Algorithms for Canvas-based Attention Scheduling with Resizing

Yigong Hu[†], Ila Gokarn[‡], Shengzhong Liu[††], Archan Misra[‡], Tarek Abdelzaher[†]
[†]University of Illinois at Urbana-Champaign, [‡]Singapore Management University, [††]Shanghai Jiao Tong University,
yigongh2@illinois.edu, ingokarn.2019@phdcs.smu.edu.sg, liu-sz@cs.sjtu.edu.cn,
archanm@smu.edu.sg, zaher@illinois.edu

***Abstract*—Canvas-based attention scheduling was recently proposed to improve the efficiency of real-time machine perception systems. This framework introduces a notion of *focus locales*, referring to those areas where the attention of the inference system should "allocate its attention". Data from these locales (e.g., parts of the input video frames containing objects of interest) are packed together into a smaller *canvas frame* which is processed by the downstream machine learning algorithm. Compared with processing the entire input data frame, this practice saves resources while maintaining inference quality. Previous work was limited to a simplified solution where the focus locales are *quantized* to a small set of allowed sizes for the ease of packing into the canvas in a best-effort manner. In this paper, we remove this limiting constraint thus obviating quantization, and derive the first *spatiotemporal schedulability bound* for objects of *arbitrary sizes* in a canvas-based attention scheduling framework. We further allow object resizing and design a set of scheduling algorithms to adapt to varying workloads dynamically. Experiments on a representative AI-powered embedded platform with a real-world video dataset demonstrate the improvements in performance and inform the design and capacity planning of modern real-time machine perception pipelines.**

I. Introduction

A growing number of Internet of Things (IoT) applications require real-time machine perception involving the use of complex Deep Neural Networks (DNN) to process one or more sensory input streams on resource-constrained edge devices. These applications increasingly use general-purpose sensors, such as cameras or LiDARs, and adapt them for downstream tasks using some form of machine intelligence. For example, video data streams are increasingly used as a general-purpose sensing modality that is then analyzed by intelligent perception systems to serve a diverse range of applications. Such applications include obstacle localization in autonomous driving systems [1], suspicious activity detection in security/surveillance systems [2], visual odometry in vision-based navigation systems [3], occupancy detection in parking lots [4], and building defect identification in visual inspection systems [5].

In applications with time constraints (e.g., collision avoidance in autonomous vehicles or drones), these complex machine inference algorithms add a heavy burden to the computational capacity of the edge computing platform. The commonplace approach to overcoming this performance bottleneck is to provision the edge device with a GPU sufficient to process the entire input frame at the desired frame rate. However, with the observation that the application-relevant parts of the stream usually constitute only a small fraction of the input data, *attention scheduling* [6] was proposed to save computational resources by identifying and processing only the targeted *focus locales* in each frame. For example, when processing a video frame for object detection, only a small fraction of frame regions contain objects of interest. The work deemed processing such parts together with less relevant ones (e.g., the background) an instance of *priority inversion*. Not surprisingly, it was shown that ample computational resources can be saved by processing only the targetted focus locales in each frame.

Attention scheduling introduces an interesting dilemma. Accelerators, such as GPUs (that are used to run embedded system AI pipelines), are most efficient when running the same kernel on all cores. This entails using inputs of standardized fixed size (since, for example, a neural network that processes input data will have a different architecture and weights depending on the input dimensions). In traditional pipelines where the perception subsystem observes the entire frame, it is trivially true that the input size (i.e., frame size) is fixed. Thus, the same neural network architecture is executed on all cores. Attention scheduling breaks that assumption since the size of individual focus locales within a frame can be arbitrary. Only the focus locales with the same sizes are batched together for execution on the GPU. The desire to unify input size (to the perception subsystem), while allowing arbitrarily-sized focus locales, has led to the emergence of *canvas-based attention scheduling* [7].

In canvas-based scheduling, the focus locales are packed into fixed-size bins, called canvas frames, that can then be processed by the perception subsystem in an efficient manner. Computational savings result from that the canvas frames are smaller than the original full frames, leveraging the insight that only a subset of the input is worth computational attention at any point in time. Compared with the previous batching-based solutions, this approach is more efficient because of the improvement in parallelism on the GPU. Canvas-based scheduling is interesting from a real-time scheduling perspective because schedulability becomes a function of both temporal attributes of observed objects (such as deadlines by which an object needs to be inspected by the perception subsystem) as well as their spatial attributes (e.g., what

2642-7346/24/$31.00 ©2024 IEEE
DOI 10.1109/RTAS61025.2024.00035

fraction of the canvas frame an object occupies). Tension arises between the spatial and temporal dimensions. Namely, the optimal order in which objects should be considered for packing into a bin (i.e., into a canvas frame) to best utilize bin capacity is different from the optimal order in which objects should be processed to meet deadlines. Thus, spatiotemporal schedulability bounds could be derived for policies that make different design decisions towards reconciling this tension. An example of such a bound was proposed in [7] to relate the ability of the edge-based perception subsystem to keep up with the state of the environment in real time to the spatiotemporal properties of surrounding objects. However, this schedulability condition was obtained under significantly simplified assumptions that the focus locales are all squares and quantized to a limited number of size levels. Such assumptions result in wasted canvas areas and reduce efficiency. In addition, the schedulability condition was merely used to estimate the capacity requirement *offline*, while not used for adapting to the dynamic workload *online*.

This paper builds on our previous study [8]. We remove the limiting quantization assumptions and derive a new schedulability bound for canvas-based attention scheduling. The bound yields a sufficient spatiotemporal schedulability condition for a perception subsystem as a function of (i) the spatial properties of inputs that need to be inspected by the subsystem (the focus locales) and (ii) the deadlines by which the inspection must occur. We further introduce resizing as a useful tool to control the trade-off between the perception quality and resource consumption for inference tasks, and design algorithms to derive the best resizing decisions. Through experiments on a representative AI-powered embedded platform with a real-world video dataset, we demonstrate the improvements in canvas utilization and efficiency. This work provides useful insights to inform the design and capacity planning for future real-time machine inference pipelines.

The rest of this paper is organized as follows. Section II presents the problem formulation. Section III describes the proposed scheduling algorithm and derives its properties. Section IV discusses the implementation. The evaluation is presented in Section V. Section VI overviews related work. A brief discussion of limitations is covered in Section VII. The paper concludes with Section VIII that summarizes the key takeaways and outlines avenues for future extensions.

II. Problem Formulation

The scheduling problem addressed in this paper extends a recently proposed canvas-based scheduling framework [7]. This framework considers machine inference systems in applications such as intelligent video cameras, autonomous driving cars, robots, and drones, that require processing complex sensor inputs such as depth maps, thermal images, or camera frames in real-time, with an AI-based perception model running on a dedicated accelerator unit such as a GPU. A processing capacity *smaller* than what is necessary to process the original full input volume at the original frame rate is used. Only selected patches of the data input, enclosing objects of interest, are placed into a *canvas frame* in each frame to be processed. Their processing occurs at different intervals (multiples of the frame duration), considering the volatility of their states. For example, static objects in the field of view of a security camera need to be processed by the perception subsystem less frequently than moving ones. When not processed, new inference results (e.g., object type and location) are updated from predictions based on the previous ones.

We extend the aforementioned framework in two respects. First, we allow the selected patches to be of arbitrary size (up to a maximum limit) as opposed to having to adhere to one of a few quantized sizes. Quantization results in wasted canvas areas and reduces efficiency and flexibility. Second, we consider object resizing as a means to further improve efficiency. While the previous scheduling framework provides an explicit schedulability bound for deciding the best computation capacity *offline*, it is unable to dynamically adapt to changing computing loads *online* if it exceeds the predetermined capacity. Resizing has been shown to be an effective way to trade off between inference accuracy and computation requirements [9]. Introducing resizing to the canvas-based processing framework results in an interesting scheduling problem. We formulate the scheduling problem in this section as follows.

A. Task Model

In this paper, we follow the standard canvas-based scheduling model from earlier work [7], with the two extensions mentioned above. Below, we recap model assumptions.

Consider a sensor that produces a multidimensional data input, such as a camera or a depth sensor. It generates a series of data frames, F, at a fixed frame rate with a per-frame volume, V. (We use the word *volume* for generality, with the understanding that it, in fact, refers to *area* in the common case of two-dimensional video frames.) Let us denote the k-th frame by F_k. Let the interval between two successive frames (i.e., frame duration) be regarded as the time unit. Frame processing by the perception system occurs on some accelerator, with a capacity to process a total volume of at most $V_{GPU} < V$ in each time unit. We call this volume the *canvas frame*, denoted by C.

At time k, a set of objects of interest, $\mathcal{O}_k$, are located in the field of view of the sensor. Their rough locations are identified by algorithms such as background subtraction or optical flow. Let object $o_i \in \mathcal{O}_k$ occupy a volume v_i. Let F_{a_i} and F_{f_i} denote the frames when the object o_i enters and leaves the sensor's view, respectively. For simplicity, if an object exits the field of view and then re-appears again, it is treated as a new object. Each object must be processed periodically by the perception subsystem while it is within the field of view. This leads to a quasi-periodic task model, where a logical task is associated with the processing of an object. The first invocation of such a task for object o_i occurs when the object first enters the sensor field of view, at time a_i. We then follow an implicit deadline model, where the object is selected for

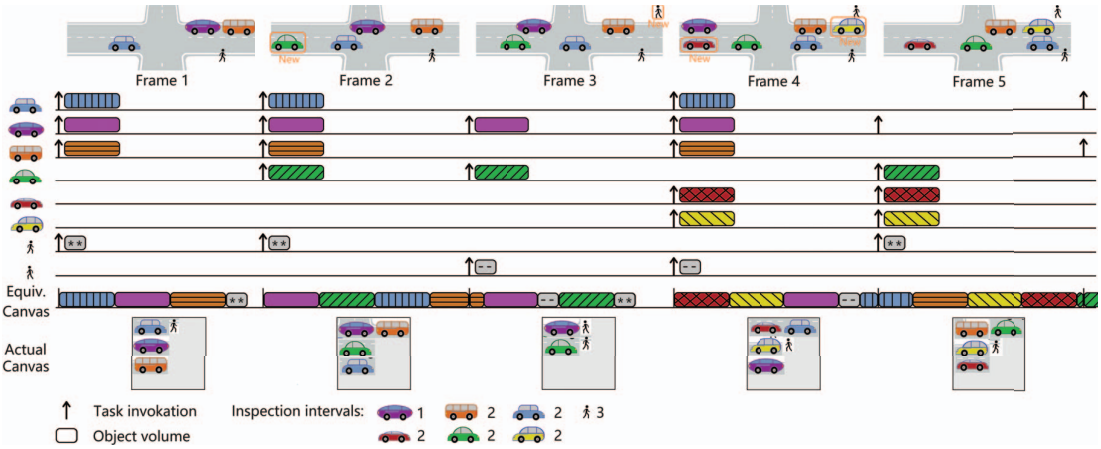

Fig. 1: An example of canvas-based attention scheduling.

inspection again after time D_i^k has elapsed from the start of the current period, where D_i^k is the inspection deadline (and period). Thus, if object o_i is selected for inspection in frame F_k, it must be processed by the inference model within time D_i^k. Otherwise, the inspection deadline is considered missed. The object must be packed in some canvas frame C_{k+m}, where $0 \leq m \leq D_{i_k}$. The next invocation happens at frame $F_{k+D_i^k}$, with a new inspection interval not necessarily the same as D_i^k, depending on the factors affecting the volatility of the object. For example, steady and static objects may have a larger D, and faster-moving objects may have a smaller D. Between invocations, the object locations are updated with predictions from a tracking algorithm by utilizing the redundancy between video frames.

B. Resizing and Packing

In each frame k, a subset of objects $\mathcal{O}_s \subseteq \mathcal{O}_k$ are selected for execution, each with a new size choice. Let V_s denote the set of all allowed new size choices, and $v_i' \in \mathcal{V}_s$ denote the new resize choice for each selected object $o_i \in \mathcal{O}_s$. The resize choice for o_i is $v_{min} \leq v_i' \leq v_i \leq v_{max}$, where v_{min} is the smallest allowed object size and v_{max} is the largest allowed object size. The objects are not allowed to be up-sized because increasing the size beyond the original size does not provide new information and hence does not improve inference quality. The selected objects are resized and placed into the canvas frame following a *packing* $\mathcal{P}$ that maps objects to canvas frames so that their union is contained in the canvas frame and no two objects overlap. Resize reduces object volume and assists packing but also results in accuracy loss, hence resizing decisions must be judiciously made so that the impact on inference quality is minimized.

C. Scheduling Problem

We define the scheduling problem in this paper as an optimization problem to derive online for each frame: a set of objects $\mathcal{O}_s \subseteq \mathcal{O}_k$ selected and their optimal resizing choices $\mathcal{V}_s$, and a corresponding packing $\mathcal{P}$, so that all deadlines are met and the inference quality is maximized. We aim to derive a bound for the *equivalent canvas* volume that can be used for schedulability analysis. The bound quantifies the least amount of volume that can be processed in each frame when there are enough objects in the frame. Then the equivalent canvas can be treated as a uniprocessor serving the object volumes. Schedulability can be determined by simply considering the scalar object volumes and their inspection deadlines. If a schedule exists on the equivalent canvas, then the tasks must be schedulable on the corresponding actual canvas frame.

An illustration of the scheduling problem with a simple example is shown in Figure 1. We consider a surveillance camera looking at a road intersection to detect objects in the field of view. In each frame, selected objects are placed into the actual canvas frame smaller than the camera frame for processing with the machine inference model. New objects entering the frame have to be processed within the same frame. After that, their processing frequency is determined by their speeds. In this example, the fast-moving purple car has to be processed every frame while the other cars have to be processed every two frames, and the pedestrians only have to be processed every three frames.

This example shows a feasible schedule both on the equivalent canvas and with packing results on the actual canvas frames. While the volume of one object can span multiple equivalent canvases, the real focal locale of the object has to be placed into one canvas in its entirety. For example, in frame 2, there are three existing cars, a pedestrian, and one new car. Both the new green car and the purple car are put into the equivalent canvas because they have to be processed within this frame. Then the remaining capacity of the equivalent canvas is used for processing the blue car, and a fraction of the brown car. While the volume of the brown car is split between equivalent canvases 2 and 3, it is actually placed into canvas 2 in its entirety. Note that the volume processed in this canvas frame is larger than that of the equivalent canvas. Similarly, both the volumes of the blue car in frame 4 and the green car in frame 5 span two equivalent canvases, but both cars are

placed into one of the real canvas frames. Both of the two real canvas frames hold a larger volume than that of the equivalent canvas.

III. Scheduling Algorithms

The scheduling problem involves deciding the sequence of selecting objects, choosing the best new sizes for them, and deriving a feasible packing of selected objects into the canvas frame. Let us consider a special sub-case where the deadline for all objects is 1 and resizing is not allowed (i.e., all objects have to be processed in every frame at their original sizes). Then the schedulability problem becomes whether all objects in each frame can be packed into the canvas. This problem of packing rectangles into a unit square is known to be strongly NP-Hard. Then we consider another special case where all objects are released at the same frame and have the same deadlines. Then the schedulability problem becomes whether these objects can be placed in a set of bins with the number equal to the deadline. The solution involves a bin packing problem to allocate objects to canvases, which is known to be NP-hard. Hence, the original scheduling problem we consider is NP-hard, as the sub-problems in both the spatial and temporal dimensions are NP-hard. In order to derive a solution that is feasible for real-time scheduling, we must apply reasonable constraints and approximations.

To ensure real-time guarantees, we limit ourselves to the case where all objects are selected in EDF order so that a solution can be derived in a reasonable amount of time. While EDF may not be optimal for the packing problem, fixing the sequence of object choices makes the analysis more tractable, and produces a sufficient condition that ensures no deadline miss. Furthermore, we later show that the bound of the equivalent canvas volume is the same for all algorithms, indicating that EDF has the same worst-case performance as any other algorithms. We then derive the equivalent canvas volume and compare two sets of algorithms with different packing strategies.

A. Packing of Quantized Objects

The first packing strategy we consider is object quantization used by Hu *et al.* [7]. Specifically, in that version, the aspect ratios of objects are restricted to 1:1, 1:2, and 2:1, and the sides are quantized to the side length of the canvas frame divided by powers of 2. As a result, any sequence of objects whose sum of volumes is smaller than V_{GPU} can be packed. We then derive the bound on equivalent canvas volume with this packing strategy.

1) Schedulability Analysis: Hu *et al.* proved when objects are packed in EDF order, the set of objects in the field of view at time k meets inspection deadlines if the instantaneous utilization ratio is no larger than $1 - v_{max}/V_{GPU}$. This translates to an equivalent canvas volume bound of:

$$V_{equiv.} = V_{GPU} - v_{max}$$

With this, our scheduling problem can be simplified into finding the optimal resizing choices $\mathcal{V}$ so that the tasks can be scheduled with EDF on the equivalent canvas, and the accuracy loss is minimized.

Formally,

$$\max_{\mathcal{V}} \sum_{o_i \in \mathcal{O}(k)} E(o_i, v_i')$$

$$\textbf{s.t.} \sum_{o_i \in \mathcal{O}(k)} \frac{v_i'}{D_i^{k_i}} \leq V_{GPU} - v_{max}$$

where $E(o_i, v_i')$ is a function reflecting the expected accuracy (or quality) of the inference algorithm on object o_i when it is reduced to the new volume v_i'. This function is obtained from offline profiling. Intuitively, larger-sized objects can withstand more aggressive resizing before causing the accuracy of downstream processing to diminish, whereas smaller objects are more sensitive to resizing.

2) Online Scheduling Algorithm: The choices for new sizes are quantized to a limited set: $v_i' \in \{V_{GPU}/2^n, n = 1, 2, 3, ...\}$. The optimization problem becomes a classic Multiple-Choice Knapsack Problem (MCKP), where the objective is to choose from a set of multiple-choice items to maximize total reward (accuracy expectation), while the total weight (object volumes weighted by deadlines) does not exceed the knapsack capacity (equivalent canvas volume). MCKP is known to be NP-hard, making an exact solution impractical to obtain in real-time. Instead, we apply a greedy heuristic to arrive at an approximate solution. It starts with all objects not selected, with $v_i' = 0$ for each object. Then it incrementally increases the sizes of objects with the highest incremental accuracy expectation over incremental volume. Let $v_i'^{-}$ denote the object volume one level smaller than v_i', and $v_i'^{+}$ denote the object volume one level larger than v_i'. We define:

$$A_{i,v_i'} = \begin{cases} E(o_i, v_i') - E(i, v_i'^{-}) & \text{if } v_i' > v_{min} \\ E(o_i, v_{min}) & \text{if } v_i' = v_{min} \end{cases}$$

and

$$B_{v_i'} = \begin{cases} (v_i' - v_i'^{-})/D_i^{k_i} & \text{if } v_i' > v_{min} \\ v_i'/D_i^{k_i} & \text{if } v_i' = v_{min} \end{cases}$$

where $A_{i,v_i'}$ is the incremental accuracy expectation and $B_{v_i'}$ is the incremental weighted volume. The incremental efficiency is then: $G_{i,v_i'} = A_{i,v_i'}/B_{v_i'}$. We calculate G for all objects and sizes, and sort them in decreasing order, together with the indices i and v_i'. Intuitively, the smaller the object is, the more easily it loses information when downsized. Supported by the empirical profiling results, we have that G monotonically decreases with v_i, satisfying the Karush-kuhn-tucker conditions [10] and ensuring that a smaller volume will always be chosen before a larger one. We iteratively select volume increments for objects and update the new sizes for corresponding tasks, until the total weighted volume reaches the equivalent canvas volume. With these resizing decisions, the tasks are schedulable on the canvas.

After we find the optimal resizing choices, the objects

are chosen according to the EDF order just before their total resized volume exceeds the equivalent canvas volume: $V_{GPU} - v_{max} \leq \sum_{o_i \in \mathcal{O}_s} v'_i \leq V_{GPU}$, where $\mathcal{O}_s$ is the set of selected objects. These objects are going to be packed into the canvas for execution in the current frame. This ensures that at least $V_{equiv.} = V_{GPU} - v_{max}$ volume is processed in each canvas frame. The reason to not pack the canvas to its maximum is that, because any sequence of objects whose sum of volumes is smaller than that V_{GPU} can be packed into the canvas, we have the opportunity to partially reverse the downsizing of some objects when the canvas is not full. The sizes of the selected objects are incremented in the same manner as before until the cumulative weighted volume exceeds the bound.

The scheduling algorithm is detailed in Algorithm 1. Lines 5-12 calculate the best resizing choices so that the total weighted volume is below the equivalent canvas capacity; Lines 14-22 pick the resized objects in EDF order so that they fit in the equivalent canvas; Lines 23-34 use the remaining available canvas volume to partially reverse the downsizing of some objects. The algorithm derives resizing choices $\mathcal{V}_s$ and packing $\mathcal{P}$.

B. Packing of Unquantized Objects

Because of the restriction on aspect ratios and quantization, the quantized volume is generally larger than the focus locale of an object, which results in wasted volume. Thus, next, we derive a schedulability result for the case where quantization is not used and objects maintain their original aspect ratios.

1) Schedulability Analysis: When we remove the constraints of quantization and aspect ratios, objects with total volumes equal to the canvas volume will most likely not fit into the canvas. Previous work [11] proved a tight lower bound of $\frac{1}{2}$ on packing rectangles with side lengths smaller than 1 into a unit square when the rotation of $90°$ is allowed and provided a simple packing heuristic that supports the bound. We prove that when packing the objects according to EDF order, this bound results in an equivalent canvas volume of $\frac{1}{2}V_{GPU} - v_{max}$.

Theorem 1: *The bound on the equivalent canvas volume for unquantized rectangular objects is at most* $\frac{1}{2}V_{GPU} - v_{max}$.

Proof: Let us assume that the bound on the equivalent canvas volume for unquantized rectangular objects can be larger than $\frac{1}{2}V_{GPU} - v_{max}$. Consider a set of inspection tasks, all with an inspection period of $D_i = 1$, and the sum of object volumes is $\sum_i v_i = 1/2 - v_{max} + \epsilon$, where ϵ is infinitesimally larger than 0. Another object o_l has a size equal to v_{max}, and an inspection period of $D >> 1$. If we include object o_l into any of the canvas frames, then the total area of objects in that canvas frame is: $\sum_i v_i + v_{max} = 1/2 + \epsilon$, which is greater than $1/2$. According to [11], when the total area of objects is larger than $1/2$, successful packing is not guaranteed, so object o_l may miss its inspection deadline. The total weighted volume of objects is: $\sum_i v_i/1 + v_{max}/D = 1/2 - v_{max} + \epsilon + v_{max}/D$, which is arbitrarily close to $1/2 - v_{max}$ when D is sufficiently large and ϵ is sufficiently small. This contradicts our assumption. □

Algorithm 1: Quantized Objects Scheduling

input : Objects $\mathcal{O}$, inspection intervals D^k, canvas capacity V_{GPU}, max object volume v_{max}, accuracy-resizing profile E
output: Selected objects Objects $\mathcal{O}_s$, Resize decision $\mathcal{V}_s$, packing $\mathcal{P}$

```
1  Sort O in EDF order
2  Initialize V : {v'_i = 0, ∀o_i ∈ O}, U = 0
3  Calculate G = A/B for all objects in O
4  Sort and index G such that G_1 > ... > G_N
5  for i = 1, 2, ..., N do
6      U = U + B_i
7      if U ≤ V_GPU − v_max then
8          idx = i break
9      else
10         v'_i = v'^+_i
11     end
12 end
13 a = 0, V_s = ∅, O_s = ∅
14 for o_i in O do
15     if a + v'_i < V_GPU − v_max then
16         a = a + v'_i
17         O_s = O_s + o_i
18         V_s = V_s + v'_i
19     else
20         break
21     end
22 end
23 for i = idx, ..., N do
24     if o_i in O_s then
25         if a + (v'^+_i − v'_i) > V_GPU then
26             break
27         else
28             v'_i = v'^+_i
29             a = a + (v'^+_i − v'_i)
30         end
31     else
32         Pass
33     end
34 end
35 Generate packing P from O_s and V_s
36 return V_s, P
```

Notice that in the above proof, we do not make any assumptions about the scheduling algorithm, hence the result applies to any scheduling algorithms including EDF. Next, we show that the above bound is tight. Similar to [7], We define the *EDF busy period* as a set of successive busy frames starting at the frame k when some task associated with object o_i starts and ends at frame $k + D_i^k$ when this object eventually misses its inspection deadline.

Lemma 1: *Within an EDF busy period, each canvas includes objects with a total volume of at least* $\frac{1}{2}V_{GPU} - v_{max}$.

Proof: Consider a canvas frame C_k in the EDF busy period. Since all objects are no larger than v_{max}, assume the total volume in the canvas is less than or equal to $\frac{1}{2}V_{GPU} - v_{max}$. After including o_i in this canvas, the canvas is less than or equal to $1/2$ utilized. According to [11], a feasible packing

exists for these objects so object o_i can be put into this canvas frame and processed before its deadline, which contradicts our assumption. □

Theorem 2: *A set of objects $\mathcal{O}$ in the input frame k meet all inspection deadlines if:*

$$\sum_{o_i \in \mathcal{O}(k)} \frac{v_i}{D_i^{k_i}} \leq \frac{1}{2} V_{GPU} - v_{max}$$

Proof: Imagine a hypothetical uniprocessor of speed $(\frac{1}{2}V_{GPU} - v_{max})$ cycles per time unit and a set of aperiodic tasks t_i that each has a deadline of $D_i^{k_i}$ time units, and needs v_i time units to finish. According to Lemma 1, a canvas frame will process at least $(\frac{1}{2}V_{GPU} - v_{max})$ volume of data. When comparing objects processed with the canvas and tasks processed on the hypothetical uniprocessor, tasks that finish at the end of a time unit correspond to objects that have been processed by the perception system. So, if the hypothetical uniprocessor is schedulable, so is the canvas-based perception system.

The schedulability condition for aperiodic tasks on a uniprocessor with EDF is known to be [12]:

$$\sum_i \frac{C_i}{D_i} \leq 1$$

Where C_i is the execution time of task t_i on the hypothetical uniprocessor, calculated by the number of cycles it needs divided by the number of cycles the uniprocessor executes each time unit, or $C_i = v_i/(\frac{1}{2}V_{GPU} - v_{max})$. Substituting C_i in the EDF schedulability condition and rearranging, we get the schedulability condition for the hypothetical uniprocessor with the aperiodic tasks:

$$\sum_{o_i \in \mathcal{O}(k)} \frac{v_i}{D_i^{k_i}} \leq \frac{1}{2} V_{GPU} - v_{max}$$

With the argument made above, this condition also applies to the canvas-based perception system. From Theorem 1, we can conclude that this bound on the equivalent canvas volume is tight. □

2) Online Scheduling Algorithm: Since the quantization restriction is removed, the volume of the resized object can be any number between v_{min} and v_{max}. In order to support more flexible resizing options, instead of allowing a finite number of resizing choices, we obtain an approximated function $R(d)$ of the accuracy expectation with regard to the object's larger side length d through offline profiling. Let the aspect ratio of object o_i be $b_i > 1$, obtained by dividing the longer side length by the shorter side length, then the volume $v = d^2/b$. Let v_i' denote the new volume of object o_i after resizing, and let $u_i = v_i'/D_i^{k_i}$ denote the weighted volume of object o_i. The relation between the accuracy expectation E and the object weighted volume u is: $E(u_i) = R(\sqrt{b_i * D_i^{k_i} * u_i})$. The optimization problem is:

$$\max_{u_i'} \sum_{o_i \in \mathcal{O}(k)} E_i(u_i)$$

$$\text{s.t.} \sum_{o_i \in \mathcal{O}(k)} u_i \leq \frac{1}{2} V_{GPU} - v_{max}$$

As the offline accuracy profiling suggests, the closer the new size is to the original size, the slower the accuracy increases. For this reason, we assume that $R(d_i)$ is concave and monotonically increasing, satisfying the Karush-kuhn-tucker conditions [10]. As a result, $E(u_i) = R(\sqrt{b_i * D_i^{k_i} * u_i})$ is also concave and monotonically increases with u_i. We obtain this function by polynomial fitting to the profiling results. We then show that the optimal solution is when function $E_i(u_i)$ has the same slope at all u_i.

Lemma 2: *At the optimal solution, $\sum_i u_i = \frac{1}{2}V_{GPU} - v_{max}$.*

Proof: Since $E(u_i)$ monotonically increases with u_i, this is trivially true. □

Theorem 3: *At the optimal solution, the slopes of all functions $E_i(u_i)$ are the same.*

Proof: Let us denote the slope of $E_i(u_i)$ as $m_i(u_i)$. Assume that at the optimal solution, at least one function $E_i(u_i)$ has a different slope. Consider this function and any other function with a different slope. Without loss of generality, let $E_1(u)$ denote the one having a larger slope at optimal value u_1 and $E_2(u)$ denote one having a smaller slope at optimal value u_2. Let $W = E_1(u_1) + E_2(u_2)$ denote the cumulative accuracy expectation. Since functions $E(u)$ are monotonically increasing and concave, their slopes $m(u)$ are monotonically decreasing, then there exists u_1' and u_2', where:

$$\begin{cases} m_1(u_1') = m_2(u_2') = m_0 \\ m_1(u_1) > m_0 > m_2(u_2) \\ u_1 < u_1' \\ u_2 > u_2' \end{cases}$$

The new cumulative accuracy expectation is:

$$W' = E_1(u_1') + E_2(u_2')$$

then:

$$\begin{aligned} W' - W &= E_1(u_1') - E_1(u_1) - (E_2(u_2) - E_2(u_2')) \\ &> m_0 * (u_1' - u_1) - m_0 * (u_2 - u_2') \\ &= m_0 * ((u_1' + u_2') - (u_1 + u_2)) \end{aligned}$$

By Lemma 2, $u_1' + u_2' = u_1 + u_2$, so $W' - W > 0$, indicating that $W' > W$, which contradicts with the assumption that W is optimal. Hence, the slopes of functions $E_i(u_i)$ for all objects are the same at the optimal solution. □

For each o_i we can derive a function for u_i when the slope equals m: $u_i' = S_i(m)$. To limit u_i' in the valid range, it is clipped by u_{min} and u_i. The optimization problem is then to find the m so that:

$$\sum_{o_i \in \mathcal{O}(k)} S_i(m) = \frac{1}{2} V_{GPU} - v_{max}$$

As there is no closed-form solution for this equation, we find the optimal new volumes by performing a binary search on m.

Algorithm 2: Rectangular Objects Scheduling

input : Objects $\mathcal{O}$, inspection period D^k, canvas capacity V_{GPU}, max object volume v_{max}, volume-slope function S_i
output: Selected objects Objects $\mathcal{O}_s$, Resize decision $\mathcal{V}_s$, packing $\mathcal{P}$

1 Sort $\mathcal{O}$ in EDF order
2 Calculate the minimum slope m_0 and maximum slope m_1
3 Binary search in $[m_0, m_1]$ for the smallest m such that: $\sum_{o_i \in \mathcal{O}} S_i(m) > \frac{1}{2} V_{GPU} - v_{max}$
4 Calculate $\mathcal{V}$: $\{v_i' = S_i(m), o_i \in \mathcal{O}\}$
5 $a = 0$, $\mathcal{O}_s = \emptyset$
6 **for** o_i *in* $\mathcal{O}$ **do**
7 **if** $a + v_i' < V_{GPU}$ **then**
8 $a = a + v_i'$
9 $\mathcal{O}_s = \mathcal{O}_s + o_i$
10 **else**
11 break
12 **end**
13 **end**
14 Binary search in in $[m, m_1]$ for the smallest m such that: $\mathcal{O}_s$ rezised to $\mathcal{V}$: $\{v_i' = S_i(m), o_i \in \mathcal{O}\}$ can be packed
15 $\mathcal{V}_s = \mathcal{V}$
16 Generate packing P from $\mathcal{O}_s$ and $\mathcal{V}_s$
17 **return**$\mathcal{V}_s$, $\mathcal{P}$

After deriving the best new volumes to satisfy the equivalent canvas volume, the resized objects are selected according to EDF order until their total volume just exceeds the bound. Similar to before, the downsizing of the selected objects can be partially restored by utilizing the unused canvas area. [11] shows that any sequence of rectangles with a total area smaller than $1/2$ can be packed into a square of size 1. However, when the total area is larger than $\frac{1}{2}$, the existence of a valid packing is unclear. For the selected objects, We find the optimal m with binary search to utilize the remaining canvas volume as much as possible, taking advantage of the efficient packing scheme in [11] to decide if a feasible packing exists. The scheduling algorithm is detailed in Algorithm 2. It first derives the best resizing choices. Then lines 6-13 select resized objects in EDF order so that their cumulative weighted volume is just smaller than the equivalent canvas capacity. Then the algorithm partially reverses the downsizing with binary search and generates the final resizing choices $\mathcal{V}_s$ and packing $\mathcal{P}$.

IV. Implementation

In this section, we describe the implementation details of the canvas-based attention scheduling framework. We consider a video surveillance application that requires the detection of objects of interest in the camera frames.

Taking advantage of the statically mounted cameras, a background subtraction-based algorithm is used to locate new and foreground objects. Starting with a one-time initial full-frame inspection of all objects, each subsequent frame is first passed through background subtraction to locate differences from the previous frame. The system thus builds a background model that needs to be updated only where differences are indicated. These differences typically identify foreground moving objects. The algorithm remembers the last position of each such object and has a notion of object permanence: if an object does not move between two frames (i.e., it is not detectable by background subtraction), it is assumed to be not moving. Whenever motion (i.e., change) is observed in a place not previously occupied by an object, a new object is assumed to have entered the field of view. If motion is observed at a position that is consistent with a previously observed (trajectory of an existing) object, the location of that object is updated accordingly. Finally, if a moving object stops, even though the stopped object becomes "invisible" to the background subtraction algorithm (that only identifies motion/change), thanks to the assumption of object permanence, the system assumes that the object remains at its last recorded location and such location is kept track of for future inspection.

The selection of deadlines, D_i^k, is another implementation decision. In this paper, the inspection deadline is set inversely proportional to object speed. The intuition is that items with a more rapidly changing state need to be tracked more closely. Specifically, the speed of an object is calculated according to optical flow and the value is mapped to a finite set of inspection interval values ranging from 1 to 5, corresponding to the object being included for inference every frame to at least once every 5 frames. Other heuristics are possible but left to future work. For example, one may opt to inspect less predictably-moving objects more often. When an object is not selected for inference, its new location is approximated with an optical flow-based tracking algorithm [13].

V. Evaluation

In this section, we evaluate the different scheduling policies with a realistic surveillance camera dataset on an AI-powered embedded platform.

A. Experiment Setup

1) Hardware Platform: We use the NVIDIA Jetson AGX Xavier SoC for our experiments. It is a representative AI-powered embedded platform equipped with an 8-core Carmel Arm v8.2 64-bit CPU, a 512-core Volta GPU, and 32 GB memory shared by both the CPU and the GPU. The Jetson AGX Xavier consumes 30 Watts at the highest performance mode and can deliver over 30 TOPs for deep learning applications. We set the power mode to "MAXN" and configure the GPU to run at a constant clock frequency for a more stable performance.

2) Dataset: We use the VIRAT Video Dataset [14] for all of our experiments. This dataset was originally collected for video surveillance applications. It consists of video feeds of natural and realistic scenes captured by surveillance cameras, covering various lighting and weather conditions. This dataset is primarily designed for activity recognition so it only provides labels for objects involved in its target activities, and only the humans and objects involved in the actions are labeled. We generate "pseudo ground truth" labels for all objects of relevance to our application. We use a pre-trained

"Xlarge" YOLOv5 [1] model to label the objects in each frame. The "Xlarge" model is the largest of all Yolo variants and thus has the best performance (close to human accuracy). The accuracy calculated based on these pseudo-labels is then used as a metric to evaluate the quality of inference of our tested algorithms. We restrict our attention to three object categories: *person*, *bicycle*, and *vehicle*. Motorcycles and bicycles are merged into class *bicycle*. Trucks, cars, and buses are merged into class *vehicle*.

3) Load Manipulation: To evaluate the performance of the algorithms under different computation loads, we need to manipulate the number of objects in the frames. We achieve this by stitching frames from various numbers of video sources together to control the number of objects in the video feed. When frames from n video feeds are combined together, each new frame is n times larger and the overall utilization ratio is roughly increased to n times.

4) Perception Model and Canvas: We use the YOLOv5 model as the inference model for object detection. Specifically, we use the model with the "large" configuration, with both the depth and the width multipliers set to 1. The YOLOv5 model includes a convolutional network that runs on the GPU which dominates the execution time, and a non-max suppression (NMS) process that runs on the CPU. The model is trained on the COCO [15] dataset, and the precision is set to FP16 for inference. The model is configured to only produce detection on the objects of our interest. We profile the inference latency with different input sizes and batch sizes in advance on the hardware platform. In order to maintain a reasonably high detection frequency for surveillance camera applications while considering the computing capacity of the hardware, we select the frame rate to be $20Hz$, corresponding to a canvas size of 512×512 according to the profiling results. The max side length of objects is set to $d_{max} = 256$, corresponding to $V_{max}/V_{GPU} = 1/4$. Larger objects are simply downsized to meet this size constraint. The smallest size is set to $d_{min} = 32$.

5) Size and Accuracy Profile: We profile the detection accuracy expectations at different object sizes to facilitate the resizing decisions. Specifically, we define the object size by the length of their larger side d and vary it from 32 to 256 with an interval of 32. For each d, we iterate through all objects in the COCO dataset, select all the larger objects, and downsize them to this size. Then the resized objects are processed by the perception model and the average accuracy is calculated. For packing with rectangles with unquantized sizes, we obtain the accuracy profile function $R(d)$ by fitting a polynomial function and ensuring that it is concave and non-decreasing. In order to prevent the algorithms from selecting a size outside the allowed size range, we define the accuracy expectation to 0 when $d < d_{min}$ or $d > d_{max}$.

B. Evaluation Metrics

We consider the detection accuracy as a metric for the overall perception performance of the framework. We use mean average precision (mAP) as an end-to-end metric to simultaneously capture both the detection and classification performance. It is calculated by comparing the detection results with the ground truth labels and finding matches based on the intersection of the union (IoU) metric between bounding boxes [16]. A detection is only considered correct when the IoU between it and the ground truth bounding box exceeds a predefined threshold (we use 0.5 in this paper) and the object class matches. To evaluate the ability of algorithms to utilize the canvas area, we consider the average canvas utilization ratio. Canvas utilization is defined as the fraction of canvas volume occupied by packed input segments. Because of the quantization, these areas can be larger than the objects of interest (focus locales) enclosed in them. We also calculate the real canvas utilization as the fraction of the canvas area occupied by the objects of interest from the input frame. With these considerations, we define the evaluation metrics as follows:

- **Mean Average Precision (mAP)**: The average precision of all object classes. It is an indicator of the overall inference performance of the system.
- **Average Precision (AP)**: The average precision for each object class. It simultaneously captures the error in object detection and classification.
- **Average Canvas Utilization**: The average canvas utilization ratio. It evaluates how much of the canvas space is used by the algorithm.
- **Average Real Canvas Utilization**: The average area ratio of areas of interest in the canvas. It evaluates how much of the canvas is used towards the essential part of the input frame.

C. Experiment Results

1) Compared Algorithms: We compare in total four scheduling algorithms with two different packing strategies:

*a) **Quantized no resize (baseline)**:* The baseline scheduling algorithm proposed in [7]. It quantizes the areas of interest and places them into the canvas for processing following EDF in a best-effort manner.

*b) **Quantized resize**:* The scheduling algorithm detailed in section III-A. It quantizes the areas of interest, calculates the best resizing choices under the utilization bound, and then packs the resized objects into the canvas.

*c) **Rectangle boost**:* The scheduling algorithm detailed in section III-B. It calculates the best resizing choices to meet the deadlines and then packs the resized objects into the canvas.

*d) **Rectangle no boost**:* Same as *Rectangle boost* except that the sizes of the objects are not boosted back to utilize the remaining canvas volume.

The overall performance of the system, characterized by the mAP, is shown in Figure 2. As the load increases, more objects have to be processed, hence the performance of all algorithms drops. However, at all computation loads, the algorithm *Rectangle boost* consistently performs better than any other algorithm. *Quantized resize* performs better than *Quantized no resize* at all loads except when the number of

[1] https://github.com/ultralytics/yolov5

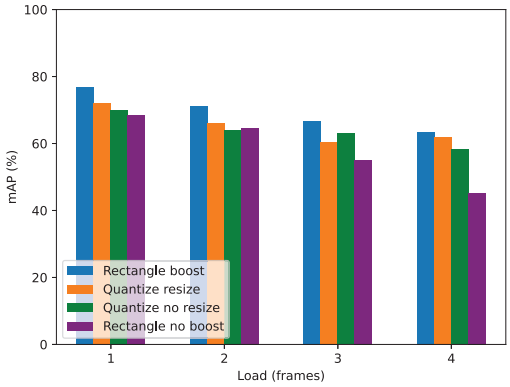

Fig. 2: The mean average precision comparison.

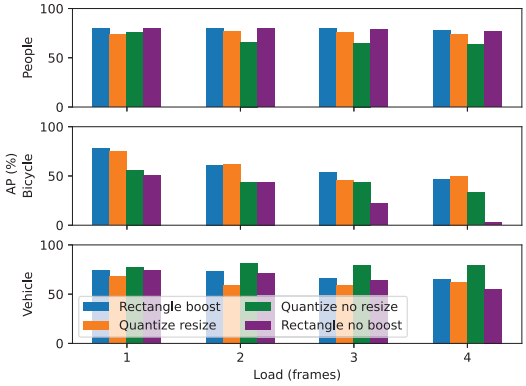

Fig. 3: Average precision of different object classes.

frames is 3, and maintains relatively high performance when the system load varies. This demonstrates the effectiveness of the resizing algorithm to selectively trade the detection accuracy of larger objects for more space to accommodate other objects. When system load increases, more objects in *Quantized no resize* miss their deadlines but the impact on detection accuracy degrades slowly, due to the utilization of temporal redundancy between frames. *Rectangle no boost* performs the worst, because of the pessimistic 25% equivalent canvas volume bound. Although the algorithm ensures no deadline is missed by meeting the bound, the aggressive downsizing results in too much information loss and affects the inference performance.

In order to obtain more insight into the impact of size manipulation, we compare the per-class average precision at different system loads, as shown in Figure 3. We observe that the algorithms with resizing maintain better accuracy for class *people* when the system load increases. People in the camera frames are usually small in size so downsizing larger objects will make enough room to include them in the canvas, maintaining the required inference frequency. *Quantized no resize*, on the other hand, processes the objects in a best-effort manner and randomly misses deadlines of objects as the load exceeds the utilization bound, essentially decreasing the inference frequency, resulting in more accuracy loss. For the object class *vehicle*, the average precision with *Quantized no resize* decreases slower than that of the other three algorithms. This is because there are a considerable number of parked vehicles in the camera frames. When the processing of these objects is skipped, the new location predictions are very accurate. With resizing, however, the detection accuracy of these objects decreases, and chances are that they will be removed from the memory because of a failed detection. Because of the nature of background subtraction, these objects are part of the background and will not be detected again. The accuracy for class *bicycle* of *Rectangle boost* drops much more sharply compared with class *vehicle*. This is because we use the same accuracy-resize function for all classes, while class *bicycle* is more susceptible to accuracy loss when downsized compared with class *vehicle*. This indicates the potential benefit of per-class treatment when making downsizing decisions.

To evaluate the ability of different algorithms to utilize the canvas, we calculate the average canvas utilization rate, as shown in Figure 4. As the load increases, the canvas utilization rate of both *Rectangle boost* and *Quantized resize* increases, indicating that they are able to utilize the unused canvas area and partially reverse the downsizing of some objects. The utilization rate of *Quantized no resize* is lower compared with *Quantized resize*, because it packs the canvas in a best-effort manner and lacks the size boost mechanism to better utilize the canvas. The utilization rate of *Rectangle no boost* stays close to 25% because it downsizes objects to meet the bound and does not restore the sizes of the objects. We further compare the real canvas utilization rate, calculated as the portion of the canvas volume that is occupied by the actual unquantized object, as shown in Figure 5. The real canvas utilization rates for the two algorithms with quantized objects are lower than the average canvas utilization rate because of the wasted area caused by quantization. In comparison, although the average utilization of *Rectangle boost* is lower, when considering the quantization area waste, it is able to utilize more canvas area for the use of the focus locales in the input frame, contributing to its better inference performance.

2) Scheduling Overhead: We report the cumulative distribution function (CDF) of the latency overhead of the scheduling algorithms, as shown in Figure 6. It reflects the latency induced by the scheduling algorithm under different conditions (number of objects, their sizes and speeds) encountered in the tested dataset. Due to the iterative binary search process involved, both *Rectangle boost* and *Rectangle no boost* require a longer time to finish compared with *Quantized resize* and *Quantized no resize*. Comparing *Rectangle no boost* and *Rectangle boost* we can conclude that boosting the downsized objects back towards their original sizes induces acceptable latency overhead, but contributes to a considerable amount of performance improvement, indicating the necessity of this

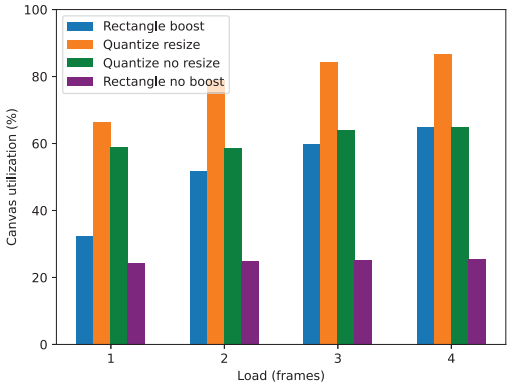

Fig. 4: The average canvas utilization rate.

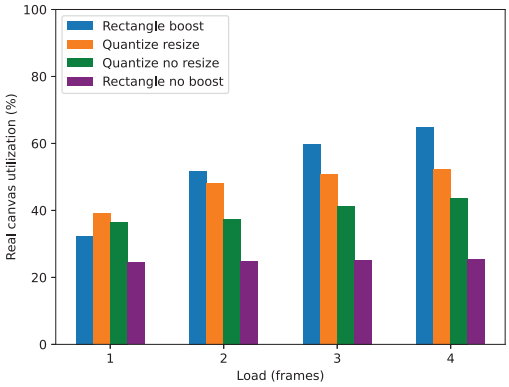

Fig. 5: The average real canvas utilization rate.

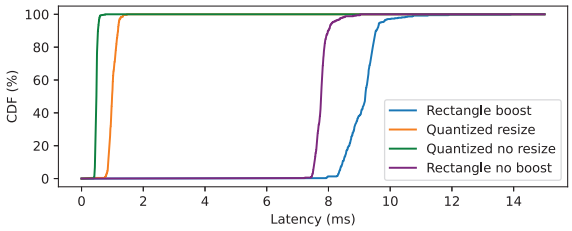

Fig. 6: Scheduling algorithm latency overhead.

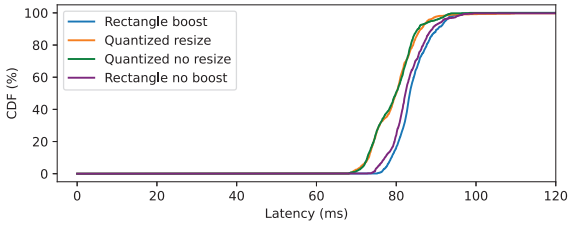

Fig. 7: End-to-end inference latency.

step. The CDF of the end-to-end inference latency is shown in Figure 7. Since all the scheduling, preprocessing, and postprocessing steps are implemented as separated threads running on different CPU cores, they pose no overhead on the inference model which is GPU dominant, and thus do not affect the throughput.

3) Deadline Misses: We also experimentally ascertained that no deadlines were missed as long as the derived equivalent canvas volume bound is met at all times, and the algorithms with resizing do not miss any deadlines. On the other hand, *Quantized no resize* misses deadlines. It packs objects in a best-effort manner (without resizing) until running out of canvas volume, so when the workload is high, not all objects can be packed. As a result, the objects that fail to be packed in time will miss their deadlines. While no figure is needed to demonstrate this result, we mention it here for completeness.

VI. Related Work

Recently, an increasing number of modern Cyber-Physical Systems (CPS) applications, such as autonomous driving, start to incorporate the execution of complex deep neural networks into mission-critical pipelines [17]. These systems often require predictable and efficient performance on resource-constrained hardware platforms.

While the current mainstream deep learning algorithms have been successful in terms of accuracy, they are not designed with explicit time constraints and constrained computation load required by Cyber-Physical Systems. To mitigate this issue, attention has been paid to optimizing deep learning frameworks with special consideration for real-time applications [18]–[24]. On the other hand, the current general-purpose hardware accelerators such as GPUs are not primarily designed for predictable timeliness, which reduces confidence in them for mission-critical real-time applications. To better understand the timing characteristic of the hardware platforms, numerous profiling efforts have been made [25]–[31]. The response time of combinations of complex AI algorithms and heterogeneous hardware was also studied: Voronov *et al.* [32] proposed techniques for response-time analysis of processing graphs running on complex heterogeneous hardware and was able to reduce the response-time bounds. The other work aimed at introducing the notion of imprecise computation to deep learning models and enabling dynamic execution time and quality trade-off [33]–[39].

Attention Scheduling [6] was proposed to tackle the *priority-inversion* problem commonly found in early perception pipelines. Instead of processing the incoming data frame in a first-come-first-served manner, attention scheduling prioritizes different regions of the input frame according to their importance. Given that usually only a small fraction of the input frame requires processing, the computation hardware is no longer required to be over-provisioned to keep up with the entire frame. To control the trade-off between inference quality and resource consumption, three degrees of freedom

are mainly considered: *spatial* [6], [40], *temporal* [13], and *quality* [9], [33], [41], [42]. To this end, *canvas-based attention scheduling* was proposed by Hu *et al.* [7] to manipulate the spatial and temporal dimension and take advantage of canvas-based processing to improve efficiency. They derived a tight schedulability bound with EDF and object quantization. However, their proposed framework does not adapt to different system loads online and only processes tasks in a best-effort manner. Moreover, the inefficiency caused by quantization prevents further improving the performance. In this paper, we extend canvas-based attention scheduling with resizing and lift the quantization constraint, leading to an interesting scheduling problem and a more efficient system.

VII. Discussion

The work presented in this paper is an attempt to explore the emerging area of spatiotemporal scheduling for AI tasks that process (multidimensional) spatial inputs, with the option of imprecise computation by resizing, subject to constraints on time (e.g., object inspection frequency constraints) and constraints on sizes of "bins" that are presented for processing with the AI algorithm. We discuss several limitations and possible extensions of this work.

a) Scheduling Policy: To derive the approximate solution, the algorithms assume that objects are placed in canvas bins according to an EDF order. However, because of the bin-packing-like nature of the canvas-based scheduling problem, EDF is not necessarily optimal. The topic remains open for designing an optimal scheduling policy that further improves canvas utilization.

b) Computational Overheads: The implementation of our scheduler and background subtraction modules does pose some overhead. What makes the scheme feasible is the fact that the AI-based perception subsystem runs on a GPU (and is itself quite computationally heavy). Background subtraction and object scheduling, in contrast, use CPU cores. Thus, there is no conflict and the GPU throughput remains unaffected, although the end-to-end latency is augmented with the CPU overhead. If this is an issue, the GPU deadline used in our equations can simply subtract the CPU overhead (which is always less than one frame period) from the original deadline.

c) Per-class treatment: In this paper, we use the same accuracy-size profile for all object classes. However, different types of objects may have different sensitivity to resizing. Also, depending on application specifics, different object classes may have different importance. An algorithm incorporating per-class object treatment is reserved for future work.

d) Problem Formulation: We followed the original canvas scheduling paper [7] and defined the tasks as quasi-periodic with implicit deadlines. The period of a task was determined by considering the factors affecting the rate of change of the object state; more specifically, its speed. Other formulations are possible. For example, one can consider a formulation similar to [13], where there is no explicit deadline for objects and the detection frequency of objects is affected by the system load. Combined with resizing and canvas-based processing, the optimization problem becomes to minimize system uncertainty caused by downsizing and object volatility.

e) Safety Considerations: Although resizing facilitates schedulability, it can cause false negatives in detection and may, in turn, affect scheduling, and potentially trigger dangerous feedback in detection and scheduling. Since the background subtraction indicates the existence of an object and then the detection algorithm decides its type and precise location, when an object is previously detected, it will only be removed if the inference algorithm confirms there's no object in the tracked location. Therefore, we can enforce that the first detection should always be at the original size, and a tracked object can only be removed if no detection is found at the tracked location at the original size. This way we eliminate false negatives caused by downsizing and prevent feedback in detection and scheduling.

VIII. Conclusions

The paper touched on an interesting dilemma that arises in scheduling real-time perception pipelines where different focus locales (selected from an original input frame) are consolidated into a smaller area (the canvas frame) for AI-based application-specific quasi-periodic inspection, subject to respective implicit deadlines. The dilemma lies in the fact that the best order in which objects need to be considered for attaining the most efficient packing of canvas frames is different from the best order in which objects need to be considered to meet deadlines. Packing frames more efficiently, however, increases effective available capacity, which improves the ability to meet deadlines. Thus, the optimal policy for this spatiotemporal scheduling problem remains unclear. The paper established a new baseline by deriving a bound for the equivalent canvas volume and using it to control object resizing online under EDF. Future extensions will consider optimality results that further improve the performance for canvas-based attention scheduling.

Acknowledgement

This work was sponsored in part by ARL W911NF-17-2-0196, NSF CNS 20-38817, IBM (IIDAI), the Boeing Company, the National Research Foundation, Singapore under NRF-NRFI05-2019-0007, and the National Natural Science Foundation of China under No. BC0301315 and BC0301340.

References

[1] R. Hussain and S. Zeadally, "Autonomous cars: Research results, issues, and future challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1275–1313, 2018.

[2] C. Amrutha, C. Jyotsna, and J. Amudha, "Deep learning approach for suspicious activity detection from surveillance video," in *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*. IEEE, 2020, pp. 335–339.

[3] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 1. Ieee, 2004, pp. I–I.

[4] G. Amato, F. Carrara, F. Falchi, C. Gennaro, and C. Vairo, "Car parking occupancy detection using smart camera networks and deep learning," in *2016 IEEE Symposium on Computers and Communication (ISCC)*. IEEE, 2016, pp. 1212–1217.

[5] H. Perez, J. H. Tah, and A. Mosavi, "Deep learning for detecting building defects using convolutional neural networks," *Sensors*, vol. 19, no. 16, p. 3556, 2019.

[6] S. Liu, S. Yao, X. Fu, R. Tabish, S. Yu, A. Bansal, H. Yun, L. Sha, and T. Abdelzaher, "On removing algorithmic priority inversion from mission-critical machine inference pipelines," in *2020 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2020, pp. 319–332.

[7] Y. Hu, I. Gokarn, S. Liu, A. Misra, and T. Abdelzaher, "Under-provisioned gpus: On sufficient capacity for real-time mission-critical perception," in *2023 32nd International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2023, pp. 1–10.

[8] Y. Hu, I. Gokarn, S. Liu, A. Misra, and A. Tarek, "Work-in-progress: Algorithms for canvas-based attention scheduling with resizing," in *2023 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2023, pp. 435–438.

[9] Y. Hu, S. Liu, T. Abdelzaher, M. Wigness, and P. David, "On exploring image resizing for optimizing criticality-based machine perception," in *2021 IEEE 27th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. IEEE, 2021, pp. 169–178.

[10] G. Gordon and R. Tibshirani, "Karush-kuhn-tucker conditions," *Optimization*, vol. 10, no. 725/36, p. 725, 2012.

[11] J. Januszewski, "Packing rectangles into the unit square," *Geometriae Dedicata*, vol. 81, no. 1-3, pp. 13–18, 2000.

[12] T. F. Abdelzaher, V. Sharma, and C. Lu, "A utilization bound for aperiodic tasks and priority driven scheduling," *IEEE Transactions on Computers*, vol. 53, no. 3, pp. 334–350, 2004.

[13] S. Liu, X. Fu, M. Wigness, P. David, S. Yao, L. Sha, and T. Abdelzaher, "Self-cueing real-time attention scheduling in criticality-driven visual machine perception," in *In Proc. 28th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, May 2022.

[14] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J. T. Lee, S. Mukherjee, J. Aggarwal, H. Lee, L. Davis *et al.*, "A large-scale benchmark dataset for event recognition in surveillance video," in *CVPR 2011*. IEEE, 2011, pp. 3153–3160.

[15] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[16] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[17] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2021.

[18] R. Pujol, H. Tabani, L. Kosmidis, E. Mezzetti, J. Abella, and F. J. Cazorla, "Generating and exploiting deep learning variants to increase heterogeneous resource utilization in the NVIDIA XAVIER," in *31st Euromicro Conference on Real-Time Systems (ECRTS 2019)*, vol. 23, 2019.

[19] Y. Xiang and H. Kim, "Pipelined data-parallel CPU/GPU scheduling for multi-DNN real-time inference," in *2019 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2019, pp. 392–405.

[20] W. Kang, K. Lee, J. Lee, I. Shin, and H. S. Chwa, "Lalarand: Flexible layer-by-layer CPU/GPU scheduling for real-time DNN tasks," in *2021 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2021, pp. 329–341.

[21] H. Li, J. K. Ng, and T. Abdelzaher, "Enabling real-time AI inference on mobile devices via GPU-CPU collaborative execution," in *2022 IEEE 28th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. IEEE, 2022, pp. 195–204.

[22] I. De Albuquerque Silva, T. Carle, A. Gauffriau, and C. Pagetti, "ACETONE: Predictable programming framework for ML applications in safety-critical systems," in *34th Euromicro Conference on Real-Time Systems (ECRTS 2022)*, 2022.

[23] M. Yang, S. Wang, J. Bakita, T. Vu, F. D. Smith, J. H. Anderson, and J.-M. Frahm, "Re-thinking CNN frameworks for time-sensitive autonomous-driving applications: Addressing an industrial challenge," in *2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2019, pp. 305–317.

[24] H. Zhou, S. Bateni, and C. Liu, "S^ 3DNN: Supervised streaming and scheduling for GPU-accelerated real-time DNN workloads," in *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2018, pp. 190–201.

[25] T. Amert, N. Otterness, M. Yang, J. H. Anderson, and F. D. Smith, "GPU scheduling on the NVIDIA TX2: Hidden details revealed," in *2017 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2017, pp. 104–115.

[26] N. Capodieci, R. Cavicchioli, M. Bertogna, and A. Paramakuru, "Deadline-based scheduling for GPU with preemption support," in *2018 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2018, pp. 119–130.

[27] M. Yang, N. Otterness, T. Amert, J. Bakita, J. H. Anderson, and F. D. Smith, "Avoiding pitfalls when using nvidia gpus for real-time tasks in autonomous systems," in *30th Euromicro Conference on Real-Time Systems (ECRTS 2018)*, 2018.

[28] J. Hanhirova, T. Kämäräinen, S. Seppälä, M. Siekkinen, V. Hirvisalo, and A. Ylä-Jääski, "Latency and throughput characterization of convolutional neural networks for mobile computer vision," in *Proceedings of the 9th ACM Multimedia Systems Conference*, 2018, pp. 204–215.

[29] N. Otterness and J. H. Anderson, "AMD GPUs as an alternative to NVIDIA for supporting real-time workloads," in *32nd Euromicro Conference on Real-Time Systems (ECRTS 2020)*, 2020.

[30] N. Otterness, M. Yang, S. Rust, E. Park, J. H. Anderson, F. D. Smith, A. Berg, and S. Wang, "An evaluation of the NVIDIA TX1 for supporting real-time computer-vision workloads," in *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2017, pp. 353–364.

[31] I. S. Olmedo, N. Capodieci, J. L. Martinez, A. Marongiu, and M. Bertogna, "Dissecting the CUDA scheduling hierarchy: a performance and predictability perspective," in *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2020, pp. 213–225.

[32] S. Voronov, S. Tang, T. Amert, and J. H. Anderson, "AI meets real-time: Addressing real-world complexities in graph response-time analysis," in *2021 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2021, pp. 82–96.

[33] S. Yao, Y. Hao, Y. Zhao, H. Shao, D. Liu, S. Liu, T. Wang, J. Li, and T. Abdelzaher, "Scheduling real-time deep learning services as imprecise computations," in *In Proc. IEEE International Conference on Embedded and Real-time Computing Systems and Applications (RTCSA)*, August 2020.

[34] S. Lee and S. Nirjon, "Subflow: A dynamic induced-subgraph strategy toward real-time dnn inference and training," in *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2020, pp. 15–29.

[35] S. Heo, S. Cho, Y. Kim, and H. Kim, "Real-time object detection system with multi-path neural networks," in *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2020, pp. 174–187.

[36] A. Soyyigit, S. Yao, and H. Yun, "Anytime-Lidar: Deadline-aware 3d object detection," in *2022 IEEE 28th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. IEEE, 2022, pp. 31–40.

[37] S. Bateni and C. Liu, "ApNet: Approximation-aware real-time neural network," in *2018 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2018, pp. 67–79.

[38] M. Yuhas, D. J. X. Ng, and A. Easwaran, "Design methodology for deep out-of-distribution detectors in real-time cyber-physical systems," in *2022 IEEE 28th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. IEEE, 2022, pp. 180–185.

[39] T. Abdelzaher, K. Agrawal, S. Baruah, A. Burns, R. I. Davis, Z. Guo, and Y. Hu, "Scheduling idk classifiers with arbitrary dependences to minimize the expected time to successful classification," *Real-Time Systems*, pp. 1–60, 2023.

[40] S. Liu, S. Yao, X. Fu, H. Shao, R. Tabish, S. Yu, A. Bansal, H. Yun, L. Sha, and T. Abdelzaher, "Real-time task scheduling for machine perception in intelligent cyber-physical systems," *IEEE Transactions on Computers*, 2021.

[41] S. Yao, Y. Zhao, A. Zhang, L. Su, and T. Abdelzaher, "Deepiot: Compressing deep neural network structures for sensing systems with a compressor-critic framework," in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. ACM, 2017, p. 4.

[42] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 2464–2469.