7-2019

Resilient Collaborative Intelligence for Adversarial IoT Environments

Dulanga WEERAKOON
Singapore Management University, mweerakoon.2019@phdis.smu.edu.sg

Kasthuri JAYARAJAH
Singapore Management University, kasthurij@smu.edu.sg

Randy TANDRIANSYAH
Singapore Management University, rtdaratan@smu.edu.sg

Archan MISRA
Singapore Management University, archanm@smu.edu.sg

Citation

WEERAKOON, Dulanga; JAYARAJAH, Kasthuri; TANDRIANSYAH, Randy; and MISRA, Archan. Resilient Collaborative Intelligence for Adversarial IoT Environments. (2019). *22nd International Conference on Information Fusion, FUSION 2019, Ottawa, Canada, July 2-5*. 1-9.
Available at: https://ink.library.smu.edu.sg/sis_research/4429

Resilient Collaborative Intelligence for Adversarial IoT Environments

Dulanga Weerakoon, Kasthuri Jayarajah, Randy Tandriansyah, Archan Misra
School of Information Systems
Singapore Management University, Singapore

***Abstract*—Many IoT networks, including for battlefield deployments, involve the deployment of resource-constrained sensors with varying degrees of redundancy/overlap (i.e., their data streams possess significant spatiotemporal correlation). Collaborative intelligence, whereby individual nodes adjust their inferencing pipelines to incorporate such correlated observations from other nodes, can improve both inferencing accuracy and performance metrics (such as latency and energy overheads). Using realworld data from a multicamera deployment, we first demonstrate the significant performance gains (up to 14% increase in accuracy) from such collaborative intelligence, achieved through two different approaches: (a) one involving statistical fusion of outputs from different nodes, and (b) another involving the development of new collaborative deep neural networks (DNNs). We then show that these collaboration-driven performance gains susceptible to adversarial behavior by one or more nodes, and thus need resilient mechanisms to provide robustness against such malicious behavior. We also introduce an under-development testbed at SMU, specifically designed to enable real-world experimentation with such collaborative IoT intelligence techniques.**

I. Introduction

We continue to march towards a world of sensor-rich physical environments, with low-cost IoT devices projected to be deployed in large numbers in various "smart spaces", such as shopping malls, factory malls and campus buildings. Such devices will also be a critical enabler of real-time information-driven decision making in future battlefield environments. To enable real-time interactive applications such as Augmented Reality (AR) based interactions and autonomous UAV routing in such smart environments, computationally complex machine intelligence functions (such as video-based tracking of objects or audio-based instruction comprehension) will need to be executed directly on such resource-constrained IoT devices– i.e., at the *edge*. This remains an open challenge, especially given the high computational complexity of Deep Neural Networks (DNN) that represent the state-of-the-art in such machine intelligence tasks.

Dense deployment of such IoT devices is likely to result in significant *spatiotemporal correlation* in sensing coverage– e.g., two cameras mounted at two ends of a campus building corridor are likely to see the same individual either concurrently (from different perspectives) or with modest temporal separation. We believe that the appropriate exploitation of such spatiotemporal correlation can offer a powerful approach for overcoming the computational challenges of executing such high-complexity DNN-based machine intelligence tasks on such resource-poor IoT platforms. In particular, we advocate the vision of *Collaborative IoT Intelligence*, where the inferencing pipelines of multiple individual devices share features and "internal state" in real time with one another, allowing the devices to collectively both overcome their individual processing bottlenecks and improve their sense-making fidelity.

In this paper, we consider the possibility of such collaborative intelligence for the specific case of a multi-camera, campus-scale video sensing network, where the cameras are tasked with executing *people counting* algorithms to collectively provide a "live" view of the occupancy levels in different parts of the campus. For concreteness, we assume that each camera locally executes a state-of-the-art Single Shot Detector (SSD) DNN pipeline, which takes individual image frames and identifies the presence of (puts a "bounding box" around) human objects. We shall see that the conventional approach, of having each camera perform its counting in *isolation*, suffers from *reduced accuracy*, possibly due to phenomenon such as poor lighting or occlusion. We shall then propose and empirically evaluate the likely benefits of two alternative *collaborative* approaches, both of which involve the sharing of identified bounding box coordinates (i.e., the estimated location of identified individuals) by one camera with its peers.

Such collaborative intelligence models, however, have one significant potential drawback: *the performance of the DNN-based pipelines on one device becomes susceptible to inadvertent or malicious errors in the DNN pipelines of other nodes*. For example, in the distributed camera-based sensing infrastructure illustrated in Figure 1, a single camera can deliberately suppress bounding box information on detected objects from other collaborating cameras, thereby causing errors in the processing logic of those peer cameras. We shall empirically demonstrate, and quantify the relative impact of, this vulnerability for both of the proposed collaborative models.

Key Contributions: Our aim is to mobilize the attention of the larger IoT/ML community to the promise of collaborative inferencing, and the challenge of making it dependable and robust in adversarial environments. We make the following key contributions:

- **Propose Collaborative models for multi-camera edge intelligence:** We describe two alternative approaches for collaborative inferencing-based "people counting", appropriate for a networked, multi-camera system system with per-node local processing capabilities. The first approach, called Collaborative Non-Maximum Suppression (**CNMS**) refines the output values (the 'bounding box' location and likelihoods) of each camera's DNN, by statistically incorporating the bounding box coordinates from other peer cameras. The second approach, called Collaborative SSD (**CSSD**) modifies the structure of the DNN itself to utilize the bounding box values (modeled as an image bit-mask) from other cameras as part of an augmented input vector. Using the PETS multi-camera benchmark dataset [4], we show that collaborative inferencing results in significantly higher accuracy (75.5% for CNMS and 82% for CSSD), compared to a non-cooperative baseline value of 68.03%.

- **Quantify the Vulnerability to Adversarial/Malicious Behavior:** As evidence of the vulnerability of collaborative inferencing, we show that the ability of one camera to detect objects accurately can degrade sharply, if collaborating camera nodes maliciously injects errors– i.e., it deliberately perturbs the histogram values or *hides* presence of bounding boxes that it shares with neighboring nodes. More specifically, the accuracy of CNMS and CSSD approaches drops significantly, to 55% and 70% respectively, when the peer cameras inject noise that reduces the SNR values to $\approx$ 15dB. However, even in such noisy situations, the collaborative models outperform the non-collaborative baseline, whose accuracy degrades to $\approx$ 50% (for SNR $\sim=$ 15dB). Our results also show that *CSSD* is more robust to such adversarial behavior, suffering less than 5% drop in accuracy even if one camera is constantly lying. Our results demonstrate the need for building resiliency as a first-class primitive in such collaborative inferencing frameworks.
- **Outline an Upcoming Experimental Testbed** We describe the design and system architecture of a medium-scale IoT testbed currently being deployed on the SMU Campus. The testbed will initially contain multiple video sensors with edge co-processors for in-situ DNN-based inferencing. Moreover, the testbed permits easy configuration and execution (even from authorized locations outside SMU) of empirical studies of different collaborative intelligence mechanisms, permitting a systematic study of the tradeoff between different metrics such as inferencing accuracy, energy overhead, processing latency and communication overheads. We hope that our described effort becomes part of a larger, federated, multi-campus, international testbed, which would provide the research community a means of testing IoT-based machine intelligence technologies under diverse environments.

II. Illustrative IOT Environment

The collaborative intelligence framework being introduced here can broadly apply to a collection of IoT nodes, of different modalities (e.g., video, audio & magnetic) and with different characteristics (e.g., static vs. mobile nodes). However, to provide a concrete demonstration of our ideas, we first introduce an illustrative scenario, one involving a network of multiple cameras with overlapping views of varying degrees and with each camera possessing local embedded computing capabilities.

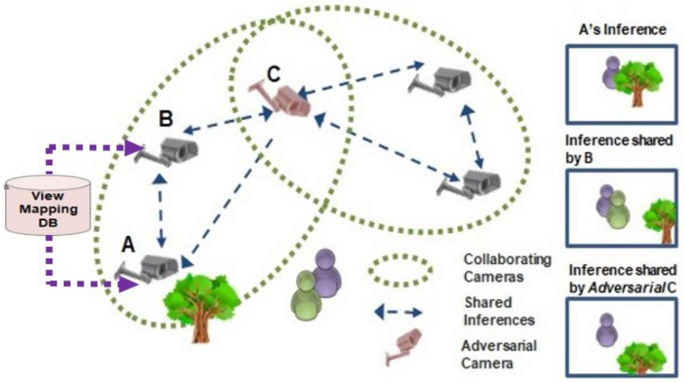


Fig. 1: Illustrative IoT environment with collaborative cameras.

Figure 1 illustrates this scenario for a campus-like environment, where multiple overlapped cameras share *spatiotemporally correlated* information. As illustrated in the figure 1, the default, *non-collaborative* model involves each camera executing a state-of-the-art SSD DNN pipeline for people detection using Intel^{TM} Movidius vision processor unit for hardware acceleration. In the collaborative approach that we consider, each camera also shares detected bounding boxes and, optionally, other features (e.g., SIFT/Color/SURF descriptors) with the neighboring cameras. In this work, we assume that this set of *neighboring cameras* is predefined and available to each camera. We shall show that such a collaborative model offers improved detection accuracy. We anticipate that such improvement occurs because an individual camera's view can be subject to various impairments, such as occlusion of certain individuals by other objects or poor illumination in part of the space being monitored. Collaborative models help to overcome such impairments as they offer the advantage of alternative field-of-view (FoV) observations, which can help to overcome such impairments.

III. Collaborative Intelligence: The Two Approaches

Our proposed collaborative inferencing system involves the following steps:

Step 1: Spatial mapping of Camera Views: As depicted in Figure 1, the initial calibration stage involves the computation of a "View Mapping" DB. This View Mapping DB computes the translation between the (x,y) coordinate systems of the FoV of different cameras. This spatial translation will help later in translating the bounding box coordinates computed by the DNN of one camera to the corresponding coordinates of another camera. This spatial relationship is computed and represented using standard homography transformation matrices [6].

Step 2: People Detection As the next step, each camera first executes it's people detection algorithm–i.e., it feeds each input frame to the DNN (executing on its Movidius co-processor) to output a set S, representing a set of people objects, along with the rectangular bounding box coordinates for each of these objects. Without loss of generality, we use a state-of-the-art deep learning-based detector called (SSD) [9] for this. The SSD pipeline accepts images as an input and generates multiple feature maps at different scales, which are subsequently provided to a downstream regression and classification network for (a) detecting object classes, (b) computing bounding box coordinates and (c) deriving the associated confidence scores. Finally the network combines these outputs from multiple feature maps to execute the Non-Maximum Suppression (NMS) step. NMS is used to avoid over-counting: it identifies bounding boxes that have significant spatial overlap and effectively unifies them into a single bounding box (corresponding to the one with the highest confidence value). In practice, such unification is performed by computing the Intersection over Union (IoU) metric, which is defined as: $Area\, of\, Overlap/Area\, of\, Union$, and which effectively describes the fraction of the area that is common to a pair of bounding boxes. Two boxes are declared to be equivalent if the IoU value exceeds a minimum threshold (e.g.,$\geq$0.7). Note that the *baseline (non-collaborative)* approach terminates at this step, with each camera independently providing the post-NMS output as the set of detected persons.

Step 3: Collaborative Intelligence To further improve the accuracy and overcome the impairments mentioned earlier, we now introduce the additional collaborative inferencing step. In this step, we focus on a single *reference* camera, outlining the modifications to its SSD pipeline to incorporate the bounding box estimates provided by its peer cameras. (In practice, each camera would implement this collaborative logic independently, refining its estimates by incorporating estimates from other cameras.) In both of our proposed collaborative approaches, each of the collaborating peer cameras first uses the homography matrix transformations to map its own bounding box coordinates to the coordinate space of the reference camera view. It then send its transformed set of bounding box coordinates (each corresponding to a potentially distinct human object) to the reference camera. In the next two subsections, we describe the two distinct ways, *CNMS* and *CSSD*, in which the reference camera incorporates this peer-supplied coordinate information.

A. Collaborative Non-Maximum Suppression (CNMS)

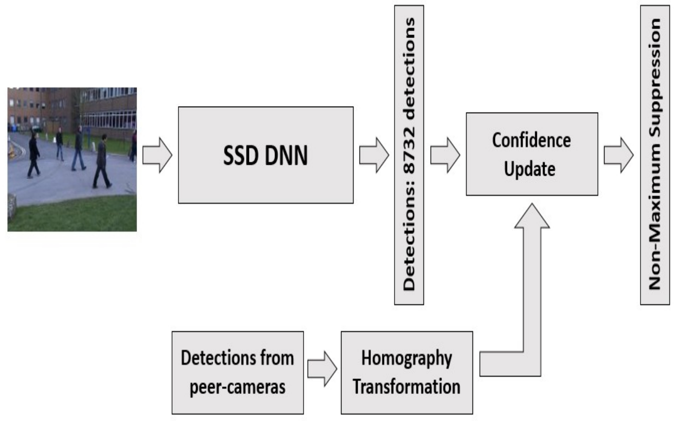
As mentioned before, Non-Maximum Suppression (NMS) is a post-processing step used in most of the object detection pipelines, which effectively assumes that bounding boxes with IoU higher than a threshold (typically, 0.7) refer to the same object, and thus suppresses (eliminates) one of the boxes. The CNMS approach performs collaborative fusion only by modifying this final NMS stage–i.e., it does not modify the core SSD DNN pipeline, but simply performs statistical fusion of the DNN's output (the set of bounding boxes). Figure 2a outlines the steps in the CNMS approach. CNMS takes the output bounding boxes of the SSD and *modifies* the output confidence value taking into account the overlap with bounding box coordinates obtained from each peer camera. Intuitively, CNMS will assign higher confidence scores to the bounding boxes (of the reference camera) that have high spatial overlap with the peer-generated bounding boxes. More formally, we first convert peer-generated bounding box coordinates to reference camera coordinates. Let us assume peer camera generates m bounding boxes of $P = \{p_1, p_2, ..., p_m\}$, while reference camera generates n bounding boxes $R = \{r_1, r_2, ..., r_n\}$ with n corresponding confidence scores represented by $C = \{c_1, c_2, ..., c_n\}$. We then compute $IoU(p_i, r_j)$ for all $i \in [1, m]$ and $j \in [1, n]$. If $IoU(p_i, r_j) > 0.8$, then we assign a new confidence score ${c_j}^{new} = (1 - \alpha) * c_j + \alpha * IoU(c_i, r_j)$, where $\alpha = 0.2$. Similarly, we execute the same confidence update procedure for multiple cameras and then we run the NMS step on this updated confidence scores.

B. Collaborative SSD (CSSD)

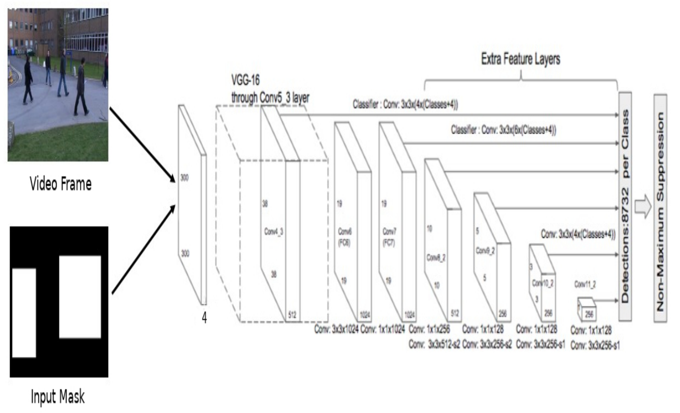
Unlike the CNMS approach, the CSSD technique modifies the structure of the SSD pipeline itself. Figure 2b illustrates the high-level CSSD approach: in this case, the input to the DNN is not just the individual video frames from the reference camera, but also an additional *input mask* with probable locations of objects (as indicated by the collaborating peer cameras). This input mask is calculated from the bounding box information from neighboring cameras. Similar to CNMS approach, we initially convert bounding boxes generated by peer cameras to the coordinate system of reference camera. Assume that the peer camera generates m bounding boxes $P = \{p_1, p_2, .., p_i, .., p_m\}$ where $p_i = ({x_i}^{min}, {y_i}^{min}, {x_i}^{max}, {y_i}^{max})$ (coordinates of the top left and bottom right of the bounding box). We then initialize the input mask of size $(300, 300)$ with zeros. Assume $pixel_{i,j}$ denotes pixel value of input mask at location (i, j). Then $\forall\, p_k (k \in [1, m])$, assign $pixel_{i,j} = 1$, if ${x_k}^{min} < i < {x_k}^{max}$ and ${y_k}^{min} < j < {y_k}^{max}$. Similarly, we do the same computation for multiple peer-cameras. Finally we pass the video frame from the reference camera and computed input mask into the CSSD DNN network (see Figure 2b).

C. Vulnerability to Adversarial/Malicious Behavior

While collaborative inference can improve detection accuracy, it is, however, susceptible to malicious or faulty behavior by peer nodes. As an example, illustrated in Figure 1, let us assume that Camera C is exhibiting adversarial behaviour–i.e. it does not report the true bounding boxes of people detected in its images. Since the inference output and feature summaries of Camera C are shared with Camera A and B, this will affect the inferencing behavior of both Camera A and B. Accordingly, it is necessary to understand the sensitivity of different collaborative techniques to such faulty or malicious behavior, especially to understand whether such incorrect peer information can cause a collaborative model to perform worse than a non-cooperative baseline.


SSD DNN
Detections: 8732 detections
Confidence Update
Non-Maximum Suppression
Detections from peer-cameras
Homography Transformation

(a) Collaborative NMS



(b) Collaborative SSD

Fig. 2: Architectures of (a) *Collaborative NMS* vs (b) *Collaborative SSD* models

IV. Multi-Camera Collaborative Sensing Test bed

To provide an effective IoT platform for testing such collaborative intelligence techniques, as well as their vulnerability to various forms of adversarial behavior, we have initiated the development and deployment of a medium-scale experimental IoT testbed on our SMU campus. The first version of the testbed involves only 25-30 IoT nodes deployed in public areas of the SMU campus, each consisting of a Raspberry Pi platform equipped with a camera sensor as well as a Movidius VPU (vision processing unit) to provide hardware support for in-situ execution of DNN pipelines. The Pi CPU is then responsible for fetching videos from the cameras, passing individual frames to the VPU for processing and handling the necessary communication between neighboring nodes. Note that VPU is included to explicitly capture *in-situ edge processing*: while CPU-based execution of DNN pipelines take $\sim$1 sec/frame, the VPU reduces the overhead to $\sim$ 85 msecs/frame. To enable peer-to-peer communication, one such IoT node is designated as the hub, to which other cameras connect via WiFi. This group of network connected Raspberry PI platforms is connected to a Web-based Dashboard at the back-end, which allows users to execute actions such as:

1) Easily add new cameras to the system and also configure the spatial relationships (the entries in the View Mapping DB) of newly added cameras
2) Record videos for future experiments and run real-time collaborative inferencing pipeline
3) Emulate real-time playback of, and experimentation with, previously recorded videos
4) Use a unified command line interface to control the configuration parameters of multiple IoT nodes.

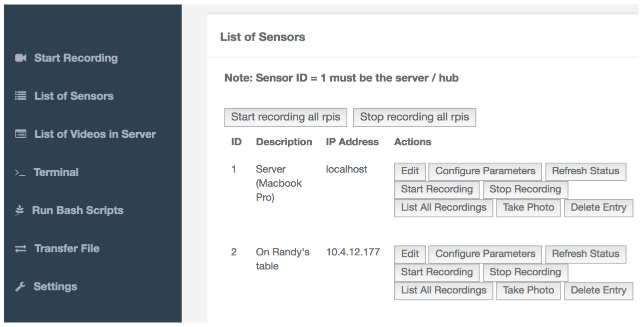
This Web interface also permits authorized remote access, thereby allowing researchers at international collaborating institutions to run experiments on this platform.

To provide a software system that generalizes beyond just the specific case of "people counting", the testbed infrastructure is configured as a flexible data processing pipeline where a new block (such as a new data source (audio,etc) or a new algorithm module) may be added in an ad-hoc manner as the research progresses, and even while experiments are being currently executed. Therefore, at the software architecture level, we implemented two things: (1) a flexible data processing architecture (described in the next section) (2) a web dashboard to act as a control centre to configure and schedule the data processing pipeline. The web dashboard (screenshots illustrated in Figure 5) includes the following functionality: (a) Addition of new cameras to the system; (b) Easy selection of cameras and video files (if they are being replayed) to be included in the upcoming experiment, and (c) Transfer of individual or multiple files (using *rsync*) between individual cameras and the server (which automatically indexes such file metadata for efficient future retrieval).

A. System Overview

Based on the above figure, each IoT node supports the following key functional components:

1) **Communication interface:** This module consists of processes to allow a camera to communicate with the control centre (to receive appropriate configuration parameters and commands) and also to all the other cameras. Each camera, as well as the control center, hosts a Web server, and uses HTTP messages to communicate among themselves. In addition, each camera has a direct UDP-based communication interface with each individual peer camera to transfer runtime data (such as the bounding box coordinates) efficiently.
2) **Event distributor:** This module interprets and forwards the instructions/data received from the communication interface to destination modules by either RPC or publish-subscribe mechanism. The instructions can range from starting/stopping processes (recording/inference), changing properties of the other running processes during runtime (e.g which inference model to use, which cameras to send/receive message from during inference stage), etc. By allowing hot swapping

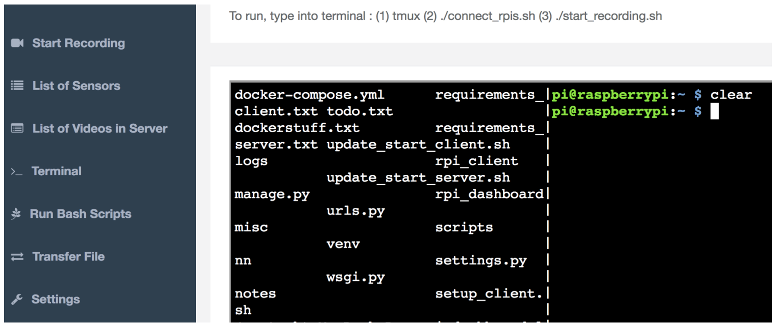


List of Sensors

Note: Sensor ID = 1 must be the server / hub

ID	Description	IP Address	Actions
1	Server (Macbook Pro)	localhost	Edit, Configure Parameters, Refresh Status, Start Recording, Stop Recording, List All Recordings, Take Photo, Delete Entry
2	On Randy's table	10.4.12.177	Edit, Configure Parameters, Refresh Status, Start Recording, Stop Recording, List All Recordings, Take Photo, Delete Entry

(a) Camera Configuration Settings



To run, type into terminal : (1) tmux (2) ./connect_rpis.sh (3) ./start_recording.sh

```
docker-compose.yml      requirements_|pi@raspberrypi:~ $ clear
client.txt todo.txt                  |pi@raspberrypi:~ $ 
dockerstuff.txt         requirements_|
server.txt update_start_client.sh    |
logs                    rpi_client   |
           update_start_server.sh    |
manage.py               rpi_dashboard|
           urls.py                   |
misc                    scripts      |
           venv                      |
nn                      settings.py  |
           wsgi.py                   |
notes                   setup_client.|
sh                                   |
```

(b) Command line interface

Fig. 3: Illustrative images for the Testbed Web Dashboard

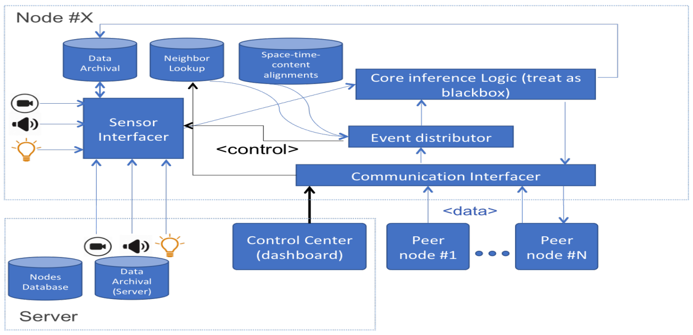


Fig. 4: Collaborative Sensing Testbed Architecture

of an individual node's inferencing module and collaboration messages, this module enables us to additionally simulate adversarial behavior by other nods (e.g., by instructing a camera to incorrectly perform random discard of bounding box information).

3) **Sensor interfacer:** This is implemented as a multi-process module that connects to the underlying sensor (video camera in initial setup), retrieve the corresponding data stream, perform appropriate buffering (e.g., discarding intermediate frames if the inference engine's throughput is lower than the frame rate) and send it to both the inference engine (for real-time machine intelligence) and also to the data archival component (for future trace-based playback and analysis). To help emulate various controlled experiments, the module also includes "virtual sensors" that effectively bind to, and replay, video files stored on the server.
4) **Core inference logic:** This module (which implements the baseline and SSD pipelines described earlier) contains the inference engine and processes to load specific models (e.g. different DNN models), execute the models and archive the inference results.
5) **Space-time correlator:** This module stores the spatiotemporal correlations among different cameras. In particular, if two cameras have overlapping views, the alignment will include the coordinate transformation matrix; on the other hand, if the two cameras are temporally correlated, this module will store the statistics (e.g., mean, variance) of the underlying temporal shift. Such shift is used by the Inferencing Engine, for example, to align the frames from different cameras for collaborative processing.
6) **Data archival:** This module stores and tags output data (video, bounding boxes) from other processes. Note that each output data may have its own custom format metadata–e.g., a video file may include parmeters such as resolution, location and recording duration. The server stores and indexes such metadata, allowing the dashboard interface to perform metadata search queries (e.g., 'find HD videos shot by camera 1 last Monday'). .

V. Evaluation

We now present our experimental results, which illustrate both the benefits of collaborative intelligence and the vulnerability of such approaches to adversarial behavior by individual nodes.

A. *Experiment Setting*

We use the PETS 2009 dataset [4] which consists of video feeds of 8 synchronous cameras in the outdoors, under varying crowd flow and density settings. The individual cameras record video at an approximate frame rate of 7 FPS and we consider 4 views (views 5-8) with considerable overlap (shown in Figure 5) in our evaluations. The resolution was fixed at 720 $\times$ 576. We processed a total of 3180 frames (i.e., 795 per camera) and consider the camera pertaining to View 005 as the *reference* camera with respect to which we report all our performance results. Because we focus only on the "people detection" task, we use the manually annotated ground truth from [15] which provides 2D annotations of 10 persons entering, passing through, staying and exiting the pictured area. We build on the Single Shot Detector (SSD)[1], proposed by [9] for object detection with model trained on the PASCAL VOC dataset [3] and focus only on the "person" object detections.

B. *Accuracy Improvements with Multiple Collaborating Cameras*

We evaluate our collaborative inferencing model on PETS with up to 4 collaborating camera views for view 5,6,7 and 8.

[1]Implementation available from https://github.com/weiliu89/caffe/tree/ssd

(a) Reference camera view (b) Collaborative camera 1 (c) Collaborative camera 2 (d) Collaborative camera 3

Fig. 5: Illustrative images from the PETS 2009 benchmarking dataset.

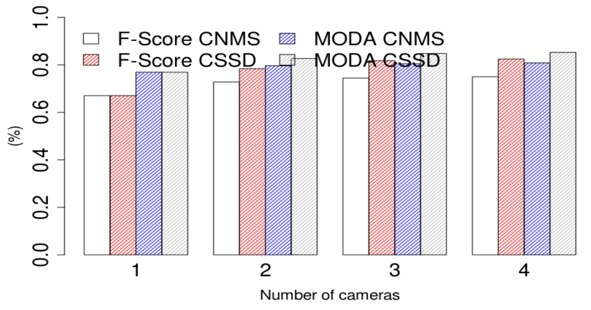


Fig. 6: Accuracy: independent vs. collaborative operation

In figure 6 we plot variation of *F-Score* and *MODA*[2] $y-$axis with varying number of collaborating cameras (N) in the $x-$axis ($x = 1$ corresponds to the baseline where camera 5 operates in isolation, purely using its own sensor data). We see that the addition of more collaborating cameras improves accuracy in both CNMS and CSSD. However, CSSD achieves significantly higher increase in accuracy ($\sim$10%) even with the addition of one collaborating camera. Moreover, we also see the law of diminishing returns in collaboration, with the performance gain saturating beyond $N = 3$ for both CNMS and CSSD.

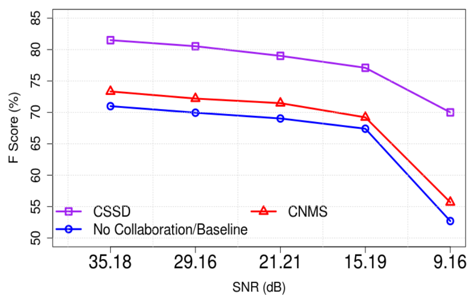


Fig. 7: Noise vs. Inferencing Accuracy.

Performance Under Noise: In the real world, cameras are susceptible to noisy inputs (e.g., due to poor lighting conditions). To further understand the benefits of collaborative inferencing, we also study the performance of *CNMS* and *CSSD* vs. the non-cooperative baseline under varying levels of Gaussian noise. Figure 7 plots the result, with the $x-$axis representing progressively smaller values of SNR (i.e., larger the noise, lower the SNR) and $y-$axis representing the corresponding detection accuracy (F-Score). We see that both *CNMS* and *CSSD* always outperform the comparative non-collaborative baseline. In addition, (a) even at high noise levels (variance = 100 and SNR $\sim$9dB), *CSSD* performance is similar to non-collaborative performance without any noise; and (b) both *CSSD* and *CNMS* are less sensitive to noise than a non-collaborative model (their accuracy dropping by $\sim$11%, compared to a drop of $\sim 19\%$ for self-inference).

[2]*Multiple Object Detection Accuracy*, a metric that normalizes the number of false positives and negatives by the number of ground truth objects

TABLE I: Comparison: CNMS vs CSSD

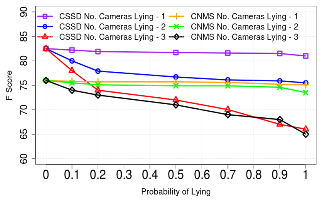
	CNMS	**CSSD**
Bandwidth per Node	150kbps	150kbps
Accuracy/F Score	75.5%	82.5%
Processing Latency	100ms	85ms

Additional Comparison: CNMS vs CSSD Besides accuracy, we also evaluated *CNMS* and *CSSD* on a couple of additional metrics, (a) the communication overhead (bandwidth/node) needed to sustain the collaborative exchange of bounding box information with peer cameras, and (b) the overall per-frame processing latency (the sum of VPU and CPU processing latency on reference node 5). Table I shows the results. We note that the communication overhead is identical (as expected) as both approaches require sharing identical information (bounding box coordinates). On the other hand, *CSSD* latency is lower than that of *CNMS*. This is due to the fact that, in *CSSD*, the entire processing pipeline is executed in the VPU, whereas in *CNMS*, the final step of confidence-weighted bounding box suppression is executed in the node's CPU.

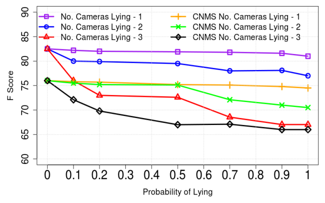
Key Takeaway: Overall, our results demonstrate that *CSSD*, which involves an explicit modification of the DNN pipeline and a corresponding collaborative learning phase, outperforms the *CNMS* approach which merely alters the final Bayesian inferencing step: it achieves both higher accuracy and lower processing latency.

C. Resilience to Adversarial Behavior

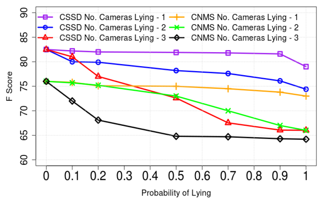
As a final study, we investigate the performance of the collaborative mechanisms proposed under adversarial conditions–



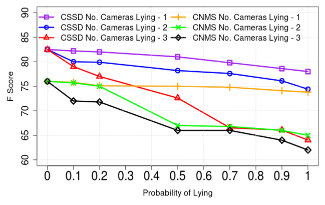
(a) Original detections



(b) Low Noise



(c) Medium Noise



(d) High Noise, Baseline $F-$score=52%

Fig. 8: Performance of (*CNMS* and *CSSD*) under Adversarial Conditions.

i.e., *when collaborating nodes falsely inject or suppress data*. To mimic such adversarial behavior, we configure each of the peer cameras to lie (falsely generate incorrect bounding boxes or suppress inferred boxes) with a variable probability p. Figure 8 plots the change in detection accuracy vs. p (for both *CSSD* and *CNMS*) under 4 different conditions, corresponding to different levels of noise. Moreover, we varying the number of 'lying' peers (from 1 to 3). The figures show that *CSSD* is significantly more robust to such adversarial behavior than *CNMS*, with its accuracy dropping marginally even for moderate noise and large values of p, *as long as the number of adversarial cameras is 1 or 2*. *CNMS*, on the other hand, experiences a sharper ($\sim$15%) drop in accuracy even under modest adversarial behavior ($p = 0.3$). The results show the promise of *CSSD's* collaborative learning approach, but also illustrate the need to build additional *explicit* mechanisms to protect such frameworks against adversarial behavior.

VI. Related Work

Edge computing technologies are pushing frontiers in enabling real-time analytics systems for situation awareness. Early examples of such systems have been described for various video-based applications and services [12, 13, 14]. Very recently, multi-device cooperation, at the edge, has piqued the interest of the research community (e.g., multi-camera systems [7, 8, 11], cooperative UAV swarms [1, 2], occupant authentication [5]) owing to its advantage of improving accuracy and reducing overheads in dealing with communication with a centralized cloud. As video processing using deep learning pipelines is considered resource-intensive, early efforts in enabling collaboration/cooperation between multi-camera systems explore cost-efficiency without sacrifice in accuracy. Qiu et al.[11] demonstrate the ability to track vehicles across a heterogeneous camera networks consisting of both fixed (e.g., surveillance) and mobile camera. By selectively activating the mobile cameras only to resolve ambiguities whilst much of the heavy-lifting of the video analytics pipeline is performed on the cloud, they achieve high accuracy without overly draining the resource-constrained mobile devices. Further, Lee et al. [8] show that by establishing space-time relationships between views of co-located cameras a-priori, significant savings in bandwidth needs can be achieved. They show that by selectively turning off (and on) downstream cameras in the network depending on the moving targets detected by upstream cameras and the respective likelihood of them appearing downstream, the amount of raw footage collected and uploaded to the cloud (for processing) can be reduced as much as by 238 times with a nominal miss rate of 15%. More recently, Jain et al [7] discuss alternative configurations of video analytics pipelines that are triggered by peer cameras that share spatiotemporal correlations between co-located cameras. The authors provide recommendations for cost efficiency (e.g., by reducing redundant processing by cameras sharing overlapping views) and higher inference accuracy (e.g., cross-camera model refinement).

VII. Discussion

This work, introducing the benefits and challenges of robust collaboration in adversarial environments, opens up the possibility of research along different dimensions.

Building Resiliency Against Adversarial Behavior: As illustrated in our results, collaborative intelligence approaches are vulnerable to incorrect data from or malicious behavior by one or more peer nodes. A variety of different approaches may help improve the robustness of collaborative sense-making by such IoT nodes. One approach (briefly introduced in [10]) is to create a Reputation-based mechanism, where each node updates a reputation score for each neighboring camera based on the observed mutual discrepancy between objects (e.g., using color histogram as a representative feature) occurring concurrently in a common field-of-view, and then using this score to modify its collaborative fusion logic. However, such reputation-based mechanisms may (a) lead to higher-than-ideal communication overheads, as building the reputation score may require the transmission of multiple object features (e.g., the color histograms) for every image frame between nodes; (b) not be suitable to accommodate other forms of correlation, e.g., when the same object is not seen concurrently, but with a varying time gap between different nodes; (c) be unable to capture transient or isolated adversarial behavior (e.g., when a camera drops the bounding boxes only for a specific individual of interest). Alternative approaches (e.g., based on cryptographic hash chains) need to be investigated to understand the tradeoffs between resiliency and metrics such as network bandwidth and computational overheads.

Autonomic Identification of Collaborative Devices: The collaborative frameworks (*CSSD* and *CNMS*) presented here implicitly assumed the *apriori* existence of a set of collaborative relationships between pairs of IoT nodes. In real-world environments, the set of IoT devices may change dynamically, and the ideal set of collaborating partners may change as well, depending for example, on changes in device locations and underlying movement patterns. Accordingly, we will need to develop frameworks that allow one or more IoT devices to autonomously identify the set of devices that share salient spatiotemporal correlations, and can thus benefit from such collaborative inferencing.

Incorporation of Additional Collaboration Constraints: The collaborative frameworks introduced here implicitly assume that the IoT resources are all under a single administrative domain (e.g., a single operator). Future operating environments may, however, involve the opportunistic and negotiated use of IoT resources belonging to different administrative entities [8] (e.g., a military unit seeking to utilize not just its own UAV-based sensors, but also surveillance cameras operated by a civilian authority or other camera nodes belonging to a coalition partner), with additional constraints: e.g., one may be able to collaborate with a coalition partner's cameras for no more than 2 hours daily. In such scenarios, the collaborative intelligence pipeline must adapt to contextual and environmental conditions to achieve the right tradeoff between accuracy and resource usage.

VIII. Conclusion

In this work, using people counting via multiple cameras as an exemplar application, we have introduced the notion of real-time collaboration as a technique for improving the accuracy vs. performance tradeoffs of complex "machine intelligence" tasks in a set of networked, resource-limited IoT devices. We introduced two different collaborative techniques: the *CNMS* approach utilizes the bounding box coordinates from peer devices in a final step of Bayesian fusion, whereas the *CSSD* approach creates a new DNN model, with an input vector that is augmented by an object mask created from peer devices. Through empirical results on the PETS dataset, we show that both collaborative approaches are superior (with around 5-14% accuracy gains) compared to a traditional self-inferencing approach, with *CSSD* showing overall superior performance in terms of accuracy, both in the absence and presence of noise. However, the drawback of such collaboration is greater susceptibility to inadvertent or deliberate failures or false information injected by erroneous or malicious nodes. We show that *CSSD* does provide resilience to such adversarial behavior, experiencing an accuracy drop of less than 5% when faced with a single 'lying' peer node. We anticipate that our work will spur research interest in developing collaborative ML-based mechanisms, for both *training* and *inferencing*, that take advantage of the spatiotemporal correlations among different nodes of uncertain fidelity.

IX. Acknowledgement

This material is supported partially by the Army Research Laboratory, under agreement number FA5209-17-C-0006, and partially by the National Research Foundation, Prime Minister's Office, Singapore under its International Research Centres in Singapore Funding Initiative. The view and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Army Research Laboratory or the US Government.

References

[1] Axel Bürkle. 2009. Collaborating miniature drones for surveillance and reconnaissance. In *Unmanned/Unattended Sensors and Sensor Networks VI*, Vol. 7480. International Society for Optics and Photonics.

[2] Xinlei Chen, Aveek Purohit, Carlos Ruiz Dominguez, Stefano Carpin, and Pei Zhang. 2015. DrunkWalk: Collaborative and Adaptive Planning for Navigation of Micro-Aerial Sensor Swarms. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys)*.

[3] Mark Everingham, S. M. Eslami, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. 2015. The Pascal Visual Object Classes Challenge: A Retrospective. *Int. J. Comput. Vision* 111, 1 (2015).

[4] James Ferryman and Ali Shahrokni. 2009. Pets2009: Dataset and challenge. In *2009 Twelfth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*. IEEE.

[5] Jun Han, Shijia Pan, Manal Kumar Sinha, Hae Young Noh, Pei Zhang, and Patrick Tague. 2018. Smart Home Occupant Identification via Sensor Fusion Across On-Object Devices. *ACM Trans. Sen. Netw.* 14, 3-4 (Dec. 2018).

[6] Richard Hartley and Andrew Zisserman. 2003. *Multiple view geometry in computer vision*. Cambridge university press.

[7] Samvit Jain, Ganesh Ananthanarayanan, Junchen Jiang, Yuanchao Shu, and Joseph Gonzalez. [n. d.]. Scaling Video Analytics Systems to Large Camera Deployments. In *In Proc. of HotMobile*.

[8] Jongdeog Lee, Tarek Abdelzaher, Hang Qiu, Ramesh Govindan, Kelvin Marcus, Reginald Hobbs, Niranjan Suri, and Will Dron. 2018. On tracking realistic targets in a megacity with contested air and spectrum access. *MILCOM* (2018).

[9] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. 2016. SSD: Single Shot MultiBox Detector. In *ECCV*.

[10] Archan Misra, Dulanga Weerakoon, and Kasthuri Jayarajah. 2019. The Challenge of Collaborative IoT-Based Inferencing in Adversarial Settings. In *The First International Workshop on Internet of Things for Adversarial Environments, INFOCOM*. IEEE.

[11] Hang Qiu, Xiaochen Liu, Swati Rallapalli, Archith J Bency, Kevin Chan, Rahul Urgaonkar, BS Manjunath, and Ramesh Govindan. 2018. Kestrel: Video Analytics for Augmented Multi-Camera Vehicle Tracking. In *Internet-of-Things Design and Implementation (IoTDI), 2018 IEEE/ACM Third International Conference on*. IEEE, 48–59.

[12] Mahadev Satyanarayanan. 2017. Edge computing for situational awareness. In *Local and Metropolitan Area Networks (LANMAN), 2017 IEEE International Symposium on*. IEEE, 1–6.

[13] Mahadev Satyanarayanan, Zhuo Chen, Kiryong Ha, Wenlu Hu, Wolfgang Richter, and Padmanabhan Pillai. 2014. Cloudlets: at the leading edge of mobile-cloud convergence. In *2014 6th International Conference on Mobile Computing, Applications and Services (MobiCASE)*. IEEE, 1–9.

[14] Pieter Simoens, Yu Xiao, Padmanabhan Pillai, Zhuo Chen, Kiryong Ha, and Mahadev Satyanarayanan. 2013. Scalable crowd-sourcing of video from mobile devices. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*. ACM, 139–152.

[15] Yuanlu Xu, Xiaobai Liu, Lei Qin, and Song-Chun Zhu. 2017. Cross-View People Tracking by Scene-Centered Spatio-Temporal Parsing.. In *AAAI*. 4299–4305.