26] 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.

[25] G. Huang, S.-H. Ieng, X. Clady, M. Pfeiffer, and R. B. Benosman, “Spatiotemporal features for asynchronous event-based data,” Frontiers Neurosci., vol. 9, p. 46, Feb. 2015.

[24] X. Lagorce, S.-H. Ieng, X. Clady, M. Pfeiffer, and R. B. Benosman, “Spatiotemporal features for asynchronous event-based data,” Frontiers Neurosci., vol. 9, p. 46, Feb. 2015.

[23] J. H. Lee, T. Delbruck, and M. Pfeiffer, “Training deep spiking neural networks using backpropagation,” Frontiers Neurosci., vol. 10, p. 508, Nov. 2016.

[22] T. Goldstein, B. O’Donoghue, S. Setzer, and R. Baraniuk, “Fast alternating direction optimization methods,” SIAM J. Imag. Sci., vol. 7, no. 3, pp. 1588–1623, Jan. 2014.

[21] D. Gehrig, H. Rebecq, G. Gallego, and D. Scaramuzza, “Asynchronous, photometric feature tracking using events and frames,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 750–765.

[20] G. Orchard, C. Meyer, R. Etienne-Cummings, C. Posch, N. Thakor, and R. Benosman, “HiFirst: A temporal approach to object recognition,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 37, no. 10, pp. 2028–2040, Oct. 2015.

[19] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, “Converting static image datasets to spiking neuromorphic datasets using saccades,” Frontiers Neurosci., vol. 9, p. 437, Nov. 2015.

[18] H. Jhuang, T. Serre, L. Wolf, and T. Poggio, “A biologically inspired system for action recognition,” in Proc. IEEE 11th Int. Conf. Comput. Vis., Oct. 2007, pp. 1–8.

[17] J. A. Perez-Carrasco, B. Zhao, C. Serrano, B. Acha, T. Serrano-Gotarredona, S. Chen, and B. Linares-Barranco, “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.

[20] R. Etienne-Cummings, “Optimization methods for spiking neurons and an adaptive PSO-BP learning for CFLNN,” Neural Netw. Learn. Syst., vol. 30, no. 4, pp. 1191–1204, Apr. 2019.

[19] D. Gehrig, H. Rebecq, G. Gallego, and D. Scaramuzza, “Asynchronous, photometric feature tracking using events and frames,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 750–765.

[18] H. Jhuang, T. Serre, L. Wolf, and T. Poggio, “A biologically inspired system for action recognition,” in Proc. IEEE 11th Int. Conf. Comput. Vis., Oct. 2007, pp. 1–8.

[17] J. A. Perez-Carrasco, B. Zhao, C. Serrano, B. Acha, T. Serrano-Gotarredona, S. Chen, and B. Linares-Barranco, “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.

[16] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Proc. Conf. Workshop Neural Inf. Process. Syst., 2012, pp. 1097–1105.

[14] S. Dehuri and S.-B. Cho, “A comprehensive survey on functional link neural networks and an adaptive PSO-BP learning for CFLNN,” Neural Comput. Appl., vol. 19, no. 2, pp. 187–205, 2010.

[13] 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,” in Proc. Conf. Workshop Neural Inf. Process. Syst., 1990, vol. 2, pp. 396–404.

[12] S. Dehuri and S.-B. Cho, “A comprehensive survey on functional link neural networks and an adaptive PSO-BP learning for CFLNN,” Neural Comput. Appl., vol. 19, no. 2, pp. 187–205, 2010.

[11] C. L. P. Chen and Z. Liu, “Broad learning system: An effective and efficient incremental learning system without the need for deep architecture,” IEEE Trans. Neural Netw. Learn. Syst., vol. 29, no. 1, pp. 10–24, Jan. 2018.

[10] C. L. P. Chen, Z. Liu, and S. Feng, “Universal approximation capability of broad learning system and its structural variations,” IEEE Trans. Neural Netw. Learn. Syst., vol. 30, no. 4, pp. 1191–1204, Apr. 2019.

[9] C. L. P. Chen and J. Z. Wan, “A rapid learning and dynamic stepwise updating algorithm for the neural networks and the application to time-series prediction,” IEEE Trans. Syst. Man. Cybern. B, Cybern., vol. 29, no. 1, pp. 62–72, Feb. 1999.

[8] X. Clady, J.-M. Maro, S. Barré, and R. B. Benosman, “A motion-based feature for event-based pattern recognition,” Frontiers Neurosci., vol. 10, p. 594, Jan. 2017.

[7] 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,” in Proc. Conf. Workshop Neural Inf. Process. Syst., 1990, vol. 2, pp. 396–404.

[6] S. Cannici, M. Ciccone, A. Romanoni, and M. Matteucci, “Attention mechanisms for object recognition with event-based cameras,” in Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV), Jan. 2019, pp. 1127–1136.

[5] 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 Netw., vol. 32, pp. 339–348, Aug. 2012.

[4] M. Cannici, M. Ciccione, A. Romanoni, and M. Matteucci, “Attention mechanisms for object recognition with event-based cameras,” in Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV), Jan. 2019, pp. 1127–1136.

[3] R. Berner, C. Brandli, M. Yang, S. 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.

[2] R. Benosman, C. Clercq, X. Lagorce, S.-H. Ieng, and C. Bartolozzi, “Multichannel texture analysis using localized spatial filters,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 12, no. 1, pp. 55–73, Jan. 1990.

[1] A. Russell, G. Orchard, Y. Dong, S. Mihalas, E. Niebur, J. Tapson, and R. Etienne-Cummings, “Optimization methods for spiking neurons and networks,” IEEE Trans. Neural Netw., vol. 21, no. 12, pp. 1950–1962, Dec. 2010.
SHAN GAO received the B.S. degree in communication engineering from Nankai University, Tianjin, China, in 2010, and the M.S. and Ph.D. degrees from the University of the Chinese Academy of Sciences, Beijing, China, in 2013 and 2016, respectively. He was a Postdoctoral Researcher with the Automation Department, Tsinghua University, Beijing, in 2016. He joined the faculty at Northwestern Polytechnical University (NPU), in 2018, where he is currently an Assistant Professor with the Unmanned System Research Institute. His research interests include object detection and tracking, image processing, and multitemporal fusion.

GUANQIAN GUO is currently pursuing the degree with the School of Electrical and Control Engineering, Shaanxi University of Science and Technology. He is currently an Intern with the Institute of Unmanned Systems Technology, Northwestern Polytechnical University. His research interests include computer vision and deep learning, and weak supervision object localization.

HANQIAO HUANG received the B.E. and M.S. degrees from Air Force Engineering University, China, in 2003 and 2006, respectively, and the Ph.D. degree from Northwestern Polytechnical University, Xi’an, China, in 2010. He came out of the Postdoctoral station, in 2015. He is currently an Associate Professor with the Unmanned System Research Institute, Northwestern Polytechnical University. He is chairing six projects, including the National Natural Science Foundation. He has published more than 40 articles in well-known journals and international conferences, 12 of which were searched by SCI and 24 by EI. His current research interests mainly include signal processing, pattern recognition, visual tracking, and intelligent vision systems for unmanned air vehicles. He serves as a Reviewer and a Technical Committee Member in several international conferences and journals.

XUEMEI CHENG received the Ph.D. degree from Northwestern Polytechnical University, Xi’an, China, in 2000. She is currently a Professor with Northwestern Polytechnical University. Her current research interests mainly include self-control, navigation, and intelligent vision systems for unmanned air vehicles.

C. L. PHILIP CHEN (Fellow, IEEE) received the Ph.D. degree from the University of Michigan at Ann Arbor, Ann Arbor, MI, USA.

Being a Program Evaluator of the Accreditation Board of Engineering and Technology Education (ABET), Baltimore, MD, USA, for computer engineering, electrical engineering, and software engineering programs, he successfully architects the University of Macao Engineering and Computer Science programs receiving accreditations from Washington/Seoul Accord through the Hong Kong Institution of Engineers (HKIE), Hong Kong, which is considered as his utmost contribution in engineering/computer science education for Macau as the Former Dean of the Faculty. His current research interests include systems, cybernetics, and computational intelligence. He is a Fellow of the American Association for the Advancement of Science, the International Association for Pattern Recognition, the Chinese Association of Automation (CAA), and HKIE.

He received the 2016 Outstanding Electrical and Computer Engineers Award from his alma mater, Purdue University. He was the Chair of the TC 9.1 Economic and Business Systems of the International Federation of Automatic Control, from 2015 to 2017. From 2012 to 2013, he was the President of the IEEE Systems, Man, and Cybernetics Society. He is currently the Vice President of CAA. He is the Editor-in-Chief of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS and an Associate Editor of the IEEE TRANSACTIONS ON FUZZY SYSTEMS and the IEEE TRANSACTIONS ON CYBERNETICS.