ComAI: Enabling Lightweight, Collaborative Intelligence by Retrofitting Vision DNNs

Kasthuri Jayarajah*, Dhanuja Wanniarachchige†, Tarek Abdelzaher‡, Archan Misra†
University of Maryland Baltimore County, Singapore Management University†, University of Illinois at Urbana-Champaign‡*
kasthuri@umbc.edu, dhanujaw.2020@phdcs.smu.edu.sg, zaher@illinois.edu, archanm@smu.edu.sg

IEEE INFOCOM 2022 - IEEE Conference on Computer Communications | 978-1-6654-5822-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/INFOCOM48880.2022.9796769

***Abstract*—While Deep Neural Network (DNN) models have transformed machine vision capabilities, their extremely high computational complexity and model sizes present a formidable deployment roadblock for AIoT applications. We show that the complexity-vs-accuracy-vs-communication tradeoffs for such DNN models can be significantly addressed via a novel, lightweight form of "collaborative machine intelligence" that requires only runtime changes to the inference process. In our proposed approach, called *ComAI*, the DNN pipelines of different vision sensors share intermediate processing state with one another, effectively providing hints about objects located within their mutually-overlapping Field-of-Views (FoVs). *CoMAI* uses two novel techniques: (a) a secondary shallow ML model that uses features from early layers of a peer DNN to predict object confidence values in the image, and (b) a pipelined sharing of such confidence values, by collaborators, that is then used to *bias* a reference DNN's outputs. We demonstrate that *CoMAI* (a) can boost accuracy (recall) of DNN inference by 20-50%, (b) works across heterogeneous DNN models and deployments, and (c) incurs negligible processing, bandwidth and processing overheads compared to non-collaborative baselines.**

I. Introduction

To support *in-situ, pervasive* execution of machine intelligence tasks (e.g, vision tasks such as object detection), there is a pressing need to further improve the accuracy-vs.-complexity tradeoff associated with DNN-based inferencing on resource-constrained mobile, wearable and IoT devices. A variety of strategies, such as model distillation [14, 33, 38], run-time execution optimization [19, 9], workload partitioning [43] or edge offloading [24, 35], have been proposed to reduce the execution overhead of DNN-based inference without compromising accuracy. Through this work, we tackle the challenge: *is it possible to improve the inference accuracy of off-the-shelf vision DNN models, in a model-agnostic fashion, without requiring custom, deployment-specific model retraining or increasing their computational overhead?* Achieving such improvements would allow existing models, already deployed on pervasive devices, to continue to be executed locally but with significantly improved accuracy. In this work, we explore the paradigm of ***Co**llaborative **Ma**chine **I**ntelligence* (**CoMAI**). Our key idea is to opportunistically improve the inference accuracy of an already-deployed DNN model on a single pervasive device, by appropriately transferring and incorporating *correlated DNN state*, at real-time, between multiple such co-sensing devices.

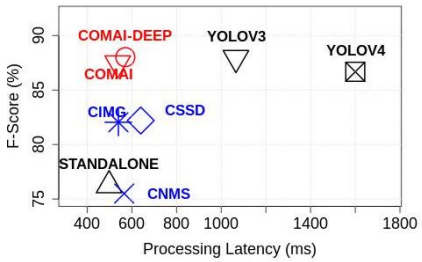
Our approach is motivated by the observation that, for many in-the-field vision sensing applications (elaborated in Section II): (i) edge/cloud offloading isn't practical, and (ii) the sensor deployment exhibits *partial spatial redundancy*– i.e., as exemplified in Figures 2a and 2b, large regions of the monitored space are simultaneously observed by multiple networked vision sensors/cameras *from different perspectives*. In such scenarios, *CoMAI* works by having an individual sensing node (a resource-constrained embedded device) execute its own DNN, but with its inferencing pipeline suitably modified or retrofitted (at runtime) to incorporate appropriate summaries of hidden-layer state of DNNs being executed by one or more *mutually overlaping, collaborating* peer nodes.

To be very precise, limited past work (e.g., [1]) has articulated the vision of collaborative machine intelligence at the edge, and also explored rudimentary strategies such as fusing the bounding box confidence values of DNN output layers [37, 4]. *CoMAI* is however is the first work to (a) identify the most effectual, bandwidth-parsimonious scene summaries that collaborating nodes can share, and (b) then modify the runtime execution of standard DNN inferencing models to take advantage of such shared summaries across collaborating sensor nodes. We shall show that the proposed *CoMAI* approach outperforms other plausible collaborative and non-collaborative baselines, achieving both significantly higher accuracy and imposing dramatically lower network bandwidth||energy||latency overheads.

We make the following **Key Contributions:**

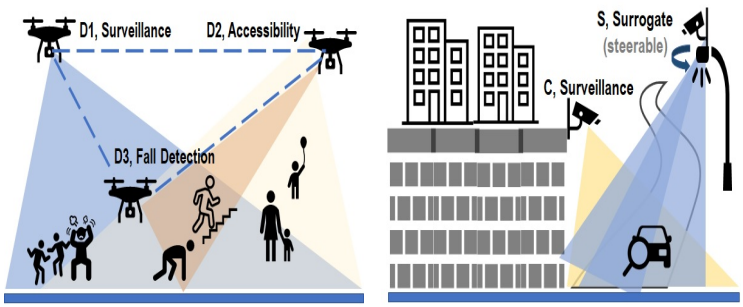
1) ***Establish the Basis for Beneficial Collaboration***: Using the VGG16-SSD, a low-complexity object detector, as an exemplar, we dissect the DNN pipeline to quantitatively show how detection errors, arising from phenomena such as occlusion, can be characterized in terms of (1) activation values of feature maps at the initial convolutional layers, and (2) confidence values at the output layers. Moreover, these errors are likely to be uncorrelated, and in fact *compensatory*, across cameras with overlapping but differing views, thereby laying the basis for how collaboration is likely to improve detection accuracy.
2) ***Develop Lightweight Collaborative Inferencing:*** We develop a computationally lightweight, bandwidth-parsimonious technique for extracting and sharing

*Work carried out while a Research Fellow at Singapore Management University



In red - our methods, blue - collaborative baselines, black- non-collaborative baselines.

Fig. 1: Comparison of accuracy vs. latency of *CoMAI* against other baselines on the PETS dataset.



(a) Drone-based Monitoring (b) Perimeter Surveillance

Fig. 2: *CoMAI* -based Scenarios

intermediate-layer DNN summaries across such partially overlapping nodes. The technique first identifies selected, highly-discriminative and deployment-agnostic feature maps from early convolutional layers of an object detector DNN, and then uses shallow classifiers to provide *fast, model-agnostic and accurate* (99% AUC on benchmark datasets) predictions of object confidence values at deeper, yet to-be executed stages of other peer DNNs. We further show that the scene summarization technique works across different deployments and is *model-agnostic:* we show that summaries extracted across five distinct models (VGG16-SSD, MobileNet-SSD, Yolo V2/3/4) can in fact be fused at the later stages of a peer DNN.

3) ***Demonstrate* ComAI's *Performance Benefits:*** We implement and deploy *CoMAI* on a multi-camera network and demonstrate its significant performance gains. Using trace-based emulation of two benchmark datasets (PETS [10] & WILDTRACK [8]), we show that collaborative confidence boosting can help increase recall by **20-50%** in overlapping areas (corresponding to 10-15% improvement in $F-$score, see preview in Fig. 1 for PETS) outperforming all other baselines with minimal communication overhead ($\leq$1 KB/frame/pair). *CoMAI* is very lightweight: it introduces $\leq$10-14% additional inference latency and incurs negligible memory overhead ($\leq 0.017\%$) over the baseline non-collaborative DNN.

Overall, we believe *ComAI's* importance lies in establishing the efficacy of the paradigm of *bandwidth parsimonious sharing of intermediate DNN state* as a means of runtime, model-agnostic collaboration across DNN pipelines to substantially improve their accuracy.

II. Illustrative Scenarios

ComAI's functional paradigm is motivated by use cases that involve multiple resource-constrained vision sensing nodes that (a) need to collectively monitor a designated region and (b) cannot effectively offload computation to a more powerful edge/cloud computing infrastructure. Figure 2 illustrates two such exemplars:

- ***Drone-based Monitoring:*** Consider a set of overhead drones being used by law enforcement to monitor crowd behavior at events, such as protests and concerts, over an urban area. The drones may remain largely stationary, with *partially overlapping* FoVs. Each drone may execute DNNs locally to perform tasks such as people counting, fall detection, and people profiling (e.g., for accessibility needs). Moreover, the drones may collaborate among themselves using a P2P wireless substrate (such as WiFi-Direct, which can support 10-20Mbps bandwidth over $\approx$100 meters), as transmitting image streams back to a terrestrial edge node for processing may impose prohibitive energy and bandwidth overheads.
- ***Base Perimeter Surveillance:*** Consider a set of vision sensors deployed (in close proximity) to provide intrusion detection alerts around the perimeter of a secure area (e.g., a military base). The sensors may have highly overlapped and/or steerable spatial coverage, and may only have a relatively low-bandwidth, long-distance wireless (several kms) to the base command post, used to transmit only processed events of interest (rather than raw image streams). In this scenario, the sensor nodes may be clustered in a hub-and-spoke fashion, with a cluster head executing the DNN inference pipeline and the other nodes acting as *surrogates* to help improve the inference accuracy.

While both scenarios conform to *ComAI's* vision of collaborative, local inference, they have one conceptual difference: for drone monitoring, each node executes its own independent DNN (as parts of the monitored area may be under the exclusive jurisdiction of an individual drone), whereas for the latter, the surrogate nodes run inferences merely to augment the accuracy of the DNN executing on the cluster head.

III. Collaboration Rationale

In this section, we analyze the computational logic employed by popular state-of-the-art DNN object detection pipelines to obtain both a deeper, empirically-driven understanding of their errors and insights where collaborative inferencing might help.

Real-time DNN Object Detectors: A Primer Modern object detector DNNs, such as YOLO [5] and SSD [27] adopt a *one shot* approach, where the model combines region proposal and object localization steps into a single deep convolutional network (DNN), thereby achieving lower inference latency than prior selective search based approaches [34]. Central to such single-shot detectors is the concept of "anchor boxes" –

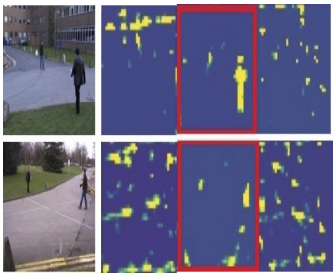


Fig. 3: Representative visualizations of hand-picked fmaps.

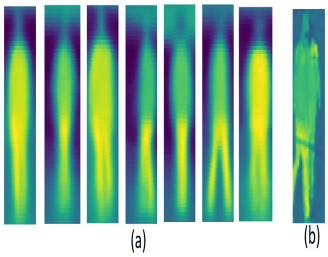


Fig. 4: Example fmap output for (a) correct & (b) incorrect (occluded) detection

Fig. 5: Example: Same person captured/missed across views.

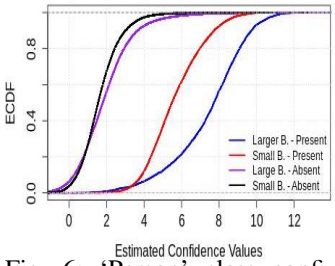


Fig. 6: 'Person' class confidence values: Small vs. Large anchor boxes.

a discrete set of pre-configured boxes, of varying sizes and aspect ratio, tiled across an image view, that represent *likely* positions/sizes of different objects. The runtime inference consists of detecting the presence of an object at each anchor box, computing the class probabilities and positional adjustments/offsets of these boxes – achieved by passing an image, through several convolutional layers (feature extractors), followed by the SoftMax layer (where the highest probability class is selected, per anchor box), and then applying the Non-Maximum Suppression (NMS) step which removes duplicate boxes. In this work, we take the VGG16-SSD detector as the exemplar DNN on which we implement *CoMAI* – to show performance benefits from collaboration.

Benchmark Data: We use video feeds from multiple synchronized cameras present in the PETS2009 [11] (4 annotated views with 795 frames each [40]) and WILDTRACK [8] (7 views with 400 frames each) datasets.

A. Dissecting The DNN Pipeline

We now investigate the causes of errors at different stages (both early and late) of the object detector pipeline through empirical analyses of the DNN's intermediate states.

Early Feature Extractor Layers: Vision DNN models typically consist of a cascade of 2-D convolutional filters in the early stages, with the optimal weights of such filters *learned* during training. It is well recognized that these hidden layers often contain *interpretable, useful information* [26, 6]. As an example, Figure 3 visualizes the activation values of the final convolution layer (for 5 handpicked filters) of the $VGG16$ feature extractor for 2 sample PETS camera frames–we can see that the high activation values (marked in yellow) of the second filter (red rectangle) correspond well with where human objects are present. Via empirical analysis, we observe that if an image's *distribution of the feature map (fmap)* activation values–i.e., outputs of the DNN's convolutional filters–differ significantly from the distributions seen during training, the DNN is likely to filter *out* the person class in the subsequent stages. Moreover, this distributional discrepancy often arises when there's a partial or occluded view of the person or when the person is in a position that is not distinguishable from other object classes (as is the case in Fig. 4, (b)).

Late Classification/Predictor Layers: Even if a person object's presence (via suitably high activation values) is retained by the convolutional layers, the SoftMax layer can filter out the person object if the eventual class confidence is low. As seen in Figure 5 where the same person appears unoccluded across two different views, the DNN detects the person in the right image (using a larger anchor box), but fails in the left where the person is further away (corresponding to a smaller anchor box). Figure 6 plots the CDF of estimated confidence values for the "person" class, for two categories of anchor boxes ("small", corresponding to outputs from SSD's first predictor layer, and "large", corresponding to outputs from the sixth predictor layer), separately when humans are present and absent. Besides confirming (as expected) that confidence scores are lower when human objects are actually absent, the plot also reveals larger anchor boxes have a significantly higher confidence value than corresponding smaller anchor boxes, confirming that *object detector DNNs have a harder time in identifying smaller, more-distant objects.*

B. Insights: Opportunities for Collaboration

These observations suggest that inputs from collaborating camera nodes may offer two distinct benefits:

- **Confidence Manipulation (Late):** Due to differences in perspective, an object that is smaller or occluded (resulting in a lower-confidence anchor box) in one camera view may appear to be larger or un-occluded in another view.
- **Convolutional Features (Early):** Suitable, discriminatory feature maps exist even in the early layers of a DNN that may be leveraged to assimilate hints about scene.

Accordingly, the disparity in perspectives from different overlapping cameras (after suitable spatial remapping) should help improve the DNN's ability to detect smaller objects and in improving its resiliency to such occlusion effects. In the next two sections, we describe how *CoMAI* systematically builds on these insights to develop an inference mechanism that improves accuracy by sharing suitable confidence values and feature maps across collaborator cameras.

IV. *CoMAI* Overview

We now introduce the high-level *CoMAI* architecture and justify some of the underlying design choices. We first observe that the general idea of collaboration can be embodied via a range of alternatives, ranging from the exchange of raw sensor data (image frames) to high-level Bayesian fusion of each sensor device's outputs.

- *Image Sharing:* While raw image frames and/or multiple intermediate layer feature maps can be exchanged and provided as additional input channels to object detector DNNs [16, 7], such an approach is limiting as it (a) requires extremely high inter-peer network bandwidth (transferring 30fps HD video frames requires O(10 Mbps) bandwidth), and (b) involves training of custom, non-standard multi-view DNNs and assumes that each peer executes an identical DNN model (DNN homogeneity).
- *Late (Output) Fusion:* Approaches such as [37] perform 'late fusion', collaboratively modifying the final NMS step. However, as we shall demonstrate, such approaches offer only modest performance improvements, as they are unable to address the *fmap* and predictor layer errors described previously.

Based on these observations, *CoMAI* adopts an alternative, intermediate approach of executing collaboration via performing 'fusion' at the *intermediate/hidden layers* of DNN models.

A. Design Goals

No custom re-training of the detector DNN: Any approach that involves deployment-specific DNN re-training, or custom architectures requires manual annotations across multiple views, and typically, a fixed set of collaborating peers. *CoMAI* is thus designed for adaptive runtime execution that does not alter the internal structure of a single-view DNN, but only *accesses* and *edits* intermediate outputs to enable collaboration.

Minimal network & computational overhead: To support the bandwidth-constrained field deployments described in Section II, *ComAI's* collaboration paradigm should be bandwidth-parsimonious and incur minimal additional latency.

P2P paradigm: *ComAI's* should support in-situ DNN execution by each sensor node independently, without assuming any computation offloading to an edge/cloud device; accordingly, *CoMAI* should provide performance gains even for low complexity DNNs. Under this paradigm, different sensor nodes may be executing different tasks (such as person re-identification, people counting and spotlight query [20, 21]) with object detection being common task primitive.

DNN heterogeneity: *ComAI's* model of runtime DNN modification should also be model-agnostic and support *cross-model collaboration*–i.e., for widespread use, *CoMAI* should be able to support interoperability between heterogeneous DNNs (such as SSD and Yolo V3) being executed by different sensor nodes.

B. System Architecture

Figure 7 provides a component-level overview of the proposed *CoMAI* framework. For ease of exposition, we shall describe the *CoMAI* components via reference to a *Reference* a *Collaborating* node pair, although in practice *CoMAI* nodes will assume both personas simultaneously, both sending scene summaries (such as *fmaps*) to other cameras and also consuming such digests (sent by other peers) to enhance their own DNN inference pipeline.

We assume that each peer node has processing capabilities on-board (e.g., CPU or a mobile Vision Processing Unit (VPU)). The peer nodes directly exchange digests to/from chosen peers. The Collaborator node's *Feature Selector* learns correspondences between its intermediate layer feature maps and the final inferences to empirically identify feature maps that best provide useful class-specific information. The *Shallow Learner* **predicts** scene summaries (e.g., class probabilities), and initiates parallel streaming to peers while inferencing.

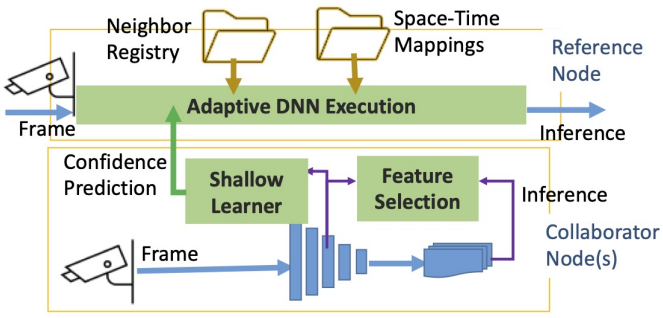


Fig. 7: Proposed *CoMAI* Architecture

To facilitate the collaboration process, each node's *Neighbor Registry* contains look-up information on local addresses of collaborating peers, while the *Spatial Mapper* contains the information needed to translate inferences from a collaborating peer's spatial coordinates to the reference node's coordinate space. In this paper, we focus primarily on establishing the mechanisms of such collaborative feature sharing, via detailed studies of the *Feature Selector*, *Shallow Learner* and *Adaptive DNN Execution* components. Accordingly, we assume that the contents of the Neighbor Registry and Spatial Mapper have been populated *a priori*, by out-of-band, well-established techniques. However, while evaluating *ComAI's* performance, we shall incorporate the inevitable errors that arise from practical execution of such out-of-band mechanisms (e.g., during spatial mapping).

V. Lightweight Collaborative Intelligence

We now introduce our three-stage lightweight methodology for perspective sharing between collaborating nodes for enhanced person detection. We first identify a set of output activation values ($fmaps$) from early convolutional layers (shown in Section III-B to provide reliable hints of the objects present in the scene); these features are then used to train a shallow classifier that predicts the target class probability for each anchor box at the final decision layers. Finally, these predicted class probabilities serve as *collaboration summaries* that are transferred to the reference DNN, which uses them to improve its own inference. We argue that the proposed approach, of early-stage prediction of class probabilities, provides two key advantages over other alternatives: (1) sharing class probabilities as summaries incurs far less bandwidth overhead compared to sharing raw images or full frame feature maps, (2) such predictive sharing parallelizes the network transfer from the collaborating node with the receiving node's execution of its DNN's later stages, thereby potentially eliminating the "blocking delay" (see Section VI-C).

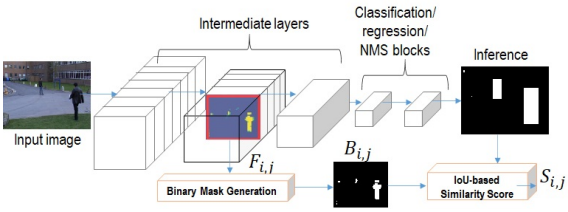


Fig. 8: Feature Map Selection Process.

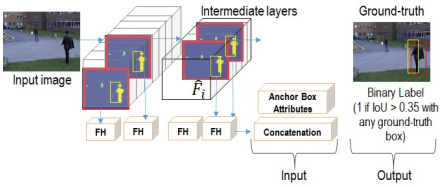


Fig. 9: Shallow Learning Process.

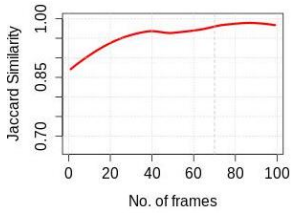


Fig. 10: Convergence in the set of chosen *fmaps* with increasing number of frames.

A. Lightweight Perspective Summarization

We first devise a simple mechanism to systematically identify, for the given object class, a small set of convolutional layer filters (feature maps (*fmaps*, for short) from multiple DNN layers that have high *correlation* with the presence/absence of objects at the final output layer of the DNN. We note that, in general, (a) each convolutional layer of the DNN consists of multiple filters, each of which generate a two-dimensional *fmap* of activation values (e.g., VGG-SDD's first convolutional layer has 64 filters each of size $300X300$), and (b) deeper convolutional layers have a larger number of low-resolution filters–i.e., filters with higher perceptive field, so as to detect the presence of larger objects.

1) Stage 1: Fmap Selection: First, during a training phase, we mine through the fmaps from all layers to determine the most *discriminative* filters that are most indicative of the presence of the target object class (i.e., "person" in the current work). As illustrated in Figure 8, this includes the following steps:

1) **Inference:** Each frame f is passed through the detector DNN to produce each intermediate fmap, $F_{f,i,j}$, at each layer i, and filter j, and finally, the inference, I_f (i.e., a series of bounding boxes).
2) **Binary Mask Generation:** For each frame f, we binarize the activation values of each fmap $F_{f,i,j}$ as follows: activation values greater than the q^{th} percentile are marked as "1", with the remainder marked as "0". The fmaps are then re-scaled and interpolated to match the dimensions of I_f. I_f is also binarized where the regions within the bounding boxes are filled with "1" and "0" elsewhere.
3) **Similarity Score-based Fmap Selection:** For each such **binary** fmap, $B_{f,i,j}$, we compute the similarity score, $S_{f,i,j}$ as the Intersection-over-Union (IoU) between the two binary masks $B_{f,i,j}$ and I_f. Finally, for each layer (i), we pick the top-k fmaps ($\hat{F}_i$) with the highest average (over N frames) similarity score as the "chosen fmaps".

Figure 10 plots the similarity in the set of $top-k$ fmaps chosen (as a Jaccard index, on the $y-$ axis) as a function of the number of training frames, N; we see that the set of *selected* fmaps converges rapidly, in as few as 70 frames with $q = 80$, set experimentally. We reiterate that fmaps extraction process (a) **does not require human annotation** as the correspondence is established based on the inferred detections (with imperfect accuracy), (b) **converges fast** and (c) is **deployment–independent**: in Section VI-A, we shall show that the same filters work well for both PETS and WILDTRACK datasets.

2) Stage 2: Shallow Classification: In this stage, as illustrated in Figure 9, we train a shallow ML-based classifier that uses this set of top-k *chosen* fmaps ($\hat{F}_i$) from the early convolutional layers to predict class-specific probabilities (comparable to the DNN's predictor layer outputs). Given an image frame (f) and an anchor box space (A), the classifier predicts the target-class probability, for each box $a \in A$, by using the following features:

1) **Features based on fmaps:** After executing layer i of the DNN, we consider the concatenation of feature summaries from all layers from 0 to i. For each instance a, we compute a histogram of activation values of length h falling within a as $FH_{i,a,k}$ where k is one of the top-k maps in $\hat{F}_i$. The final set of inputs is a concatenation of $FH_{i,a,k}$ over $i \in (0, ..., i)$ and all k, of length $h * i * k$.
2) **Anchor-box Features:** Additionally, we also consider the following parameters pertaining to each box a: L_a - the normalized (x, y) image coordinate around which the anchor box is centred, S_a - the anchor box's scale, and R_a - its aspect ratio.

The shallow model performs a binary classification task, predicting whether the corresponding anchor box a will have an object instance after the DNN is fully executed. For training, for a given frame, an anchor box a overlaps (with IoU $\geq IoU_{min}$) any person object found in the ground-truth, then a person object is considered to be present (labelled '1', and '0' otherwise). As empirical results show that selecting just the top fmap for any layer ($k = 1$) suffices, we use $q = 80, k = 1, h = 10$ for our *CoMAI* implementation. We fix anchor box space A to be the same as the default boxes of the VGG-SSD model for simplicity. In Section VI, we show that this shallow classification is highly accurate (with AUC ≥ 0.98) and generalizes across different object detector DNN models and deployments.

B. Modified DNN Inference

We now describe how the receiving (reference) node *CoMAI* uses perspective summaries shared by collaborating peers to *bias* the confidence values computed by its own DNN, and thereby improve inference accuracy. We assume that a spatial mapping function or matrix ($SM_{(C,i,R)}$) exists to translate

between the pixel coordinates of a reference camera R and its i^{th} collaborator. We also note that this biasing process occurs at runtime and does not require any retraining or structural modification of the DNN model.

1) The reference camera executes its DNN inference pipeline until the layer preceding the classification layer (typically, a soft max layer). At this point, the DNN has computed class probabilities for each trained class and the background class, for each pre-configured anchor box (CV_R).
2) In parallel, the reference camera also receives the confidence values $CV_{(C,i)}$ for each collaborator i for regions that mutually overlap (suitably re-mapped spatially as $CS_{(R,i)} \leftarrow Transform(CV_{(C,i)}, SM(C,i,R))$. Driven by our earlier observation of higher confidence values associated with larger anchor boxes, we selectively transfer only values from a collaborator's larger anchor boxes with high confidence to the reference camera's smaller anchor boxes (where the reference camera is more prone to errors)
3) Then, for each anchor box, if the cumulative confidence values of the reference and collaborating peer(s), $CV_{R,p} + \sum_i CS_{i,p}$ is above a class-specific threshold ($t_{(p)}$), we *bias* the class-specific confidence values for the p class by replacing the DNN's confidence values with this cumulative value.
4) The reference DNN then continues to execute as normal, forwarding these partially manipulated predictor-layer confidence values to the subsequent (soft max) layer for object classification.

The choice of t_p depends on the detector DNN. For instance, for VGG-SDD, we empirically set $t_{(p)} = 0.3$, which improves overall *recall* while maintaining *precision* comparable to that of the non-collaborative-SSD baseline (see Section VI-B).

VI. Evaluation

We present a systematic evaluation of *ComAI*, starting with evaluations of the individual components of perspective summarization (Section VI-A) and modified DNN inference (Section VI-B), followed by an system-level evaluation of a prototype *ComAI*implementation (Section VI-C).

A. Result: Lightweight Perspective Summarization

We report on the accuracy of the shallow ML-based binary classification task, which *predicts* the likelihood of a *person* object being *present* (labelled as the positive class) of each frame, over the discretized anchor box space.

Binary Classification: Each $(frame, anchorbox)$ instance that shares an overlap of at least iou_{min} (empirically set to 0.35) with any person bounding box in the ground-truth annotations are labelled as "1" (and "0" otherwise). We learn a Gradient Boosting Machine using 80% of the total samples (i.e., each sample referring to an instance of an anchor box in a frame, $N_{total} = 2,951,078$) for *training*, 10% of the samples for *validation* and report results using the held-out, 10% of the samples. As instances with a "person" object is the minority class, we sub-sample to create a balanced dataset ($N_{balanced} = 24372$). For each instance, we construct a set of features based on both the feature histograms (corresponding to the top-k *fmaps* (see Section V-A1) and the characteristics of the anchor boxes. We report the precision, recall and AUC of classifying whether a specific anchor box in a frame *contains* a person object, or not, for the VS (VGG16-SSD) detector on the PETS dataset, and then provide summarized evaluations from extending the technique to multiple detectors (**MN** (MobileNet-SSD), **YV2** (Yolo V2), **YV3** (Yolo V3), and **YV4** (Yolo V4)) as well as the WILDTRACK dataset.

Feature Set Vs. Accuracy: In Figure 11a, we report the average precision, recall and AUC, for different feature combinations – (a)$H-$ histograms only, (b) $HL-$ H + anchor box location, (c) $HLS-$ H + L + scale/size box, and (d) $HLSA-$ all features. Based on a **single histogram** derived from the top-1 *fmap* from the very first convolutional layer of VS, we see that $HLSA$ achieves precision, recall and AUC ($\approx 99\%$) all above 95%.

DNN Depth vs. Accuracy: Next, we investigate whether fmap-based features from deeper layers leads to better accuracy. The feature set for a depth of d consists of concatenated feature histograms (again, $k = 1$ fmap from each DNN layer) from all layers $\{1, 2, \ldots, d\}$. Figure 11b plots accuracy vs. d. While accuracy improves, as expected, as we utilize fmaps from deeper layers (94% for $d = 1$ vs. 96% for $d = 2$), the accuracy improvement effectively saturates beyond $d = 4$.

Training Frames Required vs. Accuracy: In Figure 11c, we vary the number of frames used to train the shallow classifier. If the first m frames (ordered by time) are used for training, we use the remaining $N_{balanced} - m$ frames for validation and testing. Note the classification accuracy remains fairly stable, with AUC $\geq 90\%$ even with only 30 training frames being used for training, demonstrating that the shallow classifier requires negligible training effort.

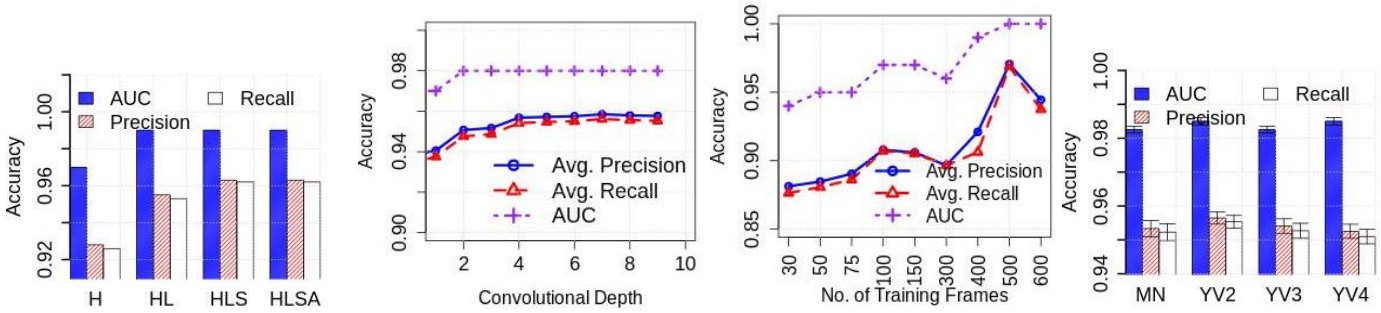
Generalizing the Results: In Figure 11d, we plot the average accuracy over all views in the PETS dataset (with $d = 1$ and $HLSA$ features) for other alternative detector DNN models. While the specific *fmap* chosen depends on the DNN model, we observe that the prediction accuracy remains consistently high (AUC $\geq$ 0.98 in all cases). Furthermore, in Figure 12, we report the accuracy on the WILDTRACK dataset for the different DNN models (AUC $\geq$ 0.98 in all cases). These results demonstrate that our approach is *model* and *deployment agnostic*.

In the following section, we use the predicted class probabilities from collaborating nodes (on the test sets) to *bias* predictor layer confidence values of reference cameras, in *CoMAI* and its variants.

B. Results: Modified DNN Inference

In this subsection, we quantify the improvement in inference accuracy *ComAI*achieves (by using the predicted class probabilities to bias the predictor-layer confidence values), comparing it to other early and late fusion alternatives.

Performance Metrics We report the precision, recall and F-score of the person detection task for each view in the datasets (as the "reference" camera) with all or a subset



(a) Feature Combinations. (b) Depths of the DNN. (c) No. of frames used. (d) Average over all views for different DNNs

Fig. 11: Accuracy of Shallow Classification under different settings on the PETS2009 dataset.

of the other cameras serving as "collaborators". We choose a class confidence threshold of 0.5 and an NMS threshold of 0.75 and 0.95 for the PETS (sparsely populated scenes) and WILDTRACK (more densely populated scenes) datasets, respectively.

Baselines and Variants: We compare *CoMAI* with the following collaborative and non-collaborative baselines based on the VGG16-SSD detector.

1) $Standalone$: This represents the non-collaborative, off-the-shelf DNNs (i.e., trained on monocular view datasets such as PASCAL/VOC[1]).
2) $CIMG$: This model represents collaborative configurations where cameras exchange the raw images. Note that this approach does not change the underlying structure of the DNN model, but changes its input (a concatenation of the reference and perspective-transformed collaborator image channels) and thus requires retraining.
3) $CSSD$ [37]: In this alternative, a collaborator camera first executes its entire DNN inference pipeline, and shares its detection result (as a binary mask) with the reference camera that then executes a custom-trained, multi-channel SSD model. As the reference camera *waits* for input from others before executing its own inference pipeline, this model introduces the most delay.
4) $CNMS$ [37]: In this 'late fusion' approach, collaborating nodes exchange their detection results (similar to $CSSD$), but the reference DNN fuses the output bounding boxes (i.e., the results post-classification and NMS steps).
5) $ComAI-DEEP$: A variant of *CoMAI* where the actual (as opposed to *predicted*) confidence values, obtained from executing inference till the deeper predictor layers are shared. As perspective sharing can be performed only after these deeper layers have been executed, this approach (similar to CNMS) is unable to parallelize the network transmission of summaries with the execution of later DNN stages.
6) $ComAI-SURROGATE$: A variant of *ComAI*, designed to support scenarios such as Figure 2(b), where the collaborator node exists merely to assist a *reference* node. The collaborator thus only executes its DNN *partially*, till the depth needed to extract shallow features, predict class probabilities and transfer the summaries to a reference node.

[1]http://host.robots.ox.ac.uk/pascal/VOC/

Single Collaborator: In Table I, we first report the precision, recall and F-score for every pair of views (in PETS) with *ComAI*-DEEP (the model that offers the highest improvement in accuracy, as we shall see in Section VI-C) implemented for the VGG16-SSD (VS) pipeline. We observe that a significant gain accuracy, *within the overlapping regions*, with even a single collaborator. For instance, with chosen collaborators (highlighted in green), each reference DNN's recall improves by **10-28**%. Note that the improvement in accuracy does not monotonically increase with an increase in the degree of spatial overlap, but also depends on other spatial properties– e.g., while camera view 5 has an ≈80% overlap with camera view 6, the resulting improvement is minimal, very likely due to the high similarity (and resulting low information gain) between the two perspectives.

Multiple Collaborators: Figure 1 (in Section I) summarizes our findings on average accuracy (across all views) achieved by *ComAI*vs. other baselines and alternatives on the PETS dataset. *ComAI*-DEEP achieves the highest gain in $F-$score (over 12%, due to $\geq$ 20% improvement in recall over the non-collaborative $standalone$ detector). Both *CoMAI* -DEEP and *CoMAI* achieve at least $\geq$ 5% improvement in $F-$score over the end-to-end re-trained $CIMG$ model; more importantly, *with such collaboration, even a shallow SSD model achieves accuracy comparable to significantly more complex, deeper models such as YoloV3 & YoloV4.*

Furthermore, we evaluated *CoMAI* with shallow class probabilities shared by **heterogeneous detector DNNs** (i.e., MN, YV2, YV3 and YV4) and achieved comparable gains ($F-$scores of 87.40%, 87.40%, 87.42%, and 87.49%, respectively). On the **WILDTRACK** dataset (which contains many small, distant human objects), we observe even more significant accuracy gains: while the non-collaborative $standalone$ baseline achieves $F-$scores of 12-37%, *CoMAI* achieves F-scores of 21-57% due to a *doubling of the recall values* (from ≈20→≈40%) achieved by the confidence biasing approach.

While we omit more detailed analyses due to space constraints, our findings establish that collaborative DNN inference, with only runtime modifications to the execution

Processing step	Latency (ms)	Complexity	Memory (MB)	Power (mW)
(1) $DNN_{input,output}$	486.371	26,285,486	2674	4803
(2) Shallow classification [3]	150+42	4500	349	1701
(3) Confidence Biasing	38.847		111	356
(4) $DNN_{output,end}$	11.965		114	416
(5) Communication (**Always-On**)	16.93		210	1884
$standalone$ $(1+4)$	498.33	26,285,486	2788	5219
ComAI-DEEP $(1+3+4+5)$	571	**26,285,486**	3442	7459
CoMAI $(max(1,2+5)+3+4)$	536.80	**26,289,986**	3442	9160
ComAI-SURROGATE	209	6292		
$CIMG$	3.95secs	26,285,486	2788	8188

TABLE II: *CoMAI* System Performance vs. baselines.

pipeline, can generate significant accuracy gains.

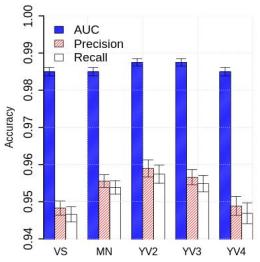


Fig. 12: Accuracy on the WILDTRACK dataset

Ref	Co	*standalone* P	R	F	*ComAI − DEEP* P	R	F	OR
5	6	92.84	75.85	83.49	93	79.78	85.88	83.1
5	7	93.21	75.88	83.66	92.85	81.1	86.58	72.9
5	8	93.21	75.88	83.66	93.18	85.28	89.05	61.9
6	5	93	58.09	71.52	93.39	82.48	87.59	77.1
6	7	93.33	58.33	71.79	94.03	70.65	80.68	61.2
6	8	92.8	59.06	72.18	93.24	87.28	90.16	71.9
7	5	94.83	67.07	78.57	92.59	84.38	88.29	60.9
7	6	94.83	67.01	78.53	92.03	81.32	86.35	55.1
7	8	94.83	67.01	78.53	92.7	76.29	83.7	56.8
8	5	93.66	66.26	77.61	93.37	70.41	80.28	40.5
8	6	93.66	66.26	77.61	93.01	76.33	83.85	50.7
8	7	93.71	66.26	77.63	93.07	74.07	82.49	44.4

TABLE I: Accuracy Gain with **single** collaborating pairs. (P-Precision, R-Recall, F-Fscore, OR-Overlap%)

C. System Performance: Putting it All Together

In this section, we report on the system performance of *CoMAI* against the baselines.

1) System Implementation: : We implemented and deployed *CoMAI* on a 4-camera network, which provides a trace-driven emulation of the benchmark datasets. Each camera node is a Raspberry Pi Camera (v2 module with 8 MP resolution, horizontal and vertical FoV of 62.2°and 48.8° respectively) connected to a Nvidia Jetson Nano board[2] equipped a 128-core Maxwell GPU. a Quad-core ARM A57 CPU and 4 GB system memory. The nodes communicate over a private WiFi network using the MQTT protocol.

2) Latency and Model Complexity: In Table II, we provide a breakdown of the processing and network latency, and the memory overhead. A key observation here is that the additional latency/complexity of *CoMAI* is only marginal compared to the accuracy gain that it is able to achieve – i.e., *CoMAI* only requires $\approx 17\%$ increase in the total latency (compared to the $standalone$ model) and is lighter than all other collaborative baselines. Similarly, the overall model complexity only increases by a marginal 0.017% (with the shallow learner requiring an additional $3 \times (|f| = 15) \times (ntrees = 100)$ parameters), where $|f|$ denotes the number of features and $ntrees$ is the number of trees used in the shallow Random Forest classifer.

3) Power Consumption and Memory Utilization: Table II further captures the power and memory consumption of the different stages of *CoMAI* in comparison to the baselines. While the power consumption increases (compared to $standalone$) due to the network transmission overhead, we note that this is true for any collaborative model. However, we point out that this is a pessimistic estimate as the network interface was kept active over the entire DNN execution duration; dynamically turning the WiFi interface ON/OFF should substantially reduce the network power overhead.

4) Communication Overhead: In the *CoMAI* -DEEP mode, both the reference and collaborator cameras run the DNN execution till their output layer, and then initiate the exchange of confidence values. We define the time taken for this transfer, over the network, as the *blocking delay*, as the DNNs are *blocked* from further execution until the collaborator summaries arrive. The blocking delay of $CIMG$, however, is more pronounced as the DNN execution halts until the reference camera receives the raw image, respectively.

The communication overhead, from i to j, $comoverhead(i,j) = s \times \frac{|SM_{(i,j)|}}{b} \times n_{classes} \times f$ where s is the size of the output (of confidence values), b is the total number of anchor boxes ($|A|$ =8732 in our case), $|SM_{(i,j)}|$ is the number of boxes for which there is overlap (or, a mapping exists), and $n_{classes}$ is the proportion of class confidences per box to be sent (i.e., 1 out of 21 classes as we consider only the 'person' class). f refers to a quantization level factor. If the confidence probability values are rounded up to 2 (corresponding to a f factor of 0.334), the average overhead across all views, per frame $= 717 \times \frac{784}{8732} \times \frac{1}{21} \times 0.334 = 1.024$ KB which translates to a network latency of $\approx$ 16.93 msec with three collaborators. On the contrary, the $CIMG$ baseline requires raw images to be sent over the network: scaled at the DNN's input resolution of 300X300, each frame incurs a bandwidth overhead of ≈300 KB (without compression) and a dramatically higher transfer latency of ≈3.95secs.

In the *CoMAI* mode (with $d = 1$), the total time taken to generate and transfer the probabilities is the combination of (time taken to execute layer 1, time taken for shallow classification and the delay value from above), averaging $150+42+16 = 209$ msec, with three collaborators. Note, however, that the reference camera's DNN execution latency is higher than this transfer latency, thereby effectively eliminating any blocking delay. Moreover, as the number of collaborator nodes grows (e.g., for a denser sensor deployment) and the network transfer latency increases, the latency gap between *ComAI*and *CoMAI* -DEEP (which is unable to parallelize network transfer and DNN execution) will grow larger.

5) $CoMAI-DEEP$ Vs. $CoMAI-SURROGATE$: In scenarios involving surrogate nodes, approaches such as *CoMAI* -DEEP and CSSD are wasteful as the collaborator DNNs must be fully executed before the collaboration summaries are generated for sharing. In contrast, the computational complexity of $CoMAI - SURROGATE$ is minimal as the nodes only need to run the early layers (e.g., compute only 6292 parameters= 1792 (for running the DNN till depth $d = 1$)+ 4500 (shallow classifier)), thereby significantly reducing the energy overheads for surrogate sensor nodes.

[2]https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[3]lightgbm python implementation was used for Shallow classification.

VII. Discussion and Future Work

Handling Spatial Transformation Errors: The performance results presented here assume the existence of accurate spatial transformations between views (available via calibrated cameras). As a first step toward *learning* such mappings without calibration, we devised a regression based approach, where bounding box parameters ({x-mid,y-mid, height, width}) are trained for each camera pair using labels generated by object ReID techniques on the underlying training data. While accurate (adjusted $R^2 \geq 0.96$) for most camera pairs, the approach performs poorly in certain cases (e.g., height estimates with $R^2 = 0.73$). To accommodate such errors when high-fidelity spatial transformations may not be possible, *CoMAI* may be modified such that peer confidence values are *diffused* across proximate anchor boxes (instead of a strictly 1-to-1 mapping). **Autonomous Operation of *ComAI*:** Currently, *ComAI's* scene summarization technique is trained using ground truth object labels (for the 'person' class). This, however, implicitly requires human annotation, which hinders the extensibility across diverse object classes and deployments. We believe that the confidence biasing models can, however, be trained *autonomously*, on 'pseudo ground truth', based on consensus from high-confidence detections across more than one camera. **Quality and Network-Aware Execution:** Our current experimental studies utilize (collaborator, reference) camera pairs defined *a priori*. As seen in Table I, the collaboration gains are deployment and camera-pair specific. In actual bandwidth-constrained operational scenarios (especially when some nodes operate purely as surrogates), individual nodes may need to (i) characterise collaboration gains ("utility") in terms of camera deployment characteristics (e.g., percentage overlap, perspective dissimilarity) and (ii) subsequently peer with a prioritised subset of available nodes, which offer the highest collaboration gain while meeting bandwidth budget constraints.

VIII. Related Work

The topic of sensing and inference at the edge for real-time applications has received attention in recent times– e.g., optimizations of individual DNNs (compression [41], caching [19, 39, 30], model porting [13]), intelligent offloading to the edge [24, 42, 36], and orchestrating concurrent DNNs[22, 32, 23, 44]). Here, we discuss a subset of pertinent works along the following sub-domains. **Multi-View Vision DNNs:** Since recently, the availability of multi-view pedestrian datasets from the computer vision community has spurred interest in applications that can leverage multiple views such as object detection [16, 7, 3, 12]. Most recently, the MVDet model [16], extracts feature maps using CNNs from multiple views and transforms the entire maps into a common ground-plane which are then aggregated to create *pedestrian occupancy maps*. Chandravora et al. [7] start off with a pretrained, single-view model over which they architect layers to ingest multiple views, and then train a multi-layer perceptron to generate a pedestrian occupancy map (similar to Hou et al. [16]). While all these works exploit multiple views for improved pedestrian detection, (1) they require end-to-end training of specialized DNNs with fixed collaborators, (2) are not designed to minimize bandwidth/processing requirements, and (3) generate occupancy maps (which is a different task than object detection). In contrast, in our work we focus on enabling collaborative intelligence through lightweight outfitting of off-the-shelf object detectors. **Distributed/Federated Learning:** Several recent works in this domain have explored sharing local features or locally trained models to create unified models that are more effective, with particular emphasis on efficiency at the edge [17, 15]. We differ from these works in that our key goal is in adapting pre-trained models to enable collaborative intelligence in a light-weight manner. While we rely on collaborative input for each frame at runtime currently, *learning*, over time, to tune such models based on such knowledge is an important avenue for future work. **Collaborative Intelligence:** Several recent works have explored optimization techniques for a groups of networked sensors to achieve efficient querying [18, 22]. Recent works [31, 25, 20, 2] have also explored the idea of selective activation of nodes in a group of collaborating sensors – e.g., Qiu et al.[31] describe a vehicle tracking scenario where mobile nodes in a hybrid (mobile/infrastructure) camera network are activated selectively, only to resolve ambiguities. Jain et al. [20] provide preliminary examples of the possibility of using inputs from peer, overlapping cameras to utilize such spatiotemporal correlations to optimize the video analytics pipeline. The idea of collaboration among AIoT devices at the edge, and its attendant challenges, has also been mooted more generally recently in [2, 1, 28]. Most similar to our work, while Hannaneh et al. [4, 29] provide some preliminary ideas on using collaboration across camera nodes to improve the inference accuracy, we are the first to develop concrete mechanisms for low-overhead collaboration and evaluate their performance extensively over multiple datasets.

IX. Concluding Remarks

We have demonstrated the feasibility and benefits of *ComAI*, a collaborative approach for DNN-based vision inferencing that utilizes features from a peer DNN to provide significant accuracy improvements for cheaper "IoT-friendly" DNN models. *ComAI's* key innovation is the use of hidden-layer features, such as the feature maps of early convolutional layers, to predict and improve the confidence of detected object classes by a different DNN through purely runtime adaptation. Experimental results show that *ComAI*-based collaboration is model agnostic and can achieve 20-50% increase in object recall; consequently, a simple SSD object detector DNN can achieve performance similar to the complex YoLov3 model.

Acknowledgements

This work was supported by the National Research Foundation, Singapore under its NRF Investigatorship grant (NRF-NRFI05-2019-0007). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

REFERENCES

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

[2] T. Abdelzaher, S. Yao, and et al. Eugene: Towards deep intelligence as a service. In *Proc. of IEEE ICDCS'19*, 2019.

[3] Baqué, Fleuret, and Fua. Deep occlusion reasoning for multi-camera multi-target detection. In *In Proc. of ICCV'17*.

[4] H. Barahouei P and T. Nadeem. Collaborative intelligent cross-camera video analytics at edge: Opportunities and challenges. In *Proc. of AIChallengeIoT'19*, 2019.

[5] A. Bochkovskiy, C.Y. Wang, and H.Y.M. Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[6] C. Canel, T. Kim, G. Zhou, C. Li, H. Lim, D. G. Andersen, M. Kaminsky, and S. R Dulloor. Scaling video analytics on constrained edge nodes. *arXiv preprint arXiv:1905.13536*, 2019.

[7] Chavdarova and Fleuret. Deep multi-camera people detection. In *ICMLA'17*.

[8] T. Chavdarova, P. Baqué, S. Bouquet, A. Maksai, C. Jose, T. Bagautdinov, L. Lettry, P. Fua, L. Van Gool, and F. Fleuret. Wildtrack: A multi-camera hd dataset for dense unscripted pedestrian detection. In *Proc. of IEEE CVPR'18*, June 2018.

[9] B. Fang, X. Zeng, F. Zhang, H. Xu, and M. Zhang. Flexdnn: Input-adaptive on-device deep learning for efficient mobile vision. In *SEC'20*.

[10] J. Ferryman and A. Shahrokni. Pets2009: Dataset and challenge. In *2009 Twelfth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pages 1–6, Dec 2009.

[11] J. Ferryman and A. Shahrokni. Pets2009: Dataset and challenge. In *Proc. of VS-PETS'09*. IEEE, 2009.

[12] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. Multicamera people tracking with a probabilistic occupancy map. *IEEE PAMI*, 30(2):267–282, 2007.

[13] Peizhen Guo, Bo Hu, and Wenjun Hu. Mistify: Automating dnn model porting for on-device inference at the edge. In *NSDI*, pages 705–719, 2021.

[14] Song H., Huizi M., and J. D. William. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *Proc. of ICLR'16*, 2016.

[15] Dong-Jun Han, Jy-yong Sohn, and Jaekyun Moon. Tibroco: A fast and secure distributed learning framework for tiered wireless edge networks. In *IEEE INFOCOM'21*.

[16] Yunzhong Hou, Liang Zheng, and Stephen Gould. Multiview detection with feature perspective transformation. In *European Conference on Computer Vision*, pages 1–18. Springer, 2020.

[17] Y. Hu, D. Niu, J. Yang, and S. Zhou. Fdml: A collaborative machine learning framework for distributed features. In *SIGKDD'19*.

[18] C. Hung, G. Ananthanarayanan, P. Bodik, L. Golubchik, M. Yu, P. Bahl, and M. Philipose. Videoedge: Processing camera streams using hierarchical clusters. In *IEEE SEC'18*.

[19] L. N. Huynh, Y. Lee, and R. Balan. Deepmon: Mobile gpu-based deep learning framework for continuous vision applications. In *Proc. of ACM MobiSys'17*.

[20] S. Jain, G. Ananthanarayanan, J. Jiang, Y. Shu, and J. Gonzalez. Scaling video analytics systems to large camera deployments. In *Proc. of ACM HotMobile*, 2019.

[21] S. Jain, X. Zhang, Y. Zhou, G. Ananthanarayanan, Jiang J., Shu Y., Bahl P., and J. Gonzalez. Spatula: Efficient cross-camera video analytics on large camera networks. In *SEC'20*.

[22] J Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica. Chameleon: Scalable adaptation of video analytics. In *SIGCOMM'18*.

[23] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia. Noscope: Optimizing neural network queries over video at scale. *Proc. VLDB Endow.*, 10(11), August 2017.

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

[25] J. Lee and al et. On tracking realistic targets in a megacity with contested air and spectrum access. *MILCOM'18*.

[26] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. NIPS'18, 2018.

[27] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, CY. Fu, and A. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*. Springer, 2016.

[28] A. Misra, K. Jayarajah, D. Weerakoon, R. Tandriansyah, S. Yao, and T. Abdelzaher. Dependable machine intelligence at the tactical edge. In *SPIE'19*.

[29] H. B. Pasandi and T. Nadeem. Convince: Collaborative cross-camera video analytics at the edge. In *PerCom Workshops'20*.

[30] G Peizhen and H. Wenjun. Potluck: Cross-application approximate deduplication for computation-intensive mobile applications. In *ASPLOS'18*.

[31] H. Qiu, X. Liu, S. Rallapalli, A. J. Bency, K. Chan, R. Urgaonkar, BS. Manjunath, and R. Govindan. Kestrel: Video analytics for augmented multi-camera vehicle tracking. In *Proc. of IEEE/ACM IoTDI'18*. IEEE, 2018.

[32] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen. Deepdecision: A mobile deep learning framework for edge video analytics. In *Proc. of IEEE INFOCOM'18*. IEEE, 2018.

[33] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *Proc. of ICASSP'13*, 2013.

[34] J. Uijlings, K. Van De Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[35] J. Wang, Z. Feng, Z. Chen, S. George, M. Bala, P. Pillai, S. Yang, and M. Satyanarayanan. Bandwidth-efficient live video analytics for drones via edge computing. In *IEEE/ACM SEC'18*.

[36] X. Wang and et al. EdgeDuet: Tiling Small Object Detection for Edge Assisted Autonomous Mobile Vision. In *In Proc. of IEEE INFOCOM*, 2021.

[37] D. Weerakoon, K. Jayarajah, R. Tandriansyah, and A. Misra. Resilient collaborative intelligence for adversarialiot environments. In *Proc. of FUSION'19*, 2019.

[38] M. Xu, F. Qian, Q. Mei, K. Huang, and X. Liu. Deeptype: On-device deep learning for input personalization service with minimal privacy concern. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2(4), December 2018.

[39] M. Xu, M. Zhu, Y. Liu, F. X. Lin, and X. Liu. Deepcache: Principled cache for mobile deep vision. In *MobiCom '18*.

[40] Y. Xu, X. Liu, L. Qin, and SC. Zhu. Cross-view people tracking by scene-centered spatio-temporal parsing. In *AAAI*, 2017.

[41] S. Yao and al et. Fastdeepiot: Towards understanding and optimizing neural network execution time on mobile and embedded devices. In *Proc. of ACM SenSys'18*.

[42] S. Yao and et al. Deep compressive offloading: Speeding up neural network inference by trading edge computation for network latency. In *SenSys '20*.

[43] X. Zeng, B. Fang, H. Shen, and M. Zhang. Distream: scaling live video analytics with workload-adaptive distributed edge intelligence. In *SenSys'20*.

[44] 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 *MobiCom '21*.