Efficient spatio-temporal feature clustering for large event-based datasets

Omar Oubari1,* Georgios Exarchakis1,2,3, Gregor Lenz1, Ryad Benosman1,4 and Sio-Hoi Ieng1,*

1 Sorbonne Université, INSERM, CNRS, Institut de la Vision, 17 rue Moreau, F-75012 Paris, France
2 Institut de chirurgie guidée par l’Image, Strasbourg, France
3 ICube, CNRS, University of Strasbourg, France
4 University of Pittsburgh, Pittsburgh, United States of America

* Author to whom any correspondence should be addressed.
E-mail: omaroubari@gmail.com and sio-hoi.ieng@upmc.fr

Keywords: clusterings, asynchronous vision, event-based processing, feature extraction, Gaussian mixture model

Abstract
Event-based cameras encode changes in a visual scene with high temporal precision and low power consumption, generating millions of events per second in the process. Current event-based processing algorithms do not scale well in terms of runtime and computational resources when applied to a large amount of data. This problem is further exacerbated by the development of high spatial resolution vision sensors. We introduce a fast and computationally efficient clustering algorithm that is particularly designed for dealing with large event-based datasets. The approach is based on the expectation-maximization (EM) algorithm and relies on a stochastic approximation of the E-step over a truncated space to reduce the computational burden and speed up the learning process. We evaluate the quality, complexity, and stability of the clustering algorithm on a variety of large event-based datasets, and then validate our approach with a classification task. The proposed algorithm is significantly faster than standard k-means and reduces computational demands by two to three orders of magnitude while being more stable, interpretable, and close to the state of the art in terms of classification accuracy.

1. Introduction
Event-based vision sensors such as the DVS [1] and ATIS [2] promise several advantages over conventional cameras, including a high dynamic range and independent pixels that are directly driven by scene activity at microsecond temporal resolution. These sensors take inspiration from the biological retina by replicating the low redundancy acquisition mechanism while consuming little power. Artificial intelligence and robotics applications might benefit from algorithms developed around such a sensory modality, especially when computing resources are limited.

A large spectrum of techniques is associated with the processing of events. On one end, we encounter algorithms that integrate events over a long time interval to produce frames which can then be processed with classical computer vision and machine learning techniques [3–5]. Naively reducing a large number of events to a frame is likely to remove information that is relevant to computing tasks such as object recognition or sensory-motor integration. Therefore, we choose to focus on algorithms that process individual events as they would maintain the benefits associated with event-based sensors [6].

In this work, we tackle the problem of learning from events through a probabilistic learning approach based on Gaussian mixture models (GMMs). To keep track of the scene dynamics, we use time-surfaces which provide us with the spatiotemporal context surrounding an event [7]. The time-surface, as further discussed in section 2.1, allows us to extend probabilistic learning techniques to events without breaking the asynchronous nature of the events.
1.1. Feature extraction on event streams

Event-based cameras are becoming increasingly popular among the computer vision community as reviewed in [8], but extracting features from a stream of events is not particularly straightforward. The sparse and asynchronous nature of the data is incompatible with algorithms such as backpropagation which form the backbone of modern deep learning [9, 10]. In an effort to leverage existing machine learning and computer vision models, a straightforward approach is to accumulate events into frames or volumes of fixed or variable lengths [4, 11–14]. The reliance on such methods negates some of the advantages of event-based vision sensors, namely the power efficiency and the high temporal resolution which are central to the neuromorphic computing paradigm. Other approaches involve converting the stream of events into geometric data structures such as 3D point clouds that can be processed by neural network architectures [15, 16] based on PointNet [17].

Spiking neural networks (SNN) are a more natural choice for dealing with events [18]. They work asynchronously to accumulate incoming events at a similar microsecond time scale as the event-based sensors. Until recently, SNNs could not reach state-of-the-art accuracy in image classification tasks. Direct training approaches such as gradient-based learning methods are showing great promise in bridging the performance gap in object recognition tasks [19–22]. Alternatively, indirect training approaches aim to convert deep convolutional neural networks into SNNs for efficient inference [23–27]. SNNs are costly to simulate on the Von Neumann architecture, and dedicated hardware is usually required to unlock their full potential [28, 29].

Another category of feature extraction algorithms that work on an event-by-event basis such as hierarchy of time-surfaces (HOTS) [7, 30, 31] or histogram of averaged time-surfaces (HATS) [32], makes use of grid-like structures called time-surfaces that take into consideration the spatio-temporal context around each event. Time-surfaces are used in combination with a clustering algorithm to extract features in a manner that takes advantage of the sparse asynchronous nature of the data, potentially achieving very low latency [32]. These clustering-based methods work particularly well on a smaller scale but they can run into problems with a high number of cluster centers or large event streams [33]. To put this into perspective, the latest high-definition event-based cameras can output up to 50 million events per second [34]. Current event-based clustering algorithms generally struggle to keep up at that scale and need to resort to downsampling and other data preprocessing methods.

1.2. GMMs and events

The GMM is a probabilistic machine learning model that spans a wide range of applications from astrophysics [35] to computer vision [36, 37] and speech recognition [38]. For computer vision especially, identifying patterns of local image features with GMMs is arguably one of the most well-studied approaches [39]. Parameters of the model are typically optimized using expectation minimization (EM) which is an algorithm for maximum likelihood estimation. The complexity of a GMM learning model was until recently bound to be a function of the number of data points and the number of mixture components. As datasets grow larger, it is increasingly difficult to apply traditional clustering methods and improvements in computational complexity are all the more necessary. Novel algorithms for training mixture models with increased efficiency [40, 41] are particularly promising for learning from large datasets.

In this work, we present a method for fast and efficient EM-based learning of Gaussian mixtures and we apply it to large event-based datasets. Probabilistic event generation models that account for sensor noise are typically represented by a Gaussian distribution as event camera noise statistics are still not completely understood [8, 42]. The proposed algorithm is sub-linear in computational complexity with respect to the number of mixed Gaussian distributions. As GMMs are highly sensitive to initialization [43], the latest advances in seeding algorithms allow us to produce good initial clusters without compromising scalability to very large datasets [44, 45]. Lightweight coresets can optionally be used to further reduce algorithmic complexity [46, 47].

Our approach introduces a stochastic method for identifying a truncated approximation of the posterior during learning. Truncated approximations [48] have been previously employed on latent variable models to identify subspaces of the posterior that contain most of the posterior mass [49–52]. Empirical evidence has shown an improved performance in avoiding local optima during learning [53, 54], however, truncation is typically carried out using deterministic methods which are inherently more sensitive to the local optimum problem. To our knowledge, no work explores stochastic variants for truncation. The proposed algorithm maintains a record of cluster similarities to identify clusters with a high probability of being near a datapoint without having to compute the posterior over all dimensions. The extension to event-based vision is ensured
by an approach that uses time-surfaces [7], a feature representation that relates events that are close to each other in space and time. The combination of these two objectives leads to the proposed event-based GMM model.

We first evaluate the performance, quality, and stability of the algorithm on large event-based datasets that have been extensively used in the relevant literature such as POKER-DVS, a 4-class playing card pips dataset [55]; N-MNIST, a handwriting recognition dataset [56]; N-CARS, a car classification dataset [32]; and DvsGesture, a gesture recognition dataset [57]. The proposed algorithm is two to three orders of magnitude faster and more efficient than standard k-means and can be applied to a wide variety of tasks without compromising cluster quality and stability which is important for reproducibility and consistent inference. We then run a classification task on the same datasets and compare our method to state-of-the-art efficient clustering techniques [58, 59] as well as some of the event-based feature extraction methods discussed in the previous section. Our algorithm manages to reach near state-of-the-art accuracy and is competitive against most current event-based learning methods, considering particularly the significant speedups in feature extraction.

2. Methods

2.1. Time-surfaces as spatio-temporal features

Event-based sensors register a change in visual information as an event \( e = (x, y, t, p) \) characterized by spatial coordinates \( x, y \), a timestamp \( t \), and a polarity \( p \in \{-1, 1\} \) indicating whether the luminance increased or decreased. As events are generated, they can be used to build and update time-surfaces (see figure 1). The polarity value does not carry useful information for this study and is therefore not taken into account.

Time-surfaces are 2D structures that make it possible to apply well-known machine learning algorithms on event streams by taking into consideration the spatio-temporal context around each event as formally introduced in [7]. They can encode both shape and motion by taking advantage of the temporal relation between neighboring events. For an event \( e_i \) occurring on pixel \((x_i, y_i)\) at time \( t_i \), we can create a time-surface \( s_i \) by applying an exponential decay kernel to the timestamps of the most recent events within a \((2r + 1)^2\) spatial neighborhood \( M_i \) of radius \( r \) such that:

\[
s_i : M_i \subset \mathbb{R}^2 \rightarrow \mathbb{R}
\]

\[
(x_k, y_k)^T \mapsto s_i(x_k, y_k) = \exp\left(\frac{t_k - t_i}{\tau}\right)
\]  

(1)

where \( \tau \) is the decay constant reflecting the scene dynamics. Since \( t_i \) is the time of the current and most recent event, we always have \( t_k \leq t_i \).

The exponential decay is a costly function to compute. As an alternative, we can use a piece-wise linear function to speed up the processing of events into time-surfaces such that:

\[
s_i : M_i \subset \mathbb{R}^2 \rightarrow \mathbb{R}
\]

\[
(x_k, y_k)^T \mapsto s_i(x_k, y_k) = \max\left(0, \frac{t_k - t_i}{\tau'} + 1\right),
\]

(2)

where \( \tau' \) is a decay constant similar to \( \tau \).

In practice, we generate the discretized form, \( s^{(i)} \), of the time-surface using the Tonic Python package [60]. \( s^{(i)} \) is a \( D \)-dimensional vector which can be rasterized into a grid of pixels centered around the pixel that recorded the event \( e_i \). The radius \( r \) is therefore reduced to a \( \rho \times \rho \)-grid, with \( D = \rho \times \rho \). This provides the time-surface with a layout similar to an image patch with the only difference being that pixel intensities encode the time difference with earlier events.

2.2. Stochastic EM approximation on GMMs

Given a dataset of \( N \) events, a set \( S = \{s^{(1)}, \ldots, s^{(N)}\} \) of time-surfaces is generated from all events according to equations (1) or (2) and is used as input to train the proposed clustering algorithm. Once the model converges, the cluster centers represent the elementary time-surfaces encountered in the dataset and can be used as features for classification. To classify an event stream into a specific group, the time-surfaces are assigned to the nearest cluster centers and a histogram of features is generated. Both non-linear and linear classifiers can be used at this stage, but we focus on the latter for increased efficiency and easily reproducible results. Figure 2 illustrates the GMM-based clustering pipeline from the input to the labeled events.

The clustering algorithm proposed in this work is based on GMMs and it assumes that each data point is distributed according to an observed random variable \( Y \) that follows one of a finite set of \( M \) multivariate
Figure 1. Time-surface generation. (A) An event-based camera records a visual scene (e.g., a moving car) and outputs a stream of events. Events indicating an increase in illuminance are represented in red and a decrease in illuminance is represented in black. (B) For an event $e_i$ occurring on pixel $(x_i, y_i)$ at time $t_i$ (white), we apply an exponential decay on the timestamps of the most recent events within a region of interest $\mathcal{M}_i$ centered at $e_i$. (C) The resulting descriptor is called a time-surface and it considers the spatio-temporal context of the event $e_i$: the value at the center is equal to 1 and all the other values have an intensity exponentially decaying to 0 depending on the time difference between the center event $e_i$ and the most recent events generated by neighboring pixels. A time-surface resembles an image patch that contains high-precision information for the content and motion of a scene.

Figure 2. Proposed event-based classification pipeline. (A) A single instance of an event-based dataset takes the form of an event stream. In each instance, time-surfaces are generated for all events and are then used to train the proposed clustering algorithm. (B) After training, time surfaces are assigned to the corresponding cluster centers which can be used as features for classification. Gaussian probability densities:

$$p(Y = s^{(n)}|C = c; \theta) = \mathcal{N}(Y = s^{(n)}|\mu_c, \sigma I)$$

with a uniform prior probability distribution:

$$p(C = c; \theta) = \frac{1}{M}$$

where $\theta = \{\mu_1:M, \sigma I, \alpha_1:M\}$ denotes the model parameters and $C$ is a set of latent variables that take values in $\{1, \ldots, M\}$.

A preliminary statistical analysis of event-based data shows that the typical covariance matrix of time-surfaces is a near-identity matrix compared to standard image features as shown in figure 3. This is priming our choice of using a single isotropic covariance $\sigma^2 I$ for the GMM, where $I$ denotes a $D \times D$ identity matrix. Each Gaussian distribution is therefore defined as:

$$p(Y = s^{(n)}|C = c; \theta) = \left(2\pi\sigma^2\right)^{-\frac{D}{2}} \exp\left(-\frac{d_t^{(n)}}{2\sigma^2}\right)$$

where $d_t^{(n)} = \|s^{(n)} - \mu_c\|^2$ is the squared Euclidean distance between the data point $s^{(n)}$ and the mean of the Gaussian indexed by $c$. 
The typical approach to fitting a GMM to a dataset is through the use of the EM algorithm \[61\], a two-step iterative process that maximizes the likelihood of a model with respect to \( \theta \):

\[
\mathcal{L}(\theta) = \prod_{n=1}^{N} p(Y = \mathbf{s}^{(n)}|\theta) = \prod_{n=1}^{N} \sum_{c=1}^{M} p(Y = \mathbf{s}^{(n)}, C = c|\theta) p(C = c|\theta).
\] (6)

Explicitly maximizing \( \mathcal{L}(\theta) \) for models with latent variables can be difficult \[62\]. Instead, the EM algorithm starts by computing a lower-bound to the log-likelihood in the E-step. This consists in identifying the posterior distribution at iteration \( t \):

\[
P_c^{(n)} = \frac{\exp(-d_c^{(n)}/2\sigma^2)}{\sum_{c'=1}^{M} \exp(-d_{c'}^{(n)}/2\sigma^2)} \delta(c \in \mathcal{K}^{(n)}).
\] (7)

The posterior probabilities are then used in the M-step to update the model parameters and maximize the lower-bound. These two steps are repeated until convergence of the parameter estimates.

The complexity of EM on GMMs is \( \mathcal{O}(NMD) \) making it already a very efficient algorithm. Nevertheless, estimating the posterior is computationally intensive, particularly with large event-based datasets. To achieve better scaling behavior and increase computational efficiency, we rely on a truncated optimization scheme \[48, 52\]. Instead of calculating \( p_c^{(n)} \) we propose to estimate the posterior over a truncated subspace of clusters \( \mathcal{K}^{(n)} \subset \{1, \ldots, M\} \), with \( |\mathcal{K}^{(n)}| = H \) as:

\[
q_c^{(n)} = \frac{\exp(-d_c^{(n)}/2\sigma^2)}{\sum_{c' \in \mathcal{K}^{(n)}} \exp(-d_{c'}^{(n)}/2\sigma^2)} \delta(c \in \mathcal{K}^{(n)}).
\] (8)

The posterior for clusters \( c \notin \mathcal{K}^{(n)} \), is never estimated and therefore \( d^{(n)} \) allows for a much faster implementation than \( p_c^{(n)} \). In terms of total variation, \( d_{TV}(q_c^{(n)}, p_c^{(n)}) = \sum_{c \in \mathcal{K}^{(n)}} q_c^{(n)} = 1 - \sum_{c \in \mathcal{K}^{(n)}} p_c^{(n)} \), the approximation is optimal when \( \mathcal{K}^{(n)} \) holds the clusters that account for most of the posterior mass.

In order to identify the set \( \mathcal{K}^{(n)} \) (equation (10)), we start by initializing it with \( H \) clusters drawn uniformly at random. At each iteration \( t \), we update \( \mathcal{K}^{(n)} \) by comparing it with \( R \) clusters, randomly sampled from a proposal distribution \( p(C_{t+1}|C_{t} = \bar{c}_t; S) \) according to equation (9), and maintaining the \( H \) best clusters:

\[
p(C_{t+1}|C_{t} = \bar{c}_t; S) = \frac{S_{\bar{c}_t,c}}{\sum_{c' \in \mathcal{K}^{(n)}(S_{\bar{c}_t,c'})} \delta(c \notin \mathcal{K}^{(n)})}
\] (9)

where \( \bar{c}_t = \arg \min_{c} \{d_c^{(n)}|c \in \mathcal{K}^{(n)}\} \) and \( S \in \mathbb{R}^{M \times M} \) is a cluster similarity matrix that assigns an increasing importance \( S_{i,j} \) to cluster pairs \( \{i, j\} \) that are near the same time-surfaces. In other words, the proposal distribution at iteration \( t \) represents the neighborhood of the cluster \( \bar{c}_t \) that was closest to the time-surface \( s^{(n)} \) in the previous iteration.

\[
\mathcal{K}^{(n)} = \left\{ \{c_{t+1}|C_{t+1} = \bar{c}_t; \theta\} \cup \{\mathcal{K}^{(n)} \cup \{\mathcal{K}^{(n)} \cup \{c_{t+1}|C_{t+1} = \bar{c}_t; \theta\} \} \} \right\}_{c \in \mathcal{K}^{(n)}}
\] (10)

where \( \cdot \)^{H}_{\cdot|\cdot} \) is an operator that keeps the \( H \) smallest elements of the set ordered by the distance \( d_c^{(n)} \). The hyperparameters \( \{H, R\} \) fully specify the approximation and enable control of the complexity of the algorithm directly as opposed to a precision estimate in typical approximations.

Figure 3. Covariance matrices of 9 \times 9 time-surfaces on N-MNIST (A) and 9 \times 9 image patches with stride \( s = 1 \) on MNIST (B).
of each M-step with process instead of the proposal distribution i.e. after the first iteration, constrain the algorithm to update we refer to as the data similarity GMM (D-GMM).

In order to highlight the importance of stochasticity in avoiding well-known local optima issues we also in order to maintain convergence [63]:

\[ \mathcal{K}^{(n)} = \left\{ \{c_i | c_i \sim p(C = c_i, \theta) \} \right\} \]

Similarly to the exact EM algorithm, we can estimate the parameter updates using the approximate posterior \( q^{(n)}_c \) from equation (8) and still maintain convergence [63]:

\[ \mu_c = \frac{\sum_{n=1}^{N} q^{(n)}_c s^{(n)}}{\sum_{n=1}^{N} q^{(n)}_c d^{(n)}_c} \quad \text{and} \quad \sigma^2 = \frac{1}{DN} \sum_{n=1}^{N} \sum_{c} q^{(n)}_c d^{(n)}_c \]

Iterating between estimating \( q^{(n)}_c \) and the parameter updates (equations (11)–(13)) defines an algorithm we refer to as the data similarity GMM (D-GMM).

D-GMM starts with a randomly initialized \( \mathcal{K}^{(n)} \) that is intended to store the centers more likely to represent that data point. The set \( \mathcal{K}^{(n)} \) is updated using samples from the proposal distribution, at the beginning of each M-step with \( R \) new centers. After estimating the distances between the centers in \( \mathcal{K}^{(n)} \) and the newly sampled centers, we only maintain the \( H \) centers in \( \mathcal{K}^{(n)} \) nearest to the data point. The approximate posterior \( q^{(n)}_c \) is estimated over the set \( \mathcal{K}^{(n)} \) and used to update the centers in the M-step. Since \( q^{(n)}_c \) is inversely proportional to the centers’ distance from the data point \( s^{(n)} \), far away centers would have a very low—exponentially decreasing—value for \( q^{(n)}_c \). The complete procedure is detailed in algorithm 1.

The complexity of the D-GMM method compared to the exact EM algorithm for GMMs reduces from \( \mathcal{O}(NMD) \) to \( \mathcal{O}(NR + H)D \) in the E-step where typically \( R + H \ll M \). The complexity becomes \( \mathcal{O}(NHD + NH^2) \) in the M-step. In section 3, we investigate applications of the algorithm where \( H^2 \) is orders of magnitude smaller than \( M \) and the model is successfully fitted to the data.

In order to highlight the importance of stochasticity in avoiding well-known local optima issues [54] we also constrain the algorithm to update \( \mathcal{K}^{(n)} \) using \( R \) samples drawn uniformly at random throughout the learning process instead of the proposal distribution i.e. after the first iteration, \( \mathcal{K}^{(n)} \) is updated using the following equation instead of equation (10):

\[ \mathcal{K}^{(n)} = \left\{ \{c_i | c_i \sim p(C = c_i, \theta) \} \right\} \]

The uniform variant of D-GMM will be referred to as stochastic GMM (S-GMM).
2.3. Implementation details

2.3.1. Initialization

During the first epoch of the proposed algorithm, we initialize the Gaussian centers \( \mu_{1:k} \) with the AFK-MC\(^2\) method, which is an efficient approximation of the k-means++ seeding algorithm [45]. This algorithm samples an initial center \( \mu_1 \in S \) uniformly at random and then uses it to derive the proposal distribution \( g(s|\mu_1) \). A Markov chain of length \( m \) is used to iteratively sample sufficiently distinct new centers \( \mu_{2:M} \) from the data. AFK-MC\(^2\) requires one pass of complexity \( O(ND) \) through the data to define the proposal distribution \( g(s) \), and the centers are sampled with a complexity of \( O(mFD) \).

2.3.2. Lightweight coresets

To further improve computational efficiency, data summarization strategies such as lightweight coresets can be used to get smaller training sets [47, 64–67]. A coreset \( S' = \{ (s_1, w_1), \ldots, (s_{N'}, w_{N'}) \} \) is a representative subset of a full dataset \( S \) where each data point is weighted according to its significance in the original dataset. The weights \( w_{1:N'} \) are a multiplicative constant on the parameters and therefore the parameter updates become:

\[
\mu_c = \frac{\sum_{n=1}^{N'} w_n d_n^{c(n)} g(n)}{\sum_{n=1}^{N'} w_n d_n^{c(n)}}
\]

(15)

\[
\sigma^2 = \frac{1}{D} \sum_{n=1}^{N'} \sum_{c} w_n d_n^{c(n)} d_n^{c(n)}
\]

(16)

\[
S_{ij} = \frac{1}{N'} \sum_{n=1}^{N'} w_n e^{-\left( (d_n^{i(n)} + d_n^{j(n)}) \right)} \delta(\{i,j\} \subset K^{c(n)}).
\]

(17)

These updates can replace equations (11)–(13) in algorithm 1 to allow applications on a coreset \( S' \). Working on coresets introduces an error in the approximation that has been analyzed rigorously in earlier work [47]. Constructing the coreset requires two iterations of complexity \( O(ND) \) over the data but reduces the complexity of D-GMM to \( O(N'(R + H)D) \) in the E-step and \( O(N'(ND + N'HD^2)) \) in the M-step. The complexity of AFK-MC\(^2\) is also reduced since the proposal distribution is defined on the coreset with complexity \( O(N'D) \).

3. Experiments and results

In this section, the performance and quality of D-GMM (algorithm 1) and the uniform variant, S-GMM, are evaluated in comparison to standard k-means initialized with the k-means++ method [43].

We first assess convergence on an artificial dataset with known ground truth [68] and then move on to an evaluation of the proposed clustering algorithms on a series of event-based vision datasets including POKER-DVS [55], N-MNIST [56], N-CARS [32], and DvsGesture [57]. The last three mentioned datasets are substantially large; to put this into perspective, a single recording of N-MNIST has an average of 4176 events and a total of more than 250 million events in the training set alone. As such, the clustering evaluation for these three datasets was done using a random subset to run the k-means baseline in a reasonable amount of time. Finally, we test the proposed algorithms on a real-world classification task, this time using the full datasets, and we compare them to feature extraction methods specifically designed for dealing with events.

The cluster centers are initialized using the AFK-MC\(^2\) method with a Markov chain length \( m = 5 \). As a stopping criterion the iterative EM process is terminated when the increase in the lower-bound of the log-likelihood is less than \( \epsilon = 10^{-4} \). Our algorithms were implemented on C++ with the help of a vectorization library, and numerical results are reported as an average of 5 trials unless otherwise stated.

As a reminder, the notations used in this work are listed below:

- \( N \): number of data points
- \( N' \): coreset size
- \( M \): number of cluster centers
- \( H \): size of the truncated subspace
- \( R \): number of new samples for the truncated subspace

3.1. Convergence analysis on artificial data

We explore the convergence properties of our truncated stochastic approach on an artificial dataset with \( N = 5000 \) data points and 15 cluster centers [68]. In figure 4 we measure the root mean squared error (RMSE) between learned centers and the ground truth across 100 trials, setting the hyperparameters to \( M = 15, H = 3 \) and \( R = 6 \). D-GMM and S-GMM are better at recovering the actual cluster centers compared to standard
Figure 4. Validation on a two-dimensional artificial dataset. Both D-GMM and S-GMM outperform k-means across 100 trials when comparing the learned centers to the ground truth.

Table 1. Theoretical time complexities.

| Method       | Time complexity (E-step) |
|--------------|--------------------------|
| K-means      | $O(NMD)$                 |
| S-GMM        | $O(N(R + H)D)$           |
| D-GMM        | $O(N(R + H)D)$           |
| HGMR [58]    | $O(N \log(M)D)$          |
| JRMPC [59]   | $O(NMD)$                 |
| Greedy [69]  | $O(NM^2D)$               |

k-means. Results suggest the stochastic approach is more effective at avoiding locally optimal solutions. Additionally, working with a truncated set can improve numeric stability by ignoring low probability clusters.

3.2. Clustering evaluation on event-based data

For a more descriptive analysis, we evaluate the complexity and stability our clustering algorithms on a series of large event-based vision datasets. We avoid using coresets initially to make a fair comparison with the k-means baseline.

3.2.1. Complexity

Table 1 gathers, upon availability from the sources, time complexities during the E-step of state-of-the-art efficient clustering algorithms to which D-GMM and S-GMM are compared. As previously mentioned, we only consider cases where $R + H < M$ as it seems to be sufficient for most applications.

We also experimentally assess the complexity of the proposed algorithms. Figure 5 shows the scaling behavior of our algorithms with an increasing number of clusters $M$ on POKER-DVS (top) and N-CARS (bottom), with $M$ ranging from 200 to 1400 clusters and hyperparameters set to $H = 5$ and $R = 10$. On the left, we show the operations complexity, measured according to the number of distance evaluations $d^{(n)}$, and on the right, the time complexity in seconds. Runtimes were measured on an Intel Core i7-9700K CPU. Results suggest K-means is linear in both time and operations complexity while the proposed algorithms scale sub-linearly. POKER-DVS is a relatively small dataset, so the overhead associated with sampling from the proposal distribution (D-GMM) compared to uniform sampling (S-GMM) does not appear in the runtime. For larger datasets such as N-CARS, the overhead becomes non-negligible, reducing the difference in time complexity between D-GMM and S-GMM compared to the operations complexity.

3.2.2. Stability

We assess the proposed algorithms’ ability to recover the same clusters using different initializations. We cluster N-CARS and POKER-DVS with hyperparameters set to $M = 500$, $H = 5$, and $R = 10$, and compare the
Figure 5. Algorithms complexity analysis on POKER-DVS and N-CARS for an increasing number of clusters. (Left) Operations complexity measure according to the number of distance evaluations. (Right) Time complexity in seconds. (Both) For a fair comparison with the k-means baseline, simulations were done on the full datasets without any coresets.

Figure 6. Stability on POKER-DVS and N-CARS over 5 trials with $H = 5$ and $R = 10$. D-GMM and S-GMM appear to be more stable than k-means in recovering the same clusters with different initialization.

learned centers of each pair of runs using a normalized RMSE. Figure 6 shows the mean and standard deviation between errors. On both datasets, both algorithms show improvements in stability over K-means which is important for reproducibility. The increased stability of S-GMM compared to D-GMM is likely due to a better local optima avoidance when drawing $R$ clusters based on the uniform distribution instead of limiting it to cluster neighborhoods [70]. These results lend further credibility to our rationale for exploring stochastic truncated approximations.

3.2.3. Hyperparameter selection

On POKER-DVS and N-CARS with $M = 500$, when selecting optimal values for the hyperparameters, we set $H = 5$ under the assumption that cluster centers with lower probabilities have a negligible effect because the probability values of the exact posterior decay exponentially. As seen in figure A1 of the appendix, on both datasets, D-GMM is faster and more computationally efficient than both S-GMM and the k-means baseline for values of $R$ in the range of [10, 30]. Beyond that, the overhead associated with sampling values from a proposal distribution impacts the runtime. Although, this is purely dependent on the implementation and can be optimized further. Our choice of $R = 10$ throughout these experiments effectively boils down to a compromise between runtime and computational efficiency.
3.2.4. Performance with coresets

Lightweight coresets can also be used to further improve performance by summarizing the training data [47]. Figure 7 investigates the cluster quality for different coreset sizes on POKER-DVS and N-CARS. This is measured using the relative quantization error with k-means on the full dataset as our baseline according to:

\[ \eta = \frac{Q_{\text{algorithm}} - Q_{k\text{-means}}}{Q_{k\text{-means}}} \]

where \(Q_{\text{algorithm}} = \sum_{i=1}^{N} \min_{\mu_c} \| \mu_c - y^{(i)} \|^2\) is the quantization error of the GMM trained with the respective algorithm. Marker size on these plots represents the number of distance evaluations and results suggest an increase in performance associated with larger coresets. In all cases, the proposed algorithms cluster data with the lowest relative quantization error to the baseline for the least amount of distance evaluations which translates into higher quality clusters than k-means.

We also expand the analysis with coresets on larger event-based datasets. Results are summarized in Table 2. The D-GMM and S-GMM algorithms are parameterized with \(H = 5\) and \(R = 10\) and are allowed to execute a different number of iterations until they satisfy the stopping criterion. Since the truncated algorithms are going through fewer clusters in each iteration, convergence requires more iterations than k-means. Nevertheless, iterations are significantly cheaper in terms of distance evaluations \(d^{(i)}\) leading to the observed increase in speed and computational efficiency. With coresets, D-GMM consistently has the lowest relative quantization error \(\eta\) across all datasets and reaches the highest average speedups in terms of runtime and distance evaluations.

| Dataset      | Algorithm                  | \(N'\) | Relative quantization error \(\eta\) | Distance evaluation speedup | Runtime speedup | Iterations |
|--------------|----------------------------|--------|--------------------------------------|----------------------------|------------------|------------|
| POKER-DVS    | k-means (baseline)         | —      | 0.0%                                 | ×1.0                       | ×1.0             | 42.0       |
| \(N_{\text{train}} = 142,587\) | k-means on coresets       | \(2^{14}\) | 5.0%                                 | ×16.3                      | ×16.1            | 23.0       |
| \(N_{\text{test}} = 63,394\) | S-GMM                     | \(2^{14}\) | 3.0%                                 | ×63.9                      | ×10.4            | 192.2      |
|              | D-GMM                     | \(2^{14}\) | 3.0%                                 | ×170.5                     | ×21.3            | 72.0       |
| N-CARS       | k-means (baseline)         | —      | 0.0%                                 | ×1.0                       | ×1.0             | 48.4       |
| \(N_{\text{train}} = 2,159,967\) | k-means on coresets       | \(2^{15}\) | 4.0%                                 | ×321.8                     | ×294.8           | 28.2       |
| \(N_{\text{test}} = 1,267,723\) | S-GMM                     | \(2^{15}\) | 3.0%                                 | ×1535.2                    | ×474.3           | 195.8      |
|              | D-GMM                     | \(2^{15}\) | 3.0%                                 | ×3703.1                    | ×796.0           | 81.4       |
| N-MNIST      | k-means (baseline)         | —      | 0.0%                                 | ×1.0                       | ×1.0             | 186.6      |
| \(N_{\text{train}} = 4,227,034\) | k-means on coresets       | \(2^{16}\) | 3.0%                                 | ×1166.2                    | ×1025.8          | 66.4       |
| \(N_{\text{test}} = 666,979\) | S-GMM                     | \(2^{16}\) | 2.0%                                 | ×6077.2                    | ×5099.1          | 389.0      |
|              | D-GMM                     | \(2^{16}\) | 2.0%                                 | ×8213.4                    | ×3138.9          | 290.4      |
| DvsGesture   | k-means (baseline)         | —      | 0.0%                                 | ×1.0                       | ×1.0             | 104.8      |
| \(N_{\text{train}} = 10,741,801\) | k-means on coresets       | \(2^{17}\) | 2.0%                                 | ×465.8                     | ×432.1           | 61.2       |
| \(N_{\text{test}} = 3,506,475\) | S-GMM                     | \(2^{17}\) | 1.0%                                 | ×3545.7                    | ×2277.2          | 265.0      |
|              | D-GMM                     | \(2^{17}\) | 1.0%                                 | ×6464.2                    | ×2723.1          | 145.4      |

*For all datasets \(D = 81\) and \(M = 500\).
3.3. Classification on event-based data

As an unsupervised learning method, clustering is useful for data exploration and to find structure in unlabeled data. For benchmarking purposes, we run a classification task on the same four event-based datasets used in the previous section.

To create features for training D-GMM and S-GMM, we generate time-surfaces for each event according to equation (1). Parameters are set to side length \( r = 5 \), decay factor \( \tau = 80000 \mu s \), and polarity \( p = 1, \forall p \in \{-1, 1\} \), resulting in a patch of dimension \( D = r \times r = 25 \). After training, each time-surface is assigned to a cluster center. In order to take into account the spatial component of a visual scene, we split the pixel array into sub-regions using a \( 4 \times 4 \) grid. We use the cluster centers in each sub-region to create spatial feature histograms describing the local statistics of the visual scene. From there, we concatenate these spatial histograms into a global histogram used to train a logistic regression classifier. The global histogram is normalized so that each dimension has zero mean and unit variance. This pre-processing step is important to get faster convergence.

In table 3, we provide an overview of the classification results compared to supervised and unsupervised methods specifically designed to deal with event-based datasets such as HOTS [7], HATS [32] and BLS [72]. To better highlight the significance of our event-based clustering pipeline, we also run the classification task on web-scale k-means [73], a popular k-means approximation, and other efficient clustering algorithms optimized for three-dimensional point clouds such as HGMR [58] and JRMPC [59]. For the last two methods, instead of generating time surfaces, we extract \( xyz \) point clouds from our data. Classification results on our algorithms are reported as an average over 5 trials; the rest is taken from the literature [7, 20, 32, 72, 74].

We set the hyperparameters to \( M = 500, N = 2^{16}, H = 5 \) and \( R = 10 \), effectively avoiding any data-specific parameter tuning to showcase the low parametrization effort required to get close to the state of the art in event-based classification. Both proposed algorithms have a similar accuracy on most datasets and show improvements over web-scale k-means. These results are in agreement with the clustering analysis which showed better local optima avoidance when working with a stochastic approximation to the EM algorithm over a truncated subspace. The classification accuracy of HGMR and JRMPC is mainly impacted by the temporal component in the data. Treating events as point clouds instead of simply pre-processing them into time-surfaces is a complex problem that is treated by a separate body of work [15–17]. The proposed clustering algorithm is better at classifying event-based datasets than most unsupervised feature extraction methods except for HATS which needs to have access to all past events to build average time-surfaces as evidenced by the use of a large decay constant. This process is costly and not scalable to large streams of events. Furthermore, HATS requires significant parametrization efforts for each new dataset compared to our method and makes use of a stronger and less interpretable classifier. State-of-the-art methods such as SLAYER [20] rely on multilayer networks to perform better on classification tasks. Using similar constraints, we also benchmark our method with a stronger classifier, a multi-layer perceptron (MLP) with one hidden layer of 256 neurons. Compared to the logistic regression, classification accuracy is improved across all datasets. Our model reaches an accuracy of 91.6% on DvsGesture and 90.38% on N-CARS, which is the best score for a clustering-based approach. Our results are noteworthy considering the speed and efficiency of our pipeline, which is more suitable for resource-constrained environments.

| Method          | DVS  | N-CARS | N-MNIST | DvsGesture |
|-----------------|------|--------|---------|------------|
| S-GMM           | 100% | 85.92% | 98.42%  | 88.03%     |
| S-GMM + MLP     | 100% | 90.38% | 98.77%  | 91.6%      |
| D-GMM           | 100% | 85.99% | 98.38%  | 87.65%     |
| Web-scale K-means [73] | 100% | 84.76% | 98.18%  | 84.47%     |
| HGMR [58]       | 100% | 79.44% | 96.96%  | 50.45%     |
| JRMPC [59]      | 94%  | 76.73% | 81.82%  | 63.8%      |
| H-FIRST [74]    | 91.6%| 56.1%  | 71.2%   |            |
| HOTS [7]        | 95%–100% | 62.4%   | 80.8%   |            |
| Gabor-SNN [32]  | 78.9%| 83.7%  |         |            |
| HATS [32]       | 90.2%| 99.1%  |         |            |
| SLAYER [20]     |      |         | 99.2%   | 93.64%     |
| BLS [72]        | 92.9%| 98.3%  |         |            |
| ResNet-34 [75]  | 90.9%| 98.4%  |         |            |

Table 3. Classification accuracy on large event-based datasets.
4. Discussion and conclusion

This paper introduced a novel clustering algorithm based on GMMs and tailored for large streams of events. The speed and computational efficiency of the proposed algorithm are significantly increased without adding complex extraneous computation unlike other efficient approximations found in the literature [41]. The algorithm is also more stable than k-means in recovering cluster centers which is important for reproducibility. A fast and user-friendly C++ implementation is publicly available. The simplicity of the algorithm also makes it easily transferable to widely used machine learning frameworks.

The speedup in operations and time complexity is particularly interesting when dealing with high-definition event-based cameras or in the context of neuromorphic computing applications with very large datasets. The time complexity of current event-by-event algorithms quickly becomes prohibitive after a certain number of events [7, 20, 32], an issue that is not well addressed in the literature due to the reliance on subsets, small datasets, and event-based cameras with low spatial resolutions.

The proposed clustering algorithm exploits the extremely sparse nature of event-based data by building time surfaces that extract the spatio-temporal neighborhood around each event. These features can then be combined in a bag-of-words approach for classification with a simple linear model. At this stage, non-linear classifiers can be explored for even better classification accuracy at the cost of efficiency and interpretability. This embodies the current state of neuromorphic computing which is associated with a trade-off between efficiency and accuracy. Pre-processing events into time-surfaces allows retaining fine-grained temporal information about scene dynamics that is typically lost when binning events to frames. This step turns out to be essential for clustering event-based data due to the non-homogeneous dimensions of raw events.

The constraints introduced to the covariance structure can affect the expressive power of our algorithm; however, event-based pixels are not strongly correlated and can be described using simpler statistics. Less constrained models were briefly explored, and as expected, non-isotropic covariance matrices significantly increased complexity with limited returns.

To maximize the potential of our algorithm, we can combine it with lightweight coresets and AFK-MC² initialization [45, 47] which are state of the art methods in the literature for seeding cluster centers and data summarization respectively. Future works aim at exploring an online learning variant of the proposed algorithm through incremental parameter updates.

Acknowledgments

This research was supported by the European Unions Horizon 2020 Research and Innovation Program under Grant Agreement No. 732642.

Data availability statement

The code used in this article is available online at: https://github.com/neuromorphic-paris/peregrine.

Conflict of interest

The authors declare that they have no conflict of interest.

Appendix.

Figure A1 shows the results of a hyperparameter sweep to determine the optimal value for the number of new samples $R$ in each EM iteration. The distance evaluation speedup and runtime speedup are measured compared to the k-means baseline according to the following equation:

$$S_{\text{distance}} = \frac{D_{\text{algorithm}}}{D_{\text{baseline}}}$$
$$S_{\text{runtime}} = \frac{R_{\text{algorithm}}}{R_{\text{baseline}}}$$

where $S$ is the speedup, $D$ is the number of distance evaluations, and $R$ denotes the algorithm runtime.
**Figure A1.** $R$ hyperparameter search on POKER-DVS and N-CARS for hyperparameters $M = 500$ and $H = 5$. (Left) Hyperparameter search for D-GMM. (Right) Hyperparameter search for S-GMM. Results are presented on a logarithmic scale relative to the K-means baseline shown as a dotted line (lower is better). Values of $R$ ranging from 10 to 40 seem to be optimal in terms of runtime and distance evaluations.

**ORCID iDs**

Georgios Exarchakis [https://orcid.org/0000-0003-2517-5782](https://orcid.org/0000-0003-2517-5782)
Gregor Lenz [https://orcid.org/0000-0003-0619-3555](https://orcid.org/0000-0003-0619-3555)
Sio-Hoi Ieng [https://orcid.org/0000-0002-0030-6574](https://orcid.org/0000-0002-0030-6574)

**References**

[1] Lichtsteiner P, Posch C and Delbrück T 2008 A 128 × 128 120 dB 15 us latency asynchronous temporal contrast vision sensor *IEEE J. Solid-State Circuits* 43 566–76
[2] Posch C, Matolin D and Wohlgemant R 2011 A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS *IEEE J. Solid-State Circuits* 46 259–75
[3] Kogler J, Subbachner C and Kubinger W 2009 Bio-inspired stereo vision system with silicon retina imagers *Int. Conf. on Computer Vision Systems (ICVS)* pp 174–83
[4] Rebecq H, Ranftl R, Kolun V and Scaramuzza D 2019 Events-to-video: bringing modern computer vision to event cameras *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* pp 3857–66
[5] Gehrig D, Rebecq H, Gallego G and Scaramuzza D 2020 Eklt: asynchronous photometric feature tracking using events and frames *Int. J. Comput. Vis.* 128 601–18
[6] Benosman R, Clercq C, Lagorce X, Ieng S and Bartolozzi C 2014 Event-based visual flow *IEEE Trans. Neural Netw. Learn. Syst.* 25 407–17
[7] Lagorce X, Orchard G, Galluppi F, Shi B E and Benosman R B 2016 Hots: a hierarchy of event-based time-surfaces for pattern recognition *IEEE Trans. Pattern Anal., Mach. Intell.* 39 1346–59
[8] Gallego G et al 2019 Event-based vision: a survey (arXiv:1904.08405)
[9] LeCun Y, Bottou L, Bengio Y and Haffner P 1998 Gradient-based learning applied to document recognition *Proc. IEEE* 86 2278–324
[10] Krizhevsky A, Sutskever I and Hinton G E 2012 Imagenet classification with deep convolutional neural networks *Advances in Neural Information Processing Systems* pp 1097–105
[11] Tedaldi D, Gallego G, Muegler F and Scaramuzza D 2016 Feature detection and tracking with the dynamic and active-pixel vision sensor (davis) 2016 2nd. Int. Conf. on Event-Based Control, Communication, and Signal Processing (EBCCSP) (Piscataway, NJ: IEEE) pp 1–7
[12] Maqueda A I, Loquercio A, Gallego G, García N and Scaramuzza D 2018 Event-based vision meets deep learning on steering prediction for self-driving cars *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* pp 5419–27
[13] Munda G, Reinhacker C and Pock T 2018 Real-time intensity-image reconstruction for event cameras using manifold regularisation *Int. J. Comput. Vis.* 126 1381–93
[14] Scheerlinck C, Rebecq H, Gehrig D, Barnes N, Mahony R and Scaramuzza D 2020 Fast image reconstruction with an event camera
The IEEE Winter Conf. on Applications of Computer Vision pp 156–63

[15] Wang Q, Zhang Y, Yuan J and Lu Y 2019 Space-time event clouds for gesture recognition: from rgb cameras to event cameras 2019 IEEE Winter Conf. on Applications of Computer Vision (WACV) (Piscataway, NJ: IEEE) pp 1826–35

[16] Sekikawa Y, Hara K and Saito H 2019 EventNet: asynchronous recursive event processing Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition pp 3887–96

[17] Qi C R, Su H, Mo K and Guibas L J 2017 PointNet: deep learning on point sets for 3D classification and segmentation Proc. IEEE Conf. on Computer Vision and Pattern Recognition pp 652–60

[18] Maass W 1997 Networks of spiking neurons: the third generation of neural network models Neural Netw. 10 1659–71

[19] Lee J H, Delbruck T and Pfeiffer M 2016 Training deep spiking neural networks using backpropagation Front. Neurosci. 10 508

[20] Shrestha S B and Orchard G 2018 Slayer: spike layer error reassignment in time Advances in Neural Information Processing Systems pp 1412–21

[21] Kaiser J, Mostafa H and Neftci E 2020 Synaptic plasticity dynamics for deep continuous local learning (decolle) Front. Neurosci. 14 424

[22] Kheradpisheh S R and Masquelier T 2020 S4nn: temporal backpropagation for spiking neural networks with one spike per neuron Int. J. Neural Syst. 30 2050027

[23] Cao Y, Chen Y and Khosla D 2015 Spiking deep convolutional neural networks for energy-efficient object recognition Int. J. Comput. Vis. 113 54–66

[24] Rueckauer B, Lungu J-A, Hu Y, Pfeiffer M and Liu S-C 2017 Conversion of continuous-valued deep networks to efficient event-driven networks for image classification Front. Neurosci. 11 682

[25] Zhang L, Zhou S, Zhi T, Du Z and Chen Y 2019 Tdann: from deep neural networks to deep spike neural networks with temporal-coding Proc. AAAI Conf. on Artificial Intelligence vol 33 pp 1319–26

[26] Lee C, Sarwar S S, Panda P, Srinivasan G and Roy K 2020 Enabling spike-based backpropagation for training deep neural network architectures Front. Neurosci. 14 119

[27] Kugele A, Pfeil T, Pfeiffer M and Chicca E 2020 Efficient processing of spatio-temporal data streams with spiking neural networks Front. Neurosci. 14 439

[28] Furber S and Temple S 2008 Neural systems engineering Computational Intelligence: A Compendium (Berlin: Springer) pp 763–96

[29] Davies M et al 2018 Loihi: a neuromorphic manycore processor with on-chip learning IEEE Micro 38 82–99

[30] Haessig G and Benosman R 2018 A sparse coding multi-scale precise-timing machine learning algorithm for neuromorphic event-based sensors Micro-and Nanotechnology Sensors, Systems, and Applications X (International Society for Optics and Photonics) vol 10639 pp 106391U

[31] Afshar S, Xu Y, Tapson J, van Schaik A and Cohen G 2019 Event-based feature extraction using adaptive selection thresholds (arXiv:1907.07855)

[32] Sironi A, Brambilla M, Bourdis N, Lagorce X and Benosman R 2018 Hats: histograms of averaged time surfaces for robust event-based object classification Proc. IEEE Conf. on Computer Vision and Pattern Recognition pp 1731–40

[33] Perot E, de Tournemire P, Nitti D, Masci J and Sironi A 2020 Learning to detect objects with a 1 megapixel event camera (arXiv:2009.13436)

[34] Chen S and Guo M 2019 Live demonstration: Celex-v: a 1 m pixel multi-mode event-based sensor Proc. IEEE Conf. on Computer Vision and Pattern Recognition Workshops

[35] VanderPlas J, Connolly A J, Ivezić Z and Gray A 2012 Introduction to astroml: machine learning for astrophysics 2012 Conf. on Intelligent Data Understanding (Piscataway, NJ: IEEE) pp 47–54

[36] Kurata T, Okuma T, Kourogi M and Sakaue K 2001 The hand mouse: Gmm hand-color classification and mean shift tracking Proc. IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems (Piscataway, NJ: IEEE) pp 119–24

[37] Zhou D and Zhang H 2005 Modified gmm background modeling and optical flow for detection of moving objects 2005 IEEE Int. Conf. on Systems, Man and Cybernetics, Piscataway, NJ IEEE vol 3 pp 2224–9

[38] Fujimoto M and Riki Y 2004 Robust speech recognition in additive and channel noise environments using gmm and em algorithm 2004 IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (Piscataway, NJ: IEEE) vol 1 pp 1–941

[39] Staufer C and Grimson W E L 1999 Adaptive background mixture models for real-time tracking Proc. 1999 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (Cat. No. PR9949) (Piscataway, NJ: IEEE) vol 2 pp 246–52

[40] Forster D and Lücke J 2017 Can clustering scale sublinearly with its clusters? A variational em acceleration of gmm and k-means (arXiv:1711.03431)

[41] Hirschberger F, Forster D and Lücke J 2021 A variational em acceleration for efficient clustering at very large scales IEEE Trans. Pattern Anal. Mach. Intell. https://doi.org/10.1109/TPAMI.2021.3133763

[42] Gallego G, Lund J E, Muegger D, Rebecq H, Delbruck T and Scaramuzza D 2017 Event-based, 6-dof camera tracking from photometric depth maps IEEE Trans. Pattern Anal. Mach. Intell. 40 2402–12

[43] Arthur D and Vassilvitskii S 2006 k-means++: the advantages of careful seeding Tech. Rep. (Stanford)

[44] Bachem O, Lucic M, Hassani S H and Krause A 2016 Approximate k-means++ in sublinear time 13th AAAI Conf. on Artificial Intelligence

[45] Bachem O, Lucic M, Hassani S H and Krause A 2016 Fast and provably good seedings for k-means (arXiv:2009.13436)

[46] Bachem O, Lucic M, Hassani S H and Krause A 2017 Approximate k-means (arXiv:1711.03431)

[47] Bachem O, Lucic M and Krause A 2018 Scalable k-means clustering via lightweight coresets Proc. 24th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining pp 1119–27

[48] Lücke J and Eggert J 2010 Expectation truncation and the benefits of preselection in training generative models J. Mach. Learn. Res. 11 2835–900

[49] Dai Z, Exarchakis G and Lücke J 2013 What are the invariant occlusive components of image patches? A probabilistic generative approach Advances in Neural Information Processing Systems pp 243–51

[50] Exarchakis G and Lücke J 2017 Discrete sparse coding Neural Comput. 29 2979–3013

[51] Forster D and Lücke J 2017 Truncated variational em for semi-supervised neural simpletrons 2017 Int. Joint Conf. on Neural Networks (IJCNN) (Piscataway, NJ: IEEE) pp 3769–76
[52] Lücke J, Dai Z and Exarchakis G 2018 Truncated variational sampling for ‘black box’ optimization of generative models Int. Conf. on Latent Variable Analysis and Signal Separation (Springer) pp 467–78
[53] Exarchakis G, Henniges M, Eggert J and Lücke J 2012 Ternary sparse coding Int. Conf. on Latent Variable Analysis and Signal Separation (Springer) pp 204–12
[54] Jin C, Zhang Y, Balakrishnan S, Wainwright M J and Jordan M I 2016 Local maxima in the likelihood of Gaussian mixture models: structural results and algorithmic consequences Advances in Neural Information Processing Systems pp 4116–24
[55] Serrano-Gotarredona T and Linares-Barranco B 2015 Poker-dvs and mnist-dvs their history, how they were made, and other details Front. Neurosci. 9 481
[56] Orchard G, Jayawant A, Cohen G K and Thakor N 2015 Converting static image datasets to spiking neuromorphic datasets using saccades Front. Neurosci. 9 437
[57] Amir A et al 2017 A low power, fully event-based gesture recognition system Proc. IEEE Conf. on Computer Vision and Pattern Recognition pp 7243–52
[58] Eckart B, Kim K and Kautz J 2018 Hgmr: hierarchical Gaussian mixtures for adaptive 3D registration Proc. European Conf. on Computer Vision (ECCV) pp 705–21
[59] Evangelidis G D and Horaud R 2017 Joint alignment of multiple point sets with batch and incremental expectation-maximization IEEE Trans. Pattern Anal. Mach. Intell. 40 1397–410
[60] Lenz G, Chaney K, Shrestha S B, Oubari O, Picaud S and Zarrella G 2021 Tonic: event-based datasets and transformations Documentation available under https://tonic.readthedocs.io
[61] Dempster A P, Laird N M and Rubin D B 1977 Maximum likelihood from incomplete data via the em algorithm J. R. Stat. Soc. B 39 1–22
[62] Bishop C M 2006 Pattern Recognition and Machine Learning (Berlin: Springer)
[63] Neal R M and Hinton G E 1998 A view of the em algorithm that justifies incremental, sparse, and other variants Learning in Graphical Models (Berlin: Springer) pp 355–68
[64] Feldman D, Faulkner M and Krause A 2011 Scalable training of mixture models via coresets Advances in Neural Information Processing Systems pp 2142–50
[65] Shah R R, Shahik A D, Yu Y, Geng W, Zimmermann R and Wu G 2015 Eventbuilder: real-time multimedia event summarization by visualizing social media Proc. 23rd ACM Int. Conf. on Multimedia pp 185–8
[66] Shah R R, Yu Y, Verma A, Tang S, Shahik A D and Zimmermann R 2016 Leveraging multimodal information for event summarization and concept-level sentiment analysis Knowl.-Based Syst. 108 102–9
[67] Lucic M, Faulkner M, Krause A and Feldman D 2017 Training Gaussian mixture models at scale via coresets J. Mach. Learn. Res. 18 5885–909
[68] Fränti P and Virmajoki O 2006 Iterative shrinking method for clustering problems Pattern Recognit. 39 761–75
[69] Verbeek J J, Vlassis N and Kröse B 2003 Efficient greedy learning of Gaussian mixture models Neural Comput. 15 469–85
[70] von Luxor U 2010 Clustering stability: an overview Found. Trends 2 235–74
[71] Zhang H and Zhao D 2004 Spatial histogram features for face detection in color images Pacific-Rim Conf. on Multimedia (Springer) pp 377–84
[72] Gao S, Guo G, Huang H, Cheng X and Chen C L P 2020 An end-to-end broad learning system for event-based object classification IEEE Access 8 45974–84
[73] Sculley D 2010 Web-scale k-means clustering Proc. 19th Int. Conf. on World Wide Web (New York: ACM) pp 1177–8
[74] Orchard G, Meyer C, Etienne-Cummings R, Posch C, Thakor N and Benosman R 2015 Hfirst: a temporal approach to object recognition IEEE Trans. Pattern Anal. Mach. Intell. 37 2028–40
[75] He K, Zhang X, Ren S and Sun J 2016 Deep residual learning for image recognition Proc. IEEE Conf. on Computer Vision and Pattern Recognition pp 770–8