11-2024

CAS: Fusing DNN optimization & adaptive sensing for energy-efficient multi-modal inference

Dulanga WEERAKOON
Singapore-MIT Alliance for Research & Technology

Vigneshwaran SUBBARAJU
Institute of High Performance Computing

Joo Hwee LIM
InfoComm Research

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

CAS: Fusing DNN Optimization & Adaptive Sensing for Energy-Efficient Multi-Modal Inference

Dulanga Weerakoon[1], Vigneshwaran Subbaraju[2], Joo Hwee Lim[3] and Archan Misra[4]

***Abstract*—Intelligent virtual agents are used to accomplish complex multi-modal tasks such as human instruction comprehension in mixed-reality environments by increasingly adopting richer, energy-intensive sensors and processing pipelines. In such applications, the *context* for activating sensors and processing blocks required to accomplish a given task instance is usually manifested via multiple sensing modes. Based on this observation, we introduce a novel Commit-and-Switch (*CAS*) paradigm that simultaneously seeks to reduce both *sensing* and *processing* energy. In *CAS*, we first commit to a low-energy computational pipeline with a subset of available sensors. Then, the task context estimated by this pipeline is used to optionally switch to another energy-intensive DNN pipeline and activate additional sensors. We demonstrate how *CAS's* paradigm of interweaving DNN computation and sensor triggering can be instantiated principally by constructing multi-head DNN models and jointly optimizing the accuracy and sensing costs associated with different heads. We exemplify *CAS* via the development of the *RealGIN-MH* model for multi-modal target acquisition tasks, a core enabler of immersive human-agent interaction. *RealGIN-MH* achieves 12.9x reduction in energy overheads, while outperforming baseline dynamic model optimization approaches.**

***Index Terms*—Vision and Sensor-Based Control; Deep Learning for Visual Perception; Embedded Systems for Robotic and Automation; Human-Robot Collaboration; RGB-D Perception**

I. Introduction

THE progression of new sensing (e.g.LiDAR-based depth sensing) and DNN-based advanced perception capabilities in mobile and wearable devices enables more sophisticated *multi-modal, situated* mixed-reality and spatial computing applications. Figure 1 illustrates a Shopping Assistant application, where a customer with an Augmented Reality (AR) smart-glass (e.g. Microsoft Hololens™, Apple VisionPro™) gazes to a shelf in a supermarket aisle, points in the general direction of a product of interest and asks "What's the price of that white soap bottle with a green cap?". Using a *multi-modal* DNN inference pipeline (e.g., [1]), the smart-glass fuses verbal, visual and pointing cues (captured by an embedded microphone, RGB camera and depth sensors) to extract the *target* object and generate a real-time response. Such inference pipelines are vital for emerging spatial computing applications involving human-agent interaction, but they are computationally complex and energy-intensive.

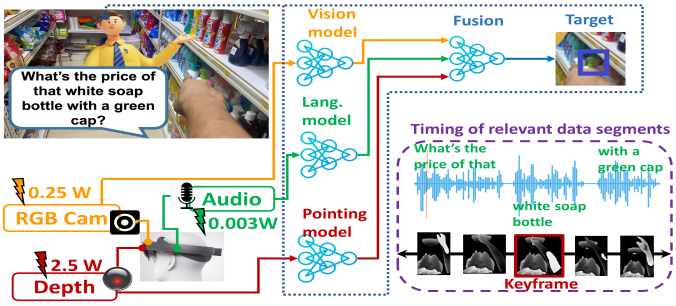


Fig. 1: Motivating application- A virtual shopping assistant

In this paper, we study an exemplar *multi-modal target acquisition* task of identifying the user's referred object in Figure 1 to highlight the following software and hardware factors contributing to resource-consumption: (a) the computational footprint of the multi-modal inference pipeline and (b) the energy-cost of energy-hungry sensors (especially LIDAR sensors). To address the software factor, recent work has optimized state-of-the-art single-stage multi-modal DNN perception models (e.g., RealGIN [1], RCCF [2]) by employing techniques such as (a) static model pruning (ShuffleNet [3]) and (b) complexity-aware dynamic model selection [4]. The core idea in such optimizations is to use some early feature representation to identify potential *instance-dependent* redundancy among the cross-modal cues, thus simplifying the overall computation. For example, in an uncluttered environment (Figure 2(b)), a rough estimation of pointing direction together with RGB scene analysis may be sufficient for identifying the target-object, whereas a more cluttered environment (Figure 2(a)) may require RGB scene analysis plus more complex parsing of the longer verbal command as well as precise, depth sensor-based, estimation of pointing coordinates. However, these approaches do not optimize the *hardware sensing energy* overhead (even though the energy

Manuscript received: May, 28, 2024; Revised August, 22, 2024; Accepted September, 08, 2024.

This paper was recommended for publication by Editor Pascal Vasseur upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the following funding organizations: 1) National Research Foundation, Prime Minister's Office, Singapore under both its Campus for Research Excellence and Technological Enterprise (CREATE) program and the NRF Investigatorship grant (NRF-NRFI05-2019-0007). The Mens, Manus, and Machina (M3S) is an interdisciplinary research group (IRG) of the Singapore MIT Alliance for Research and Technology (SMART) centre; 2) Agency for Science, Technology and Research (A*STAR), Singapore under its AME Programmatic Funding Scheme (Project #A18A2b0046). 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 or the A*STAR, Singapore

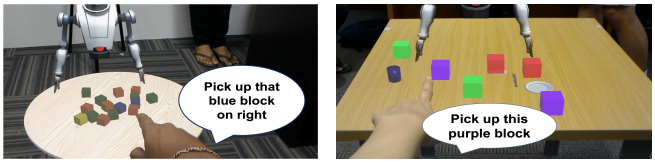
[1]First author is with Singapore-MIT Alliance for Research & Technology dulanga.weerakoon@smart.mit.edu

[2]Second Author is with Inst. of High Perf. Computing, A*STAR, Singapore vigneshwaran_subbaraju@ihpc.a-star.edu.sg

[3]Third Author is with Inst. for Infocomm Research, A*STAR, Singapore joohwee@i2r.a-star.edu.sg

[2]Fourth Author is with Singapore Management University archanm@smu.edu.sg

Digital Object Identifier (DOI): see top of this page.



(a) Cluttered setup (b) Uncluttered setup

Fig. 2: Diversity in multi-modal instructions

overhead of sensing is comparable to that of DNN-based inference–see Section VI-B), as all the sensors remain active throughout irrespective of their eventual relevance for a specific task instance. More specifically, as illustrated in Figure 1, the depth sensor needed for accurate pointing resolution is very energy hungry, consuming ∼10x the power of on-board RGB and microphone sensors.

We thus, propose a new paradigm, called *Commit-and-Switch* (**CAS**), designed to simultaneously reduce both *hardware sensing* and *software inference* overheads associated with the on-device execution of such multi-modal DNNs. The core *CAS* concept involves the use of triggered sensor activation, whereby more energy-hungry sensors are activated on demand, only if deemed necessary, *based on the complexity of the current task instance*. However, this is challenging for two important reasons:

1) Determining the task complexity is non-trivial and may need different sensing modalities and a distinct DNN (e.g.,see [4]) for itself.
2) The "interval of relevance" of the data from each sensor dynamically varies for different task instances, making it challenging to trigger sensors on-the-fly. For example, in Figure 1, the depth sensor's 'keyframe' (when the user's hand/fingers point at the target) occurs *before* the completion of the verbal command. So, using verbal instruction complexity to trigger the depth sensor may miss the pointing gesture, although the microphone is the most energy-efficient of the three sensors in Figure 1.

To overcome these challenges and leverage triggered sensing, *CAS* unifies complexity determination and task inference into a single DNN pipeline with multiple complexity-driven processing branches (associated with dynamic sensor triggering) and heads, identified via a *principled* cost-benefit analysis technique. Our main contributions are,

- We propose the *CAS* paradigm that simultaneously addresses the issues of sensing (hardware) and processing energy (software) overheads by dynamically switching between different processing piplelines and activating the corresponding sensors on-the-fly.
- We demonstrate *CAS* by developing *RealGIN-MH*, a multi-branch model for multi-modal target object acquisition instruction comprehension. Given a set of three possible sensor combinations (RGB camera alone, <RGB cam+audio> and <RGB cam+audio+depth>) *RealGIN-MH* initially commits to a branch that uses RGB camera data alone. It uses features from the committed branch to enable the energy-intensive depth camera on-the-fly, only when warranted. *RealGIN-MH* uses a new module to regenerate past depth key-frame from the depth frames available after sensor activation. This unified model utilizes ∼12.9x lower energy while achieving similar accuracy.
- We demonstrate the generalizability of *CAS* by using an additional multi-modal task: semantic segmentation of simultaneously acquired RGB and thermal camera images. For this, we apply *CAS* to develop a new multi-headed *PSTNet-Thermal-MH* model, which activates the power-hungry thermal camera only 36% of the time, achieving $> 50\%$ energy savings over a baseline CNN-based embedded PSTNet-Thermal [5] model.

Overall, *CAS* makes the case for tighter software coupling between GPU-based inferencing and sensor hardware, thereby allowing intermediate states of DNN pipelines to be used for dynamic activation and control of sensors.

II. Related Work

Various mobile/wearable sensing techniques have been proposed to capture audio and gestural instructions to support interactive AR applications. While [6] demonstrated the importance of audio/speech interactions for natural MR interactions, gestures such as pointing, grabbing, and stretching have been shown to increase the immersiveness of MR systems [7]. Researchers have also explored [8]–[10] the joint use of gestural and audio cues to better capture human intent. The task of real-time fusion of pointing gestures with verbal instructions to interpret instructions referring to tabletop objects has been studied by several works on human-robot interaction [11], [12]. The M2GESTIC [13] system cues from pointing gestures could enhance performance, especially by reducing the ambiguity in verbal instructions. Dogan et al [14] have shown how the inclusion of *depth* features, in addition to RGB camera sensing, can significantly increase the accuracy of target acquisition tasks. These baseline multi-modal DNN models are, however, too heavyweight and need to be optimized for on-device execution on pervasive devices.

Both static optimization [15], [16] and runtime dynamic optimization [17], [18] approaches have been proposed to support low-latency DNN-based inference on pervasive devices, albeit principally using a single modality of sensor data. A limited body of work has recently applied the concept of dynamically switching between multiple different models to optimize execution efficiency, applied almost exclusively for vision tasks. For example, the MobiSR system [19] dispatches different portions of an image to different models to support complexity-aware generation (upscaling) of super-resolution images, whereas Verelst et al., [20] utilize the concept of dynamically sparsified processing to execute intensive convolution operations only on important Regions of Interest (RoI) in an image. While generic, these approaches consider a single sensing modality and fail to account for the redundancy and/or correlation across multiple sensor modalities. The recent COSM2IC approach [4] extends the dynamic model switching paradigm to a multi-modal instruction comprehension task. COSM2IC loads up multiple DNN models, with differing sensor inputs and accuracy, into an embedded device, and then uses a lightweight complexity estimator as a preprocessor to dynamically demultiplex inference execution across these models, thereby reducing average latency and processing en-

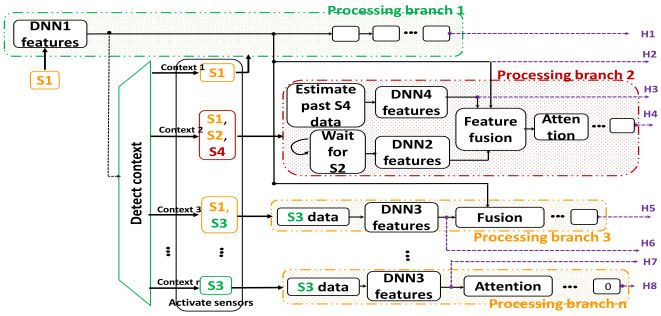


Fig. 3: Commit-and-Switch (CAS) paradigm.

ergy. However, it does not support on-demand sensor triggering and assumes all sensors remain active throughout, even if unused. In contrast to these approaches, we aim to jointly optimise *both* sensing and processing energy.

III. CAS: Overview

Our proposed *CAS* paradigm is illustrated in Figure 3, where the sensors are denoted by S1, S2 etc. and their corresponding font colors represent their power consumption (Green $\rightarrow$ Low power, Orange $\rightarrow$ Medium power and Red $\rightarrow$ High power). Similar color codes are used for branches representing data processing. In this approach, the determination of the sensing context is not a distinct step, but is *integrated* into the DNN-based inference pipeline. This context detector can conceptually utilize an intermediate state from any layer of the DNN processing pipeline, chosen so as to exploit problem-dependent trade-offs between efficiency and accuracy. In *CAS*, we first *commit* to a processing branch (branch 1 in the figure) that depends on a low/medium power sensor(s). The energy-intensive sensors and their corresponding processing pipelines are inactive at this point. Even as the processing in the initially committed branch goes on, *CAS* piggy-backs on the DNN features already generated in this branch to make a classification of the task context; consequently, context determination is considered to be relatively low energy (Green). This context is then used to potentially switch to other processing branches (e.g., branch 2), which may require the activation of corresponding additional energy-intensive sensors; else, the initially committed branch is executed in its entirety without activating any additional sensors. CAS must also accommodate the likelihood that even a modest triggering latency can cause task-critical sensor data (say from S4) to be missing. We note, however, that in multi-modal sensing, the likely correlation across sensor observations raises the possibility of *estimating* the missing data of a sensor from the currently-available data stream of other sensor(s). This is reflected by the processing block "Estimate past S4 data" in branch 2. The reverse situation, illustrated by the block "Wait for S2" in the figure, whereby some sensor data may not be readily available is also possible. For example, if the user issues a long verbal command, the inferencing task may need to wait until the verbal instruction is complete.

Thus, *CAS* based optimization of a complex multi-DNN inference pipeline (Eg. RealGIN-lite [4]) involves – (a) determination of efficient processing branches and (b) determination of precise intermediate processing step where the task context is estimated. In step (a), we consider the entire inference pipeline and examine potential early-exit opportunities (known as output-heads). The key objective in this step is to carefully identify energy-efficient paths within the pipeline that can be used to perform reliable inference, at least under certain task contexts. For step (b), the decision point for task context should ideally be as late as possible to ensure the highest accuracy and minimal activation of unnecessary sensors, but not too late as to miss task-critical segments of sensor data. In *CAS*, we identify suitable points for such context switching from the energy-efficient processing branches identified in step (a) and then utilize their accuracy-vs.-energy trade-off characteristics to choose an optimal candidate.

IV. Multi-modal instructions setup

The problem of *target acquisition* from naturalistic multi-modal instructions has been studied in detail by several earlier works such as [4], [11]–[13]. Recent work by Weerakoon et al. builds upon the earlier works to introduce a comprehensive corpus of multi-modal instructions (known as the COSM2IC dataset) involving different levels of ambiguity, clutter etc. Figure 2 illustrates a typical multi-modal instruction. As shown in the figure, the instructions involve a pointing gesture as well as a verbal description of the target object that needs to be selected. The dataset comprises a total of $\sim$200 unique <block arrangement, target block> tuples, corresponding to different levels of scene complexity, with 28 unique individuals generating a total of $\sim$3000 instructions across these tuples. Therefore, the COSM2IC dataset, which includes approximately 3,000 instructions from 28 unique users, effectively captures a broad range of human behavior in issuing instructions. For each instruction, we use (a) the data from the RGB camera (from the view-point shown in the figure), (b) the transcribed text of the verbal instruction, and (c) depth camera data (from the same view-point as the RGB camera).

Sensor Energy Profiles: To determine judicious choices for different inference branches, we also need to quantify the relative energy overheads of the different sensors. We used RealSense L515 [21] as our representative depth sensor in our evaluations. Measurements performed using a Monsoon power monitor revealed that RealSense consumes $\sim$2.5W of power for capturing depth frames, which was nearly 10x higher than the operating power of a typical RGB camera. As we shall later see (Section VI), this implies an energy consumption of about 388 mJ (nearly half of the inference energy of the RealG(2)In-Lite model) if the depth sensor is active for a duration equal to the average execution latency of the RealG(2)In-Lite model on the COSM2IC dataset, when evaluated on a Jetson TX2 device. An analysis of the dataset reveals the following characteristics that will influence the choice of different branches and triggering sensors:

- *Possible Sensing Redundancy:* The location indicated by the pointing gesture can be sensed either via the RGB camera or via the depth camera. While depth data provides higher pointing resolution, under low scene clutter, the RGB camera alone may suffice. Hence, the energy-intensive depth camera should be activated only when needed. Also, given

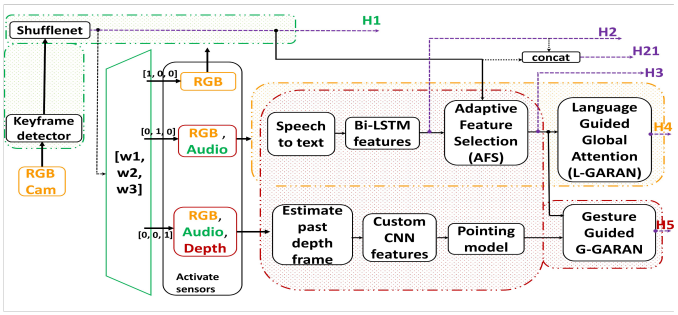


Fig. 4: *RealGIN-MH* Architecture

this redundancy, it *may* be possible to regenerate past values of depth camera data using a combination of concurrent RGB data and depth frames acquired with a modest delay.

- *Optimizing Audio Sensing:* The RGB camera sensor data is indispensable for target acquisition, as object detection typically relies on it. While in some cases, target identification may be possible using only RGB and pointing data (without verbal input), especially in uncluttered scenes. However, on-demand, delayed activation of the audio sensor is not feasible, as there is simply no alternate way to reconstruct past verbal instructions.

V. REALGIN-MH: CAS-BASED INFERENCING

We now detail the design of *RealGIN-MH*, which employs the CAS to perform on-device multi-modal instruction comprehension. Figure 4 shows the exact sensors used in each of the three main processing branches involved in *RealGIN-MH*. The low complexity branch (enclosed by green dotted lines) performs the comprehension task using only the RGB data. This branch involves a key-frame detector that identifies a key-frame and a Shufflenet [3] visual backbone to extract features from the detected key-frame. Further processing blocks in this branch use these features to directly output the location of the target block (based solely on the pointed location from the key-frame) via output head **H1**. The features from the Shufflenet backbone are also used by the context detector (represented as the green demultiplexer box) to output a 3-element binary vector representing the branch chosen for subsequent execution.

If the context detector detects a low complexity context ([1,0,0]), the inferencing process continues along the low complexity branch. The medium complexity branch is activated for a context vector value of [0,1,0]) and is enclosed by the orange dotted lines in Figure 4. This block uses sensor data from both the RGB camera and audio sensor. The processing blocks involved in this branch are the speech-to-text module, the Bidirectional LSTM to extract features from the text, the Adaptive Feature Selection (AFS) module and the language-guided global attention L-GARAN (all explained shortly), which outputs the target object (head **H4**). Similarly, the high complexity branch is enclosed by the red-dotted lines in the figure. This branch uses all three sensors (RGB camera, audio and depth camera). This branch has significant overlap with the blocks in the medium processing pipeline as it also uses the text-to-speech module, Bi-LSTM and the AFS. Additionally, when processing the depth data, it first tries to *reconstruct* an estimate of the past depth-frame that is in sync with the RGB keyframe. The features from this depth image are provided to a pointing model. Finally, a gesture-guided G-GARAN module (instead of the L-GARAN used in the medium complexity pipeline) is used to output the target object location via head **H5**. Across all pipelines, the RGB and audio sensor (capturing verbal inputs) are always active (even though the low complexity branch does not utilize verbal cues), with *RealGIN-MH* focusing on dynamic activation of the energy-hungry depth sensor.

A. Keyframe extraction and Shufflenet backbone

Empirical observation shows that the most informative segment (which we call the "keyframe") in a pointing gesture corresponds to one where the hand is momentarily stationary, steadily indicating the target. Keyframe detection is done via a 4-layer CNN network with ReLU [22] activation for intermediate layers and Softmax activation for the final layer. This model, trained for 10 epochs using a balanced set of COSM2IC ground truth data, accepts an incoming RGB frame as an input and outputs its probability of being a key frame (class 0='not key', class 1='key'). During inferencing, an RGB frame is identified as a keyframe if class 1 probability is $\geq$0.8; the Shufflenet visual backbone then extracts the visual features used by subsequent processing blocks.

B. AFS & Language-guided Global Attentive Reasoning (L-GARAN)

We follow the same approach as explained in [1] for calculating AFS & L-GARAN features. The visual backbone computes features at different feature scales. Let us assume that these features are $F_{v1} \in \mathcal{R}^{m_1 \times m_1 \times s_1}$, $F_{v2} \in \mathcal{R}^{m_2 \times m_2 \times s_2}$, $F_{v3} \in \mathcal{R}^{m_3 \times m_3 \times s_3}$. $m_1 > m_2 > m_3$ refer to the resolutions of feature maps and s_1, s_2 and s_3 refers to the feature channels. Let language feature embedding computed with an LSTM be f_t. AFS features are calculated as follows,

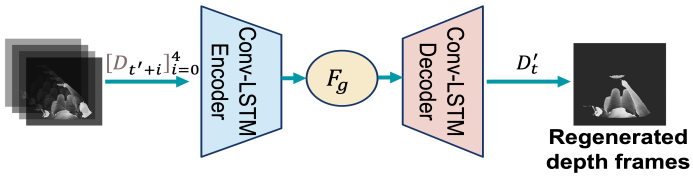
$$\begin{aligned} [\beta_1, \beta_2, \beta_3] &= F_{AFS}(f_t) \\ F_v &= \beta_1 * F_{v1} + \beta_2 * F_{v2} + \beta_3 * F_{v3} \end{aligned} \tag{1}$$

$\beta_1, \beta_2, \beta_3$ are determined from the f_t language embedding.

L-GARAN is a multi-modal attention component that uses language features as a pivot to compute a language attentional feature map F_{L-att} that identifies important regions in the visual feature map. This module is activated when $w_2 = 1$, and takes the AFS features as an input.

C. DDPM: Delayed Depth Backpropagation

RealGIN-MH employs on-demand triggering for the energy-hungry depth sensor: once this module is activated, a trigger signal is sent to the depth camera to capture and stream $N = 5$ depth frames. After streaming, the depth sensor reverts to its low-power 'sleep' mode. The power overhead in 'sleep' and 'streaming' state is 1.5W and 2.5W, respectively. We experimentally observed an activation delay of $\sim$400 msec across 3 different commercial depth sensors (Leapmotion, Kinect DK and RealSense. This delay can cause the captured depth image frame to be significantly delayed from the *key* RGB frame, thereby resulting in incorrect pointing resolution.



(a) DDPM architecture

(b) Original image (c) DDPM output

Fig. 5: Delayed Depth backProp. Model (DDPM)

Therefore, we have developed the DDPM model to regenerate, using the 5 delayed depth frames captured on activation, the depth frame corresponding to the RGB keyframe. As shown in figure 5(a), we use a Convolutional-LSTM Encoder and Decoder model to perform this regeneration. Let D_t be the depth frame at the *key* frame, $D_{t'}$ be the first depth frame after the startup delay, $D_{(t'+1)}$ - $D_{(t'+4)}$ be the subsequent 4 depth frames and $D_t^{'}$ be the regenerated depth frame. Then, our encoder-decoder Conv-LSTM model is calculated as follows.

$$\begin{aligned} F_g &= Conv-LSTM_{Encoder}(\{D_{(t'+i)}\}_{i=0}^{4}) \\ D_t^{'} &= Conv-LSTM_{Decoder}(F_g) \end{aligned} \tag{2}$$

D. Gesture-guided Global Attentive Reasoning (G-GARAN)

The computation of this component is identical to L-GARAN except, instead of f_t language features, we use F_g (encoder output of DDPM) as a pivot. Intuitively, the model focuses greater visual attention around the pointed location.

E. CAS - *Selection of output heads*

As previously discussed, the instruction comprehension pipeline consists of multiple modules that accept different modalities as inputs. The baseline RealG(2)IN-lite contains only one output-head (corresponding to H5 in Fig 4) that utilizes all of these sensor inputs and modules for every single input instruction. So as a first step in the *CAS* paradigm, we identify potential exit-paths that constitute the branches of processing pipelines, each offering varying accuracy and energy trade-offs. To determine the optimal branch points and compute heads, we propose an iterative training approach.

In this iterative training approach, we initially introduce $N = 6$ compute heads into the RealG(2)IN-Lite comprehension pipeline, as depicted in Figure 4. H1 – H5 represent various potential exit points from different processing blocks of the RealG(2)IN-Lite model, utilizing different sensor combinations. We also introduced a hybrid-branch H21, which concatenates the features for H2 and H1 and thereby uses both audio and RGB camera sensor data streams. Note that the context detector is disabled at this step. These compute heads are strategically selected to cover different endpoints of the comprehension pipeline, and each head is associated with an energy cost C_i. Here, C_i represents the sum of processing and sensing energy required for executing the i^{th} compute head. Subsequently, we iteratively train each compute head, following the forward computation order for each batch of data samples from the training set. This training process helps determine the IoU $A_i = IoU(pred, gt)$, where A_i quantifies the intersection over union (IoU) value between the predicted and ground truth bounding boxes. To select the optimal $K = 3$ heads from the initial set of N heads, we follow the following principles:

1) We always choose the head with the highest A_i to limit the drop in accuracy resulting from dynamic switching among different heads. Usually, this tends to be the head that involves the most energy-intensive and high-fidelity sensing and processing.
2) We compute the Efficiency, $E_i = \frac{A_i}{C_i}$. The remaining $K-1$ branches are then selected based on the highest efficiencies, achieving a balance between accuracy and energy cost. Usually, these are heads that can do the job far more efficiently than the most accurate head, for a significant proportion of the inputs, but fail when encountered with complicated inputs.

TABLE I: Accuracy, cost and efficiency for various compute heads on COSM2IC dataset

	H1	H2	H21	H3	**H4**	**H5**
Cost	**0.3**	0.3	0.5	0.5	**0.5**	**10.9**
Accuracy	**0.67**	0.01	0.7	0.73	**0.74**	**0.78**
Efficiency	**2.23**	0.03	1.40	1.46	**1.48**	**0.07**

Table I provides the accuracy, cost and efficiency of each compute head. Heads that are marked in bold are the chosen K heads. Based on the *CAS* principle, we first chose H5 which yields the highest accuracy. We then chose H1 and H4 as the two highest-efficiency compute heads.

F. CAS - *Determining the timing of context detection and initial branch*

Next, we explored how the choice of placing the context detector at the early exit-points of the energy-efficient compute heads (H1 and H4) impacts *RealGIN-MH* performance.

As shown in Table II, the configuration '*Context @ Shufflenet*' (RealGIN-MH) achieves the highest accuracy, latency, and energy efficiency. This indicates that Shufflenet features are effective in making an accurate enough context determination, in-time. On the other hand, the configuration '*Context @ LSTM*', which relies solely on LSTM features from the audio data, achieves significantly lower accuracy, primarily for task instances where pointing input is important. This is expected since it is very likely that the verbal instruction ends much later than the corresponding pointing gesture, at which point it is too late to trigger the RGB and depth sensor to capture the pointing hand. This suggests that even though the audio sensor consumes the least energy, it is not suitable as a detector of task context. This example also illustrates that blindly relying on the lowest-energy sensor to determine the context to trigger the high-energy sensors is not appropriate for our multi-modal instruction comprehension task, due to asynchronous input. The configuration '*Context @ LSTM+Shufflenet*', which combines language and visual features, offers a potentially better feature representation for context determination. However, it comes at the cost of higher overall latency and energy consumption, with a lower accuracy

TABLE II: Performance variations with context detectors added at various branch points

Context @	Acc. (%)	Lat. (ms)	Energy (mJ)		
			Proc.	Sens.	Total
Shufflenet (H1)	**76.46**	**130**	**710**	**130**	**840**
LSTM (H2)	59.23	138	740	145	885
LSTM + Shufflenet (H21)	75.29	145	775	145	920
AFS (H3)	74.19	147	800	140	940
L-GARAN (H4)	73.56	150	820	135	955

compared to '*Context @ Shufflenet*'. This loss in performance can be attributed to the increased delay in the decision to activate the depth camera, leading to higher pixel errors according to Figure 6(a). Thus, from Table II, we can decide that our initial committed processing branch would be H1, which uses the RGB camera data.

G. RealGIN-MH - *Multiple output heads*

As depicted in figure 4, H2, H21 and H3 marked in purple are the redundant compute heads based on *CAS* -based optimal head selection approach. Thus, we remove these redundant heads and only activate H1, H4 and H5 for this step to subsequently train the context detector. Each output head provides a bounding box of the target object via Feature Pyramid Network (FPN) and regression. Let us assume, I_t as the key frame (RGB), L as the text instruction and G as the depth frames captured after the sensor is triggered. Let F_{out} be the final feature map for bounding box regression. At runtime, we dynamically choose a specific compute head based on the context estimated by the context detector module.

To predict the task context, we use the visual features generated by the Shufflenet backbone. We add a 2-layer CNN network followed by a fully connected layer to compute the feature embedding necessary to predict the visual complexity. This feature embedding is then sent to 3-neuron fully connected layer with Gumbel-Softmax activation function [23] to compute the discrete task context triple:

$$w_1, w_2, w_3 = G(F(f_v)); \quad where\ w_1, w_2, w_3 \in \{0, 1\} \quad (3)$$

As depicted in equation 3, we compute w_1, w_2, w_3 representing 3 distinct complexity levels, and the corresponding branches. When $w_1 = 1$ *RealGIN-MH* only uses a Shufflenet backbone for comprehension; when $w_2 = 1$, Shufflenet backbone for vision, Bi-LSTM for language and AFS gets activated, while $w_3 = 1$ implies the activation of all the modules (including the depth camera and the DDPM module).

Once the context is determined, we then define two forward computations in training and inference mode.

- **Training Mode** - In the training mode, to achieve differentiability during the backpropagation stage, we compute all three branches (regardless of the computed values w_1, w_2, w_3) as:

$$F_{H1} = H1(I_t); F_{H4} = H4(F_{H1}, L); F_{H5} = H5(F_{H4}, G)$$
$$F_{out} = w_1 * F_{H1} + w_2 * F_{H4} + w_3 * F_{H5} \quad (4)$$

Furthermore, we modify the original loss function of RealG(2)In-Lite l_{orig} as follows to add the policy for selecting the optimal compute head:

$$loss = l_{orig} + \frac{1}{N} * \sum_{i=0}^{N} (e_1 * w_1^i + e_2 * w_2^i + e_3 * w_3^i) \quad (5)$$

Here, e_1, e_2 and e_3 are the relative energy costs for respective branch point and N is the batch size. Based on our energy profiling on Jetson TX2, we identified that the total energy for $H1$ is 561 mJ, $H4$ is 775 mJ and $H5$ is 10,915 mJ ($\sim$20x higher than $H1$). Thus we choose, $e_1 = 561/(561 + 775 + 10915) = 0.046$, $e_2 = 775/(561 + 775 + 10915) = 0.063$ and $e_3 = 10915/(561 + 775 + 10915) = 0.89$.

- **Inference Mode** - In the inference mode, to achieve savings in latency we compute *only* the relevant branch based on the task complexity.

$$if\ \ w_1 = 1 \rightarrow F_{out} = H1(I_t)$$
$$if\ \ w_2 = 1 \rightarrow F_{out} = H4(H1(I_t), L) \quad (6)$$
$$if\ \ w_3 = 1 \rightarrow F_{out} = H5(H4(H1(I_t), L), G)$$

VI. Results

We evaluated *RealGIN-MH* and various baselines using the COSM2IC multi-modal instruction dataset. Since the Microsoft Hololens lacked computational resources for baseline models and could not measure energy consumption when toggling the depth sensor, we used an NVIDIA Jetson TX2 device to execute our models interfaced with a RealSense L515 depth sensor that can be easily toggled On/Off (via software commands). Power consumption was accurately measured using a Monsoon power monitor. The Jetson TX2 also ran a real-time speech-to-text model, Picovoice cheetah [24], converting audio into text. Thus, in our experimental setup, audio and RGB camera data corresponding to COSM2IC's environmental setup and verbal instructions, are captured on the HoloLens and then streamed to the nearby Jetson TX2 for executing *RealGIN-MH* and baselines.

A. Evaluation Metrics

Similar to COSM2IC, we assume that the comprehension task is successful if the mid-point of the predicted bounding box lies within the ground-truth target object boundary. We measure the depth sensing energy separately, as the other sensors are always on and thus have a constant energy consumption across all approaches. Since we observed that the L515 sensor consumes 1.5W of static power, we only measure the additional dynamic power consumed when the sensor is triggered to stream depth frames. For a comparison of energy consumption (Tables III & IV), we use the *average energy* consumed over all the instructions in the COSM2IC dataset– i.e., total energy consumed for the entire set of instructions, divided by the total instructions in the dataset.

B. RealGIN-MH *Performance against other baselines*

Table III summarizes the performance of *RealGIN-MH* against other baselines. For both RealG(2)In-Lite (end-to-end DNN) and COSM2IC (branch switching approach), the

TABLE III: *RealGIN-MH* performance against baselines

Model	Acc. (%)	Lat. (ms)	Energy (mJ)		
			Proc.	Sens.	Total
RealG(2)In-Lite	78.51	155	853	10000	10853
COSM2IC	76.13	110	590	10000	10590
RealGIN-MH-noDDPM	73.97	115	630	135	745
RealGIN-MH	**76.46**	**130**	**710**	**130**	**840**
RealG(2)In-MobViT	80.14	250	1410	10000	11410
RealGIN-MH-MobViT	**78.28**	**210**	**1100**	**140**	**1240**

depth sensor is assumed to be always-on, thereby consuming ∼10,000mJ energy/instruction. While COSM2IC optimizes the inferencing overhead, this translates to only 5.5% savings in the total energy cost. In contrast, *CAS*-based *RealGIN-MH* jointly reduces both processing energy and sensing energy significantly. In total, *RealGIN-MH* achieves ∼**12.9x** savings in total energy in comparison to RealG(2)In-Lite while maintaining a similar latency and suffering $< 2\%$ loss in task accuracy. We also evaluated the performance without the DDPM module (RealGIN-MH-noDDPM). This incurs ∼10% lower latency and consumes ∼10% (80mJ) lower energy than *RealGIN-MH*. However, the task accuracy drops by an additional ∼2.5%. We also studied the performance after replacing the Shufflenet backbone in both RealG(2)In-Lite and *RealGIN-MH* with a Mobile-VIT transformer [25]. We observe similar savings in latency and energy. Furthermore, Figure 7 shows the distribution of latency and total energy RealGIN-MH for the COSM2IC dataset. Given the adaptive nature of RealGIN-MH, latency varies between 90msec - 170msec and total energy between 550mJ - 1200mJ

(a) Startup Delay (b) No. of DDPM frames

Fig. 6: Average pixel error with DDPM

C. *Branch-specific performance of* RealGIN-MH

Table IV provides the performance of *RealGIN-MH* when all the instructions were forced to pass through a particular head. We see that while H1 (RGB only) or H4 (RGB + Audio) do not consume any depth-sensing energy, they both suffer from a significant degradation in accuracy. In comparison, solely using the branch H5 with always-on depth sensing results in a superior accuracy of 78.20% while consuming a higher sensing energy consumption of 10000 mJ. *RealGIN-MH* dynamically chooses these branch points based on a complexity assessment, executing heads H1, H4 and H5 for 25.75%, 23.16% and 51.08% of the total instructions, respectively. Consequently, *RealGIN-MH* achieves task accuracy (76.43%) which is comparable to H5, but with a much lower sensing energy of 130mJ.

D. *Pointing sensitivity analysis*

In Figure 6(a), we plot and observe how the average pointing error (in pixel distance) increases as the sensor activation delay increases. Thus, equipping future pervasive devices with faster sensor triggering capabilities may enable more accurate

TABLE IV: Head-based Perf. of *RealGIN-MH*

Head	Acc. (%)	Lat. (ms)	Energy (mJ)	
			Proc.	Sens.
H1	68.51	102	561	0
H4	73.18	141	775	0
H5	78.20	165	915	10000
MH	76.46	130	710	130

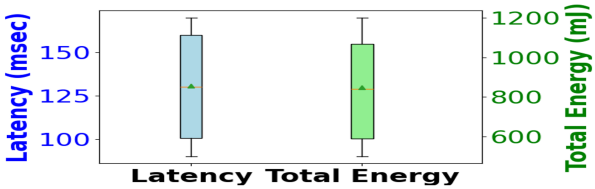


Fig. 7: *RealGIN-MH* Latency and Energy

pointing resolution and higher task accuracy. By varying the number of frames (N) used as an input to the DDPM, we observe (Figure 6(b)) that using a larger number of frames results in a lower pointing error, but increases the sensing energy overhead. We empirically chose $N = 5$ frames, as additional frames provide only a negligible reduction in the pointing error. Figure 5(b) & 5(c) visually illustrate (a) the 'keyframe' depth image–i.e., the depth image that we would have ideally used if the sensor was always on, and (b) the depth image regenerated using DDPM $N = 5$ frames. While the regeneration is not perfect, the pointing resolution is evidently adequate for the G-GARAN module in *RealGIN-MH*.

E. *Generalizability of* CAS

To evaluate the generalizability of *CAS* to other tasks, we applied it to a multi-modal segmentation task proposed in [5], where a DNN (PSTNet-Thermal) takes an RGB image and a thermal image as inputs and produces a segmentation output with 5 different classes.

TABLE V: Head-based accuracy on PST900 dataset

Branch	Cost	Accuracy	Efficiency
H0	0.03	0.46	15.33
H1	**0.04**	**0.67**	**16.75**
H2	**0.23**	**0.69**	**3**

Following the *CAS* principle, we added $N = 3$ heads to the PSTNet-Thermal, as illustrated in Figure 8. Through iterative training, we determined the accuracies and efficiencies of these heads, which are summarized in Table V. Based on our first principles, we then selected H1 and H2 for our dynamic, multi-head model called *PSTNet-Thermal-MH*, which was trained using the dynamic triggering approach.

PSTNet-Thermal-MH seeks to intelligently trigger the power-hungry thermal camera, which consumes 2.5W as per

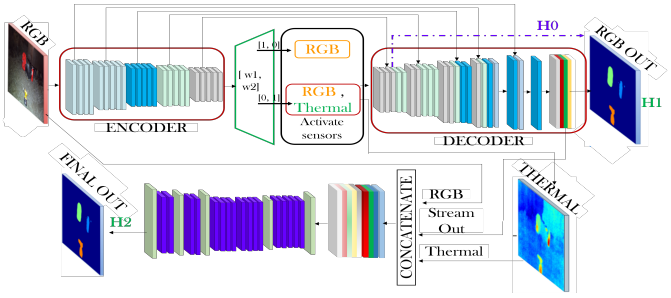


Fig. 8: Architecture of PSTNet-Thermal-MH

its technical specs) using a context detector placed at the encoder as shown in Figure 8. Table VI plots the resulting accuracy and energy overheads, using spec-based power values (thermal= 2.5W, RGB=0.5W), of *PSTNet-Thermal-MH* vs. alternative baselines. We observed that PSTNet-Thermal-MH activates the thermal camera only 36% of the time, and achieves a **2x** reduction in total energy consumption, compared to PSTNet-Thermal, without any accuracy loss.

TABLE VI: PSTNet-Therm-MH performance Vs baselines

Model	Acc. (mIoU)	Lat. (ms)	Energy (mJ)		
			Proc.	Sens.	Total
PSTNet	0.6765	20	50	5	55
PSTNet-Thermal	0.6837	45	121.50	123.75	245.25
PSTNet-Therm-MH	**0.6822**	**31**	**83.7**	**32.86**	**116.46**

VII. Discussion

A. Hardware triggering

RealGIN-MH only utilized software-based activation of the depth sensor. We observed a static power consumption of $\sim$1.5W even when the depth sensor is presumably in a low-power *standby* state. Additional energy savings can clearly be realized by introducing a hardware switch and supporting much faster ($\leq$100 msecs) sensor activation.

B. Improving DDPM Energy Efficiency

DDPM module, where a past depth frame is estimated using only a series of other depth frames, currently consumes non-trivial energy. For further energy optimization, it may be possible to perform depth image synthesis, using approaches such as Wofk et al. [26], from the already-available RGB frames sharing the same viewpoint. We could also consider stereo vision cameras (where *CAS* is used to selectively invoke the second camera) to replace expensive depth sensors.

VIII. Conclusion

We have introduced the *CAS* paradigm, which simultaneously reduces sensing and inferencing energy by dynamically switching between computational branches. Our *RealGIN-MH* model, which uses *CAS* optimization paradigm, for the task of multi-modal instruction comprehension, achieves a 12.9x reduction in energy overheads compared to baseline while achieving higher accuracy. Additionally, we demonstrate the generalizability of the *CAS* paradigm in a multi-modal segmentation task, where the *CAS*-based PSTNet-Thermal-MH model consumes approximately 2x less energy.

References

[1] Y. Zhou, R. Ji, G. Luo, X. Sun, J. Su, X. Ding, C.-W. Lin, and Q. Tian, "A real-time global inference network for one-stage referring expression comprehension," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[2] Y. Liao, S. Liu, G. Li, F. Wang, Y. Chen, C. Qian, and B. Li, "A real-time cross-modality correlation filtering method for referring expression comprehension," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 880–10 889.

[3] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6848–6856.

[4] D. Weerakoon , V. Subbaraju, T. Tran, and A. Misra, "COSM2IC: Optimizing real-time multi-modal instruction comprehension," vol. 7, no. 4, pp. 10 697–10 704, 2022.

[5] S. S. Shivakumar, N. Rodrigues, A. Zhou, I. D. Miller, V. Kumar, and C. J. Taylor, "Pst900: Rgb-thermal calibration, dataset and segmentation network," in *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2020, pp. 9441–9447.

[6] M. Cavazza, F. Charles, S. J. Mead, O. Martin, X. Marichal, and A. Nandi, "Multimodal acting in mixed reality interactive storytelling," *IEEE MultiMedia*, vol. 11, no. 3, pp. 30–39, July 2004.

[7] J. Y. Lee, G. W. Rhee, and D. W. Seo, "Hand gesture-based tangible interactions for manipulating virtual objects in a mixed reality environment," *The International Journal of Advanced Manufacturing Technology*, vol. 51, no. 9-12, pp. 1069–1082, 2010.

[8] M. Sargin, O. Aran, A. Karpov, F. Ofli, Y. Yasinnik, S. Wilson, E. Erzin, Y. Yemez, and A. Tekalp, "Combined gesture-speech analysis and speech driven gesture synthesis," *2012 IEEE International Conference on Multimedia and Expo*, vol. 0, pp. 893–896, 07 2006.

[9] I. Hara, F. Asano, H. Asoh, J. Ogata, N. Ichimura, Y. Kawai, F. Kanehiro, H. Hirukawa, and K. Yamamoto, "Robust speech interface based on audio and video information fusion for humanoid hrp-2," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, 2004, pp. 2404–2410 vol.3.

[10] E. Wolf, S. Klüber, C. Zimmerer, J.-L. Lugrin, and M. E. Latoschik, "" paint that object yellow": Multimodal interaction to enhance creativity during design tasks in vr," in *2019 International Conference on Multimodal Interaction*, 2019, pp. 195–204.

[11] D. Whitney, M. Eldon, J. Oberlin, and S. Tellex, "Interpreting multimodal referring expressions in real time," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 3331–3338.

[12] R. Scalise, S. Li, H. Admoni, S. Rosenthal, and S. S. Srinivasa, "Natural language instructions for human–robot collaborative manipulation," *The International Journal of Robotics Research*, vol. 37, no. 6, pp. 558–565, 2018.

[13] D. Weerakoon, V. Subbaraju, N. Karumpulli, T. Tran, Q. Xu, U.-X. Tan, J. H. Lim, and A. Misra, "Gesture enhanced comprehension of ambiguous human-to-robot instructions," in *Proceedings of the 2020 International Conference on Multimodal Interaction*, 2020, pp. 251–259.

[14] F. I. Dogan and I. Leite, "Using depth for improving referring expression comprehension in real-world environments," *arXiv preprint arXiv:2107.04658*, 2021.

[15] S. Yao, Y. Zhao, A. Zhang, L. Su, and T. Abdelzaher, "Deepiot: Compressing deep neural network structures for sensing systems with a compressor-critic framework," in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, 2017, pp. 1–14.

[16] S. Bhattacharya and N. D. Lane, "Sparsification and separation of deep learning layers for constrained resource inference on wearables," in *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, 2016, pp. 176–189.

[17] T. Bolukbasi, J. Wang, O. Dekel, and V. Saligrama, "Adaptive neural networks for efficient inference," in *International Conference on Machine Learning*, 2017, pp. 527–536.

[18] J. Lin, Y. Rao, J. Lu, and J. Zhou, "Runtime neural pruning," in *Advances in neural information processing systems*, 2017, pp. 2181–2191.

[19] R. Lee, S. I. Venieris, L. Dudziak, S. Bhattacharya, and N. D. Lane, "Mobisr: Efficient on-device super-resolution through heterogeneous mobile processors," in *The 25th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '19. New York, NY, USA: Association for Computing Machinery, 2019.

[20] T. Verelst and T. Tuytelaars, "Dynamic convolutions: Exploiting spatial sparsity for faster inference," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2320–2329.

[21] "Intel realsense lidar camera l515," https://www.intelrealsense.com/lidar-camera-l515/, accessed: 2022-10-08.

[22] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Icml*, 2010.

[23] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *International Conference on Learning Representations*, 2017.

[24] "Picovoice," https://picovoice.ai/, accessed: 2022-09-12.

[25] S. Mehta and M. Rastegari, "Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer," in *International Conference on Learning Representations*, 2022.

[26] D. Wofk, F. Ma, T.-J. Yang, S. Karaman, and V. Sze, "Fastdepth: Fast monocular depth estimation on embedded systems," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6101–6108.