Poster Abstract: Data-Driven Windows to Accelerate Video Stream Content Extraction in Complex Event Processing

Piyush Yadav
Lero- Irish Software Research Centre
National University of Ireland
Galway, Ireland
piyush.yadav@lero.ie

Dibya Prakash Das
Indian Institute of Technology
Kharagpur, IIT Kharagpur
India
dibyadas@iitkgp.ac.in

Edward Curry
Lero- Irish Software Research Centre
National University of Ireland
Galway, Ireland
edward.curry@lero.ie

ABSTRACT

This work presents a data-driven adaptive windowing approach to accelerate video content extraction in DNN-based Complex Event Processing (CEP) systems. The CEP windows continuously monitor low-level content of incoming video frames and exploit interframe correlations to accelerate the overall DNN content extraction process. The two main contributions are: 1) technique to create micro-batches of similar frames within the window by measuring dissimilarities among them, and 2) optimal frame resolution within micro-batches under specified accuracy thresholds for fast model processing. The initial experimental results show that our adaptive micro-batching approach improves **3.75X** model throughput execution while maintaining application-level latency bounds under required accuracy constraints.

CCS Concepts **• Information systems→ Multimedia streaming • Applied computing → Event-driven architectures**

Keywords: Windows, Complex Event Processing, Video Processing, Deep Neural Network, High Throughput

ACM Reference format:
Piyush Yadav, Dibya Prakash Das and Edward Curry. 2019. Poster: Data-Driven Windows to Accelerate Video Stream Content Extraction in Complex Event Processing. In *Proceedings of Middleware Posters and Demos '19, Davis, CA, US December 8–13, 2019, 2 pages.*

1 INTRODUCTION

Complex Event Processing (CEP) systems mine patterns over data streams and provide fast reasoning with high throughput and low latency [2]. Windows are considered an essential primitive of CEP systems, which captures the finite subset (state) of an unbounded stream and apply event pattern rules over them. With the rise of the Internet of Multimedia Things (IoMT), visual sensors like smartphones and CCTV cameras are now generating a massive amount of video streams. Detecting event patterns in real-time from video streams is challenging [6][7].
Motivating Example: Table 1 shows two different CEP queries. In query 1 (Q1), a user is interested in the average price from the 'Stock' event stream in every 5 seconds time window. Performi-

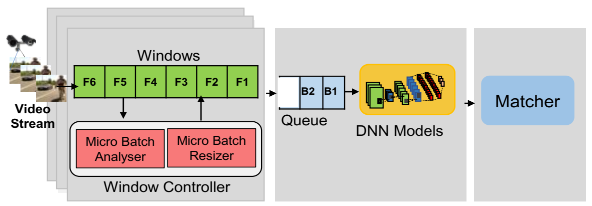

Figure 1. Window controller continuously monitors incoming frames over the window and forms optimal micro-batches in real-time. Micro-batching amortizes DNN execution cost, which is then passed for pattern matching.

-ng different aggregation techniques like 'AVERAGE', 'SUM', 'MIN' over windows is a well-understood problem in stream processing. Aggregations and other adaptive methods over windows are performed with the assumption that the data it is receiving has a structured format like key-value pairs (e.g. price values). In query 2 (Q2), a user is interested in the number of frames (Count) having 'Car' object over a video stream within the window of 5 seconds with an accuracy threshold (70% here). Querying video requires content extraction method like Deep Neural Networks (DNN), which are computationally expensive and have high inference time.

Table 1 CEP Queries Over Stock Events and Video Stream

Q1	`SELECT AVG(price) FROM Stock WHERE Stock = 'X' WITHIN TIMEWINDOW (5) WITH AVG_PRICE > $80`
Q2	`SELECT COUNT(frame) FROM Camera WHERE Frame = 'Car' WITHIN TIMEWINDOW (5) WITH Accuracy > 70%`

There is a need to bring adaptivity over windows which can infer video content and exploit DNN properties for fast model inferencing. Bifet et al. [1] proposed the concept of content-driven windows where window length changes as per change in the data distribution. These works consider the data stream having a fixed data model and have not focused on unstructured content like videos. Following the concept of content-driven windows [1], we exploit low-level video characteristics for achieving high system performance (Fig. 1). We propose: 1) an adaptive batching technique to identify optimal micro-batches of similar video frames within windows in real-time and 2) transforming micro-batches to optimal resolution under the specified matching threshold for fast model inferencing.

2 APPROACH

2.1 Selecting Input Parameters for DNN Model

Various optimization techniques like compression and specialization [5] have been proposed for faster execution of

ACM ISBN 978-1-4503-7009-7/19/12.
DOI: 10.1145/3366627.3368115

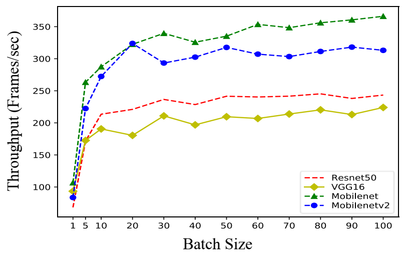

Figure 2. Effect of Batch Size on throughput for different DNN models

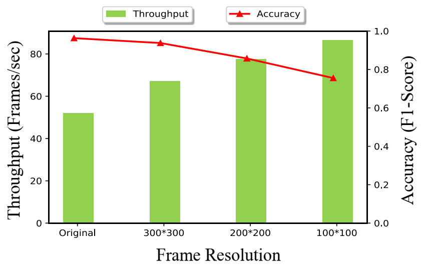

Figure 3. Effect of frame resolution on throughput and accuracy

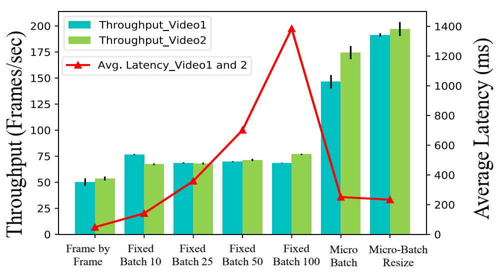

Figure 4. Model throughput and latency for different batching strategies

DNN models. Since video data is highly dynamic and changes frequently, loading different optimized model in memory incurs high runtime costs. We focus on two input-based transformation parameters which a DNN model can accept from the window.

- ***Batch Size:*** DNN model execution time is low if it receives the batch of input frames as a higher dimension input vector. This is because the kernel does not need to load the input data every time which stalls the memory, leading to high processor utilization. Fig. 2 shows that the batching of image frames improves the throughput performance of several DNN models.
- ***Frame Resolution:*** Reducing the image size decreases the input information, requiring fewer operations over the DNN model. This leads to a decrease in accuracy as the model is relying on less information to predict the output. Fig. 3 shows that decreasing frame resolution increases model throughput, but its overall prediction accuracy decreases.

2.2 Data-driven Windows for Video Streams

- ***Identifying Micro-batches:*** Color histogram technique is applied to identify the similarity between images. The frames are converted to a HSV space, and correlation distance between two histograms is used to calculate the similarity score. The algorithm treats the first frame of the batch as a reference frame and calculates the similarity score with respect to the reference frame. Thus, a window over a stream (S) is composition of several *unique* micro-batches (MB_i) as shown in equation 1:

$$\boldsymbol{WIN(S) = MB_1 \bigcup MB_2 \bigcup ... \bigcup MB_p \ , \forall MB_i \bigcap \forall MB_j = \emptyset} \qquad \textbf{(1)}$$

- ***Micro-batch Resizing:*** If a user is interested in getting information within a threshold accuracy (Q2-70%), then this information can be leveraged to resize the image proportionally such that the required accuracy is still maintained, but now with higher throughput. In Fig. 1 the reference frame of micro-batch is sent to the *Micro-Batch Resizer* where it tests frame prediction accuracy against a lightweight DNN model at a different resolution and sets a minimum resolution for the reference frame which satisfies the query accuracy constraints. Selecting different resolutions is based on the Image Pyramid [4] technique where images are rescaled to different levels. The *Micro-Batch Resizer* resizes the full micro-batch of frames as per the received reference frame resolution and sends them to DNN models for processing.

3 EXPERIMENTAL RESULTS

- **Implementation and Datasets:** The prototype is implemented in Python 3 and evaluated on a Linux system with a 3.1 GHz processor and a Nvidia Titan Xp GPU. We implemented a fine-tuned ResNet50 [3] model in Keras using transfer learning techniques. Two videos are used, namely the Jackson town square (video 1) [5] and a video clip from Pexels[1] website (video 2).
- **Throughput and Latency:** Fig. 4 shows the throughput and latency acquired by two videos in different settings. All the experiments have been repeated ten times to show the change in error using the black line on each bar. Video 1 and 2 achieve a throughput of 50.21 and 53.57 fps when processed frame by frame. To test the efficacy of the micro-batching approach, we compare with the processing of fixed batches of 10, 25, 50, and 100. Micro-batching achieves a throughput of 146.41 and 174.42 fps for video 1 and 2 respectively, while micro-batch-resize has a throughput of 191.40 and 197.04 fps. Micro-batching achieves 3.1X times and micro-batch-resize achieves 3.75X times more throughput than frame by frame processing. The average latency is the time taken by each frame to process from window to DNN models. The red line shows the average latency where frames take 48 *ms* in frame by frame processing while micro-batch-resize take 238 *ms* (batch latency). Although in our approach, the latency is high, as the system has processed the batch of frames, but it outperforms fixed batch latencies, increasing the overall system performance.

4 CONCLUSION AND FUTURE WORK

We presented a data-driven windowing technique to accelerate the DNN model inference time for video streams. The proposed approach accelerates 3.75X throughput with minimal latency overhead. Future work will create methods for efficient frame-based micro-batching for sliding windows.

Acknowledgements: This work was supported under the Science Foundation Ireland grant 13/RC/2094.

REFERENCES

[1] Bifet, A. and Gavaldà, R. 2007. Learning from Time-Changing Data with Adaptive Windowing *. SDM (2007).
[2] Cugola, G. and Margara, A. 2012. Processing flows of information. ACM Computing Surveys. 44, 3 (2012), 1–62.
[3] He, K., Zhang, X., Ren, S. and Sun, J. 2016. Deep Residual Learning for Image Recognition. IEEE CVPR (2016).
[4] Image Pyramids — OpenCV Documentation: https://bit.ly/2mjqkdp.
[5] Kang, D., Emmons, J., Abuzaid, F., Bailis, P. and Zaharia, M. 2017. NoScope: Optimizing Neural Network Queries over Video at Scale. VLDB (2017).
[6] Yadav, P. and Curry, E. 2019. VEKG: Video Event Knowledge Graph to Represent Video Streams for Complex Pattern Matching. IEEE Graph Computing (2019).
[7] Yadav, P. and Curry, E. 2019. VidCEP: Complex Event Processing Framework to Detect Spatiotemporal Patterns in Video Streams. IEEE BigData (2019).

[1] *https://www.pexels.com/videos/*