Attention Mechanisms for Object Recognition with Event-Based Cameras

Marco Cannici   Marco Ciccone   Andrea Romanoni   Matteo Matteucci
Politecnico di Milano, Italy
{marco.cannici,marco.ciccone,andrea.romanoni,matteo.matteucci}@polimi.it

Abstract

Event-based cameras are neuromorphic sensors capable of efficiently encoding visual information in the form of sparse sequences of events. Being biologically inspired, they are commonly used to exploit some of the computational and power consumption benefits of biological vision. In this paper we focus on a specific feature of vision: visual attention. We propose two attentive models for event based vision: an algorithm that tracks events activity within the field of view to locate regions of interest and a fully-differentiable attention procedure based on DRAW neural model. We highlight the strengths and weaknesses of the proposed methods on four datasets, the Shifted N-MNIST, Shifted MNIST-DVS, CIFAR10-DVS and N-Caltech101 collections, using the Phased LSTM recognition network as a baseline reference model obtaining improvements in terms of both translation and scale invariance.

1. Introduction

Convolutional neural networks (CNNs) are currently the state of the art in a variety of challenging computer vision tasks that involve the extraction of visual features. These include, among the others, image classification [28, 7, 26], object detection [22, 21, 8, 13] as well as semantic scene labeling [12, 2, 11]. When video sequences captured by conventional frame-based cameras are considered, CNNs great performance in terms of accuracy, however, is achieved at the cost of a high computational and time complexity. Indeed even while capturing static scenes, these devices output a stream of mostly identical frames, requiring CNNs to process the same redundant data several times.

On the other hand, primates are able to achieve remarkable results in most vision tasks while using a fraction of energy and computational power with respect to their artificial counterparts. As an attempt to reproduce the benefits of biological vision, research is now focusing on developing vision systems based on neuromorphic, or event-based, cameras, *i.e.*, a type of sensors that tries to emulate the functioning of biological retinas. Unlike conventional cameras,

these devices output sequences of asynchronous events that efficiently encode pixel-level brightness changes caused by objects moving inside the scene. The result is a sensor able to produce a stream of events $e = \langle x, y, ts \rangle$ indicating the time instant ts, the position (x, y) and the polarity $p \in \{-1, 1\}$ of every change detected inside the scene.

A key characteristic of biological vision systems is their ability to selectively focus their attention on the salient portions of the scene, drastically reducing the amount of information that needs to be processed. Selective attention mechanisms that mimic this behavior are nowadays widely adopted in several vision tasks, like for instance in image and video captioning [29, 3, 4], image generation [6], object recognition [15, 27] and person re-identification [24]. A similar effort has been made in the design of attention mechanisms able to directly process event-based information produced by neuromorphic cameras [25, 20]. These devices are indeed inherently able to detect relevant portions of the field of view as they emit events only when something changes. Events encode important information regarding the objects contained inside the scene and can thus be used to precisely locate them in space and time.

These neuromorphic systems make often use of Spiking Neural Networks (SNNs) [14], a type of artificial neural networks based on units that communicate with each other through spikes and perform computation only when and where needed. However, a big limitation of these models is that they are not differentiable. When multiple processing layers are involved, this makes the training procedure much more complex than the back-propagation algorithm used in conventional neural networks. For this reason, another approach adopted in literature makes use of conventional convolutional or recurrent networks properly adapted to handle event based information [18, 16, 1]. Despite being easier to train, however, such networks usually require the scene to be reconstructed as a sequence of frames, thus potentially ignoring all the advantages of the events encoding.

In this paper we focus on enhancing conventional architectures by designing attention mechanisms that can be used to make these networks focus only on relevant instants of events recordings and only on the salient portions of frames,

limiting the increased data redundancy caused by the frame integration process.

The main contributions of this paper are:

- An algorithmic attention mechanism which monitors the events activity within the scene to extract patches from reconstructed frames (Section 3).
- An adaptation of the popular DRAW [6] attention mechanism for image classification able to recognize objects within reconstructed frames (Section 4.1).
- An event-based variant of the previous network which directly uses events to locate the relevant portions of the frame (Section 4.2).

2. Background

This section presents three basic tools adopted to design the attention mechanisms proposed in this paper: the Phased LSTM Network, the DRAW attention mechanism and the Leaky Frame Integrator.

Phased LSTM Recognition Network The Phased LSTM recognition network [16] is a simple architecture for object classification with event-based cameras. It is based on Phased LSTM (pLSTM) cells, a variant of the vanilla LSTM which makes use of a *time gate* to learn the time scales of incoming events, and uses of a word embedding layer to extract relevant features from a stream of events. Its structure is depicted in Figure 1.

Despite achieving good results on simple datasets, however, the network lacks in the ability to extract general features as its embedding layer is only able to learn simple mappings between coordinates and learned sets of features. This results in a model with poor translation and scale invariance properties. This paper focuses on improving its performance in conditions where objects may appear with multiple scales and in different portions of the field of view.

DRAW Selective Attention The *Deep Recurrent Attentive Writer (DRAW)* [6] is a network for image generation that makes use of a novel fully-differentiable procedure to focus attention on the salient portion of a frame. Its core components are a *recurrent neural network* (RNN), usually an LSTM, and the selective attentive operator *read*.

The *read* operator is used to force the network to only see a certain portion of the original frame. Using the abstract representation encoded by the RNN, the parameters of a grid of 2D Gaussian filters are first computed and then used to extract a $N \times N$ patch of the image. The final patch is obtained through a fixed number of progressive refinements in which the RNN, starting from the whole frame at the beginning, progressively modifies its previous representation to better zoom on the salient portion of the image. Varying the stride and variance of the filters, the network can adaptively enlarge or reduce its field of view while still extracting patches of a fixed dimension.
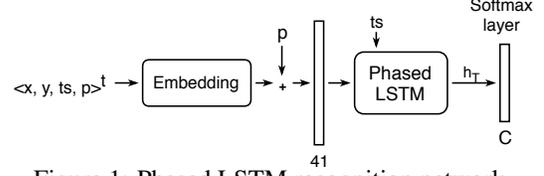

Figure 1: Phased LSTM recognition network.

More specifically, denoting as $\mathbf{h}_t$ the output of the RNN at the time t, a patch is extracted as it follows:

$$read(\mathbf{x}, \mathbf{h}_t^{dec}) = \gamma \left(\mathbf{F}_Y \, \mathbf{x} \, \mathbf{F}_X^T \right) \quad (1)$$

where $\mathbf{F}_Y$ and $\mathbf{F}_X^T$ (with dimension $N \times H_f$ and $W_f \times N$ respectively) are the Gaussian filters obtained using a linear transformation of $\mathbf{h}_t$, x is the original $H_f \times W_f$ frame, and γ is a scalar.

Even if originally designed for image generation, this procedure can also be used as an attention mechanism in object recognition architectures. Please refer to the original DRAW paper [6] for a detailed description of the model.

Leaky Frame Integration All the attention mechanisms designed to improve the pLSTM Recognition Network proposed in this paper are based on the frame reconstruction procedure described in [1]. This simple mechanism, inspired by the functioning of spiking neurons, integrates events in time producing a sequence of frames on which conventional computer vision techniques can be applied. The pixel values of the reconstructed frame are updated whenever a new event $e = (x_e, y_e, ts)^t$ arrives, as it follows:

$$q_{x_m,y_m}^t = max(p_{x_m,y_m}^{t-1} - \lambda \cdot \Delta_{ts}, 0) \quad (2)$$

$$p_{x_m,y_m}^t = \begin{cases} q_{x_m,y_m}^t + \Delta_{incr} & if (x_m, y_m)^t = (x_e, y_e)^t \\ q_{x_m,y_m}^t & otherwise \end{cases}, \quad (3)$$

where $\Delta_{ts} = ts^t - ts^{t-1}$ decrements the whole frame of a quantity that depends on the time elapsed between the last received event, ts^t, and the previous one. As in the original YOLE paper [1], we fix $\Delta_{incr} = 1$, varying only λ based on the dataset to be processed and in particular on the speed at which objects move.

3. Patch Extractor Recognition Network

By monitoring the events activity inside the field of view of the neuromorphic camera, regions of interest can be detected and used as candidates for the object recognition process. For this purpose, we developed an algorithm that detects peaks of events activity and uses them to extract patches from reconstructed frames. This approach takes inspiration from the spiking recognition network proposed in [30], where a peak detection mechanism is used to decide
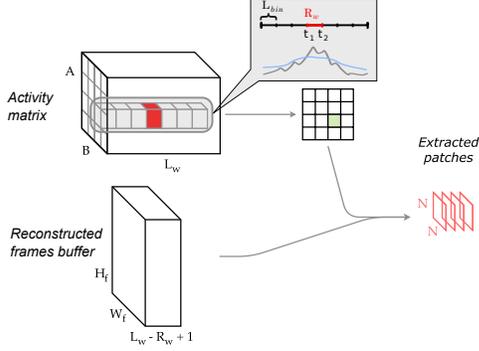
Figure 2: The detection of a peak in position R_w and the confidence interval, as a blue line, are represented at the top of the figure. For simplicity, windows are grouped together into an *activity matrix*.

when to output predictions. Instead of leaky *integrate-and-fire* neurons, however, our method makes use of region-wise events statistics to identify and localize peaks.

3.1. Peak Detection Algorithm

The Peak Detection Algorithm we designed subdivides the $H_f \times W_f$ field of view into a grid of possibly overlapping $H_r \times W_r$ regions spaced by a fixed stride s_r. A moving window in time of length L_w is associated to each tile; each *activity value* of L_w represents the number of events received inside the region within a certain interval of length L_{bin}. These *activity windows* are used to detect peaks of activity inside each region by comparing the value contained in a fixed position R_w of the window, which we call *representative value*, with the remaining activity values in the same window. As time passes, each activity value slides through the activity window and therefore, at some time, each value becomes the representative value R_w. We usually set R_w to be the middle point in the window, but other configurations are also possible.

Periodically, each activity window is checked in order to determine the presence of peaks. A peak of activity is detected in a certain region whenever R_w becomes the maximum value inside the window. In this case, the interval (t_1, t_2), with $t_2 = t_1 + L_{bin}$, corresponding to the representative value is considered a peak and a *patch extraction algorithm* is used to extract $N \times N$ patches inside this region using the frame reconstructed at the time instant t_2.

Since R_w is usually not the first element of the window, the algorithm must wait the following $L_w - R_w + 1$ intervals before (t_1, t_2) becomes the representative value and can consequently be analyzed. Being the peak detection delayed of $L_w - R_w + 1$ intervals, a buffer of integrated frames must be maintained to allow the extraction of patches from the right frame.

Every time a new event $\mathbf{e} = (x_e, y_e, ts_e)$ arrives, the frame in the most recent position of the buffer is updated,

Figure 3: Comparison between patches extracted with the two versions of the patch extraction algorithm. **(a)** The follower and **(b)** centered variants.

as described in Section 2, as well as all the activity windows associated to regions in which e is contained. Peak detection is only performed if the current interval has finished (*i.e.*, ts_e is more that L_{bin} time instants after the beginning of the current interval) and at least L_w activity values have been accumulated. In this case all the activity windows are checked and patches are extracted whenever a peak is detected. At the end of this process, the oldest frame in the buffer and the oldest values in all activity windows are removed to make room for the next interval.

To avoid false detections caused by noisy events received during time intervals of poor events activity and increase the robustness of the algorithm, we enhanced the peak detection procedure with a moving average approach. A peak is considered to be valid if its value x is above the confidence interval $x > \mu^t + \alpha \cdot \sigma^t$ where α is a parameter and μ^t, σ^t are the mean and standard deviation statistics of the whole field of view.

These are updated at the end of each interval as it follows:

$$\mu^t = \frac{sum_{val}}{N_{val}}, \quad \sigma^t = \sqrt{\frac{sum_{val^2}}{N_{val}} - (\mu^t)^2} \quad (4)$$

where sum_{val} and sum_{val^2} are respectively the sum of the activity values and the sum of their squares, and $N_{val} = N_{int} * A * B$, with N_{int} the number of processed intervals. Both sum_{val} and sum_{val^2} are incrementally updated at the end of each interval. The equation of σ^t is obtained from the relation between the mean and the variance of a stochastic variable, namely $Var[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$.

3.2. Patch Extraction Algorithms

We developed two mechanism for patches extraction. One that covers the whole object by centering a patch on activated regions, which we called *Centered Patch Extraction*, and the other one which instead extracts small details by following the contours of the objects, which we called *Follower Patch Extraction*. Examples of patches extracted with these two methods are shown in Figure 3. A video showing the detection of peaks and the extraction of patches on similar event-based recordings is available at https://youtu.be/BV_ikdS4m3g.
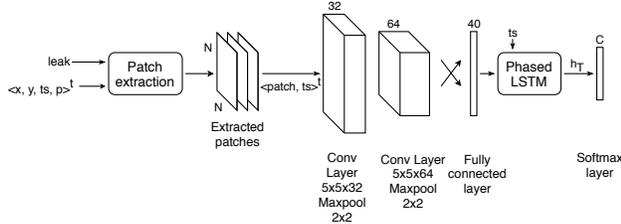
Figure 4: The convolutional network used to classify the sequence of extracted patches.

Centered Patch Extraction The result provided by the peak detection unit is a two-dimensional boolean matrix that indicates which regions of the $A \times B$ grid activated, *i.e.*, in which of these regions a peak has been detected. The goal of the Centered Patch Extraction algorithm is to extract patches which cover as much as possible the detected object. For this reason, active regions are grouped into *macro-regions* by joining together adjacent active regions. For each macro-region one or multiple equally spaced $N \times N$ patches are extracted by covering the entire activated region. This procedure is performed for every group of active regions and all the extracted patches are labeled with the timestamp associated to the frame from which they have been extracted.

Follower Patch Extraction In the Follower variant of the Patch Extraction algorithm we choose the dimensions of the patches so that only small object details are extracted. We then extract patches by following the object outline with a simple recursive algorithm that extracts a patch as soon as an uncovered object pixel is visited. As for the centered version of the algorithm, the timestamp of the frame from which patches have been obtained is also saved.

3.3. Classification network

The sequence of extracted patches constitutes the input of the recognition network that uses the timestamp information to correlate patches over time by means of a pLSTM layer (Figure 4). The network is similar to the original pLSTM recognition network from [16], where the word embedding layer has been replaced with a convolutional neural network. We used feature vectors of the same length of the ones extracted by the original embedding layer. However, no polarity information is added in this case.

The structure of this network is based on the idea that the patch extraction algorithm can be used as a way to convolve filters sparsely in space and time, driven by the events activity. Each extracted patch can indeed be considered as a single receptive field on which a small portion of a wider convolutional network, that potentially covers the whole input frame, is applied. The patch extraction algorithm, by monitoring the events activity during time and selecting the active receptive fields, allows to compute an event-based convolution of the filters only when and where a peak of

activity has been detected. Features extracted from these receptive fields are then used by the pLSTM to reconstruct the global appearance of the object and its output is finally used for the overall prediction. We used the same network with both versions of the patch extraction algorithm.

4. N-DRAW Recognition Network

The patch extraction algorithm we presented in the previous section effectively extracts patches from integrated frames coming from neuromorphic cameras. The fact that patch extraction is driven by the events activity and that patches are computed and analyzed only when enough information has been accumulated, make the algorithm fit well in event-based scenarios.

However, both patch extraction networks require the tuning of dataset-specific parameters and are not able to adapt to objects of variable dimension. To improve the performance of this network, which still achieve better results than the pLSTM baseline when objects are not centered in the filed of view, we extended the patch extraction network obtaining a trainable procedure based on DRAW [6] and whose functioning is similar to the original procedure. Being designed on top of a recurrent neural network that gradually encodes visual information and being able to gradually adjust its predictions over time, DRAW naturally fits the sequential nature of event-based imaging. We finally used the networks presented in the previous sections as additional baselines to evaluate how much the network performance improves when a patch extraction procedure able to automatically adjust to changes is used.

We designed the *N-DRAW patch-based* network by combining the architecture of the previous patch extraction algorithm with the DRAW recognition model. Then, we designed a second variant, *i.e.*, the *N-DRAW event-based* network, that directly uses the sequence of events as input to the encoder network.

4.1. Patch-based model description (p-N-DRAW)

We modified the original DRAW network to detect objects captured with event-based cameras by using a frame reconstruction mechanism as the first layer of our architecture (Figure 5). The *read* operation takes as inputs the most recent frame $frame^t$ and the output of the encoder at the previous time step $\mathbf{h}_{enc}^{t-1}$. This extracts the parameters of 2D Gaussian filters and uses them to transform the $A \times B$ input frame into a fixed size $N \times N$ patch. The timestamp ts^t associated with the current frame is used as an additional input for the recurrent network. In contrast to the original architecture that uses a simple LSTM network, we use a pLSTM layer as the encoder so that the timestamp associated with each patch can also be exploited. By doing that, the network learns to sparsely update its internal representation based on the timing of the input features.
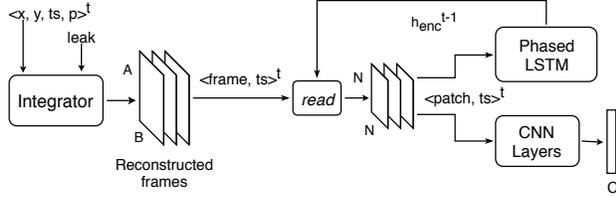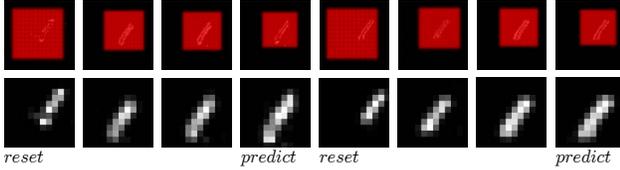
Figure 5: The p-N-DRAW recognition network.

reset   *predict*   *reset*   *predict*

Figure 6: The top row shows 8 consecutive 68×68 frames and, in red, the grid of 12×12 2D Gaussian filters, whereas the bottom one the corresponding extracted patches. In this example, the encoder state is reset every 4 frames, which causes the filter to reposition itself to cover most of the frame.

Differently from the original model, where the image is static, we deal instead with a sequence of integrated frames that may slightly differ from each other. The *read* operation, therefore, has to decide where to attend at the current time step by using the encoder output produced while observing the previous frame, where the object may be in a slightly different position. We found, however, that this does not constitute a problem for the recurrent architecture since it can learn to compensate the objects movement by comparing consecutive frames.

We want our network to be able to recognize objects as soon as enough information has been accumulated. For this reason, we decided to perform a prediction regularly rather than after having seen the whole sequence, as opposed to the standard DRAW architecture. Since using every patch for prediction may prevent the network to learn a good extraction mechanism (not having the encoder a fixed reasoning period which can be used to gradually zoom and refine the prediction), we perform instead a prediction every M successive frames.

If M is not too large (we used $M = 4$ in our experiments), the network can still generate predictions quite often allowing the model to be used for continuous classification. After the fixed M steps, the internal state of the encoder can either be reset or maintained as a starting point for the next prediction. We found to be beneficial to maintain the internal state when objects do not move too much, as the network can continue to refine the previous prediction. However, if objects move fast the network performs better when the state is reset, as this allows it to see the whole frame and progressively re-locate the object. An example is shown in Figure 6.
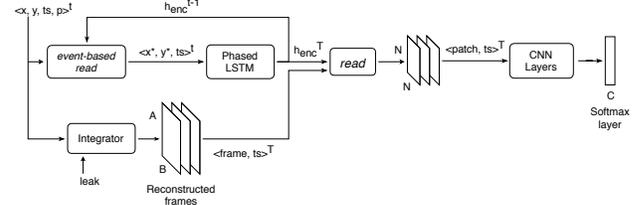

Figure 7: The e-N-DRAW architecture. The projected coordinates (x_e^*, y_e^*) are used to guide the attention mechanism in finding the filters parameters (loop connection at the top of the figure).

4.2. Event-based model description (e-N-DRAW)

N-DRAW event-based extends the DRAW attention mechanism to directly process the stream of incoming events and uses it as a reference to locate the relevant part of the scene, in a similar way as in the patch extraction algorithm. This variant, depicted in Figure 7, makes use of two read operations: *event-based read*, the modified attention mechanism that processes events, and *read*, the vanilla DRAW's operation that extracts patches from reconstructed frames. The sequence of events is partitioned into *intervals* of equal temporal length T. Events are used both to reconstruct frames through the frame integration procedure and to detect the relevant part of the scene by means of the recurrent pLSTM layer. Once the whole sequence of events inside the interval has been processed by the encoder, its output $\mathbf{h}_{enc}^T$, is used to extract a $N \times N$ patch from the last integrated frame $frame^T$ using the standard extraction procedure *read*. The extracted patch is then processed as usual by applying a sequence of convolutional layers and by using the extracted representation to predict the class label.

The *event-based read* projects the input coordinates into the *patch space* to provide the encoder network a feedback on the transformation applied by the Gaussian filters. Given an event at location (x_e, y_e) in the input space, the *event-based read* produces as output a new event with the same timestamp ts, but having as coordinates (x_e^*, y_e^*) those that the original event has in the $N \times N$ patch space. In this way the encoder network progressively follows the events activity and modifies its internal state so that the extracted patch will be centered on the object.

The operation performed by the *event-based read* procedure is based on the original transformation. A frame containing a single positive pixel in correspondence of the incoming event coordinates (x_e, y_e) is considered. A patch is extracted from this frame by means of the original *read* operation using the set of Gaussian filters obtained from the encoder output at the previous step $\mathbf{h}_{enc}^{t-1}$. This patch contains a possibly blurred dot in a certain location whose coordinates (x_e^*, y_e^*) can be obtained by looking at the brightest pixel of the patch. In particular, the output coordi-

nates are defined as $(x_e^*, y_e^*) = \arg\max_{(x,y)} \gamma \left(\mathbf{F}_Y \, \mathbf{x} \, \mathbf{F}_X^T \right)$, where $\mathbf{F}_Y$ and $\mathbf{F}_X$ are the set of extracted Gaussian filters. The *event-based read* and *read* operations share the same linear transformation that allows the encoder output to be transformed into the filter parameters. In this way, the same transformation learned while observing the sequence of events will be also used to extract the actual patch from the reconstructed frame.

As it happens with the original attention mechanism, the network starts by considering a patch that roughly covers the whole frame so that most of the incoming events will be contained in the extracted patch. As more events arrive, the network becomes more confident about the position of the object and it starts reducing the dimensions of the filter ignoring irrelevant events. Events that are not contained inside the filter's region, *i.e.*, those for which the patch extracted with the *event-based read* operation is completely blank, are ignored by the network that skips them during the recursive execution. This is in line with the original DRAW patch extraction procedure which ignore the part of the scene not contained inside the extracted patch. Figure 8 shows the successive stages of the *event-based* patch extraction procedure on some Shifted N-MNIST examples.

This attention procedure resemble the patch extraction algorithm we presented in the previous section; the spatial location of the events is directly used to discover regions of interest in the input scene which are then used to extract patches from integrated frames. This new mechanism has the advantage of being a trainable procedure that can be learned together with the rest of the classification network thanks to its fully-differentiable nature. As for the patch extraction algorithm, though, this mechanism only bases its predictions on the events activity, without any visual feedback, as opposed to the *patch-based* architecture. This characteristic limits the network performance in scenarios in which the background is also moving with respect to the camera and the attention mechanism has to discriminate between events emitted by the object and those emitted by the rest of the scene, as we registered testing the network on CIFAR10-DVS [10] and N-Caltech101 [17] datasets. Moreover, we found that this network has difficulties in centering and zooming on the object with respect to the *patch-based* one, as reported in the next section.

5. Experiments

Datasets The performance of the proposed attention-based networks have been tested on four datasets available in literature, the N-MNIST [17], MNIST-DVS [23], CIFAR10-DVS (CIF10) [10] and N-Caltech101 (Cal101) [17] collections. These datasets have been obtained by displaying original images in front of a neuromorphic camera and by moving them, or the camera itself, following a predefined trajectory that resemble human saccades.
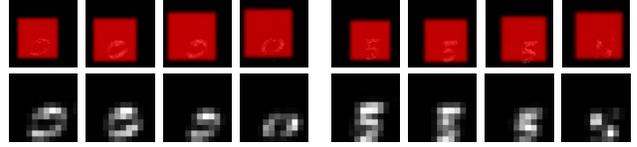

Figure 8: The filter changes during the successive stages of the patch extraction process on two Shifted N-MNSIT examples. Contrary to Figure 6, we do not see the filter gradually zooming in, since in this case the network uses the sequence of events to progressively refine its prediction. When the filter is first applied to the frame, it has already been perfected.

Table 1: Patch extraction algorithm parameters.

				S-DVS			S-N	CIF10	Cal101
		sc4	sc8	sc16	sc4+8	all			
centered	s_r	11	24	24	24	24	5	10	10
	$W_r=H_r$	24	32	32	32	32	23	48	48
	N	29	55	105	55	105	29	105	105
follower	s_r	5	15	24	24	24	5	12	12
	$W_r=H_r$	9	23	32	32	32	9	32	32
	N	13	23	53	23	53	13	75	75

Since these MNIST datasets are quite simple, especially N-MNIST, we also considered the *Shifted N-MNIST* (S-N) and *Shifted MNIST-DVS* (S-DVS) variants [1] in which the original digits (*i.e.*, the sequence of events representing them) are placed in a random location of a bigger field of view.

5.1. Experiments Setup

All the results presented in this paper were obtained by optimizing the cross entropy loss function using Adam [9] with default parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$) and learning rate 10^{-4}. Networks parameters were initialized using the mechanism proposed in [5] and early-stopping was applied to prevent overfitting.

Patch extractor networks Due to the high number of parameters on which the patch extraction algorithm depends, we decided to fix part of their values manually by inspecting the quality of the results produced by the patch extraction process, *i.e.*, the sequence of extracted patches. Even though this approach does not provide a complete exploration of the space of possible values and does not allow to compare the recognition performances of the final model in response to the change of every single parameter, it enabled us to quickly analyze their effects and determine the way these parameters interact with each other.

The parameters of the activity windows were chosen by analyzing the rate of events generated from the event-based camera during the entire recording period. As reported in [17], N-MNIST peaks of activity are correlated with the speed of the objects moving inside the scene. Since the same set of movements has been used to record the entire dataset, the activity outlines are very similar between ex-

amples. For these reasons we used $L_w = 101$, $R_w = 51$ and L_{bin} = 1ms, obtaining an activity window that covers 101ms, which is roughly the temporal length of the saccade movement used to record digits. We used the same parameters also with N-Caltech101 samples, as they have been recorded with the same procedure, and with CIFAR10-DVS recordings. MNIST-DVS digits, instead, being registered using a different and more noisy procedure, do not show a clear activity outline. To cope with the higher variability of event peaks and make a more reactive detection, we decided to use a smaller window with parameters $L_w = 81$, $R_w = 41$, and the same interval length $L_{bin} = 1$ms.

Regions parameters were chosen in order to cover a good portion of the objects and also to make regions overlap between each other to obtain good translation invariance properties. The patch dimension $N \times N$ was chosen in such a way to extract the entire object with the *centered* patch extraction procedure, and to cover only small details of each object in the *follower* variant. Table 1 reports the specific parameters we used.

N-DRAW networks The hyperparameters for the N-DRAW architectures were chosen using a "greedy" approach for parameter optimization. We focused on the optimization of a single parameter at the time by gradually changing its value and registering the resulting effect in network performance.

Since N-MNIST and scale4 (sc4) MNIST-DVS digits were recorded to have roughly the same dimensions of the original MNIST digits, we decided to use the same patch size, *i.e.*, 12×12, that was used with the original DRAW architecture [6] to process Cluttered MNIST [15] digits, a variation of MNIST in which digits are placed in a random location of a bigger frame, similarly to the Shifted N-MNIST and Shifted MNIST-DVS collections. Given that digits of a certain MNIST-DVS scale are roughly double the size of the previous scale, we used 24×24 and 48×48 patches for *scale8* (sc8) and *scale16* (sc16) examples. We finally used 48×48 patches also for N-Caltech101 and CIFAR10-DVS samples since they both feature objects that occupy most of the 128×128 frame, similarly to *scale16* MNIST-DVS samples.

The number M of recursive iterations was instead determined by using a simplified version of the network, which resembles the original DRAW classification network. We found $M = 4$ to be the optimal value.

Finally, to process the extracted patches, we used the same set of convolutional layers we used in the patch extractor architectures, *i.e.*, two convolutional layers and a fully connected layer that maps features into 40-dimensional vectors. The size of the encoder network was set to be equal to the number of cells used in the original pLSTM recognition network, *i.e.*, 110.

Table 2: pLSTM's baseline accuracy on the Shifted N-MNIST.

Frame	Embedding	Encoder	Augmented	Test Accuracy
34×34 (original)	41	110	No	97.4
68×68	41	110	No	26.0
68×68	101	200	No	81.7
68×68	101	200	Yes	90.3

Table 3: Comparison between the performances of the proposed models.

	S-DVS					S-N	CIF10	Cal101
	sc4	sc8	sc16	sc4+8	all			
pLSTM	82.20	87.01	81.60	86.60	83.63	90.30	17.10	1.39
p. centered	**98.30**	95.90	96.30	95.90	95.53	**97.37**	**44.10**	21.39
p. follower	91.30	90.50	95.10	-	-	91.07	37.40	18.47
e-N-DRAW	91.35	96.50	95.69	96.74	95.10	92.30	36.89	**28.95**
p-N-DRAW	94.81	96.88	95.32	**97.96**	93.19	96.42	38.17	27.69
p-N-DRAW (reset)	94.10	**97.39**	**96.71**	96.61	**98.24**	95.15	41.29	27.70

5.2. Results and Discussion

Baseline We compared the performance of the proposed models with the results obtained by the Phased LSTM object recognition network described in [16]. All the proposed networks are indeed based on pLSTM cells and they were originally designed to overcome some limitations of the original pLSTM model. Since the pLSTM architecture only uses an embedding layer to extract features from events, it does not show any scale or translation invariance property, as reported in Table 2. The loss in performance reduces when an augmented Shifted N-MNIST dataset, obtained randomizing the position of each digit after every epoch, and therefore using a higher number of training samples, is used to train the model.

Table 3 reports the results obtained on the Shifted N-MNIST, Shifted MNIST-DVS, CIFAR10-DVS and N-Caltech101 datasets, using the same layers configuration described in [16]. To reduce the size of the model (which depends on the frame size due to the presence of the embedding layer) and speed up training, we cropped the central portion of MNIST-DVS examples obtaining smaller samples containing only the digits. In particular we used 35×35, 65×65 and 100×100 field of views for the sc4, sc8 and sc16 examples respectively. In case of mixed scales we used the size of the bigger scale in the dataset.

Results Table 3 shows the results we obtained on multiple datasets using the proposed models. All the models achieve better results w.r.t. the pLSTM architecture, highlighting the advantages of using attention mechanisms to improve translation invariance.

As expected, the *follower* variant of the patch extraction network achieved worse results with respect to the *centered* version. While the use of smaller patches allows the network to maintain its event-based nature, reacting to small details as soon as they become visible, the task the pLSTM

layer needs to learn is much harder. The overall appearance of the object needs indeed to be reconstructed by only looking at the sequence of details, whose order is not always the same among objects of the same class since it depends on where and when peaks are detected. Note that we did not test the follower patch-extraction algorithm on mixed MNIST-DVS scales because, being patches of fixed dimensions, this would have meant to extract small details in sc16 samples, or the entire digit in sc4 samples.

The N-DRAW patch-based variant performed better than its event-based counterpart in almost all datasets. This difference in accuracy is explained by the fact that the encoder of the event-based architecture predicts the final set of filters parameters only based on the sequence of incoming events. This mechanism does not provide any visual cue regarding the effect that the set of predicted filters have on the actual extracted patch. However, even if it does not reach the same classification accuracies of the N-DRAW patch based algorithm, this mechanism still represents a valid event-based attention mechanism being able, by only using the events sequence, to identify regions of interests inside the scene. Using this procedure we obtained indeed similar results of the ones achieved using the centered patch extraction network in almost all datasets.

When evaluated on datasets composed of multiple scales, the N-DRAW architecture outperforms the patch extractor network but using a fully trainable model. N-DRAW allows indeed to adaptively zoom on the objects and enables the extraction of patches containing a reduced variability of objects dimensions. Large objects are extracted as they are whereas smaller ones are enlarged to better fit the patch. This behavior is depicted in Figure 9.

We also tested the proposed models on more challenging datasets in terms of both background noise and objects complexity, N-Caltech101 and CIFAR10-DVS. All the proposed networks achieve better results than the pLSTM baseline, showing advantages on the use of convolutional layers and attention mechanisms. The obtained results, however, do not achieve the state of the art accuracy obtained using the DART [19] descriptor ($65.43 \pm 0.35\%$ on CIFAR10-DVS and 65.6% on N-Caltech101). This lack of performance can be traced back to the poor capabilities of the proposed extraction algorithm to distinguish between background and foreground events in very noisy environments (where the moving average approach is not enough) and to the need of a greater number of samples to allow the trainable DRAW mechanism to learn effective filter transformations. When evaluating models on the two original image based collections, the lack of training samples is indeed usually addressed using pre-trained feature extractors obtained from larger collection of samples, which however are still missing in the neuromorphic field.
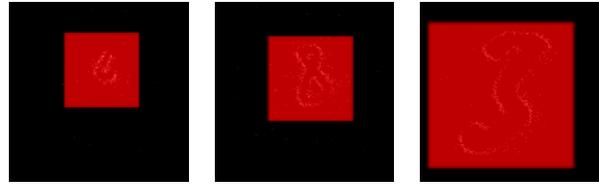

Figure 9: The DRAW attention procedure learned by a single network trained to recognize all MNIST-DVS scales.

6. Conclusions and Future Works

In this paper we proposed two approaches for event-based visual attention. The first one makes use of a simple algorithm to identify regions of interest from events while improving the translation invariance properties of the original pLSTM model. The second one is a fully-differentiable procedure based on the popular DRAW attention mechanism which improves the scale invariance properties of the first network. Using the proposed methods we were able to obtain promising results in improving the effectiveness of conventional CNNs for event-based computation obtaining an architecture capable to deal with real-world applications where it is likely to find objects in different positions and scales.

As a first improvement of the proposed models, we plan to extend the original leaky frame integration procedure with an adaptive procedure able to dynamically vary the leak parameter and adapt the trained model to the speed of observed objects. Moreover, as we aim to design a fully event-based network which does not rely on reconstructed frames to recognize objects, we are also considering to extend the event-based N-DRAW model by directly processing the filtered coordinates with an additional pLSTM layer, as in the original Phased LSTM recognition network [16], without making use of frames to extract features. The network could indeed still maintain good translation and scale invariance properties by exploiting the ability of its *event-based read* to filter out irrelevant events while maintaining and centering the relevant ones in the network field of view.

As a final remark let's note that the feasibility of the proposed approach is also supported by the features of advanced many-core architectures, e.g., high-performance and low-latency communication subsystems [32] and caches able to dynamically adapt their memory capacity [31], that allow to efficiently process the asynchronous and bursty sequence of events imposed by the event-based cameras.

References

[1] M. Cannici, M. Ciccone, A. Romanoni, and M. Matteucci. Asynchronous Convolutional Networks for Object Detection in Neuromorphic Cameras. *arXiv*, Nov 2018. 1, 2, 6

[2] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, April 2018. 1

[3] L. Chen, H. Zhang, J. Xiao, L. Nie, J. Shao, W. Liu, and T. Chua. Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6298–6306, July 2017. 1

[4] L. Gao, Z. Guo, H. Zhang, X. Xu, and H. T. Shen. Video captioning with attention-based lstm and semantic consistency. *IEEE Transactions on Multimedia*, 19(9):2045–2055, 2017. 1

[5] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *PMLR*, pages 249–256, Mar 2010. 6

[6] K. Gregor, I. Danihelka, A. Graves, D. Rezende, and D. Wierstra. Draw: A recurrent neural network for image generation. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1462–1471, Lille, France, 07–09 Jul 2015. PMLR. 1, 2, 4, 7

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 1

[8] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, and W. Ouyang. T-cnn: Tubelets with convolutional neural networks for object detection from videos. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1, 2018. 1

[9] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv*, Dec 2014. 6

[10] H. Li, H. Liu, X. Ji, G. Li, and L. Shi. CIFAR10-DVS: An Event-Stream Dataset for Object Classification. *Front. Neurosci.*, 11:309, May 2017. 6

[11] D. Lin, Y. Ji, D. Lischinski, D. Cohen-Or, and H. Huang. Multi-scale context intertwining for semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 603–619, 2018. 1

[12] G. Lin, A. Milan, C. Shen, and I. D. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Cvpr*, volume 1, page 5, 2017. 1

[13] T. Lin, P. Dollr, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, July 2017. 1

[14] W. Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9):1659–1671, Dec 1997. 1

[15] V. Mnih, N. Heess, A. Graves, and k. kavukcuoglu. Recurrent models of visual attention. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2204–2212. Curran Associates, Inc., 2014. 1, 7

[16] D. Neil, M. Pfeiffer, and S.-C. Liu. Phased lstm: Accelerating recurrent network training for long or event-based sequences. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3882–3890. Curran Associates, Inc., 2016. 1, 2, 4, 7, 8

[17] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor. Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades. *Front. Neurosci.*, 9, Nov 2015. 6

[18] J. A. Pérez-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.*, 35(11):2706–2719, Nov 2013. 1

[19] B. Ramesh, H. Yang, G. Orchard, N. A. L. Thi, and C. Xiang. DART: Distribution Aware Retinal Transform for Event-based Cameras. *arXiv*, Oct 2017. 8

[20] F. Rea, G. Metta, and C. Bartolozzi. Event-driven visual attention for the humanoid robot iCub. *Front. Neurosci.*, 7:234, Dec 2013. 1

[21] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 00, pages 6517–6525, July 2017. 1

[22] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 39(6):1137–1149, June 2017. 1

[23] T. Serrano-Gotarredona and B. Linares-Barranco. Poker-DVS and MNIST-DVS. Their History, How They Were Made, and Other Details. *Front. Neurosci.*, 9, Dec 2015. 6

[24] C. Song, Y. Huang, W. Ouyang, and L. Wang. Mask-guided contrastive attention model for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1179–1188, 2018. 1

[25] D. Sonnleithner and G. Indiveri. A neuromorphic saliency-map based active vision system. *2011 45th Annual Conference on Information Sciences and Systems*, pages 1–6, Mar 2011. 1

[26] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017. 1

[27] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang. Residual attention network for image classification. *arXiv preprint arXiv:1704.06904*, 2017. 1

[28] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5987–5995. IEEE, 2017. 1

[29] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudi-nov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2048–2057, Lille, France, 07–09 Jul 2015. PMLR. 1

[30] 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 Trans. Neural Networks Learn. Syst.*, 26(9):1963–1978, Sep 2015. 2

[31] D. Zoni, L. Colombo, and W. Fornaciari. Darkcache: Energy-performance optimization of tiled multi-cores by adaptively power-gating llc banks. *ACM Transactions on Architecture and Code Optimization (TACO)*, 15(2):21, 2018. 8

[32] D. Zoni, J. Flich, and W. Fornaciari. Cutbuf: Buffer management and router design for traffic mixing in vnet-based nocs. *IEEE Transactions on Parallel and Distributed Systems*, 27(6):1603–1616, June 2016. 8