Event-LSTM: An Unsupervised and Asynchronous Learning-based Representation for Event-based Data

Lakshmi Annamalai[1], Vignesh Ramanathan[2], and Chetan Singh Thakur[3]

***Abstract*—Event cameras are activity-driven bio-inspired vision sensors that respond asynchronously to intensity changes resulting in sparse data known as events. It has potential advantages over conventional cameras, such as high temporal resolution, low latency, and low power consumption. Given the sparse and asynchronous spatio-temporal nature of the data, event processing is predominantly solved by transforming events into a $2D$ spatial grid representation and applying standard vision pipelines. In this work, we propose an auto-encoder architecture named as *Event-LSTM* to generate $2D$ spatial grid representation. Ours has the following main advantages 1) Unsupervised, task-agnostic learning of $2D$ spatial grid. Ours is ideally suited for the event domain, where task-specific labeled data is scarce, 2) Asynchronous sampling of event $2D$ spatial grid. This leads to speed invariant and energy-efficient representation. Evaluations on appearance-based and motion-based tasks demonstrate that our approach yields improvement over state-of-the-art techniques while providing the flexibility to learn spatial grid representation from unlabelled data.**

***Index Terms*—Deep Learning Methods, Deep Learning for Visual Perception, Representation Learning, Event Camera, LSTM**

I. Introduction

Event-based cameras, also known as silicon retinas, are a novel type of biologically inspired sensors that encode per-pixel scene dynamics asynchronously with microsecond resolution in the form of a stream of events. Key advantages of an event camera are: high temporal resolution, sparse data, high dynamic range, and low power requirements [1], which makes it a suitable choice for resource-constrained environments. However, one of the most challenging aspects of working with event cameras is the continuous and asynchronous nature of the data. This has prompted a paradigm shift that allows efficient extraction of meaningful information from the space-time event data without sacrificing the sparsity and temporal resolution.

Inspired by the benchmark set by the traditional vision and deep learning approaches, one of the predominant areas of research in event data focuses on aggregating the information conveyed by individual events onto a spatial grid representation. This ensures its compatibility with the tools available from the conventional vision domain. While interest in converting events into spatial representation by hand-crafted data transformations is growing, only very few approaches have looked into the more complex solutions that data-driven deep learning methods can provide. The recent supervised deep-learning works were proposed by the authors of [2], [33] and [3]. However, not every application has enough volume of labeled data to quench the data-hunger thirst of supervised deep learning algorithms, limiting the design of deep supervised networks to approximate complex functions. Though supervised extraction of features is robust, the key challenge lies in extracting domain-specific features for each task. Convergence is guaranteed only if sufficiently labeled data is available; otherwise, the networks are more prone to get stuck in local minima, particularly for event data that is highly non-linear. Hence, the palpable advantage could be attained if the problem could be formulated to avoid over-fitting the training data caused by the dearth of learning parameters.

The main contribution of this paper is a generic, deep learning-based task-independent architecture (*Event-LSTM*) for transforming raw events into spatial grid representation. We achieve task independence by operating the popular architecture, LSTM, in an unsupervised setting to learn a mapping from raw events into a *task-unaware* spatial representation, which we call LSTM Time Surface (*LSTM-TS*). The proposed *Event-LSTM* puts forth unsupervised event data representation generation as an alternative to data-hungry supervised learning approaches. It can also pave the way to enable the modeling of complex structures by stacking multiple independently trainable feature extraction layers and eliminating the need for large quantities of labeled data for each task at hand. The learned model could also be utilized to initialize the supervised architectures.

Driven by the need for efficient speed invariant and energy-efficient feature extraction and to take advantage of the asynchronous sensing principle of event cameras, we propose an asynchronous method of sampling $2D$ spatial grid. Despite the fact that asynchronous mode has been explored in the literature previously [15] [16] [17], deep learning-based asynchronous spatial grid generation remains unexplored.

II. Related Work

This section involves a brief review of the event-based spatial grid representations of event data. A detailed review is provided at [4] [22]. The most commonly used framework for an asynchronous event by event processing is a

Manuscript received: Oct, 11, 2021; Revised Jan, 5, 2021; Accepted Jan, 30, 2022.

This paper was recommended for publication by Editor Jens Kober upon evaluation of the Associate Editor and Reviewers' comments.

[2]Vignesh Ramanathan and [3]Chetan Singh Thakur are with Indian Institute of Science, Banaglore, India. vigneshr@iisc.ac.in, csthakur@iisc.ac.in

[1]Lakshmi Annamalai is with Defence Research and Development Oraganization, India and Indian Institute of Science, India lakshmia@iisc.ac.in

Digital Object Identifier (DOI): see top of this page.

promising research field known as Spiking Neural Networks (SNN) [5] [23]. However, there is a lack of standard training procedures due to their non-differentiability. Another line of recent research is to leverage graph-based learning [25] [26]. This involves representing events as graphs followed by graph convolution learning networks. However, it is challenging to solve graph-based learning problems as event camera data is not native to the graph structure. Hence graph representation learning is in itself a challenging task. Moreover, unlike applications such as social detection analysis, event camera graph representation learning is a dynamic problem, making it much more complex. The alternative line of research is to perform pre-processing on events to convert them into formats compatible with conventional image-based vision architectures [37] [38] [39]. Combined with a high signal-to-noise ratio compared to raw events, the traditional, proven vision solutions have resulted in high accuracy. However, this involves coming up with a better and optimal choice of converting raw asynchronous event stream into conventional vision algorithm compatible structures. This effort has led to different initial representation stages such as event image [6] [19] [20] [8], Time Surface (TS), voxel grid [7] and motion compensated event image [21].

Researchers came up with a popular spatial representation called Time Surface (*TS*), which stores a time value at each pixel. This results in encoding motion history at each pixel, thus making it sensitive to the motion of the edges. This gives *TS* an edge over other spatial representations, especially for the vision tasks that involve motion analysis. Several handcrafted $2d$ grid representations [9] [10] have been proposed over years. However, coupling deep learning methods with event data will allow us to take advantage of event data and the learning algorithms. In [2], authors proposed a Multi-Layer Perceptron (MLP) architecture towards converting event data to spatial grid representation. As the architecture is defined by differentiable operators, it makes the process completely learnable. The information at each pixel is formed by accumulating the MLP feature generated from the events at the particular spatial location. The major disadvantage of this method is that the accumulation of events does not depend on the sequence of events, thus inhibiting the network from learning the memory embedded in the events. This has been overcome by [12] and [3] by leveraging the memory property of LSTM.

[12] proposed Phased LSTM, which does not result in the explicit formation of spatial grid representation. This prevents us from leveraging the spatial structure of the data, which could be learned using CNN architectures, thus restricting it to simple applications [11]. Moreover, it also introduced huge latency as the events have to pass through the network sequentially. Hence, [3] explored the utility of LSTM as $M \times N$ (pixel grid size) matrix of cells to learn the mapping from raw event sequence to a dense spatial grid. LSTM cells process the sequence of events at each pixel location, condensing it into a single output vector that populates the spatial grid. They have demonstrated good accuracy in object recognition and optical flow estimation by replacing the corresponding architecture's input representation with the MatrixLSTM layer. To make the feature learning process translation invariant, they experimented with parameter sharing across LSTM cells.

The work proposed in this paper is related to [3]. The significant advantages of the proposed method as compared to that of [3] are i) An unsupervised deep learning solution to learn the best mapping from events to $2D$ spatial grid, which is task-independent suited for situations where the process of collecting labeled data is very costly, ii) Having multiple LSTM layers at encoder allows us to model the dynamicity present in event sequences, iii) Asynchronous sampling of $2D$ spatial grid has allowed us to take advantage of the asynchronous nature of event camera.

III. Proposed Solution

This section formalizes the methodology to convert raw events into spatial representation (Fig. 1). An event camera with pixel grid size $M \times N$ results in an asynchronous stream of events only on those locations where the change in logarithmic brightness change reaches a threshold. The events are tuples with the format $e_i = (x_i, y_i, t_i, p_i)$, where (x_i, y_i) represents the pixel location of the i^{th} event, $p_i \in \{+1, -1\}$ known as ON and OFF events represents the sign of brightness change and $t_i \geq 0$ indicates the timestamp of the event with microseconds resolution. The time of occurrence of an event is a continuous parameter, and hence the number of events with the same timestamp will be extremely limited to get processed. This mandates the need to accumulate the events over a period δT to process effectively. This representation of events will more robustly describe the dynamics of the scene.

A. Event-LSTM

The goal here is to learn a mapping ϕ from a set of raw events sequence represented as $E = \{e_i\}$ to a $2D$ grid representation *LSTM-TS* of size $M \times N$. The mapping should be learned in an unsupervised manner, based solely on the data and independent of the task at hand to account for the situations where the amount of labeled data available for each task is very limited in relation to the number of parameters to be learned. Hence, in this work, we propose an LSTM-based autoencoder structure known as *Event-LSTM*, which learns an encoder and decoder function ϕ and ψ respectively to predict feature and reconstruct the input sequence, respectively.

B. Network Architecture

To ensure translation invariance, our *Event-LSTM* is comprised of a single autoencoder LSTM across all pixels. This reduces the number of parameters to be learned. The event sequence at each pixel location is processed sequentially during the training phase. However, processing can be applied in parallel across all pixels during the feature extraction phase. Encoder is comprised of three layers, with N_{xy}, 16 and 1 output units, where N_{xy} is the number of events at pixel (x, y). The output of the preceding layer becomes input features for the subsequent layers. The decoder is designed to unfold the encoding. Hence decoder is composed of three layers stacked in the reverse order. The latent space feature dimension has been restrained to one to force the architecture to learn salient

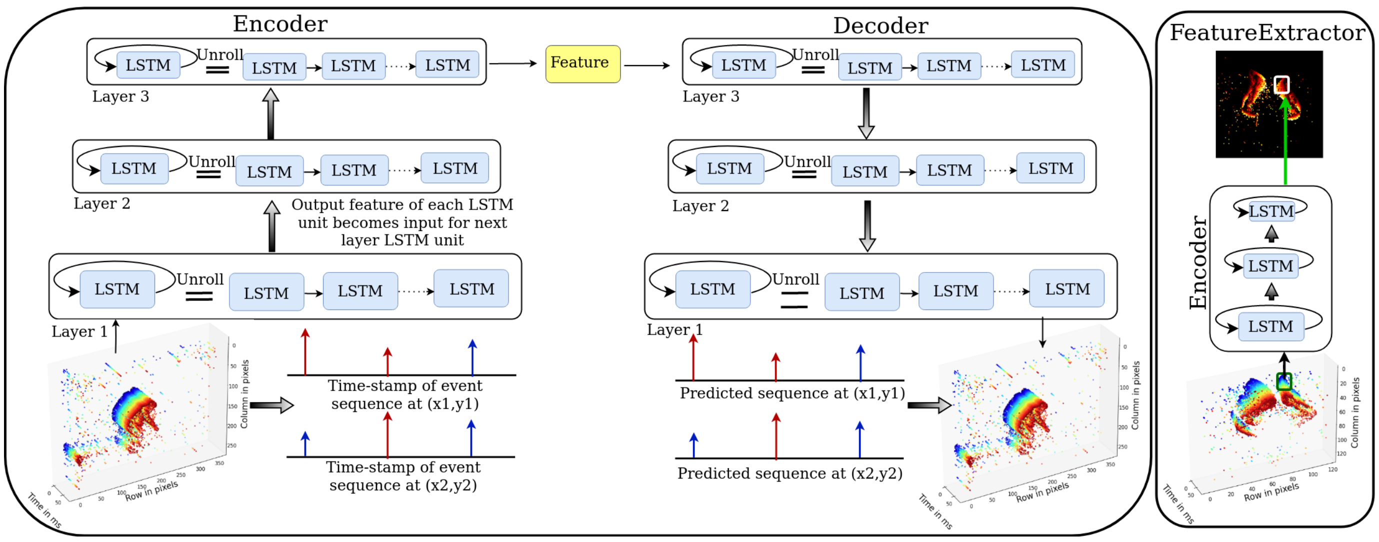
Encoder
Decoder
FeatureExtractor
LSTM
Unroll
Layer 3
Layer 2
Layer 1
Feature
Output feature of each LSTM unit becomes input for next layer LSTM unit
Time-stamp of event sequence at (x1,y1)
Time-stamp of event sequence at (x2,y2)
Predicted sequence at (x1,y1)
Predicted sequence at (x2,y2)
Time in ms
Row in pixels
Column in pixels
Encoder

Fig. 1. The architecture of the proposed *Event-LSTM* during the training (Left figure) and feature extraction (Right figure) phase. Training Phase: The event sequence at each pixel is grouped in synchronous/asynchronous mode and fed to the encoder of LSTM one by one. Encoder and decoder are composed of three layers of LSTM units. The encoder and decoder are trained to learn a good feature at each pixel and reconstruct the corresponding event sequence. Feature Extraction Phase: The event sequence at each pixel is grouped in synchronous/asynchronous mode and fed to the encoder of LSTM. Only the encoder is used during the feature extraction phase. The features extracted at each pixel is populated into *LSTM-TS*

features. The major difference between Matrix-LSTM and the proposed solution lies in extending the former to multi-layer encoder-decoder architecture, which enables *Event-LSTM* to learn temporal features in an unsupervised setting irrespective of the dynamic nature of event sequence across tasks.

C. *LSTM-TS: Learning and Extraction*

Let $E_{xy} = (x_n, y_n, t_n, p_n)$ denote the set of events generated at $(x_n = x, y_n = y)$ with varying lengths $|E_{xy}|$ depending on the dynamic activity level at that specific pixel. Each event sequence E_{xy} is represented by a sequence of time stamps t_n occurring at that specific location $(x_n = x, y_n = y)$ $(T_{xy} = \{t_n | x_n = x, y_n = y\})$. LSTM auto-encoder is fed with these time stamp sequences T_{xy}. *Event-LSTM* with its internal memory learns an encoder $(\phi(T_{xy}))$ transformation that estimates a meaningful feature *LSTM-TS*$_{xy}$ from the input T_{xy} sequence and a decoder transformation $(\psi(\phi(T_{xy})))$ that tries to recreate the input sequence by minimising the error $|T_{xy} - \widehat{T}_{xy}|$.

The $M \times N$ *LSTM-TS* is realized by populating a spatial grid with the features learned by encoder at each (x, y) location (right side of Fig. 1). This preserves the sparsity of the event data as computation of features occurs only at locations (x, y) where the sequence T_{xy} is non-empty.

D. *Windowing*

The proposed *Event-LSTM* operates on temporal windows. The input event sequence E_{xy} is split into smaller sequences E^l_{xy}, resulting in independent time stamp sequence T^l_{xy} for each window δ. This makes sure that *Event-LSTM* does not have to deal with very long sequences. Although LSTM is better at handling vanishing gradient problems effectively better than RNN, splitting long sequences into smaller sequences ensures that *Event-LSTM* can effectively retain local time information. We have proposed a novel asynchronous (async) $2D$ spatial grid sampling technique as described in the sections below.

In the asynchronous $2D$ spatial grid sampling technique, the smaller sequences are formed such that the number of total events $|E^l|$ remains constant across L sequences as against synchronous (sync) $2D$ spatial grid sampling where the time remains constant across L sequences. This will result in $L = \frac{|E|}{|E^l|}$ sequences, with $|E|$ representing the total number of events in the parent sequence and $|E^l|$ representing the number of events in the individual sequence. This form of windowing's significant benefits has been substantiated with simulation and real data experiments in supplementary and Section IV respectively. The advantages of the proposed windowing are discussed below.

1) Speed Invariant Classification: Note that the asynchronous $2D$ spatial grid sampling approach enables speed invariant feature extraction necessary to cope with intraclass variations. Asynchronous feature extraction captures the pattern of motion, irrespective of its speed.

2) Energy Saving Feature Extraction: Asynchronous $2D$ spatial grid sampling initiates processing only when a specified number of events is accumulated, resulting in non-redundant *LSTM-TS* while still extracting richer information from the raw event sequence.

IV. EVALUATION AND RESULTS

The experimental section is devoted to demonstrating the potential of *Event-LSTM* as spatial representation learning for appearance-based and motion-based vision tasks such as object recognition, large-scale action recognition, and gesture recognition. In section IV-A, we focus on the experiments that demonstrate the efficacy of the proposed *Event-LSTM* to deal with the lack of labels when compared to conventional handcrafted unsupervised spatial representation learning methods.

In section IV-B, we present results to display the ability of *Event-LSTM* to convert raw events into a spatial representation which supervised deep learning architectures could utilize for higher-level vision tasks.

We have generated $2D$ spatial grids by bifurcating the parent sample into smaller windows. This will make individual grids less informative, leading to increased noise in decision-making. To mitigate this, a decision rolling buffer mechanism has been adapted in [36], where the results of past and current classification are combined effectively.

However, we have not adopted the decision rolling buffer mechanism for the following two reasons, i) To make the proposed $2d$ spatial representation generic enough for other vision applications as well such as visual odometry, optical flow estimation, etc., which mandates decision-based on current observation alone and ii) To validate the utility of the proposed solution in scenarios which have high dynamicity in terms of the classes.

A. LSTM-TS in End-to-End Unsupervised Setting

In this section, we evaluated the *LSTM-TS* features in an end-to-end unsupervised setting with minimal supervision. This validates the performance of the proposed spatial representation under the scenarios where the number of labeled data is minimal. Towards this, we have conducted experiments for prevalent vision tasks known as action recognition and gesture recognition on datasets [13] [18], which requires temporal information extraction. Beyond providing results to validate the asynchronous mode of $2D$ spatial grid sampling over the synchronous mode, we benchmarked our results against state-of-the-art non-deep learning unsupervised spatial representation learning methods.

1) Implementation: To evaluate *Event-LSTM* in minimal supervision scenario, we have created a pipeline with *MobileNet* (pre-trained on ImageNet) as the primary spatial feature extractor of *LSTM-TS*, followed by the popular classifier referred to as Support Vector Machine (SVM). We remark that the only network which requires labeled data is SVM, unlike end-to-end supervised approaches. Since SVM is a shallow classifier, the amount of labeled data needed is substantially lesser than that demanded by deep networks.

To prove the task-independent nature of *Event-LSTM*, we trained it on action recognition data alone and evaluated it on both applications. ADAM has been used for optimization with a learning rate of 0.001. A batch size of 64 has been used. Hyperparameter ($|E^l|$ and δt) search has been carried out empirically. We then select the integration interval as $|E^l| = 10k$ events (asynchronous mode) and $\delta t = 100ms$ (synchronous mode). Input time features are normalized between 0 and 1. The *LSTM-TS* representations that result from single-parent action/gesture clippings have been considered as a set of input samples for higher-level vision tasks. The difference in the window length of synchronous and asynchronous sampling and variation in the duration of parent clipping resulted in the imbalanced distribution of the $2D$ spatial grid. To counter this effect, we have adapted resampling to change data composition to an average of 196 and 1877 spatial grids in each class of action and gesture recognition datasets, respectively.

Analysis of datasets [13] and [18] revealed that the event camera's sensitivity to temporal noise and junction leakage currents resulted in events generated under constant illumination with no activity in the scene, referred to as BA (Background Activity) noise. To mitigate its effect, we have implemented a Spatio-temporal filter that considers the evidence provided from all the past events within the given spatiotemporal window. Details of this Spatio-temporal filter are provided in supplementary.

2) Action Recognition: Action recognition is a well-known vision task with its application in various scenarios such as surveillance. Action recognition is a motion analytics task whose accuracy depends on how well a feature extractor can capture the motion information embedded in the event data.

a) Dataset: For this evaluation, we have used publicly available datasets provided at [13], which is a collection of recordings from an empty office captured with DAVIS346. The dataset comes with 15 subjects acting 12 different actions, with each action lasting $5s$.

b) Synchronous vs. Asynchronous: We start by comparing the performance of *Event-LSTM* in synchronous and asynchronous modes. Results of the two variants are reported in Table I for individual activities in terms of multi-class metrics. It could be seen that asynchronous setting performs consistently better on a wide range of activities. This substantiates the goal of our asynchronous $2D$ spatial grid sampling to capture motion information from events irrespective of the speed of the motion. In addition to accuracy metrics, we have also given the statistics of the average number of *LSTM-TS* generated by the two modes and their corresponding processing time. The average was estimated over 3 parent clips. Lesser number of *LSTM-TS* generated in asynchronous mode across a wide number of classes proves its energy-saving capability.

c) Perfomance w.r.t State-of-the-art: For further validation of the proposed solution, we demonstrate in Table II that *Event-LSTM* is a better-suited solution than state-of-the-art methods in the context of action recognition. *Event-LSTM* is an unsupervised feature extraction process of estimating a mapping to convert raw event sequence into a spatial representation. To keep the comparison fair, we have considered only unsupervised methods of generating a spatial representation. The various state-of-the-art methods used for comparison are Surface of Active Events (SAE: the timestamp of the recent event), SNN (Count of number of firing spikes), EvOn (Number of on events), EvOff (Number of off events) [8], EvCount (number of events at each pixel in the given count of total events) [15] and Exp (time stamp weighted with exponential function) [10]. Through an extensive evaluation, we show that using *LSTM-TS* in async. mode improves the performance of multi-class action recognition over hand-crafted features by a good margin.

3) Gesture Recognition: Gesture recognition is a popular field of research with its application in various areas such as robotics, visual cognition, human-computer interaction, etc. Real-time gesture recognition is well suited to event cameras. In this section, we study the effect of the proposed spatial representation method on gesture recognition application.

Actions	Asynchronous			Synchronous			Gestures	Asynchronous			Syncronous		
	F1 Score	N	T	F1 Score	N	T		F1 Score	N	T	F1 Score	N	T
ArmCross	0.80	71	335.12	0.62	145	684.4	**LH_C**	0.89	44	207.68	0.85	48	226.56
Picking	0.88	241	1137.52	0.86	242	1142.24	**FA_RB**	0.95	35	165.2	0.89	46	217.12
Falling	1.00	282	1331.04	0.95	277	1307.44	**RH_W**	0.94	26	122.72	0.92	42	198.24
Waving	1.00	112	528.64	0.67	103	486.16	**LH_CC**	0.18	41	193.52	0.50	44	207.68
GetUp	0.88	80	377.6	0.86	143	674.96	**RH_C**	0.84	50	236	0.77	44	207.68
Walking	0.92	141	665.52	0.95	163	769.36	**LH_W**	0.96	25	118	0.93	44	207.68
Turning	1.00	83	391.76	0.80	153	722.16	**Clap**	0.85	15	70.8	0.91	37	174.64
Sit	1.00	289	1364.08	0.93	300	1416	**Drums**	0.91	31	146.32	0.84	43	202.96
Kicking	0.67	93	438.96	0.67	149	703.28	**RH_CC**	0.84	50	236	0.79	44	207.68
Jumping	0.89	103	486.16	0.80	96	453.12	**Guitar**	0.90	22	103.84	0.67	43	202.96
Tying	0.88	181	854.32	0.67	244	1151.68							
Throwing	1.00	76	358.72	0.80	151	712.72							

TABLE I

ANALYSIS OF *LSTM-TS* IN SYNC VS. ASYNC ON ACTION AND GESTURE RECOGNITION IN TERMS OF F1-SCORE. N AND T: AVERAGE NUMBER OF FRAMES GENERATED (FROM THREE AND ONE PARENT CLIPS OF ACTIVITY AND GESTURE RECOGNITION, RESPECTIVELY)) AND TIME (IN SECS) REQUIRED FOR PROCESSING THE SAME. RH-RIGHT HAND, FA-FORE ARM, LH-LEFT HAND, C- CLOCKWISE, CC-COUNTERCLOCKWISE, W-WAVE.

Method	Accuracy		Method	Accuracy	
	AR	GR		AR	GR
Async	0.93	0.91	**Sync**	0.85	0.82
EvOn	0.62	0.71	**EvOff**	0.51	0.78
Voxel	0.74	0.71	**SAE**	0.77	0.73
Exp	0.81	0.79	**EvCount**	0.78	0.79
SNN	0.85	0.79			

TABLE II

COMPARISON OF *LSTM-TS* ON ACTION RECOGNITION (AR) AND GESTURE RECOGNITION (GR) IN TERMS OF ACCURACY. A PRE-TRAINED MOBILENET ARCHITECTURE HAS BEEN USED AS A FEATURE EXTRACTOR. CLASSIFICATION IS DONE WITH SVM. GOOD ACCURACY SHOWS THE PROPOSED ARCHITECTURE'S UTILITY WHEN THERE IS SCARCITY IN LABELED DATA.

a) Dataset: For gesture recognition evaluation, we have used the publicly available dataset provided at [18]. The gesture recognition dataset [18] comprises 11 hand and arm gestures comprising of 1595 instances. It has been collected from 29 subjects. As the proposed method primarily targets the scenario where the labeled data is minimal, we have used only 319 instances.

b) Synchronous vs. Asynchronous: Table I furnishes the comparison of *Event-LSTM* in synchronous and asynchronous settings in terms of classification metrics over individual gestures. As expected, the windowing technique followed impacts the performance of the recognition task. It could be seen that asynchronous configuration displays improved performance on most gestures. This highlights the capability of asynchronous $2D$ spatial grid sampling to model a wide variety of gestures irrespective of their speed of motion. The average number of *LSTM-TS* generated and their corresponding processing time was also furnished to substantiate the energy-saving efficacy of the asynchronous mode.

c) Perfomance w.r.t State-of-the-art: We compare the proposed method with state-of-the-art unsupervised representations on gesture recognition in Table II. The details of various methods considered for comparison are provided in the section. IV-A2.

B. LSTM-TS with Supervised Deep Learning Architectures

Despite the state-of-the-art performance of convolutional neural network (CNN) architectures, they cannot be readily used on event data due to their inherent difference in the data structure. The main objective of this paper is to address this gap. Hence, in this section, we present an in-depth analysis of the proposed task-agnostic and asynchronous *LSTM-TS* in resolving the issue of leveraging the popular supervised deep learning architectures for event data. Evaluation of the proposed method on complex object recognition datasets, N-Cars and N-Caltech101, and a prominent and 101 class count action recognition dataset, UCF101-DVS, is established.

1) Action Recognition on UCF101-DVS: The datasets [14] and [18] used for analysis were collected in less complex scenarios. They are also modest in the number of classes. Hence, we have also evaluated the proposed *Event-LSTM* on a complex dataset known as UCF101-DVS, which was captured by playing back the APS UCF-101 dataset. It consists of $13,320$ videos of 101 different human actions.

a) Implementation: We have followed a similar *LSTM-TS* extraction protocol as mentioned in section IV-A2. However, no data pre-processing has been done as the data analysis revealed the former's nonnecessity. The *Event-LSTM* network has been trained with a standard ADAM optimizer with a momentum of 0.9. The learning rate has been set to 0.0001. *LSTM-TS* are constructed in asynchronous mode with $|E^l| = 10k$ events. The hyperparameter $|E^l|$ was fixed such that our representation covers optimal temporal extent, which was decided based on the performance obtained in action classification. Spatial features are learned on *LSTM-TS* using deep learning architecture with MobileNet as base network (weights pre-trained on ImageNet) padded with a classification layer of 101 nodes, whose weights are randomly initialized. The MobileNet has been trained using an ADAM optimizer with a learning rate of 0.01. Train and test split considered are 67% and 33% respectively.

b) Results: We present our results on UCF101-DVS in Table III, which provides a comparison of the proposed method with graph convolutional network [26] and $3D$ models proposed for conventional camera UCF-101 dataset, which has been cited in [26]. The justification for the poor performance of APS domain models on the UCF101-DVS dataset has been provided in [26]. Action recognition is a motion analytics task that requires good Spatio-temporal features. In [26], it was not feasible to create a graph for a longer duration. Hence, to capture motion dynamics, spatial features are learned for smaller

Method	Accuracy	Method	Accuracy
C3D [27]	0.472	**ResNet-50** [31]	0.602
ResNet-34 [28]	0.579	**I3D** [32]	0.635
P3D-63 [29]	0.534	**RG-CNN+Incep. 3D** [26]	0.678
R2+1D-36 [30]	0.628	**Proposed**	**0.776**

TABLE III
COMPARISON OF ACTION RECOGNITION ON UCF101-DVS DATASET. *Event-LSTM* IS TRAINED IN ASYNC. MODE ON THE UCF101-DVS DATASET. MOBILENET IS FINE-TUNED WITH THE EXTRACTED SPATIAL FEATURE REPRESENTATIONS. THE METRICS OF OTHER METHODS ARE AS CITED IN [26]. OUR METHOD GIVES A VERY GOOD ACCURACY OF 77.6% FOR A COMPLEX DATASET SUCH AS UCF101-DVS.

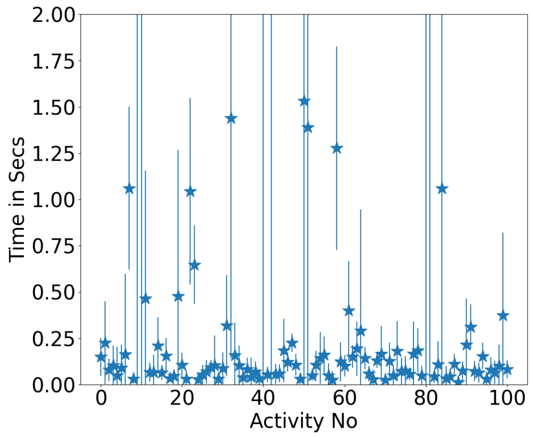

Fig. 2. The average time elapsed during $10k$ event generation vs. activities of UCF101-DVS. It also gives the intra-class variance of the time elapsed. High variance indicates the speed invariant information extraction property of asynchronous mode.

durations and stacked over temporal dimensions. These $3D$ structures were later fed to $3D$ models to learn Spatio-temporal features for classification.

It can be seen from Table III that *Event-LSTM* was able to achieve good recall and F1-Score of 0.776 for a complex dataset such as UCF101-DVS. The proposed asynchronous mode of sampling was able to capture motion dynamics seamlessly independent of the speed of the motion. To validate the same, an analysis of the time elapsed (along with its intraclass variance) in generating $10k$ events has been provided in Fig. 2 (Y axes value above $2s$ are clipped for the purpose of visualization). It is evident that most of the activities have large intra-class variance, thus indicating a high dynamical range of the speed at which they were performed and speed-invariant feature extraction capability of asynchronous mode. This enabled us to feed our *LSTM-TS* to lesser parameterized $2D$ supervised deep learning architecture, MobileNet, to learn action classification.

Fig. 3 gives the statistics of the number of *Event-LSTM* generated in synchronous mode (100 ms per grid) for an event count of $10k$ (Y axes value above 20 are clipped for the purpose of visualization). On most of the activities, the number of *Event-LSTM* generated in synchronous mode is greater than that of asynchronous mode, thus emphasizing the energy-saving advantage of the latter.

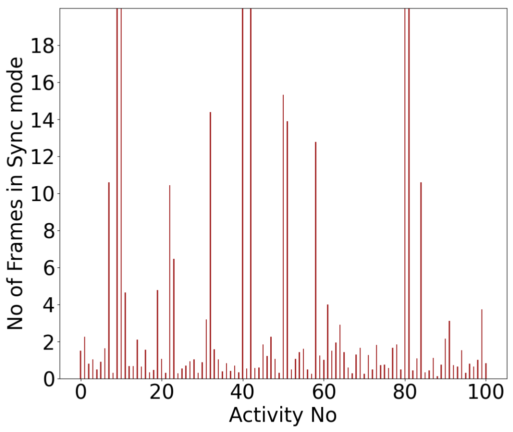

Fig. 3. Number of *LSTM-TS* generated in synchronous mode ($100ms$ per grid) for an event count of $10k$ vs activities of UCF101-DVS. For most of the activities, the number of $2D$ spatial grids generated in synchronous mode is larger than that of asynchronous mode. This proves the energy saving capability of asychronous mode of *LSTM-TS* sampling.

2) *Object Recognition:*

Object recognition is a vital vision application that finds utility in various fields such as surveillance, robotics, etc. The comprehensive experiments conducted are targeted towards demonstrating the task-agnostic nature of *Event-LSTM* generation. Finally, we also highlight where we stand in terms of end-to-end supervised learning architectures.

a) N-Cars: We evaluated the efficacy of *LSTM-TS* on event-based object recognition task with publicly available dataset N-Cars. N-Cars is a public dataset recorded with an ATIS sensor with binary categories, cars, and urban backgrounds. It consists of a total of $24,029$ samples each of length $100ms$.

Implementation To emphasize the task-agnostic nature of our *Event-LSTM*, we have conducted experiments mentioned above with two different *Event-LSTM* models, model *mUCF* and model *mN-Cars*. *mUCF* is the model obtained by training *Event-LSTM* on the UCF101-DVS dataset and *mN-Cars* is the UCF101-DVS *Event-LSTM* model fine-tuned with 70% of N-Cars dataset. The network was optimized with an ADAM optimizer with a learning rate of 0.01 and a batch size of 64. The creation of *LSTM-TS* involved synchronous and asynchronous $2D$ spatial grid sampling. For synchronous mode, we have considered time interval as $\delta t = 100ms$. We have fixed the number of events as $|E^l| = 3000$ for asynchronous mode. These hyperparameters were fixed by empirical analysis. Accuracy decreases with decrease in δt and $|E^l|$ due to lack of motion. However, an increase in these hyperparameters resulted in a lesser number of *LSTM-TS*, depriving the deep learning architectures of getting enough data for learning.

Ablation Study For the object recognition task, we investigated *Event-LSTM* models *mUCF* and model *mN-Cars* in synchronous and asynchronous $2D$ spatial grid sampling using various deep learning architectures. Specifically, we have used three different networks, ResNet18, ResNet34, and MobileNet, under two different configurations, i) base network followed

Network	*mUCF*		*mN-Cars*	
	DenCls	*Cls*	*DenCls*	*Cls*
ResNet-18	0.935/0.940	0.929/0.943	0.939/0.945	0.935/0.939
ResNet-34	0.948/0.940	0.935/0.944	0.949/0.953	0.938/0.959
MobileNet	0.944/0.954	0.948/0.944	0.958/0.951	0.942/0.945

TABLE IV
ANALYSIS OF *LSTM-TS* IN SYNC. VS ASYNC. ON N-CARS DATASET. THE BASE NETWORKS WERE TESTED IN TWO DIFFERENT CONFIGURATIONS KNOWN AS *DenCls* (BASE NETWORK AUGMENTED WITH TWO DENSE LAYER AND A CLASSIFICATION LAYER), AND *Cls* (BASE NETWORK WITH ITS CLASSIFICATION LAYER REPLACED). MODEL *mUCF* AND *mN-Cars* REFERS TO THE MODELS TRAINED ON UCF101-DVS AND FINE-TUNED ON N-CARS, RESPECTIVELY. ALMOST EQUAL PERFORMANCE OF MODEL *mUCF* AND *mN-Cars* EMPHASIZES THE TASK-UNAWARE BENEFIT OF *Event-LSTM*

Method	Accuracy	Method	Accuracy
H-First [24]	0.054	**HATS**	0.642
HOTS [9]	0.210	**Gabor-SNN** [10]	0.196
HATS+ResNet34	0.691	**Voxel+ResNet34**	0.754
Inception-V4	0.578	**ResNet-50**	0.637
RG-CNN [26]	0.657	**EST-1** [2]	0.837
EST-2 [3]	0.81	**M-LSTM-1** [3]	0.843 ± 0.005
M-LSTM-2 [3]	0.812 ± 0.0013	**Proposed**	0.802

TABLE VI
COMPARISON OF *LSTM-TS* IN ASYNC. MODE WITH STATE-OF-THE-ART METHODS ON THE N-CALTECH101 DATASET. THE METRICS OF OTHER METHODS ARE AS CITED IN [26] [2] [3]. ACCURACIES OF [3] AND [2] ARE THAT OF 2 CHANNEL 16 BIN (**M-LSTM-1**), 16 CHANNEL 1 BIN (**M-LSTM-2**) AND 2 CHANNEL 16 BINS (**EST-1**), 2 CHANNEL 9 BINS (**EST-2**) RESPECTIVELY.

Method	Accuracy	Method	Accuracy
HFISRT [24]	0.561	**G-CNN** [26]	0.902
HOTS [9]	0.624	**Gabor-SNN** [10]	0.789
HATS [10]	0.909	**TCI** [6]	0.917
VGG-19	0.728	**Inception-V4**	0.864
ResNet-50	0.903	**Voxel** [7]	0.847
G-CNN	0.902	**RG-CNNs** [26]	0.914
EST [2]	0.925	**MLSTM-1** [3]	0.958 ± 0.005
MLSTM-2 [3]	0.943 ± 0.004	**Proposed**	**0.959**

TABLE V
COMPARISON OF *LSTM-TS* IN ASYNC. MODE WITH STATE-OF-THE-ART METHODS ON THE N-CARS DATASET ON OBJECT RECOGNITION. [3] AND [2] ARE SUPERVISED ARCHITECTURES. WE HAVE PROVIDED THE RESULTS OF THE CONFIGURATIONS, WHICH YIELDED THE BEST RESULT AS CITED IN [26]. [2]: TWO-CHANNEL CONFIGURATION. THE ACCURACIES MLSTM-1 AND MLSTM-2 OF [3] REPORTED HERE ARE THE BEST ACCURACIES ACHIEVED WITH RESNET34-EV2VID AND RESNET34-EST.

by two dense layers and a classification layer, which we have named as *DenCls* and ii) base network followed by classification layer alone, referred as *Cls* here.

The quantitatively evaluated result on the N-Cars dataset has been provided in Table IV. It could be seen that model $mUCF$, which was not aware of the object recognition task, performs at par with that of model *mN-Cars*. UCF101-DVS was recorded with DVS, whereas N-Cars were captured with an ATIS camera. This proves the utility of *Event-LSTM* across various event camera modalities.

Comparison with State-of-the-Art We compare the accuracy obtained by our model with that of state-of-the-art hand-crafted event representations [24] [9] [10] [10] [6] [8], graph convolutional networks [26] and the recent fully supervised, end-to-end trainable architectures [3] and [2]. In addition, results of conventional deep learning architectures VGG-16, ResNet-50 and Inception-V4 provided in [26] are also given in the Table V. These frameworks were fed with a two-channel image, which records the number of positive and negative events at each pixel, respectively. Table V shows that the accuracy achieved by our method surpasses that of other methods. Though [2] is a fully supervised method, it couldn't achieve the accuracy of the proposed *Event-LSTM* as it accumulates events with no effort to learn the memory information carried by the events.

b) N-Caltech101: As N-Cars was a binary classification problem, we have further evaluated our method on a publicly available complex dataset known as N-Caltech101, which forms the most complex object recognition benchmark dataset with $8,687$ samples distributed across 101 classes.

Implementation Through extensive evaluation, we have fixed the hyperparameter $|E^l|$ as $20k$. ADAM was used as an optimizer with a learning rate of 0.0001 and early stopping based on the official split provided at [2]. We have used Resnet-18 architecture on the extracted *LSTM-TS* features for spatial feature extraction. The Resnet-18 has been pre-trained on Imagenet. The architecture was fine-tuned in a supervised fashion using an ADAM optimizer with a learning rate of 0.0001, batch size of 128, and augmentation, as proposed in citeAutoaugment. Three subsequent $20k$ *LSTM-TS* have been fed as input channels to ResNet18 to enable it to extract more meaningful features. The last layer was replaced to suit the number of classes and initialized with random weights.

Comparison with State-of-the-Art This section compares our approach with state-of-the-art methods, which also involve hand-crafted and end-to-end learned architectures. With $20k$ $|E^l|$, we achieved state-of-the-art accuracy. To keep the comparison fair, along with the best accuracy, we have cited here the accuracy of the EST variants of [3] (16 channel 1 bin) and [2] (2 channel 9 bins) with the convolution input channels matching that of the experiments conducted here. In addition to this, we have also provided the best accuracies achieved by [3] and [2] with 2 channels and 16 bins. Though our architecture is an unsupervised $2D$ spatial grid generator, it performs only slightly lesser than supervised event representation generators.

V. CONCLUSION

In this work, we presented a generic task-independent framework named as *Event-LSTM* to generate spatial representation from the sequence of raw events. By modeling the transformation with an unsupervised architecture of LSTM, we have made it a suitable solution to learn features from unlabelled data, primarily to mitigate the task-specific data-hungriness of supervised deep learning algorithms. Our framework lays out a speed invariant and energy-efficient feature extraction methodology by proposing event-dependent windowing. Furthermore, a thorough evaluation of *Event-LSTM* has been carried out on appearance-based (object recognition) and motion-based vision tasks (Action recognition and gesture recognition). Overall, our approach outperforms state-of-the-art spatial event grid representation, thus enabling us to proceed forward in *data-driven* and *task-unaware* mapping of event data to spatial representation.

References

[1] B Son and Y Suh and S Kim and H Jung and JS Kim and C Shin and K Park and K Lee and J Parkand J Woo and Y Roh and H Lee and Y Wang, *A 640x480 dynamic vision sensor with a 9 m pixel and 300Meps address-event representation*, IEEE Intl. Solid-State Circuits Conf., 2017.

[2] Daniel Gehrig and Antonio Loquercio and Konstantinos G. Derpanis and Davide Scaramuzza, *End-to-end learning of representations for asynchronous event-based data*, Intl. Conf. Comput. Vis., 2019.

[3] M. Cannici and M. Ciccone and A. Romanoni and M. Matteucci, *A Differentiable Recurrent Surface for Asynchronous Event-Based Data*, European Conference on Computer Vision, 2020.

[4] Guillermo Gallego and Tobi Delbruck and Garrick Orchard and Chiara Bartolozzi and Brian Taba and Andrea Censi and Stefan Leutenegger and Andrew J Davison and Jorg Conradt and Kostas Daniilidisand Davide Scaramuzza, *Event-based Vision: A Survey*, IEEE Trans. Pattern Anal. Mach. Intell., 2020.

[5] Dongsung Huh and Terrence J Sejnowski, *Gradient descent for spiking neural networks*, Conf. Neural Inf. Process. Syst., 2018.

[6] Ana I and Maqueda and Antonio Loquercio and Guillermo Gallego and Narciso Garcia and Davide Scaramuzza, *Event-based vision meets deep learning on steering prediction for self-driving cars*, IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2018.

[7] Alex and Liangzhe Yuan and Kenneth Chaney and Kostas Daniilidis, *Unsupervised event-based learning of Optical flow depth and egomotion*, IEEE Conf. Comput. Vis. Pattern Recog., 2019.

[8] Alex Zihao Zhu and Liangzhe Yuan and Kenneth Chaney and Kostas Daniilidis, *EV-FlowNet: Self-supervised optical flow estimation for event-based cameras*, Robotics: Science and Systems (RSS), 2018.

[9] X. Lagorce and G. Orchard and F. Galluppi and B. E. Shi and R. B. Benosman, *Hots: a hierarchy of event-based time-surfaces for pattern recognition*, IEEE Trans. Pattern Anal. Mach. Intell., 2017.

[10] Amos Sironi and Manuele Brambilla and Nicolas Bourdis and Xavier Lagorce and Ryad Benosman, *HATS: Histograms of averaged time surfaces for robust event-based object classification*, IEEE Conf. Comput. Vis. Pattern Recog., 2018.

[11] Marco Cannici and Marco Ciccone and Andrea Romanoni and Matteo Matteucci, *Attention mechanisms for object recognition with event-based cameras*, IEEE Winter Conf. on Applications of Comput. Vis., 2019.

[12] Daniel Neil and Michael Pfeiffer and Shih-Chii Liu, *Phased lstm: Accelerating recurrent network training for long or event-based sequences*, Advances in Neural Inf. Process. Syst., 2016.

[13] Hu and Yuhuang and Hongjie Liu and Michael Pfeiffer and Tobi Delbruck, *DVS benchmark datasets for object tracking, action recognition, and object recognition*, Frontiers in neuroscience, 2016.

[14] Miao Shu and Chen Guang and Ning Xiangyu and Zi Yang and Ren Kejia and Bing Zhenshan and Knoll Alois, *Neuromorphic benchmark datasets for pedestrian detection, action recognition, and fall detection*, Frontiers in neurorobotics, 2019.

[15] E. Calabrese and G. Taverni and C. Awai Easthope C et. al., *Dhp19: Dynamic vision sensor 3d human pose dataset*, IEEE Workshop on Comput. Vis. Pattern Recog., 2019.

[16] I. Alzugaray and M. Chli, *ACE: An Efficient Asynchronous Corner Tracker for Event Cameras*, Intl. Conf. on 3D Vis., 2018.

[17] J. Manderscheid and A. Sironi and N. Bourdis and D. Migliore and V. Lepetit, *Speed Invariant Time Surface for Learning to Detect Corner Points with Event-Based Cameras*, IEEE Conf. Comput. Vis. Pattern Recog., 2019.

[18] Arnon Amir and Brian Taba and David J Berg and TimothyMelano and Jeffrey L McKinstry and Carmelo Di Nolfo and Tapan K Nayak and Alexander Andreopoulos and Guil-laume Garreau and Marcela Mendoza et al., *A low power, fully event-based gesture recognition system*, IEEE Conf. Comput. Vis. Pattern Recog., 2017.

[19] H. Rebecq and T. Horstschafer and G. Gallego and D. Scaramuzza, *EVO: A geometric approach to event-based 6-DOF paralleltracking and mapping in real-time*, IEEE Robot. Autom. Lett., 2017.

[20] M. Liu and T. Delbruck, *Adaptive time-slice block-matching optical flow algorithm for dynamic vision sensors*, British Mach. Vis. Conf., 2018.

[21] G. Gallego and D. Scaramuzza, *Accurate angular velocity estimation with an event camera*, IEEE Robotics and Automation Letters, 2017.

[22] Lakshmi Annamalai and Anirban Chakraborty and Chetan S. Thakur, *Neuromorphic vision: From sensors to event based algorithms*, WIREs Data mining and knowledge discovery, 2019.

[23] C. S. Thakur and Jamal Molin and G. Cauwenberghs et. al, *Large-Scale Neuromorphic Spiking Array Processors: A quest to mimic the brain*, Frontiers in Neuroscience, 2018.

[24] G. Orchard and C. Meyer and Etienne Cummings and C. Posch and N. Thakor and R. Benosman, *Hfirst: a temporal approach to object recognition*, IEEE Trans. Pattern Anal. Mach. Intell., 2015.

[25] Y. Bi and A. Chadha and A. Abbas and E. Bourtsoulatze and Y. Andreopoulos, *Graph-based object classification for neuromorphic vision sensing*, Proceedings of the IEEE/CVF International Conf. on Comput. Vis., 2019.

[26] Bi and A. Chadha and A. Abbas and E. Bourtsoulatze and Y. Andreopoulos, *Graph-based spatio-temporal feature learning for neuromorphic vision sensing*, IEEE Trans. on Image Processing, 2020.

[27] D Tran and L Bourdev and R Fergus and L Torresani and M Paluri, *Learning spatiotemporal features with 3D convolutional networks*, Proc. IEEE Int. Conf. Comput. Vision, 2015.

[28] K He and X Zhang and S Ren and J Sun, *Deep residual learning for image recognition*, Proc. IEEE Conf. Comput. Vision and Pattern Recog., 2016.

[29] Z Qiu and T Yao and T Mei, *Learning spatio-temporal representation-with pseudo-3D residual networks*, IEEE Int. Conf. Comput. Vis., 2017.

[30] D Tran and H Wang, *A closer look at spatiotemporal convolutions for action recognition*, Proc. IEEE Conf. Comput. Vis. and Pattern Recog., 2018.

[31] K Hara and H Kataoka and Y Satoh, *Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and ImageNet*, IEEE Conf. Comput. Vis. and Pattern Recog., 2018.

[32] J Carreira and A Zisserman, *Quo vadis, action recognition? A new model and the kinetics dataset*, Proc. IEEE Conf. Comput. Vis. and Pattern Recog., 2017.

[33] Tulyakov Stepan and Francois Fleuret and Martin Kiefel and Peter Gehler and Michael Hirsch, *Learning an event sequence embedding for dense event-based deep stereo*, Proceedings of the IEEE/CVF Int. Conf. on Comput. Vis., 2019.

[34] Cubuk and Zoph and Mane and Vasudevan and Le, *Autoaugment: Learning augmentation strategies from data*, Proc. IEEE Conf. Comput. Vis. and Pattern Recog., 2019.

[35] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[36] Wang, Qinyi, et al., *Space-time event clouds for gesture recognition: From RGB cameras to event cameras*, IEEE Winter Conference on Applications of Computer Vision (WACV), 2019.

[37] Lakshmi A., Chakraborty A., Thakur C. S., *EvAn: Neuromorphic Event-based Anomaly Detection*. Frontiers in Neuroscience, 2021.

[38] A.R. Mangalore, C.S. Seelamantula, C.S. Thakur, *Neuromorphic Fringe Projection Profilometry*, IEEE Signal Processing Letters, 2020.

[39] B. R. Pradhan, Y. Bethi, S. Narayanan, A. Chakraborty, Thakur, C. S, *n-HAR: A Neuromorphic Event-Based Human Activity Recognition System Using Memory Surfaces*, IEEE International Symposium on Circuits and Systems (ISCAS), 2019