Real-Time Fusion of Device, Cloudlet & Cloud AI

Qifei Dong, Jingao Xu, Padmanabhan Pillai, Roberta Klatzky, Mahadev Satyanarayanan
Carnegie Mellon University

Abstract

We propose a new abstraction called a *Correctable Operation (C-Op)* for accurate, real-time mobile AI applications. Based on sensed input, a mobile device initiates concurrent execution of both device-optimized local AI and state-of-the-art remote AI. Speculative action is triggered by the faster local result, but may be corrected by a more accurate remote result. We show that this approach can improve the success rate of a time-critical sensing and actuation application, and discuss future research to explore the C-Op abstraction.

CCS Concepts

• **Computer systems organization → Embedded and cyber-physical systems**; **Real-time systems**.

Keywords

Edge Computing; Cloud Computing; Multi-Tier AI Fusion; Edge AI; Drones; Robotics; Augmented Reality; End-to-End Latency; Disconnected Operation; Cyber-Human and Cyber-Physical Systems

ACM Reference Format:
Qifei Dong, Jingao Xu, Padmanabhan Pillai, Roberta Klatzky, Mahadev Satyanarayanan. 2026. Real-Time Fusion of Device, Cloudlet & Cloud AI. In *The 24th Annual International Conference on Mobile Systems, Applications and Services (MobiSys Workshop '26), June 21–25, 2026, Cambridge, United Kingdom.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3812836.3814780

1 Introduction

Achieving *fast* and *accurate* AI is difficult on size-, weight-, and power-optimized (SWaP) devices such as smartglasses, smartwatches, and ultralight drones. As AI algorithms improve, so does their voracious appetite for processing. State-of-the-art (SOTA) machine learning (ML) on servers is likely to be far too slow on SWaP devices. On today's devices, only the object detection models in the first seven rows of Figure 1 are capable of real-time AI, defined as taking less than 100 ms end-to-end [9]. More accurate models are too slow. SWaP device improvements are swamped by the processing demands of newer SOTA AI. This "arms race" continues unabated.

How do we attain both real-time speed and SOTA accuracy for AI on a mobile device? Offloading AI execution to a remote server enables mobile devices to benefit immediately from new ML advances. It avoids on-device slowdown, algorithmic compromise, and software porting effort. For example, most use of LLMs today is via offload to the cloud. However, offloading suffers from the overhead of wireless transmission and multi-hop network queuing delays. *Edge computing* was originally proposed to reduce these delays, leading to the 3-tier reference architecture of device, cloudlet, and cloud (Figure 2) [41, 42, 44]. Can we do even better?

Model, Size, and Processing Demand			mAP (%)	
Model	Params (M)	GFLOPs	Float32	Int8
EfficientDet-Lite0	3.9	1.0‡	N/A*	25.6
EfficientDet-Lite1	6.1	1.4‡		29.9
EfficientDet-Lite2	8.1	1.9‡		33.0
YOLO11n (Nano)	2.7	6.5	37.9	29.9
YOLO11s (Small)	9.7	21.5	45.4	40.1
YOLO11m (Medium)	20.9	68.0	50.1	43.3
YOLO11l (Large)	26.2	86.9	51.8	44.7
QueryInst	44.0	23.3	56.1	N/A*
DyHead	60.0	201.0	60.3	
DETR	41.3	86.0	63.7	
EVA	64.0	722.0‡	64.3	

* N/A indicates this precision is not available for the given model.
‡ Computational cost of this model is computed as an estimate.

Figure 1: Object Detection mAP Comparison [9]

We propose a programming abstraction called a *Correctable Operation (C-Op)* to fuse results from multiple tiers for average-case execution that is both fast and accurate. A C-Op enables speculative execution that exploits asymmetry in the precision-recall tradeoff space of cyber-human and cyber-physical systems: false positives are often less expensive than false negatives. For example, if a target is detected by a weak object detection model on a drone, a descent to lower altitude for better sensing and confirmation can be started right away. Waiting for a more accurate result from Tier-2 or Tier-1 may allow the target to escape. If the Tier-3 AI was wrong, little harm is done: the drone just returns to its original altitude.

Error-cost asymmetry occurs in many cyber-physical systems due to mechanical actuation. It is present in cyber-human systems because there is often a complex state that is expensive to create (e.g., a new scene to be rendered or new guidance to be given). Presentation of that state to the user can be suppressed if the early result from Tier-3 was wrong. The speculative action enabled by a C-Op can partially mask the offload latency of a more accurate

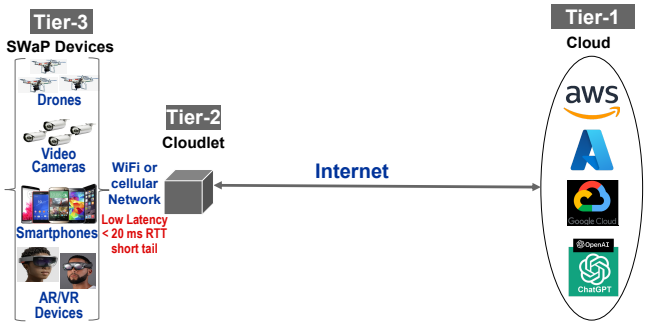


Figure 2: Tiered Architecture of Device, Cloudlet and Cloud

This work is licensed under a Creative Commons Attribution 4.0 International License.
MobiSys Workshop '26, Cambridge, United Kingdom
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2712-2/2026/06
https://doi.org/10.1145/3812836.3814780

	Typical Server		Typical Mobile Device	
Year	Processor	Speed	Device	Speed
1997	Pentium II	266 MHz	Palm Pilot	16 MHz
2002	Itanium	1 GHz	Blackberry 5810	133 MHz
2007	Intel Core 2	9.6 GHz (4 cores)	Apple iPhone	412 MHz
2011	Intel Xeon X5	32 GHz (2x6 cores)	Samsung Galaxy S2	2.4 GHz (2 cores)
2013	Intel Xeon E5-2697v2	64 GHz (2x12 cores)	Samsung Galaxy S4	6.4 GHz (4 cores)
			Google Glass Explorer Edition	2.4 GHz (2 cores)
2016	Intel Xeon E5-2698v4	88.0 GHz (2x20 cores)	Samsung Galaxy S7	7.5 GHz (4 cores)
			Microsoft HoloLens	4.16 GHz (4 cores)
2020	Intel Xeon Gold 6348H	110.4 GHz (2x24 cores)	Samsung Galaxy S20	18.5 GHz (8 Cores)
			Microsoft HoloLens 2	19.0 GHz (8 cores)
			Google Glass Enterprise Ed. 2	15.2 GHz (8 cores)
2025	Intel Xeon 6732P	243.2 GHz (2x32 cores)	Pixel 9 Pro	18.6 GHz (8 Cores)
			OnePlus 13	29.8 GHz (8 cores)
			RayNeo X2	17.3 GHz (8 cores)

Adapted from earlier work [43].
"Speed" = (number of processors × number of cores × clock speed)
Only CPU cores are included: GPUs and other accelerators are ignored.

Figure 3: The Mobility Penalty Over Three Decades

result. A C-Op may include a lateness limit, beyond which no correction exception is triggered because the new results are too late to be useful — the operation is no longer correctable.

2 Background and Related Work

The C-Op abstraction differs from prior work in using speculative execution to achieve both speed and accuracy in multi-tier AI fusion. It builds on the concepts discussed below.

2.1 The Mobility Penalty

It has long been known that, at any given point in time, there is a performance gap between SOTA hardware for static and mobile computing [39, 43]. Known as the *mobility penalty,* this gap is not a constraint of physics but an empirical observation akin to Moore's Law. It can be viewed as the price paid for being mobile. For CPU cores, Figure 3 shows the mobility penalty over three decades from the 1990s to today. For AI processing, Figure 4 shows the weight and processing power of NVIDIA mobile GPUs. For comparison, the GPU on a SOTA smartphone today (Google Pixel 9) is rated at 1.7 TFLOPS, placing it in the middle of the GPUs in Figure 4. The last row of the table (NVIDIA RTX 4090) shows a server-class GPU. It is 40–130 times as powerful as these low-end GPUs. Because of the mobility penalty, SOTA AI algorithms on server hardware run much slower on mobile hardware of the same vintage or may not be runnable at all. This leads to the need for multi-tier AI fusion. How best to achieve this fusion is an open question today.

2.2 Offloading, Split Computing and Early Exit

The use of remote execution to overcome the mobility penalty was first shown by Noble et al [34], in the context of speech recognition. Abstracting and generalizing from this work, the concept of "cyber foraging" was introduced [40]. Today, we use the term "offloading"

NVIDIA Jetson Type	Weight† (g)	INT8 TOPS (dense)	FP32 TFLOPS (dense)
Orin Nano 4GB	158	10	0.6
Orin Nano 8GB	176	33	1.3
Xavier NX	182	21	1.4
AGX Xavier	649	32	1.4
Orin NX 8GB	700	35	1.6
Orin NX 16GB	873	50	1.9
AGX Orin 64GB	873	138	5.3
AGX Thor T5000	1856	170	8.1
‡NVIDIA RTX 4090	2180		82.6

†Includes cooling such as heat sink, fan, etc. but not battery
‡Server GPU, shown for comparison

Figure 4: Weight and Processing Power of Mobile GPUs

instead of "cyber foraging." As summarized by Flinn [13], there has been extensive research in this space [1, 2, 6, 14, 15, 19, 32, 35, 48]. MAUI [5], Odessa [37], CloneCloud [4], and COMET [18] explored programming language and virtual machine support for offloading. Today, offloading is a basic tool for SOTA AI on mobile devices.

More recently, Kang et al [23] showed how offloading could be used in AI to split a deep neural network (DNN) so that its lightweight head executes at Tier-3 and its heavyweight tail executes at Tier-2 or Tier-1. The partitioning is called *split computing (SC).* When combined with *early exit (EE),* where the head result indicates that tail execution is unnecessary, it can significantly lower network bandwidth demand and cloudlet load for SOTA AI [3, 11, 20, 22, 25, 28, 29]. In contrast to SC/EE, a C-Op does not split DNN execution. Rather, it runs a completely different device-optimized model at Tier-3 and a server-optimized model at Tier-2 and/or Tier-1. The fusion of results from different tiers happens at a higher level of the AI software stack. No ML expertise is needed to use C-Ops, since no DNN surgery is involved.

2.3 Speculative Execution

The use of speculative execution in operating systems, distributed systems, and computer architecture stretches back over multiple decades. For example, I/O prefetching through read-ahead was used early in the evolution of operating systems [38]. Speculation has been widely used in contexts such as distributed file systems [33], virtual machines[10], and fault tolerance [49]. Closest to the C-Op concept is Li et al's use of speculative inference to achieve low latency in multi-modal data streams [27]. That work fuses processing results from diverse sensor streams (e.g., audio and video) by leveraging the fact that some sensor streams may arrive earlier than others. The C-Op concept differs in two ways. First, it is concerned with processing across multiple tiers of computing rather than multiple sensor streams. Second, the speculation resulting from the C-Op programming abstraction is exposed to the application. This recognizes the fact that when speculation proves to be incorrect, only the application layer can handle the resulting exception.

2.4 Nonstandard Computation Models

The standard definition of correctness for an algorithm involves a rigid output specification — no deviation is allowed. A C-Op redefines "correctness." A preliminary result is available within a pre-specified time bound, and meeting this time bound is part

of the correctness criteria. However, the result may change later, based on a better result that is available from Tier-2 or Tier-1. The change may not just be a minor refinement of the original result (e.g., higher precision), but a radically different result (e.g., "target appears to be present" changes to "no target present," or vice versa).

While the C-Op concept is new, it follows a long history of alternative computation models. Closest in spirit is multi-fidelity computation [32, 34, 45], which reverses the roles of dependent and independent variables. Other proposed models include approximation algorithms [16, 17], anytime algorithms [7], any-dimension algorithms [31], imprecise computations [12, 21], and randomized algorithms [30]. In contrast to this large body of prior work, a C-Op does not aim to reduce the total workload. Rather, it aims to reduce the negative impact of the mobility penalty as well the negative impact of network latency and bandwidth, multi-hop routing, and multi-tenant scheduling of shared Tier-2 and Tier-1 infrastructure. In fact, a C-Op is inherently "wasteful" — the same task is being performed on the same data at multiple tiers. It is intelligent use of this "wastefulness" that is the critical aspect of a C-Op.

3 Unavailable Knowledge

Until this point in the paper, we have assumed that the mobility penalty is the sole reason that inference at Tier-3 is not as accurate as inference at Tier-2 or Tier-1. In reality, the advantage of lower tiers may have more complex origins. For example, a single cloudlet at Tier-2 may process inputs from many associated Tier-3 devices, such as all the drones of a swarm. If one drone flies over an area that was recently overflown by one of its cohorts, their shared cloudlet can reuse knowledge from the previous overflight to resolve ambiguity, and thereby increase its confidence in a current inference. The object currently visible, audible, or otherwise sensed by a drone may already have been sensed by one of the other drones at the same or nearby location, possibly from a different viewpoint. The cloudlet thus has significant prior knowledge about this object and can factor that into its processing of the current input. In ML terms, it has a valuable Bayesian prior for its inference. In the real world, "accuracy" typically means "contextually accurate" and includes use of such priors. Thus, the advantage of lower tiers under this broader figure of merit may be greater than indicated by a simple technical metric such as the mAP of an object detection model on an isolated test image from a single drone.

One way to even the playing field is for Tier-2 or Tier-1 to share relevant knowledge with Tier-3, for example via caching or prefetching. However, there are real-world constraints that undermine such an approach. Caching is inherently reactive, and incurs additional latency for handling the cold cache miss. It is not clear that there will always be temporal locality to benefit from caching. Prefetching can be more effective, but its implementation is inherently temperamental, timing-dependent, and difficult to maintain. There may also be privacy and security reasons why knowledge at Tier-1 cannot be shared with Tier-3. For example, a face recognition database at Tier-1 may reveal the identity of a person, and thereby sharpen the activity recognition that can be performed on a video segment offloaded from Tier-3. Because Tier-3 is physically more vulnerable to security compromise than Tier-1, it may never be acceptable to export such sensitive information from Tier-1.

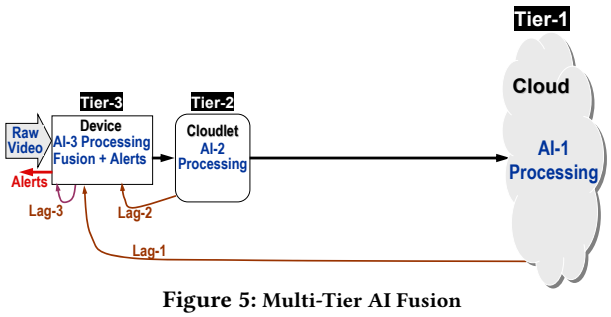


Figure 5: Multi-Tier AI Fusion

These considerations suggest that Tier-2 and Tier-1 can utilize information that is unavailable at Tier-3 or unexportable there. However, using Tier-2 or Tier-1 adds latency as discussed in §2.2.

4 Implementing C-Ops

Figure 5 illustrates the steps involved in multi-tier AI fusion. Using the drone example from the introduction (§1), video frames from the drone's camera are processed on-drone (Tier-3) using a weak model (AI-3 processing). Concurrently, they are transmitted to other tiers (Tier-2 and Tier-1) for processing with better models and leveraging of additional information that is not available at Tier-3 (AI-2 and AI-1 processing). Those results become available at the drone after delays Lag-3, Lag-2, and Lag-1, respectively.

For example, on a Tier-3 device such as a SOTA smartphone (Google Pixel 9), Dong et al [9] report that object detection takes 56.3 ms using the YOLO11-small model. The results presented later in this paper (§5) show that a cloud-based LLM (Tier-1) takes 1420 ms for the task of OCR. Thus, Lag-3 is much smaller than Lag-1 in this case. However, the relationship between lags may not always be monotonic because it depends on multiple factors: end-to-end networking latency, multi-tenant queuing and scheduling delays, AI model complexity, hardware (especially AI acceleration hardware), etc. In some cases, the more accurate processing at a lower tier may result in less lag, even after taking into account other overheads. Such is the case, for example, with the object detection results presented for Tier-2 by Dong et al [9]. Even using a much more accurate model (YOLO11-large), object detection on a cloudlet that is equipped with a powerful GPU only takes 23.9 ms. This includes the end-to-end networking cost of WiFi, but excludes multi-tenancy delays since the cloudlet is single-user. In this case, Lag-2 is significantly lower than Lag-3 even though AI-2 processing is much more accurate than AI-3 processing (Figure 1). Unfortunately, the lags from lower tiers may exhibit large variability because they are dependent on complex networking and server runtime conditions. For example, an extra 35 ms of networking delay would make Lag-2 larger than Lag-3. A single TCP timeout and retransmission could easily result in such extra delay. Static measurements or calibration at initialization, therefore, will not suffice.

Due to these considerations, the implementation of a C-Op requires significant runtime support, and cannot be an exclusively compile-time or configuration-time concept. Our initial support is therefore implemented as a runtime library, rather than a language-level construct. As we gain experience with the C-Op concept, we expect to better understand what aspects of it can be moved upstream to the compilation or runtime configuration stages.

5 Case Study: Catching a Bus

Catching a bus is a familiar example of a human C-Op. Initially, the bus may be too far away to clearly read its display, due to low visual acuity under its retinal projection – a natural analog of a low-latency, but possibly lower accuracy, Tier-3 operation. Anticipating that it may be your bus, you move closer to the stop. As it approaches, you keep glancing at the display to check the route number. Once high-acuity vision – a lower-tier operation – comes into play, if it is the right number, you proceed to get on; if not, you step back and wait for the right bus. If you rush to board without checking, you may end up on the wrong bus (uncorrected false positive). If you fail to reach the stop in time, you may miss your bus (too slow actuation, possibly due to an early false negative).

We implement a proof-of-concept C-Op by emulating this human behavior on a robot that sees brief video segments of buses projected on a large screen. To decide whether to move forward, it uses object detection to discover an approaching bus and OCR to read its route number. If a previous decision was wrong, the robot self-corrects. Accuracy (boarding the right bus) and low latency (acting before the bus departs) are both critical in this application.

5.1 Experimental Methodology

Dataset and Setup: Our evaluation is based on 70 public video clips of Pittsburgh buses arriving at stops [47]. The dataset covers diverse bus routes, with an average video duration of 10.0 s. Figure 6 shows some example frames from this dataset. As shown in Figure 7, we play these videos on a large screen for the robot to observe. Object detection is performed on each frame to detect a bus, and OCR is performed to identify its route number. If an approaching bus appears to have the desired number, the robot moves forward 0.5 m; otherwise, it remains still. If the robot is at the forward position by the end of a video clip, it is deemed to have boarded the bus. By setting the target bus number to match the ground-truth or not, we split the 70 clips into a test set of 35 positives and 35 negatives.

Approaches Compared

AI-3 only: All computation occurs on the robot. The RoboMaster carries a Tachyon development board[36] that runs AI-3, consisting of YOLO[24] and Tesseract[46] for bus detection and OCR. The robot takes action immediately after the local result is produced.

AI-1 only: The captured image is sent to AI-1, an LLM (GPT-4o) running at Tier-1, for analysis. The LLM extracts the bus number and returns the result to the robot. The robot waits for the response containing the AI-1 result before taking any action.

C-Op: Multi-tier AI fusion is used. Tier-3 performs local OCR, while concurrently sending the bus image to a Tier-1 LLM. The robot acts as soon as AI-3 completes. When the AI-1 result is eventually received, it is assumed to better match ground truth and hence overrides AI-3. In rare cases, this may be an incorrect assumption because AI-1 is based on an old frame with a distant bus while the overridden result is based on more recent frame with a closer bus.

Given the dynamic nature of real-world scenes, both AI-3 and AI-1 may occasionally produce inaccurate results for a given video frame. Each approach therefore uses a confidence metric from the AI algorithm to decide whether to capture another image for further processing and decision-making.

Success Rate KPI: Our key performance indicator (KPI) is the number of times the robot successfully "boards" a desired bus, or avoids boarding an incorrect one. We call this KPI "success rate." We deem the robot to have boarded the bus if it has moved forward 0.5m to the finish line by the time the video clip ends. Otherwise, it has not boarded the bus. At a speed of 0.1m/s, the robot takes 5s to complete the motion.

5.2 Baseline Accuracy and Latency

We first conduct a preliminary experiment to obtain AI-3 and AI-1 accuracy independent of timing requirements.

Accuracy: Figure 8 compares the accuracy of the bus identification at the two tiers. AI-1's LLM achieves higher precision, recall, and accuracy than AI-3's onboard OCR. Without timing constraints, it reaches 94% accuracy with almost no false positives (FPs) or false negatives (FNs), consistent with the high-accuracy of Tier-1.

Latency: The higher accuracy of AI-1 comes at a higher latency cost. As shown in Figure 9, AI-3 averages 360 ms per image, while AI-1 averages 1420 ms, about four times slower, with a long tail (p99 = 5730 ms, max > 20 s). The latter includes both LLM inference and network delay, which can increase under heavy server load or poor connectivity. The no-op Tier-1 round-trip time (RTT) averages 200 ms (p99 = 386 ms), obtained by sending a request to the LLM with an invalid API key as described by Dong et al [8].

5.3 KPI of Approaches

Figure 10 summarizes the success rate for the three different approaches. AI-3 exhibits many FPs and FNs, resulting in the lowest success rate of 51%. AI-1 achieves a higher success rate of 74%, mainly by performing well in negative cases. However, its high recall (Figure 8) does not help in positive cases due to high latency — results often arrive too late. Latency can also fluctuate widely due to WAN jitter and LLM server load variability.

The C-Op approach improves the overall success rate. In positive situations, the 56% success rate of the AI-3 is a floor. Timely AI-1 corrections further raise the success rate by about 10%. In negative situations, AI-3 either correctly keeps the robot stationary or, if it errs, the AI-1 result subsequently corrects the action. Of course, the wasted effort of the robot cannot be recovered. With success rates of 66% in positive situations and 100% in negative ones, the C-Op approach achieves the highest overall success rate of 83%.

5.4 Detailed Analysis

To understand how C-Op improves the success rate, Figure 11 classifies the final robot operations into four categories based on how they are affected by AI-1:

Corrected: AI-3 first produces an incorrect result, causing the robot to start the wrong action. Later, the correct result is returned from AI-1, and the previous action is corrected.

Confirmed: AI-3 produces a correct result and the robot starts the right action. Later, AI-1 returns the same correct result, and the robot will continue its action until completion.

Uncorrected: AI-3 produces a result first. However, the AI-1 result arrives too late to be useful.

Misled: AI-3 gives a result but AI-1 returns an incorrect result later, causing the robot to perform the wrong action.



Figure 6: Example Images from the Pittsburgh Bus Video Dataset



Figure 7: Experimental Setup

Approach	TP	TN	FP	FN	Precision	Recall	Acc
AI-3	25	12	23	10	52%	71%	53%
AI-1	34	32	3	1	92%	97%	94%

Results obtained from an example run over the 70 video clips.

Figure 8: Bus Identification Accuracy Comparison

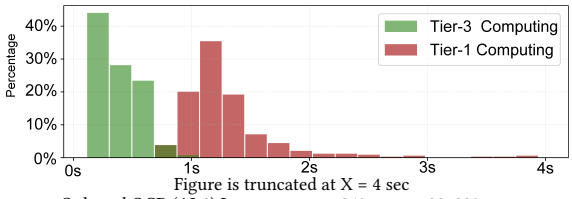


Figure is truncated at X = 4 sec
Onboard OCR (AI-3) Latency: mean: 360 msec, p99: 830 msec
Offload LLM (AI-1) Latency: mean: 1420 msec, p99: 5730 msec

Figure 9: Per-frame End-to-end Latency Distribution

Approach	Success Rate		
	Positive Cases	Negative Cases	Overall
AI-3	56% ± 4%	46% ± 6%	51% ± 2%
AI-1	50% ± 11%	97% ± 0%	74% ± 5%
C-Op	66% ± 2%	100% ± 0%	83% ± 1%

Each result is the mean and standard deviation of three runs.

Figure 10: Overall Success Rate Comparison

In a *corrected* case, the action is improved by the high accuracy of AI-1, while in a *confirmed* case, it benefits from the fast response of AI-3. 82% of executions fall into these two categories, accounting for nearly all successful cases. In positive situations, 28% are *uncorrected*, mostly due to AI-3 failing to identify the correct bus and AI-1 responses arriving too late. In contrast, only 1% of negative situations are *uncorrected*. A failed case of this type may occur when AI-3 causes the robot to board the wrong bus before AI-1 intervenes. From our results, the robot is more likely to miss the bus than board the wrong one when AI-1 responses are delayed. The gap between the two cases suggests a Tier-3 optimization: biasing toward positive, speculative operations to improve success rates.

In positive situations, 7% of the cases are classified as *misled*. These failures result from the real-time nature of the task and the staleness in AI-1 results. Many of the video clips in the dataset start with a bus approaching from a distance. The accuracy of bus number identification is low at the beginning for both AI-3 and AI-1, but gradually increases as the bus moves closer and the

Category	Ratio		
	Positive Cases	Negative Cases	Overall
Corrected	42% ± 5%	27% ± 5%	34% ± 3%
Confirmed	24% ± 4%	72% ± 5%	48% ± 3%
Uncorrected	28% ± 5%	1% ± 1%	14% ± 2%
Misled	7% ± 4%	0% ± 0%	3% ± 2%

Each result is the mean and standard deviation of three runs.

Figure 11: Tier-1 Correction Outcome by Category

display becomes more visible. In the *misled* cases, AI-3 correctly identifies the target bus based on the latest image, but the operation is miscorrected by a late-arriving incorrect AI-1 result based on an early image. Future work should include a more comprehensive way of combining multi-tier results to account for staleness.

6 Open Research Questions

Our case study is just one data point that shows the potential of C-Ops. In addition to broader validation via additional use cases, many open questions remain to be answered.

6.1 Domain of applicability

How general is the C-Op approach? In particular, over what range of application characteristics do C-Ops work well? Clearly, if time is not limited, an offloaded SOTA model is likely the most accurate. If latency bounds are too tight, only the simplest local models will be fast enough. How large is the sweet spot for C-Ops? Is it an approach that is broadly applicable, or only of niche utility? Broader exploration of C-Ops is clearly a valuable future endeavor.

6.2 Implementation Complexity

Utilizing C-Ops is inherently more complex than straightforward local or remote AI inference. Our ad-hoc implementation for the case study involved multiple threads to track concurrent local and remote inference requests and manage interactions with the main camera capture and robot actuation threads. Is there a more principled way to handle this problem? Can we design abstractions and build tools, libraries, or APIs to encapsulate and generalize C-Op functionality, disentangling it from application-specific logic? Such abstractions could greatly simplify development of new C-Op-enabled applications, reduce the likelihood of bugs, and allow independent development and evolution of general C-Op strategies.

6.3 Information Diversity

In some ways, our case study represents one of the most challenging scenarios for C-Ops. Here, all data is captured at Tier-3, and any benefits are solely due to differential accuracy and end-to-end latency of Tier-1 and Tier-3 inference. In other use cases, Tier-2 or Tier 1 may have additional or different information from Tier-3, e.g., an additional camera view with a different perspective, or a collective global map. How much does the diversity of information affect the efficacy of C-Ops? Does the information diversity necessarily imply greater accuracy? Does this change the calculus of actuation strategy with Tier-3 results? In the limit, how does the assumption of a perfect but slow remote oracle change the picture?

6.4 Temporal Issues

C-Op use cases have complex and varied timelines and behaviors. A fast local inference may execute multiple times on newer data while waiting for the first Tier-1 response on the original data item. If the results conflict, which should we trust, the more accurate model on Tier-1 using older data, or the simpler Tier-3 model using the latest data? Is there a principled mechanism to help make this determination? Perhaps we can discount confidence based on the age of the input, akin to discounting future returns in finance. Alternatively, we could build on the concept of streaming perception introduced by Li et al [26]. Sometimes, it may be more important to consider the contents of the sensed data rather than its age. For example, if the images from a camera suddenly and drastically change, perhaps older results may not be reliable. How common and effective are such heuristics in guiding C-Op strategy? Can they be generalized and expressed programmatically in an API?

7 Conclusion

This paper presents the concept of correctable operations as a promising means of leveraging the strengths of both onboard and offloaded AI, and an initial case study illustrating the potential benefits of the approach. By using fast, local, but error-prone AI models along with remotely-executed SOTA models to correct many of the errors, one can achieve higher success rates in time-critical sensing and actuation applications than either local-only or offload-only approaches. Correctable operations enables a practical and effective form of multi-tier AI fusion for latency-sensitive applications.

Acknowledgments

We thank the anonymous reviewers for their thoughtful and constructive feedback that improved the presentation of this paper. This material is based upon work supported by the U.S. Army Research Office and the U.S. Army Futures Command under Contract No. W519TC-23-C-0003 and by the National Science Foundation under grant number CNS-2106862. The content of the information does not necessarily reflect the position or the policy of the government and no official endorsement should be inferred. This work was done in the CMU Living Edge Lab, which is supported by Intel, Arm, Vodafone, Deutsche Telekom, CableLabs, Crown Castle, InterDigital, Seagate, Microsoft, the VMware University Research Fund, IAI, and the Conklin Kistler family fund. Any opinions, findings, conclusions or recommendations expressed in this document are those of the authors and do not necessarily reflect the view(s) of their employers or funding sources.

References

[1] Balan, R., Flinn, J., Satyanarayanan, M., Sinnamohideen, S., and Yang, H.-I. The Case for Cyber Foraging. In *Proceedings of the 10th ACM SIGOPS European Workshop* (Saint-Emilion, France, July 2002).

[2] Balan, R., Gergle, D., Satyanarayanan, M., and Herbsleb, J. Simplifying Cyber Foraging for Mobile Devices. In *International Conference on Mobile Systems Applications and Services* (San Juan, Puerto Rico, June 2007).

[3] Choi, H., and Bajic, I. V. Deep Feature Compression for Collaborative Object Detection. In *International Conference on Image Processing (ICIP)* (2018).

[4] Chun, B.-G., Ihm, S., Maniatis, P., Naik, M., and Patti, A. CloneCloud: Elastic Execution between Mobile Device and Cloud. In *Proceedings of EuroSys 2011* (Salzburg, Switzerland, April 2011).

[5] Cuervo, E., Balasubramanian, A., ki Cho, D., Wolman, A., Saroiu, S., Chandra, R., and Bahl, P. MAUI: Making Smartphones Last Longer with Code Offload. In *Proceedings of MobiSys 2010* (June 2010).

[6] DarøKristensen, M. Execution Plans for Cyber Foraging. In *MobMid '08: Proceedings of the 1st Workshop on Mobile Middleware* (Leuven, Belgium, 2008).

[7] Dean, T., and Boddy, M. An Analysis of Time-Dependent Planning. In *Proceedings of AAAI 1988* (1988).

[8] Dong, Q., Chen, X., and Satyanarayanan, M. Creating Edge AI from Cloud-based LLMs. In *Proc. of the 25th International Workshop on Mobile Computing Systems and Applications* (San Diego, CA, 2024).

[9] Dong, Q., Xu, J., Pillai, P., and Satyanarayanan, M. Does Accurate Real-Time AI Need Edge Offload? In *The Tenth ACM/IEEE Symposium on Edge Computing* (December 2025).

[10] Dunlap, G. W., King, S. T., Cinar, S., Basrai, M. A., and Chen, P. M. ReVirt: Enabling Intrusion Analysis Through Virtual Machine Logging and Replay. In *Proceedings of 5th Symposium on Operating Systems Design and Implementation* (December 2002).

[11] Eshratifar, A. E., Esmaili, A., and Pedram, M. BottleNet: A Deep Learning Architecture for Intelligent Mobile Cloud Computing Services. In *Symposium on Low Power Electronics and Design (ISLPED)* (2019).

[12] Feng, W., and Liu, J. W. S. An Extended Imprecise Computation Model for Time-constrained Speech Processing and Generation. In *Proceedings of the IEEE Workshop on Real-Time Applications* (1993).

[13] Flinn, J. *Cyber Foraging: Bridging Mobile and Cloud Computing via Opportunistic Offload.* Morgan & Claypool Publishers, 2012.

[14] Flinn, J., Narayanan, D., and Satyanarayanan, M. Self-Tuned Remote Execution for Pervasive Computing. In *Proceedings of the 8th IEEE Workshop on Hot Topics in Operating Systems* (Schloss Elmau, Germany, May 2001).

[15] Flinn, J., Park, S., and Satyanarayanan, M. Balancing Performance, Energy Conservation and Application Quality in Pervasive Computing. In *Proceedings of the 22nd International Conference on Distributed Computing Systems* (Vienna, Austria, July 2002).

[16] Frieze, A., Kannan, R., and Vempala, S. Fast Monte-Carlo Algorithms for Finding Low-Rank Approximations. In *FOCS: IEEE Symposium on Foundations of Computer Science (FOCS)* (1998).

[17] Garey, M., and Johnson, D. *Computers and Intractability.* Freeman and Co., 1979.

[18] Gordon, M. S., Jamshidi, D. A., Mahlke, S., Mao, Z. M., and Chen, X. COMET: Code Offload by Migrating Execution Transparently. In *10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)* (Hollywood, CA, October 2012).

[19] Goyal, S., and Carter, J. A Lightweight Secure Cyber Foraging Infrastructure for Resource-constrained Devices. In *Proceedings of the 6th IEEE Workshop on Mobile Computing Systems and Applications* (2004).

[20] Hsu, K.-J., Bhardwaj, K., and Gavrilovska, A. Couper: DNN Model Slicing for Visual Analytics Containers at the Edge. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing* (Arlington, VA, 2019).

[21] Hull, D., Feng, W., and Liu, J. W. S. Operating System Support for Imprecise Computation. In *Flexible Computation in Intelligent Systems: Results, Issues, and Opportunities* (1996).

[22] Jeong, H.-J., Jeong, I., Lee, H.-J., and Moon, S.-M. Computation Offloading for Machine Learning Web Apps in the Edge Server Environment. In *International Conference on Distributed Computing Systems (ICDCS)* (2018).

[23] Kang, Y., Hauswald, J., Gao, C., Rovinski, A., Mudge, T., Mars, J., and Tang, L. Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge. In *ASPLOS '17: Architectural Support for Programming Languages and Operating Systems* (Xi'an, China, April 2017).

[24] Khanam, R., and Hussain, M. Yolov11: An Overview of the Key Architectural Enhancements. *arXiv preprint arXiv:2410.17725* (2024).

[25] Li, G., Liu, L., Wang, X., Dong, X., Zhao, P., and Feng, X. Auto-tuning Neural Network Quantization Framework for Collaborative Inference Between the Cloud and Edge. In *Int. Conference on Artiicial Neural Networks* (2018).

[26] Li, M., Wang, Y., and Ramanan, D. Towards Streaming Perception. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2020).

[27] Li, T., Huang, J., Risinger, E., and Ganesan, D. Low-Latency Speculative

Inference on Distributed Multi-Modal Data Streams. In *Proceedings of the 19th Annual Intl. Conf. on Mobile Systems, Applications, and Services* (2021).

[28] Matsubara, Y., Baidya, S., Callegaro, D., Levorato, M., and Singh, S. Distilled Split Deep Neural Networks for Edge-Assisted Real-Time Systems. In *MobiCom Workshop on Hot Topics in Video Analytics and Intelligent Edges* (2019).

[29] Matsubara, Y., Levorato, M., and Restuccia, F. Split Computing and Early Exiting for Deep Learning Applications: Survey and Research Challenges. *ACM Computing Surveys* (March 2022).

[30] Motwani, R., and Raghavan, P. *Randomized Algorithms*. Cambridge University Press, 1995.

[31] Musliner, D. J., Durfee, E. H., and Shin, K. G. Any-Dimension Algorithms. In *Proc. of the Workshop on Real-Time Operating Systems and Software* (1992).

[32] Narayanan, D., and Satyanarayanan, M. Predictive Resource Management for Wearable Computing. In *Proceedings of MobiSys 2003* (San Francisco, CA, 2003).

[33] Nightingale, E. B., Chen, P. M., and Flinn, J. Speculative Execution in a Distributed File System. In *Proceedings of the Twentieth ACM Symposium on Operating Systems Principles* (2005).

[34] Noble, B. D., Satyanarayanan, M., Narayanan, D., Tilton, E. J., Flinn, J., and Walker, K. R. Agile Application-Aware Adaptation for Mobility. In *Proceedings of the 16th ACM Symposium on Operating Systems Principles* (Saint-Malo, France, October 1997).

[35] Ok, M., Seo, J.-W., and Park, M.-s. A Distributed Resource Furnishing to Offload Resource-Constrained Devices in Cyber Foraging Toward Pervasive Computing. In *Network-Based Information Systems*, T. Enokido, L. Barolli, and M. Takizawa, Eds., vol. 4658 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2007.

[36] Particle Industries, Inc. Tachyon Getting Started Guide. (https://docs.particle.io/getting-started/getting-started/), 2025. Last accessed on October 30, 2025.

[37] Ra, M., Sheth, A., Mummert, L., Pillai, P., Wetherall, D., and Govindan, R. Odessa: Enabling Interactive Perception Applications on Mobile Devices. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys 2011)* (Bethesda, MD, June 2011).

[38] Ritchie, D. M., and Thompson, K. The UNIX Time-Sharing System. *Communications of the ACM 17*, 7 (July 1974).

[39] Satyanarayanan, M. Hot Topics: Mobile Computing. *IEEE Computer 26*, 9 (1993), 81–82.

[40] Satyanarayanan, M. Pervasive Computing: Vision and Challenges. *IEEE Personal Communications 8*, 4 (2001).

[41] Satyanarayanan, M. The Emergence of Edge Computing. *IEEE Computer 50*, 1 (2017).

[42] Satyanarayanan, M., Bahl, P., Caceres, R., and Davies, N. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Computing 8*, 4 (2009).

[43] Satyanarayanan, M., Beckmann, N., Lewis, G. A., and Lucia, B. The Role of Edge Offload for Hardware-Accelerated Mobile Devices. In *Proceedings of HotMobile 2021* (2021).

[44] Satyanarayanan, M., Gao, W., and Lucia, B. The Computing Landscape of the 21st Century. In *Proceedings of HotMobile 2019* (2019).

[45] Satyanarayanan, M., and Narayanan, D. Multi-Fidelity Algorithms for Interactive Mobile Applications. *Wireless Networks 7*, 6 (Nov. 2001).

[46] Smith, R. An Overview of The Tesseract OCR Engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)* (2007), vol. 2, pp. 629–633.

[47] TransVideo Productions. Pittsburgh Bus Videos. (https://www.youtube.com/watch?v=OccH2mA1ou4), 2025. Last accessed on October 28, 2025.

[48] Ya-Yunn, S., and Flinn, J. Slingshot: Deploying Stateful Services in Wireless Hotspots. In *Proceedings of the 3rd International Conference on Mobile systems, Applications, and Services* (2005).

[49] Younis, M., Tsai, G., Marlowe, T., and Stoyenko, A. Using Speculative Execution for Fault Tolerance in a Real-Time System. In *Proceedings of First IEEE International Conference on Engineering of Complex Computer Systems. ICECCS'95* (1995).