5-2024

Enabling criticality-aware optimized machine perception at the edge

Ila Nitin GOKARN

ENABLING CRITICALITY-AWARE OPTIMISED MACHINE PERCEPTION AT THE EDGE

ILA NITIN GOKARN

SINGAPORE MANAGEMENT UNIVERSITY
2024

Enabling Criticality-Aware Optimised Machine Perception at the Edge

by

Ila Nitin GOKARN

Submitted to School of Computing and Information Systems in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Computer Science

Dissertation Committee:

Archan MISRA (Supervisor / Chair)
Professor of School of Computing and Information Systems
Singapore Management University

David LO
Professor of School of Computing and Information Systems
Singapore Management University

Dong MA
Assistant Professor of School of Computing and Information Systems
Singapore Management University

Tarek ABDELZAHER
Professor of Computer Science
University of Illinois at Urbana Champagne

Singapore Management University
2024

I hereby declare that this PhD dissertation is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in this dissertation.

This PhD dissertation has also not been submitted for any degree in any university previously.

Ila Nitin GOKARN

20 May 2024

Abstract

Cyber-physical systems and applications have fundamentally changed people and processes in the way they interact with the physical world, ushering in the fourth industrial revolution. Supported by a variety of sensors, hardware platforms, artificial intelligence and machine learning models, and systems frameworks, CPS applications aim to automate and ease the burden of repetitive, laborious, or unsafe tasks borne by humans. Machine visual perception, encompassing tasks such as object detection, object tracking and activity analysis, is a key technical enabler of such CPS applications. Efficient execution of such machine vision perception tasks on resource-constrained edge devices, especially in terms of ensuring both high fidelity and processing throughput, remains a formidable challenge. This is due to the continuing increase in resolution of sensor streams (e.g., video input streams generated by 4K/8K cameras and high-volume event streams generated by emerging neuromorphic event cameras) and the computational complexity of the Deep Neural Network (DNN) models that underpin such perception capabilities, which overwhelms edge platforms, adversely impacting machine perception efficiency. This challenge is even more severe when a perception pipeline operating on a single edge device must process *multiple concurrent* video streams for accurate sense-making of the physical world. Given the insufficiency of the available computation resources, a question then arises on whether parts of the perception task can be prioritized (and executed preferentially) to achieve highest task fidelity while adhering to the resource budget. This thesis introduces the paradigm of *Canvas-based Processing* and *Criticality Awareness* to tackle the challenge of multi-sensor machine perception pipelines on resource-constrained platforms. The proposed paradigm guides perception pipelines and systems on "what" to pay attention to in the sensing field and "when", across multiple camera streams, to significantly increase both perception fidelity under computational constraints and achievable system throughput on a single edge device. By creating spatial and temporal degrees of freedom for stimuli/regions of

interest from their original video streams, such a perception pipeline can "pick and choose" which stimuli to ascribe more priority for preferential DNN inference over time, thereby reducing the total computational load. The thesis explores how such prioritized and selective processing, across multiple RGB and event sensor streams, needs to be designed to support both non-streaming and streaming perception tasks. With multiple strategies for fine-tuning such a perception pipeline for real-world deployment characteristics such as bandwidth constrained wireless networks, variable workloads at the edge, spatial overlap between cameras, this thesis demonstrates that it is possible to achieve multiplicative gains in processing throughput with no cost to DNN task accuracy, across multiple concurrent RGB and event camera streams at the resource-constrained edge. The proposed techniques are especially applicable for real-time multi-sensor machine perception tasks such as drone-based surveillance and multi-camera traffic analysis.

Table of Contents

List of Figures

List of Tables

Acknowledgments

This dissertation would not have been possible without the support and contributions of many individuals who pumped wind in my sails and motivated me throughout my PhD journey.

First and foremost, a heartfelt thank you to my advisor, Prof Archan Misra, for your guidance and support throughout my PhD journey. I'm ever so grateful that you accepted me as your student, fresh out of the corporate world, and gave me the platform to grow into the researcher I am today. You have always pushed me to expand my thinking process and given me innumerable opportunities to explore various facets of research. What has truly inspired me is your kindness and your ability to make time for your students no matter how busy you are in your other professional activities. Thank you for all the mentorship and life advice you have given me, I hope to emulate your teachings as I grow in my career.

I would also like to thank my dissertation committee members, Professors Tarek, David, Zimu, and Dong, who graciously lent their time and wisdom at various stages in the development of this dissertation. I am also grateful to my collaborators - Prof Tarek, Hemanth, Yigong, and Shenzhong, who provided spirited discussions and analyses in the development of the canvas-based processing paradigm. I thank Isuri, Prof Thivya, Argha, and Nuwan, for detailed discussions on different aspects of event camera operation and neuromorphic edge computing. I also extend my thanks to Dr. Chulhong Min and Prof. Fahim Kawsar for hosting me as an intern at Nokia Bell Labs, and for the illuminating discussions and patent brain-storming sessions which gave me new research perspectives.

I wish to express my sincere gratitude to Prof. Venky and Prof. Swapna - you both have been incredibly supportive of my career goals since my undergraduate days when I first expressed a desire to pursue a PhD. Your advice to explore the corporate world to gain exposure and maturity before diving into research was so invaluable; it is a piece of advice I find myself giving others who express an interest in pursuing a PhD. I will never forget the coffee chat where you both enthusiastically encouraged me to apply for a PhD at SMU, I am grateful for all your advice over the last decade.

I'm incredibly lucky to have shared my PhD journey with the most tenacious group - Rosi, Dulanga, and Tai, thank you for all the strategy/pep talks and milestone celebrations over the years! Rosi, I'm so grateful that we shared this journey together - innumerable ice creams, Hollin bubble tea treks, and both our weddings - thank you for all the memories. To my lab mates, Anuradha, Darshana, Dulaj, Jiyan, Dhanuja, Vengat, Kasun, Manoj, Siyan, Andrew, and Hemanth, thank you for all the tea breaks, billiards and foos-ball games, and chats. Last but not the least, to Prof Thivya, Meera and Kasthuri, three veritable sources of information and mentorship in my PhD journey, thank you so much for all the advice you gave me. I also thank SMU for the Full Scholarship, and support extended by Prof. Baihua, Alice, Pei Huan, Chui Ngoh, Nikki, Caroline, and Veronica throughout my PhD.

Finally, and most importantly, my Guru and my family - Amma, Papa, and Mallika, who are my absolute rock-pillars of love and support. Who I am today is all credit to you; you are my box of blue crayons, the ones who colour my sky. Thank you for teaching me how to balance everything big and small in life. Mallika, my little-big sister, thank you for making sure I was resilient and had fun every step of the way. To my husband, Anvesh, you've been so incredibly patient with me with your kindness, love, and steady supply of biryani, thank you for exploring the true meaning of partnership with me, for taking care of me when I worked long hours, and for *always* motivating me. A big thank you also to my in-laws for their constant support and encouragement, and my grandparents for their continued blessings.

To

Amma, Papa, Mallika, and Anvesh

Chapter 1

Introduction

1.1 Multi-Sensor Visual Perception at the Edge

Visual machine perception is a fundamental enabler of cyber-physical systems that enables applications such as augmented reality (AR/VR), autonomous cars or drones, and assistive robots. Real-time sensing and accurate sense-making is crucial for these applications to execute sub-tasks such as object detection, localization, and mapping, yielding the ability to perceive, navigate, and interact with their physical environment. State of the art visual perception pipelines rely on the processing of video streams from commodity cameras, and often leverage multiple cameras concurrently to achieve fine-grained, multi-perspective, or wide-range visual perception [53]. In parallel, advances in edge computing have enabled Deep Neural Network (DNN) based models on resource constrained pervasive edge devices to consume such camera streams and distill the perception of physical phenomena into actionable knowledge [53]. However, simultaneous advancements in (i) newer and more complex visual Deep Learning Networks (DNNs) that impose higher memory-latency complexity, and (ii) higher resolution cameras (i.e. 4k/8k cameras [12] and neuromorphic event cameras capable of 1 billion events/second [4]) that generate data at higher rates and resolutions, make it challenging to guarantee real-time and accurate visual perception over multiple concurrent camera streams using a *single*

resource-constrained edge device.

In this thesis, I introduce the paradigm of *Criticality Aware Canvas-based Processing* and explore multi-sensor visual perception pipelines on resource constrained edge platforms, with a focus on improving end-to-end system efficiency. Inspired by the concept of *attention* from human psychology, I consider criticality-awareness as the selective concentration of limited computation resources on a smaller subset of stimuli among multiple perceivable stimuli [81]. While the concept of *attention* has been adapted by the deep learning community to make DNNs dynamically focus on relevant parts of the input data, I make a key distinction between attention mechanisms and criticality-awareness in that the focus is on fine-tuning the *entire* edge system and application across *multiple inputs*, not the structure of the DNN itself. Canvas-based Processing involves the extraction of critical stimuli or Regions of Interest (RoI) from multiple concurrent camera streams, to create a spatially and temporally differentiated curation of each stimuli from its original video source. This allows the system to spatiotemporally multiplex the selected stimuli onto a shared, resource-limited computational resource for DNN-based inference. In general, a *Canvas* is defined as the maximum size of an input frame that a DNN operating on an edge GPU device can consume to yield an inference throughput above the desired real-time processing threshold. This blank canvas frame essentially serves to define a spatial budget within which the perception pipeline must curate and construct a composite image, constructed of extracted prioritised critical stimuli via 2D bin-packing for downstream DNN inference such as object detection, illustrated in Figure 1.1. In this thesis, I discuss various aspects of fine-tuning such a system which cater to considerations such as: (i) *what* subset of stimuli the system selects for fine-grained perception of downstream DNN task execution (ii) *when* the system re-selects already perceived objects to monitor changes in relative criticality to the perception task, (iii) *how* the resource-constrained edge GPU schedules critical stimuli for downstream DNN inference, and (iv) *how* stimuli from multiple other sensors might inform the criticality of different objects/events in the sensing field

relative to the perception task. I hypothesize that, in general, continuous recognition is not strictly necessary for machine perception; this is based on the observation of how human perception follows similar principles of prioritization and selective attention [81] to maximize human neural efficiency. Finally, I conclude that by fine-tuning the system's understanding of the different stimuli across multiple sensors, visual perception tasks operating on a single edge device may achieve multiplicative gains in system capacity/throughput (i.e. number of cameras that a single edge GPU can concurrently process) while maintaining DNN task accuracy.

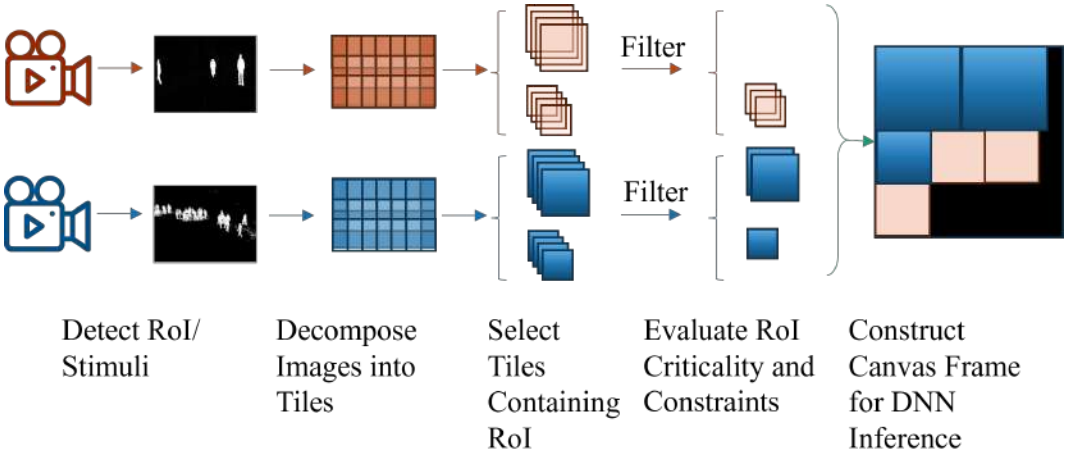

Figure 1.1: Conceptual overview of *Criticality Aware Canvas-based Processing* over multiple input streams at a single edge GPU

1.2 Key Challenges & Opportunities

Advances in computer vision and edge computing have pushed the state of the art in visual perception on resource constrained edge devices to consume multiple sensor streams and distill the perception of physical phenomena into actionable knowledge. This trend fundamentally hinges on advancements in three distinct areas - (i) new and continuously improving deep learning network architectures (ii) innovative ways to gather, clean, and consume the visual sensing data and (iii) hardware infrastructure that supports the communication and computation of the available data. The interplay between these three pillars of advancement, each important and significant in their own right, creates significant challenges to machine perception on edge devices.

Deep learning models today are getting deeper and more complex with hundreds of computational layers, with ever increasing space-time complexity and cost to trainability. On the other hand, perception data is characterised by high volumes of data being generated at different velocities and resolutions. For example, latest neuromorphic event cameras can generate 12 Million events/second [3] to 1 billion events/second [4] when capturing dynamic, fast-moving scenes, amounting to several GB/sec of raw data to be transmitted and processed by the perception pipeline. Such volumes of data makes efficient and accurate machine perception a challenge. An exponential growth in the number and variety of sensors required for a single perception task adds another layer of complexity to designing edge-based perception pipelines. Lastly, innovations in chip and processor designs such as the Jetson AGX Orin based on the NVIDIA Ampere GPU architecture[1], and the Dolphin Raptor on the Tiny RAPTOR processor [2], bring more powerful computation capabilities to edge devices. However, modern deep learning models, designed predominantly for the cloud, are out-pacing Moore's law [118] and prohibitively intensive in their computational load, significantly overwhelming edge platforms, and incurring high processing and computation latencies for perception tasks [5]. Machine perception tasks however, need to be *accurate* with *high processing throughput* to proactively respond to dynamically changing environments.

The prevailing wisdom to overcoming these three distinct challenges is to adopt a combination of the following strategies:

- **Deploy Smaller Edge-Scale DNNs:** Deploying smaller, less accurate, and computationally cheaper DNN models on edge scale devices incurs lesser processing latency at the cost of perception task accuracy. For example, in the latest iteration of the YOLO family of object detection models - YOLOv8 [139], the smallest edge-scale model YOLOv8-nano or YOLOv8n featuring 3.2 million parameters and 8.7 billion FLOPs achieves real-time inference of 32 FPS on a Jetson AGX Orin Nano [1] with a mean average precision of

37.3% [139] on the COCO dataset [92]. On the other hand, YOLOv8-large or YOLOv8l featuring 43.7 million parameters and 165.2 billion FLOPs incurs significant processing latency to yield a processing throughput of only 6 FPS but with a significantly higher mean average precision of 52.9% [139] on the COCO dataset [92]. Edge-based applications often prefer throughput (YOLOv8n) to accuracy (YOLOv8l), to tackle cost/complexity-vs-accuracy tradeoffs. Approaches for static and dynamic DNN model optimization (such as pruning [91], knowledge distillation [58] or sparsification [140]) help to reduce, but cannot completely eliminate, the accuracy gap resulting from such smaller, edge-friendly models.

- **Downsample Input Resolutions:** Edge-scale DNNs incur lower processing latency over smaller, downsampled input image sizes with pipelines typically downsampling the inputs to $\leq 640 \times 640$ for DNN inference. Such downsampling sacrifices the high-definition sensing capabilities embedded within the onboard sensors, adopting instead low-resolution images that promote throughput over accuracy. Recent works have shown the efficacy of selective downsampling to generate multi-resolution images [146, 32, 74], retaining higher resolutions for regions of interest in the input video, which could prove useful in the context of multi-sensor visual pipelines.

- **Deploy More Powerful Edge GPUs:** Larger, more capable edge GPU devices such as the production-class Jetson AGX Orin 32/64GB [1] alleviate some of the challenges to deployment by offering greater resources promoting both throughput and accuracy albeit with high infrastructure costs ($\sim$USD 2000 per unit).

- **Selective Computation and Offload:** Recent works have proposed either (a) *Criticality-aware processing* techniques which process selected Regions of Interest (RoI) with higher fidelity by masking[56, 97], batching [96], patching [74], tiling [154], or offloading processing to the cloud or a more powerful

edge device [161] and (b) *Selective computation approaches*, where certain DNN layers are simplified [26] or skipped [142] with imprecise computations [24, 80]. Current approaches, however, do not consider scenarios where *multiple* sensor streams, with dynamically varying scene characteristics share the same computational resources on an edge device and must be processed *concurrently*.

Although not exhaustive, these strategies indicate that there is an opportunity to explore the development of *content-aware* and *criticality-aware* mechanisms across *multiple* sensor data streams that are processed by a single perception pipeline at the resource constrained edge. By introducing Canvas-based Processing and the concept of spatiotemporal dissociation of the perceived stimuli/objects from the camera input stream itself, the pipeline can gain significant degrees of freedom to "pick and choose" which stimuli to ascribe more priority to for preferential processing, creating multiplicative gains in processing throughput with no loss of perception task accuracy. Such prioritized processing also helps to tackle the challenge of *priority inversion* [61] that has been observed in the real-time stream processing literature, where low-priority regions of the sensing field (such as the background/sky) are given higher priority for processing over high-priority regions of the critical stimuli/RoI. To understand the practical opportunities and challenges of such criticality-aware canvas-based machine perception, I explore three key scenarios.

Scenario 1: Drone-based Factory/Retail Floor Monitoring

Consider a factory floor that is surveilled by a fleet of M drones which monitor worker and equipment safety; each drone observing discrete sections of the site, while wirelessly transmitting the camera streams to an edge node for application-specific inference, illustrated in Figure 1.2. Alternatively, envision a retail shopping floor with a fleet of drones providing personalised service to customers by helping them navigate to aisles that stack items they are looking for, or picking up items that they

might have forgotten. Each drone covers a *distinct, non-overlapping physical section* of the factory/retail floor and perceives unique object size/appearance distributions per-camera. Such deployments require that all drone camera feeds be simultaneously transmitted to and inferred upon by a single edge GPU for real-time processing (i.e. quality of service) to each drone.

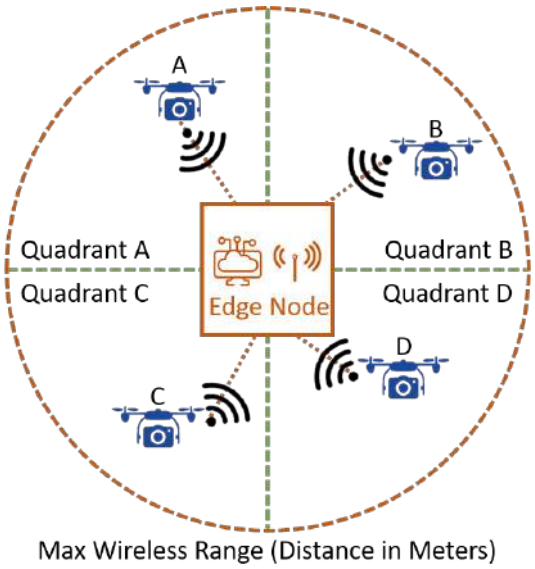

Figure 1.2: Scenario 1: Multiple discrete autonomous drone camera streams transferred wirelessly to, and processed by a single edge node

To answer this challenge, an edge node must spatially channel limited computational resources to selected stimuli from M concurrent drone camera feeds. Such a perception pipeline must be able to:

1. Optimize the transmission of camera frames from the drone to the edge over the wireless network to (i) adapt to available bandwidth and latency in real-world wireless networks, and (ii) preserve resolutions of the regions of the frame that might contain objects of interest to the perception pipeline.

2. Detect critical stimuli or Regions of Interest (RoI) across all camera streams using a computationally lightweight mechanism.

3. Extract regions of interest from the individual camera feeds to create a spatial degree of freedom for the detected RoI.

4. Filter RoI representations to ensure the most appropriate instance of each unique RoI (i.e. appropriately extracted or not unduly cropped) is considered/selected for inference.

5. Construct a canvas frame by bin-packing selected RoI on a blank canvas frame for DNN inference within the DNN processing deadline. Such canvas construction spatially channels the GPU resources to the selected RoI from multiple input camera streams.

This thesis accomplishes objectives 2-5 to explore the introduction of a spatial degree of freedom for canvas-based processing at the edge, and presents the design trade-offs which determine the operating bounds of such a perception pipeline in Chapter 2. Chapter 3 extends the pipelines across the drones and the edge to adapt to the available wireless network bandwidth (i.e. objective 1) and the available workload as perceived across the multiple camera streams to adaptively and opportunistically facilitate gains in both throughput and accuracy.

Scenario 2: Multi-Camera Surveillance Systems

Consider a city traffic monitoring application where an edge node deployed on a road-side unit must process M stationary traffic-light camera feeds with groups of cameras covering the same physical space from different perspectives or viewing angles, illustrated in Figure 1.3. One could make two significant observations for real-time video analysis across cameras in such a deployment. First, these deployments generally feature distinct pre-determined spatial overlaps between cameras to prevent surveillance blind spots. Second, each camera observes unique traffic volumes at different times of the day, with a significant amount of slow-moving/stationary traffic during peak hours. Both phenomena compel the DNN to repeat temporally redundant inference over multiple instances of the same object/RoI as perceived by two or more spatially overlapped cameras in the sensing field, yielding under-optimized and wasteful DNN computation at the edge.

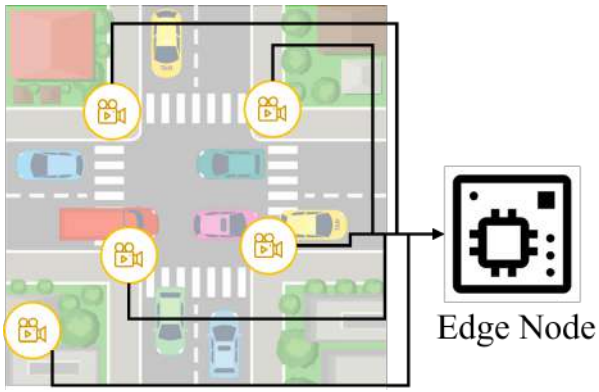

Figure 1.3: Scenario 2: Edge-based traffic surveillance with multiple spatially-overlapped cameras

To improve DNN inference for such deployments, a perception pipeline featuring canvas-based spatiotemporal multiplexing must introduce a temporal degree of freedom in addition to the existing spatial degree of freedom. This allows the perception pipelines to not only fine-tune (i) "which" instance of a unique stimuli as observed from multiple spatially overlapped cameras must be chosen for inference, but also (ii) "when" already perceived stimuli/objects/RoI must be re-selected for DNN inference. Such a pipeline must achieve the following:

1. Maintain a form of "temporal admission control" of RoI across cameras over time for inclusion on the canvas frame and subsequent DNN inference. RoIs observed from multiple camera sensors should be pre-processed to not only share the pixels within a single canvas frame, but also be differentially interleaved in time (and even dropped) to optimally utilize pixels across multiple consecutive canvas frames. Intuitively, RoIs corresponding to faster moving objects likely need to be processed more frequently than slower-moving or quasi-stationary objects.

2. Explicitly account for the reality that multi-camera urban deployments often exhibit non-trivial *spatial overlap* between cameras. Such overlap implies that multiple cameras sometimes monitor the same object from different perspectives and thus possess a level of information redundancy. The pipeline must autonomously monitor and exploit such object-level redundancy to reduce the

number of RoI across the M camera streams that must be evaluated.

3. Balance between an accurate perception of the state of the physical world and delay induced by the "temporal admission control" of RoI in the processing pipeline by optimizing the perception pipeline for localisation latency in addition to object detection accuracy.

4. Account for real-world bandwidth and latency constraints in a wireless network to create intelligent camera stream ingest pipelines for evaluation of per-camera spatial overlap and spatiotemporal filtering of RoI.

5. Construct a canvas frame by bin-packing RoI that are filtered over time and across spatially-overlapped cameras for DNN inference.

This thesis focuses on these objectives to offer techniques and insights on the bounds of operation for a perception pipeline executing spatiotemporal multiplexing over multiple concurrent spatially-overlapped cameras at a single edge device in Chapter 4. Additionally, discussions are presented on how *the resource-constrained edge GPU must schedule critical stimuli for downstream DNN inference derived with spatiotemporal schedulability bounds and Earliest Deadline First based bin-packing algorithms.*

Scenario 3: Low-power Visual Occupancy Detection in Poor Lighting

Consider dimly-lit warehouses or indoor retail spaces dedicated to inventory or storage where traditional RGB cameras may struggle to capture clear images. Multiple cameras may also be required on each storage aisle to detect worker occupancy, intruders, and/or safety concerns, resulting in high infrastructure costs, particularly energy demands. This need is complicated by the reality that recent advancements in RGB camera quality or resolution pose increased energy consumption during operation, creating a bottleneck to low-power efficient sensing on edge devices. In comparison, biologically-inspired neuromorphic event cameras mimic the human

retina to capture a continuous temporal stream of events that indicate changes in light intensity in the sensing field, illustrated in Figure 1.4. Event cameras operate with extremely low power consumption($10 - 30mW$), have highly reactive $O(\mu s)$ sensing capabilities, support higher dynamic range up to $140dB$, and are capable of capturing events even in poor or dimly lit scenarios. "Images" can be synthesized from this continuous temporal stream of events, resulting in low-dimension "framed representations" that capture the sensing field with limited to no information on nuanced features like colour and texture. While some event cameras produce only a continuous stream of events [4], latest advancements have introduced hybrid sensors such as Inivation's DVS 346 [3] which is capable of both low spatial resolution gray-scale frames and high temporal resolution event streams, illustrated in Figure 1.5. Low-dimension event representations captured at $O(\mu s)$ latency and feature-rich high-dimension standard/grayscale images captured at ≤ 30 FPS represent two extremes in sensing fidelity, creating an intriguing opportunity where both sensor streams can be intelligently fused to jointly contribute to low-power low-light visual perception. A canvas-based perception pipeline can then concurrently process multiple such Frame+Event fused representations, providing multiplicative gains in processing throughput with negligible loss in DNN task accuracy.

To evaluate this opportunity, canvas-based perception pipelines must leverage neuromorphic event streams to supplement traditional video streams, triggering either sensor stream on demand per the perception pipeline's needs and optimization goals. To achieve this vision, the perception pipeline must be able to:

1. Ingest high-volume event streams and feature-rich standard camera streams with intelligent mechanisms to trigger feature-rich RGB streams on demand to thread the balance between energy consumption, accuracy, and achievable processing throughput.

2. Fuse both event+RGB streams asynchronously into a common representation of the sensing field.

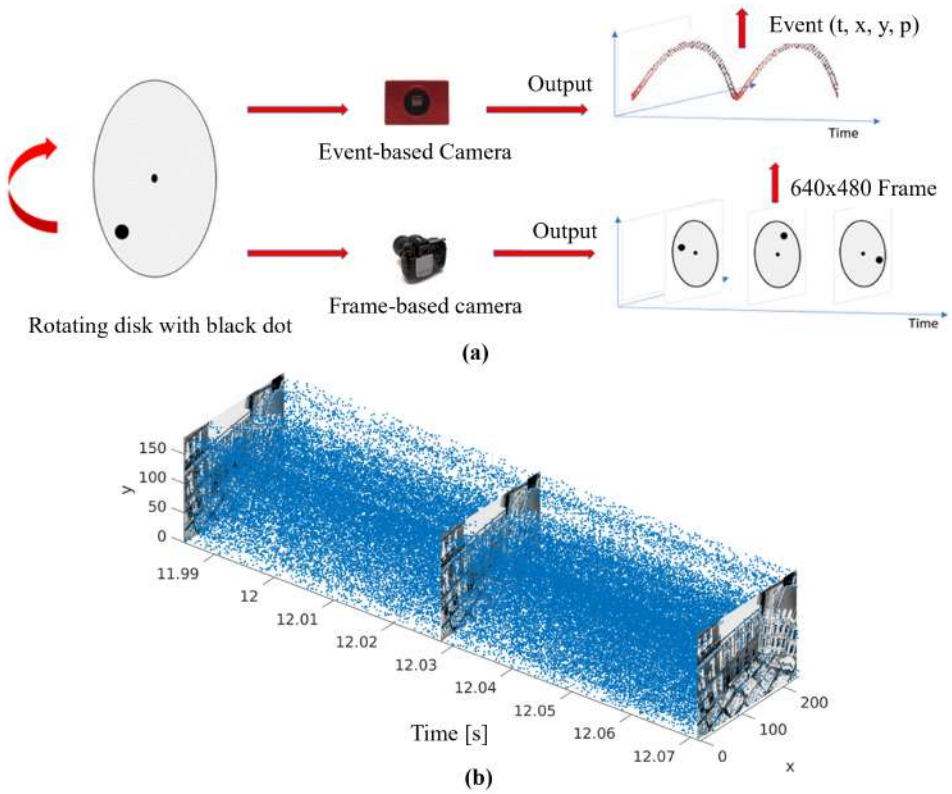

Figure 1.4: Scenario 3: Low-power Visual Occupancy Detection in Poor Lighting: (a) Visualizing the Differences in Sensor Stream Outputs between an RGB Camera and an Event-based Neuromorphic Camera [104] (b) Visualising the DAVIS camera output: Events at each pixel fired asynchronously constitutes a sensed 2D image [105]

3. Extract critical stimuli/regions of interest from the fused event+RGB representation for spatially multiplexed canvas construction.

This thesis addresses the above objectives, first principle studies, design trade-offs, and pipeline designs to leverage both low-latency event streams and feature-rich RGB streams in tandem to achieve canvas-based processing in challenging low-light scenarios with low-power consumption for execution, described in Chapter 5. This thesis also discusses event cameras as inherent criticality estimators and presents insights on distilling event streams to guide spatiotemporal multiplexing of RGB streams at the edge, similar to techniques discussed in Chapter 4.

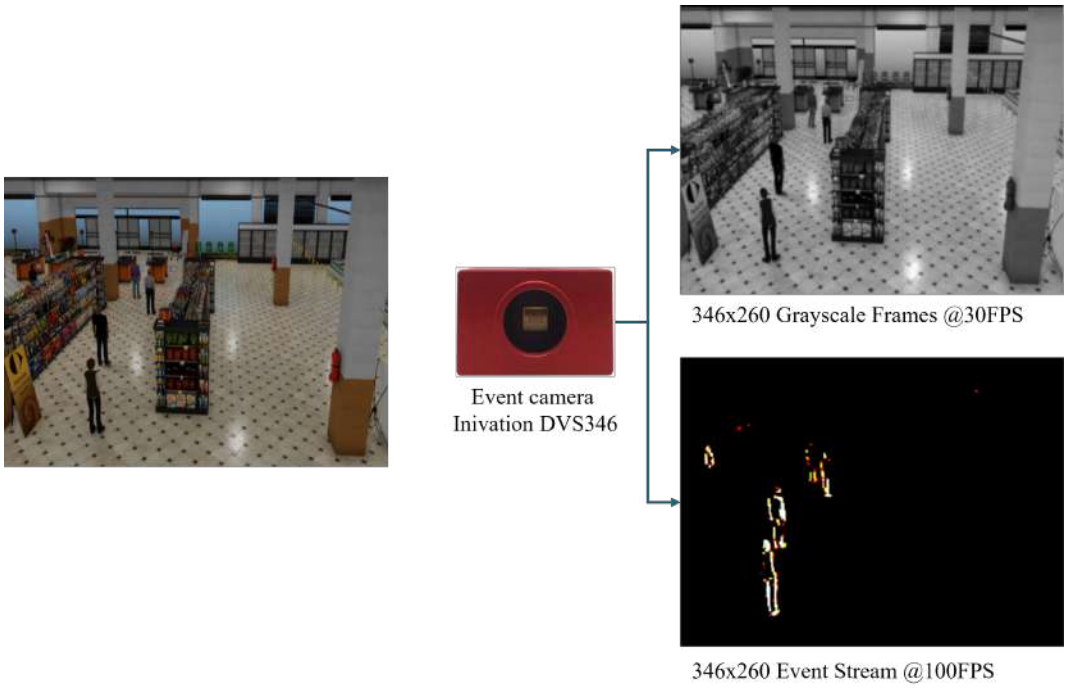

Figure 1.5: Scenario 3: Low-power Visual Occupancy Detection in Poor Lighting: DVS346 [3] observing a retail/warehouse floor with asynchronous (i) grayscale output of dimensions 346×260 at 30 FPS, and (ii) event stream of spatial resolution 346×260 captured at 100 FPS for visualization

1.2.1 Challenges & Learnings

I now present some of the pertinent challenges experienced and learnings gained during the design of criticality-based canvas-based processing pipelines.

1. **Lightweight Criticality Estimation:** Understanding what stimuli are considered critical by the perception pipeline can vary from application to application. For example, a drone might prioritize fast-moving nearby objects as more critical whereas a perimeter surveillance application might prioritize occluded or distant objects as more critical for early detection and tracking of potential intruders. Stimuli might be deemed critical based on the presence/absence of motion/velocity, size, and object distance from the camera, among other considerations, and can be modelled with depth perception models, physical motion models, and traditional computer vision techniques for edge detection, among many other methods. This estimation is not trivial across all notions of criticality. As criticality estimation is a pre-processing technique designed to

guide the operation of the rest of the perception pipeline, it is imperative that the adopted criticality estimation mechanism be *computationally lightweight* to facilitate non-blocking real-time processing of the camera streams. More specifically, the latency incurred in such criticality estimation must be low enough to not negate any benefits that accrue from downstream canvas-based spatially multiplexed DNN inference. In this thesis, I consider objects in motion as critical stimuli and leverage lightweight background subtraction based classical vision techniques to detect such stimuli across multiple concurrent camera streams.

2. **Spatial Degree of Freedom and Loss of Object Context:** The spatial degree of freedom borne out of extracting stimuli/RoI from its original frame (and therefore its background context), challenges the belief that accurate object detection requires sufficient background scene information. In this thesis, I leverage the findings of Barnea et. al. [21] to believe that background context provides only marginal benefits for object detection/localization tasks [155, 145, 158]. I show that extraction of stimuli/RoI from its context purely for detection purposes is justified, improving achievable throughput and detection accuracy without any loss of generalizability. Lastly, I recover all capabilities relying on background context of the detected object by post-processing the detections to the original input frame, thus treating object detection as a pre-processing pipeline that may be used for additional downstream functionality such as action recognition [90] or semantic scene labeling [93].

3. **Temporal Degree of Freedom and Keeping up with the Physical World:** Introducing a temporal admission control allows the pipeline to choose when to re-evaluate an RoI/stimulus but such mechanisms might risk a "lag" between perception of an object and physical reality. To counter such a risk, the admission control mechanism must keep up with physical reality by optimizing for localization latency in addition to detection accuracy. In this thesis, I leverage

the novel streaming accuracy metric [86] which compares the predictions of an object with the most recent groundtruth to evaluate the latency induced during scheduling and computation/inference of the object. I show that temporal admission control designed to incorporate such a streaming accuracy metric ensures that the pipeline keeps abreast of real-world object kinematics.

4. **Bandwidth and Latency Requirements:** Transmission of multiple camera streams over a wireless network for simultaneous inference on an edge device faces two key challenges in real-world wireless deployments. First, bandwidth and latency issues in the wireless network may force cameras to reduce their sampling frame-rate of their resolution for transmission, reducing achievable accuracy via perception of low-quality images. Second, any pipeline that evaluates spatial overlap between cameras must receive time-synchronised camera streams or incorporate intelligent ingest mechanisms that synchronizes camera frames on receipt at the edge to detect common/unique stimuli/RoI across multiple spatially overlapped cameras. In this thesis, I show that (i) bandwidth adaptive mechanisms deployed at the camera can alleviate wireless network challenges by creating multi-resolution images optimised for the available bandwidth, retaining high pixel fidelity for areas of interest and downstream DNN task accuracy, and (ii) intelligent ingest mechanisms at the edge can deal with natural network jitter and still align received camera frames in a best-effort manner for canvas construction to recover detection of common objects across spatially overlapped cameras.

5. **Detecting Spatial Overlap over Cameras in Motion:** In this thesis, I limit the analysis of spatial overlap across multiple cameras to stationary cameras that have pre-determined camera poses and fixed spatial overlap in the sensing field. I conduct the analysis of spatial overlap as a one-time offline bootstrapping process before the deployment of the real-time perception pipeline at the edge. Detecting dynamic spatial overlap over cameras in motion such as cameras

mounted on drones is extremely challenging. The perception pipeline will have to evaluate (i) the physical location of the camera in a 3D geometric space, (ii) potential cameras in the physical vicinity of each camera that might share spatial overlap, (iii) the pose of each camera, and (iv) actual spatial overlap between cameras, *all* as a pre-processing step within milliseconds over transient camera feeds. I leave this as an open research problem on resource-constrained edge devices.

6. **Costs of Pre-processing and Post-processing:** Canvas construction over multiple RGB and event sensors involves a non-trivial amount of pre-processing in the form of (i) Criticality estimation of detected stimuli (ii) Ingest of high volumes of event streams and processing thereof into framed representations for canvas construction and DNN inference (iii) Evaluation of common objects observed from spatially overlapped cameras. The post-processing of object detections from the DNN and mapping of the predictions back to the original camera frames is also a sizeable component of the perception pipeline as it must include mappings for those RoI that were excluded from the canvas frame due to spatial overlap between multiple cameras. Both classes of processing require fine-tuned mechanisms designed for accuracy *and* speed so as to allow non-blocking canvas construction and reduce GPU idle time as much as possible. I show how such mechanisms designed for nuanced pre-processing and fast post-processing can contribute to the end-to-end optimisation of the perception pipeline.

7. **System Scalability:** The goal of canvas-based perception pipelines is to push the envelope on the number of cameras that can be processed at a single edge device, thereby increasing processing throughput, without impacting DNN task accuracy. Through my work, I shall detail multiple spatial, temporal, and adaptive optimisations of the perception pipeline that enable this vision. However, devising intelligent system strategies for scalability of support for multiple

cameras using a multi-device edge infrastructure (thereby affecting achievable system throughput) is an independent challenge. Especially considering a network of cameras and multiple supporting edge devices, such scalability challenges may be addressed by classical distributed systems optimisation techniques such as offloading, workload sharing, and workload stealing with appropriate measures in place to ensure that spatially overlapped cameras are serviced by a single edge node. In this thesis, I address system scalability with on-demand activation of additional edge nodes with intelligent camera cluster offload. I leave the exploration of more nuanced mechanisms for distributed workload processing for future work.

1.3 Thesis Statement

Previous sections highlight the opportunities and challenges that arise from criticality-aware canvas-based processing at the edge. In this dissertation:

> I show that it is feasible to significantly enhance the accuracy and throughput of real-time machine perception tasks involving multiple visual sensing streams processed concurrently by a computationally-constrained edge device by applying the concept of criticality to (i) determine the priority of different regions of the sensing field and constituent objects with both spatial and temporal sensitivity, and then (ii) perform criticality-driven differentiated processing of such regions/objects spanning multiple video streams, from both conventional RGB cameras and emerging neuromorphic vision sensors.

This dissertation establishes the thesis through the following steps:

1. **Introducing a Spatial Degree of Freedom:** It first addresses the notion of *what* subset of stimuli the system must select for fine-grained perception by introducing a spatial degree of freedom to the stimuli or critical RoI, evaluated

by the perception pipeline *MOSAIC*, detailed in Chapter 2. *MOSAIC* identifies critical stimuli/RoI in the sensing field using motion-based background subtraction and decomposes each input frame from M concurrent camera streams into "a bag of tiles" or regions of the frame, capturing the detected RoI at different camera-specific scales or "levels of zoom". As multiple tiles at different scales may capture the same object, *MOSAIC* filters and selects only those tiles that completely capture all the RoI at their appropriate scale, ultimately spatially multiplexing these filtered tiles via Inverse 2D Bin Packing onto the limited volume of the canvas frame for DNN inference.

2. **Adapting to Real-World Dynamics:** Next, it addresses (i) real-world wireless bandwidth and latency constraints, and (ii) the reality that individual cameras have varying workloads over time, to introduce the *Resource Adaptive MOSAIC (RA-MOSAIC)* pipeline in Chapter 3. *RA-MOSAIC* extends the notion of criticality aware spatial multiplexing of limited GPU resources introduced in Chapter 2 to accommodate real-world sensing dynamics. First, at the camera sensor, a *bandwidth-adaptive* method applies differential down-sampling to create mixed-resolution individual frames that preferentially preserve resolution for critical ROIs, before being transmitted to the edge node. Second, at the edge, multi-resolution video streams received from multiple cameras are decomposed into multi-scale "bags of tiles" and spatially packed using a novel *workload-adaptive* bin-packing strategy into a single 'canvas frame'. Notably, the canvas frame itself is dynamically sized such that the edge device can opportunistically provide higher processing throughput for selected high-priority tiles during periods of lower aggregate workloads.

3. **Exploring a Temporal Degree of Freedom:** Next, it addresses the notion of *when* the system re-selects already perceived objects to monitor changes in its physical environment to introduce the *JIGSAW* pipeline Chapter 4. *JIGSAW* introduces a temporal degree of freedom using a novel age utility-based

weighted scheduler to preferentially deprioritize recently seen unique objects for inclusion on a canvas frame constructed over multiple spatially-overlapped cameras. To balance between accurate perception of the state of the physical world and delay induced by the age metric in the processing pipeline, *JIGSAW* selectively discards multiple instances of RoI over time to optimize the novel streaming accuracy metric [86] which accounts for localisation latency in addition to object detection accuracy. Additional discussions are presented on spatiotemporal schedulability bounds and Earliest Deadline First based bin-packing algorithms.

4. **Leveraging Event Cameras for Low-Power Sensing:** Finally, it evaluates the adoption of newer classes of sensors such as neuromorphic event cameras to extend the notion of canvas-based processing to high-volume event streams, enable low-power sensing, and recover sensing capabilities in low or poorly lit areas. Chapter 5 introduces *TANDEM*, a canvas-based perception pipeline at the edge that intelligently orchestrates and fuses $\{$feature-rich, $O(ms)$ latency, $O(W)$ power consumption$\}$ standard camera streams in gray-scale with $\{$low-dimension, $O(\mu s)$ latency , $O(mW)$ power consumption$\}$ event camera streams to compensate each other *on demand*. Multiple such pairs of Frame+Event streams thread the gap between energy efficiency and sensing fidelity to jointly provide feature-rich sensing perception with high spatial gray-scale image resolution and high temporal event resolution. *TANDEM* leverages insights from *MOSAIC* in Chapter 2 to spatially multiplex stimuli/RoI from fused Frame+Event representations for DNN inference, providing multiplicative increase in throughput with no cost to DNN task accuracy. Additional discussions are presented on how event cameras can act as native motion-based criticality estimators to inform spatiotemporal multiplexing strategies introduced by *JIGSAW* in Chapter 4.

Chapter 2

Introducing a Spatial Degree of Freedom: *MOSAIC*

In this chapter I discuss the introduction of a spatial degree of freedom in the perception pipeline as a cornerstone of this thesis and describe the components of the Canvas-based Processing paradigm over multiple camera input streams at the resource constrained edge. I evaluate spatially-multiplexed canvas based processing against two applications: (i) drone-based person detection and (ii) moving vehicle-based automatic license plate detection which relies on Optical Character Recognition to read license plates. I show that extracting objects out of their context for canvas-based object detection and optical character recognition retains achievable DNN task accuracy while simultaneously facilitating multiplicative gains in processing throughput (i.e. camera capacity) at the edge.

2.1 Extracting Critical Stimuli/RoI for Canvas-based Processing

A growing number of real-time applications of machine perception (e.g., drone-based worker safety detection and vehicular tracking by street-mounted cameras) involves the execution of DNN-based inference over *multiple, concurrent, high-resolution*

multimedia sensor data streams on a resource-constrained edge device. Real-time edge-based execution of such perception tasks remains challenging, given the rapid increase in both DNN model complexity and data size/resolution (e.g., 4K image frames). For example, an NVIDIA Jetson TX2 [109] device can process a maximum of only $\sim$2 frames per second (FPS) when executing the YOLOv5L6 (191 layers, 47M parameters) object detector at FP16 precision on a 1280×1280 image frame. Conventional approaches for overcoming this throughput/latency challenge include either (a) the use of smaller, less accurate DNN models executing on lower resolution data (e.g., a less complex 300×300 SSD model[98] can achieve a processing throughput of 10-15 FPS on the TX2 with TensorRT optimizations) or (b) the use of more expensive, higher-resourced edge devices (e.g., the Jetson Xavier[109]). Such approaches impose an unfavorable *cost/complexity vs. accuracy* tradeoff.

To reduce this computational overhead, recent works have proposed either (a) *criticality-aware processing* approaches, where only selective high-value portions of individual image frames are processed with higher attention or fidelity [48, 156, 150] or offloaded for DNN task inference [161] or (b) *selective computation approaches*, where certain DNN layers are simplified [26] or skipped [142]. These approaches, however, do not consider scenarios where *multiple* sensor streams, with dynamically varying scene characteristics (e.g., object sizes and speeds) share the same computational resources on an edge device and must be processed *concurrently*. Liu et. al. [96] introduce a multi-camera system design which extracts regions of interest from camera frames for batched DNN inference, but force all extracted regions to be downsampled to a $\{$uniform, smaller$\}$ spatial dimension irrespective of criticality or appearance to take advantage of GPU speed-up, facilitating higher throughput at the cost of DNN task accuracy.

To address this gap, I introduce *MOSAIC*, a criticality-driven perception pipeline that optimizes the edge-based execution of DNN-based inferencing tasks over multiple image streams. Through this optimization, *MOSAIC* provides a **multiplicative increase in the *per-stream throughput*** that an edge device can sustain **without**

sacrificing task accuracy. *MOSAIC* can benefit a wide variety of applications including, but not limited to, (i) city traffic monitoring, where multiple camera streams monitoring an intersection are processed on a single edge node, and (ii) urban event monitoring, where multiple drones' camera streams are processed by a single handheld edge control unit, as illustrated in Figure 1.2 in Chapter 1.

MOSAIC's central concept involves the notion of a *Canvas*, defined intuitively as the maximum size of an input frame (say C) that a DNN, executing on a GPU-equipped edge device, can consume while ensuring that the processing throughput remains above a minimal FPS threshold. I call these reduced-resolution frames, which the GPU can keep up with, *canvas frames*. The challenge of concurrently processing multiple (say M) camera streams can then be framed as one of spatially *packing* or fitting high-priority regions from M independent image frames into a C-sized canvas frame. Conceptually, *MOSAIC* replaces the baseline mode of independent, *sequential* DNN execution on each individual frame with a *spatially-multiplexed* paradigm, where M frames (one from each camera sensor) are processed concurrently.

MOSAIC's design addresses two key challenges with this paradigm: (a) identifying and extracting critical regions from frames with very low overhead, so as to sustain high throughput, and (b) allocating the shared canvas space equitably across critical regions with dynamically varying object characteristics. To pack the canvas frame appropriately, *MOSAIC* efficiently *creates a spatial degree of freedom* and decomposes an input frame into multiple tiles (sub-regions), defined at different scales or "levels of zoom" that collectively represent objects/RoI of different sizes. The selected tiles that contain faithful representations of the objects or Regions of Interest (RoI) are then *inverse-2D bin packed* [35] onto a canvas frame, thereby providing an $M - fold$ boost in processing throughput as illustrated in Figure 2.1(b)). *MOSAIC's* packing is carefully designed to ensure that (a) tiles are proportionally resized based on their criticality while ensuring that the underlying object sizes conform to application-defined spatial bounds, and (b) the tiling process, invoked

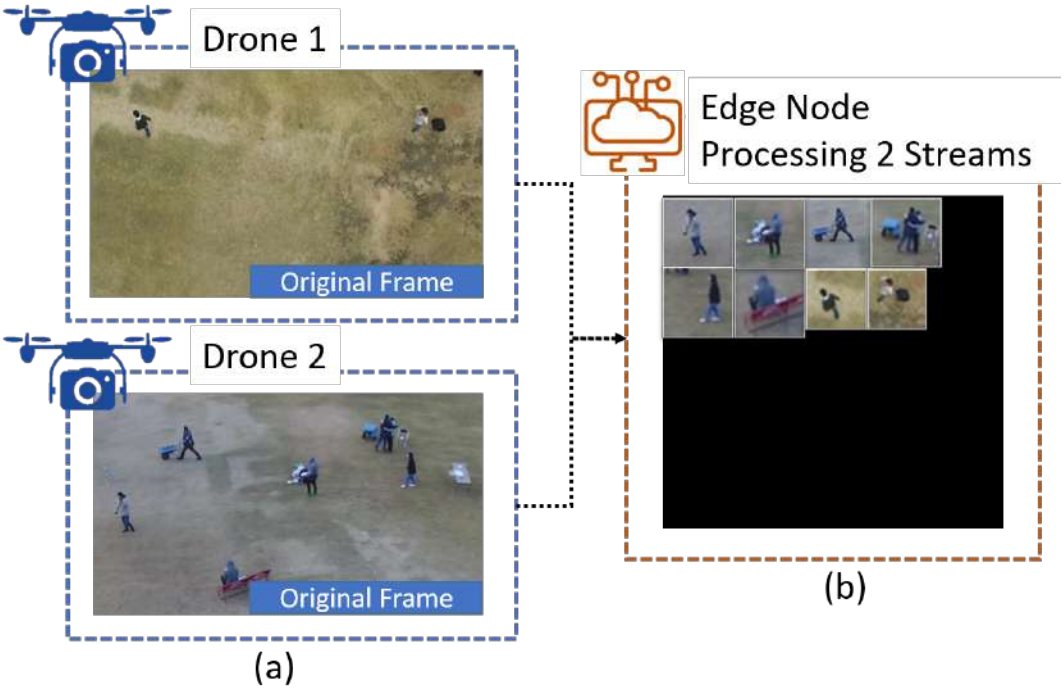

Figure 2.1: *MOSAIC* Overall Functionality: (a) Input frames captured by cameras (b) Packing tiles from multiple images onto a single canvas frame (image not to scale).

intermittently, is very low-overhead.

Via benchmark datasets for two distinct applications - *Okutama-Action* [20] for drone-based pedestrian detection and *UFPR-ALPR* [83] for license plate recognition (*LPR*), I demonstrate *MOSAIC's* ability to significantly improve the throughput-vs.-accuracy tradeoff for diverse machine perception tasks across diverse camera settings.

2.1.1 *MOSAIC*: Key Contributions

In this chapter, I make the following key contributions:

- *Criticality-Preserving Canvas-Based Processing:* I develop a 3-stage innovative pipeline, called Mosaic Across Scales (MoS), to dynamically fit a variable number of critical regions, from multiple camera input images, into a single canvas frame: (i) a multi-scale tiling mechanism that uses a Min-Cost Min Set Cover algorithm [7] to select an appropriate *minimal* subset of all possible tiles, for any

given camera input image, that both capture all likely objects of interest while assuring such objects the largest area possible in the eventual canvas frame, (ii) a Min-Max optimization technique to differentially resize such individual selected tiles based on their computed criticality values, and (iii) a Differential Evolution Algorithm [101] with geometric constraints for 2D Inverse Bin Packing [35] all selected tiles (across M distinct images) onto a canvas frame. For pedestrian detection, MoS suffers a negligible ($\leq 1\%$) accuracy loss when compared to the low-throughput, sequential FCFS processing of images; it also achieves an 8% increase in accuracy when compared to a strategy of uniformly downsampling and packing all $M = 6$ images onto a single canvas frame. For LPR, *MOSAIC* can pack tiles containing vehicles differentially from $M = 3$ images to achieve Optical Character Recognition (OCR) Character Error Rate (CER) of $\leq 33\%$, far superior to a baseline uniform resizing approach, that results in 100% CER (complete loss of readability) even when frames are processed individually ($M = 1$) i.e. FCFS processing of each input frame at the same DNN inference image size (or canvas frame size) as *MOSAIC*.

- *Real-time MOSAIC Implementation and Performance Gains:* I implement *MOSAIC* using the NVIDIA Jetson TX2 as the edge device. The developed *MOSAIC* system uses a combination of intermittent full-frame object localization and continuous motion tracking to support low-overhead, approximate extraction of critical regions. Table 2.1 details the throughput for packing images from M =6 cameras for pedestrian detection, and $M = 3$ images for LPR. For both the pedestrian detection and LPR applications, *MOSAIC* achieves 5x (batch size=1, 18 FPS per camera) and 4.75x (batch size=4, 23 FPS per camera) increase in processing throughput, compared to an FCFS baseline that processes each input frame sequentially.

MOSAIC on Jetson TX2	Batch Size = 1		Batch Size = 4	
	FPS Per Camera	CFPS	FPS Per Camera	CFPS
Pedestrian Detection 6 Cameras	18 FPS	108 FPS	**23 FPS**	**138 FPS**
License Plate Recognition 3 Cameras	18 FPS	54 FPS	23 FPS	69 FPS

Table 2.1: *MOSAIC* System Performance on Jetson TX2

2.2 Motivating *MOSAIC*

MOSAIC's primary objective is to pack high-priority or critical regions from M discrete camera streams onto a single canvas-frame to increase frame processing throughput for each stream, without sacrificing inference task accuracy. To this end, I first explore the target applications and basic principles that underpin *MOSAIC's* key design decisions.

2.2.1 Target Applications

I envision that a wide variety of applications could benefit from the increased throughput afforded by canvas-based processing–examples include surveillance, counting or detection scenarios where both the camera sensor and the edge node could be stationary or in motion. Consider an edge node deployed on a mobile robot or handheld control unit that processes autonomous drone camera feeds for aerial monitoring in factory/retail worker safety monitoring, search-and-rescue, wildlife poaching or urban crowd control applications. For all of these scenarios, infrastructure costs can be reduced with shorter response times without compromising perception fidelity by allocating computing resources *non-uniformly* and *selectively*, to only relevant portions of the captured image frames. I assume that each camera sensor monitors a distinct, non-overlapping physical region, although *MOSAIC* can likely be further optimized to take advantage of any spatial overlaps, discussed in

Section 2.6.

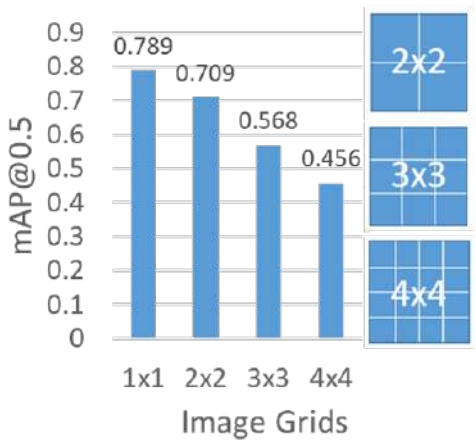

Figure 2.2: *MOSAIC* First Principles: Evaluating Accuracy vs. Grid Size (Uniformly Packing)

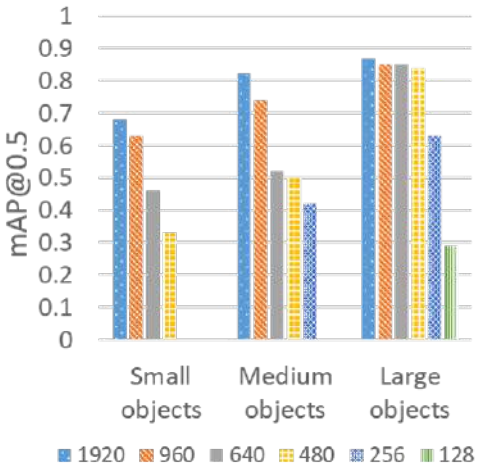

Figure 2.3: *MOSAIC* First Principles: Object Detection Accuracy vs. {Object size, Resolution}

2.2.2 First Principles

Increasing Throughput by Packing Multiple Images

I first study the implications of spatially packing multiple input image frames uniformly (without any criticality awareness) into an image grid, as a means of increasing processing throughput. Figure 2.2 plots the object detection accuracy (computed as *mAP* or mean average precision), as a function of the number of such image frames packed, for frames from the Okutama-Action[20] that have a native resolution of 3840×2160. For all studies in this work, canvas size is set as 640×640, adopted from reported benchmarks for YOLOv5 [138], the model of choice in this work. As seen in Figure 2.2, the higher the number of elements (distinct frames) in the grid, the smaller the resulting region (pixels) allotted to each input frame, and the lower the object detection accuracy. Intuitively, due to uniform downsizing, smaller objects become progressively smaller and less distinguishable, to the point of loss of detect-ability. There is therefore an opportunity for the system to increase inference accuracy by providing high-priority regions of the original frame a larger spatial

share of the canvas.

Factors Impacting Object Detection Confidence

To further delve into why packing an image into a smaller sized grid (i.e., reducing its overall pixel count) results in lower detection accuracy, I additionally analyze the variation in object detection accuracy over different object sizes. I experimentally observe that object detection accuracy values degrade either due to a reduction in an object's size (a natural consequence of downsizing an image frame to fit into a smaller grid) or a loss in object resolution (greater pixelation). As shown in Figure 2.3, as image sizes increase, object detectability increases leading to an increase in average object object detection accuracy. However, an excessive increase in the object size to the point of greater super-resolution images can also result in a saturation or loss of detectability, due to artifacts such as excessive pixelation, implying that there is a limit beyond which scaling up an object does not help. Overall, image downsizing has a *variable* impact: small and medium sized objects stand to receive the largest confidence boost from the increase in resolution and size, while the detectability of large objects remains relatively robust to resolution loss.

2.2.3 *MOSAIC's* Design Choices

At the edge, *MOSAIC* dispenses with the straightforward approach of using uniform downsizing to pack multiple image frames into a single canvas frame. Instead, *MOSAIC* seeks a differential downsizing strategy, which seeks to *reduce the disparity in pixel areas* corresponding to likely objects (varying in size and spanning multiple input images) embedded in the canvas frame prior to DNN inference. To reduce such disparity (which improves the accuracy for smaller objects without disproportionately penalized larger objects), *MOSAIC* first uses variable-sized tiles to optimally capture pixel regions with likely objects, and then spatially resizes such tiles within acceptable bounds to fit within the target canvas frame.

2.3 *MOSAIC* Design Overview

I now describe the criticality-aware adaptive processing performed by *MOSAIC* at the edge, as illustrated in Figure 2.4. *MOSAIC* alternates between two modes of operation (1) Mosaic Across Scales (**MoS**) and (2) Periodic Stabilization (**PS**), both of which interact with the Memory Function to support *MOSAIC* objectives: (i) extraction of critical regions from an input frame and (ii) bin-packing of these critical regions into a canvas frame for DNN inference. The Periodic Stabilization (PS) operation initialises and intermittently refreshes *MOSAIC* pipeline by running full-frame batched DNN inference on all camera streams to detect class-specific critical objects. For each camera, PS localises the newly detected objects, identifies stationary objects, and updates the object tracker maintained in *MOSAIC's* Memory Function to correct all tracker uncertainties accumulated from the previous round of MoS operation. PS also prompts *MOSAIC* to examine the recently observed RoI and object size distributions per camera to compute a set of camera-specific RoI scales. These RoI scales are updated in the Memory Function and are used to instruct the next round of MoS operation on the sizes of expected RoI in each camera stream. For each camera stream, the MoS operation first estimates the locations of critical RoI with motion-based background subtraction and updates the camera-specific object tracker. MoS extends the philosophy of criticality by tiling the input image at each identified RoI scale, identifying the tiles that contain all the RoIs maintained by the tracker, and resizing the tiles (and by extension the RoI contained within the tile) to larger dimensions as much as possible to boost object detection accuracy. This approach effectively "spatially channels" the limited computation resources available at the edge to the critical regions. Of course, such resizing is a zero-sum game overall that should preferentially enlarge smaller, distant RoI; this objective is complicated by the reality that a single tile in an input frame can contain RoI of different sizes (e.g., a mix of nearer and distant objects).

At a high-level, the MoS process must balance two conflicting objectives: (a) the

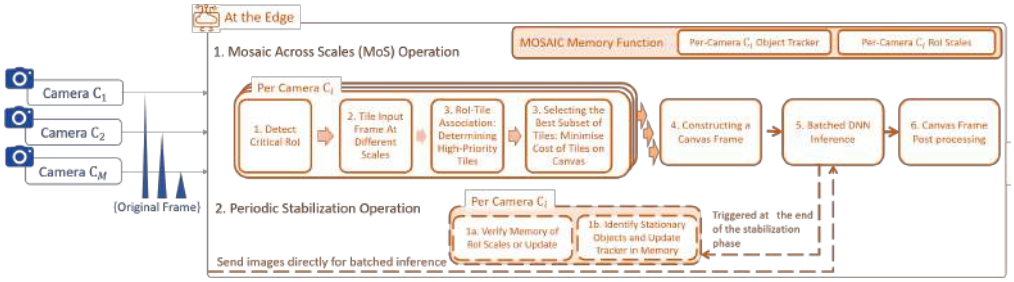

Figure 2.4: *MOSAIC* Block Diagram of Sub-Components Operating at the Edge

RoI should ideally consume a large-enough fraction of a tile such that tile resizing does not eventually result in a dramatically smaller object–*this criterion favors smaller tiles*, and (b) the total number of tiles to be fitted into a canvas should ideally be minimized, so as to allow each tile a larger share of the canvas–*this criterion favors larger tile sizes*. MoS creates a canvas frame through a number of innovative sequential steps, described next, that collectively balance these two objectives by:

- First, extracting a minimal set of tiles (at each of the camera specific RoI scales maintained in the Memory Function) within each image frame to encompass the likely ROI maintained by the tracker
- Then, 2-D bin "inverse" packing such multi-scale tiles (i.e., ensuring all relevant tiles are packed) to construct a canvas frame as a composite image of such tiles; this canvas frame is then sent to the DNN model for inference.

2.3.1 Periodic Stabilization (PS) Mode

The PS operation initializes and refreshes the entire *MOSAIC* pipeline with class-specific objects, their locations, and object size distributions evaluated through batched full-frame DNN inference on all the incoming camera streams for the entire PS duration. The objective of the PS operation is to address two significant challenges faced during MoS operation. First, cameras may observe a variety of object size distributions based on the physical installation of the camera and the observer-object distance. For example, a camera mounted on a lamp-post may observe a mix of large foreground and smaller background objects of interest, whereas a camera mounted on

a drone may observe uniformly small object distributions. Object size distributions may also be dynamic over time, for example, a camera mounted on a drone may observe varying object sizes as it increases/decreases its flying altitude. To best capture critical RoI of different sizes, MoS tiles the input frame at different scales and evaluates which subset of tiles contain critical RoI from the input frame. MoS relies on the object size distribution evaluated during the PS operation to understand how many scales to use for such a tiling step and what tiling dimensions each scale must adopt. Second, the MoS pipeline estimates critical RoI by performing background subtraction, which captures the RoI where objects are likely in motion and may miss stationary, halted, or occluded objects of interest. MoS similarly relies on the detected object locations obtained by the PS operation to update a camera-specific object tracker with the locations of objects that might be missed by the background subtraction based estimation of critical RoI in the input frame. The periodicity and duration of PS is a configurable parameter and describes the expected average rate of change in the observed object size distribution. However, the PS periodicity parameter and overall *MOSAIC* achievable throughput is inversely related: a shorter PS and longer MoS period provides larger throughput gain, as the PS period effectively processes frames at full resolution (without any spatial multiplexing gain). After completing the full frame detections for the PS duration across all camera streams, *MOSAIC* is triggered to refresh the per-camera RoI scales and object tracker, described next.

(1a) Calculate Per-Camera RoI Scales: To determine the camera-specific RoI scales and their dimensions, this function first collects the object size distribution observed during the PS operation and assesses if any of the detected objects are overlapping with each other or are in close proximity. Such objects may be detected as a single RoI during the MoS operation so this function adds the minimum enclosing rectangle for such overlapping/nearby objects to the observed object distribution. The resulting object size distribution is then clustered using a KNN clustering model, with an elbow-detection method to determine the appropriate value of k (the number

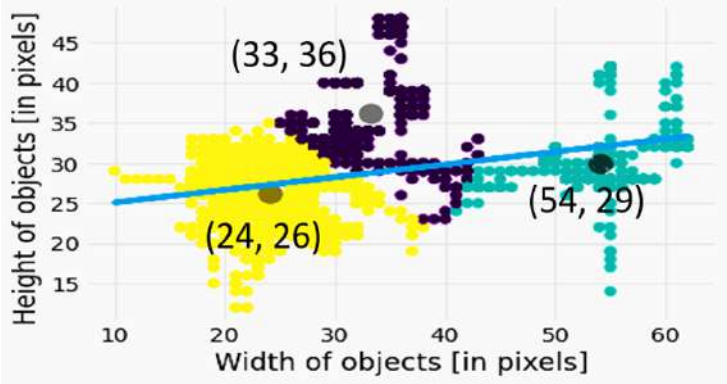

Figure 2.5: *MOSAIC* Calculating Per-Camera Scales: Object Size Distribution & Clusters in Okutama-Action Drone Sequence 1.1.8

of distinct clusters). This k value then determines the number of scales that MoS will employ for that specific camera, with the centroid of each of these k clusters determining the size of the corresponding tile. MoS takes the larger value between each centroid's x and y coordinate for each scale i.e. $max(x_{centroid}, y_{centroid})$ as the tiling dimension for that scale, rounded to the nearest multiple of 32 for computational efficiency, to best represent objects whose size falls within this cluster.

Figure 2.5 illustrates this process by plotting the object distribution observed over all frames by Camera 1 from the Okutama-Action dataset; clustering identifies 3 clusters with centroids $(24, 26)$, $(33, 36)$ and $(54, 29)$ respectively. MoS consequently determines the 2 scales of tiles to be 32×32 and 64×64 respectively. MoS also introduces a *catch-all* tile, $\sim 1.5x$ larger than the largest determined tile (in this example, this results in an additional 96×96 tile), to accommodate the possibility of subsequently observing objects/RoI larger than anything seen during the preceding PS operation.

(1b) Identify Stationary Objects and Update Per-Camera Tracker in Memory Function: This function initializes and refreshes a per-camera Kalman Filter Centroid-based tracker which maintains the most recently observed location and state for all objects in the camera FoV. This state refers to whether the object is "active" (i.e. in motion) or "stationary" and is computed by observing the distance travelled by each centroid during the entire PS duration.

2.3.2 Mosaic Across Scales (MoS) Mode

MoS begins operation on the camera frames ingested immediately after the PS operation completes. As seen in Figure 2.4, Steps 1 through to 4 in the MoS operation are carried out on each camera stream in parallel until the relevant tiles containing critical RoI from input frames across *all* camera streams are determined. These relevant tiles are then bin-packed into a canvas frame for DNN inference and post-processing.

(1) Determining Critical Regions of the Input Image: This MoS sub-component assembles a list of mask bounding boxes where critical RoI are estimated to be present in the input frame. To achieve this, the ingested input frame is compared to the previous frame for background subtraction which detects critical RoI that might contain objects in motion. MoS updates the mask bounding boxes and the camera object tracker with these detected RoI locations for matching with known RoI tracks and assigns the status of the updated tracks as "active". This method is robust to new objects that enter or existing objects that move in the camera Field of View (FoV). Among the tracks not updated by the RoI from the current input frame, MoS assesses the status of each track. For tracks marked "stationary" and missed by the background subtraction-based motion estimation, MoS retrieves their last known locations from the tracker memory to add to the mask bounding boxes. For the remaining unmatched tracks, MoS assumes that the object that was previously in motion has either come to rest, crossed paths and jumped tracks with another object, or occluded by another object and therefore merged with the other RoI. In all these cases, MoS assigns the status of the track as "last-seen" and includes all the unmatched track locations to the list of mask bounding boxes for the current input frame. Such "last-seen" objects are maintained in memory until they are reactivated or until the end of the MoS operation period. This is done to avoid missing objects that might have come to a halt at its last known location for the remaining duration of MoS operation. Lastly, in the event of detected camera ego-motion (e.g., for a

Figure 2.6: *MOSAIC* Determining High-Priority Tiles: Tiles at different scales, capturing objects with different "goodness"

camera mounted on a moving drone), this sub-component also performs camera motion compensation which detects frame keypoints, matches the descriptors of the current and previous input frame, and calculates the new location of all known tracks in the current input frame by modelling the camera motion as a partial 2D affine transformation.

(2) Tiling an Input Image: MoS then generates a bag of tiles, at each scale dimension maintained by *MOSAIC* Memory Function with a configurable overlap parameter that determines the tile strides.

(3) Determining High-Priority Tiles: The generated bag of tiles at k different scales may have no objects/RoIs, partial views of objects, or completely contained objects. MoS next determines the subset of such tiles that adequately capture the critical regions, while balancing the two conflicting objectives mentioned earlier. MoS constructs a spatial quadtree from all the generated tiles in the bag of tiles and then uses the assembled mask bounding boxes from Step 1 to perform an intersecting bounding box search. All tiles that intersect with each of the masks are then evaluated for "goodness of fit of the mask" in the individual tile. Figure 2.6 illustrates such a selection, where MoS chooses tile (A) over tile (B), as tile B only partially captures the human object.

More formally, MoS assigns a mask to a tile if it satisfies two distinct "goodness" criteria. First, the tile must capture a significant portion ($\geq 95\%$) of the mask,

both in width and height; this ensures that the object is sufficiently visible (and not unduly cropped) to be successfully detected by the downstream object detector DNN. Second, the mask:tile height ratio must lie in the range $(0.5, 0.9)$–i.e., the object must not encompass either too small or too large a fraction of the tile's area. The lower bound increases the probability that the object will retain larger dimensions after being resized and 2D bin-packed onto the canvas; masks smaller than half the tile's dimensions are effectively assigned to tiles of smaller scales. Conversely, the upper bound (0.9) ensures that relatively large-sized objects are preferentially assigned to tiles of larger scale (which can likely accommodate additional objects), while also minimizing the risk of undue cropping due to an inaccurate mask. Via this process, MoS curates a filtered bag of tiles, each containing faithful representations of estimated objects at the appropriate scale, thereby assuring that such objects will remain reasonably sized (with higher likelihood of successful DNN detection) when the canvas frame is composed.

(4) Selecting The Best Subset of Tiles: A number of different combinations of tiles might "cover" (in a set-theoretic sense) all masks/objects that need to be included onto the canvas for inference. However, to promote efficient (less-redundant) packing onto the canvas frame, it is imperative to select only those tiles that are not only likely to preserve object dimensions in the canvas but also that minimize 'wasted pixels' (intuitively, the total number of pixels representing the background or other irrelevant objects, as well as objects captured in multiple tiles). This dual optimization process can be conceptualized as bin-in-a-bin packing problem, where MoS must not only ensure that all objects are 'covered' by the chosen tiles, but also that the chosen tiles generate the lowest count of 'wasted pixels' possible. I perform such selection by using a greedy approximation (due to the problem being intrinsically NP-Hard) of the Min-Cost Min-Set Cover (*MCMSC*) Algorithm summarized in Algorithm 2.

Intuitively, the MCMSC algorithm selects those tiles that together minimize the cost of wasted background/non-object pixels appearing in the tile while ensuring that each object of interest is part of at least one tile that satisfies the goodness criteria

Algorithm 1 Greedy Min Cost Min Set Cover Algorithm

Subset of chosen tiles $S\ Universe = m_1, m_2, ...m_m$ Set of M masks $Tiles = t_1, t_2, ...t_n$ Set of N tiles that may contain one or more assigned masks $Costs = c_1, c_2, ...c_n$ Set of costs for N tiles $S \leftarrow \emptyset$

while $S! = Universe$ **do**

if $\mid t_i - S \mid \geq 0\ \ \&\ \ (c_i / \mid t_i - S \mid) > 0$ **then** $Subset \leftarrow \min(c_i / \mid t_i - S \mid)$ minimize the number of tiles containing the same mask and minimize the additional cost to the canvas associated with adding an additional tile

$S \leftarrow S \cup Subset$

mentioned earlier. To achieve this goal, MoS assembles two distinct views of the mask-to-tile relationships in the filtered bag of tiles. In the first view, for each mask, MoS assembles a set of acceptable tiles that best capture that mask. To calculate the the cost of including the tile into the canvas, the second view concurrently consolidates all the masks assigned to an individual tile. A Min Set Cover over the first view ensures that all masks (possible object-related pixels) are included in the canvas, while a Min Cost over the second view selects the minimal subset of tiles that the canvas must accommodate.

This unique formulation has several advantages. First, objects that can feasibly be mapped to multiple tiles typically appear in only one (or at most two) tiles in the eventual canvas, reducing the likelihood of false positives in the post-DNN non-maximal suppression (NMS) based inference step. Second, multiple objects that occur in close spatial proximity are usually represented by a single larger-scale tile (with reduced wasted pixels), instead of being represented by multiple individual smaller-scale tiles. In particular, the min set cover step (line 7) in the optimization chooses the tile containing the most number of masks not already present in the canvas, while the min cost step (also in line 7) takes into account the cardinality or the unique number of masks added to the canvas by a single tile and the associated cost of including the tile in the canvas. At the end of this step, MoS computes the final, optimal subset of tiles for each individual camera sensor frame. For each tile in the chosen subset, MoS also computes a spatial sizing bound and an *elasticity factor* (based on the combination of object/tile's scale and application requirements), as the

range and amount of resize that can be tolerated during canvas frame construction. I empirically observe that this size bound is *scale-dependent*: tiles of different scales can tolerate different ranges of resizing, outside of which objects either become intolerably small or suffer from excessive pixellation on enlargement, severely impacting DNN inference accuracy. Small objects are, in fact, especially sensitive to drastic differences in spatial resizing.

(5) Constructing a Canvas Frame: After the previous step, MoS has effectively curated a set of tiles, their spatial sizing bounds, and elasticity factor, say $\mathcal{ST}_i$, with heterogeneous dimensions for each input image frame (F_i). To now pack sensor data from multiple image sensors onto a canvas frame of a given dimension, MoS needs to determine the modified dimensions of all tiles across all of these M subsets– i.e., $\forall$ tile t : such that $t \in \bigcup_{k=1}^{M} \mathcal{ST}_k$ such that they can be packed onto a single canvas frame with defined dimensions. This can be generalised as an Inverse Bin Packing Problem [35] where given a defined set of items and bins, the algorithm must converge on the minimum perturbation to the item-size vector such that all the items can be packed into the prescribed number of bins.

MoS approximates such an optimization by using a *computationally-fast* Differential Evolution Algorithm [101] (a form of meta-heuristic optimization) with a Min-Max Optimization objective function that minimises the largest dimensions obtainable within its defined bounds such that the combined area of all included tiles is less than the canvas frame area. The optimizer also takes in the elasticity factor as the tile weight, and spatial sizing bounds limiting the size of each individual tile. The Differential Evolution Algorithm also takes an Equality Penalty function which monitors if the number of packed tiles is lower than the number selected for packing by MoS. If so, the next generation of tiles is required to monotonically decrease or "squeeze" the size of each tile (based on the elasticity factor) to pack all the tiles onto the canvas frame, while ensuring that the amount of "squeeze" does not violate the tile's defined lower spatial bounds. In essence, the solver maximises each tile's size within its acceptable bound, while attempting to 2D bin pack *all* the tiles onto the

canvas frame. In the event that all tiles are "squeezed" to their individual minimum permissible size, MoS relaxes the lower bound and notifies the user of a possible loss of accuracy with the advice to assign fewer camera streams to the edge node (a form of "admission control"). Upon solver convergence, MoS obtains not only the resized *dimensions* for each tile, but also the *canvas position* where it must be packed for optimal fit.

(6) Postprocessing: As mentioned earlier, the DNN then executes the inference task on the resulting canvas frame, consisting of the re-packed, re-positioned tiles. Finally, MoS also maintains the tile→bin spatial mapping for each input image included in the canvas, and uses this to perform post-inference coordinate translation of all detected objects (and their locations) back to the original input frame. After translation, MoS also executes a general Non-Maximal Suppression (NMS) step on the translated boxes for each original input frame to remove any double-counting for objects that might have appeared more than once (inside tiles of different scales) on the canvas. Any other downstream vision processing pipeline is applied thereafter on the post-processed objects.

2.4 System Design

The PS and MoS pipelines are deployed at the edge, as visualised in Figure 2.4. During the PS operation, input frames from M cameras are batched and directly sent for DNN inference. By default, *MOSAIC* executes the PS operation over 10 frames, with a a periodicity of 30 seconds. This setting (see Section 2.5.2 for a deeper analysis) adequately balances the need for sufficient frames to detect and classify stationary vs. moving objects with the desire to minimally impact *MOSAIC's* overall achievable throughput. During the MoS operation, input frames from M cameras are received and concurrently evaluated by the MoS pipeline to construct a canvas frame, as visualised in Figure 2.7. MoS receives the i^{th} input frames, $f_m^i \forall m \in M$ cameras, and constructs a canvas frame C_i from the chosen subset of tiles across

all M frames. The DNN inference task is configured for a batch size of b which allows the pipelined canvas construction of the next b frames $(f_i, \ldots, f_{i+b})$ onto canvas frames $(C_I, \ldots, C_{i+b})$ during DNN inference on the previous batch. The achievable throughput on *MOSAIC* is thus a function of both the PS and MoS modes of operation. I empirically establish that on a Jetson TX2, a TensorRT-optimised YOLOv5s model achieves reasonable accuracy and inference latency of $\sim$ 170msec on 640×640-sized canvas frames with batch size=4 (thereby achieving $\approx$ 6FPS per camera or cumulatively 24 FPS). *MOSAIC* adopts this configuration and determines the maximum value of M or the maximum number of cameras that can be supported at a single edge device for a chosen application and set of camera streams for the MoS operation. This value of M is constrained by two key factors: (i) the canvas construction time for b canvas frames from M camera input frames must not exceed the DNN inference time ($\sim\leq 170ms$) to allow seamless pipelined execution, and (ii) all chosen tiles from M cameras must not violate their application-dependent and scale-dependent spatial sizing bounds when packed onto the canvas frame. *MOSAIC* adheres to both constraints to choose the most appropriate value to M for the application to ensure that *MOSAIC always* achieves 24 FPS for *all* M camera input streams during the MoS phase. However, *MOSAIC's* overall achievable throughput is reduced due to the periodic PS operation that processes all incoming camera frames sequentially and without modification. With a batch size $b = 4$, the PS operation adds a delay of $10 \times M/24$ seconds for processing 10 stabilization frames across M camera input frames under these default settings. For example, for $M = 6$ camera streams, batch size $b = 4$, the PS operation adds an overall processing latency of $\sim$2.5 seconds, resulting in an overall achievable *MOSAIC* throughput of 23 FPS across both PS and MoS phases.

Evaluation Platform: I evaluate *MOSAIC* on the NVIDIA Jetson TX2 [108], a representative edge device equipped with a 256 CUDA-core PASCAL GPU, and an ARMv8 multi-processor architecture supporting both a dual-core NVIDIA Denver 2 CPU and a quad-core ARM Cortex A57 MPCore CPU.

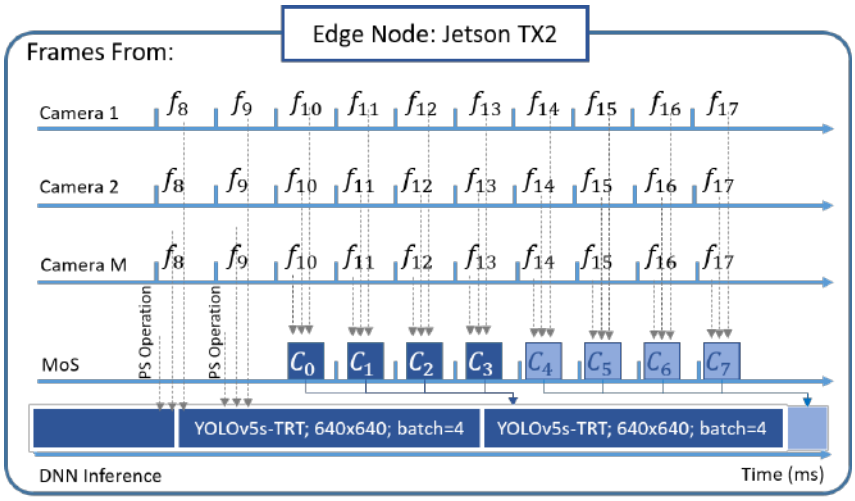

Figure 2.7: Conceptual design of MOSAIC's MoS pipeline at the Jetson TX2 Edge Node

Benchmark Datasets: I evaluate *MOSAIC* using two benchmark datasets for two distinct applications - *Okutama-Action* [20] for drone-based pedestrian detection and *UFPR-ALPR* [83] for license plate recognition. The Okutama-Action dataset comprises 43 drone sequences at 4K (3840×2160) resolution encoded at 30FPS, yielding 54664 and 14210 images for training and testing respectively. The UFPR-ALPR dataset similarly comprises 90 video sequences at 1920×1080 resolution encoded at 30FPS, yielding 3600 and 1800 frames for training and testing respectively. I consider each video sequence as a distinct camera and use combinations of M cameras without duplication of camera input streams for fair comparisons. *MOSAIC* constructs canvas frames with "person" objects for the Okutama-Action dataset and vehicle objects of classes {car, motorcycle, bus} for the UFPR-ALPR dataset; for UFPR-ALPR, the downstream OCR pipeline then performs LPR on the detected license plate bounding boxes.

Evaluation Model: For object detection on the canvas frames, I employ a TensorRT-optimised YOLOv5s model, an edge-compatible pretrained model with 7.2M parameters and 16.5 GFLOPs. It is pre-trained using the MS COCO dataset [92] and fine-tuned on the selected datasets for greater sensitivity to occluded, unseen, and small-sized low-resolution objects.

Evaluation Metrics: To evaluate possible gains in perception accuracy in the pedestrian detection application, I report the mean average precision of the model at an IoU threshold of 0.5 –i.e. mAP@0.5, and report the inference latency i.e. FPS_C and Cumulative-FPS CFPS = $M \times FPS_C \times b$, where FPS_C is the throughput achieved for canvas frames of size C, with batch size b, and M cameras per canvas frame. For the license plate detection application, I evaluate the downstream Optical Character Recognition (OCR) quality through Character Error Rate (CER) metric. CER employs the Levenshtein distance metric to calculate the minimum number of single-character changes (i.e. insertions, deletions, or substitutions) required to change the predicted string of characters to the groundtruth, averaged by the number of characters in the groundtruth. The lower the CER rate, the better the OCR performance, with 0 indicating perfect recognition.

Evaluation Baselines: I compare *MOSAIC's* performance against three baselines:

1. FCFS: Received frames resized to the canvas frame dimensions and sent for batch DNN inference without any spatial modification–in effect, this is identical to *MOSAIC's* behavior during the PS phase.

2. Uniform-M: Denoted as Uni-M where M signifies the number of images packed onto a single canvas frame. Uniform-M divides a canvas into equal number of grid rows and columns and assigns each input image to a single cell in the grid. Uniform-M also determines which methodology among grid, horizontal, or vertical stacking of M input images creates the best grid structure such that each cell affords its corresponding input frame the lowest possible downsize ratio when compared to its original dimensions.

3. Batched Inference of Extracted RoI: I compare *MOSAIC's* performance against prior literature on edge-based multi-camera systems which include a notion of criticality in their evaluation [94, 61, 95]. These works extract regions of interest from the camera frame (similar to *MOSAIC*) but resize all regions to a uniform spatial dimension to leverage GPU acceleration yielded by batched inference.

2.5 *MOSAIC* System Evaluation

I first evaluate the validity of the fundamental hypotheses, that criticality-aware processing of multiple input frames with the MoS methodology helps improve the throughput vs. accuracy tradeoff for diverse object distributions, camera settings, and applications. Next, I compare canvas-based processing employed by MoS with batched inferencing methods employed by recent works. For deeper insights into the benefits of MoS, I also conduct ablation studies on how *MOSAIC* performs varies with different parameter settings.

2.5.1 *MOSAIC* System Performance

In general, I expect throughput and accuracy to be inversely related: increasing M (the number of camera frames being packed into a single canvas frame) should provide an M-fold increase in throughput but result in lower mean object detection accuracy. Figures 2.8 & 2.9 plot this accuracy vs. throughput, for different *MOSAIC* configurations and baseline approaches, for M varying between $\{1,\ldots,6\}$ for Okutama-Action and $\{1,\ldots,3\}$ for UPFR-ALPR, respectively on a canvas frame size C of 640×640; the throughput is plotted per camera by dividing the overall processing throughput (i.e. CFPS) by M. As detailed in Section 2.4, the maximum number of cameras M that can be packed onto a canvas frame depends on the canvas construction time, object sizes/scales, and spatial sizing bounds of the resulting subset of tiles. As the objective in UFPR is to eventually perform OCR on the detected license plate objects, it imposes stricter spatial sizing bounds for each tile which limits the number of cameras for this dataset to $M = 3$. This is unlike the pedestrian detection application of Okutama-Action which permits more relaxed spatial sizing bounds (even though it contains smaller ($\sim 64 \times 64$) person class objects), allowing $M = 6$ cameras to be successfully packed onto a canvas frame. For batch size $b = 1$, *MOSAIC* takes on average $\sim 9ms$ and $\sim 13ms$ ($\sim 36ms$ and $\sim 52ms$ for batch size $b = 4$) to build a canvas frame from $M = 3$ and $M = 6$ cameras respectively,

well within the inference deadline of 170ms. Both constraints on M thus satisfied, I compare Uniform-M for each application accordingly.

Pedestrian Detection Application: In Figure 2.8, for batch size $b = 1$, FCFS (where each image is processed independently and sequentially–i.e., $M = 1$) offers the highest accuracy $\sim$0.79 but suffers from very low throughput $\leq$3FPS per camera in a 6-camera setting. On the other end of the spectrum, Uniform-6 (where $M = 6$ images are uniformly compacted into the canvas frame) offers the highest throughput ($\sim$ 19 FPS) per camera, but suffers a significant 8% loss in accuracy to $\sim$ 0.71. In contrast, *MOSAIC* offers a significantly more favorable tradeoff as M is varied. With $M = 6$: (a) compared to FCFS, MOSAIC-6 experiences only a negligible $\sim< 1\%$ accuracy loss but achieves over 500% (5x) higher throughput; (b) compared to Uniform-6, MOSAIC-6 achieves significantly higher accuracy $+7.8\%$ with minor 0.04% reduction in throughput (Uniform-6 = 19 FPS; MOSAIC-6 = 18 FPS). This slight reduction is due to the PS operation running for 10 frames every 30 seconds. Similar throughput-vs.-accuracy tradeoffs can be observed with batch size $b = 4$ with the exception that the Uniform-6 and MOSAIC-6 methods both achieve 24 FPS and 23 FPS per camera respectively across 6 cameras (144 and 138 Cumulative FPS), with *MOSAIC's* method achieving significantly higher accuracy by $+8\%$ over Uniform-6.

License Plate Recognition Application: In Figure 2.9, for batch size $b = 1$, Uniform-1 suffers CER=100% (complete OCR failure) which indicates that uniformly downsizing a single image to fit onto a 640×640 canvas causes severe, catastrophic pixelation of the license plate. On the other hand, MOS-1 retains the pixel resolution of the high-priority vehicle object to completely recover OCR capabilities downstream within a reasonable OCR CER value of 15%. *MOSAIC* further pushes the envelope on OCR quality downstream by also packing tiles likely containing vehicle RoIs from $M = 2$ and $M = 3$ images onto a canvas frame to similarly recover OCR ability downstream within reasonable OCR CER bounds (29% and 33% respectively), while also achieving a much higher system throughput

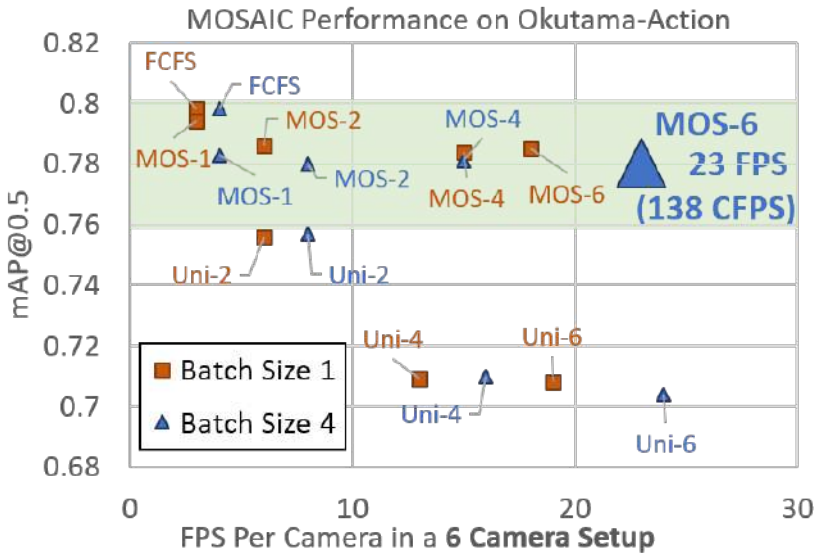

Figure 2.8: *MOSAIC's* Performance for Pedestrian Detection on the Okutama-Action dataset

of 23 FPS per camera for the 3-camera system (Cumulative FPS=69).

Comparative Study with Batched Processing of Individual Objects: I compare *MOSAIC's* performance against recent priority-aware processing works [94, 61, 95] which extract high-priority regions of interest from a camera frame and perform batched inference on the selected regions. Compared with sequential frame inference, batched inference processes multiple inputs at one time to improve parallelism and better utilize the computing capacity of the GPU. However, unlike *MOSAIC's* canvas-based execution where tiles of different sizes can be packed into one canvas frame, batched inference requires all input tile images to be the same size. I therefore compare the mean accuracy from the pedestrian detection application to illustrate the benefits of *MOSAIC's* differential tile resizing strategy in the MoS pipeline at the edge. I first obtain the average execution times of a 640×640 canvas frame and batched tile images at different batch size/image size combinations through system profiling of the Jetson TX2. The best tile image size for batching-based execution is then determined online as the largest tile image size to run all input tiles and finish no slower than executing one canvas frame. The curated subset of tiles from MoS are resized with padding to this input image size for batched inference. This

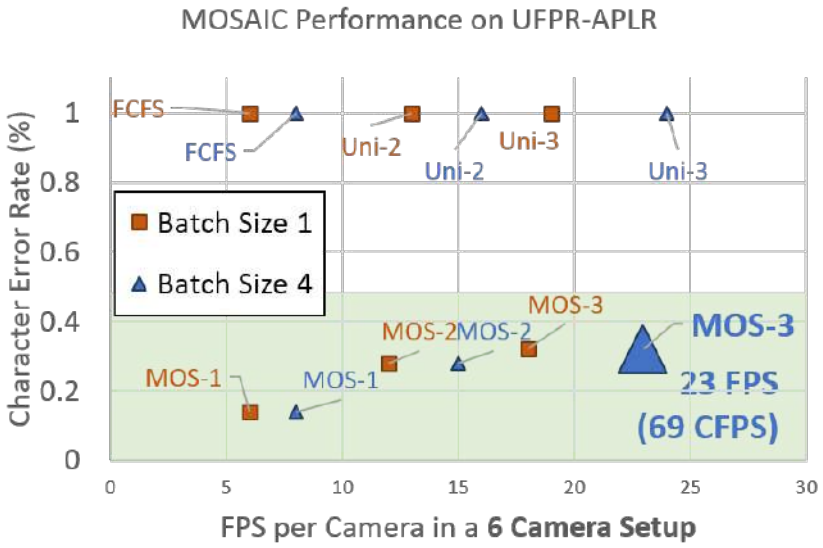

Figure 2.9: *MOSAIC's* Performance Automatic License Plate Recognition on the UFPR-ALPR dataset

way, batched inference achieves the same throughput as *MOSAIC's* canvas-based inference for batch size $b = 1$.

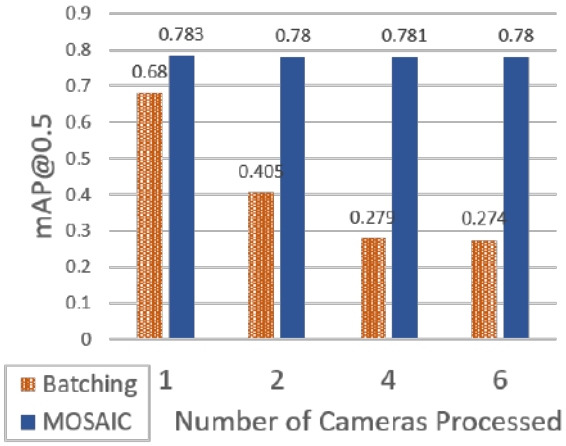

Figure 2.10: *MOSAIC* vs. Tile Batching: mAP vs Number of Cameras Processed for Tile Processing Techniques at the Edge

Figure 2.10 visualizes the mAP score of processing B=$\{1, 2, 4, 6\}$ input frames at the same time, using the Okutama-Action dataset. I note that both canvas-based execution and batching-based execution achieve good accuracy when the number of input frames is 1. As M increases, the accuracy achieved by batched inference drops, with *MOSAIC's* canvas-based method significantly outperforming (a 3-fold higher

accuracy when processing 4 or 6 frames concurrently) the batching-based method as B increases. The accuracy for batching-based execution drops much faster due to the requirement that *all* images must be the same size, which leads to a dramatic loss of accuracy especially for the small objects. On the other hand, the non-uniform sizing supported by the canvas-based method is able better to preserve the accuracy of smaller objects. Compared with batching-based execution, canvas-based execution achieves a dramatic 3-fold accuracy improvement when evaluating $M = 4$ and $M = 6$ concurrent camera input frames.

2.5.2 Ablation Studies

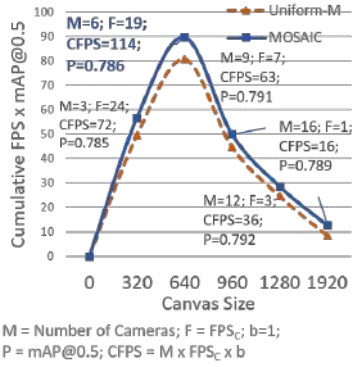

Figure 2.11: Chosen Canvas Size vs Cumulative FPS $\times$ mAP@0.5

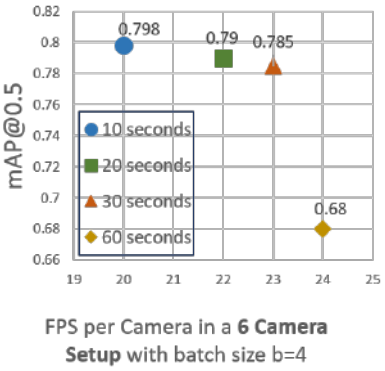

Figure 2.12: *MOSAIC's* Performance Gains vs. PS Periodicity

MOSAIC's Performance Gains on Different Canvas Sizes

Given that the maximum number of cameras, M, that can be packed onto a canvas frame is limited by the object distribution observed by the cameras and their spatial sizing bounds, it stands to reason that increasing the canvas frame size C could allow for a higher value of M (thereby increasing throughput) and arguably, higher accuracy if all tiles assume the maximum dimension within their sizing bound. However, larger canvas frame sizes $C \geq 640 \times 640$ impose greater computation loads and incur higher inference latencies per canvas frame. This increase in canvas frame inference latency reflects in a lower reduced cumulative inference throughput,

i.e. CFPS = $M \times FPS_C \times b$, where FPS_C is the throughput achieved for canvas frames of size C with batch size b. Figure 2.11 illustrates such a throughput-accuracy tradeoff by plotting a joint "*throughput-accuracy*" metric (defined as Cumulative FPS $\times$ mAP@0.5) vs. canvas frame size for the Okutama-Action dataset. I conclude that *MOSAIC* consistently outperforms the uniform resizing and packing or Uniform-M baseline, regardless of the canvas frame size C. I also note that the preferred canvas frame size of 640×640 provides the highest throughput-accuracy gains (15.4%), compared to all other canvas frame sizes.

MOSAIC's Performance Gains vs. PS Periodicity

I have established that the PS operation periodicity impacts the achievable *MOSAIC* throughput due to the processing latency incurred by the FCFS-based DNN inference on all camera input frames from M cameras. A shorter PS periodicity would yield higher number of FCFS processed frames and therefore more accurate class-specific object detections, while negatively impacting *MOSAIC's* overall achievable throughput. A longer PS periodicity might, conversely, incur higher tracker failure for "last-seen" and lost or unmatched tracks, in turn creating additional tiles (which may or may not contain objects) that MoS will need to spatially pack, thereby reducing task accuracy. Figure 2.12 plots the resulting relationship between PS periodicity, *MOSAIC* throughput, and mAP@0.5 for the Okutama-Action dataset. I show that a PS period=30 secs appropriately balances the dual requirements of high throughput (23 FPS) and high object detection accuracy (78.5%), while a PS period=60 secs increases throughput by only 1 FPS while suffering a steep ($\sim$ 10%) mAP drop.

2.6 Discussion

Canvas Construction under Network Bandwidth Constraints: In a real-world wireless network with bandwidth constraints, a compute-capable camera platform can choose to downsample the camera streams to a lower resolution for transmission.

This enables the payload of the camera stream to be right-sized for the available bandwidth, thereby allowing the camera stream to be delivered to the edge within the expected latency. Such camera operation would be beneficial to the overall camera-to-edge system efficiency, especially if the edge will use *MOSAIC's* pipelines to downsize frames during processing. However, uniformly downsampling the camera streams to lower resolutions before transmission will have a disproportionate impact on the achievable DNN task accuracy due to image quality degradation. Figure 2.13, shows two scenarios of bandwidth constrained frame transmission from the camera to the edge modelled by downsampling video streams for person detection (on the Okutama action dataset): (i) downsampling images to fit in canvas frame dimensions 640×480 and (ii) downsampling frames to 320×227 i.e. to half the canvas size 640×640. The second case suffers DNN task accuracy degradation of 14% when evaluating frames on a First Come First Serve basis. With *MOSAIC's* pipelines, packing $M = 6$ such downsampled camera frames yields a DNN task accuracy degradation of 15.6%. This indicates that more nuanced down-sampling techniques which focuses available bandwidth/pixels onto regions of interest will preserve object resolutions for efficient canvas-based processing. Prior work has shown how criticality-aware techniques for differential downsampling of input frames (e.g., MRIM [146]), can outperform uniform downsampling by helping preserve object details for edge-based vision inferencing tasks. Such cameras can also implement lightweight multi-object detectors and trackers, such as FastMoT [153], to identify class-specific objects of interest even in challenging environmental conditions such as low-lighting and fog. *MOSAIC's* MoS operation can not only co-exist with such criticality-aware transmission, but in fact can benefit from such on-camera processing by effectively eliminating the intermittent *PS* phase. I address such design changes in Chapter 3.

Canvas Construction under Network Latency Constraints: Real-world wireless deployments also result in variable network transfer latency, arising out of various factors such as interference from other wireless devices, physical distances, and

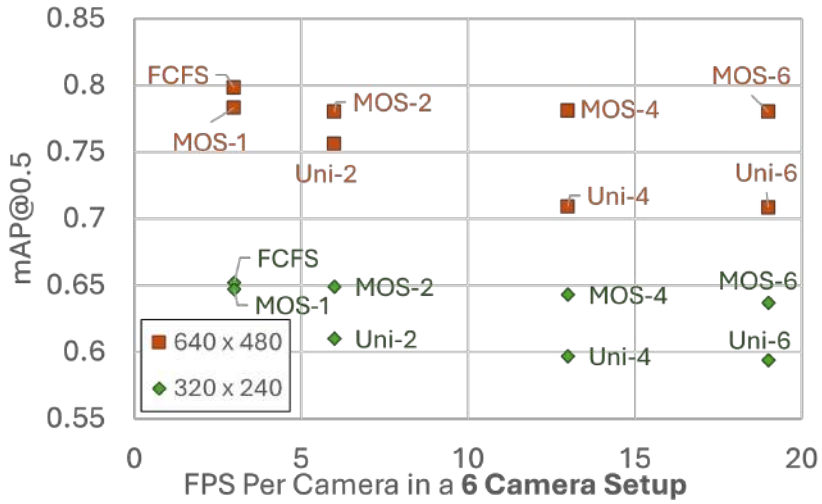

Figure 2.13: Impact of Bandwidth Constraints on *MOSAIC's* Performance for Pedestrian Detection on the Okutama-Action dataset (batch size 1)

outdated hardware. To understand how possible differences in network latency impact the dynamics and efficacy of canvas construction, consider a deployment where three cameras transmitting frames to an edge node suffer variable latency resulting in frames being delivered at the edge every (84 ms, 71 ms and 40 ms), illustrated in Figure 2.14. For simplicity, I limit the analysis of this study to a batch size $b = 1$, where a single canvas frame is constructed from received frames. As soon as camera frames are received at the edge, *MOSAIC* proceeds to construct canvas frames and schedule its DNN inference to *prevent GPU idle time* as shown in Figure 2.7 and discussed in Section 2.5.1. By the time the second canvas frame is ready to be built, camera frames from all three cameras have been received at the edge. However, this is not ideal as this canvas processed the *first* frames from Camera 1 and 2 and the *second* frame from Camera 3 *on the same canvas frame*. While this does not impact achievable object detection accuracy, I note that not only is there is a delay in the perception of the physical world, but such processing also cannot guarantee a uniform perception *across cameras*. This could be challenging in scenarios where the system needs to synchronise frames across cameras for canvas construction, such as in spatially overlapped camera deployments. Another outcome observed here is that consecutive frames from the same camera (Camera 3) are

mapped to the same canvas frame C5 (indicated with red arrows in Figure 2.14).

Such a pipeline design introduces significant lag between the physical world and the perception pipeline and is wasteful of DNN computation when multiple frames from the same camera are processed onto the same canvas frame. This indicates that (i) there may be value in introducing a mechanism to introduce a temporal degree of freedom in the system which allows it to pick just the latest instance of the same object if more than one instance available at the edge has not yet been inferred upon (ii) intelligent ingest pipelines are required at the edge to ensure that the perception pipeline is always keeping up with the physical world. I discuss the adoption of such mechanisms in Chapter 4.

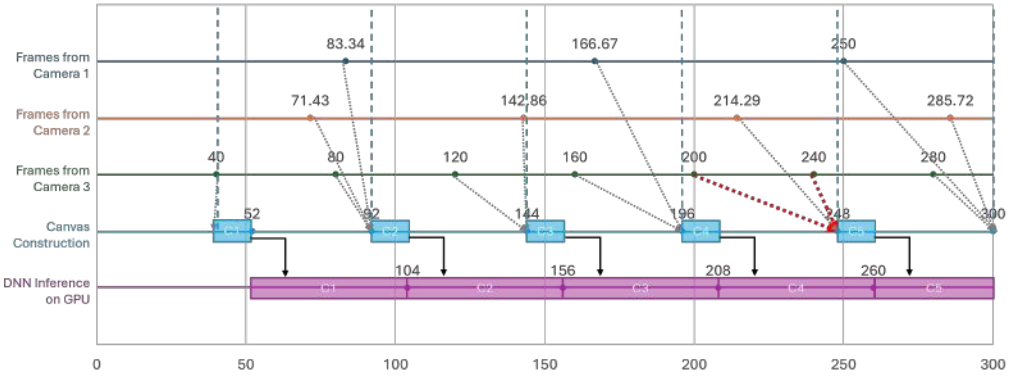

Figure 2.14: Impact of Wireless Latency on Canvas Construction - Scenario 1: Consecutive Frames from a Camera are Mapped to the Same Canvas

Need for Camera-Specific Tiling Scales: I have earlier described (Section 2.3) *MOSAIC's* mechanism of determining custom scales for objects observed by each camera. Such custom scales clearly introduce additional complexity, and one may indeed question if an alternative approach, of using pre-defined scales across all cameras, may not be preferable. To address this question, I conducted additional studies where *MOSAIC* applied one of 3 fixed scales across *all* cameras in the Okutama-Action dataset. The resulting accuracy (for $M = 6$), however, exhibited a sharp impairment, dropping from 78% (using camera-specific scales) to 32%. Manual inspection of cameras in both Okutama-Action and UFPR-ALPR indeed reveal significant variations in object sizes, based on camera positioning and perspective,

and explain the importance of choosing the correct custom scales for MoS tiling.

Supporting Heterogeneity in DNN Model and Execution Platforms: The *MOSAIC* pipelines utilize an unmodified YOLOv5 model for the task of object detection, an underlying primitive in many vision-based applications. This conscious design decision enables *MOSAIC* to adopt different and/or better DNN models such as YOLOv8 [139] in future iterations. A similar design decision was employed for the choice of the Jetson TX2 [108] as the evaluation platform, more capable GPUs can offer lower DNN inference latency and thereby higher system processing throughput. I assert however, that the multiplicative increase in throughput of *MOSAIC* will remain valid so long as the DNN inference time of a single canvas frame containing tiles from M input frames is less than the inference time of M camera full input frames in parallel.

Adapting to Multi-camera Systems with Established Spatial Arrangements: The *MOSAIC* pipelines make no apriori assumptions on the physical arrangement of the cameras and expect each camera stream to be processed independently for canvas construction. However, most modern multi-camera systems have established spatial arrangements with pre-determined spatial overlap between cameras to prevent blind spots in the sensing field. This presents an opportunity to further optimize the system to reduce the computational load at the edge by (i) recognizing object representations that are common to multiple camera streams, and (ii) filtering or picking a representation for each unique object for inclusion onto the canvas frame. Such nuanced processing would intuitively boost DNN task accuracy due to (i) inclusion of larger or more well-defined unique object representations on the canvas frame, and (ii) reduced volume of tiles included on the canvas frame, allowing each tile to maximize its spatial sizing bound. I discuss system design adaptations for such multi-camera deployments in Chapter 4.

Chapter 3

Adapting to Real-World Dynamics: RA-MOSAIC

In Chapter 2 I introduced MOSAIC presented *Canvas-based Processing* as a mechanism to spatially multiplex RoI from multiple cameras onto a single frame of smaller volume for fast DNN inference using a single GPU task, illustrated in Figure 3.1 (c). MOSAIC exploits a spatial degree of freedom by (i) decomposing RoI or objects from multiple camera frames into independent tiles or patches, and (ii) resizing (or squeezing) the tiles, by 2D Inverse Bin Packing, onto a fixed-size canvas frame. This composite constructed "canvas frame" is sized to maintain a sufficiently high throughput on the resource constrained edge device. While *MOSAIC* was shown to provide impressive gains ($\geq 4.75\times$) in throughput, the work has a few key limitations in terms of adaptability to varying network and workload conditions. First, *MOSAIC* did not directly address network-level issues, such as variable transmission latency (jitter) or bandwidth limitations between a pervasive sensing device and the edge node. Second, *MOSAIC* assumes steady workloads (i.e. object/RoI volume) received from each camera at the edge, and does not address real-world temporal variability in object volume distributions, which could have a significant impact on the canvas construction process.

In this chapter, I further enhance MOSAIC's Canvas-based processing paradigm

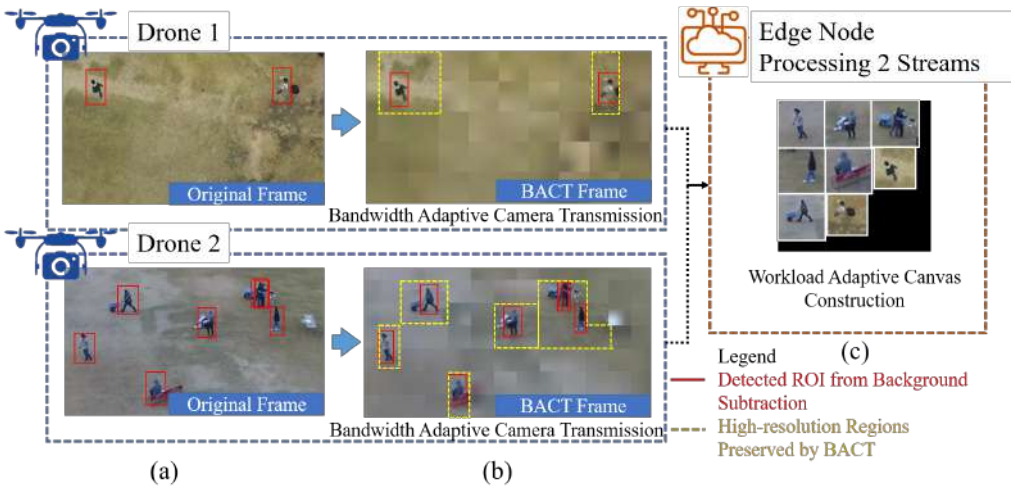

Figure 3.1: *RA-MOSAIC's* Overall Functionality (Best Viewed in Colour): (a) Input frames captured by the camera with High-Priority Object Proposals (red boxes) (b) Criticality-aware Mixed-Resolution Camera Transfer (BACT) (preserving resolution in yellow boxes) (c) Packing RoI tiles from multiple images onto a single canvas frame that adapts to the ROI workload (image not to scale).

to introduce Resource Adaptive MOSAIC (*RA-MOSAIC*), which enables network bandwidth-aware and workload-specific adaptations to further improve the accuracy-vs-throughput tradeoff. First, *RA-MOSAIC* symbiotically extends the notion of criticality to optimize the camera→edge wireless data transmission overhead while remaining compatible with the downstream Canvas-based processing paradigm. The key idea is for each camera sensor to first perform fast but *approximate* determination of ROI in a captured image (see Figure 3.1(a)) and then differentially preserve pixel count (resolution) while transmitting areas of higher criticality and reducing the resolution for less critical regions (illustrated in Figure 3.1(b)) within a bandwidth-determined pixel budget. Second, *RA-MOSAIC* dispenses with MOSAIC's implicit assumption of steady object detection workloads (from each camera stream) and incorporates the reality of possible temporal variations, even over shorter timescales, in object {size, appearance, volume} distributions/workloads within each camera's image stream. Multi-resolution frames and ROI bounding boxes received from multiple camera streams are processed in parallel to decompose the camera frame into a bag of tiles that capture the RoI at different scales or "levels of zoom". Selected tiles that represent each RoI at its appropriate scale are assembled for canvas frame

construction. I propose a novel Bloom2Squeeze variant of the 2D Inverse Bin Packing algorithm [35] that relaxes the fixed size constraint of the canvas frame. Instead, the canvas frame is allowed to "bloom" (or conversely "squeeze") depending on the workload (i.e. the number of tiles selected for canvas construction), (illustrated in Figure 3.1(c)). *RA-MOSAIC* retains an upper bound constraint of the original canvas dimension i.e. 640×640 to ensure that it maintains the minimum required throughput *in the worst case*. This variation of canvas-based processing enables *RA-MOSAIC* to adjust its target throughput rate to fluctuations in workload (a capability missing in *MOSAIC*). For example, during periods of low ROI in several cameras, *RA-MOSAIC* can reduce the canvas frame size to process image streams faster without adversely affecting the vision task accuracy.

By using two benchmark datasets for two distinct applications - *Okutama-Action* [20] for drone-based pedestrian detection and *UFPR-ALPR* [83] for license plate recognition (*LPR*), I demonstrate how *RA-MOSAIC's* bandwidth-aware operation and workload-aware operation can further improve the throughput-vs.-accuracy tradeoff for machine perception.

3.0.1 *RA-MOSAIC*: Key Contributions

In this chapter, I make the following key contributions:

1. *Bandwidth-Adaptive Criticality-Preserving Optimized Sensor→Edge Frame Transfer:* At each camera, I propose a lightweight, fast Bandwidth Adaptive Camera Transmission (BACT) approach that preserves higher resolution (pixel count) for likely critical portions of the input frame and downsamples those areas deemed less critical. BACT effectively transmits a *mixed-resolution image*, much smaller in size than the original raw frame, while ensuring that crucial details are preserved for the subsequent *RA-MOSAIC* stages of tiling and bin packing at the edge. I show that BACT accuracy is fairly robust and $\sim$40% higher than a baseline approach of uniform downsampling for pedestrian detection, even when

the transmitted frame size is restricted to only 30% of the original captured frame.

2. *Workload-Adaptive Canvas-based Processing:* By adapting the canvas-based processing paradigm according to the available workload, the Workload Adaptive Canvas Construction (WACC) pipeline of *RA-MOSAIC* can opportunistically adopt smaller canvas frame sizes to incur lesser computation latency and thus provide a higher processing throughput at no cost to task accuracy. *RA-MOSAIC's* novel Bloom2Squeeze variant of 2D Inverse Bin Packing allows a 22.2% gain in processing throughput with lower workloads (during the "bloom" operation) presented by $M = 3$ cameras when compared to MOSAIC's naive fixed-size canvas construction technique. Even at higher workloads presented by $M = 6$ cameras (during the "squeeze" operation), *RA-MOSAIC* presents a 11.11% gain in processing throughput when compared to MOSAIC's baseline performance, due to the elimination of *MOSAIC's* Periodic Stabilisation phase at the edge.

3. *RA-MOSAIC Implementation and Performance:* I describe how both resource-adaptive mechanisms in *RA-MOSAIC* design work in tandem to further improve the system performance for *both* throughput and accuracy in bandwidth-constrained wireless deployments. I show that the joint operation of *RA-MOSAIC's* BACT and WACC pipelines provide a 22.2% gain in throughput and a simultaneous 15% gain in accuracy in bandwidth-constrained environments which support lower workloads from $M = 3$ cameras. At higher workloads with $M = 6$ cameras, *RA-MOSAIC* provides a simultaneous 11.11% gain in throughput and 14.3% gain in accuracy over the naive MOSAIC baseline. Compared to prior state of the art methods for "batched inference" over extracted RoI [94, 61, 95], *RA-MOSAIC* achieves a 26.9% gain in accuracy even when processing RoI tiles extracted from a single $M = 1$ camera stream in a bandwidth-constrained environment. As the number of cameras $M \geq 4$ increases, *RA-MOSAIC* outperforms batched inference by improving mean accuracy by $\geq 60\%$.

3.1 Motivating Dynamic Adaptations in Canvas-based Processing

RA-MOSAIC's primary objectives are to (i) facilitate *Bandwidth-Adaptive* wireless transmission of criticality preserving multi-resolution images from the camera to the edge node within a pre-determined pixel budget, and (ii) enable *Workload-Adaptive* construction of canvas frames that temporally adapts the canvas frame dimensions to the object distributions perceived across all M cameras to opportunistically reduce processing latency. I hypothesize that extending the concept of criticality to the camera→edge transmission phase can help preserve higher resolutions and pixel nuances for RoIs, especially when operating over bandwidth-constrained wireless networks. Such differentiated resolution can in turn boost object detection accuracy that enables finer-grained tasks such as Optical Character Recognition to achieve lower character error rates. Further, I also hypothesize that enabling opportunistic adoption of smaller canvas frame sizes would increase the achievable processing throughput on average, thus providing gains over MOSAIC in both *accuracy and throughput*. I first present the target applications and basic principles that support these hypotheses.

3.1.1 Target Applications

I envision a variety of camera-based edge-deployed applications that could benefit from *RA-MOSAIC's* resource adaptive canvas-based processing. For example, consider a construction site that is monitored by a fleet of drones which predict equipment safety and supply chain efficiency; each drone observing discrete sections of the site, while wirelessly transmitting the camera streams to an edge node for application-specific inference, illustrated in Figure 1.2 in Chapter 1. Alternatively, envision a retail shopping floor with a fleet of drones providing personalised service to customers by helping them navigate to aisles that stack items they are looking for,

or picking up items that they might have forgotten. In these cases, *RA-MOSAIC* can reduce infrastructure costs and deliver faster response times by adapting to the available bandwidth on the factory or shopping floor, while also dynamically adapting to the workload to reduce processing latency. It is assumed that each camera sensor monitors a distinct region of the physical world which does not overlap spatially with regions observed by other cameras, though further optimization of *RA-MOSAIC* could potentially exploit any spatial overlaps (see Section 3.5).

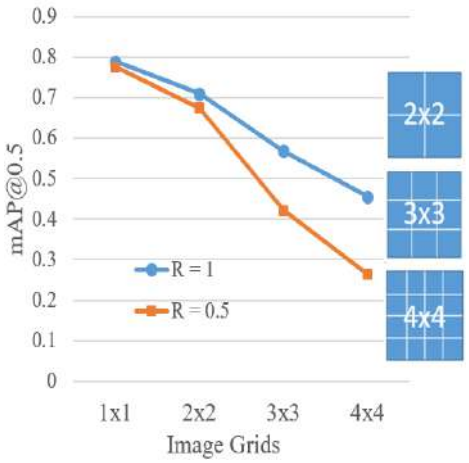

Figure 3.2: *RA-MOSAIC*: Evaluating mean accuracy over grids of uniformly packed images

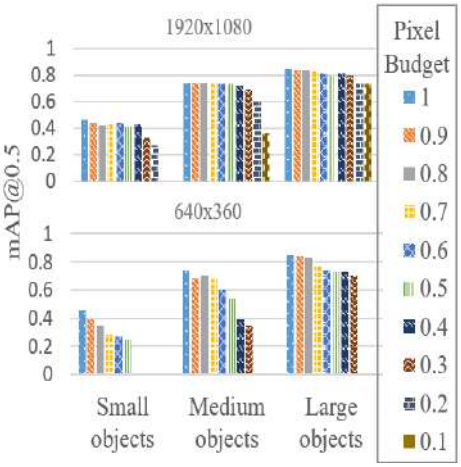

Figure 3.3: *RA-MOSAIC*: Increasing object detection confidence by increasing object resolutions

3.1.2 Bandwidth Constraints and Naive Packing of M Frames

In a real-world system with bandwidth constraints, camera sensors should ideally avoid high transmission overheads by downgrading an image's resolution before transmitting, especially if the edge will downsize frames during processing. I model such constraints through a *pixel budget*, which is a represented as a ratio r between image resolution/size during frame transfer and image resolution used for DNN inference. As discussed in Chapter 2, a naive method to achieve higher throughput across all M cameras is to downsample M frames uniformly into a canvas frame grid without any notion of criticality. To understand the impact of bandwidth constraints on a naive uniform packing baseline, I fix the canvas frame size to 640×640 as it

is the largest empirically determined canvas frame size that can be inferred upon by a YOLOv5s (a representative edge-scale model) on a Jetson TX2 edge device with latency low enough to meet the minimum throughput requirement of 20 FPS. Figure 3.2 plots mean average precision (mAP) of object detection accuracy, as a function of M, for frames from the Okutama-Action[20] dataset. These drone-captured frames have a native resolution of 3840×2160 which would benefit from downsampling to the canvas frame size $640 \times (640/aspect)$ (where maintaining $aspect = Image_{width}/Image_{height}$ prevents image warping) *before* transmission to save on bandwidth overheads. As seen in Figure 3.2, for $M \in (1, 4, 9, 16)$ cameras distributed into $i \times i; i \in (1, 2, 3, 4)$ grids, as i increases, smaller regions of the canvas frame are allotted to each of the M images, yielding lower object detection accuracy. Intuitively, as i increases, uniform downsampling of images to smaller dimensions force small objects to become progressively smaller and less distinguishable, until a point where the DNN model is unable to detect the objects. Crucially, Figure 3.2 shows the comparative loss of object detection accuracy between $r = 1$ (when the captured image is transferred at full permissible resolution $640 \times (640/aspect)$ under no pixel budget constraints) and $r = 0.5$ (where the image is downsampled to $320 \times (320/aspect)$ to fit the pixel budget). I show that the mAP loss is worse for downsampled images that adhere to a lower pixel budget $r = 0.5$. This demonstrates that *RA-MOSAIC* design must ensure that individual cameras apply something more sophisticated than uniform down-sampling to the available pixel budget when determining how the camera frames should be transmitted to the edge for processing.

3.1.3 Resolution Sensitivity in Camera Transmission

I next analyze the impact of down-sampling camera frames to fit the pre-determined pixel budget (or equivalently, the loss of image resolution) on the downstream object detection task accuracy *by object size*. Figure 3.3 shows that resolution loss exhibits a harsher impact on small and medium objects as the pixel budget reduces $r \rightarrow 0.1$,

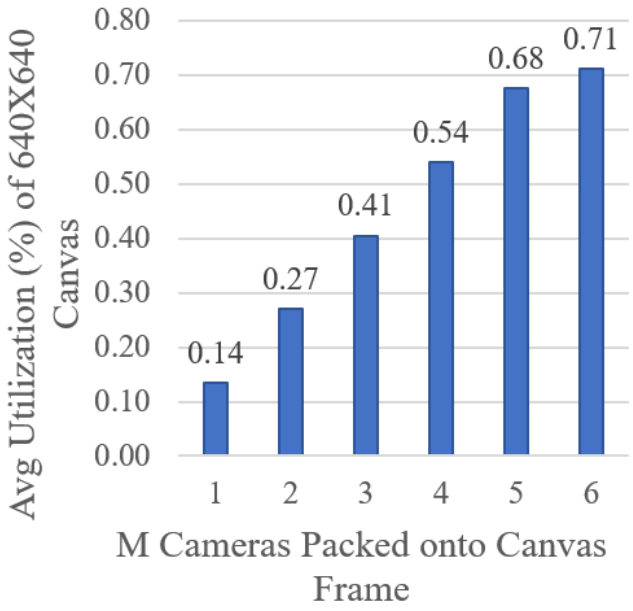

Figure 3.4: Under-utilization of fixed size 640×640 canvas with fixed throughput of 19 FPS

to the point of loss of object detection capability. Conversely when the pixel budget increases with $r \rightarrow 1$, the resulting increase in object resolution has a *variable impact* on objects' detection accuracy: small objects receive a relatively larger gain in object detection accuracy, whereas medium and large objects have marginal gains. This indicates that *variable, object-size aware* resolution adjustment of camera frames, optimized such that higher proportions of the pixel budget are concentrated on regions of the camera frame containing smaller objects, will boost the accuracy of a downstream object detection task.

3.1.4 Optimizing Canvas Utilization

Canvas-based processing primarily focuses on extracting and packing RoI from M cameras into a single canvas frame of fixed size. However, real-world camera streams may have temporal variations in (i) the perceived number of objects/RoI within each camera stream (e.g., a camera monitoring vehicular traffic at an intersection will see a larger number of vehicles during rush hour), and/or (ii) dynamic M number of cameras requiring edge processing at any given time (e.g., if cameras are activated

only on demand [123]), with both conditions resulting in dynamic workloads at the edge. Packing extracted RoI into a canvas frame of fixed size enforces a uniform processing capacity at the edge which leads to both lower canvas frame utilization (in the event of smaller number of RoI or M) with a fixed/lower processing throughput. Figure 3.4 illustrates the canvas utilisation rate for $M \in (1, ..., 6)$ cameras from the Okutama Action dataset [20] when packed onto a fixed canvas frame size of 640×640, which yields a fixed processing throughput of 19 FPS on an NVIDIA Jetson TX2, regardless of the workload/object volume being processed from M cameras. Lower volume of objects accumulated at the edge from a smaller number of cameras ($M \leq 4$) shows a significant portion $\geq 50\%$ of the canvas frame being under-utilized while incurring constant/higher processing throughput of 19 FPS, indicating some *wastage* of DNN computational resources over "blank" areas of the fixed size canvas frame. Intuitively, this creates an optimization opportunity for *RA-MOSAIC* to adapt to the incoming workload at the edge by dynamically sizing the canvas frame appropriately (while always staying within the throughput-mandated maximum bound), based on (i) the number of camera streams M received at the edge and (ii) the number of RoI contained therein. *RA-MOSAIC* can then opportunistically incur lower DNN inference latency (and conversely, faster processing throughput) over smaller canvas frame sizes when workloads are low.

3.2 *RA-MOSAIC* Design Overview

I now present *RA-MOSAIC's* end-to-end solution that (i) extends the concept of criticality estimation to the camera for optimized, *bandwidth adaptive* multi-resolution frame transfer, and (ii) performs *workload adaptive* canvas-based processing over M camera streams at the edge, illustrated in Figure 3.5.

At each camera, the multi-resolution Bandwidth Aware Camera Transmission (BACT) pipeline first estimates potential Regions of Interest (RoI) in the camera frame by alternating between two modes of operation: (i) periodic full frame detec-

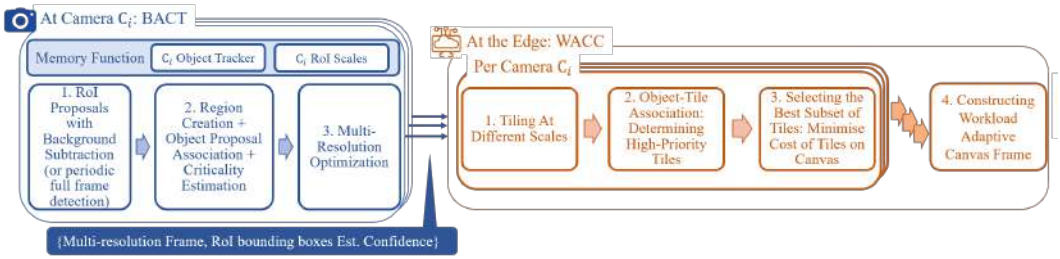

Figure 3.5: *RA-MOSAIC* Conceptual Block Diagram with Bandwidth Adaptive Camera Transmission (BACT) operating at the camera and Workload-Adaptive Canvas Construction (WACC) operating at the edge (Note: Best viewed in color).

tion using a lightweight object detection model yielding object bounding boxes and confidence values–this is performed every T seconds (T is a user-defined valued, with default$= 30$ seconds), and (ii) motion-based RoI detection using background subtraction, with Random Forest Regression (RFR) of confidence values based on RoI size and aspect ratio; both modes of operation yield a calculated/estimated object detection confidence value. The detected object bounding boxes yielded by YOLOv5n (an ultra-light model with only 1.9M parameters, suitable for execution on camera processor boards) and RFR, provide a rough and ready estimate of the spatial distribution, size, and detection confidence for relevant objects of interest, described in detail in Section 3.2.1. With these estimates, BACT then determines the criticality associated with the corresponding regions (assumed to be square *tiles* for computational efficiency) of the frame. Finally, given an overall pixel budget as a constraint and a set of criticality weights, BACT seeks to determine the resolution (pixel count) of each such tile so as to *optimally* utilize the pixel budget.

In addition to the BACT pipeline, *RA-MOSAIC's* camera operation integrates two background tasks. First, a Kalman Filter based object tracker keeps account of the detected objects' estimated positions during the background subtraction based RoI estimation phase. The object tracker resolves the drawbacks of motion-based RoI detection by maintaining in memory stationary objects and/or objects that might have come to a halt (and subsequently missed by the background subtraction estimation) before the next round of full frame detection. Second, a K-Means clustering algorithm periodically observes RoI size distributions to determine the

number and dimensions of camera-specific tiling scales that will best capture the RoI for canvas-based construction at the edge–details of this clustering algorithm are available in Chapter 2 Section 2.3. Periodic (re)calculation at the camera allows *RA-MOSAIC* to remain current with object distribution patterns, thereby reducing the risk of incorrect canvas construction downstream that reduces achievable object detection task accuracy.

The camera finally transmits the multi-resolution BACT frame, the detected bounding boxes, and if necessary the updated/changed camera tiling scales to the edge for further processing. The BACT pipeline is reliant on *fast and lightweight* estimation of criticality to create multi-resolution images in real-time for *RA-MOSAIC's* optimal performance.

At the edge, the Workload Adaptive Canvas Construction (WACC) pipeline processes each camera in parallel over the received tuples of $\{$multi-resolution frames, estimated RoI bounding boxes, (optional, updated camera tiling scales)$\}$. The multi-resolution frame is decomposed into a "bag of tiles" at each of the determined per-camera tiling scales and evaluated for presence of RoI bounding boxes at each scale, similar to mechanisms introduced in Chapter 2. Spatial sizing bounds are calculated for each of the selected high priority tiles which constrain the elasticity of each tile during canvas construction. Diverging from the canvas construction process introduced by *MOSAIC*, the novel Bloom2Squeeze 2D bin packing algorithm allows the canvas frame to "bloom" with additional tiles that are added to the canvas frame at its maximum spatial dimension, until the canvas frame reaches the size limit of 640×640, after which the tiles are squeezed within their spatial sizing bounds for bin packing and subsequent DNN inference. Such workload adaptive canvas construction allows *RA-MOSAIC* to opportunistically adopt smaller canvas frame sizes that incur lesser processing latency, thereby achieving faster processing throughput.

3.2.1 Bandwidth Adaptive Camera Transmission

The Bandwidth Adaptive Camera Transmission (BACT) pipeline marks the first stage of bandwidth-aware adaptation in the overall *RA-MOSAIC* system, as illustrated in Figure 3.5. BACT takes advantage of the growing adoption of AI cameras with onboard GPU support to perform lightweight computation to detect objects of interest "approximately", and assign application-dependent criticality values to different regions of the frame. BACT is a *lightweight, fast* technique executed **on the camera** to non-uniformly downsize an image so that high priority regions of the frame retain higher resolution–i.e., preserve details. A secondary goal of BACT is to ensure that the resulting mixed-resolution frame fits into the reduced pixel budget allocated for transmission, thereby conserving the camera's bandwidth.

Initialization Phase

BACT begins operation with an Initialization Phase wherein it conducts full frame YOLOv5n DNN inference on the first user-defined C camera frames (default, $C = 50$). These full frame detection results initialize both the RFR used for criticality estimation and the K-means clustering algorithm that determines the per-camera tiling scales.

RFR Initialization: The RFR is trained with class-agnostic input variables of $\{$object width, object height, object width:height aspect ratio, ratio of object width to image width, ratio of object height to image height$\}$, with 'object detection confidence' being the target/dependent variable. The R most recent (default, $R = 100$) periodic full frame detection results are also stored locally as inputs for optional retraining of the RFR, if the average estimated confidence falls below a user-defined threshold (default, 0.3).

Per-Camera Tiling Scales: Camera-specific tiling scales are essential to *RA-MOSAIC's* decomposition of each camera frame into a bag of tiles for canvas construction during *RA-MOSAIC's* downstream edge operation. Appropriate tiling

scales ensure that *all* RoI detected in the camera stream can be adequately captured at their appropriate scale, ensuring that the bloom/squeeze operation during canvas construction has the appropriate resizing impact on the RoI (i.e., the RoI consumes a large enough fraction of the tile where resizing the tile does not yield a dramatically smaller RoI). The detected object size distributions from the full frame results are first evaluated for any overlaps between objects and assigned a minimum enclosing rectangle, as motion-based background subtraction may detect such overlapped/occluded objects as a single RoI. The detections and processed boxes for overlapping objects are then clustered using a K-Means algorithm, as illustrated in Chapter 2, with an elbow detection method to determine the most suitable k, or equivalently, the number of tiling scales. Briefly, the centroid $(x_{centroid}, y_{centroid})$ of each determined cluster yields the tiling dimension for each of the k scales, with the larger value $(max(x_{centroid}, y_{centroid}))$ adopted to get "square" tiling dimensions, rounded up to the nearest multiple of 32 for computational efficiency. Additionally, *RA-MOSAIC* introduces a *catch-all* tile, approximately $1.5\times$ larger than the largest determined tile to accommodate the potential observation of objects or RoI larger than those encountered in the initialization phase. *RA-MOSAIC* monitors the periodic full frame detection results for drift in object size distributions, utilizing the most recent R (default $R = 100$) periodic full frame detection results to re-evaluate the per-camera tiling scales.

Bandwidth Adaptive Camera Transmission (BACT)

Once initialized, BACT begins run-time operations in three stages, as illustrated in Figure 3.5.

(1) RoI Proposals: To optimize the processing latency incurred during detection of RoI in the camera frame, BACT alternates between the periodic full frame detection phase (which incurs higher frame latency) and lightweight motion-based RoI estimation. With a user-defined periodicity T (default, $T = 30$ seconds), BACT infers on the camera frame with a YOLOv5n DNN model to yield a list of objects

and their detection confidence. These objects positions are updated in the Object Tracker maintained by the camera, to ensure continued detection of stationary/halted objects that might be missed in the next stage of motion-based estimation. BACT then compensates for the induced latency from running full frame detection by (i) estimating RoI positions/sizes using background subtraction and updating the Object Tracker memory, (ii) recovering potentially halted/stationary RoI from the Object Tracker memory that have not been updated, and (iii) estimating the confidence of the determined RoI using the initialized RFR model. Both modes of operation yield a set of RoI bounding boxes and an estimated/calculated confidence value $conf_{est}$, contributing to the metric of *criticality* to determine which regions of the camera frame must retain higher resolutions or criticality weights.

(2A) Region Creation: To assess which regions of the frame must be assigned higher criticality weights, the input frame is first divided into smaller contiguous regions (i.e. with no overlap between the regions), with each region's dimensions $t_w = t_h$ derived from the input frame dimensions in Equation 3.1:

$$t_w = \begin{cases} \gcd(I_w, I_h), & \text{if } \gcd(I_w, I_h) < I_h \\ \gcd(I_w, I_h)/d, & \text{otherwise, downscale by } d = 4 \text{ default} \end{cases} \tag{3.1}$$

where I_w and I_h are the input frame's width and height respectively. Intuitively, the region size is chosen so that an integral number of regions together cover the entire original image. For example, an input frame of 640×360 can be divided into equal regions of dimension 40×40 (i.e., $\gcd(640, 360)$), producing a total of 144 contiguous square regions of the input image.

(2B) Region→Object Association and Region Criticality Estimation: BACT next constructs a spatial quadtree with the contiguous regions and conducts an intersecting bounding box search using each RoI bounding box (identified in (1)) as a query. An RoI bounding box is mapped to a specific region if either (i) the RoI bounding box is contained entirely within the region, with the regions' dimensions implicitly being larger than the object, or (ii) the object partially intersects with the

region (i.e., it straddles multiple regions) and its fractional presence in this region is larger compared to neighboring regions. This association handles occluded objects including instances where objects of different scales might be occluding each other (for e.g. a car occluding a person).

Each region is then assigned a criticality weight w_i, determined as the complement of the average confidence of the r RoIs assigned to that region, i.e. $w_i = 1 - (\overline{conf_{est}})$ where $\overline{conf_{est}} = \frac{1}{r}\sum_{i=1}^{r} conf_{est}$. As the estimated confidence $conf_{est}$ of each RoI is a factor of its size and detection confidence, such a criticality weight ensures that higher criticality is assigned to regions with lower average confidence.

3. Multi-Resolution Optimization: The set of tiles, together with their criticality weights, then serve as an input to a Max-Min optimizer that adjusts the resolution of each tile relative to its weight to the input frame, while adhering to an overall pixel budget for image transfer to the edge. Equation 3.2 captures the optimization objective and constraints: for each of n tiles, the objective function takes in a term x_i which computes the ratio of the t_i tile's weight w_i relative to the tile's computed final resolution $(t_w \times t_h)$, where t_w and t_h denote the tile's width and height respectively, and w_i is scaled by the quadratic factor $k = 2$. The solver minimizes x_i, standardized by the mean and standard deviation of all such x_i terms for n tiles, and scaled by an empirically determined term $\alpha = 0.9$ for each objective term x_i. Intuitively, the optimization seeks to maintain each tile's size to be relatively proportional to its weight (by seeking to minimize the standard deviation of the normalized tile size), while also ensuring that each tile does not deviate from its original aspect ratio.

The first constraint in Equation 3.2 ensures that the sum of all the tile areas do not exceed the predefined pixel budget P_b, while the second constraint seeks to maintain the aspect ratio of each tile, post resizing, to be as close to the original square tile so as to avoid egregious scaling distortions of the original image. In practice, this optimization problem is tackled using a Sequential Quadratic Programming solver [130], which upon convergence yields as output the optimised tile resolutions

(t_w, t_h) for each tile. BACT resizes each tile, using fast bilinear interpolation, to this optimized dimension through either upsampling or downsampling.

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} \alpha \times x_i + (1-\alpha) \times \frac{x_i - \mu}{\sigma} \\ \text{where} \quad & x_i = \frac{w_i^k}{t_w \times t_h}, \text{where} w_i = 1 - \frac{1}{r}\sum_{i=1}^{r} conf_{est} \forall r \in t_i \\ & \mu = \frac{1}{n}\sum_{i=1}^{n} x_i, \sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{n}} \\ \text{s.t.} \quad & \sum t_w \times t_h \leq P_b \text{ with } 0.95 \leq \frac{t_w}{t_h} \leq 1.1 \;\; k = 2, \alpha = 0.9 \end{aligned} \tag{3.2}$$

The resulting tiles, with varying sizes and resolutions, are then JPEG encoded and transmitted to the edge for further processing along with metadata representing the detected ROI bounding boxes and their estimated/detected confidence values. Figure 3.6 visualises the BACT output for two different pixel budget constraints of $640 \times 360, (R = 1)$ and $320 \times 180, R = 0.5$ and illustrates the efficacy of the BACT methodology in preserving resolutions for critical regions of the frame. I note that BACT's *tiling-based* resizing technique differs from other recently-described multi-resolution spatial sampling strategies [150, 146, 161], as it is intentionally designed to enable efficient spatial packing by the subsequent WACC mechanism at the edge.

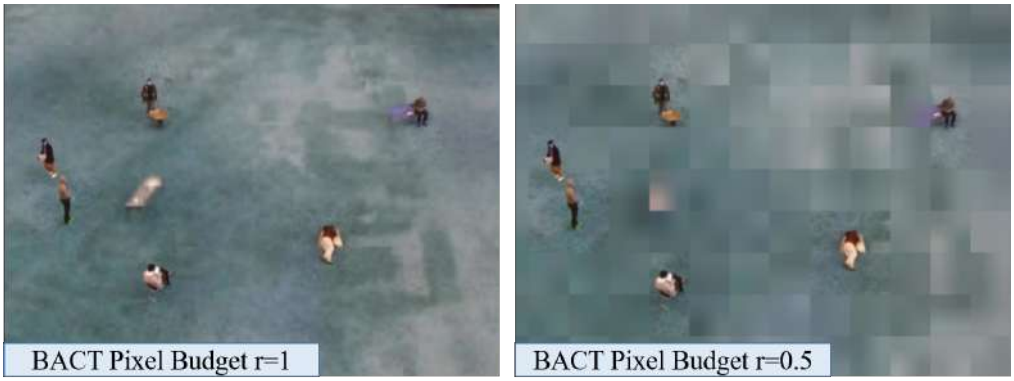

Figure 3.6: BACT under bandwidth conditions yielding pixel budget ratios $r = 1$ and $r = 0.5$ (note the 'blockiness' of the background for $r = 0.5$)

3.2.2 Workload Adaptive Canvas Construction

The Workload Aware Canvas Construction (WACC) pipeline deployed *at the edge* marks the second stage of *RA-MOSAIC's* overall design, illustrated in Figure 3.5. The WACC pipeline ingests mixed-resolution BACT frames, estimated/detected ROI bounding boxes, their estimated/detected confidence values, and the per-camera tiling scales on a *per-camera* basis. At its core, the WACC pipeline decomposes the BACT frames from each of the M camera streams into a bag of tiles, which faithfully represents the RoIs from each camera, for joint evaluation and inclusion onto a canvas frame for subsequent DNN inference. The WACC pipeline adopts the canvas-based processing fundamentals described in Chapter 2 with the additional optimizations of workload-aware dynamic canvas sizing deployed in the novel Bloom2Squeeze canvas construction algorithm. WACC balances the following conflicting objectives:

1. *For RoI tile evaluation:* An RoI bounding box received from the camera-based BACT pipeline must be ideally encompassed in an extracted tile at its appropriate RoI-specific scale (i.e. consume a large enough fraction of the tile without cropping the RoI). This a condition which preferentially selects smaller tiles, which are likely to minimize 'wasted' background pixels.

2. *For RoI tile evaluation:* The total number of tiles selected for inclusion on the canvas frame must be minimised while ensuring all RoI bounding boxes are included. This is a condition which naturally prefers larger tiles, which may individually encompass more RoI bounding boxes.

3. *For canvas construction:* The dynamically selected canvas size must allow for each selected RoI tile to attain the maximum possible spatial dimensions within its application-determined spatial sizing bound. This favours "blooming" the tiles to larger dimensions that might yield more confident/accurate detections.

4. *For canvas construction:* The dynamically selected canvas size must be minimized as much as possible so as to incur lower processing latency and con-

versely higher processing throughput. This favours squeezing the tiles to smaller tile dimensions within their spatial sizing bound at the cost of task accuracy.

I now describe the four stages of the WACC pipeline that together provide an appropriate balance between the throughput-accuracy trade-off by adapting canvas-based processing to the available workload from the t selected tiles across M cameras. The first three stages occur in parallel per-camera, culminating in canvas construction, a global final stage.

(1) Generating the Bag of Tiles: As described earlier in Section 3.2.1, the BACT pipeline evaluates incoming object size distributions to determine (i) the number of camera specific tiling scales k, and (ii) the tiling dimensions to be adopted by each of the k scales. The first stage of the WACC utilizes these pre-computed per-camera tiling scales to extract tiles at each of the k scales from the BACT frame with a user-configurable amount of overlap (i.e. stride s, default $s = 0.5$) between two consecutive tiles.

(2) Determining High-Priority Tiles: The previous stage of WACC tiles the input BACT frame with no adherence to the estimated RoI bounding boxes, yielding a bag of tiles wherein some tiles may completely/partially encompass RoI bounding boxes or not at all. WACC next determines the tiles that completely encompass the RoI bounding boxes while balancing the conflicting objectives for RoI tile evaluation described earlier. WACC leverages a spatial quadtree data structure to perform an intersecting bounding box search for the tiles that completely intersect with each of the RoI bounding boxes, favouring tiles that completely encompass the RoI bounding box. For example, WACC prefers tile A over tile B as illustrated in Figure 3.7. At this stage, WACC also computes a "goodness of fit" criteria for each of the ROI:tile pairs used in the next stage of processing. This "goodness of fit" criteria refines the selected RoI:tile pairings by filtering only those tiles whose contained RoI consume between $(50\%, 90\%)$ of the tile height (for vertical rectangular RoI boxes) or width

(for horizontal rectangular ROI boxes). RoI boxes that consume a smaller $\leq 50\%$ proportion of the tile height or width are better serviced by smaller scale tiles while RoI boxes that consume a very large $\geq 90\%$ of the tile height/width might suffer undue cropping and thus be better serviced by a larger-scale tile.

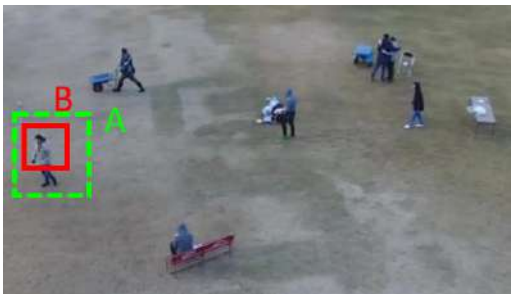

Figure 3.7: WACC evaluating tiles A and B with differing "goodness of fit"; WACC prefers Tile A for the enclosed RoI

(3) Selecting The Best Subset of Tiles: From a set-theoretic perspective, many combinations of tiles might feature or spatially "cover" all RoI bounding boxes captured at different scales or in adjacent tiles extracted at the same scale. To enable efficient canvas construction, WACC has to contend with the following conditions: (i) all RoI bounding boxes must be included/featured at least once in the canvas frame to yield accurate object detection capability, this condition assembles the smallest set/cardinal number of tiles that "cover" all RoI boxes, and (ii) only those tiles must be included on the canvas that are likely to retain the best possible object dimensions after resizing (i.e. the object must take a large enough proportion of the enclosing tile) while also minimizing cost of including that tile on the canvas frame or the number of "wasted pixels" (i.e. background pixels or non-RoI pixels), this conditions the set generation to select tiles that minimize the presence of non-RoI pixels. This dual optimization is achieved by a greedy approximation of the NP-Hard Min Cost Min Set Algorithm (MCMSA), described in Algorithm 2. The MCMSA algorithm essentially ensures that only those tiles that satisfy both the conditions above are selected for canvas construction.

This greedy approximation has two key advantages: (i) RoI are limited to appearing in at least one or at most two unique tiles of different scales in the constructed

Algorithm 2 Greedy Min Cost Min Set Cover Algorithm

$Universe = \{m_1, m_2, \ldots, m_m\}$ ▷ Set of M masks
$Tiles = \{t_1, t_2, \ldots, t_n\}$ ▷ Set of N tiles that may contain one or more assigned masks
$Costs = \{c_1, c_2, \ldots, c_n\}$ ▷ Set of costs/wasted pixels for N tiles
$S \leftarrow \emptyset$
while $S \neq Universe$ **do**
    if $\mid t_i - S \mid \geq 0 \ \ \& \ \ (c_i / \mid t_i - S \mid) > 0$ **then**
        $Subset \leftarrow \min(c_i / \mid t_i - S \mid)$ ▷ minimize the number of tiles containing the same mask and minimize the additional cost to the canvas associated with adding an additional tile
        $S \leftarrow S \cup Subset$
    end if
end while
    =0

canvas frame, reducing the need for Non Maxmum Suppression post-processing for each RoI across multiple tiles, and (ii) multiple RoI detected in close proximity are usually encompassed in a single large tile, rather than multiple small tiles, reducing the overall number of wasted pixels on the canvas frame.

Finally, for each selected tile in this subset, WACC computes two parameters used for canvas construction. The first parameter is a spatial sizing bound within which the tile (and encompassed RoI) can either bloom or squeeze during canvas construction. This bound is empirically determined offline as the minimum and maximum resizing ratios for each RoI (for example $(0.7x, 1.2x)$ for large RoI tiles) subject to which the achievable object detection confidence is not adversely impacted more than a pre-determined threshold, (default, threshold 3%). This follows the intuition that this sizing bound is RoI/scale dependent where large RoI are more resilient to resizing and can still be detected with reasonable confidence even when reduced to 70% of their original size. Regions containing smaller RoI adopt tighter bounds due to the RoI's sensitivity to detection confidence capability where the RoI become impossibly small/undetectable when down-sized or excessively pixellated when up-sized, both conditions impeding accurate object detection. The second parameter is an elasticity factor that determines how much a tile should be squeezed within this determined spatial sizing bound at every iteration/attempt at canvas construction, if the tiles need

to be squeezed onto the canvas frame to admit all selected t tiles across M cameras. A higher elasticity factor potentially allows for faster convergence of the constructed canvas frame but unduly squeezes each tile towards its lower sizing bound to achieve convergence. A smaller elasticity factor, on the other hand, might yield more optimal resizing at the cost of longer convergence times, impeding WACC's asynchronous canvas construction and DNN inference abilities. I profile the convergence times for a variety of $\{$tile size, canvas size$\}$ combinations in an offline setting (i.e. no convergence deadline) and average (i) the squeeze/elasticity suffered by each tiling scale at every iteration to achieve the desired convergence time, and (ii) the achieved detection confidence. I select the elasticity factor for each tile online using both bounds; first, the desired convergence time before the DNN inference cycle of the previous canvas frame finishes (to maintain asynchronous processing), and (ii) the amount of squeeze that protects the object from being detected with confidence no lower than the estimation received from the BACT pipeline.

(4) Constructing a Canvas Frame: The WACC converges all per-camera pipelines to yield a set of t tiles filtered and selected for canvas construction across M cameras. The Bloom2Squeeze algorithm, described in Algorithm 3, describes the final cross-camera stage of the WACC pipeline that yields a constructed canvas frame, comprising tiles across the M cameras, that is sent for DNN inference.

The algorithm begins by initializing the canvas frame size as the square-root of the total pixel demand presented across all t tiles, rounded to the nearest multiple of 32 for ease of DNN inference. Such initialization avoids the sequential tile-by-tile expansion of the canvas frame and forms the first optimization to promote faster convergence time of the algorithm. If the total pixel demand from all t tiles is greater than 640×640, the algorithm proceeds to the Inverse 2D "Squeeze" Bin Packing step immediately as detailed in Chapter 2, where minimal perturbations in the input tile dimensions allow for their successful packing into a fixed size canvas frame. This formulation allows *RA-MOSAIC* to perform no worse than MOSAIC. If the pixel demand is less than 640×640, the Bloom2Squeeze algorithm methodically

Algorithm 3 Bloom2Squeeze Algorithm

```
function BLOOM2SQUEEZE(tiles)
    totalPixels ← calculateTotalArea(tiles)
    initialCanvasSize ← roundToNearestMultipleOf32(squareRoot(totalPixels))
    sortedTiles ← sortTilesByHeightDescending(tiles)
    if initialCanvasSize ≥ 640 then
        canvas ← initializeCanvas(640, 640)
        inverse2DBinPackAllTiles()
    else
        initializeCanvas(initialCanvasSize)
        while not all tiles are packed do
            tile ← popLargestTile(sortedTiles)
            while cell not found and packing not successful do
                packingSuccess ← packTileInCell(tile, cell)
                if not packingSuccess then
                    if insufficientVerticalSpaceInCell() and nextCellInColumnAvailable() then
                        cell ← nextCellInColumn()
                    else if insufficientHorizontalSpaceInCell() then
                        if cellOnRightIsAvailable() then
                            cell ← MergeCells()
                        else
                            cell ← topMostAvailableCellInNextColumn()
                        end if
                    end if
                else if packingSuccess then
                    splitCanvasRowAndColumnAtCell(cell)
                end if
            end while
            if not packingSuccess then
                if numPackedTiles ≤ 0.9×totalTiles and canvasWidth ≤ 640 then
                    bloomCanvas()
                else if numPackedTiles > 0.9× totalTiles or canvasWidth = 640 then
                    inverse2DBinPackAllTiles()
                end if
            end if
        end while
    end if
    return canvas
end function
```

attempts packing all available tiles by adopting spatial division techniques such as "grid splitting" typically employed by CSS-sprite equipped websites to optimize space utilization on a dynamically sized webpage [119, 76]. First, the available tiles are sorted by size with the largest tile available for packing taking precedence. Placement of the largest tile at the top-left-most position (i.e. at pixel position $(0, 0)$) of the canvas frame splits the remaining canvas frame along the tile width and height, resulting in 4 cells, out of which the top-left cell is occupied by the placed tile. Bloom2Squeeze repeats this process by scanning the canvas frame for available cells from top to bottom and left to right, preferring to move to another cell in case of insufficient cell:tile height, merge adjacent available cells in case of insufficient cell:tile width, and finally performing a grid split along the height and width of each placed tile. In case the initial canvas frame size is evaluated to be insufficient with more than 10% of t tiles yet to be packed, Bloom2Squeeze "blooms" the canvas size to the next multiple of 32, removes the last placed i tiles (user-defined, default = $i = 5$) to allow for better packing decisions, and continues the process for the remaining tiles. If however, only a smaller proportion $\leq 10\%$ of the t tiles are yet to be packed, or if the canvas size has already boomed to the maximum permissible dimension of 640×640, Bloom2Squeeze initiates Inverse 2D Bin Packing by squeezing all remaining tiles onto the canvas frame.

The Inverse 2D Bin Packing problem [35] contends with a fixed-size bin (or in this case the maximum permissible canvas size 640×640) and dynamically sized items or tiles, where the algorithm must converge on the minimum resizing/perturbation to the size of the items such that they can be packed into the fixed size bin. MOSAIC and by extension *RA-MOSAIC* employ a Differential Evolution Algorithm (DEA) for Min-Max Optimization of the given set of t tiles, their individual elasticity factors and spatial sizing bounds. At each iteration until convergence, the DEA algorithm attempts to pack all t tiles while allowing each tile to obtain the maximum possible squeezed spatial dimensions within its bounds. The resulting canvas after convergence details the dimension and location of each of the packed t tiles. WACC

concludes by mapping each of the tiles spatially to its designated location and dimension, before DNN inference.

Upon DNN inference, WACC performs post-processing of the detected bounding boxes, translating the boxes to the original BACT frame and performing NMS on RoI/object boxes that might have appeared in more than one tile (of different scales) on the canvas frame.

3.3 *RA-MOSAIC* System Design

The BACT pipeline is deployed at the camera and the WACC pipeline at the edge. Each camera independently and concurrently evaluates its own video stream for regions of interest (RoI) and transmits both the RoI bounding boxes and the optimized multi-resolution image in accordance with the available pixel budget (described earlier in Section 3.2.1). With a user-defined periodicity of T seconds (default, $T = 30$ seconds), the BACT pipeline runs a full frame detection on the camera frame to locate stationary objects, and collect data on the continued suitability of the Random Forest Regressor (RFR) model and the evaluated tiling scales for that camera. Using a Tensor-RT optimized YOLOv5n model on the NVIDIA Jetson Nano, a representative compute-enabled camera, this incurs an additional processing latency of $48ms$ for every full frame detection. This additional latency reduces the *average* BACT video stream frame-rate to from the original camera frame-rate of 30 FPS to 29.95 FPS, a negligible cost to the entire end-to-end processing pipeline. At the edge, the WACC pipeline receives a stream of BACT frames from each of the M cameras. M BACT images are decomposed into a bag of tiles in parallel and subsequently jointly evaluated for inclusion onto a dynamically-sized canvas frame to balance the need for both high throughput and high task fidelity (illustrated in Figure 3.8). Deviating from the system design described by *MOSAIC*, I restrict the batch size to 1, $b = 1$, to allow *RA-MOSAIC* to right size the canvas frame size to the *current* workload without any apriori assumption on future workload. The achieved

processing throughput is thus a function of the number of tiles across N cameras that are selected for inclusion on the dynamically sized canvas frame and their spatial size bounds within which the canvas frame size blooms or conversely squeezes. I establish through empirical studies that with a batch size $b = 1$, the NVIDIA Jetson TX2 incurs a processing latency of $52ms$ (or equivalently a throughput of 19 FPS) over a 640×640-sized canvas frame. I adopt this configuration as the maximum permissible processing latency *in the worst case*, with *RA-MOSAIC* preferentially adopting smaller canvas frame sizes to opportunistically boost the throughput. In evaluating the M or the maximum number of cameras that can be supported, *RA-MOSAIC* ensures that (i) the selection of critical tiles across M cameras and subsequent canvas construction does not take longer than the DNN inference deadline of $52ms$ to enable streamlined asynchronous inference, and (ii) the chosen tiles from M cameras are appropriately sized within their spatial sizing bounds during the *RA-MOSAIC's* Bloom2Squeeze canvas construction operation. To adher to both conditions, *RA-MOSAIC* adopts the maximum camera processing capacity M so as to guarantee 19 FPS processing throughput *in the worst case.* *RA-MOSAIC* adopts $M = 6$ for the pedestrian detection application and $M = 3$ for the license plate recognition tasks. The lower number of cameras supported i.e. $M = 3$ for the license plate recognition task is due to the stricter spatial sizing bounds enforced by the OCR pipeline downstream.

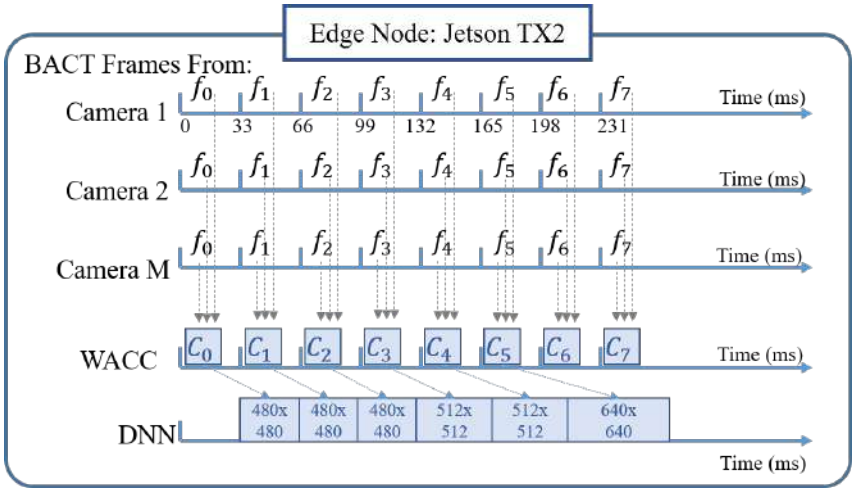

Figure 3.8: *RA-MOSAIC* System operation of the WACC pipeline at the edge

Evaluation Platform: I evaluate *RA-MOSAIC* on the NVIDIA Jetson TX2 [108], an edge-scale device featuring a 256 CUDA-core PASCAL GPU, and an ARMv8 multi-processor architecture supporting both a quad-core ARM Cortex A57 MPCore and a dual-core NVIDIA Denver 2 CPU. On the other hand, I deploy the NVIDIA Jetson Nano [111] as a representative compute-enabled camera with a Quad-core ARM A57 CPU. The Jetson Nano transmits the images to the edge TX2 device over WiFi.

Evaluation Model: At the camera, *RA-MOSAIC's* BACT pipeline uses a pre-trained YOLOv5n model for periodic full frame detection. At the edge, *RA-MOSAIC's* WACC pipeline adopts a TensorRT-optimised YOLOv5s model, a representative edge-scale model with 7.2M parameters and 16.5 GFLOPs. I fine-tune a pre-trained model (pre-trained on MS COCO [92]) on the Okutama-Action and UFPR-ALPR datasets to achieve higher sensitivity to objects at different perspectives/occlusion/sizes.

Benchmark Datasets: To maintain a fair comparison with *MOSAIC*, I adopt the same benchmarks tasks of (i) drone-based pedestrian detection using the Okutama-Action dataset [20], and (ii) automatic license plate recognition using the UFPR-ALPR dataset [83]. Both tasks feature object detection as a fundamental step in the processing pipelines with distinct application-level requirements for image/resizing fidelity for downstream processing. This is reflected in the spatial-sizing bound that is selected for the vehicle objects in the UFPR-ALPR benchmark that ensures adequate pixel fidelity of license plates to enable accurate downstream Optical Character Recognition. 43 drone sequences from the Okutama-Action dataset, captured at 4K (3840×2160) image resolution at 30 FPS, are processed into 54664 and 14210 frames for training and testing the YOLOv5s edge model respectively. On the other hand, the UFPR-ALPR dataset features 90 sequences captured from a car-mounted camera at 1920×1080 resolution at 30 FPS, providing 3600 and 1800 frames for training and testing the YOLOv5s edge model respectively. Each sequence is considered as an independent camera stream, with *RA-MOSAIC* utilizing M sequences without duplication for canvas construction.

Evaluation Metrics: To monitor gains in perception/detection accuracy, I evaluate the Mean Average Precision metric at an IoU threshold of 0.5 i.e. mAP@0.5, and also report the processing throughput i.e., inference frames per second FPS as well as Cumulative FPS (C-FPS) across all M cameras. For the license plate recognition task, I evaluate the impact of *RA-MOSAIC's* operations on Optical Character Recognition capabilities by monitoring the Character Error Rate Metric (CER). CER computes the Levenshtein distance metric that counts the minimum number of single-character changes that are required for the predicted string to converge with the ground truth, averaged by the number of characters included in the groundtruth. Perfect character recognition capabilities are achieved with lower CER$\rightarrow 0$ rates.

Evaluation Baselines: I compare *MOSAIC* with batch size restricted to $b = 1$ (MOS-M) and *RA-MOSAIC* (RA-MOS-M) against each other, using four distinct baselines, described below.

1. Naive FCFS: Frames are processed in a naive First come First Served basis on a canvas frame of size 640×640 without any modifications.

2. Naive Uniform-M (Uni-M): This baseline divides a canvas frame of size 640×640 into a grid structure with each cell naively assigned to a single frame from each of the M cameras. Uni-M also determines the best choice between horizontal and vertical stacking variations during the creation of the grid structure, such that each cell affords each assigned input frame the best/lowest possible resizing ratio in comparison to the frame's original spatial dimensions.

3. Bandwidth Adaptive FCFS (BA-FCFS): Multi-resolution frames generated by BACT deployed at the camera processed in a FCFS manner at the edge for DNN inference.

4. Bandwidth Adaptive Uniform-M (BA-Uni-M): Multi-resolution frames generated by BACT deployed at the camera packed into uniform grids at the edge for DNN inference.

3.4 Evaluation

I first describe how the bandwidth and resource adaptive mechanisms of *RA-MOSAIC* impact the throughput-accuracy tradeoff for dynamic workloads featuring a variety of object sizes and camera perspectives, for both drone-based pedestrian detection and automatic license plate recognition tasks. I also conduct ablation studies to isolate and understand the impact of each of the dynamic adaptations to *RA-MOSAIC* processing pipeline.

3.4.1 Pedestrian Detection Application

I vary (i) M, the number of cameras deployed in each experiment and (ii) r, the bandwidth-determined pixel budget ratio describing the amount of resolution down-sampling applied to the camera frame before transmission, to observe *RA-MOSAIC's* resource adaptations described in Sections 3.2.1 and 3.2.2, and their impact on the throughput-accuracy tradeoff. I select three scenarios to showcase *RA-MOSAIC's* system performance and gains over a variety of real-world scenarios. First, an ideal scenario featuring high workloads at the edge from $M = 6$ cameras with each camera transmitting camera frames under *no* bandwidth restrictions, represented by a pixel budget $r = 1$. Second, a less ideal scenario with similarly high workloads from $M = 6$ cameras, with each camera transmitting frames in a bandwidth-constrained environment, represented by a pixel budget $r = 0.5$. Third, a similarly bandwidth-constrained scenario (pixel budget $r = 0.5$) but with the edge processing a lower workload represented by $M = 3$ cameras. I show how *RA-MOSAIC always* outperforms competitive baselines, across this diversity of workloads and bandwidth conditions.

High Workload and No Bandwidth Restriction: I first evaluate the Pedestrian Detection baseline in an ideal setting with $M = 6$ cameras, the maximum supported workload on a single edge-device for this dataset, described earlier. Figure 3.9 plots the throughput-accuracy tradeoff where (i) all $M = 6$ cameras are active,

(ii) adequate bandwidth is available for transmission across all $M = 6$ cameras (i.e. a pixel budget ratio of $r = 1$), and (iii) DNN inference is conducted with a batch size of $b = 1$. Processing each of the BACT frames from $M = 6$ cameras sequentially, the bandwidth adaptive FCFS (BA-FCFS) baseline achieves the highest accuracy of 79.9% as each BACT frame is resized to fit into a 640×640 canvas frame for inference, but suffers a lower throughput of 3FPS per camera. *RA-MOSAIC* on the other hand achieves comparable accuracy with $\leq 1\%$ loss while achieving $5.6\times$ or 566.67% higher throughput for $M = 6$ cameras, achieving 20 FPS on average *per-camera* (cumulatively, an impressive 120 FPS!). Uniform-M for $M = 6$ cameras (depicted as Uni-6) and bandwidth adaptive Uniform-M (BA-Uni-6) suffer a throughput loss of 5% with 19 FPS achieved per-camera while sacrificing 8.1% and 7.7% accuracy, respectively, compared to *RA-MOSAIC*. Compared to MOSAIC, *RA-MOSAIC* achieves (i) an 11.11% higher throughput of 20 FPS on average *for each camera* (cumulatively 120 FPS across $M = 6$ cameras) due to workload adaptations and (ii) a $\sim$1% higher task accuracy due to the bandwidth adaptations *even when no throughput restrictions are applied*. This goes to show the value of the resolution preservation capabilities of the BACT pipeline operating at the camera, even while operating in the best possible transmission environment.

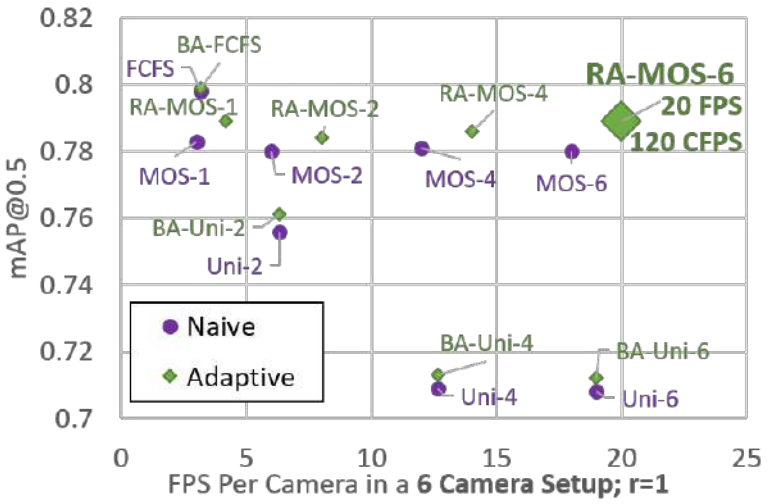

Figure 3.9: *RA-MOSAIC* Throughput-vs-Accuracy for pedestrian detection, *high workload* from $M = 6$ cameras with *no* bandwidth restriction

High Workload with Bandwidth Restriction: In a bandwidth-constrained environment with an operating pixel budget ratio $r = 0.5$, I start to observe diverging system behaviours between MOSAIC and *RA-MOSAIC* even when processing high workloads from $M = 6$ cameras, illustrated in Figure 3.10. Due to the bandwidth-adaptive pixel preservation techniques of the BACT pipeline, *RA-MOSAIC* achieves a 14.3% and 17% gain in accuracy while also maintaining an average throughput gain of 11.11% and 5% over MOSAIC and BA-Uniform-6 respectively. In effect, *RA-MOSAIC* appears to push the Pareto frontier for both accuracy and throughput objectives.

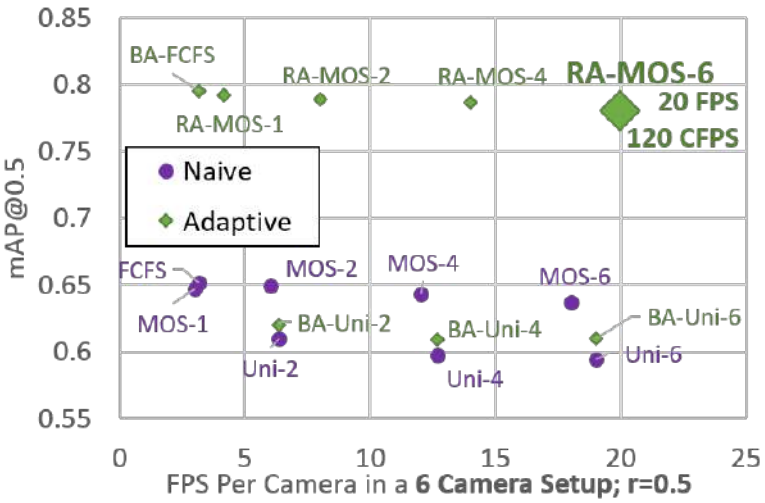

Figure 3.10: *RA-MOSAIC* Throughput-vs-Accuracy for pedestrian detection, *high workload* from $M = 6$ cameras with bandwidth restriction ($r = 0.5$)

Low Workload with Bandwidth Restriction: Lower workloads from $M = 3$ cameras would allow the WACC pipeline at the edge to adaptively choose smaller canvas sizes on average so as to reduce the incurred processing latency and increase throughput. This can be seen in Figure 3.11 where *RA-MOSAIC* significantly outperforms MOSAIC with a (i) 22.22% gain in throughput to 22 FPS *per-camera* (cumulatively 66 FPS) and a simultaneous (ii) 15% gain in accuracy, showcasing how the bandwidth and workload adaptations from the BACT and WACC pipelines work in tandem to provide the best possible gains in the throughput-accuracy trade-off continuum.

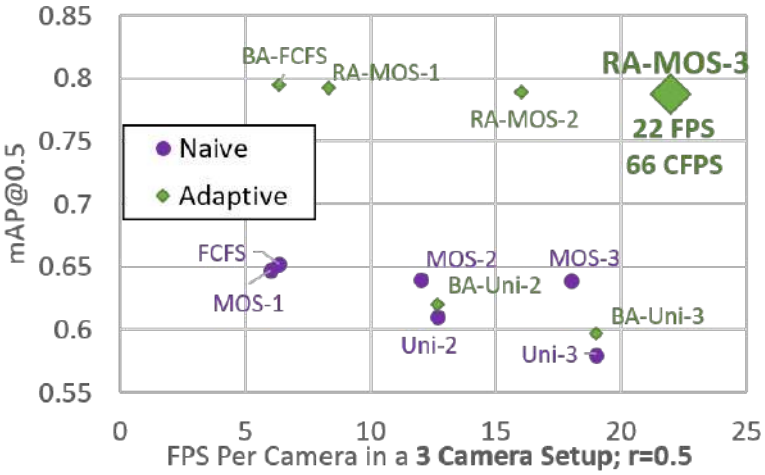

Figure 3.11: *RA-MOSAIC* Throughput-vs-Accuracy for pedestrian detection, *low workload* from $M = 3$ cameras with bandwidth restriction ($r = 0.5$)

3.4.2 License Plate Recognition Application

Significant gains in OCR capability are also observed in the License Plate Recognition application, with the resolution adjustments of the BACT pipeline reducing the character error rate (CER) by 9% in bandwidth unconstrained environments $r = 1$ (Figure 3.12), and by 26% in bandwidth-constrained environments $r = 0.5$ (Figure 3.13) when compared to MOSAIC for $M = 3$ and $M = 2$ respectively. *RA-MOSAIC* also showcases on average a 11.11% gain in throughput (20 FPS per camera) across both high and low workloads from $M = 3$ and $M = 2$ cameras respectively, illustrated in Figures 3.12 and 3.13 respectively.

3.4.3 Comparative Study with Batched Processing of RoI Tiles

A widely used alternative to canvas-based processing is criticality-aware *batched processing* of extracted RoI tiles/patches [94, 61, 95]. Batched processing allows a single DNN inference cycle to be simultaneously used for multiple inputs, facilitating parallelism and more efficient use of the available GPU resources. However, one key limitation of such batched processing is that all input images within a single DNN cycle *must* be of the same spatial dimensions for DNN inference. This requirement

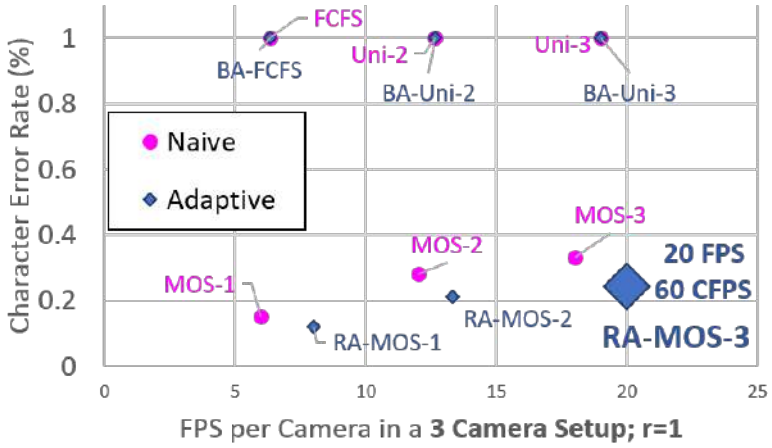

Figure 3.12: *RA-MOSAIC* Throughput-vs-CER for license plate recognition, *high workload* from $M = 3$ cameras with no bandwidth restriction

stands in contrast to the dynamic RoI tile resizing strategy (within each tile's individual spatial sizing bound) adopted by canvas-based processing exhibited by both MOSAIC and *RA-MOSAIC*. I now explore the impact of this difference in terms of the achievable mean task accuracy for pedestrian detection. I first profile the DNN inference times offline for various $\{$RoI tile size, batch size$\}$ combinations as well as average inference time for a fixed size 640×640-sized canvas frame on the Jetson TX2 edge device [108]. Subsequently, in the online evaluation of batched inference, I select the appropriate combination of $\{$tile size, batch size$\}$ for the available t tiles that are selected across M cameras for batched DNN inference so that the DNN execution cycle at the edge ends no later than the DNN inference over one 640×640 fixed-size canvas frame. By relaxing *RA-MOSAIC's* workload-aware adaptations, I allow for fair comparisons over batched inference, MOSAIC with batch size $b = 1$, and *RA-MOSAIC* (natively $b = 1$), highlighting the value of bandwidth-aware multi-resolution adaptations exercised by *RA-MOSAIC* in its BACT processing pipeline. I conduct this evaluation in a bandwidth constrained environment by limiting the available pixel budget to $r = 0.5$.

Figure 3.14 plots the achievable mean accuracy (i.e. mAP@0.5) from batched processing, MOSAIC processing, and *RA-MOSAIC* processing over $M \in (1, 2, 4, 6)$

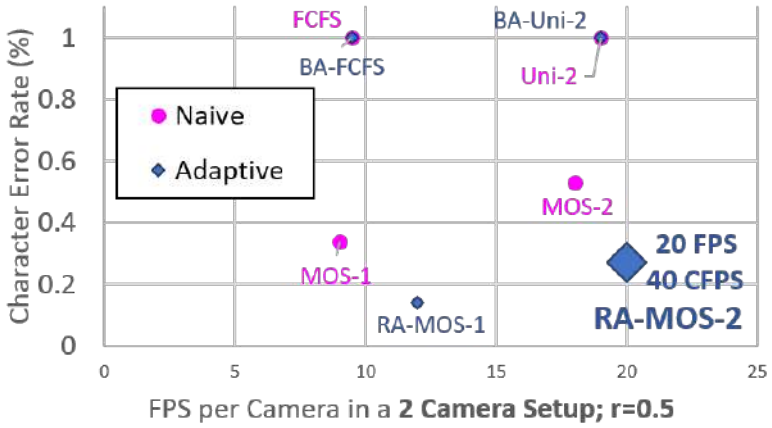

Figure 3.13: *RA-MOSAIC* Throughput-vs-CER for license plate recognition, *low workload* from $M = 2$ cameras with bandwidth restrictions ($r = 0.5$)

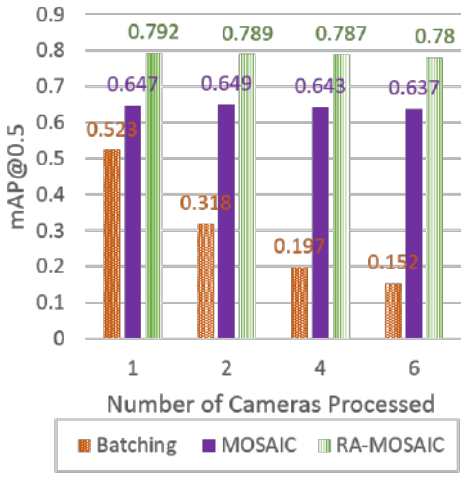

Figure 3.14: *RA-MOSAIC:*Accuracy vs number of cameras processed for different RoI processing methods under bandwidth constraints

camera simultaneous camera streams. I see that the bandwidth adaptations utilized by *RA-MOSAIC* enables a 26.9% gain in accuracy over batched inference even when processing a single $M = 1$ camera stream in a bandwidth-constrained environment. As the number of cameras supported (i.e. M) increases, I see a significant $\geq 60\%$ degradation in the achievable accuracy from batched inference of $M = 4$ and $M = 6$ camera streams due to the dual challenges from (i) uniform down-sampling of images to fit the required pixel budget, and (ii) the requirement that all RoI tiles be of the same input dimension for batched DNN inference. In contrast, *RA-MOSAIC's* object

detection capability remains effectively unchanged, demonstrating its resilience to object/tile resizing for DNN inference due to the resolution preservation from the BACT pipeline and the non-uniform tile resizing strategy during canvas construction.

3.4.4 Ablation Studies

Impact of Bandwidth Adaptive Camera Transmission on Task Accuracy

BACT redistributes pixels from less critical regions of the frame to more critical regions which might contain high-priority objects while adhering to a pre-determined pixel budget. To more carefully observe the impact of criticality-awareness at the camera on mean accuracy, I therefore evaluate BACT-resizing vs. uniform resizing, for varying values of pixel budget ratio r. As seen in Figure 3.15, criticality-aware BACT resizing preserves pixel resolution for the more critical regions of the frame, thereby boosting object detection accuracy by $\sim 2\%$ (compared to uniform resizing) even when $r = 1$. I conclude that BACT is *always* superior to uniform resizing due to the redistribution of pixels to more critical areas of the frame, with objects gaining pixel density and resolution. Moreover, as r decreases up to 0.3 (tighter pixel budget), BACT accuracy is relatively stable and can exceed the mAP of uniform resizing by as much as 40%. For low-power, low-bandwidth deployments, *RA-MOSAIC* can thus allow the camera node to reduce the data transfer overhead by up to $\sim$70%, without sacrificing DNN task accuracy.

Impact of Workload Adaptive Canvas Construction on Throughput

I observe *RA-MOSAIC's* dynamic and instantaneous adaptation to varying workloads by observing the canvas size that is selected by the WACC pipeline for the t available tiles. Such workload variation is generated, for the Okutama-Action dataset, by dynamically introducing an additional camera for processing at the edge, increasing from $M = 2$ camera streams to $M = 3$ camera streams. Illustrated in Figure 3.16, I see the how the canvas size chosen varies even with $M = 2$ camera streams, arising

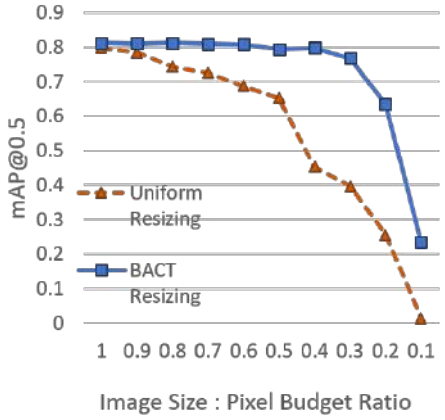

Figure 3.15: Pixel budget ratios vs mAP@0.5 for naive and bandwidth adaptive resizing methods at the camera

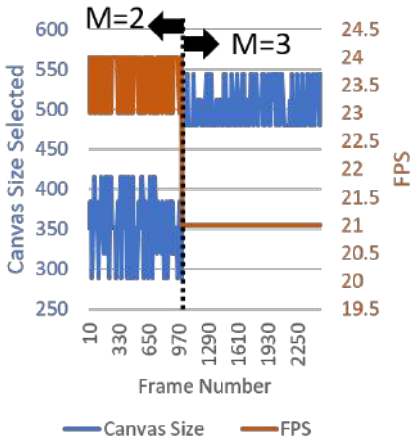

Figure 3.16: Workload Adaptations in *RA-MOSAIC* WACC pipeline when a camera is added for edge processing

out of fluctuations in the observed workload from the perceived number of objects and resulting RoI tiles across both cameras. I observe that the corresponding WACC processing throughput achieved over the $M = 2$ camera streams also fluctuates depending on the canvas size selected for inference, varying between 24 to 23 FPS for canvas sizes between 288×288 and 416×416. I can better observe *RA-MOSAIC's* workload adaptations from the WACC pipeline when a new camera is added for processing at the edge as the resulting canvas sizes selected for the new and increased workload "blooms" between 480×480 and 544×544, resulting in a slightly lower 21.23 to 20.82 FPS average throughput.

3.5 Discussion

Alternative Strategies to Tackle Variable Workloads: In this chapter I showed how a canvas-based perception pipeline can adapt to lower workloads by opportunistically adopting smaller canvas sizes to yield faster processing throughput. Another strategy to deal with lower workloads is to allow the ingest of more camera streams to consistently maximize the canvas volume and GPU resource. Such a strategy would yield similar performance trade-offs as shown by the MOSAIC pipeline in Chapter 2. However, to realise such a strategy, the perception pipeline would need to

be supplemented by a fog-level multi-edge application logic which monitors workloads across multiple such edge nodes to redirect camera streams to an edge node processing lower workloads. While out of the scope of this thesis, I hypothesize that classical distributed systems mechanisms for workload stealing or sharing similar to the latency-aware mechanisms shown by Yi et. al. [157] could address this need.

Canvas-based Processing of Spatiotemporally Correlated Multi-Camera Streams: In the case of multi-camera systems with established spatial overlap between cameras, multiple objects may be visible from different camera streams at different observer-object distances, angles, and perspectives. With respect to the MOSAIC pipeline at the edge (shown in Figure 3.5), there arises the opportunity to further fine-tune the spatial sizing bounds used for canvas construction based on (i) appearance of object across multiple intra-camera and inter-camera frames (ii) need for shorter or longer detection periodicity (i.e. how often an object is included on the canvas frame for inspection). Such modifications to the criticality and application dependant spatial sizing bound can certainly expand the functionality of *RA-MOSAIC's* system to a more diverse range of applications. I describe changes to the pipeline to accommodate such camera deployment characteristics in Chapter 4.

Considering Object/RoI Arrival Patterns: Different objects may arrive and traverse the sensing field at different physical velocities resulting in variable pixel displacements between successive frames that contain the object. Modelling object arrivals could help further fine-tune the selection of critical tiles across all objects sensed from M cameras. Such estimation is dependent on variables like object class, object-camera distance, camera extrinsic calibration, and might benefit from the integration of physical models to model object motion. However, to preserve high DNN inference throughput, any such sophisticated modeling will need to remain computationally lightweight. This thesis does not explicitly model object arrival times into the perception pipeline, leaving such modelling of object arrival times as an open problem.

Real-time Adjustment to Available Wireless Bandwidth: In the Bandwidth-Aware

Camera Transmission (BACT) pipeline, I introduce the pixel budget as a meta-variable representing available network bandwidth. I show how the BACT pipeline takes this pixel budget as a constraint to create appropriate multi-resolution images which focus available bandwidth on regions of the frame that contain potential regions of interest, and also show in Section 3.4 how this optimization improves achievable DNN task accuracy even in bandwidth restricted environments. However, in the construction of the BACT pipeline, I make an important assumption that this pixel budget meta-variable is determined in real-time by a lightweight mechanism operating at the camera, which estimates image transmission latency from heartbeat pings to the edge device to determine available up-link bandwidth, similar to prior works such as Chameleon [73] by Jiang et. al.

Variations in System Design: I also expect that the following variations in system design could be built into *RA-MOSAIC*: (i) Accuracy-Aware Canvas Construction, where tiles are prioritized and mapped to canvas frames inferred upon by a zoo of models, chosen from a throughput-accuracy tradeoff continuum, and (ii) Heterogeneity of Compute where canvas frames are built and mapped to an available mix of CPU/GPU/VPU/TPU hardware for differentiated quality of service to different tiles/cameras. I leave these for future work.

Canvas-construction for Other Applications: In addition to DNN-based inference over composite canvas frames constructed from multiple input camera streams, another intriguing application for canvas construction could be in long-term storage of surveillance videos. Currently, videos are either stored in high resolution or compressed for long term archival purposes, amounting to significant volumes of stored data. Canvas frames featuring key objects of interest constructed from multiple cameras could achieve multiplicative *savings* in storage needs over time. I hypothesize that generative vision models could be trained to fuse objects recovered from canvas frames with a static frame from the original camera stream to reconstruct object $\{$appearance, motion$\}$ over time to *recreate* in a sense, the original camera stream. While such reconstruction is intuitively feasible for static cameras, more

exploration would be required on the feasibility of reconstruction of similar videos from cameras in motion (such as those mounted on autonomous cars, robots, and drones). Such a research direction is out of scope of this thesis but presents interesting paradigm shifts for applications outside the domain of real-time processing.

Chapter 4

Exploring a Temporal Degree of Freedom: JIGSAW

In this chapter, I introduce a temporal degree of freedom in the Canvas-based Processing paradigm to provide further flexibility to schedule "when" the perception pipelines (re)selects critical stimuli or regions of interest for (re)inspection. However, such a scheduler stands the risk of artificially introducing a temporal lag between when a critical stimuli is captured by the camera and when it is inferred upon by the DNN, which can cause a significant time lag between the state of the physical world and that estimated by the perception pipeline. To this end, I utilize the concept of *streaming latency*, which explicitly accounts for object localization latency and object localization accuracy, to develop techniques that allow the perception pipeline to stay abreast of real world kinematics.

4.0.1 Streaming Perception and Canvas Construction

Fast *and* accurate machine perception is a cornerstone for many real-time "streaming" urban applications, such as camera based vehicular tracking and real-time collision avoidance [5, 61]. While modern DNN-based vision models offer high accuracy, their high computation demands are less favorable to 'fast', real-time perception [5, 86], especially on resource-constrained edge devices. This generates a tension between

edge computing and streaming perception: while edge-based processing is critical to reduce the transmission latency and bandwidth for high-volume data streams, the higher *processing latency* on edge devices can lead to a dissonance between the state of the physical world and that perceived by DNN-based inference algorithms. More specifically, considering a vision-based object localization task, streaming perception frameworks need to optimize *both* the accuracy and latency of machine perception. The recently proposed *streaming perception paradigm* [86] explicitly considers this need to balance object localization accuracy and latency, selectively discarding frames to maximize a novel *streaming accuracy metric*, illustrated in Figure 4.1. As the Figure shows, if the frame capturing the location of the vehicle at $\phi(t_i)$ is processed by the perception pipeline with a latency $\delta = t_i - \phi(t_i)$, the inference is available only at time t_i. At that time, the state (location) of the vehicle may have evolved significantly since the time instant$\phi(t_i)$; as a consequence, even if the object detection accuracy is 100%, the streaming accuracy is degraded due to the difference between the inferred location and the actual ground truth location at time t_i.

State-of-the-art techniques for reducing the inference latency for vision perception tasks adopt one of the following approaches: a) Utilize *lightweight, smaller DNN models*, typically with lower input resolution, that have lower processing latency but suffer from reduced task accuracy; (b) Adopt *an imprecise computing paradigm*, aborting DNN execution at intermediate layers (often with reduced accuracy) to meet a processing deadline [80, 24]; (c) Employ system designs, such as intelligently reducing the resolution or frame rate of input camera streams [146, 32]. These approaches, however, all tackle individual sensor streams in isolation, without considering multi-camera deployments and potential spatiotemporal correlations across sensors, and do not aim to directly maximize "streaming accuracy".

This mismatch between edge resources and streaming perception needs is further amplified in multi-tenancy deployments, where a single GPU-equipped edge device (e.g., a Jetson TX2) concurrently executes DNN-based perception over video streams

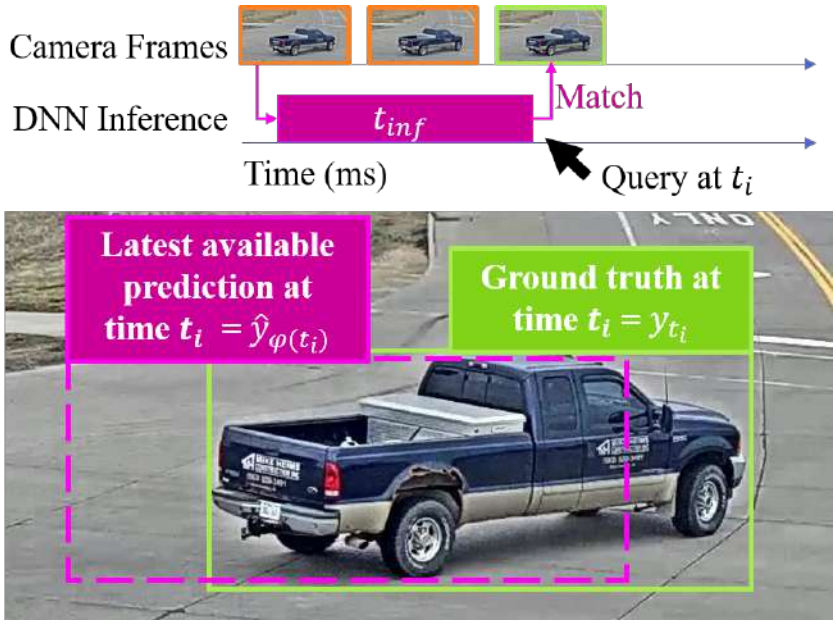

Figure 4.1: Streaming Perception: When DNN inference completes with t_{inf} latency, the car has moved (green box) from its predicted location (magenta box). Streaming perception queries the state of the predicted world at time t_i and matches the groundtruth y_{t_i} with the latest available prediction i.e. $\hat{y}_{\phi(t_i)}$.

from *multiple* $N \geq 2$ vision sensors. To mediate the contention for shared GPU resources across such N streams, exemplar approaches such as *MOSAIC* introduced in Chapter 2 and TETRIS [132], utilize the concept of *spatial multiplexing*, effectively arbitrating among multiple spatial regions across N distinct frames generated by the vision sensors. The *MOSAIC* system in Chapter 2 introduces the concept of *canvas frames*, a unit of shared 2D pixel space that is sized to ensure sufficiently high DNN processing throughput. However, pure spatial multiplexing approaches are unsuitable for streaming perception across multiple camera feeds, as they do not directly consider the latency impacts of stream processing.

In this chapter, I build on the canvas-based spatial multiplexing paradigm, exemplified by *MOSAIC* and *RA-MOSAIC*, to develop *JIGSAW*, a technique that explicitly optimizes edge DNN-based inference for multi-camera **streaming applications**, such as traffic surveillance. While *JIGSAW* builds upon the conceptual model of canvas frames, it possesses two key novel features. (a) First, it significantly extends the notion of canvas-based spatial multiplexing to support *spatiotemporal multiplexing*.

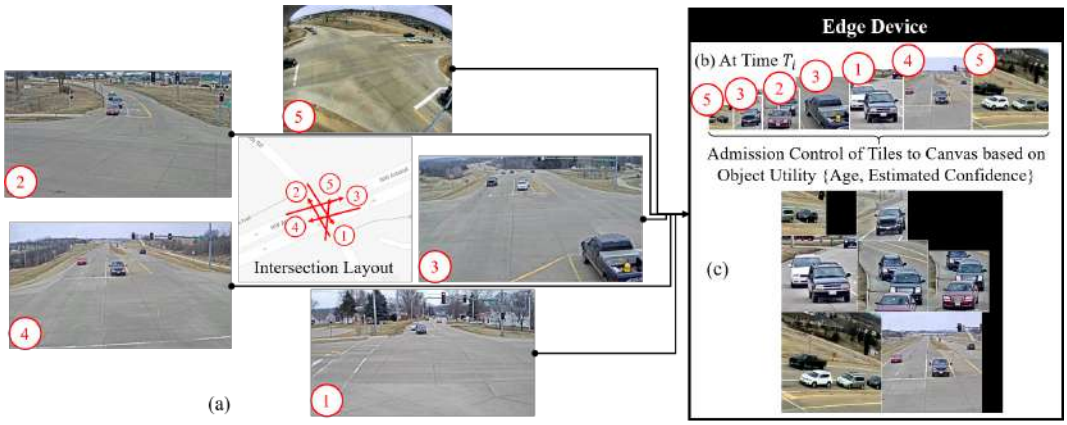

Figure 4.2: ***JIGSAW*** Overall Functionality: (a) Multiple spatially overlapped cameras deployed at a traffic intersection are processed by a single edge device (b) At DNN inference time T_i, estimated regions of interest (tiles) are extracted at their appropriate scale from the source camera frames and (c) evaluated for their utility to the streaming perception task before being packed onto a canvas frame.

In this new paradigm, sequences of frames arriving from multiple camera sensors are pre-processed to not only share the pixels within a single canvas frame, but are also differentially interleaved in time (and even dropped) to optimally utilize pixels across multiple consecutive canvas frames. (b) Second, it explicitly accounts for the reality that multi-camera urban deployments often exhibit non-trivial *spatial overlap* between cameras. Such overlap implies that multiple cameras sometimes monitor the same object from different perspectives and thus possess a level of information redundancy. *JIGSAW* monitors and exploits such object-level redundancy to reduce the number of regions of interest, across the N camera streams, that must be squeezed within individual canvas frames, thereby improving the system's maximum camera capacity.

JIGSAW fundamentally utilizes the spatiotemporal redundancy in a multi-camera setting to both (a) increase the admissible camera capacity, and (b) improve the edge device's throughput vs. accuracy trade-off. Figure 4.2 illustrates the high-level operation of *JIGSAW*. Frames from different camera sensors are processed, using very lightweight techniques, to create a set of tiles (Figure 4.2(b)), indexed by each unique object in the sensing field, that capture an object at multiple spatial scales and possibly from multiple perspectives. A subset of such tiles, corresponding to multiple

different objects, are then packed into a sequence of canvas frames (Figure 4.2(c)) after suitable curation (including potentially discarding all tiles for some objects) and resizing, with DNN inferencing then executed on this sequence of composite canvas frames. Conceptually, while MOSAIC previously addressed the question of *how* to spatially apportion a shared canvas frame across multiple object-specific tiles, *JIGSAW* additionally determines (a) *when* and (b) *what* regions of interest from different camera sensor streams should be multiplexed on to a shared canvas. Using the benchmark CityflowV2 multi-view traffic surveillance dataset [134], I shall show how *JIGSAW* effectively creates a unified, autonomous system for streaming intelligence that can simultaneously handle (i) real-world variations in object density and kinematics, (ii) varying degrees of spatial overlap between different camera views, and (iii) heterogeneity in camera frame rates and resolution.

4.0.2 *JIGSAW*: Key Contributions

In this chapter, I make the following key contributions:

- *Judiciously exploit multi-camera spatial overlap*: I show how *JIGSAW* identifies the distinct set of tiles (sub-regions), across multiple camera sensor streams with partial spatial overlap, associated with an individual object and then judiciously selects *what* subset will best balance computational overhead and perception accuracy. By developing a utility maximizing formalism for such subset selection, *JIGSAW* offers a superior accuracy-vs.-overhead calculus compared to prior extremes that either (a) effectively partition the sensing field across cameras (e.g., [56]), selecting only one tile and discarding others to save computation, or (b) include all tiles across cameras (e.g.,*MOSAIC*), to maximize perception accuracy while ignoring computational cost. Empirical results on the CityflowV2 [134] dataset shows that (i) compared to (b), *JIGSAW* effectively reduces the number of tiles/pixels contending for DNN inferencing by 32% with a 4.3% gain in accuracy, and (ii) compared to (a), *JIGSAW* offers 9.55% improvement in task accuracy with only

2.3 ms higher tile processing overhead on average.

- *Develop a spatiotemporal pipeline for maximizing multi-camera streaming accuracy:* I show how *JIGSAW* uses a multi-stage, *computationally lightweight* pipeline to both compute *when* object specific tiles should be scheduled for DNN inferencing and *how* such tiles should be spatially combined/squeezed into individual shared canvas frames. Using a lightweight per-object tracker, *JIGSAW* dispenses with the paradigm of scheduling every incoming frame (object tile) for DNN inferencing and instead adopts a priority-based scheduler technique to schedule object-level inference intermittently. To subsequently fit selected candidate multi-view tiles, across multiple objects, into a single canvas frame, *JIGSAW* uses an enhanced inverse-bin packing algorithm that explicitly differentiates between mandatory and optional multi-view tiles. By providing greater flexibility in tile selection, *JIGSAW* avoids the small-object problem (especially when N is large) where too many tiles are packed onto the canvas frame leading to loss of object distinguishability and detectability. Experimental studies on CityflowV2 show that, for $N = 25$ cameras, *JIGSAW's* bin-packing achieves 28.7% greater streaming object recall (sAR) when compared to MOSAIC which naively packs all tiles from all cameras into a canvas frame.

- *Quantify JIGSAW's performance gains and flexibility:* I use an Nvidia Jetson TX2-based [108] implementation of *JIGSAW* to quantify its superior performance. Using CityflowV2, I show that, in contrast to a baseline FCFS processing (where each incoming camera frame is processed separately), *JIGSAW* can simultaneously process ***25 cameras*** on a single GPU with a 66.6% increase in accuracy and a simultaneous 18x gain in throughput to 19 FPS per camera (cumulatively 475 FPS) on a single TX2. Likewise, in contrast to MOSAIC (where all incoming frames are spatially multiplexed into a canvas frame, but without consideration of multi-view redundancy), *JIGSAW* achieves 42.3% increase in streaming perception accuracy without any loss in throughput.

- *Demonstrate Adaptation to Dynamic Workloads & Network Conditions:* I show how *JIGSAW* adapts to real-world variability of the underlying wireless network, providing 65.1% increase in accuracy and a simultaneous 14x gain in throughput to 15 FPS per camera (cumulatively 375 FPS) across ***25 cameras*** on a single GPU. In addition, *JIGSAW* can seamlessly handle heterogeneity in camera frame rates: in a simulated bandwidth constrained wireless environment for CityflowV2 with $N = 25$ cameras, where 12 cameras operate only at half their original frame rate, *JIGSAW* achieves $\sim$54.9% gain in streaming perception accuracy compared to FCFS.

4.1 *JIGSAW* Design Overview

I now describe *JIGSAW's* edge-based, spatiotemporally aware streaming perception pipeline. Figure 4.3 (best viewed in colour) illustrates the components. At a high level, *JIGSAW* is guided by a Dynamic Scheduler (DS) that orchestrates between the different pipeline components. A per-camera object tracker OT_i, executing independently for each $i \in \{1, \ldots, N\}$ camera, processes all frames from the i^{th} stream to maintain an updated time-stamped estimation of the sensed physical world. If DS determines that the ongoing DNN inference cycle is nearing completion, DS signals each "Per-Camera Operation" to (a) decompose its most recent camera frame into a set of tiles that extract objects of interest (OoI) at their appropriate scale, and (b) evaluate each tile's utility to the streaming perception accuracy metric. The "Cross-Camera Operation" then consolidates objects/tiles across all cameras using a Cross-Camera Tile Mapping database and classifies such tiles as either $\{$Mandatory, Optional$\}$. These set of tiles are then prioritized and down-selected, before being Inverse Bin Packed [35] onto the canvas frame for inference, with the resulting detected bounding boxes then post-processed for translation to the coordinates of the original camera frame(s). Lastly, the per-camera trackers update all associated objects with the detection confidence and inference timestamp before the next cycle

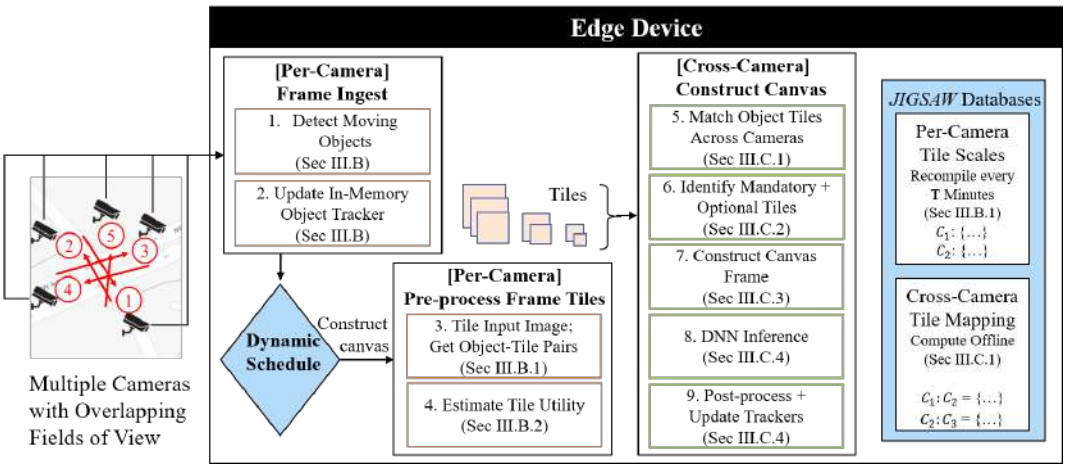

Figure 4.3: ***JIGSAW's*** system block diagram (Best viewed in colour). *JIGSAW* processes spatially overlapped multi-camera streams (such as from a traffic intersection) with Per-Camera (in orange) and Cross-Camera (in green) run-time operations, with assistance from databases (in blue) built offline prior to deployment

of canvas construction is initiated.

4.1.1 Design Choices

To achieve its goal of maximizing the *streaming accuracy* metric under diverse real-world artifacts, *JIGSAW* dispenses with the common practice of using object-specific deadlines, which typically are not generalizable and depend heavily on dataset distributions, observer-object distance, physical velocity, and camera angle/perspective. Instead, *JIGSAW* evaluates the state of the observed world just prior to canvas construction to determine the marginal utility of including specific objects in the canvas frame. Such a design is accommodating of hardware or network heterogeneity, which can require *JIGSAW* to support camera streams with varying frame rates. Finally, *JIGSAW* makes no a-priori assumption on either the presence or absence of overlap between cameras and can operate seamlessly under different deployment conditions.

4.1.2 Dynamic Scheduler

The Dynamic Scheduler (DS) orchestrates the execution of other sub-components to support canvas-based streaming perception across the N cameras. DS is based

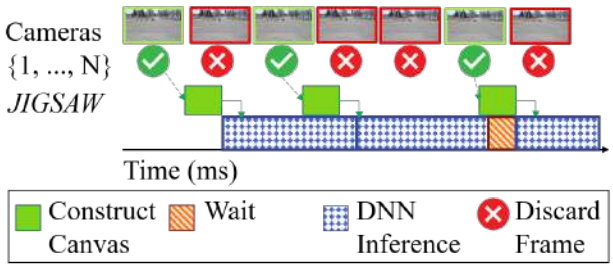

Figure 4.4: Conceptual schedule of *JIGSAW's* edge-based streaming perception components

on the key observation [86] that the physical world changes dynamically during DNN inference, and that streaming perception is thus often enhanced by skipping prior, *stale* frames and instead running *timely inference* on more recent frames. Accordingly, to minimize processing latency, DS occasionally performs idle-wait, preserving GPU cycles for anticipated fresher frames (instead of processing stale frames). Figure 4.4 illustrates the non-work conserving DS operation where it uses estimated DNN inference latency to decide whether or not to (a) stitch a new canvas frame from N cameras, (b) discard stale frames after the initial processing described in Section 4.1.3, or (c) idly wait for the imminent arrival of fresher frames.

4.1.3 Per Camera Operation

JIGSAW uses a per-camera frame ingest pipeline (orange components in Figure 4.3) to pre-process frames from N cameras in parallel. The i^{th} ingest pipeline receives the latest frame transmitted from camera C_i and prepares it for downstream canvas-based inferencing, while also maintaining a per-frame representation of the perceived state of the physical world. Objects in motion are estimated using a background subtraction mechanism and the resulting bounding boxes are updated using a Kalman Filter-based Centroid tracker. The tracker maintains a memory function over the last N time-stamps (N is a user-defined variable, default=5) and retains objects in memory even if are missed by the background subtraction mechanism (for example, if they are stationary for brief periods).

Tiling Scales per Camera: *JIGSAW* adopts *MOSAIC's* mechanism of decomposing

each frame into a bag of tiles (at multiple spatial scales), each of which may contain objects of interest (OoI). This tiling operation is performed on-demand, once DS decides to proceed with canvas construction, using the freshest (most recent) camera frames.

The number of tiles and their scale dimensions are first computed offline and then asynchronously updated during a periodic profiling phase executed every T minutes (depicted in blue in Figure 4.3). Full frame detection, on a subset of frames from camera C_i, is used to profile the observed OoI size (i.e. $\{$height, width$\}$) distribution. This OoI size distribution is clustered by a K-Means algorithm with an elbow detector to determine a suitable value of k or the number of tiling scales required by camera C_i, similar to the mechanism detailed in Chapter 2. Such profiling-based choice of tiles ensures that *JIGSAW* generates tiles that can encapsulate OoI at *their appropriate scale*. It is important to note that an OoI–tile mapping can be many-to-many: a single OoI may be contained in multiple tiles, at different scales, while a single tile can contain multiple objects.

Evaluating Tile Utility: After creating such per-object, OoI-tile mappings, *JIGSAW* next evaluates the *marginal utility* of each tile for each contained object. Given the desire to both maximize accuracy and minimize the latency impacts of executing inference on the set of selected tiles, I split the calculation of such a marginal utility into two sub-operations. Within the the per-camera pipeline, the first-half of the utility function estimates the object detection confidence that can be achieved for each tile (and its contained OoI(s)) from camera C_i if it were to be included onto the canvas frame, *without being squeezed*. I define the tile utility as the ratio of this estimated confidence of the OoI-tile pair to the area consumed by the tile without any squeeze on a canvas frame. This enables *JIGSAW* to prioritize those tiles that provide the highest detection confidence for an encapsulated object while also moderating the object's canvas utilization, allowing fair space allocation across all selected tiles during canvas frame construction. *JIGSAW* also estimates the tile-specific spatial sizing bound (min & max), violating which would drastically reduce the

detection accuracy. This sizing bound is empirically evaluated to (1x, 1.1x) for the smallest-scale tile, (0.7x, 1.2x) for the next/medium scales of tiles and (0.5x, 1x) for the largest-scale tile. Intuitively, smaller tiles would suffer the largest drop in accuracy if they are squeezed to smaller dimensions during canvas construction; conversely, excessive upscaling would lead to excessive pixellation. In contrast, the detection accuracy for medium and large tiles is much more robust to resizing. The second-half of the utility function seeks to minimize per-object processing latency by prioritizing "older" objects (for which DNN inference has not been executed recently) for inclusion on the canvas frame. However, this latency-based utility must be evaluated over *all unique* objects across all N cameras, as described later in Section 4.1.4.

Predicting OoI-Tile Detection Confidence: To estimate the detection confidence for every OoI-tile pair, I utilize a Random Forest Regressor (RFR) [129] trained offline using ground-truth features from all cameras views of the CityflowV2 dataset. The RFR estimates an object's detection confidence obtained for the object, when inference is performed using YOLOv5s model, the default model in the *JIGSAW* implementation. The RFR predictor, is trained using training and testing samples of 110,730 and 36,910 unique object-tile pairs, respectively and uses the following covariates: (i) OoI width & height (ii) ratio of OoI width to OoI height (iii) OoI area (iv) tile width & height, (v) tile area (vi) ratio of object width—height—area to tile width—height—area, respectively (vii) tile coordinates in the original camera frame (i.e. $tile_{xmin}$ & $tile_{ymin}$), and (viii) object class. The RFR model achieves an accuracy of 91.46% and a Mean Absolute Error (MAE) of 5% for the predicted detection confidence per object-tile pair. Feature analysis shows that object aspect ratio (OoI width to height ratio) has the highest feature importance=15%, whereas 11% for $tile_{xmin}$ (which effectively serves as a proxy for the vertical position of the object relative to the camera's pose) has the second highest feature importance=11%.

I empirically observe that the detection confidence of the object not only relies on the object and tile characteristics, but also on macro-characteristics such as

object orientation and perspective which cannot be estimated using background subtraction at runtime. However, I generalize that certain regions in a camera frame might consistently observe object features that are better predictors of the detection confidence. For example, a camera mounted on a traffic light may observe the features of the bonnets or side-views of cars which are better predictors of vehicle class objects when compared to the rear-ends or fish-eye perspectives of the cars. The position of the object-tile pair in the camera frame, represented by $tile_{xmin}$ and $tile_{ymin}$ features, are a good approximation of such phenomena, with a feature importance of 11% for $tile_{xmin}$ to the criterion. This is second only to object aspect ratio (OoI width to height ratio) with feature importance of 15% representing another approximation for the observed perspective/angle of the object in that camera field of view (FoV).

At runtime, *JIGSAW* evaluates the OoI-tile mappings from camera C_i to estimate the detection confidence per OoI-tile pair, the resulting tile utility relative to canvas frame utilization, and the spatial sizing bounds of the tile. Every camera then C_i opportunistically submits the per-object top-k (user-defined, default $k = 1$) tiles of the highest utility to *JIGSAW* for canvas construction, representing its best possible estimation of the physical world in its FoV.

4.1.4 Cross Camera Operation

Cross-Camera Tile Mapping Database: Once the per-camera pipelines have culminated, *JIGSAW* collates the OoI-tile mappings across all N cameras. Due to the possible spatial overlap among camera views, certain objects may be visible in frames generated by multiple cameras, and thus captured via OoI-tile mappings of varying utility. A Cross-Camera Tile Mapping database therefore assists *JIGSAW* in matching such tiles, corresponding to the same object, across all N cameras. This database (depicted in blue in Figure 4.3) is pre-computed prior to system deployment by applying object-ReID algorithms over time-synchronized frames across each of

the $\binom{N}{2}$ camera pairs. However, at run time, trying to match objects across $N-1$ cameras would incur high latency. To alleviate this issue, I divide each camera frame into 120×120 (= $gcd(img_{width}, img_{height})$) non-overlapping *super-regions* and register the matched pairs of objects to a single super-region in their respective camera frames. The database then stores a record of super-region tuples that are observed to share views of at least one physical object.

At runtime (depicted in green in Figure 4.3), *JIGSAW* assembles the cross-camera mapping for each object with pairwise evaluation of cameras that have valid super-region tuples. For the 'source' camera, *JIGSAW* constructs a spatial quadtree index of super-regions and performs an intersecting box search, for each OoI, with its constituent tiles. The super-region that intersects with the object tile is used as a key in the database to retrieve the corresponding super-region in the 'destination' camera. A similar quadtree based intersecting search, using this retrieved super-region, is used to identify the tiles that intersect with this super-region. As a single tile may contain multiple OoIs, *JIGSAW* then uses a SIFT feature descriptor to find the closest matching unique object in such tiles. At the end of this computationally-efficient matching process, *JIGSAW* computes, for each unique OoI, a collection of tiles (across multiple camera views) containing the object.

Tile Prioritization: For every unique OoI, *JIGSAW* classifies the tile providing the highest utility (out of all the OoI-tile mappings collected across N cameras) as the *mandatory* tile, with the other tiles being labeled as *optional*. The resulting elements of the set of mandatory tiles (with cardinality equal to the set of unique objects) are then prioritized in decreasing order of the "object age" metric maintained by the object tracker. This age is computed as the difference between the timestamp of the last DNN inference performed on *any* instance of the OoI and the timestamp at which the object's location was last approximated using background subtraction.

Canvas Frame Construction: *JIGSAW* then uses a meta-heuristic approximation of an Inverse Bin Packing algorithm [35] to squeeze as many tiles, across the N cameras, onto the canvas frame of fixed volume. *JIGSAW* aims to balance two

conflicting requirements: (i) the need to process *all* high-priority objects observed across *all* cameras to ensure high streaming accuracy, while (ii) ensuring that each bin-packed tile (and the contained object instance) is not squeezed dramatically to a level where the object detector confidence is dramatically reduced.

When bin-packing tiles, *JIGSAW* operates by prioritizing *mandatory* tiles with higher age. The age utility metric therefore acts as an implicit mechanism of admission control that prioritizes the tiles belong to 'older' objects, thereby assuring that the tracking latency does not degrade dramatically. As described in Section 4.1.1, such a system design adapts automatically to varying class-specific processing deadlines, while also being tolerant of differences in frame rates between cameras. *JIGSAW* attempts to pack all *mandatory* tiles, and as many additional *optional* tiles as feasible, into a canvas frame. This reflects the balance between the two conflicting determinants of object detection accuracy: while accuracy is likely enhanced from multiple tiles/perspectives, it is likely to be diminished if one attempts to pack too many tiles, by squeezing individual tile dimensions, onto a fixed canvas size. Additionally, if *JIGSAW* is unable to pack all mandatory tiles (e.g., when the total number of objects is very high), the age metric will be updated for all the 'starved' objects so as to ensure that those objects are prioritized in the subsequent round of canvas construction and DNN inference.

DNN Inference and Post Processing: The constructed canvas frame is then submitted to the edge device's GPU for DNN-based inference. *JIGSAW* then asynchronously translates the detections (output of the DNN) to their original camera frame coordinates and performs per-camera Non Maximum Suppression on the bounding boxes to handle cases where a single OoI was included in multiple tiles from the *same* camera frame. The cross-camera database is also used to additionally create 'virtual bounding boxes' on the frames of other overlapping cameras that may not have been included in the canvas frames with the maximum detection confidence available per-object across the included tiles. Finally, *JIGSAW* also updates all associated per-camera trackers with these post-processed detections and the DNN inference

timestamp (thus, updating their 'age' metric).

4.2 System Design

I now describe the prototype *JIGSAW* implementation and the process for evaluating its effectiveness for a multi-camera streaming perception task.

Evaluation Platform and Model: I implemented *JIGSAW* on the NVIDIA Jetson TX2 device, an edge computing platform that features a 256-core NVIDIA Pascal GPU architecture with 256 NVIDIA CUDA cores and a dual CPU unit comprising of one Dual-Core NVIDIA Denver 2 64-Bit CPU and one Quad-Core ARM Cortex-A57 MPCore module. I employ another TX2 device as a secondary edge node to enable scalability of the system, by performing on-demand offloading of camera streams if the total object density (across all camera streams) exceeds a predefined capacity threshold. The secondary edge device takes $\sim$140-150ms to initialize and switch to a ready-state to receive camera streams. The two TX2 devices are connected to a desktop transmitting the camera streams using both wired and wireless connections, illustrated in Figure 4.5. *JIGSAW's* default DNN is a half-precision TensorRT optimised YOLOv5s model, containing 7.2M parameters operating at 16.5 FLOPs for an image size of 640x640. I size all canvas frames at 640×640 for DNN inference in a batch size $b = 1$ (unless specified otherwise); on the TX2, this yields an inference latency of $\sim 52ms$ per canvas frame or, equivalently, a throughput of 19 canvas frames per second.

Benchmark Dataset: I evaluate *JIGSAW* using the CityflowV2 dataset [134]. It features 3 hours of synchronised video at 10 FPS from 40 cameras installed across 10 intersections. The dataset features a mix of scenarios, with residential and highway traffic-flow conditions with clusters of cameras exhibiting spatial overlap in their FoV as illustrated in Figure 4.2. I principally use scenarios 'S01', 'S02', 'S03', and 'S04' in the evaluation. Scenario 'S01' and 'S02' observe higher density and velocity in the "highway" setting with an observed object density per second of 10.8 and 10.96

Figure 4.5: *JIGSAW's* System Test Bed with 2x NVIDIA Jetson TX2 device (primary/secondary)

respectively, and new object arrival per second of 0.6 and 1.24 across $N = 5$ and $N = 4$ cameras respectively. On the other hand, scenario 'S03' and 'S04' observe lower density and velocity in the "residential" setting with an observed object density per second of 3.65 and 5.58 respectively, and new object arrival per second of 0.15 and 0.39 across $N = 6$ and $N = 25$ cameras respectively. This difference in object density and kinematics gives an opportunity for a more nuanced understanding of multi-camera streaming perception accuracy in various real-world settings.

Evaluation Metrics: The streaming perception paradigm introduces the metrics of streaming average precision (sAP) and streaming object recall (sAR) [86]. Both these metrics differ from the standard object detection evaluation metrics of Average Precision (AP) and Average Recall (AR) in that the streaming detections are aligned for evaluation by the time of DNN inference rather than the input frame index. In other words, standard evaluation compares $(y_t, \hat{y_t})$ for each frame index at time t while streaming evaluation compares $(y_t, y_{\hat{\phi(t)}})$ where $\phi(t)$ is the prediction time-stamp of the most recent prediction before time t–i.e., $\underset{j}{\operatorname{argmax}}\ s_j < t$, where s_j are the time-stamps of preceding predictions. Such an evaluation jointly evaluates both accuracy and latency of object localisation for real-time machine perception. I set the object capacity threshold as 250 specifically for *JIGSAW* operation, above which the system offloads dense camera streams to the secondary edge device.

Evaluation Baselines: I compare *JIGSAW's* performance against the following baselines:

1. N-GPU Streaming FCFS $b = 1$: The evaluation baseline represents the maximum achievable streaming metric value when N cameras are mapped to N GPUs (an 'infinite' GPU setting) to process fresh frames available from the camera C_i in a First Come First Served (FCFS) fashion.

2. 1-GPU Streaming FCFS – Batching $b = N$: Fresh frames received from N cameras are batched together for DNN inference by a single GPU task and evaluated for streaming accuracy and recall. The batch size b for this baseline is equal to the number of cameras i.e. $b = N$. This baseline represents a naive strategy to batch all N frames for DNN inference and process them in parallel.

3. 1-GPU MOSAIC $b = 1$: I compare *JIGSAW* against MOSAIC which packs all available tiles from fresh frames across N cameras onto a single canvas frame for processing. For this baseline, I choose the streaming canvas-based evaluation setting with batch size $b = 1$.

4. 1-GPU JIGSAW $b = 1$: *JIGSAW* attempts to pack onto the canvas frame all mandatory and as many optional tiles across N cameras, but prioritized by object-specific cross-camera spatiotemporal utility values tuned to the streaming perception task. *JIGSAW* also adopts a batch size $b = 1$ for evaluation.

4.3 Evaluation

I first evaluate the accuracy vs. throughput trade-off achieved by the streaming perception task for N cameras on a single GPU. I then explore the impact of *JIGSAW's* design decisions on streaming object recall and achieved throughput in wireless and bandwidth-constrained environments.

4.3.1 Streaming Accuracy vs. Throughput

Real-time perception pipelines expect that the system ensures both high fidelity object detection and high perception throughput. *JIGSAW's* system design jointly leverages (i) the canvas-based processing paradigm to maximise the object detection accuracy across N cameras, and (ii) the streaming perception paradigm to maximise the perception throughput to consistently sustain 19 FPS per camera (i.e. the expected canvas processing throughput) across N cameras on the Jetson TX2. Figures 4.6, 4.7, 4.8, and 4.9 show that for all evaluated scenarios 'S01', 'S02', 'S03', and 'S04', each with distinct object density and arrival patterns, *JIGSAW* consistently outperforms the 1-GPU FCFS Batching baseline by $57.9\%, 57.4\%, 37.7\%$, and 66.62% respectively in streaming AP (sAP) metric within each scenario. The results show that batching frames from N cameras to support their simultaneous streaming perception on a single GPU provides *almost no benefit*. For the 1-GPU FCFS Batching baseline, a larger N implies higher resource contention due to larger batch sizes (i.e. $b = N$) and proportionately higher latency (equivalently lower throughput), which in turn causes the perception pipeline to *lag behind* the real world dynamics. Indeed, the 1-GPU FCFS Batching baseline adapts to such contention by discarding an increasing number of received frames, thereby avoiding allocating GPU resources to unacceptably-delayed inferencing tasks. This increases the discrepancy between the state of objects in the real world and the DNN detections, causing the IoU to fall below acceptable limits, reducing achievable sAP for 1-GPU FCFS Batching operation. It is important to note here that there is *no* linearity in the relationship between the number of cameras N and sAP (which evaluates how the physical world has changed *during* DNN inference), and is instead a function of $\{N$, new object arrivals rate/kinematics$\}$ which is distinct across the evaluated scenarios. In general, this is reflected in sAP for 'S01' and 'S02' (faster object arrival rate/kinematics) and 'S04' (higher N), when compared to sAP for 'S03' which has a lower object arrival rate across a smaller N.

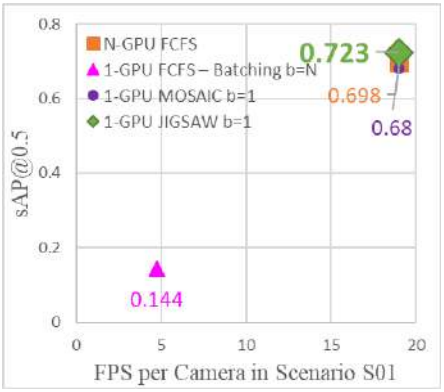

Figure 4.6: sAP@0.5 vs. FPS per Camera in 'S01'; $N = 5$; Density: 10.8 obj/sec; Arrival: 0.58 obj/sec

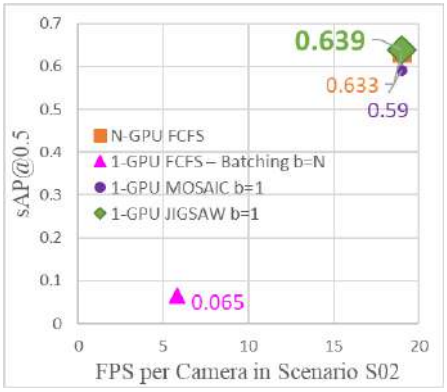

Figure 4.7: sAP@0.5 vs. FPS per Camera in 'S02'; $N = 4$; Density: 10.9 obj/sec; Arrival: 1.25 obj/sec

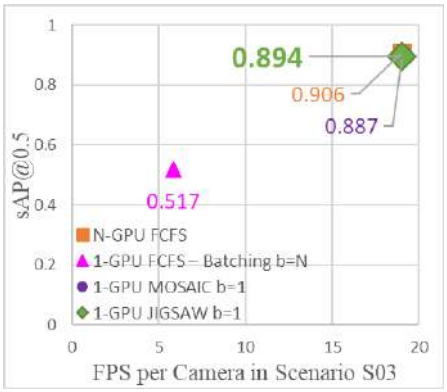

Figure 4.8: sAP@0.5 vs. FPS per Camera in 'S03'; $N = 6$; Density: 3.65 obj/sec; Arrival: 0.15 obj/sec

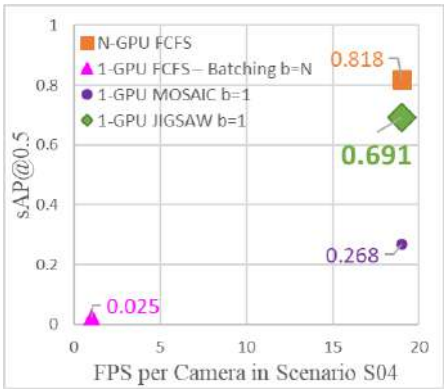

Figure 4.9: sAP@0.5 vs. FPS per Camera in 'S04'; $N = 25$; Density: 5.58 obj/sec; Arrival: 0.39 obj/sec

I note that MOSAIC and *JIGSAW* have comparable performance when the number of cameras/objects packed onto the canvas frame is low. Figures 4.6, 4.7, and 4.8 all show that for $N = (5, 4, 6)$ for Scenario 'S01', 'S02', and 'S03' respectively, *JIGSAW* and MOSAIC achieve comparable results due to the fact that *JIGSAW* is able to squeeze all mandatory and most optional tiles onto the canvas frame within their spatial sizing bounds. Qualitatively, I observe that *JIGSAW* packs $\sim 32\%$ less tiles than MOSAIC due to the cross-camera per-object matching and utility evaluation, thus allowing each packed tile to acquire slightly larger dimensions in-turn resulting in a $\sim 1-4\%$ accuracy gain over MOSAIC across Scenarios 'S01', 'S02', and 'S03'. On the other hand, when the number of cameras/objects mapped to a canvas frame is high (Scenario 'S04', where $N = 25$), Figure 4.9 shows that *JIGSAW* outperforms

MOSAIC significantly, achieving 42.3% higher streaming AP, even though both sustain a high processing throughput of 19 FPS per camera. Qualitatively, I note that at high workloads from $N = 25$ cameras, *JIGSAW* discards a slightly higher percentage of 48% tiles for inclusion on the canvas frame. This is due to the age metric of the utility function which when unable to pack *all* mandatory tiles from $N = 25$ cameras prioritises tiles containing older OoI. This forces the Dynamic Scheduler to evaluate newer representations (when available) for OoI that are yet to be evaluated for inclusion on the canvas frame, causing further discard of older representations as each OoI *waits* for its inclusion on the canvas frame. As anticipated, MOSAIC's inability to discriminate and differential schedule/discard tiles causes it to unfairly squeeze each tile beyond its spatial size bounds, thereby leading to a significant loss in object detection confidence (and resulting sAP). *JIGSAW* on the other hand strikes a fine balance between the need to achieve high detection accuracy and maintain streaming throughput, taking advantage of cross-camera overlap to often avoid redundant processing. In effect, *JIGSAW* is able to achieve accurate detection/tracking with a **far higher camera capacity** (at least 4-fold) than any prior technique employing either FCFS scheduling or spatial multiplexing on a **single edge device**, without the need for offloading camera streams to the secondary device.

Finally, I observe that *JIGSAW's* performance (using a single GPU) is roughly comparable to that achieved by the N-GPU FCFS baseline, with a < 0.127 loss in sAP even for $N = 25$ cameras. In effect, *JIGSAW* provides significant resource savings by multiplexing 25 streams on to a single GPU, compared to N-GPU FCFS (which requires 25 edge devices).

4.3.2 *JIGSAW's* Performance in Wireless Networks

I evaluate *JIGSAW's* performance in wireless networks where the edge requests frames from the cameras when the DS determines that a canvas frame should be constructed for inference. Employing the Python libraries of imagezmq [23] for

	'S01' $N = 5$		'S02' $N = 4$		'S03' $N = 6$		'S04' $N = 25$	
	sAP@0.5	FPS	sAP@0.5	FPS	sAP@0.5	FPS	sAP@0.5	FPS
FCFS N-GPU	0.548 (↓ 0.15)	18 (↓1)	0.434 (↓ 0.199)	18 (↓1)	0.861 (↓ 0.045)	18 (↓1)	0.708 (↓ 0.11)	18 (↓1)
FCFS Batching 1 GPU	0.144	5	0.065	6	0.517	4	0.025	1
MOSAIC 1 GPU	0.532 (↓ 0.148)	16 (↓3)	0.421 (↓ 0.169)	16 (↓3)	0.849 (↓ 0.038)	16 (↓3)	0.197 (↓ 0.071)	15 (↓4)
JIGSAW 1 GPU	0.557 (↓ 0.166)	16 (↓3)	0.436 (↓ 0.203)	16 (↓3)	0.857 (↓ 0.037)	16 (↓3)	0.676 (↓ 0.015)	15 (↓ 4)

Table 4.1: *JIGSAW's* wireless system design: achievable streaming accuracy and throughput - values in brackets indicate differences with wired system results

frame transfer and simplejpeg [46] for frame encoding, the cameras compress the individual frames for wireless transmission. The DS factors in the additional wireless Round Trip Time (RTT) ($\sim 47 - 52ms$) and JPEG decoding time (4.05 ms) into its evaluation for this *pull-based* system design. At the cameras, if the request from the edge arrives before half the inter-frame interval, the camera transmits the previous frame. This enables faster frame availability at the edge, yielding some processing latency savings and reducing GPU idle time. Conversely, if the camera receives the request after half the inter-frame interval, the camera waits for the fresher frame to be captured for transmission. This decision comes at some processing latency cost as canvas construction may not remain asynchronous (i.e. GPU may be idle for some time), however, this cost is acceptable as the resulting sAP is not adversely impacted owing to the receipt of the freshest frame.

With such system modifications for wireless operation, I observe that *JIGSAW* continues to outperform baselines in each scenario although the overheads from (i) wireless RTT and (ii) wait time at the camera for a fresher frame effectively reduce the achievable throughput at the edge (down to 15-16 FPS per camera) with minor $\leq 1\%$ reduction in achievable streaming AP, detailed in Table 4.1. Some loss in streaming AP can also be attributed to the JPEG encoding at the camera, and I note that additional optimizations at the camera such as multi-resolution image transfer could recover some of the reduction in sAP [146, 74]. Lastly, I see that the impact of wireless overheads on sAP are felt more keenly for faster-moving dense

highway-based traffic in Scenarios 'S01' and 'S02', whereas the less dense residential scenarios 'S03' and 'S04' which qualitatively feature more intersection/paused traffic are not impacted.

4.3.3 System Overheads and Scalability

I characterise the system overheads to better understand system behaviour. On average, transmission of compressed JPEG images takes 47-52 ms, JPEG encode/decode takes 4.05 ms, and tiling and canvas construction takes 12-19 ms (sub-linear increase for higher N).

4.3.4 Ablation Studies

1. *JIGSAW's* Impact on Streaming Object Recall: Figure 4.10 describes the number of canvas frames evaluated during every baseline evaluation and the resulting streaming average recall for Scenario 'S04' with $N = 25$. In the FCFS baselines, a single camera frame is placed onto the canvas frame for inference. The N-GPU FCFS baseline processes all camera streams independently and results in the most number of assembled canvas frames that require DNN inference. The 1-GPU FCFS-Batching baseline batches $N = 25$ cameras but suffers from a high DNN computation latency, limiting both the number of canvas frames assembled for evaluation and the resulting sAR. MOSAIC suffers from the "small object problem" due to its canvas construction methodology which packs *all* available tiles, also reducing sAR. Finally, *JIGSAW* provides the most desirable trade-off between number of constructed canvas frames from $N = 25$ cameras and the resulting sAR with a gain of 28.3% and 37.8% over MOSAIC and 1-GPU FCFS-Batching respectively. *JIGSAW* endures a $\sim 14.2\%$ loss in sAR, when compared to the resource-demanding, N-GPU FCFS baseline. This is due to the cyclical gate-keeping by the tile utility function that prioritises older and higher detection utility tiles. However, this loss is decidedly modest when I consider that *JIGSAW* evaluates only a fraction i.e. 28.5% of the total number of

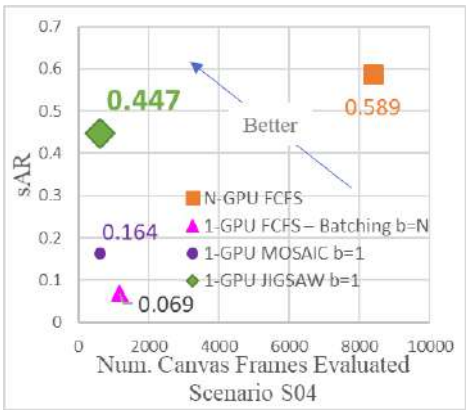

Figure 4.10: sAR vs. Number of Canvas Frames Sent for DNN Inference in 'S04'; $N = 25$

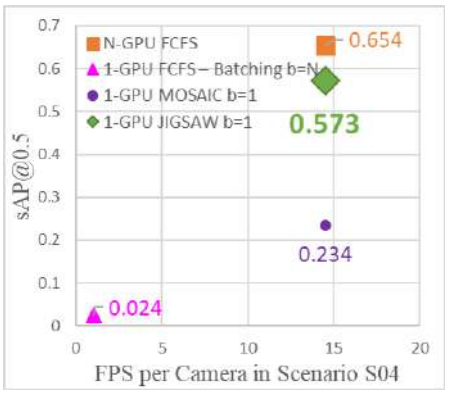

Figure 4.11: sAP@0.5 vs. FPS per Camera in 'S04'; $N = 25$ in a simulated bandwidth constrained wireless deployment

canvas frames on a single GPU when compared to the N-GPU FCFS baseline (which processes 100% of all arriving frames)–in other words, *JIGSAW* has a much higher *computational efficiency*.

2. *JIGSAW's* Streaming Detection Metrics vs Classical Detection Metrics: I also perform standard evaluation across all 4 scenarios using the mean average precision (mAP) and average recall (AR) metrics on the canvas-frames constructed by *JIGSAW*. This evaluation matches the post-processed translated predictions and the groundtruth by the frame index. I observe that *JIGSAW* results in an mAP value of 0.972 at the IoU threshold of 0.5 and an AR value of 0.934. A standard FCFS evaluation on all 4 scenarios of CityflowV2 on the other hand yields an mAP value of 0.82 and an AR value of 0.8. I conclude that *JIGSAW's* nuanced tiling and tile utility maximization contribute strongly to providing better object detection fidelity, along with a high streaming throughput.

3. System Performance in Heterogeneous Wireless Environments: I simulate a heterogeneous, bandwidth constrained wireless environment for Scenario 'S04' (with $N = 25$ cameras, illustrated in Figure 4.11) where half of the cameras drop every second frame to achieve an input rate of 5 FPS, while the other cameras maintain the original rate of 10 FPS. I evaluate *JIGSAW* on this deployment and report 8.1% loss in sAP over the $N = 25$ cameras from the N-GPU FCFS baseline. This is

	Tile Selection Overhead	sAP@0.5	Tile Selection %
CrossROI – Largest Tile	0 ms	0.627	62.86%
MOSAIC	0 ms	0.68	100%
JIGSAW	2.3 ms	0.723	68.04%

Table 4.2: Comparison of *JIGSAW's* tile utility-based selection of mandatory tiles with alternative paradigms for Scenario 'S01'

due to the IoU-based evaluation of the sAP metric, which suffers especially for low FPS streams. However, *JIGSAW's* DS and tile utility maximization techniques both work in tandem to recover 33.9% and 54.9% streaming accuracy, when compared to MOSAIC and the 1-GPU FCFS Batching baselines respectively, while also gaining an 14x or 1400% gain in throughput to 15 FPS per camera (cumulatively 375 FPS) when compared to the 1-GPU FCFS Batching baseline.

4. Largest Tile vs Tile Utility: I evaluate *JIGSAW's* selection of mandatory tiles by order of tile utility to the streaming perception task for canvas construction by comparing against alternative schemes such as (a) CrossROI [56], which selects the largest representation of a unique object for inference, and (b) MOSAIC which selects all available tiles for inclusion into the canvas frame. Detailed in Table 4.2, compared to (a), *JIGSAW* offers 9.55% improvement in task accuracy with only 2.3 ms higher tile processing overhead, while compared to (b), *JIGSAW* effectively reduces the number of tiles/pixels contending for DNN inferencing by 32% while achieving an accuracy gain of 4.3%. *JIGSAW* selects tiles based on their utility to the streaming perception task in context of the estimated confidence of the object:tile pair and encompassing tile's canvas utilisation, thus optimizing *which* tiles are "squeezed" onto the canvas frame. I conclude that in *JIGSAW's* paradigm, the "best" representation of a unique object may not always be the "largest" representation.

4.4 Discussion

Camera Workload Offloading Decisions for System Scalability: *JIGSAW's* design allows for system scalability with an on-demand secondary edge device that

can be triggered when object density across N cameras goes above the threshold (empirically determined) of 250 on the primary edge device. Offloading decisions pertaining to "which" camera streams to offload and "when" need to consider the following. First, clusters of cameras which share spatial overlap between its fields of view must be considered as a single "unit" to allow the pipeline to achieve throughput and accuracy gains facilitated by the selection of mandatory tiles across this set of cameras. Second, these camera clusters must be ranked by a metric that evaluates the achievable workload reduction (a function of the number of cameras per-cluster, and number of unique objects per-camera) at the primary edge device and facilitates a cut-off point to prevent further offload once an acceptable workload remains mapped to the primary edge device. One approach is to offload by most-dense-first, evaluating camera clusters with high per-camera and cross-camera workloads for transfer to the secondary device, with the objective of reaching workload stabilisation in the fastest manner. However, further offloading decision optimizations can be made based on the motion of the objects themselves, discussed next.

Considering OoI Motion Patterns: In addition to varying camera workloads over time, another perspective on temporal variability of sensed objects in a camera FoV is that different objects may arrive and traverse the sensing field at different physical velocities resulting in variable pixel displacements between successive frames that contain the object. Streaming detection accuracy may be higher for slower moving objects which exhibit lesser pixel displacement between successive frames and vice-versa for high-velocity objects. Modelling both the arrival and inter-frame movement pattern of individual objects could help further fine-tune the selection of high-priority tiles across all objects sensed from N cameras. For example, the system can determine that a stationary object might not benefit from frequent DNN inference and de-prioritise (or increase the objects' age) in favour of higher-velocity objects for inclusion onto the canvas frame. Over standard RGB camera streams, such velocity estimation is dependent on variables like object class, object-camera distance, camera extrinsic calibration, and might benefit from the integration of physical models

to model object motion. To preserve high DNN inference throughput, any such sophisticated modeling will need to remain computationally lightweight. On the other hand, high-temporal resolution event streams obtained from neuromorphic event cameras naturally lend themselves to modeling motion through optical flow estimation over OoI events. I discuss the integration of event cameras to determine OoI motion criticality in Chapter 5.

Motion modeling can also optimize workload offloading decisions to a secondary edge device. For example, a camera stream observing largely stationary objects, such as in a parking lot, can be considered to present *low* workloads at the edge as temporal delays in including the object onto the camera frame does not adversely impact streaming accuracy. On the other hand, a camera stream observing low density but fast-moving objects, such as vehicles on a highway, can be considered to present *high* workloads at the edge, and will have to prioritize all available OoI consistently for accurate streaming perception. A more nuanced metric considering both object density and object motion or arrival patterns would then fairly redistribute workloads between the primary and secondary edge device to not just optimise for available workload but also achievable streaming perception accuracy.

Bandwidth-Aware Frame Transfer Mechanisms: While the pipelines described in this chapter focus on a wireless pull-based design to contend with variable network latency, additional modifications could facilitate Bandwidth Adaptive Camera Transmission (BACT) techniques described in Chapter 3. Such modifications could operate in tandem with the Dynamic Scheduler-based admission control mechanism to further optimize transmission in the context of streaming perception. For example, specific/unique object-level resolutions could be optimized to carry a higher priority/-fidelity/resolution during camera transmission if the Dynamic Scheduler determines that it has not been scheduled for canvas construction for a while. I leave further characterisations for future work.

***JIGSAW* over Cameras in Motion:** Extending the notions of edge-based optimized stream processing to multiple non-stationary camera sensors, where spatial overlap

is transient and dynamic, remains an open research problem. Additionally, I believe that cameras with ego-centric motion, such as those mounted on an autonomous car or drone, are more likely to exhibit temporal correlation (an object appear in the FoV of different cameras at different time instants) rather than concurrent spatial overlap. I hypothesize that *JIGSAW's* tile utility function would have to incorporate such temporal mappings between cameras and their sensed objects to add another layer of prioritization of tiles from different cameras onto a canvas frame.

Enabling Multi-Target Multi-Camera (MTMC) Tracking: MTMC Tracking would certainly be a logical extension to *JIGSAW's* system design. In the current design, *JIGSAW* implements a Kalman Filter-based centroid tracker at the edge for maintaining the association between matched objects across cameras as advised by the Cross-Camera Tile Mapping database. MTMC tracking on the other hand, has been shown to perform better with the use of deep feature embeddings derived from models such as DeepSort [144]. However, DNN based preprocessing approaches will incur additional processing latency at the edge, thereby negatively impacting both the streaming detection accuracy and achievable throughput. I speculate that future works on *JIGSAW's* design could benefit from a more integrated sensor-edge design of MTMC tracking [134], with Multi-Target Single Camera detection and tracking performed at a compute-enabled camera, and cross-camera MTMC evaluation performed at the edge. Future work could also consider cross-camera reID refinement strategies as described in [50] to extend the matching of objects across even those cameras that do not share any spatial overlap with the source camera.

Chapter 5

Exploring Neuromorphic Event Cameras for Canvas Construction: *TANDEM*

In this chapter, I consider the challenge of supporting visual perception on a multi-tenancy edge device where each sensing stream consists of a combination of a CMOS camera and a biologically-inspired neuromorphic event camera. This combination is intrinsically appealing because both types of cameras represent two extremes in sensing power consumption, sensing throughput, and sensing fidelity in the quality of images captured, which an intelligent edge device can leverage to moderate the overall (perception throughput, energy, accuracy) trade-off. To elaborate, recent advancements in camera quality or resolution pose increased energy consumption during operation, creating a bottleneck to low-power efficient sensing on edge devices. For example, Arducam's latest 64MP camera [12] consumes $\sim 2W$ power to generate high-resolution images of 1270×720 resolution at 120 FPS, which a DNN can infer over with high task accuracy. This could amount to significant power consumption on the edge device for perception pipelines relying on multiple concurrent CMOS cameras. In comparison, biologically-inspired neuromorphic event cameras mimic the human retina to provide extremely low power ($10 - 30mW$), highly reactive

$O(\mu s)$ sensing capabilities, higher dynamic range up to $140dB$, showing competitive performance in DNN task accuracy against CMOS cameras [120] for tasks such as object recognition and tracking. In Section 5.1.1, I explain how event cameras move away from the CMOS sensor and concept of a "frame" to capture delta changes in light intensity (both positive and negative changes/polarity) incident on every pixel asynchronously. Further, in Section 5.1.2, I describe how 2D "images" can be synthesized/aggregated from a continuous stream of events that are reported as a tuple of ($pixel_x$, $pixel_y$, timestamp, polarity). Such "images" unlike a CMOS frame, capture low-dimension framed representations of the sensing field, encapsulating only objects/edges that are in motion without defining nuances such as colour or texture. These "images" or framed representations of the event stream are used as inputs to an off-the-shelf DNN, leveraging decades of computer vision research on CMOS frames, but suffer from lower DNN task accuracy due to the absence of nuances such as texture and defined object edges. Latest advancements in event cameras have introduced hybrid sensors such as Inivation's DVS346 [3] that are capable of producing both low spatial resolution gray-scale frames and high temporal resolution event streams, illustrated in Figure 1.5. Low-power low-dimension event representations capturing motion in the sensing field and feature-rich high-dimension power-consumptive standard/grayscale frames represent two extremes of sensing fidelity, presenting an opportunity for perception pipeline to leverage these sensing trade-offs to balance energy-aware optimizations with fast and accurate perception at the edge.

5.0.1 Employing Fused CMOS+Event Streams at the Edge

I now introduce *TANDEM*, a novel perception pipeline that leverages both camera and event sensor streams to jointly contribute to accurate low-power low-latency visual perception. *TANDEM* leverages *multiple* DVS346 CMOS+Event streams to utilize their complementary feature and energy characteristics for optimised and

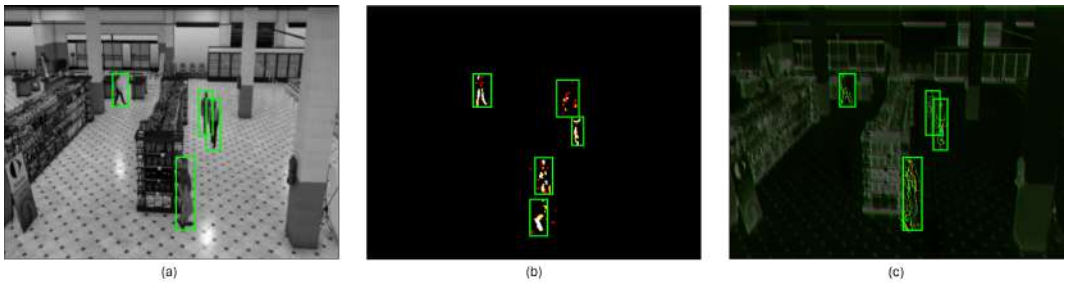

Figure 5.1: Differences in object detection capabilities over (a) Pure CMOS stream captured at 30 FPS (high accuracy) (b) Pure event stream captured at 100 FPS (low accuracy from uneven object edges due to occluded/slow-moving objects) (c) *TANDEM's* fused CMOS+Event representations with CMOS streams captured at 30 FPS and Event streams at 100 FPS (recovered object detection capability)

efficient vision perception at the edge, providing multiplicative gains in processing throughput with negligible loss in DNN task accuracy.

The first aspect of *TANDEM's* system design focuses on high fidelity sensing balanced between the two extremes provided by frame cameras and event cameras. Feature-rich frame cameras, generating frames at 30 FPS, provide crucial texture/edge information for accurate object detection at the cost of high energy consumption $(1 - 2W)$ for sensor operation, illustrated in Figure 5.1 (a). The low power consumption and high $O(\mu s)$ temporal resolution of the asynchronously generated event camera streams yield fast object understanding with reduced DNN accuracy due to the lack of defining object features resulting from occlusions and/or partial motion, illustrated in Figure 5.1 (b). *TANDEM* intelligently orchestrates and fuses the sensor streams, illustrated in Figure 5.1 (c) to compensate each other *on demand* to balance between power consumption and sensing fidelity, triggering the power-consumptive high-dimension CMOS camera only when object uncertainty from event streams increases above a predefined threshold, while simultaneously leveraging the lower latency of the event stream to push the envelope on perception throughput. Multiple such pairs of CMOS+Event streams thread the gap between (achievable throughput, energy efficiency, sensing fidelity) to jointly provide feature-rich multi-sensor perception. Such high throughput, low power, high accuracy support is critically needed for certain high kinematic machine perception tasks such as autonomous

vehicle perception and interaction between humans and assistive robots in high-risk environments such as handling high-velocity machinery.

The second aspect of *TANDEM's* system design focuses on the efficient processing of these CMOS+Event pairs of sensor streams. *TANDEM* leverages the *Canvas-based Processing* spatial multiplexing paradigm introduced in Chapter 2 adapted to the sensor fusion across both sensor streams. Traditionally, canvas-based processing extracts critical Regions-of-Interest (RoI) across multiple CMOS camera input streams and spatially multiplexes (or equivalently, tiles) these RoI on a "canvas frame" for dramatic gains in processing efficiency with negligible loss to task accuracy. In adopting this computation paradigm, *TANDEM* addresses two challenges: (i) many-to-one mapping of event streams to an DNN with RoI CMOS+Event tiles extracted, compressed, and spatially multiplexed onto a canvas frame for inference, and (ii) interleaving of two disparate sensing modalities on a single "canvas" for processing using a single DNN on an edge device.

5.0.2 Key Contributions

I make the following key contributions:

- **Optimised Ingest Pipelines:** I describe optimized ingest pipelines which pre-process multiple high volume event data streams (at 100 FPS) intelligently into framed representations for DNN processing. Event stream volumes are evaluated upon ingest for suitable Random Downsampling to reduce the volume of events to $\leq$ 20000 events within one event window of $t = 10ms$. These downsampled events are constructed as 12-channel Voxel Grid framed representations within $6.35 - 7.4ms$, well before the next window of events is ready for ingest at the edge.

- **Asynchronous On-Demand Fusion of CMOS Frames and Events:** *TANDEM* leverages the hybrid nature of DVS346 cameras to ingest grayscale frames at standard frame rate (i.e. $\leq 30FPS$) and event streams quantized

for processing at 100 FPS. While both sensor streams are independent of each other, *TANDEM* intelligently fuses both sensor streams *asynchronously* and at its received frame rates into a common representation that incorporates information from both sensor streams for a joint representation of the sensed world. I describe a simple 3-layer Multi Layer Perceptron network that facilitates such fusion at disparate framerates. I also characterise an Uncertainty Estimator which evaluates the framed representations from the event stream for its utility to canvas construction and triggers the fusion of a CMOS frame on-demand when the estimated uncertainty increases above a pre-defined threshold. In Figure 5.2, I show the trade-offs for DNN inference on (i) pure event streams captured at 100 FPS (ii) standard CMOS streams captuerd at 30 FPS and (ii) *TANDEM* operating on event streams at 100 FPS and CMOS streams at 30 FPS. I note that *TANDEM* operates between the two extremes presented by pure-event processing (low DNN task accuracy 43.2%, low power consumption $0.71W$, high perception throughput 89 FPS) and pure-CMOS processing (high DNN task accuracy 99.1%, high power consumption $0.91W$, low perception throughput 30 FPS) to provide superior gains in (high DNN accuracy 90.8%, low power consumption $0.71W$, high perception throughput 89 FPS). In Section 5.4, I show that such *on-wake* design of the more power-consumptive CMOS sensor yields high perception throughput of 89 FPS and 22.22% power-savings at the cost of 7.2% object detection accuracy over constructed canvas frames from $M = 1$ DVS camera, when compared to canvas frames constructed over low throughput 30 FPS pure-CMOS streams. Potential design changes to improve the achievable DNN accuracy by enabling fusion of "stale" CMOS frames with current event representations are discussed in Section 5.5. On the other hand, *TANDEM* yields high throughput of 89 FPS, higher 48.52% power-savings, at the cost of 8.1% object detection accuracy over constructed canvas frames from $M = 1$ DVS camera, when compared to canvas frames constructed over standard RGB cameras capturing frames

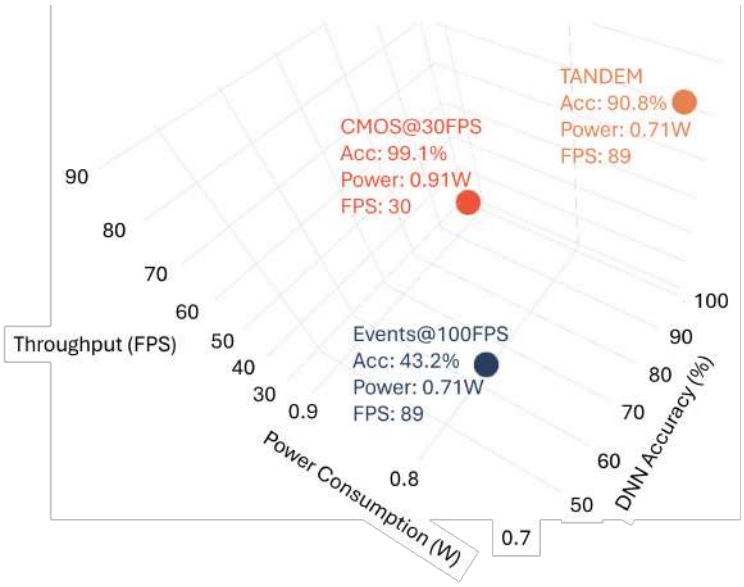

Figure 5.2: *TANDEM* Performance gains over perception throughput, power consumption, and DNN accuracy

at 30 FPS. This indicates that by replacing a standard RGB CMOS camera, *TANDEM* enables energy savings of $0.66Wh$ for every DVS346 camera that is run at the edge for 1 hour continuously. For a Jetson Orin AGX edge device receiving $M = 10$ CMOS+Event sensor streams, *TANDEM* therefore presents energy savings of $6.6Wh$ over an hour of continuous operation and $158.4Wh$ for 24 hours of continuous operation.

- **Spatial Multiplexing of Multiple CMOS+Event Sensor Streams:** *TANDEM* decomposes the fused CMOS+Event representations from each DVS346 camera into a "bag of tiles" for canvas-based processing. With a representative person detection-based surveillance application (described in Section 1.2) using the CityFlow AI Track 1 dataset [44], I show how canvas-based processing of fused representations of (intermittent/triggered CMOS frames, periodically spaced Event representations) from $M = 10$ DVS346 cameras provides multiplicative gains of $88\times$ or 888.88% increase in processing throughput on a single Jetson Orin AGX device at 89 FPS *per-camera* and 890 FPS cumulatively with a minor loss of 1.7% in object detection accuracy.

5.1 Motivating *TANDEM's* Design Decisions

Before I introduce the canvas-based perception pipeline that incorporates event cameras, I first introduce the nuances in an event camera's operation, and present two foundational studies that underpin the design decisions adopted in this work.

5.1.1 Introducing the Event Camera

Biological vision systems operate without the frame-based structure typically seen in CMOS sensors of RGB cameras. Photoreceptors, specifically rods, present in the retina instead actively observe the physical environment and produce activations or spikes when they detect changes in light intensity. This spike generation occurs continuously and asynchronously, triggered by high spatial or temporal contrasts in the scene. These spikes are then transmitted via the optic nerve for processing in the visual cortex. Neuromorphic event cameras mimic the biological retina by reporting changes in light intensity (both positive and negative) on a *per-pixel, asynchronous* basis yielding a "stream" of sparse spikes that correspond best with moving objects that create changes in contrast/light intensity, illustrated in Figure 5.3. Each pixel (x, y) measures the log intensity of light incident upon it $L = ln(I)$ until the brightness changes with positive or negative polarity $p \in \{-1, 1\}$ (also referred to as "ON" events and "OFF" events respectively) over any delta period of time Δt exceed a pre-defined threshold θ as:

$$L(x, y, t) - L(x, y, t + \Delta t) = p\theta \tag{5.1}$$

Each pixel independently evaluates and fires a spike every time the brightness changes over threshold $p\theta$ with the data representation of (x, y, t, p), otherwise known as Address Event Representation. The minimal overheads incurred from evaluating each pixel and generating such an AER data point allows event cameras to (i) operate with microWatt power consumption, (ii) avoid the latency induced by a synchronous sensor that waits for all pixels to accumulate a change, (iii) allows a

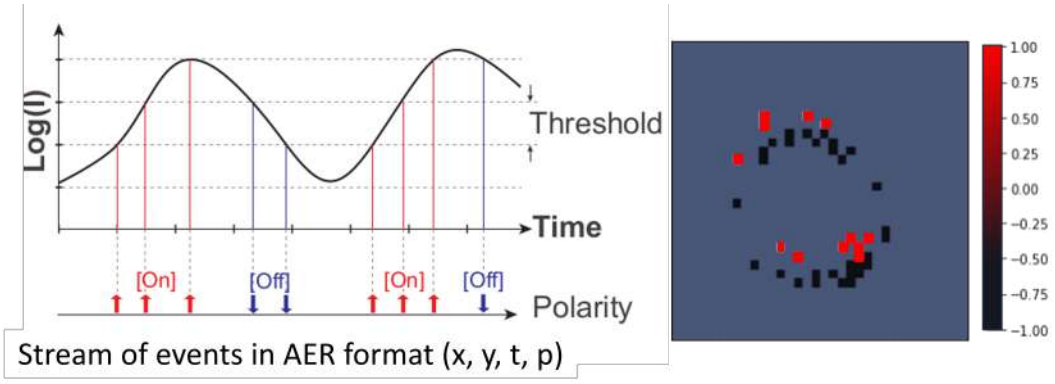

Figure 5.3: Event Cameras: Generation of AER data based on thresholded log luminescence at each camera pixel (left) with a frame-representation constructed with events of both polarities over a quantized period of time T (right)

high dynamic range owing to the logarithmic conversion of the intensity values, and (iv) reduces bandwidth consumption as a relatively stable intensity value (caused by little or no motion in the receptive field of the pixel) yields fewer events. On the other hand, highly dynamic or textured scenes might yield a temporal resolution up to 1.06 Geps (giga events per second), resulting in a computational load that can easily overwhelm desktop GPUs and resource constrained edge devices like the NVIDIA Jetson class of devices. In addition to this scale of computational load, the main drawbacks of event cameras lie in (i) the absence of defining edge/colour/texture features in event streams leading to poor object detection task accuracy, (ii) pixel noise characteristics, and (iii) the challenge associated with reconstructing precise intensity representations (similar to RGB frames) from delta-based events. Despite this limitation, event cameras find practical use in scenarios where capturing the edges of moving objects is crucial. This is particularly evident in robotics and navigation applications where rapid response times take precedence over high human visual quality. Examples of such applications include object detection and tracking, gesture recognition, and SLAM (Simultaneous Localization and Mapping).

5.1.2 Processing and Inferring over Event Streams

Recent works which explore the processing of event streams can be largely divided into two categories with diverging principles.

The first category of works choose to remain true to the biological inspiration behind the event cameras and consume the event stream as a "spike train" for processing by a Spiking Neural Network (SNN). SNNs are modelled after the membrane potential based spike-activation of the human neuron, and have been shown to effectively process high temporal resolution event streams to surpass RGB-frame based DNN processing for tasks such as object recognition and tracking [40, 102]. The spiking neurons maintain a memory of input events in a Leaky Integrate and Fire (LIF) model *over a set of timesteps* (ranging from 4 - 3000 [133]) and fire a spike when the membrane potential crosses a predefined threshold. At each time step, the SNN forward pass yields a set of output spike activations over a quantized section or chunk of the event stream which yield fast, efficient, and low-power task inference on custom neuromorphic chips such as Intel Loihi [69] and IBM NorthPole [67]. While this approach remains true to the low-power and low-latency biological primitives that underpin the design of the event camera, SNNs show significant challenges to trainability. SNNs also show poor performance when compared to classical DNNs for regression-based tasks such as object detection due to the non-differentiable nature of the spike trains and activations [52]. Although recent SNN models such as EMS-YOLO [133] have considerably narrowed the performance gap, showing comparable accuracy with DNNs while consuming $5.83\times$ less energy, they still rely on the deployment of custom neuromorphic SoCs which present hardware limitations on the scale, complexity, and processing throughput of the SNNs, especially at the edge.

The second category of works choose to default to the known medium of 2D frames or images that are synthesized from a continuous temporal stream of events, by quantizing the stream into time windows and aggregating all events to generate

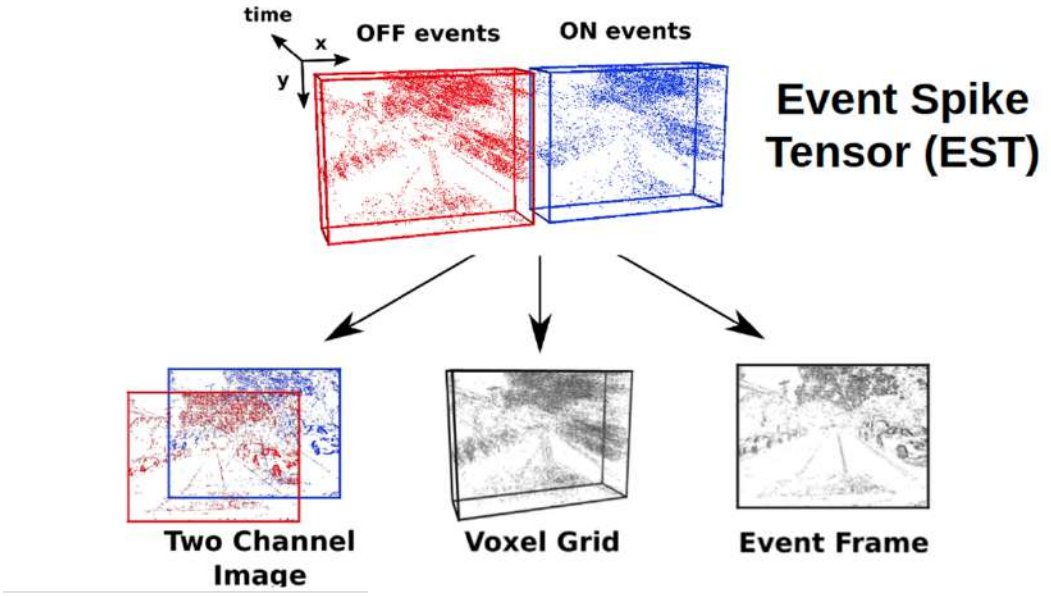

Figure 5.4: Event Camera: Synthesizing 2D representations from event streams [51]

a 2D representation of the sensed physical world as described in Chapter 1 and illustrated in Figure 5.4. These 2D framed representations depict the sensed physical world as low-dimension images, devoid of colour or texture, capturing only those pixels where significant movement or changes in light intensity occurred. Such 2D frames are then inferred upon by standard off the shelf DNN models deployed on standard GPU devices, leveraging years of progress in computer vision research. This approach presents attractive advantages in the reuse of existing perception pipeline components built for traditional frames/cameras. However, converting event streams into a 2D frame representation imposes a non-trivial pre-processing overhead. This overhead is caused by the CPU having to cycle through all available events, within an inter-frame intervals, for the construction of the 2D framed representation.. Such processing also discards/discounts the high temporal resolution and per-pixel asynchronicity provided by the raw event stream. However, I believe that this cost has to be measured against the motion of the objects of interest, whether this motion needs to be represented at $O(\mu s)$ latency, and whether quantization into framed representations at high frame-rates (e.g. 100 FPS or 10 ms latency) can still recover perception capability for said objects in the context of the deployed application.

Studies in both approaches have thus far been conducted on server-class or desktop-class GPU devices, with no consideration for design trade-offs presented by limited computational resources at the edge. To appropriately ingest high volume event streams at the edge, I believe that a perception pipeline must choose whether (i) events should be downsampled and pre-processed into lower volume spike trains for inference by an SNN, or (ii) events should be downsampled and pre-processed into 2D framed representations to be processed by an off-the-shelf DNN. This choice must be made depending on the throughput-vs-accuracy tradeoffs incurred by both classes of event processing pipelines at the edge. I therefore first characterise the processing overheads presented by both approaches at the Jetson AGX Orin, a representative edge device to present the trade-offs between processing throughout and achievable task DNN/SNN accuracy. Lastly, I present the design decisions on how event streams can be best consumed by a perception pipeline at the edge.

Study 1: Processing Overheads for Raw Event Streams

Frame-space vs Event-space Pre-processing: Although event camera streams are sparse by nature, responding only to changes in the environment, highly dynamic scenes that feature considerable movement or changes in contrast might yield 1 Mega events per second [3] up to 1.06 Giga events per second [4]. The objective of this study is to evaluate (i) whether it is computationally cheaper to pre-process events into a smaller volume in frame space or in event space on a resource constrained edge device, and (ii) if there are any differences introduced by the two pre-processing methods which impact the achievable task accuracy due to the loss of temporal resolution or event asynchronicity when condensing events into a 2D framed representation. I characterise the processing latency-vs-accuracy achieved in pre-processing raw events into a smaller volume of events in:

1. Event space: Raw AER events are converted into spike tensors and pre-processed to a lower event volume using spiking Bilinear and Bicubic in-

terpolation [55]

2. Frame space: Raw AER events are aggregated into frames and downsampled into a lower spatial volume using standard OpenCV interpolation techniques such as Bilinear and Bicubic Interpolation [113]

To this end, I employ the NMNIST [115] dataset which is the spiking event version of the original frame-based MNIST dataset at 34×34 spatial resolution. 60000 samples from the dataset are loaded and prepared for processing, with 10000 frames used to evaluate the task accuracy on the Jetson AGX Orin edge device. As both frame-based and event-based representations can be consumed by an SNN (as opposed to a DNN that can only consume frame-space representations), I employ a simple convolutional SNN comprised of Leaky Integrate and Fire neurons to compare object recognition accuracy using both pre-processing methods. The SNN model, implemented using the snnTorch library [42] and illustrated in Figure 5.5, comprises of 2 convolutional layers, 2 max pooling layers, and 2 fully connected layers for object recognition, and ingests a 34×34 spatial resolution input over 128 timesteps to recognize the object. I evaluate the accuracy of the object recognition task by observing the mean average precision (mAP).

As the original NMNIST resolution and the object recognition SNN are both of 34×34 spatial resolution, I simulate higher event spatial volumes by upsampling the event stream to 128×128 spatial resolution. For frame-based evaluation, I construct 2D frames by aggregating all events over both polarities into a single frame. I then downsample the events to 34×34 using classical RGB-frame based Drop, Bilinear, and Bicubic interpolation (from the OpenCV vision library). For event-space evaluation, I employ the log-luminescence based Linear and Bicubic Estimation as the event-space downsampling methods described by Gruel et. al. [54].

To evaluate whether both methods introduce any differences that impact task accuracy, each recording in the NMNIST dataset is zero-padded to equal duration of $300ms$ and divided into bins of time duration $t \in (2.34, 4.68, 9.37, 18.75)$ mil-

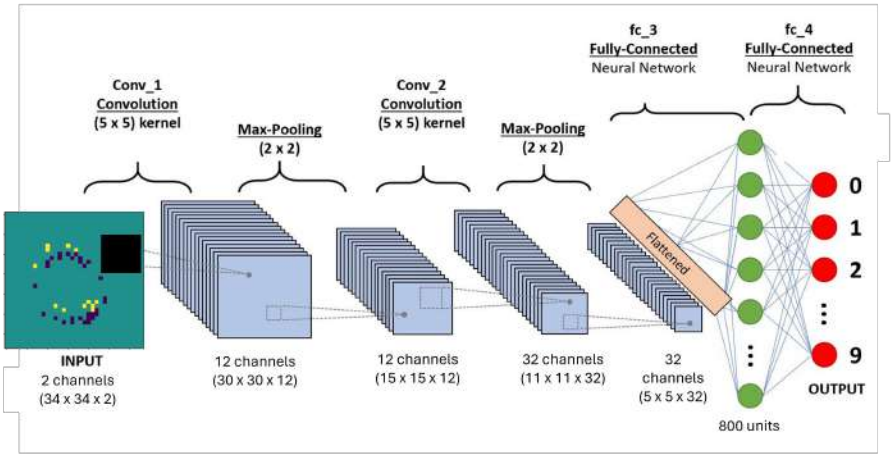

Figure 5.5: 4-layer Convolutional Spiking Neural Network (SNN) for Object Recognition on 34×34 input spike trains from the NMNIST dataset

liseconds respectively, each bin representing a different event spatiotemporal volume for SNN inference. I hypothesize that recognition accuracy for events may vary for different segment-lengths of event bins that contain varying amounts of events/information over time, which could be impacted by both event pre-processing methods in different ways.

I now describe the tradeoffs between frame space (FS) and event space (ES) pre-processing of event streams of different spatiotemporal event volumes. Figures 5.6 and 5.7 describe the impact of both methods of downsampling on object recognition. I make the following observations:

- Both frame space (FS) and event space (ES) methods have comparable SNN task accuracy which leads us to believe that event space pre-processing has (i) no clear benefits of processing in event space even at a temporal resolution of $18.75ms$ (or equivalently, no apparent impact on object recognition due to loss of asynchronicity at this temporal resolution as seen in Figure 5.6).

- Frame-space pre-processing incurs significantly $3 \times -13\times$ less overheads when compared to event-space pre-processing as seen in Figure 5.7.

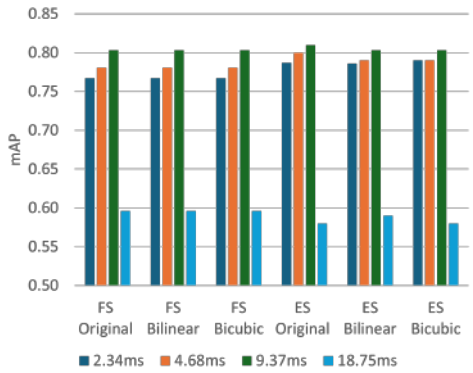

Figure 5.6: Comparison of event space (ES) and frame space (FS) compression methods over bins of time duration $t \in (2.34, 4.68, 9.37, 18.75)$ on recognition task accuracy.

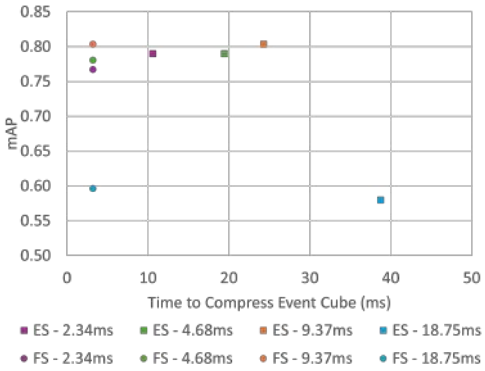

Figure 5.7: Processing overheads from event space (ES) and frame space (FS) Bicubic Compression over bins of time duration $t \in (2.34, 4.68, 9.37, 18.75)$ versus recognition task accuracy.

- Dividing an event stream recording into 32 bins of time-period $\sim 10ms$ for recognition seems to accumulate the "ideal" number of events on average in a frame, yielding higher recognition accuracy. This seems to suggest that there exists an ideal number of events that must be aggregated into one event bin (and by extension one generated frame-representation) for optimal recognition accuracy, which I believe is task and object velocity specific.

DNN vs SNN Inference: In addition to the clear advantages of pre-processing events in frame-space over event-space, I also evaluate the inference time for a forward pass of a standard DNN and of the 4 layer convolutional SNN employed in Study 1. On the Jetson AGX Orin, a YOLOv8n or an off-the-shelf nano-scale version of the YOLOv8 model takes a mere $11.38ms$ on average to infer over a 640×640 input image. On the other hand, one forward pass over 128 timesteps of the 4 layer SNN takes $\sim 800ms$ (although these are numbers obtained by software-based SNN emulation, due to the absence of neuromorphic hardware). While SNNs could incur significantly lesser inference latency if deployed on a neuromorphic SoC, the low processing overheads of frame-space event representations and the low inference latency of a YOLOv8n model together make DNN inference over frame-space representations the more attractive processing paradigm. This is especially given the

accuracy gains of a DNN (YOLOv3, mAP=0.31) over an SNN (SpikingDenseNet, mAP=0.18) when evaluated over the challenging pedestrian and vehicle *detection* dataset Gen1 by Prophesee [39].

Study 2: Processing Overheads for 2D Framed Representations

While Study 1 focuses on simple spatial interpolation for a naive 2D aggregated frame-space representation (hereafter referred to as a framed representation for simplicity), prior literature has in fact introduced a number of techniques to aggregate event data into a 2D frame, such as Voxel Cubes, Mixxed Density Event Stack, Time Ordered Surface of Events, and Averaged Time Surface to name a few [106, 82, 17, 36]. However, these works disregard real-time processing latency costs over high volume event streams and either (i) focus on offline applications which post-process large volumes of event data for analysis, (ii) assume the deployment of a server-class GPU, or (iii) characterise event pre-processing as an offline task. This leaves real-time event processing at the edge an open problem with no clear rationale on the superiority of one framed representation technique over the other, or design choices represented by each of the framed representations. In this study, I describe the design of a profiler and the trade-offs observed between the volume of events evaluated, pre-processing techniques used to reduce the volume of events, framed representation variant generated, quality of frame representation, and processing time incurred, to derive the pareto optimal setting for frame-space event processing on the NVIDIA Jetson Orin AGX [110] edge device. I present the operation of the profiler over the CityFlow AI [44] person detection dataset to align this study more closely with a potential edge-based application that leverages event camera streams.

Profiler Overview and Results: The profiler comprises of 4 stages, described below.

1. Simulating Events: The profiler first synthesizes event streams from RGB videos using the v2e (video to events) software tool [62], aggregating events at a user-defined temporal resolution (default, $t = 10$ms or 100 FPS as derived from Study 1) to evaluate the expected volume of events if an event camera were to replace

the RGB camera. The profiler's goal is to identify the best {framed representation, pre-processing technique} combination for the given dataset which provides the best quality of framed representation of the events before the next window of events is aggregated i.e. within t of 10ms. While I use $t = 10ms$ as a basis for this study, the profiler can also evaluate different configurations for time windows to understand whether $t = 10ms$ is the ideal event aggregation window for that particular dataset which best balances object kinematics, the resulting event volume generated, the quality of the framed representation generated, and the resulting downstream DNN task accuracy.

2. Applying Pre-processing Techniques: The profiler then evaluates sampling techniques to reduce the volume of events and the resulting processing time: (i) No sampling (ii) Spatiotemporal filtering (drop isolated events in a 1 pixel radius over 1ms window) (iii) Temporal downsampling (drop every 2nd event per-pixel) (iv) Random sampling (drop events with $p = 0.5$ probability), and (v) Spatial interpolation (bilinear interpolation to reduce spatial volume by 25%).

3. Creating Framed Representations: Next, the profiler loads the events in each time bin into memory to convert the events into one of the following event representations:

1. VoxelGrid [162]: $t = 10ms$ time window is split into 12 equal, non-overlapping time windows to create a 12 channel VoxelGrid. Events in each sub-window are aggregated by summing their polarity on a per-pixel basis using bilinear voting.

2. Binary Histogram [99]: Events in the $t = 10ms$ time window are split by polarity and aggregated into 2 channels.

3. Mixed Density Event Stack [106]: To create a 12 channel MDES, each channel aggregates events at different time scales within a $t = 10ms$ window. For every channel c, MDES selects the most recent $\frac{N_e}{2^c}$ events and aggregates the polarity at every pixel for the selected events, where N_e indicates the of events

in the selected time window.

4. Time Surface [82]: A time surface is a 2D grid that stores the timestamp of the last event that occurred at each pixel (x, y) location. By storing the timestamp of the last event, the time surface encodes the temporal history of brightness changes at each pixel. This allows the rich temporal information of the event stream to be preserved in a compact, image-like format. This time surface is then sampled at equally spaced intervals to create 12 channels.

5. Time Ordered Recent Events [17]: TORE stores event timestamps in per-pixel per-polarity queues and fivides them equally into 12 channels.

6. 12-Channel Learned Representation [163]: The representation is learned using a trainable kernel operation, optimised for object detection tasks.

4. Calculating Quality of Generated Representation: Lastly, the profiler leverages the Gromov-Wasserstein Discrepancy (GWD) as a metric for comparing the quality of event representations efficiently by measuring the distortion arising from the conversion of raw events to the framed representations. The GWD metric calculates the similarity between event and feature pairs during the construction of an event representation, with a lower GWD score indicating lesser distortion (or equivalent, better representation quality and preservation of events) and better downstream DNN task accuracy [163]. All representations feature 12 channels except Binary Histogram, which features 2. This is in line with the findings by Zubic et. al. [163] that suggest that a higher number of channels allows a learned representation to achieve lower distortion rates. The VoxelGrid, MDES, TORE, and TimeSurface representations are similarly constructed as 12 channel representations for fair comparisons.

Profiler Evaluation Results: In this study, I utilize the the CityFlow AI dataset [44] (1920×1080 resolution videos captured at 30FPS) for a person detection application. In lieu of comparing multiple datasets that generate different event volumes, I utilize the *same* dataset and vary the sensor resolution settings to control the average number

Framed Representation	Processing Time of Event Volume (ms)				
	10000	20000	30000	40000	50000
Voxel Grid	3.4	6.35	11.36	24.04	38.97
Time Surface [82]	34.82	45.63	52.49	91.15	128.4
Binary Histogram	3.17	4.38	6.81	14.94	21.87
MDES [106]	8.63	16.75	24.11	77.06	99.52
Zubic et a [163]l	28.57	46.23	61.72	152.92	196.67
TORE [17]	99.51	173.2	202.88	573.81	884.01

Table 5.1: Processing time vs event volumes in a 10ms window

of events accumulated in a single time bin, simulating for both (i) sudden/unexpected bursts of high volume events and (ii) different choices of event camera resolutions. I simulate events with resolution settings (120×90, 346×260 (the resolution of DVS346 [68], a widely deployed event camera), 480×270, 512×290, 640×480) to generate (10000,..., 50000) events on average in a single time bin of $t = 10$ms to understand the processing latency on the Jetson AGX Orin over different event volumes.

Table 5.1 describes a linear relationship between event volume and processing time, with Voxel Grids and Binary Histograms processing up to 20000 events within the processing deadline t of 10ms. This indicates that in general, if we are able to pre-process high volumes of event data down to $\sim$20000 events, we can achieve fast non-blocking event processing (i.e., with processing time ≤ 10 ms) on a Jetson AGX Orin. Figure 5.8 describes the trade-off between distortion and processing time with pre-processing techniques applied to an event stream that generates $\sim$40000 events on average in a 10ms window. Nuanced event representations (e.g. MDES and TORE) achieve a lower distortion rate $\leq 30\%$ but suffer $30 \times -50\times$ higher processing times than the required 10ms. Random sampling gives the most amount of control over the volume of events filtered without suffering much distortion compared to the original event stream. Temporal filtering on the other hand suffers the most distortion with no relative gains in processing time. Finally, the profiler determines that for the CityFlow AI dataset for person detection [44], Voxel Grids can achieve both a lower distortion rate of 0.31% and a fast processing time of 7.89ms when

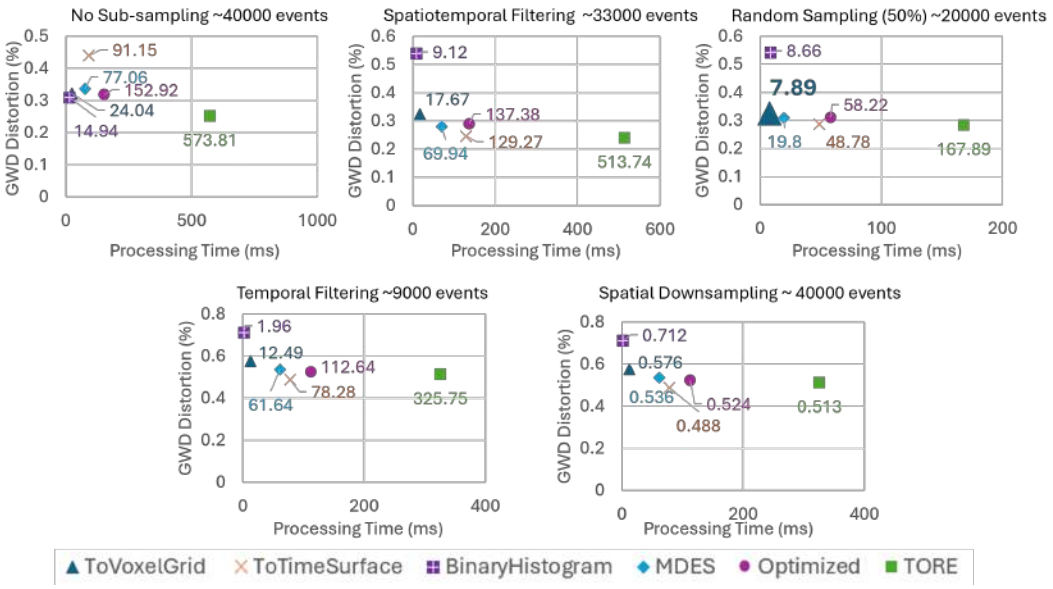

Figure 5.8: Study 2: Quality of 2D framed representations measured as distortion rate (lower, better) vs processing time (lower, better) of ∼40000 events accumulated in $t = 10$ms

random sampling is applied to reduce the volume of events to ∼20000 events.

5.1.3 Design Choices for Event Cameras at the Edge

Both studies in Section 5.1.2 yield the following observations about adopting event cameras for edge-based perception:

1. Frame-space pre-processing is on average faster and more scalable for edge-scale computing devices as compared to event-space pre-processing.

2. DNNs are the obvious choice for inference over frame-space event representations given their low inference latency of $11.23ms$ and high DNN task accuracy when compared to emulated SNNs, especially in complex regression-based tasks such as object detection.

3. If an event camera generates event streams over a $t = 10ms$ time window (equivalently, 100 FPS), an event ingest pipeline at the edge can pre-process up to ~ 20000 events into a Voxel Grid representation within this processing deadline of $10ms$ i.e. before the events in the next $10ms$ window is accu-

mulated. I believe that this setting should be generally applicable to a wide range of edge-based perception applications. However, if there is a very sparse event stream that requires smaller event accumulation windows $t \leq 10ms$, the profiler can be run again for the smaller time window to determine the best $\{$framed representation, pre-processing method$\}$ combination which moderates the distortion introduced during framed representation construction and also finishes pre-processing within the processing deadline.

4. If the event stream yields no more than 20000 events in a single $10ms$ window, the event stream need not be sub-sampled for framed representation generation. On the other hand, for higher volume event streams, random sampling the event volume down to the desired volume of 20000 events provides the most amount of control over the generated volume of events and also does not suffer much distortion in the generated frame representation, thus guaranteeing comparable downstream DNN task accuracy.

With these observations, I now present the design of *TANDEM* an edge-based pipeline which intelligently uses both CMOS and Event sensing modalities in *tandem*, incorporating fused CMOS+Event representations from multiple DVS346 cameras for high-fidelity, low-latency, and low-power canvas-based processing at the edge.

5.2 *TANDEM* Design Overview

I first describe the proposed system design for *TANDEM's* canvas-based processing over both CMOS and event sensor modalities. I envision *TANDEM* as an edge-based system capable of arbitrating between multiple pairs of $\{$CMOS (low frame-rate, feature-rich, high power overhead), Event (high temporal resolution, low-dimension, low power overhead)$\}$ streams to effectively increase processing throughput and reduce sensing energy consumption without impacting task accuracy. *TANDEM's* edge deployed design, illustrated in Figure 5.9, triggers the power expensive RGB

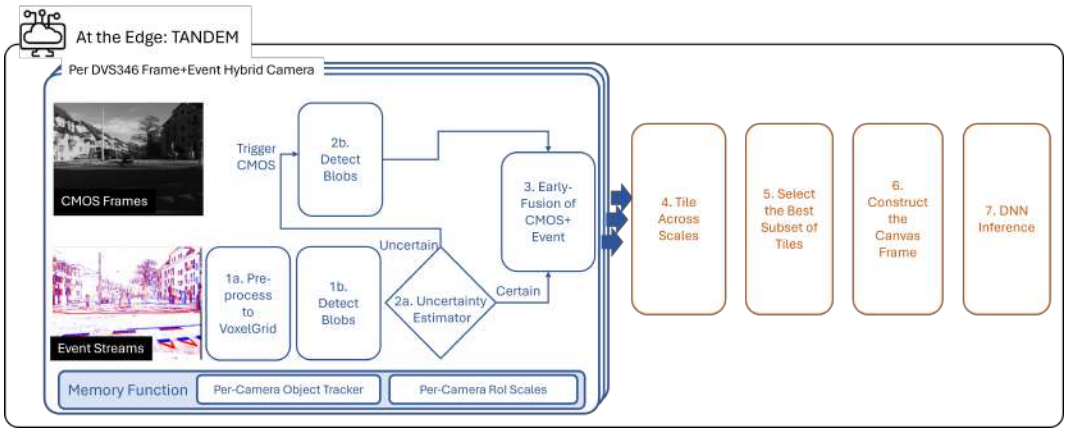

Figure 5.9: *TANDEM* Block Diagram of Sub-Components Operating at the Edge

camera infrequently i.e. with a reduced frame-rate ($\leq 1FPS$) and/or only when object uncertainty over event streams increases above a predefined threshold, thus reducing the overall sensing energy consumption while intelligently recovering sensing fidelity. Per-Camera Operations evaluate CMOS and Event streams from a single DVS346 camera (illustrated in blue in Figure 5.9, while Cross-Camera Operations for canvas construction, similar to the MOSAIC pipeline described in Chapter 2, construct the canvas frame across multiple DVS346 cameras for DNN inference (illustrated in orange in Figure 5.9).

5.2.1 Per-Camera Operation

Optimized Parallel Ingest Pipelines and Pre-processing: *TANDEM* first processes each sensor stream in parallel to prepare the inputs for subsequent fusion. In general, *TANDEM* expects an event stream to be processed in $t = 10ms$ (equivalently, 100 FPS) time windows, and expects the CMOS stream to be triggered with a longer $t \geq 33ms$ (equivalently, ≤ 30 FPS) time window. On the event stream, *TANDEM* design incorporates pre-processing of high temporal resolution event streams by reducing the event stream volume suitably for processing on a resource constrained edge device, using Random Sampling to reduce the volume of events to ≤ 20000 events if necessary. Subsequently, the event stream is composed into a 12 channel VoxelGrid framed representation for further processing, illustrated in Figure 5.10. An

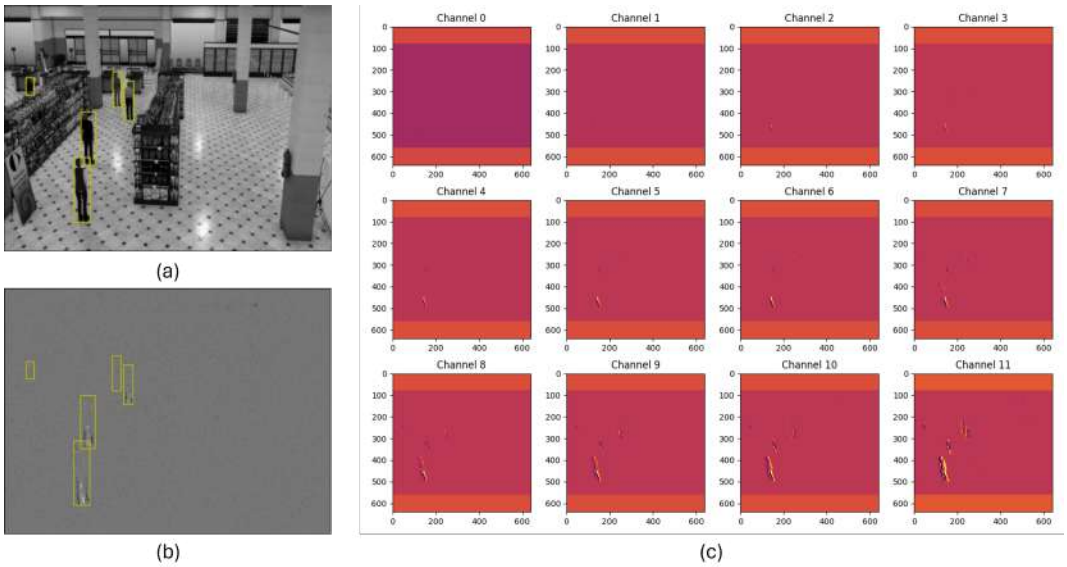

Figure 5.10: *TANDEM* Ingest Pipelines (a) Grayscale frames with RoI highlighted for ease of visualization (b) Event Stream framed representation accumulated over $t = 10ms$ with RoI highlighted for ease of visualization (c) Constructed 12 channel VoxelGrid

OpenCV-based [114] blob detector then evaluates the VoxelGrid for blobs to detect the dimensions of the detected Regions of Interest (RoI) for use by the Uncertainty Estimator, described next. On the CMOS stream, *TANDEM* ingests grayscale frames *when triggered* by the Uncertainty Estimator.

Triggering CMOS Streams On Demand - Uncertainty Estimator: The Uncertainty Estimator evaluates the detected blobs from the VoxelGrid against the Per-Camera Memory Function to evaluate the quality of the captured RoI for effective DNN inference downstream. If the Estimator evaluates a degree of uncertainty in the event stream, it opportunistically triggers the capture of a CMOS frame to recover some of the uncertainty or lost sensing capability. The following conditions result in an "Uncertain" state by the estimator, triggering the CMOS stream to capture a single frame:

- If the number of detected blobs are insufficient (less than 75%) compared to the number of blobs in the Memory Function. This indicates that the object(s)/RoI have disappeared, been occluded, or become stationary.

- If all blobs are present but the RoI height:width ratio is not consistent with the Memory Function for more than 50% of the detected blobs. This indicates that a significant number of the RoI might be occluded or in partial motion.
- If a new blob is detected which is not in memory. This indicates the appearance of a new object/RoI.

Once the CMOS stream is triggered for frame capture, the grayscale frame yielded is evaluated for blobs using OpenCV SimpleBlobDetector [114] to both (i) update the Memory Function with the size and location of the RoI, and (ii) contribute to tiling for canvas construction in the Cross-Camera Operation. The Memory Function is updated with the location and dimensions of fresh blobs and also employs a centroid-based Kalman Tracker to maintain and track RoI quality across successive VoxelGrids and triggered CMOS frames. If the CMOS frame is not triggered, and the Uncertainty Estimator is fairly certain that the information captured by the event VoxelGrid is sufficient for DNN inference, the Estimator prepares the detected blob bounding boxes for downstream Cross-camera tiling for canvas construction. The Estimator essentially pads the detected RoI whose dimensions do not match with the Memory Function with the difference detected along both height and width dimensions to ensure that the potential RoIs are not unduly cropped during tiling, impacting achievable DNN accuracy.

Asynchronous CMOS+Event Fusion: *TANDEM's* design maintains a unified/fused representation of the physical world that is jointly updated by encoded streams from both the CMOS and event camera at dramatically different "framerates" for a cohesive representation of the sensed physical world. When a grayscale CMOS frame is not available for fusion, the 12-channel VoxelGrid is convolved using a simple 3-layer MLP network into a 3 channel representation which interestingly constructs polarity as red and green channel information, illustrated in Figure 5.11 (a). When available, the grayscale single-channel image and the 12-channel VoxelGrid are concatenated together for the MLP to convolve the 13-channel image into 3 channels.

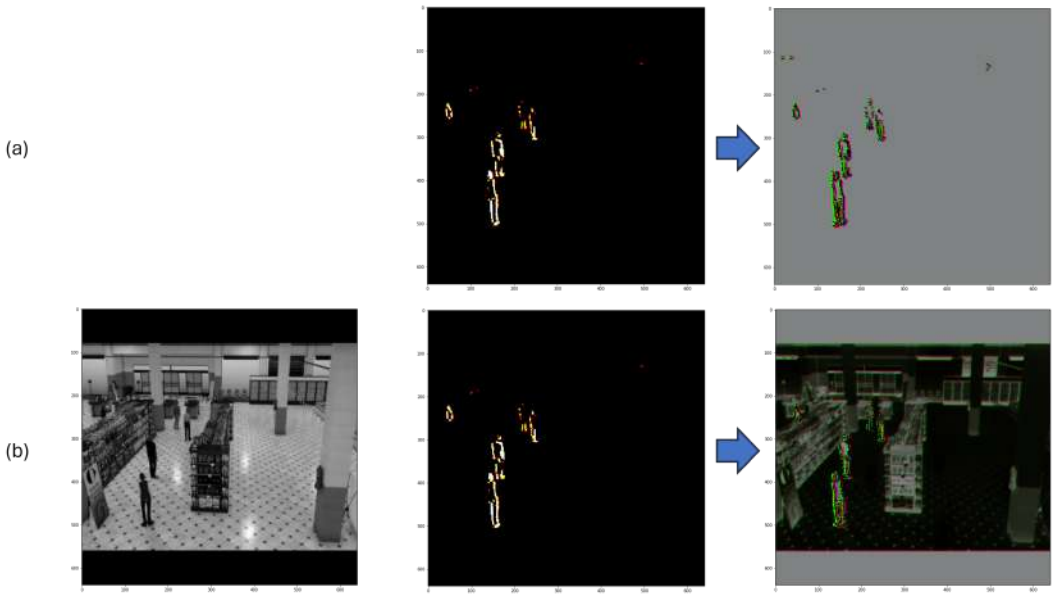

Figure 5.11: *TANDEM* CMOS+Event Fusion: (a) 3-channel representation derived from VoxelGrid only (b) 3-channel representation derived from the fusion of VoxelGrid and grayscale CMOS image.

Such a fusion operation incorporated edges and texture nuances essentially "filling in the gaps" for the event stream perception, with the VoxelGrid again retaining polarity information as red and green channel information, illustrated in Figure 5.11 (b). This fused representation is then sampled by the Cross-Camera operation for canvas construction. While such on-demand fusion is performed for a single CMOS and event representation, an alternative mechanism is to allow fusion of "stale" CMOS frames to sustain fusion for a larger ≥ 1 number of event VoxelGrids over time. However, such processing requires certain nuances to be explored, discussed in Section 5.5.

5.2.2 Cross-Camera Operation

Canvas construction across multiple fused DVS346 camera streams follows the basic principles outlined by *MOSAIC* in Chapter 2, re-iterated in brief below.

Tiling Across Scales: The fused representations from multiple DVS346 cameras are sampled by the Cross-Camera Tiling operation to decompose the fused representation into a "bag of tiles" at different scales. Per-Camera tiling scales are bootstrapped

offline and maintained in the Memory Function which allows the *TANDEM* tiling function to gain an understanding of the estimated object size distributions and the resulting tiling scales that will effectively capture the expected objects. Expected object height and width dimensions are clustered offline prior to deployment using K-Means Clustering with an elbow detection method to ascertain the ideal number of clusters/tiling scales required for each camera. The centroid of each cluster is evaluated for the larger dimension (i.e. $max(centroid_x, centroid_y)$ which is rounded to the next multiple of 32 for computational efficiency. A larger catch-all tile $1.5\times$ larger than the largest tiling scale detected is also computed and maintained in memory to account for the possibility of observing objects/RoI larger than anything seen during the initial offline operation. During *TANDEM's* operation, tiling scales are recomputed periodically (user-defined, default 30 minutes) over the dimensions of the blobs/RoI that are detected from the CMOS grayscale images. *TANDEM* tiles the fused representation at each of the determined per-camera scales with a user-defined overlap (default 0.5) between two consecutive tiles.

Selecting the Best Subset of Tiles: The *TANDEM* pipeline evaluates the bag of tiles for the best subset of tiles that accurately captures the detected blobs/RoI. Consequently, *TANDEM* constructs a spatial quadtree from the bag of tiles to conduct an intersecting bounding box search with the detected RoI/blobs. Each tile that an RoI/blob intersects with is evaluated for "goodness of fit" to ensure that (i) the RoI is not unduly cropped by the tile (ii) the RoI consumes a *sufficient* proportion of the tile (i.e. the tile captures the RoI at its appropriate scale). This yields a set of tiles which appropriately capture all the detected RoI at different scales, although many tiles may encompass the same object captured at multiple scales, requiring another filtering step to select *which* of these permissible tiles must be included onto the canvas frame. A Min Cost Min Set (MCMS) algorithm further filters the selected tiles to include each RoI on the canvas frame at least once while ensuring minimal "cost" to the canvas, evaluated as pixel wastage posed by non-RoI/background pixels. Per-tile spatial sizing bounds are also evaluated for each of the tiles to determine

the amount of resizing that the tile can suffer during canvas construction without adversely impacting DNN accuracy.

Constructing the Canvas Frame: A Differential Evolution based genetic algorithm performs 2D Inverse Bin Packing over the selected subset of tiles derived from multiple DVS346 fused representations. The constructed canvas frame is then inferred upon by a standard-off the shelf DNN model for object detection.

Post-processing DNN Predictions: The DNN predictions are mapped back from the canvas frame to the original fused representation for evaluation. *TANDEM* also conducts a final Non Maximum Suppression step to collate detections if the same object appeared in more than one tile on the canvas frame.

5.3 System Design

I now describe the prototype *TANDEM* implementation and the process for evaluating its effectiveness for a multi-camera streaming perception task.

Evaluation Platform: I evaluate *TANDEM* on the NVIDIA Jetson AGX Orin [1] which features 2048-core NVIDIA Ampere architecture GPU with 64 Tensor Cores, and a 12-core Arm Cortex-A78AE 64-bit CPU capable of 275 TOPS.

Benchmark Datasets: I evaluate *TANDEM* using the Cityflow AI Person Detection dataset from the 2023 Track 1 Challenge [44]. It features real-world and synthetic videos of a retail floor amounting to 1,491 minutes of videos from a total of 130 cameras divided into 22 subsets, each subset of cameras demonstrating spatial overlap between some/all of the cameras. All video streams are in high resolution (1920x1080), captured at 30 FPS. The dataset is loaded in grayscale and at the DVS346 CMOS dimensions of 346×260 before being fed to the v2e [62] software tool which converts the videos into event streams at the DVS346 resolution, with event aggregation at time windows of $t = 10ms$ or 100 FPS.

Evaluation Model: For object detection on the canvas frames, I employ a TensorRT-optimised YOLOv8s model, an edge-compatible pretrained model with 11.2M pa-

rameters and 28.6 GFLOPs, pre-trained using the MS COCO dataset [92]. I fine-tune the DNN for the selected dataset with both representations (illustrated in Figure 5.11) that are distinctive to *TANDEM*, (i) fused 3-channel CMOS+Event representations (when a CMOS frame is available), and (ii) 3-channel Event representations (when a CMOS frame is unavailable). The DNN is fine-tuned with the train-set of the Cityflow AI Person Detection dataset, featuring 1065602 fused representations from 58 cameras capturing 4375736 RoI, with the CMOS frames generated at 30 FPS, and event stream simulated at 100 FPS. A single inference cycle of the YOLOv8s tensorRT model incurs a processing latency of $11.23ms$, equivalently 89 FPS over these fused representations.

Evaluation Metrics: To evaluate possible gains in perception accuracy in the pedestrian detection application, I report the mean average precision of the model at an IoU threshold of 0.5 –i.e. mAP@0.5, and report the inference throughput (FPS). For all experiments, I evaluate *TANDEM* with a batch size $b = 1$.

Evaluation Baselines: I compare *TANDEM's* performance against three baselines under different combinations of CMOS and event frame rates to evaluate the power-vs-accuracy trade-off afforded by *TANDEM's* on-wake CMOS sensor design:

1. FCFS: Fused representations are resized to the canvas frame dimensions 640×640 and sent for DNN inference without any spatial modification.

2. Uniform-M: Denoted as Uni-M where M signifies the number of images packed onto a single canvas frame. Uniform-M divides a canvas into equal number of grid rows and columns and assigns each input image to a single cell in the grid. Uniform-M also determines which methodology among grid, horizontal, or vertical stacking of M input images creates the best grid structure such that each cell affords its corresponding input frame the lowest possible downsize ratio when compared to its original dimensions.

3. MOSAIC: Lastly, I evaluate *TANDEM* against the *MOSAIC* baseline to evaluate gains from the CMOS+Event sensor fusion and Uncertainty Estimator unique to *TANDEM*.

5.4 Evaluation

I first evaluate the validity of the fundamental hypotheses, that canvas-based processing of fused CMOS+Event streams from multiple DVS346 cameras helps improve the energy consumption and throughput vs. accuracy tradeoff for diverse object distributions and camera settings.

5.4.1 Characterising *TANDEM's* Energy Savings

Figure 5.12 describes *TANDEM's* performance in reducing the sensing power-consumption at the camera without significant loss of DNN task accuracy at the edge. Evaluated for CMOS+Event sensor streams from $M = 1$ DVS346 cameras and $M = 1$ standard RGB cameras [12], I observe the following:

1. RGB Camera Processing: FCFS and *MOSAIC*-based processing over standard RGB frames from the CityFlow Person detection dataset yields very high accuracy of 0.993 at 640×640 but suffers high sensing power consumption of $\sim 1.36W$.

2. Using DVS346 Sensor Streams in Isolation: Simply replacing the RGB camera with a neuromorphic camera for FCFS and *MOSAIC*-based processing over only grayscale CMOS images ingested at 30FPS maintains the accuracy but still consumes $\sim 0.9W$ of power. For *MOSAIC*-based processing over just the event VoxelGrids captured at 100 FPS, I note that extracting RoI into tiles without an understanding of the (un)certainty of the quality of the RoI incurs heavy losses of 55.2% in object detection accuracy due to undue cropping of RoI during the tiling and canvas construction process. On the other hand, FCFS based processing of only event VoxelGrids generated at 100 FPS provides the best balance between sensing power consumption ($0.7W$) and accuracy (0.936), but this comes at the cost of achievable throughput, discussed in Section 5.4.2.

3. *TANDEM's* Fused CMOS+Event Sensing: I evaluate *TANDEM's* sensing power consumption and object detection accuracy performance for different CMOS sensor settings (CMOS FPS $\in$ [30, 15, 10, 1, On-Demand]). I note that as the CMOS sensor

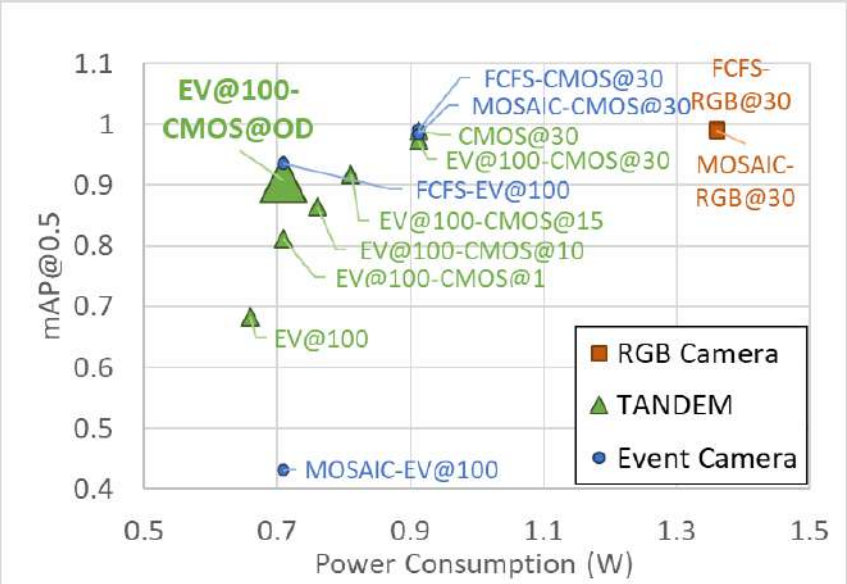

Figure 5.12: *TANDEM* System Performance: Sensing Power Consumption vs Object Detection Accuracy for $M = 1$ camera over different baselines.

is triggered more infrequently at periodic intervals, the sensing power consumption reduces as expected but the uncertainty evaluated from the event stream is not alleviated appropriately by periodic CMOS fusion, resulting in reduced object detecton accuracy. On-demand sensing recovers much of the DNN task accuracy (0.908) while consuming only $0.7W$. This presents 22.22% savings in sensing power consumption at the cost of 7.2% decrease in object detection accuracy when compared to a MOSAIC-based processing baseline operating on CMOS event streams from the DVS346 captured at 30 FPS. When compared to a standard RGB CMOS camera, *TANDEM* presents 48.52% savings in sensing power consumption, albeit with 8.1% reduction in object detection accuracy when compared to MOSAIC-based processing baseline operating on RGB CMOS streams captured at 30 FPS.

In effect, by replacing a standard RGB CMOS camera, *TANDEM* enables energy savings of $0.66Wh$ for every DVS346 camera that is run at the edge for 1 hour continuously. Consequently, for an edge device receiving $M = 10$ CMOS+Event DVS346 streams for canvas-based processing, *TANDEM* presents energy savings of $6.6Wh$ over an hour of continuous operation and $158.4Wh$ for 24 hours of continuous operation.

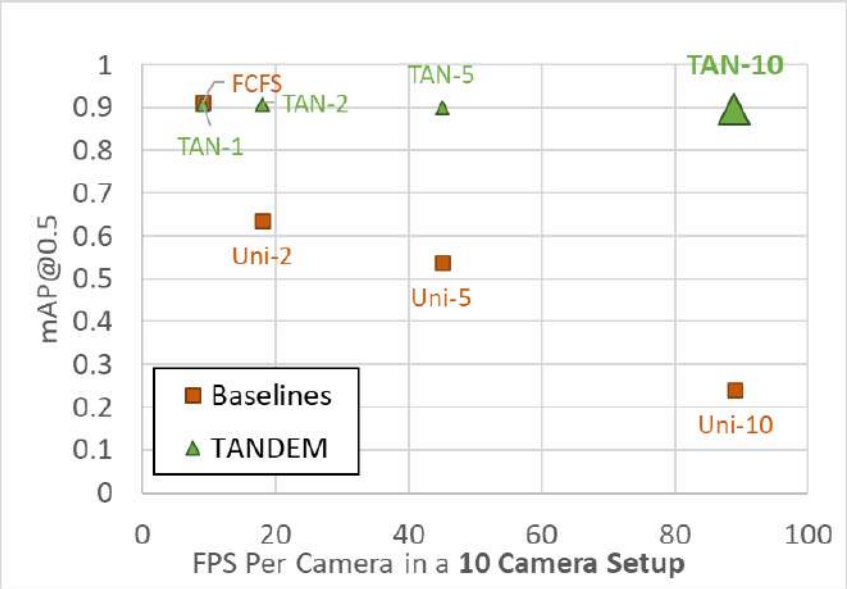

Figure 5.13: *TANDEM* System Performance: System throughput vs Object Detection Accuracy for $M = 10$ cameras.

5.4.2 Throughput-vs-Accuracy over Multiple CMOS+Event Cameras

When comparing *TANDEM's* performance against an FCFS baseline for fused representations (i.e. with the CMOS camera triggered on demand) from $M = 10$ camera streams operating at the edge, I note that *TANDEM* suffers a loss minor loss in object detection accuracy of 1.7% but gains $88\times$ or 888.88% increase in processing throughput, yielding a processing throughput of 89 FPS *per-camera* and 890 FPS cumulatively across $M = 10$ cameras on a single Jetson Orin AGX device. When compared to Uniform-10 or uniformly packing fused representations from $M = 10$ onto a canvas frame, *TANDEM* suffers no loss in throughput and achieves 65.5% gains in object detection accuracy, showing the power of canvas-based processing over fused CMOS+Event streams from hybrid neuromorphic cameras such as the DVS346.

5.5 Discussion

Building a Real-Time System: In this chapter, I make some important assumptions: (i) the canvas frames representing tiles from $M = 10$ DVS346 fused representations are constructed within the DNN processing deadline of a single inference cycle of a tensorRT optimised YOLOv8s model on the Jetson AGX Orin i.e. within $11.23ms$ (ii) OpenCV's blob detection conducted over both CMOS and event streams are accelerated by the GPU (iii) there exists a scheduling mechanism for the GPU to interleave DNN inference and pre-processing tasks such as blob detection. These assumptions lay the foundation for the system design requirements of a real-time implementation of *TANDEM* to achieve similar power-throughput-accuracy tradeoffs as the results presented in this chapter. It is important to note a singular difference between *MOSAIC* and *TANDEM* while designing a real-time system. *MOSAIC* was evaluated on the less powerful Jetson TX2 edge GPU [108] which incurred a significantly high DNN inference latency of $55ms(b = 1)$ and $172ms(b = 4)$. This allowed GPU non-blocking asynchronous canvas construction for the next frame(s) from $M = 6$ cameras. When using the more powerful Jetson Orin AGX [1] device, *TANDEM* will have to contend with and incorporate canvas construction pipelines that incur processing latency $\leq 11.23ms$ required for GPU non-blocking operation. This is a challenge given significant overheads from CPU-based OpenCV operations such as blob detection ($\sim 6ms$) and the time to construct a canvas from $M = 10$ fused representations $\sim 9ms$. I also note here that Inivation's own software library for the DVS346 includes accelerated algorithms for Blob Detection and Kalman-based Tracking of selected RoI keypoints, which could yield savings in processing latency, allowing the realization of a real-time implementation of *TANDEM*. Preliminary studies for event-based blob detection show that blobs can be reliably extracted from the raw event stream with an average IoU of 0.69 over the ground truth. While not a perfect mechanism, such processing shows speedups of $2.2ms$ when comparing the the processing latency of event blob detection ($3.5ms$) to OpenCV's blob detection

($\sim 5.7ms$). I leave further characterisations in pursuit of a real-time implementation for future work.

Event Cameras to Aid Streaming Perception: In Chapter 4, I introduce the streaming perception paradigm *JIGSAW*, and the age-based dynamic scheduler that allows a canvas-based pipeline to adopt a temporal degree of freedom to pick "when" it would include an RoI on the canvas for DNN inference. I also discussed the value of modelling object arrival and motion patterns to further fine-tune this dynamic schedule, allowing the pipeline to deprioritize objects depending on their motion (i.e. stationary or slow-moving objects) for inclusion on the canvas frame. Such nuanced control over age-based criticality estimation could yield higher system processing throughput, allowing more cameras $M \geq 25$ (as evaluated in Chapter 4) to be serviced by a single edge node. I believe that neuromorphic event cameras naturally lend themselves to object motion and velocity detection. A streaming perception pipeline could leverage the event stream to simply inform the dynamic scheduler on the object/RoI's age and motion related criticality, while limiting the canvas to be constructed from 30FPS CMOS stream as shown in Chapters 2 and 4. Preliminary studies of a Kalman-filter based tracker on a raw event stream indicate that the velocity of displacement of the centroid can be reliably extracted, though further study is required for modelling motion from this detected velocity with respect to *JIGSAW's* age-based criticality estimation and dynamic schedule. This presents an alternative paradigm for incorporating event cameras at the edge for canvas-based processing, wherein the event stream is used to determine the age-related criticality of an object/RoI and is not explicitly incorporated as a fused representation onto the canvas frame. While this may negate the energy savings presented in this chapter, additional studies or changes in system design might be able to constrain the pre-processing time for motion modelling in addition to Frame+Event fusion. Such joint optimisation within the required canvas processing deadline of $11.23ms$ could facilitate energy savings from a fused representation while also enabling streaming perception.

Increasing DNN Accuracy over Fused Frame+Event Representations: In Section 5.4.1, I discuss how the on-demand fusion of CMOS+Event representations by *TANDEM* shows significant power savings of 22.22% and 48.52% over CMOS frames from an event camera and RGB camera respectively. However, this comes at a significant cost of 8.4% and 8.5% reduction in DNN task accuracy over the FCFS processing baseline. I hypothesize that DNN task accuracy could be increased if the fusion were allowed to sustain for ≥ 1 frame at a time, thereby allowing for greater DNN task confidence over a larger volume of fused frames, yielding higher DNN task accuracy on average. To enable this, an intelligent delay mechanism would have to evaluate (i) whether there are significant per-pixel differences between two consecutive event Voxel Grids representations, and (ii) whether the fusion of a stale CMOS frame with the current event VoxelGrid could produce a distortion in the object pixels that is large enough to lead to a drop in object detection accuracy. I hypothesize that introducing a "fading" mechanism for CMOS fusion over time could best balance both needs and yield higher DNN task accuracy while sustaining the power consumption gains shown by *TANDEM*. Preliminary studies on an FCFS baseline for $M = 1$ cameras evaluating the fusion of CMOS frames at 30 FPS and event Voxel Cubes generated every 33ms i.e. at 30 FPS, show that fusion of *every* frame yields a DNN task accuracy of 0.983. This translates to a minor 0.9% reduction in DNN task accuracy, a lower DNN processing throughput of 30 FPS (as compared to 89 FPS per camera), and comparable $0.91W$ of power consumption when compared to FCFS processing of CMOS frames captured at 30 FPS. When compared to *TANDEM's* on-demand fusion, FCFS processing over *every* CMOS+Event fused representation shows a gain of 7.5% in DNN task accuracy with 22.22% higher power consumption. This indicates that higher DNN task accuracy could be achieved with more frequent fusion of CMOS and Event representations, though it is imperative that this must be achieved without actually triggering the power-consumptive CMOS sensor to sustain *TANDEM's* power savings.

Use of Temporally-Aware DNN Models: Recurrent Neural Network-based models

and Transformer-based models naturally incorporate a temporal memory between subsequent detections which have been shown to outperform CNN-based models like YOLOv3 [52] for object detection on event camera streams. I note two potential issues with adopting such models for canvas-based event stream processing at the edge: (i) Transformer-based models are typically computationally heavier than standard CNN models, for example, a Recurrent Vision Transformer [52] for object detection over event streams takes $\sim 770ms$ for a single inference cycle on the Jetson AGX Orin, and (ii) As objects traverse the sensing field, their relative spatial and temporal criticality informs their sizing bounds and inclusion on the canvas frame; these behaviours may be difficult to reconcile in the model architecture of a Transformer/RNN-based model which expects an undisturbed degree of continuity between consecutive frames. To this end, I believe that standard off the shelf DNNs like the YOLOv8s model used in this chapter, presents the superior option considering the processing throughput vs complexity tradeoffs.

Incorporating Multiple, Spatially Overlapped DVS346 Cameras: In this chapter, I designed *TANDEM's* pipelines to process each CMOS+Event camera stream at the edge independently, similar to the *MOSAIC* pipelines in Chapter 2. Incorporating spatial overlap between DVS346 cameras for a multi-camera setup is a challenging open problem in the context of event streams due to lack of defining features that facilitate any modicum of object re-identification across multiple camera streams, even when the camera pose(s) are established ahead of deployment. To facilitate accurate object re-identification, the CMOS camera would have to be triggered as often as possible, thus negating the energy savings obtained from *TANDEM's* operation. I leave the design of a system that threads the balance between energy savings and re-identification accuracy for future work, to facilitate multi-camera optimizations similar to the *JIGSAW* pipelines described in Chapter 4.

Event Cameras for Wireless Networks: Recent works incorporating event cameras [36, 120, 52], all assume wired connections between the camera and the edge processing device due to the volume of events (up to 1 Giga Events per second)

that could overwhelm any wireless network. While research works look to compressive sensing paradigms [27] to intelligently reduce event volumes, event stream optimizations for wireless networks continues to be an open problem [13].

Chapter 6

Literature Review

In this chapter, I present a variety of research directions which relate to the key contributions of this thesis. I first describe key approaches adopted in early and recent works that explore *fast and efficient* live video analytics on resource constrained devices. I first introduce early approaches such as edge-to-cloud offload which compensated for limited computation capabilities on edge devices by offloading inference to a more powerful computation device to improve processing latency. Next, I discuss the concept of *criticality awareness* in the context of input pre-processing and modification at the camera for reducing the volume of data for wireless transmission to the edge, reducing both network load and volume of data received at the edge. I also explore the concept of *attention* with respect to modifications to the inference pipelines and DNN structures at the edge for faster processing. Lastly, I introduce prior works on neuromorphic event cameras which look at efficient processing and inference over high volumes of event data and explore the notion of edge computing in the event camera context.

6.1 Live Video Analytics at the Edge

6.1.1 Cloud Offload for Faster Inference Throughput

Resource-efficient and accurate visual understanding is key to the success of live video analytics. At the time of inception of real-time video processing at the edge, research works focused on tasks such as object detection and tracking and due to limited available resources, these tasks were supported by offloading DNN feature maps or weights from a partially executed visual pipeline to the cloud or a co-located edge computing device [78, 124, 57, 157].

Kang et. al. [78] designed Neurosurgeon, a lightweight scheduler that distributed the DNN computation pipeline over the camera-adjacent edge and the cloud, completing initial layers of DNN processing at the edge and offloading intermediate feature maps to the cloud for further fine-grained inference. Neurosurgeon [78] showed significant savings in end-to-end latency, mobile energy consumption, and datacenter throughput, showcasing the value of edge-cloud offload paradigms. Similarly, DeePar [64], JointDNN [43], and Tian et. al. [135] proposed edge-cloud offload mechanisms, targeting metrics such as bandwidth availability, training time, user mobility and offload energy. Odessa proposed by Ra et. al. [124] introduced dynamism into edge-cloud, optimising offloading and parallelism choices for mobile interactive perception. Such mechanisms are more suitable for deployments featuring mobile or edge devices which had little to no onboard capabilities or embedded devices that will incur higher computational latency when compared to processing latency from cloud offload which accounts for both communication and cloud processing latency.

Another perspective to offloading computation is to prefer a "closer" edge device (in terms of network hops and round trip wireless latency) which might have available computation resources instead of offloading to the cloud. To this end, Yi et. al. introduced LAVEA [157] which offloaded computation between collaborating edge nodes near the camera, with intelligent distributed systems mechanisms for inter-edge

collaboration over a set of inference tasks. Such collaborative edge intelligence is shown to provide significant $1.3\times$ to $4\times$ savings in inference latency, when compared to the camera-cloud offload configuration. Similarly, DiStream by Zeng et. al. [159] adaptively balances workloads generated by multiple smart cameras over a cluster of edge devices for improved performance in processing throughput and latency.

In recent years, (i) improvements in edge GPU hardware, (ii) need for ultra-fast processing latency for camera streams on latency sensitive applications such as autonomous navigation, and (iii) mounting bottlenecks in cloud-offloaded computation of visual pipelines with respect to requirements in processing latency, communication latency, and privacy concerns, have motivated recent works to instead focus on live video processing at the network edge, resulting in a research area rich with interesting solutions to address difference aspects of deployment for domains such as smart homes, traffic surveillance, and healthcare [32, 100, 65, 9, 160, 41, 121, 16, 49, 8, 22, 44, 30]. This thesis does not explore cloud or edge-offload themes, and instead focuses on optimising the entire pipeline on the edge device itself.

6.1.2 Content-Aware Pre-processing to Reduce Computational Volume

A popular approach to enabling faster computation of real-time video at the edge is pre-processing or modifying the input data or the camera stream. Such processing has two significant advantages; first, the camera can choose to intelligently reduce the *amount* of data that is sent to the DNN/edge for processing. Second, such mechanisms can optimise data transfer choices for available bandwidth in the wireless network connecting the camera to the edge device. This body of work assumes the availability of computational capability at the camera for lightweight pre-processing *before* the modified camera stream is transmitted to the edge device for processing and also assumes the wireless network bandwidth to be the key bottleneck to achieving

real-time video analytics.

Limiting Camera Frame Transfer

One approach to on-camera content filtering is sampling *entire* frames at the camera to only transmit those frames that are relevant to the perception pipeline. Filter-Forward by Canel et. al. [29] installs lightweight per-application microclassifiers to evaluate the camera frames for specific types of visual content, allowing backhaul only for those frames that are considered relevant to the application. Reducto by Li et. al. [87] similarly uses adaptive filtering decisions according to the time-varying correlation between desired content filtering and achievable perception accuracy. Glimpse by Chen et. al. [32] integrates camera frame transmission trigger decisions with cloud offload, and uses an active cache of recent frames on-camera to track objects using stale hints received from the cloud device to determine trigger frames. Focus by Hsieh et. al [60] takes a different approach to frame filtering, and uses an intermediate edge device positioned between the camera and the cloud to infer on the camera frames using a lightweight DNN, pruning redundant information from the camera stream before offloading the reduced content to the cloud for further processing. These approaches adopt an "all-or-nothing" paradigm when filtering frames and therefore do not facilitate more fine-grained criticality-aware or RoI-aware frame transfer to the edge.

RoI-based Frame Transfer

This body of work on RoI-based or content criticality-aware pre-processing for frame transfer is most pertinent to this thesis. Similar to the BACT pipeline described in Chapter 3, MRIM proposed by Wu et. al. [146] approaches content-aware pre-processing by uniformly partitioning the camera frame, identifying critical regions of interest in each of these partitions, and adjusting the resolutions of these partitions to maximize the bandwidth available for frame transfer without impacting DNN task accuracy. The key difference between both approaches is in the identification

of critical regions of the frame. While the BACT pipeline uses motion-based background subtraction to remain computationally lightweight, MRIM uses Histograms of Oriented Gradients, which suffers from slow computation speed and increased likelihood of false detections. Similar to MRIM's approach, Jiang et. al. propose REMIX [74] deployed at the camera which takes in a latency budget as an input and creates a set of non-uniform partitions for frame transfer, with each partition downsampled to different resolutions depending on application needs. In a different vein, VaBUS by Wang et. al. [141] semantically compresses streams from stationary cameras by maintaining a background image of the scene at the edge and transmitting only highly confident RoI using adaptive weighting and encoding, allowing the edge to reconstruct an understanding of the scene for DNN processing.

Given the reality that most camera deployments feature multiple spatially overlapped cameras observing the same physical space, recent works have leveraged spatiotemporal correlations or similarities between cameras to fine-tune the content and volume of data transferred to the edge for processing. Jiang et. al. propose Chameleon [73] which dynamically adjusts the resolutions of and framerates from multiple cameras for frame transfer by leveraging spatiotemporal correlations determined by a leader camera. Chameleon is shown to reduce computation costs across cameras without significant loss in DNN inference accuracy. On the other hand, Guo et. al. introduce CrossROI [56] which removes or masks repeating instances of the same object as seen from multiple cameras and retains a single instance of the object to ensure comprehensive coverage of the scene. In contrast, Adamask by Liu et. al. [97] adaptively masks frames to preserve only regions of interest to the pipeline, dropping other regions for frame transfer. In a query-based multi-camera system, Spatula [71] uses cross-camera correlations to reduce communication and computation costs by reducing the search space given a query object. Another approach proposed by Liu et. al. [96] shows how spatiotemporal corrections can support the reduction of DNN execution latency bu eliminating spatially redundant DNN pipeline executions on objects seen from multiple cameras. BALB introduces a

fine-grained workload-aware latency balancing approach by extracting and retaining only critical RoI for evaluation using DNN batching in a distributed edge processing setting. The *JIGSAW* pipeline described in Chapter 4 describes the adoption of cross camera spatiotemporal correlations for the selection of mandatory tiles/instances of an RoI that must be included on the canvas frame for DNN inference. While philosophically similar to CrossROI in the identification of repeating instances of the same object across multiple cameras, the *JIGSAW* pipeline is distinguished by the tiling-based canvas construction using these identified mandatory tiles.

6.1.3 DNN Optimizations & Scheduling for Faster Throughput

Recent works on real-time visual perception pipelines deployed on edge devices are motivated by three simultaneous phenomena – (1) the proliferation of affordable and high-quality sensing and computation hardware [14, 107, 126, 11] (2) the demonstrated need for mission-critical concurrent computation of diverse and high-resolution sensor streams at the resource constrained edge with respect to available computation power and network bandwidth [5, 70, 124] and (3) that modern deep neural networks for vision and perception tasks are not designed to incorporate attention and scheduling constraints imposed by the resource constrained edge [5, 61]. In this section, I describe three fundamental approaches to optimizing perception pipelines at the edge to facilitate fast DNN task inference.

End-to-end Optimizations at the Edge

Single Camera Systems: These works do not focus on the structure of the DNN itself but introduce optimization mechanisms and pre-processing techniques applied *prior* to DNN inference. DeepCache [148] proposed by Xu et. al. takes advantage of temporal locality or redundant information across consecutive frames by caching input frames and their intermediate feature maps from prior DNN predictions for reuse in subsequent DNN inference tasks. To accelerate the processing of video

frames on edge devices, prior Region-of-Interest (ROI) approaches such as REMIX proposed by Jiang et. al. [74] attempt to selectively execute heavyweight object detector DNNs only on selected portions of an incoming image frame, thereby reducing the average inference latency. Selection of such portions is performed using approaches such as background subtraction [156], the use of a lower-complexity, 'pre-processor' DNN [48] and, most recently, in identifying *patches* [150] of varying sizes where tracking-based approaches are likely to fail. While these techniques are conceptually similar to this thesis' selection of critical regions of a frame, they focus purely on a single sensor feed as opposed to the approach of spatially packing multiple sensor streams within a single image *canvas*.

Multi-Camera Systems: Most similar to the MOSAIC pipeline discussed in Chapter 2 is the TETRIS pipeline proposed by Stone [132] and the recently proposed MONDRIAN pipeline by Jeon et. al. [72]. Both pipelines extract critical RoI from multiple cameras for RoI-based canvas construction by decomposing multiple concurrent video streams in parallel with CPU-based tiling of "active regions" or critical RoI as referred to in this thesis. However, while TETRIS and MONDRIAN extract each RoI as an individual stimuli, Chapter 2 shows how *MOSAIC's* tiling strategy allows multiple ROI that might be nearby/occluding each other to be extracted in the same tile. Chapter 3 additionally shows how such a tiling strategy adapts to bandwidth aware and workload aware canvas construction approaches, an aspect not addressed by TETRIS and MONDRIAN.

Edge GPU Scheduling

With computation power as the key bottleneck, real-time edge AI has attracted increased academic interest. From the system perspective, earlier-stage works focused on analyzing and understanding the intelligent edge platforms with GPUs [117, 151, 116], CUDA scheduling [112], CPU/GPU co-scheduling [25, 147], as well as the time variability in representative autonomous driving stacks (e.g., Apollo) [6, 122]. Alternatively, from the machine learning model perspective, there

have been works on optimizing the flexibility in DNN execution [152, 24, 80, 59, 84], which could further facilitate their deployment in real-time applications. They essentially modify the DNN execution to support various forms of *preemption*, so that corresponding real-time scheduling algorithms could be applied. Until recently, attention scheduling proposed by Liu et. al. [94, 95, 77] was proposed and utilized as a novel data-level optimization and scheduling strategy to enable real-time edge AI, where no modification on the DNN model or the underlying operating system is performed. However, they mostly explored the GPU parallel processing capacity by using task batching. Another approach to moderating/scheduling input for DNN execution is represented by the streaming perception paradigm. Li et. al. [86] proposed a new metric for streaming perception which selectively ignores frames received from the camera to evaluate localisation latency as well as localisation accuracy to ensure that DNN computation remains abreast of the kinematics of objects in the physical world. Models such as StreamYOLO [149] extend the streaming perception paradigm to evaluate DNN tasks such as object detection.

DNN Modifications for Inference

Within the broader vision and perception community, a large body of work addresses the computational overhead of DNNs with techniques for early exit and model selection [28], model compression [33], weights quantization and weights/feature maps sparsification [140], dynamic pruning [91], weights sharing [58], and model merging for models designed for different inference tasks [34]. While these strategies reduce the computation load at the edge, the methodology for canvas-based processing aims to optimize the twin metrics of bandwidth/energy consumption and edge processing throughput.

6.2 Neuromorphic Event Camera Streams at the Edge

In this thesis, I discuss event stream pre-processing on edge devices for canvas-based processing of two or more event camera streams. I now introduce works related to pre-processing high volumes of event data, RGB-event fusion, and inference paradigms for event streams at the edge.

6.2.1 Pre-processing High Volume Event Streams

Although event camera streams are sparse by nature, responding only to changes in the environment, highly dynamic scenes that feature considerable movement or changes in contrast might yield up to 1.06 Giga events per second per pixel, depending on the sensor capability. Processing such volume and velocity of data require significant processing resources and is therefore challenging on resource constrained devices. Event compression for processing lends itself to the processing of such dynamic scenes, with prior works approaching the problem in four distinct ways.

1. Frame Conversion and Interpolation: The first approach to pre-processing event streams is to reduce/collate it to the form of a well-studied two-dimensional form for processing with well researched vision algorithms, DNNs, and techniques built for RGB frames[63, 128]. This method has the advantage of leveraging decades long research in computer vision and machine perception but largely quantizes/discards the asynchronous sparse nature of event data. Popular event representations include:

- Binary Histograms [19] which represent multiple binary event images as a single frame of N-bit numbers
- Voxel Grids [162] representing events across pixels accumulated over time
- Time Surface [82] which retains temporal information from events to show recent temporal activity within a spatial region of the event camera

- Mixed Density Event Stack (MDES) [106] which constructs frames aggregating events over different time scales
- Time Ordered Recent Events (TORE) [17] which compactly represent raw event information by aggregating per-pixel queues into frames

Standard resizing techniques applied to RGB frames, such as Linear and Bicubic interpolation [113] are then applied to the resulting frames as a compression mechanism. I adopt this approach to pre-process event streams on edge devices due to the low computational overhead of computing Voxel Grids and Binary Histograms as discussed in Chapter 5.

2. Spatial Compression: More recently, Gruel et. al. have proposed the idea of using (i) log-luminescence reconstruction [54] and (ii) activation functions or input neurons from Spiking Neural Networks [55] as a mechanism for downsampling event data without conversion or collation of the events into a frame. However, these methods incur significant processing overheads on edge devices as shown in Chapter 5.

3. Temporal Compression: Another body of work looks at maintaining the spatial resolution of event data but reducing the temporal resolution without any loss in processing or task accuracy. Algorithms such as TALVEN [79] and QuadTree-based approaches [18] *evaluate the 2D spatial priority within the 3D space-time volume* [18]. These methods focus on frame-representations of the event data and differentially accumulate regions of the frame based on the 2D *spatial criticality* of the region. Regions that have more "activity" or objects of interest are accumulated in smaller bins that maintain the temporal resolution of the events whereas static areas or regions without activity are collated in larger bins. These bins are then encoded with lossy entropy encoding techniques similar to HEVC or H26X video encoding techiniques. VoxelGrids [36] maintains reduces the temporal resolution without loss in information by encoding polarity in micro time-bins which are also proven to aid in multiplicative gains in compression without loss in object recognition

accuracy. Lastly, the recently proposed ADDER framework [47] introduces a *per-pixel* temporal compression strategy with each pixel determining its own decimation factor or threshold which determines how many events must be accumulated before firing a downsampled event. More active pixels will increase their threshold over time whereas less active pixels will adopt a lower threshold to enable greater sensitivity to scene changes.

4. Spatiotemporal Encoding: The fourth paradigm SpikeCoding [27] is in spatiotemporal encoding of event streams, introduced for event video compression. The event stream is partitioned into quadtree-based cubes based on event frequency, and each cube is evaluated on an inter-cube and intra-cube basis for (i) impact of the cube's compression on the vision task and (ii) optical consistency of the resulting frame-space video. Such compression methods maintain the information gain from the high temporal resolution of the event sensors without losing the its asynchronicity but incur extremely high processing latency on edge devices.

6.2.2 Fusion of RGB and Event Streams

Recent works also look at intelligent fusion of RGB and event sensor streams to jointly take advantage of the high spatial resolution of feature-rich RGB streams with the high temporal resolution of event streams. Tomy et. al. [136] proposes a feature pyramid sensor early-fusion model which extracts features from both RGB and event streams independently at three different spatial scales. At each scale both sets of features are concatenated before being fed to a RetinaNet-50 network. Cress et. al. [37] explore pre-processing techniques to calibrate RGB and event sensors for effective fusion, especially when both cameras are in motion. Whereas Gehrig et. al. [38] explore late stage concatenation of event and RGB streams encoded into feature representations for monocular depth estimation. In Chapter 5, I introduce early-stage fusion of both event and RGB streams into a common fused frame representation using a simple MLP-based encoder for canvas construction.

6.2.3 Inference & Processing Paradigms

In this section, I present inference paradigms for event streams that have been explored by recent works, with recent research works broadly falling into one of three categories: (i) classical DNN inference over framed representations of event streams (ii) Spiking Neural Network inference over raw event streams, and (iii) hybrid ANN-SNN models which perform late-stage fusion of ANN inference over traditional RGB camera streams and SNN inference over event streams.

DNN Inference

Recent works explore Recurrent Neural Networks and Deep Neural Networks to take advantage of existing inference paradigms, optimization techniques such as sparsification and neural pruning by inferring on framed representations of event streams. Iacono et. al. explored a standard Inception+SSD network [66] for standard off-the-shelf DNN inference over event streams and found reasonable performance, while RRC-Events proposed by Chen e. al. [31] used intermediate pseudo-labels as targets for DNN learning over event streams to show superior performance over DNN inference over RGB cameras in the case of motion blur and low illumination. Similarly, YOLOv3-Events [75], RED [120], and ASTMNet [85] find reasonable performance with improved operational metrics in processing latency and deployment characteristics over standard RGB stream processing. As for attention-based vision transformers for event streams, while some works explore applications in object classification and recognition [125, 143], object detection transformer models are still an evolving area of research with RVT proposed by Gehrig et. al. [52] showing superior performance over other ANN methods in both detection accuracy and processing latency. Crucial to note here however, is that most implementations are designed for the cloud where no memory and computation restrictions are posed on the pre-processing and inference over event streams, leaving exploration of event stream processing on edge devices an open research problem.

To take advantage of the high temporal resolution of event streams, recent works proposed by Li e. al. [88], Schaefer et. al. [127], Li et. al. [89], Feichtenhofer et. al. [45], and Mitrokhin et. al. [103] explore the applicability of Graph Neural Networks to model the spatiotemporal characteristics of events. Events are sub-sampled and modelled as nodes, and edges are constructed when events occur within a pre-defined spatiotemporal "distance" of each other. However, more work needs to be done to explore how such graphs evolve over time, especially when sub-sampling that is required to accelerate GNN inference on edge devices can lead to significant information loss over space-time.

SNN Inference

Inspired by the biological neuron and synapse, Spiking Neural Networks (SNNs) embody an asynchronous form of artificial neural networks. Their design aims to replicate the dynamics of neural membranes and action potentials over time. SNNs process information in the form of spike trains, representing a non-monotonous sequence of activations. The construction of an SNN involves connecting populations of SNN neurons based on specific rules and a defined architecture, with SNN neurons emitting spikes when their internal voltage threshold has been reached. SNNs offer advantages such as improved energy efficiency, robustness to noise, and temporal coding capabilities, but the spike generation mechanism is not differentiable, leading to optimisation issues for SNN pipelines and low SNN task accuracy as shown by Cordone et. al. [36] where SNNs achieves a very low mean accuracy, showing an accuracy drop of $\sim 28.3\%$ when compared to other DNN models [52].

Hybrid SNN-DNN Inference

State of the art works in hybrid RGB-Event pipelines [143, 10] have primarily leveraged event streams as efficient encoders of temporal information. SNN processed spiking outputs, considered as encoded temporal information, are fused with features extracted by DNN pipelines inferring on RGB frames. Such late-stage fusion of

the independent outputs of (i) the SNN inferring on event streams and (ii) DNNs inferring on RGB streams, relies on methods such as heuristics, temporal filtering, or accumulation of detections based on the ANN outputs. However, both SNN and ANN pipelines process their input streams independently without sharing features between the networks, making this fusion shallow.

Chapter 7

Conclusions and Future Outlook

I conclude the thesis by summarising the key contributions of each research work before outlining some key directions for future works.

7.0.1 Summary of Contributions

In this thesis, I show that nuanced processing of multiple concurrent camera streams across both RGB cameras and neuromorphic event cameras can push the envelope, in terms of processing capacity and throughput with negligible loss of DNN task accuracy, that is achievable on resource-constrained edge devices. I introduce the concept of "Criticality Awareness" both at the edge and a compute-enabled camera to facilitate the novel "Canvas-based Processing" paradigm which spatiotemporally channels limited computation resources to critical regions from multiple input streams to enable *accurate* and *efficient* live video analytics at the edge. I also demonstrate how leveraging concepts such as optimised bandwidth-aware camera→edge frame transmission, workload adaptive canvas construction, streaming perception evaluation, and fusion with event camera streams can provide a perception pipeline with greater flexibility and control over achievable system performance outcomes.

MOSAIC: In Chapter 2, I introduce *MOSAIC*, a criticality-aware "spatial multiplexing" based approach for DNN-based inferencing on edge devices that extracts high-priority regions of individual images and then spatially packs them into a com-

posite canvas frame of smaller size, so as to ensure high processing throughput. *MOSAIC's* key innovation is the introduction of a *spatial degree of freedom* as the Mosaic-of-Scales (MoS) pipeline, a multi-scale tiling approach that ensures that objects of varying sizes are both represented at adequate dimensions, and with minimal redundancy on the canvas. Experimental studies with a representative Jetson TX2 edge device demonstrate how *MOSAIC* can provide a multiplicative increase in throughput—e.g., by packing critical regions from 6 distinct camera images into a single canvas frame, *MOSAIC* can achieve a cumulative throughput of $\sim$ 138 FPS while achieving pedestrian detection accuracy of 79% on the Okutama-Action dataset. In contrast, processing each image frame individually provides a slight increase ($\leq 1\%$) in accuracy to 80% but with sharply lower throughput ($\sim 18FPS$), while simplistically packing 6 image frames uniformly into a canvas frame can achieve similar throughput ($\sim$144 FPS) but with significantly lower accuracy (71%). Similar gains are observed for a separate License Plate Recognition application, thereby demonstrating the generalizability of *MOSAIC*.

RA-MOSAIC: In Chapter 3, I introduce *RA-MOSAIC*, an end-to-end resource adaptive pipeline with bandwidth adaptive camera$\rightarrow$edge frame transmission and workload adaptive canvas construction over M camera streams, providing significant gains in the achievable throughput-accuracy tradeoff in bandwidth constrained and dynamic workload environments. Conceptually, RA-MOSAIC extends MOSAIC by showing how integrating the sensors and edge devices into a joint optimization of both sensor data transmission and processing can yield superior performance. At the camera, *RA-MOSAIC* differentially downsamples critical regions of the camera frame with higher resolution afforded to those regions that contain regions/objects of interest. At the edge, *RA-MOSAIC* tiles and spatially multiplexes multiresolution frames from M cameras onto a dynamically sized canvas frame, to opportunistically afford faster processing throughput when workloads from M cameras are low. I show that in a real-world bandwidth constrained wireless environment, *RA-MOSAIC* packs tiles from $M = 6$ concurrent camera streams to provide a simultaneous (i)

14.3% gain in accuracy and (ii) **11.11%** gain in average throughput over MOSAIC. Compared to bandwidth adaptive FCFS and naive uniform grid packing baselines, *RA-MOSAIC* suffers no loss in accuracy and a 17% gain in accuracy respectively, while providing a $5.6\times$ or 566.67% gain in throughput and 5% gain in throughput respectively, up to 20 FPS on average per-camera, cumulatively 120 FPS over $M = 6$ cameras.

JIGSAW: In Chapter 4, I introduce *JIGSAW* as a novel system that utilizes canvas-based processing of multiple sensor input streams to optimize streaming perception tasks on a single GPU-equipped edge device. Supported by an intelligent dynamic scheduler, tile utility maximization mechanisms and cross-camera view mappings, *JIGSAW* constructs canvas frames representing unique objects sensed across multiple cameras, some of which might have spatial overlap in their field of view. *JIGSAW* utilizes the insight that streaming perception may benefit from delaying or discarding selected arriving frames (or regions within such frames), and thus dispenses with *MOSAIC's* approach of attempting to process every incoming frame from each camera. A Jetson TX2 based implementation demonstrates that *JIGSAW* is capable of performing accurate streaming perception over 25 concurrent camera streams with a throughput of 19 FPS per camera–this represents a 316.67% increase in processing capacity and a 59.1% increase in object detection accuracy from state-of-the-art baselines.

TANDEM: Lastly, in Chapter 5, I introduce *TANDEM*, an edge-based system that intelligently and asynchronously *fuses* CMOS streams(grayscale, 30 FPS) and event streams (100 FPS) on demand to balance between the triple of (sensing energy consumption, processing throughput and perception accuracy). In effect, *TANDEM* provides a mechanism to support significantly higher processing frame rates (than would be achievable with RGB sensors alone) together with low power consumption and high object detection task accuracy, that can benefit tasks such as autonomous navigation and mechanical analysis of machinery. I also show how canvas-based processing can multiplex multiple pairs of {feature-rich CMOS streams, energy-efficient

event streams} for many-to-one Canvas-based Processing using a single DNN on a resource constrained edge device. *TANDEM's* intelligent on-demand sensor fusion recovers the uncertainty that is introduced in the canvas construction pipeline by low-dimension event frame representations which might lack defining object features due to lack of significant object/RoI motion within the pre-determined sensing window. Implemented on a Jetson AGX Orin device *TANDEM* simultaneously packs fused CMOS+Event representations from $M = 10$ cameras onto a canvas frame for DNN inference providing a $88\times$ or 888.88% gain in processing throughput at 89 FPS *per-camera* while suffering a minor loss in object detection accuracy of 1.7%.

7.0.2 Publications

The research work described in this thesis has contributed to the following peer-reviewed conference publications:

1. **Gokarn, I.**, & Misra, A. (2024). "Poster: Profiling Event Vision Processing on Edge Devices". Proceedings of 22nd ACM International Conference on Mobile Systems, Applications, and Services (to appear).

2. **Gokarn, I.** (2024). "Criticality Aware Canvas-based Visual Perception at the Edge". Proceedings of 22nd ACM International Conference on Mobile Systems, Applications, and Services (to appear).

3. **Gokarn, I.**, Hu, Y., Abdelzaher, T., & Misra, A. (2024). "*JIGSAW*: Edge-based Streaming Perception over Spatially Overlapped Multi-Camera Deployments." Proceedings of 2024 IEEE International Conference on Multimedia and Expo (to appear).

4. Y. Hu, **I. Gokarn**, S. Liu, A. Misra and T. Abdelzaher, "Algorithms for Canvas-Based Attention Scheduling with Resizing," Proceedings of 30th IEEE Real-Time and Embedded Technology and Applications Symposium (to appear).

5. **Gokarn, I.**, Sabbella, H., Hu, Y., Abdelzaher, T., & Misra, A. (2023, June). *MOSAIC*: Spatially-multiplexed edge AI optimization over multiple concurrent video sensing streams. In Proceedings of the 14th Conference on ACM Multimedia Systems(pp. 278-288).

6. Y. Hu, **I. Gokarn**, S. Liu, A. Misra and T. Abdelzaher, "Underprovisioned GPUs: On Sufficient Capacity for Real-Time Mission-Critical Perception," 2023 32nd International Conference on Computer Communications and Networks (ICCCN), Honolulu, HI, USA, 2023, pp. 1-10.

7. Y. Hu, **I. Gokarn**, S. Liu, A. Misra and T. Abdelzaher, "Work-in-Progress: Algorithms for Canvas-Based Attention Scheduling with Resizing," 2023 IEEE Real-Time Systems Symposium (RTSS), Taipei, Taiwan, 2023.

7.0.3 Future Directions

Canvas-based Processing for Multiple Applications at the Edge: In this thesis, I have considered a single application deployed at the edge, performing a single object detection task over multiple spatiotemporally multiplexed camera streams. However, such a deployment discounts the need for multiple applications to be inferring on the *same* camera feeds to achieve different objectives. For example, the stream from a camera deployed on a traffic light might be consumed by multiple applications performing traffic congestion monitoring, pedestrian jaywalking violations detection, as well as vehicular license plate detection. If a single edge device were to run all 3 applications simultaneously, the oversubscribed GPU would not be able to achieve the throughput-vs-accuracy gains I have demonstrated in this thesis. In addition, the different applications might have different latency and accuracy requirements, which result in different operating points on a latency-vs.-accuracy curve. In Chapter 2, I show how multiple applications may impose different spatial sizing bounds for resizing each RoI onto the canvas frame to retain different levels of object sensing fidelity. For instance, the license plate recognition applications prevents the vehicle

object to be downsampled too harshly to retain Optical Character Recognition (OCR) capabilities. I intend to build on this intuition to study the impact of packing RoI extracted from different cameras, with each RoI or multiple instances of the same RoI contributing to the inference/understanding of multiple applications downstream. I believe the following mechanisms might be necessary to achieve this vision (i) many-to-one mappings between RoI and downstream application inferences (ii) mixed application RoI criticality estimator and spatial sizing bound aggregator across applications to determine a fair mapping between the size of the included RoI and the achiveable accuracy for each of the mapped applications (iii) profiler to automatically determine which applications can rely on the same RoI representation and which applications might need a much *larger* RoI representation to be mapped to the canvas frame, and (iv) a more nuanced scheduler which performs admission control over these application-mapped RoI to meet task accuracy and latency targets over all applications on average to best meet the possibly divergent task and latency requirements of different applications.

Generative AI for Reconstruction of Camera Streams from Canvas Frames: I believe that constructed canvas frames present a new paradigm for multi-image *compression*, with key details from multiple camera feeds represented in a single canvas frame. However, decompressing these canvas frames into their original camera streams for reconstruction of the sensed scene *on demand* can pose some challenges: (i) cut-mix of the extracted RoI onto a static background image may not yield realistic images (ii) scene reconstruction with respect to environmental artifacts such as the weather and ambient lighting could be difficult to model, and (iii) camera streams from dynamic cameras in motion would be challenging to construct. I hypothesize that these challenges can be addressed with a generative AI model that constructs realistic background scenes adjusted for weather, ambient light, and motion for a particular physical geography, with objects mapped from a canvas frame onto the scene and visually integrated to reconstruct a reasonable approximation of the original camera stream. Such a pipeline could present enormous savings in

storage volume demands for archival video.

Native Edge Detection in Raw Event Streams: During the development of the *TANDEM* pipeline described in Chapter 5 I discovered an open research problem in detecting edges of an object of interest in the raw event stream. Typically, to execute traditional computer vision algorithms like edge detection, the event stream must be first transformed into a framed 2D representation, discarding the rich temporal information that event streams provide. This adds a significant amount of pre-processing with CPU-deployed OpenCV [113] algorithms incurring high processing latency. Recently, Hough Transforms [131] have been used to detect straight lines in raw event streams to detect star illumination patterns [15] and poles/tracks in high speed rail lines [137]. Seifozzakerini et. al. also show how Hough Transforms can be generalised for non-linear patterns modelled as a series of linear patterns [131]. I hypothesize that Graph Neural Networks might be able to capture the adage "neurons that fire together wire together" in a more generalizable manner, adapting to dynamic cameras in motion which might observe varying object sizes and event firing patterns over time. I also believe that such GNN construction will be able to better filter out noise and create comprehensive notions of rigid objects of interest within the raw event stream, replacing DNNs for 2D or frame-based object detection.

Multi-Resolution Encoding of Event Streams: Presently, events are captured and processed at their native sensor latency when evaluating raw event streams, or quantized *uniformly* when evaluating framed representations. I hypothesize that a more nuanced Fast Fourier Transform (FFT) based event encoding method that adaptively aggregates events in different spatial regions of the event stream over different temporal lengths/windows could yield a better representation of rigid objects of interest and filter out noise from the background when evaluating event streams generated by dynamic event cameras in motion (i.e. onboard autonomous drones/robots/cars). I hypothesize that the resulting multi-spatiotemporal resolution event cube can also overcome bandwidth constraints if a event camera is wirelessly transmitting event streams to an edge device for processing. I believe that such

a formulation could open the doors to a new class of event camera-based edge perception pipelines.

Bibliography

[1] Jetson agx orin. URL https://www.nvidia.com/en-sg/autonomous-machines/embedded-systems/jetson-orin/.

[2] Dolphin raptor. URL https://www.dolphin-design.fr/tinyml-edge-ai-platform/.

[3] Inivation davis 346, 2024. URL https://inivation.com/wp-content/uploads/2019/08/DAVIS346.pdf.

[4] Prophesee evk4, 2024. URL https://www.prophesee.ai/event-camera-evk4/.

[5] T. Abdelzaher, Y. Hao, K. Jayarajah, A. Misra, P. Skarin, S. Yao, D. Weerakoon, and K.-E. Årzén. Five challenges in cloud-enabled intelligence and control. *ACM Transactions on Internet Technology (TOIT)*, 20(1):1–19, 2020.

[6] M. Alcon, H. Tabani, L. Kosmidis, E. Mezzetti, J. Abella, and F. J. Cazorla. Timing of autonomous driving software: Problem analysis and prospects for future solutions. In *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 267–280. IEEE, 2020.

[7] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor. The online set cover problem. *SIAM Journal on Computing*, 39(2):361–370, 2009.

[8] G. Ananthanarayanan, P. Bahl, R. Bodík, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha. Real-time video analytics: The killer app for edge computing. *Computer*, 50(10):58–67, 2017.

[9] G. Ananthanarayanan, V. Bahl, L. Cox, A. Crown, S. Nogbahi, and Y. Shu. Video analytics-killer app for edge computing. In *Proceedings of the 17th annual international conference on mobile systems, applications, and services*, pages 695–696, 2019.

[10] A. N. Angelopoulos, J. N. Martel, A. P. Kohli, J. Conradt, and G. Wetzstein. Event based, near eye gaze tracking beyond 10,000 hz. *arXiv preprint arXiv:2004.03577*, 2020.

[11] Apple. Apple vision pro. *Apple*, 2024. URL https://www.apple.com/apple-vision-pro/.

[12] Arducam. Arducam 64mp camera datasheet, 2024. URL https://tinyurl.com/9kd5cjfk.

[13] G. Attanasio. *Event-based camera communications: a measurement-based analysis*. PhD thesis, Politecnico di Torino, 2019.

[14] AWS. Aws deeplens, 2022. URL `https://aws.amazon.com/deeplens/`.

[15] S. Bagchi and T.-J. Chin. Event-based star tracking via multiresolution progressive hough transforms. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2143–2152, 2020.

[16] A. Balasundaram and C. Chellappan. An intelligent video analytics model for abnormal event detection in online surveillance video. *Journal of Real-Time Image Processing*, 17(4):915–930, 2020.

[17] R. W. Baldwin, R. Liu, M. Almatrafi, V. Asari, and K. Hirakawa. Time-ordered recent event (tore) volumes for event cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):2519–2532, 2022.

[18] S. Banerjee, Z. W. Wang, H. H. Chopp, O. Cossairt, and A. K. Katsaggelos. Lossy event compression based on image-derived quad trees and poisson disk sampling. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 2154–2158. IEEE, 2021.

[19] S. Barchid, J. Mennesson, and C. Djéraba. Bina-rep event frames: A simple and effective representation for event-based cameras. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 3998–4002. IEEE, 2022.

[20] M. Barekatain, M. Martí, H.-F. Shih, S. Murray, K. Nakayama, Y. Matsuo, and H. Prendinger. Okutama-action: An aerial view video dataset for concurrent human action detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 28–35, 2017.

[21] E. Barnea and O. Ben-Shahar. Exploring the bounds of the utility of context for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7412–7420, 2019.

[22] J. Barthélemy, N. Verstaevel, H. Forehead, and P. Perez. Edge-computing video analytics for real-time traffic monitoring in a smart city. *Sensors*, 19(9):2048, 2019.

[23] J. Bass. Imagezmq. *Github*, 2023. URL `https://github.com/jeffbass/imagezmq`.

[24] S. Bateni and C. Liu. Apnet: Approximation-aware real-time neural network. In *2018 IEEE Real-Time Systems Symposium (RTSS)*, pages 67–79. IEEE, 2018.

[25] S. Bateni, Z. Wang, Y. Zhu, Y. Hu, and C. Liu. Co-optimizing performance and memory footprint via integrated cpu/gpu memory management, an implementation on autonomous driving platform. In *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 310–323. IEEE, 2020.

[26] S. Bhattacharya and N. D. Lane. Sparsification and separation of deep learning layers for constrained resource inference on wearables. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, SenSys '16, page 176–189. Association for Computing Machinery, 2016.

[27] Z. Bi, S. Dong, Y. Tian, and T. Huang. Spike coding for dynamic vision sensors. In *2018 Data Compression Conference*, pages 117–126. IEEE, 2018.

[28] T. Bolukbasi, J. Wang, O. Dekel, and V. Saligrama. Adaptive neural networks for efficient inference. In *International Conference on Machine Learning*, pages 527–536. PMLR, 2017.

[29] C. Canel, T. Kim, G. Zhou, C. Li, H. Lim, D. G. Andersen, M. Kaminsky, and S. Dulloor. Scaling video analytics on constrained edge nodes. *Proceedings of Machine Learning and Systems*, 1:406–417, 2019.

[30] M.-C. Chang, C.-K. Chiang, C.-M. Tsai, Y.-K. Chang, H.-L. Chiang, Y.-A. Wang, S.-Y. Chang, Y.-L. Li, M.-S. Tsai, and H.-Y. Tseng. Ai city challenge 2020-computer vision for smart transportation applications. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 620–621, 2020.

[31] N. F. Chen. Pseudo-labels for supervised learning on dynamic vision sensor data, applied to object detection under ego-motion. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 644–653, 2018.

[32] T. Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan. Glimpse: Continuous, real-time object recognition on mobile devices. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 155–168, 2015.

[33] Y. Cheng, D. Wang, P. Zhou, and T. Zhang. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 35(1):126–136, 2018.

[34] Y.-M. Chou, Y.-M. Chan, J.-H. Lee, C.-Y. Chiu, and C.-S. Chen. Unifying and merging well-trained deep neural networks for inference stage. *arXiv preprint arXiv:1805.04980*, 2018.

[35] Y. Chung and M.-J. Park. Notes on inverse bin-packing problems. *Information Processing Letters*, 115(1):60–68, 2015.

[36] L. Cordone, B. Miramond, and P. Thierion. Object detection with spiking neural networks on automotive event data. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2022.

[37] C. Cress et al. Tumtraf event: Calibration and fusion resulting in a dataset for roadside event-based and rgb cameras. *arXiv preprint arXiv:2401.08474*, 2024.

[38] M. G. J. H.-C. Daniel Gehrig, Michelle Rüegg and D. Scaramuzza. Combining events and frames using recurrent asynchronous multimodal networks for monocular depth prediction. *IEEE Robotic and Automation Letters. (RA-L)*, 2021. URL `http://rpg.ifi.uzh.ch/docs/RAL21_Gehrig.pdf`.

[39] P. De Tournemire, D. Nitti, E. Perot, D. Migliore, and A. Sironi. A large scale event-based detection dataset for automotive. *arXiv preprint arXiv:2001.08499*, 2020.

[40] L. Deng, Y. Wu, X. Hu, L. Liang, Y. Ding, G. Li, G. Zhao, P. Li, and Y. Xie. Rethinking the performance comparison between snns and anns. *Neural networks*, 121:294–307, 2020.

[41] M. Eldib, F. Deboeverie, W. Philips, and H. Aghajan. Behavior analysis for elderly care using a network of low-resolution visual sensors. *Journal of Electronic Imaging*, 25(4):041003, 2016.

[42] J. K. Eshraghian. snntorch, 2024. URL `https://pypi.org/project/snntorch/0.1.5/`.

[43] A. E. Eshratifar, M. S. Abrishami, and M. Pedram. Jointdnn: An efficient training and inference engine for intelligent mobile cloud computing services. *IEEE Transactions on Mobile Computing*, 20(2):565–576, 2019.

[44] M. N. et. al. The 7th ai city challenge. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2023.

[45] C. Feichtenhofer, Y. Li, K. He, et al. Masked autoencoders as spatiotemporal learners. *Advances in neural information processing systems*, 35:35946–35958, 2022.

[46] J. Folz. Simplejpeg. *PyPI*, 2023. URL `https://pypi.org/project/simplejpeg/1.3.0/`.

[47] A. C. Freeman, M. Singh, and K. Mayer-Patel. An asynchronous intensity representation for framed and event video sources. In *Proceedings of the 14th Conference on ACM Multimedia Systems*, pages 74–85, 2023.

[48] M. Gao, R. Yu, A. Li, V. I. Morariu, and L. S. Davis. Dynamic zoom-in network for fast object detection in large images. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6926–6935, Los Alamitos, CA, USA, jun 2018. IEEE Computer Society.

[49] K. Gautam and S. K. Thangavel. Video analytics-based intelligent surveillance system for smart buildings. *Soft Computing*, 23(8):2813–2837, 2019.

[50] W. Ge, C. Pan, A. Wu, H. Zheng, and W.-S. Zheng. Cross-camera feature prediction for intra-camera supervised person re-identification across distant scenes. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 3644–3653, 2021.

[51] D. Gehrig, A. Loquercio, K. G. Derpanis, and D. Scaramuzza. End-to-end learning of representations for asynchronous event-based data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5633–5643, 2019.

[52] M. Gehrig and D. Scaramuzza. Recurrent vision transformers for object detection with event cameras. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13884–13893, 2023.

[53] I. Gokarn, K. Jayarajah, and A. Misra. Lightweight collaborative perception at the edge. In *Artificial Intelligence for Edge Computing*, pages 265–296. Springer, 2023.

[54] A. Gruel, J. Martinet, M. T. Serrano Gotarredona, and B. Linares Barranco. Event data downscaling for embedded computer vision. In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (2022), pp. 245-253.* SciTePress, 2022.

[55] A. Gruel, J. Martinet, B. Linares-Barranco, and T. Serrano-Gotarredona. Performance comparison of dvs data spatial downscaling methods using spiking neural networks. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6494–6502, 2023.

[56] H. Guo, S. Yao, Z. Yang, Q. Zhou, and K. Nahrstedt. Crossroi: cross-camera region of interest optimization for efficient real time video analytics at scale. In *Proceedings of the 12th ACM Multimedia Systems Conference*, pages 186–199, 2021.

[57] S. Han, H. Shen, M. Philipose, S. Agarwal, A. Wolman, and A. Krishnamurthy. Mcdnn: An approximation-based execution framework for deep stream processing under resource constraints. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, pages 123–136, 2016.

[58] X. He, Z. Zhou, and L. Thiele. Multi-task zipping via layer-wise neuron sharing. *Advances in Neural Information Processing Systems*, 31, 2018.

[59] 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)*, pages 174–187. IEEE, 2020.

[60] K. Hsieh, G. Ananthanarayanan, P. Bodik, S. Venkataraman, P. Bahl, M. Philipose, P. B. Gibbons, and O. Mutlu. Focus: Querying large video datasets with low latency and low cost. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 269–286, 2018.

[61] 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)*, pages 169–178. IEEE, 2021.

[62] Y. Hu, S. C. Liu, and T. Delbruck. v2e: From video frames to realistic DVS events. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 2021.

[63] C. Huang. Event-based timestamp image encoding network for human action recognition and anticipation. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2021.

[64] Y. Huang, F. Wang, F. Wang, and J. Liu. Deepar: A hybrid device-edge-cloud execution framework for mobile deep learning applications. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 892–897. IEEE, 2019.

[65] L. N. Huynh, Y. Lee, and R. K. Balan. Deepmon: Mobile gpu-based deep learning framework for continuous vision applications. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 82–95, 2017.

[66] M. Iacono, S. Weber, A. Glover, and C. Bartolozzi. Towards event-driven object detection with off-the-shelf deep learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018.

[67] IBM. Ibm truenorth. URL https://research.ibm.com/publications/truenorth-design-and-tool-flow-of-a-65-mw-1-million-neuron-programmable-neurosynaptic-chip.

[68] Inivation. Inivation dvs346, 2024. URL https://tinyurl.com/4bsv5acx.

[69] INTEL. Intel loihi. URL https://www.intel.com/content/www/us/en/research/neuromorphic-computing.html.

[70] S. Jain, G. Ananthanarayanan, J. Jiang, Y. Shu, and J. Gonzalez. Scaling video analytics systems to large camera deployments. In *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*, pages 9–14, 2019.

[71] S. Jain, X. Zhang, Y. Zhou, G. Ananthanarayanan, J. Jiang, Y. Shu, P. Bahl, and J. Gonzalez. Spatula: Efficient cross-camera video analytics on large camera networks. In *2020 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 110–124. IEEE, 2020.

[72] C. Jeon, S. Kim, J. Yi, and Y. Lee. Mondrian: On-device high-performance video analytics with compressive packed inference. *arXiv preprint arXiv:2403.07598*, 2024.

[73] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica. Chameleon: scalable adaptation of video analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 253–266, 2018.

[74] S. Jiang, Z. Lin, Y. Li, Y. Shu, and Y. Liu. Flexible high-resolution object detection on edge devices with tunable latency. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pages 559–572, 2021.

[75] Z. Jiang, P. Xia, K. Huang, W. Stechele, G. Chen, Z. Bing, and A. Knoll. Mixed frame-/event-driven fast pedestrian detection. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8332–8338. IEEE, 2019.

[76] J. Jylänki. A thousand ways to pack the bin-a practical approach to two-dimensional rectangle bin packing. 2010.

[77] W. Kang, S. Chung, J. Y. Kim, Y. Lee, K. Lee, J. Lee, K. G. Shin, and H. S. Chwa. Dnn-sam: Split-and-merge dnn execution for real-time object detection. May 2022.

[78] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *ACM SIGARCH Computer Architecture News*, 45(1):615–629, 2017.

[79] N. Khan, K. Iqbal, and M. G. Martini. Time-aggregation-based lossless video encoding for neuromorphic vision sensor data. *IEEE Internet of Things Journal*, 8(1):596–609, 2020.

[80] J.-E. Kim, R. Bradford, and Z. Shao. Anytimenet: Controlling time-quality tradeoffs in deep neural network architectures. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 945–950. IEEE, 2020.

[81] C. Koch and S. Ullman. Shifts in selective visual attention: towards the underlying neural circuitry. In *Matters of intelligence*, pages 115–141. Springer, 1987.

[82] X. Lagorce, G. Orchard, F. Gallupi, B. E. Shi, and R. Benosman. Hots: A hierarchy of event-based time-surfaces for pattern recognition. *IEEE T-PAMI*, 39(7):1346–1359, 2017.

[83] R. Laroca, E. Severo, L. A. Zanlorensi, L. S. Oliveira, G. R. Gonçalves, W. R. Schwartz, and D. Menotti. A robust real-time automatic license plate recognition based on the yolo detector. In *2018 international joint conference on neural networks (ijcnn)*, pages 1–10. IEEE, 2018.

[84] 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)*, pages 15–29. IEEE, 2020.

[85] J. Li, J. Li, L. Zhu, X. Xiang, T. Huang, and Y. Tian. Asynchronous spatio-temporal memory network for continuous event-based object detection. *IEEE Transactions on Image Processing*, 31:2975–2987, 2022.

[86] M. Li, Y. Wang, and D. Ramanan. Towards streaming perception. *ECCV*, 2020.

[87] Y. Li, A. Padmanabhan, P. Zhao, Y. Wang, G. H. Xu, and R. Netravali. Reducto: On-camera filtering for resource-efficient real-time video analytics. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 359–376, 2020.

[88] Y. Li, H. Zhou, B. Yang, Y. Zhang, Z. Cui, H. Bao, and G. Zhang. Graph-based asynchronous event processing for rapid object recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 934–943, 2021.

[89] Z. Li, S. Niklaus, N. Snavely, and O. Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6498–6508, 2021.

[90] C. Limberg, A. Gonçalves, B. Rigault, and H. Prendinger. Leveraging yolo-world and gpt-4v lmms for zero-shot person detection and action recognition in drone imagery. *arXiv preprint arXiv:2404.01571*, 2024.

[91] J. Lin, Y. Rao, J. Lu, and J. Zhou. Runtime neural pruning. *Advances in neural information processing systems*, 30, 2017.

[92] 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*, pages 740–755. Springer, 2014.

[93] J. Liu, Z. Yu, T. P. Breckon, and H. P. Shum. U3ds3: Unsupervised 3d semantic scene segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3759–3768, 2024.

[94] S. Liu, S. Yao, X. Fu, R. Tabish, S. Yu, H. Yun, L. Sha, and T. Abdelzaher. On removing algorithmic priority inversion from mission-critical machine inference pipelines. December 2020.

[95] 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.

[96] S. Liu, T. Wang, H. Guo, X. Fu, P. David, M. Wigness, A. Misra, and T. Abdelzaher. Multi-view scheduling of onboard live video analytics to minimize frame processing latency. In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, pages 503–514. IEEE, 2022.

[97] S. Liu, T. Wang, J. Li, D. Sun, M. Srivastava, and T. Abdelzaher. Adamask: Enabling machine-centric video streaming with adaptive frame masking for dnn inference offloading. In *Proceedings of the 30th ACM international conference on multimedia*, pages 3035–3044, 2022.

[98] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[99] A. I. Maqueda, A. Loquercio, G. Gallego, N. García, and D. Scaramuzza. Event-based vision meets deep learning on steering prediction for self-driving cars. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5419–5427, 2018.

[100] A. Mathur, N. D. Lane, S. Bhattacharya, A. Boran, C. Forlivesi, and F. Kawsar. Deepeye: Resource efficient local execution of multiple deep vision models using wearable commodity hardware. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 68–81, 2017.

[101] M. McKerns, P. Hung, and M. Aivazis. mystic: highly-constrained non-convex optimization and uq, 2021. URL `https://github.com/uqfoundation/mystic`.

[102] N. Messikommer, C. Fang, M. Gehrig, and D. Scaramuzza. Data-driven feature tracking for event cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5642–5651, 2023.

[103] A. Mitrokhin, Z. Hua, C. Fermuller, and Y. Aloimonos. Learning visual motion segmentation using event surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14414–14423, 2020.

[104] A. Mondal, J. H. Giraldo, T. Bouwmans, A. S. Chowdhury, et al. Moving object detection for event-based vision using graph spectral clustering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 876–884, 2021.

[105] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam. *The International Journal of Robotics Research*, 36(2):142–149, 2017.

[106] Y. Nam, M. Mostafavi, K.-J. Yoon, and J. Choi. Stereo depth from events cameras: Concentrate and focus on the future. In *Conference of Computer Vision and Pattern Recognition (CVPR)*, pages 6114–6123, June 2022.

[107] Q. D. Network. Vision ai development kit, 2024. URL https://developer.qualcomm.com/hardware/vision-ai-development-kit.

[108] NVIDIA. Jetson tx2 developer kit. *NVIDIA*, 2022. URL https://developer.nvidia.com/embedded/jetson-tx2.

[109] NVIDIA. Jetson xavier nx developer kit. *NVIDIA*, 2022. URL https://developer.nvidia.com/embedded/jetson-xavier-nx-devkit.

[110] NVIDIA. Jetson orin agx, 2024. URL https://tinyurl.com/4yc8ny93.

[111] NVIDIA. Jetson nano developer kit. *NVIDIA*, 2024. URL https://developer.nvidia.com/embedded/jetson-nano-developer-kit.

[112] 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)*, pages 213–225. IEEE, 2020.

[113] OpenCV. Opencv, . URL https://docs.opencv.org/4.x/da/d54/group__imgproc__transform.html.

[114] OpenCV. Opencv, . URL https://docs.opencv.org/3.4/d0/d7a/classcv_1_1SimpleBlobDetector.html.

[115] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in neuroscience*, 9:437, 2015.

[116] 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)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[117] 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)*, pages 353–364. IEEE, 2017.

[118] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean. Carbon emissions and large neural network training. arxiv 2021. *arXiv preprint arXiv:2104.10350*.

[119] M. Perdeck. Dynamically building css sprites with grid splitting, 2011. URL https://www.codeproject.com/Articles/210979/Fast-optimizing-rectangle-packing-algorithm-for-bu#basic.

[120] E. Perot, P. De Tournemire, D. Nitti, J. Masci, and A. Sironi. Learning to detect objects with a 1 megapixel event camera. *Advances in Neural Information Processing Systems*, 33:16639–16652, 2020.

[121] A. Prati, C. Shan, and K.-I.-C. Wang. Sensors, vision and networks: From video surveillance to activity recognition and health monitoring. *Journal of Ambient Intelligence and Smart Environments*, 11(1):5–22, 2019.

[122] 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)*, volume 23, 2019.

[123] H. Qiu, X. Liu, S. Rallapalli, A. J. Bency, K. Chan, R. Urgaonkar, B. Manjunath, and R. Govindan. Kestrel: Video analytics for augmented multi-camera vehicle tracking. In *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 48–59, 2018. doi: 10.1109/IoTDI.2018.00015.

[124] M.-R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan. Odessa: enabling interactive perception applications on mobile devices. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 43–56, 2011.

[125] A. Sabater, L. Montesano, and A. C. Murillo. Event transformer. a sparse-aware solution for efficient event data processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2677–2686, 2022.

[126] Samsung. Samsung s24 — s24+ 5g, 2024. URL `https://www.samsung.com/sg/smartphones/galaxy-s24/`.

[127] S. Schaefer, D. Gehrig, and D. Scaramuzza. Aegnn: Asynchronous event-based graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12371–12381, 2022.

[128] C. Scheerlinck, N. Barnes, and R. Mahony. Continuous-time intensity estimation using event cameras. In *Asian Conference on Computer Vision*, pages 308–324. Springer, 2018.

[129] Scikit-Learn. Scikit-learn random forest regressor. *Scikit-Learn*, 2007. URL `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html`.

[130] Scipy. Scipy slsqp, 2021. URL `https://docs.scipy.org/doc/scipy/reference/optimize.minimize-slsqp.html`.

[131] S. Seifozzakerini, W.-Y. Yau, K. Mao, and H. Nejati. Hough transform implementation for event-based systems: Concepts and challenges. *Frontiers in computational neuroscience*, 12:103, 2018.

[132] T. Stone, N. Stone, P. Jain, Y. Jiang, K.-H. Kim, and S. Nelakuditi. Towards scalable video analytics at the edge. In *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9, 2019. doi: 10.1109/SAHCN.2019.8824876.

[133] Q. Su, Y. Chou, Y. Hu, J. Li, S. Mei, Z. Zhang, and G. Li. Deep directly-trained spiking neural networks for object detection, 2023.

[134] Z. Tang, M. Naphade, M.-Y. Liu, X. Yang, S. Birchfield, S. Wang, R. Kumar, D. Anastasiu, and J.-N. Hwang. Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8797–8806, 2019.

[135] X. Tian, J. Zhu, T. Xu, and Y. Li. Mobility-included dnn partition offloading from mobile devices to edge clouds. *Sensors*, 21(1):229, 2021.

[136] A. Tomy, A. Paigwar, K. S. Mann, A. Renzaglia, and C. Laugier. Fusing event-based and rgb camera for robust object detection in adverse conditions. In *2022 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8524–8530. IEEE, 2022.

[137] F. Tschopp, C. Von Einem, A. Cramariuc, D. Hug, A. W. Palmer, R. Siegwart, M. Chli, and J. Nieto. Hough ̂2 map–iterative event-based hough transform for high-speed railway mapping. *IEEE Robotics and Automation Letters*, 6(2):2745–2752, 2021.

[138] Ultralytics. Ultralytics/yolov5: Yolov5 in pytorch, onnx, coreml, and tflite. *GitHub*, 2022. URL `https://github.com/ultralytics/yolov5`.

[139] Ultralytics. Ultralytics: Yolov8 in pytorch, onnx, coreml, and tflite. *GitHub*, 2022. URL `https://github.com/ultralytics/ultralytics`.

[140] T. Verelst and T. Tuytelaars. Dynamic convolutions: Exploiting spatial sparsity for faster inference. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 2320–2329, 2020.

[141] H. Wang, Q. Li, H. Sun, Z. Chen, Y. Hao, J. Peng, Z. Yuan, J. Fu, and Y. Jiang. Vabus: Edge-cloud real-time video analytics via background understanding and subtraction. *IEEE Journal on Selected Areas in Communications*, 41(1):90–106, 2023. doi: 10.1109/JSAC.2022.3221995.

[142] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Computer Vision – ECCV 2018: 15th European Conference*, page 420–436. Springer-Verlag, 2018.

[143] X. Wang, Z. Wu, Y. Rong, L. Zhu, B. Jiang, J. Tang, and Y. Tian. Sstformer: Bridging spiking neural network and memory support transformer for frame-event based recognition, 2023.

[144] N. Wojke, A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric. *CoRR*, abs/1703.07402, 2017. URL `http://arxiv.org/abs/1703.07402`.

[145] L. Wolf and S. Bileschi. A critical view of context. *International Journal of Computer Vision*, 69:251–261, 2006.

[146] J.-Y. Wu, V. Subasharan, T. Tran, and A. Misra. Mrim: Enabling mixed-resolution imaging for low-power pervasive vision tasks. In *2022 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 44–53. IEEE, 2022.

[147] 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)*, pages 392–405. IEEE, 2019.

[148] M. Xu, M. Zhu, Y. Liu, F. X. Lin, and X. Liu. Deepcache: Principled cache for mobile deep vision. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pages 129–144, 2018.

[149] J. Yang, S. Liu, Z. Li, X. Li, and J. Sun. Streamyolo: Real-time object detection for streaming perception, 2022.

[150] K. Yang, J. Yi, K. Lee, and Y. Lee. Flexpatch: Fast and accurate object detection for on-device high-resolution live video analytics. In *IEEE INFOCOM*. IEEE, may 2022.

[151] 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)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[152] 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)*, pages 305–317. IEEE, 2019.

[153] Y. Yang. FastMOT: High-Performance Multiple Object Tracking Based on Deep SORT and KLT, Nov. 2020. URL `https://doi.org/10.5281/zenodo.4294717`.

[154] Z. Yang, X. Wang, J. Wu, Y. Zhao, Q. Ma, X. Miao, L. Zhang, and Z. Zhou. Edgeduet: Tiling small object detection for edge assisted autonomous mobile vision. *IEEE/ACM Transactions on Networking*, 2022.

[155] B. Yao and L. Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 17–24. IEEE, 2010.

[156] J. Yi, S. Choi, and Y. Lee. Eagleeye: Wearable camera-based person identification in crowded urban spaces. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, New York, NY, USA, 2020. Association for Computing Machinery.

[157] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li. Lavea: Latency-aware video analytics on edge computing platform. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, pages 1–13, 2017.

[158] R. Yu, X. Chen, V. I. Morariu, and L. S. Davis. The role of context selection in object detection. *arXiv preprint arXiv:1609.02948*, 2016.

[159] X. Zeng, B. Fang, H. Shen, and M. Zhang. Distream: scaling live video analytics with workload-adaptive distributed edge intelligence. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, SenSys '20, page 409–421, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450375900. doi: 10.1145/3384419.3430721. URL `https://doi.org/10.1145/3384419.3430721`.

[160] Q. Zhang, H. Sun, X. Wu, and H. Zhong. Edge video analytics for public safety: A review. *Proceedings of the IEEE*, 107(8):1675–1696, 2019.

[161] W. Zhang, Z. He, L. Liu, Z. Jia, Y. Liu, M. Gruteser, D. Raychaudhuri, and Y. Zhang. Elf: accelerate high-resolution mobile deep vision with content-aware parallel offloading. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pages 201–214, 2021.

[162] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis. Unsupervised event-based learning of optical flow, depth, and egomotion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 989–997, 2019.

[163] N. Zubić, D. Gehrig, M. Gehrig, and D. Scaramuzza. From chaos comes order: Ordering event representations for object recognition and detection. In *Proceedings of IEEE/CVF ICCV*, pages 12846–12856, 2023.