Context-aware Multi-Model Object Detection for Diversely Heterogeneous Compute Systems

Justin Davis
Computer Science
Colorado School of Mines
jcdavis@mines.edu

Mehmet E. Belviranli
Computer Science
Colorado School of Mines
belviranli@mines.edu

***Abstract*—In recent years, deep neural networks (DNNs) have gained widespread adoption for continuous mobile object detection (OD) tasks, particularly in autonomous systems. However, a prevalent issue in their deployment is the one-size-fits-all approach, where a single DNN is used, resulting in inefficient utilization of computational resources. This inefficiency is particularly detrimental in energy-constrained systems, as it degrades overall system efficiency. We identify that, the contextual information embedded in the input data stream (*e.g.*, the frames in the camera feed that the OD models are run on) could be exploited to allow a more efficient multi-model-based OD process. In this paper, we propose *SHIFT* which continuously selects from a variety of DNN-based OD models depending on the dynamically changing contextual information and computational constraints. During this selection, *SHIFT* uniquely considers multi-accelerator execution to better optimize the energy-efficiency while satisfying the latency constraints. Our proposed methodology results in improvements of up to 7.5x in energy usage and 2.8x in latency compared to state-of-the-art GPU-based single model OD approaches.**

***Index Terms*—accelerator, autonomous, context-aware, object detection, gpu, heterogeneous**

I. Introduction

Modern autonomous systems employ deep neural networks (DNN) for various tasks and all-in-one system-on-chips (SoC) to enable decision making on-the-go without human intervention. Typically, such autonomous systems are equipped with SoCs featuring graphical processing units (GPU) for the execution of DNNs. A common and critical autonomous task is object detection (OD) which identifies objects of interest in the environment, captured by the stream of images obtained by the camera. A common practice employed by system developers is to select and configure a single DNN, such as YoloV7 [1], and map it to the fastest processor in the SoC, which is typically a GPU. In this conventional setup, there is limited room for improving the latency and/or energy usage of the autonomous system, as the model and the target processing unit is fixed. In response, several studies [2], [3] propose offloading the computation to a remote server, while others [4]–[6] attempt to reduce the computational demand by modifying the underlying model or using a subset of the data stream. However, offloading is not a viable option due to the latency overhead associated with remote processing. On the other hand, modifying models or selectively skipping data often results in a significant compromise in accuracy. Instead, in this work, we explore optimizing the system performance by employing a context-aware multi-model execution and leveraging different type of accelerators available in SoCs.

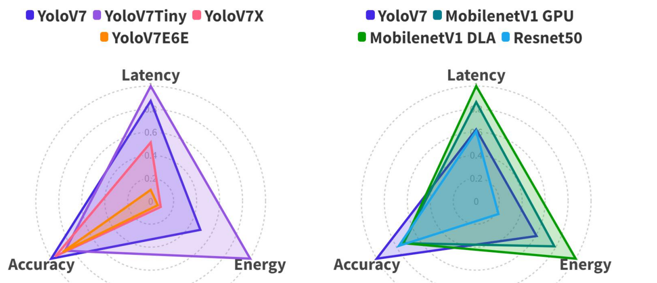

Fig. 1: Comparison of (a) single-model with multiple parameter sizes on the left against (b) multi-model object detection architectures on the right. The larger the value along each axis the better: a perfect model would be largest triangle across all axes.

Modern SoCs often embed neural network (NN) accelerators alongside with GPUs to perform low-power DNN inference. For example, the Nvidia DLA in the Jetson Xavier series allows DNN inference with up to 2.5x energy savings compared to the GPU in the same SoC, at the cost of 2x slower latency. Another accelerator, the RCV2 on the OAK-D from Luxonis, provides 4 TOPs of computation at 5 Watts, surpassing GPU energy efficiency. Having different types of DNN accelerators in the system enables a tunable trade-off between latency and energy [7].

A common approach to save energy and reduce computational demands in the OD process is to quantize a given object detection model (ODM) and create less accurate but more energy and performance efficient versions of the same model [8]. While this method allows for the execution of state-of-the-art models on performance-limited SoCs, efficiency-related optimizations are constrained due to the entire OD process being confined to a single DNN model. A more comprehensive and

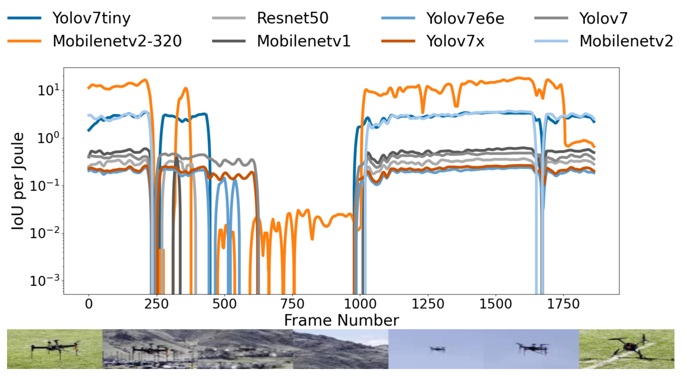

Fig. 2: Single model object detection efficiency on GPU for commonly used DNNs and their variations on a test set for continuous detection and tracking of an aerial drone. Efficiency is quantified by intersection over union (IoU) per Joule of energy (see Sec. IV for details).

Model	IoU	Inference (s)			Power (W)			Energy (J)		
		CPU	GPU	DLA	CPU	GPU	DLA	CPU	GPU	DLA
YoloV7	0.62	1.65	0.13	0.12	7.60	15.1	15.1	20.5	1.97	1.78
YoloV7Tiny	0.53	0.38	0.03	0.02	7.20	11.2	11.2	4.19	0.28	0.27
MobilenetV1	0.45	-	0.09	0.09	-	16.2	6.10	-	1.52	0.56

TABLE I: Average statistics for two architectures of object detection models and their performance on CPU, GPU, and GPU/DLA.

flexible way to further increase energy savings is to dynamically switch to simpler, less compute demanding ODMs if their detection accuracy is sufficient for the changing (*i.e.*, *contextual*) characteristics of the captured frames. For example, if the target object being detected is in front of a solid, contrasted background within a close distance, then both simple and advanced models perform equally well. Fig. 2 demonstrates this *contextual* change by depicting the timeline of varying accuracies that different ODMs exhibit on one of our test sets (see Sec. V for details). Through optimal utilization of all available heterogeneous models, we observed enhancements in accuracy by 3%, latency by 5.2x, and energy usage by 13.6x, or a combination thereof. We illustrate the three-way energy-accuracy-latency (*e-a-l*) relationship between the standard YOLOv7 model [1] and other simpler model families in Fig. 1. While smaller variations of YOLOv7 (Fig. 1.a) result in a monotonic decrease of energy and latency; the use of different type of ODMs (Fig. 1.b) exhibits a non-monotonically changing relationship between the three metrics. Detailed statistics on the *e-a-l* trade-off for different accelerators and the CPU could be found in Table I.

By leveraging the optimization potential offered by multi-accelerator systems and multi-model object detection, there exists an opportunity to enhance accuracy, reduce latency, and conserve energy in real-time operations within autonomous systems. However, creating such an OD scheme presents several challenges: (i) Models need to be characterized in advance to determine their performance and energy characteristics. (ii) Since the accuracy of each model depends on the dynamic context, predicting accuracies without running them for every encountered frame is not trivial. (iii) Not all models considered by the system can be simultaneously loaded into memory due to limitations in available resources.

In this paper, we present *SHIFT* to enable energy and latency efficient multi-model and multi-accelerator object detection for autonomous systems. This approach takes into consideration the changing contextual features of the input frames, accelerators in the system, and ODMs with different execution characteristics. Our contributions are as follows:

- We build a unique graph-based mechanism, named confidence graphs, that is used to rapidly predict the changing accuracy of different ODMs during runtime.
- We create a novel scheduler that can adapt to specific system constraints by targeting model accuracy, latency, or energy consumption, based on real-time contextual information derived from the input stream of frames. This scheduler decides the ODM and accelerator to run such that system objectives are achieved under given constraints.
- We deploy a dynamic model loading mechanism that is capable of managing memory resources for each ODM and facilitates switching between ODMs when necessary.
- We evaluate the utility and efficiency of *SHIFT* on three unique off-the-shelf accelerators designed for autonomous systems. We show that our proposed methodology results in improvements of up to **7.5x** in energy usage and **2.8x** in latency compared to GPU-based single model OD approaches.

Feature \ Related Work	Glimpse [2]	MARLIN [5]	AdaVP [4]	RoaD-RuNNer [9]	Fast UQ [10]	Herald [11]	AxoNN [7]	***SHIFT***
Context Awareness	✗	✓	✓	✓	✗	✗	✗	✓
Multi-Accelerator	✗	✗	✗	✗	✗	✓	✓	✓
Multi-DNN	✗	✗	✗	✗	✓	✗	✗	✓
Energy-Aware	✗	✓	✓	✓	✗	✓	✓	✓
No-Offloading	✗	✓	✓	✗	✓	✓	✓	✓
Continuous	✓	✓	✗	✓	✗	✗	✗	✓

TABLE II: Comparison of the features offered by related works.

II. Related Work

The related work could be categorized as follows:

Continous Detection: Glimpse [2], RoaD-RuNNer [9], and FlexPatch [3] propose using an edge-server computational setup to decrease latency and reduce energy by using techniques such as edge level object tracking, edge/server model partitioning and selective tracking, respecitvely. However, such approaches rely on stable connections to servers, and none of them consider the use of multiple accelerators or multiple DNN models. Marlin [5], and AvaVP [4] are studies which aim to reduce the energy usage onboard a mobile device doing OD. Marlin [5] proposes an approach where, instead of running the DNN every frame, the system alternates between a tracking algorithm and DNN. AvaVP [4] extends Marlin [5] by varying the input size of the DNN and skipping frames to adjust the *e-a-l* trade-off during runtime. Neither considers the benefit of employing multiple DNN models or utilizing accelerators other than a GPU.

DNN Inference on Multi-Accelerator Systems: Inference on multi-accelerator systems is an active research area with several recent studies being published [7], [11]–[16]. Herald [11] and AxoNN [7] generate optimized schedules for performing inference with a single DNN on a system using a layer-by-layer mapping scheme. NeuLens [12] and Band [13] reduce DNN inference time by splitting a DNN into subgraphs for processing. Deepmon [14] and CoDL [15] optimize latency on mobile execution by scheduling layers on both CPU and GPU. While these solutions consider latency and energy constraints, neither of them considers using multiple, different types of DNN models for OD in a context-aware manner.

Multi Model Detection: Fast UQ [10] uses different types of DNN architectures to extract more accurate poses in 3D space. They identify that combining multiple DNNs with domain specific metrics could lead to significant accuracy improvements.

A feature comparison between the most relevant related work and our approach can be found in Table II. *SHIFT* is uniquely able to handle context-aware, continuous multi-model (*i.e.*, multi-DNN) OD across multiple accelerators while satisfying energy, latency and accuracy constraints.

III. Methodology

SHIFT is composed of three primary components: (a) We first *characterize* the target set of ODMs by determining their core traits for each available accelerator and build a *confidence graph* to enable fast accuracy prediction at runtime. (b) Then, *our multi-model, multi-accelerator scheduler* is responsible for determining the current best ODM to run, for each incoming frame. The scheduler uses the traits identified in the characterization process and the confidence graph, and selects the model for the next frame that most effectively meets accuracy, energy and latency constraints. (c) Finally, our scheduler employs a *dynamic model loader* that utilizes characterization data and runtime memory footprint measurements to determine where to allocate models and, if necessary, which models to deallocate.

A. *Object Detection Model Characterization*

The latency, energy consumption, and detection accuracy of OD in an autonomous system depend on traits varying by the ODM, frame-context, and target accelerators. To select the best model for each frame encountered at runtime, we need to continously predict the accuracy of the ODMs at hand. However, this is not trivial unless each ODM is run everytime the frame-context changes, which is a very costly approach. To address this problem, we build a two-step approach where we first (1) perform an offline characterization of ODMs into a discrete set of traits and then, (2) using these traits, we construct a confidence graph which will later be used by the runtime to predict the accuracy of a model. Notably, our approach for model characterization and graph construction is generic in nature. It relies solely on a testing or validation subset of the dataset used for training the models.

ODM Trait Identification: For each ODM, we collect the following traits: **(i) Accuracy** is characterized by running the ODM on the testing data set and, for each frame, computing the intersection over union (IoU) between the bounding boxes of the detected object and labeled ground truth. **(ii) Confidence score** represents an internal accuracy assessment of the underlying DNN and can be used for quickly acquiring a performance estimate. However, these scores can be influenced by over-fitting and sometimes they are 'over-confident'; therefore, they are not consistent across different ODM architectures. **(iii) Latency** of an ODM is found by measuring the execution time of the model on each target accelerator. **(iv) Energy** is characterized by measuring the $time \times power\_draw$ across all power rails during execution with a given model. **(v) Model loading cost** needs to be accounted for as part of multi-model execution overhead. This cost includes the memory footprint, time to load the model, and energy draw during this time.

Confidence Graph Creation: For a given frame, the confidence scores reported from multiple types of ODMs vary, while versions of the same ODM produce similar scores. Correlating the scores between different types is essential for accurate predictions, since we cannot run each model type on each frame due to energy, latency, and memory constraints.

To quickly predict the accuracy of ODMs on the fly, we construct a novel *confidence graph* (CG) as follows:

1) Each node represents a discrete confidence score range of an ODM and its expected accuracy (*e.g.*, we create a node, `YoloV7-(0.5-0.6)`, to represent YoloV7's performance in the confidence score range of 0.5 to 0.6).

2) To find edges, we run each ODM on a testing/validation dataset. For each image in the dataset, we locate the nodes for each ODM's confidence score range on that image and create edges between them. For example, for a given image, if YoloV7 resulted in a conf. score of 0.53 and MobileNet results in 0.42, we create an edge between the two corresponding nodes: `YoloV7-(0.5-0.6)` and `MobileNet-(0.4-0.5)`. If an edge already exists, we increment its weight by 1.

3) We then normalize all edges to have a weight w between 0 and 1 and invert weights so that two highly connected nodes will have a lower cost when traversed. The normalization is performed within the edges directly connecting a single node, such that global maximums will not take over.

4) Next we run a breadth first search starting at all nodes in the graph, to get the set of neighbor nodes which have a distance less than or equal to a given distance threshold.

5) Since a given set of neighbors can contain multiple nodes representing the same model, said nodes are consolidated by taking a weighted average of each node's expected accuracy by the distance to traverse to said node.

6) We then store all results from the CG in a map where each node is a key to the accuracy predictions of its neighbors.

In summary, our proposed CG structure generates a map that transforms the confidence score of a single ODM into accuracy predictions for all ODMs. Instead of relying on costly classifiers, an ensemble, or less expensive predictors employed by similar works [4], [5], we can execute a map lookup at runtime.

B. *SHIFT Scheduler*

The *SHIFT* scheduler is designed to perform the decision-making process at runtime, leveraging contextual information from both model runtime behavior and the input data stream (*i.e.*, frames). Our scheduler can make fast decisions with minimal computational overhead by utilizing the *CG* and the analysis of frames using computationally efficient metrics. As a result, the scheduler maintains an overhead of less than 2 milliseconds per frame. The scheduler operation can be delineated into two fundamental components: context detection, and a heuristic algorithm for selecting appropriate models.

Context Detection: *SHIFT* scheduler relies on input frames to detect the changes in the context, so that the proper ODM could be employed. While the *CG* lets us rapidly predict the accuracy of an ODM by deriving information from confidence scores reported by DNNs, these scores are intrinsically linked to model error [17] and may not be reliable when the input data is further outside the scope of the training data than the testing set. The frame context also plays a crucial role in predicting the accuracy of an ODM [18]. Extracting a comprehensive range of context from these images is computationally expensive and not viable for real-time processing on edge devices. Instead, the *SHIFT* scheduler assesses frame similarity using the normalized cross-correlation (NCC) between consecutive bounding box results and image frames:

$$\text{NCC}(p, c) = \frac{\sum (p - mean(p))(c - mean(c))}{(\sqrt{\sum (c - mean(c))^2} \times \sqrt{\sum (p - mean(p))^2}} \quad (1)$$

where p and c are grayscale images of the same size representing the previous and current frames in an input stream. By employing NCC, the scheduler can identify when the input stream has changed significantly, prompting re-scheduling of the current ODM. This can aid in identifying when the current ODM may incorrectly continue to report high confidence scores despite objects not being present or easily detectable.

Scheduling Heuristic: Algorithm 1 describes the *SHIFT* scheduler which utilizes a heuristic to assign weights to schedulable models. The scheduler takes the current ODM m, conf. score c, input frame i and bounding box b as input and returns the highest scoring model as output. The energy and latency characteristics of available models are pre-determined, normalized to a 0 to 1 range, and inverted for bigger-is-better performance indication (lines 6, 7). At runtime, the scheduler invokes the confidence graph to estimate model accuracies for the most recent frame (line 9). The scheduler averages accuracy predictions for all models and aggregates those meeting the desired accuracy threshold (lines 11-15). In the absence of models meeting the threshold, all available models are considered valid (lines 16-17). Post model selection, weights are assigned based on user-defined parameters, and the best model is outputted (lines 19-24). As energy and latency are converted to bigger-is-better metrics, maximum search of candidate models suffices for optimal selection.

The image similarity score is computed as the minimum of the NCC between the last two images and the NCC across the last two bounding box detections. This score determines whether the scheduler should initiate the selection of a new model/accelerator pair. The metric is then multiplied by the current model confidence to facilitate scheduling during periods when the existing ODM may exhibit unstable detections. By reserving the scheduling of new models for instances characterized by rapid changes in the overall image context, bounding box, or a reduction in model confidence, the overall cost incurred by model swapping can be minimized.

C. *Dynamic Model Loader*

Every model which can be executed has profiling information available about the process of loading the model into memory. When there is a scheduling decision and a new model is requested to be loaded into memory, the dynamic model loader (DML) will query the system's available memory. The DML will attempt to occupy the entire memory with ODMs, if it is able to. This reduces the costly switching between ODMs and improves the performance by enabling quicker model swapping. The DML is able to differentiate between accelerators and will allocate to them separately. When replacing models the DML will replace the model which was least recently requested. Since accelerators do not all share the same memory, the DML needs to have the knowledge about whether an accelerator can execute a specific ODM.

IV. Experimental Setup

Hardware and Accelerators: We performed our experiments on Nvidia Xavier NX SoC, a commonly used platform for aerial autonomous vehicles, and a Luxonis OAK-D Lite, a stereo camera with DNN execution capability. The platform includes a CPU, GPU, 2 DLAs, and an OAK-D for DNN execution. Due to model layer incompatibility, limitations on model size, and support constraints in libraries, the DLA and OAK-D do not support some layers and models we use in our experiments.

Algorithm 1 Model Scheduling

```
 1: procedure SHIFT SCHEDULE($m, c, i, b$)
 2:     $s = \min(\text{NCC}(\text{lastImage}, i), \text{NCC}(\text{lastBbox}, b))$
 3:     if $s \times c \geq accuracyThreshold$ then
 4:         return m
 5:     end if
 6:     $E$ = scheduler.energy                ▷ $0 \to 1$ model energy
 7:     $L$ = scheduler.latency               ▷ $0 \to 1$ model latency
 8:     $W$ = scheduler.weights               ▷ Tuned knobs
 9:     $C$ = graphPredict($m, c$)            ▷ set of (name, acc, dist)
10:     $R, scores$ = map(), map()
11:     for $(n, a, d) \in C$ do
12:         a.Buffer.$append(a)$
13:         $R[n]$ = average(a.Buffer)
14:     end for
15:     $V = \{ n \mid n \in R, n \geq accuracyThreshold \}$
16:     if length($V$) == 0 then
17:         $V = R$
18:     end if
19:     for $n \in R$.keys() do
20:         $s = R[n] * W[0] + E[n] * W[1] + L[n] * W[2]$
21:         $scores[n] = s$
22:     end for
23:     return max($scores$)
24: end procedure
```

Dataset: We train the ODMs with a dataset focused on the detection of unmanned autonomous vehicles (UAVs) [19], which comprises a training set of 50,000 images and a validation set of 2,500 images. Each image contains at most a single UAV, hence all implementation and evaluation are within the context of a single-class, single-object detection problem.

Model Training: YoloV7 [1] based models were trained using the training scripts and pipelines provided by the authors. All YoloV7 models use an IoU threshold of 0.5 and a confidence threshold of 0.35 in non-maximum-suppression. MobileNetV1, MobileNetV2, and Resnet50 were trained with the single-shot detector methodology. These models were trained using the Tensorflow object detection API. All models use an input size of 640x640 unless otherwise stated.

Method Evaluation & Comparison: Comprehensive assessments were conducted across various standalone ODMs, considering their compatibility with distinct accelerators. Our analysis includes a comparative analysis with Marlin [5], recognized for its energy-efficient executions on mobile devices.

To test all methods' ability to handle real-life challenges we created a custom evaluation dataset consisting of six scenarios as a set of videos, each comprising between 500 to 2,500 frames. The videos have two indoor and four outdoor scenarios where the targeted UAV is at different distances from the camera with varying backgrounds and positioning in the frame.

All models on the GPU/DLA were executed using TensorRT. OpenVINO was used to compile models for the OAK-D. All GPU layers for models were executed in FP32 due to severe accuracy degradation during quantization with TensorRT for

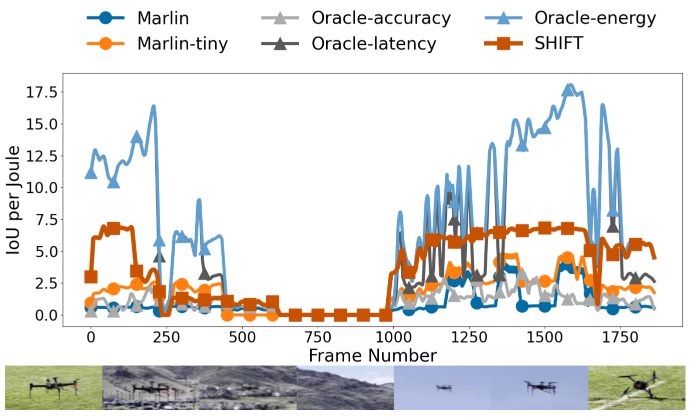

Fig. 3: Scenario 1: Drone navigates across multiple backgrounds at *varying* distances from the camera.

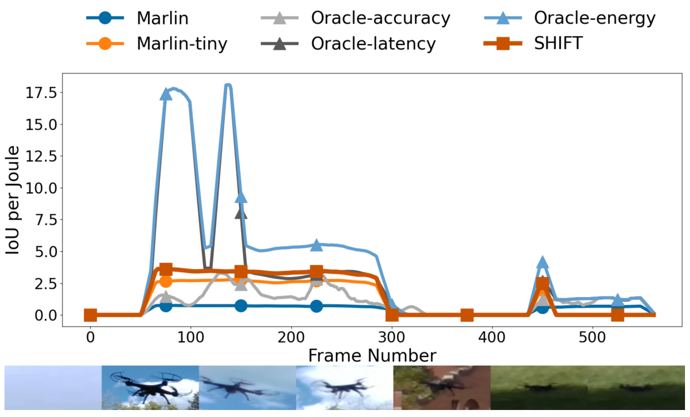

Fig. 4: Scenario 2: Drone navigates across multiple backgrounds at a *fixed* distance.

YoloV7 models. To establish a performance ceiling, an Oracle methodology was created. This Oracle identifies all models surpassing a 0.5 intersection-over-union (IoU) threshold, subsequently selecting the one that optimizes the targeted metric. In cases where no models meet the IoU criterion, selection is solely based on metric optimization. Since the Oracle methods represents a maximum performance, it assumes that all models are loaded into memory and thus have no cost to switch. The choice of a 0.5 IoU threshold aligns with the common practice of using a 0.5 threshold as the minimum when evaluating object detection models [1]. Additionally, we define the metric *success rate* as the percentage of frames which have an IoU $\geq$ 0.5. Since the testing videos contain a single UAV, the average IoU effectively captures all relevant accuracy information.

V. Evaluation

A. Main results

The detailed traits from the characterization of all ODMs we employ are given in Table IV. Table III presents the overall results of our experiments, by giving a comparison between average accuracy, latency and energy obtained by Marlin [5], *SHIFT* and Oracle executions. Additionally, Figures 3 and 4 illustrate the accompanying timelines and frame context changes for two of the videos contributing to Table III.

Overall averages given in Table III show that *SHIFT* consistently achieves a success rate surpassing all single-model executions except the top-performing YoloV7 model. *Uniquely, SHIFT manages to uphold energy and latency efficiency superior to Marlin and all single-model runs while consistently maintaining desired accuracy ranges.* Notably, *SHIFT* outperforms the state-of-the-art method Marlin irrespective of the underlying ODM which Marlin utilizes.

In the first scenario, which is given in Figure 3, the UAV executes maneuvers across intricate backgrounds distant from the camera before returning. A proficient dynamic system must adeptly identify crucial context changes, as evidenced by sharp fluctuations in model efficiency values. *SHIFT* successfully recognizes all context changes, implementing transitions to more resource-intensive ODMs at frame markers $\sim$ 500 and $\sim$ 1100. Simultaneously, it strategically conserves resources by transitioning at frames $\sim$ 50 and $\sim$ 1650. The unique capability of *SHIFT* to augment resource usage during challenging or simple inputs contributes to its superiority over solutions like Marlin. Importantly, *SHIFT* can conservatively allocate resources during periods without valid detections.

In the second scenario given in Figure 4, the UAV moves horizontally across simpler backgrounds while gradually moving across the camera's perspective. Analogous to the first scenario, we observe abrupt drops in ODM accuracy, and thus efficiency, when the UAV is between backgrounds. Notably, *SHIFT* successfully identifies when the UAV enters the camera view, prompting model swaps to enhance efficiency. The discernable delay in *SHIFT*'s response compared to Marlin and the Oracles is attributed to its reactionary model swapping. It is noted that *SHIFT* did not detect the UAV beyond frame $\sim$ 450 due to confidence scores of the current ODM indicating no UAV.

From Table III, *SHIFT* utilizes fewer ODM-accelerator pairs than the Oracle methods while still maintaining higher efficiency than previous state-of-the-art Marlin [5]. Despite higher utilization of heterogeneous resources compared to the Oracles, *SHIFT* loads less models into memory and uses fewer swaps between ODMs or accelerators. For scenario 1 (pictured in Figure 3), *SHIFT* only loaded ODMs which were smaller than YoloV7; hence showcasing *SHIFT*'s preference for efficient inference via the tunable weights.

Overall, we observe that *SHIFT* is able to successfully optimize across energy and latency while swapping ODMs during runtime based on contextual information from the input stream in all real-world scenarios evaluated.

Methodology	IoU	Time (s)	Energy (J)	Success Rate	Non-GPU	Model Swaps	Pairs Used
Marlin	0.614	0.132	1.201	74.0%	0%	0	1
Marlin Tiny	0.529	0.036	0.33	64.0%	0%	0	1
SHIFT	0.598	0.047	0.262	72.2%	68.7%	42	4.3
Oracle E	0.535	0.025	0.144	76.0%	31.5%	94	6.7
Oracle A	0.657	0.108	1.423	76.0%	44.9%	409	12.3
Oracle L	0.522	0.025	0.169	76.0%	11.3%	112	6.8

TABLE III: Average runtime performance of continuous object detection with *SHIFT*. *SHIFT* parameters: goal accuracy 0.25, momentum 30, distance threshold 0.5, knobs: accuracy 1.0, and energy/latency 0.5. Goal accuracy reduced from 0.5, based on observation from Figure 5. Pairs are model accelerator pairs, a total of 18 combinations were possible. Includes overhead for *SHIFT* and Marlin methods.

Model Name	Accuracy		Avg. Time (s)			Avg. Energy (Joules)			Avg. Power Draw (W)		
	Avg. IoU	Success Rate	GPU	GPU/DLA	OAK-D	GPU	GPU/DLA	OAK-D	GPU	GPU/DLA	OAK-D
YoloV7-E6E	0.564	65.8%	0.255	0.221	-	3.947	1.228	-	15.48	5.56	-
YoloV7-X	0.593	71.1%	0.222	0.195	-	3.586	1.088	-	16.15	5.57	-
YoloV7	0.618	74.1%	0.130	0.118	0.894	1.968	0.656	1.391	15.14	5.56	1.56
YoloV7-Tiny	0.533	64.0%	0.025	0.024	0.107	0.280	0.134	0.206	11.2	5.58	1.93
SSD Resnet50	0.480	58.9%	0.151	0.138	-	2.504	0.816	-	16.58	5.91	-
SSD MobilenetV1	0.452	55.4%	0.094	0.092	-	1.519	0.561	-	16.16	6.10	-
SSD MobilenetV2	0.401	51.3%	0.023	0.058	-	0.248	0.307	-	10.78	5.29	-
SSD MobilenetV2 320x320	0.304	36.2%	0.009	0.023	-	0.046	0.100	-	5.11	4.35	-

TABLE IV: Collected accuracy and performance traits of all models.

B. Sensitivity analysis

We performed a sensitivity analysis on *SHIFT* to ascertain the robustness of system performance against variations in input parameters. A total of 1860 parameter configurations were tested. The outcomes of this analysis are depicted in Figure 5. The analysis indicates that the system's performance conforms to expectations with respect to all knob parameters. By increasing the value of the energy or latency knob, we observe a negative correlation with the actual ODM's energy and latency as expected. Conversely, the accuracy knob has a positive correlation since more expensive ODMs are more accurate. The accuracy threshold parameter inversely affects all primary metrics; this is because when *SHIFT* fails to find any models meeting the goal accuracy it defaults to optimization based on the knob settings alone. We observe that ODM accuracy is underestimated and lowering the accuracy goal for runtime improves efficiency. The momentum parameter, indicative of the number of frames over which to average the predicted accuracy of a given ODM, exhibits minor correlation with the performance metrics, suggesting that frame-to-frame results are inherently stable. Additionally, the distance threshold parameter has a distinct correlation with reducing the average ODM latency, due to more ODMs being considered at runtime.

VI. Conclusion

We introduce *SHIFT*, capable of dynamically switching between heterogeneous DNNs and target hardware based on continuous input stream context. *SHIFT* facilitates multi-model swaps for accuracy, energy, and latency trade-offs, leveraging multiple accelerators. Experimental results demonstrate *SHIFT*'s effectiveness, showing significant improvements. Compared to a state-of-the-art ODM on GPU, *SHIFT* achieves up to a **2.8x** reduction in latency and a **7.5x** decrease in energy consumption, with only a modest 0.97x reduction in successful frames and 0.97x reduction in average IoU. Notably, *SHIFT* maintains performance without inter-frame object tracking or skipping input frames.

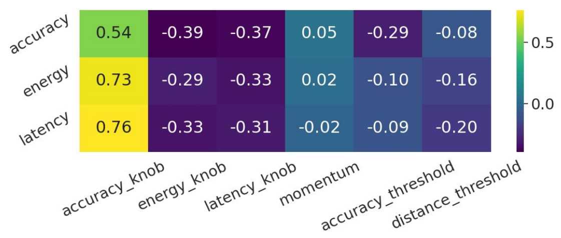

Fig. 5: Sensitivity analysis of the *SHIFT* parameters against the mean accuracy, energy, and latency values.

Acknowledgements

This material is based upon work supported by the National Science Foundation (NSF) under Grants No. CCF-2124010 and CHE-2235143. Any opinions, findings, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

References

[1] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *CVPR'23*.

[2] T. Y.-H. Chen, L. Ravindranath, S. Deng, and et al., "Glimpse: Continuous, real-time object recognition on mobile devices," in *SenSys'15*.

[3] K. Yang, J. Yi, K. Lee, and Y. Lee, "Flexpatch: Fast and accurate object detection for on-device high-resolution live video analytics," in *INFOCOM'22*.

[4] M. Liu, X. Ding, and W. Du, "Continuous, real-time object detection on mobile devices without offloading," in *ICDCS'20*.

[5] K. Apicharttrisorn, X. Ran, J. Chen, and et al., "Frugal following: Power thrifty object detection and tracking for mobile augmented reality," in *SenSys'19*.

[6] M. Adnan Arefeen, S. Tabassum Nimi, and M. Yusuf Sarwar Uddin, "Framehopper: Selective processing of video frames in detection-driven real-time video analytics," in *DCOSS'22*.

[7] I. Dagli, A. Cieslewicz, J. McClurg, and M. E. Belviranli, "Axonn: Energy-aware execution of neural network inference on multi-accelerator heterogeneous socs," in *DAC'22*.

[8] A. Gholami, S. Kim, Z. Dong, and et al., "A survey of quantization methods for efficient neural network inference," *Low-Power Computer Vision*, 2021.

[9] A. K. Kakolyris, M. Katsaragakis, D. Masouros, and D. Soudris, "Roadrunner: Collaborative dnn partitioning and offloading on heterogeneous edge systems," in *DATE'23*.

[10] G. Shi, Y. Zhu, J. Tremblay, and et al., "Fast uncertainty quantification for deep object pose estimation," in *ICRA'21*.

[11] H. Kwon, L. Lai, M. Pellauer, and et al., "Heterogeneous dataflow accelerators for multi-dnn workloads," in *HPCA'21*.

[12] X. Hou, Y. Guan, and T. Han, "Neulens: Spatial-based dynamic acceleration of convolutional neural networks on edge," in *MobiCom '22*.

[13] J. S. Jeong, J. Lee, D. Kim, C. Jeon, C. Jeong, Y. Lee, and B.-G. Chun, "Band: Coordinated multi-dnn inference on heterogeneous mobile processors," in *MobiSys'22*.

[14] L. N. Huynh, Y. Lee, and R. K. Balan, "Deepmon: Mobile gpu-based deep learning framework for continuous vision applications," in *MobiSys'17*.

[15] F. Jia, D. Zhang, T. Cao, S. Jiang, Y. Liu, J. Ren, and Y. Zhang, "Codl: Efficient cpu-gpu co-execution for deep learning inference on mobile devices," in *MobiSys '22*.

[16] B. Fang, X. Zeng, and M. Zhang, "Nestdnn: Resource-aware multi-tenant on-device deep learning for continuous mobile vision," in *MobiCom'18*.

[17] A. Loquercio, M. Segu, and D. Scaramuzza, "A general framework for uncertainty estimation in deep learning," *IEEE Robotics and Automation Letters*, 2020.

[18] M. Raghu, K. Blumer, R. Sayres, Z. Obermeyer, R. Kleinberg, S. Mullainathan, and J. Kleinberg, "Direct uncertainty prediction for medical second opinions," in *ICML'19*.

[19] M. L. Pawelczyk and M. Wojtyra, "Real world object detection dataset for quadcopter unmanned aerial vehicle detection," *IEEE Access*, 2020.