# SMG: A System-level Modality Gating Facility for Fast and Energy-Efficient Multimodal Computing

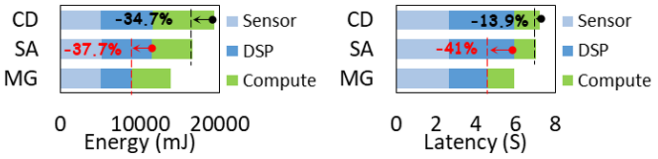
Xiaofeng Hou[1], Peng Tang[1], Chao Li[1], Jiacheng Liu[2], Cheng Xu[1], Kwang-Ting Cheng[3], Minyi Guo[1]
[1]*Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China*
[2]*Department of Computer Science and Engineering, The Chinese University of Hong Kong, China*
[3]*Hong Kong University of Science and Technology, Hong Kong, China*

***Abstract*—Achieving low-latency and high-efficiency multimodal computing (MMC) is crucial for deploying high-performance autonomous embedded systems (AES) that has limited energy budgets. However, existing methods have mainly focused on optimizing the computing phase and have overlooked the significant energy and latency overhead during the sensing phase. Therefore, we propose *SMG*, a *s*ystem-level *m*odality *g*ating facility to optimize this. Our approach introduces a software-defined DSP gating technique that enables MMC tasks to bypass both the sensing and computing phases of unimportant modalities. We also propose a raw data-activated MMC mechanism that comprises a fast modality tester and adaptive modality executor, which adapts to the modality gating architecture and performs energy-efficient MMC. To evaluate SMG, we implement a prototype of *SMG* by integrating it into existing AES and analyze it with extensive multimodal video recognition workloads. Our experimental results show that *SMG* outperforms SOTA approaches by adaptively gating some DSP operations, resulting in substantial improvements in both energy consumption and task latency.**

## I. Introduction

Multimodal computing (MMC) has emerged as a critical high-performance edge AI technology for building diverse autonomous embedded systems (AES) [1]–[3], which play a pivotal role in shaping future edge intelligence systems, such as edge micro data centers. It has been shown to improve the accuracy of intelligent tasks for a wide range of AES applications by up to 30% [4]. Researchers are actively developing energy-efficient MMC approaches to accelerate the deployment of MMC in real-world AES applications that are often interactive with the environment and operate under limited energy budgets [2], [5], [6]. These approaches aim to ensure situational robustness for safe and reliable operations while adhering to strict energy budgets [1], [5], [7]. They capitalize on the unique feature of MMC, where each modality exhibits different importance to different tasks. By adapting the number of computed modalities for different MMC tasks, they can significantly improve the efficiency of AES [5], [8].

Nevertheless, the existing methods [5], [9], [10] significantly overlook the overheads of the modality sensing phase, making their benefits underutilized. Multimodal AES applications rely on numerous sensors and sophisticated digital signal processing (DSP) modules to collect and preprocess environmental data in real-time [2], [11]. These modules account for the majority of energy consumption and execution latency in a MMC task. Our analysis in Section II shows that the sensing phase consumes approximately 44% and 73% of the total energy consumption and execution time in a MMC task, with DSP modules responsible for approximately 58% and 52% of the overall sensing energy and latency. This prompts us to think about a question: is there a method to implement situation-aware multimodal sensing and computing before DSP? The answer lies in *modality gating* which we define as a technique that enables flexible control and management of modality sensing stages. As shown in Figure 1 (detailed in Section II), modality gating (`MG`) shows great promise in reducing the energy and latency of MMC than the existing methods.




(a) Energy reduction (b) Latency improvement

Figure 1: Advantages of modality gating (`MG`) over the conventional designs (`CD`) and the state-of-the-art situation-aware approaches (`SA`): `MG` reduces more energy and latency in the sensing phase (i.e., `DSP`) by 37.7% and 41%, respectively.

Implementing modality gating for MMC is a non-trivial task because many existing architectural features and system mechanisms are not built for this purpose. Typically, the modality sensing phase involves collecting the modality signal at the front-end sensor array, directly feeding it into the DSP modules, and storing it in DRAM for MMC tasks [12], [13]. To provide more powerful DSP capabilities and streamline the sensors' architecture, many AES have migrated the DSP modules to the motherboard with dedicated interfaces and kernels. For example, the mainstream AES boards such as `NVIDIA's Jetson boards` [14] and `Raspberry pi boards` [15] have integrated various hardware DSP modules to enhance the sensing phases of images and videos. In addition, several general sensing control frameworks such as `gstreamer` [16], `libargus` [17] and `V4L2src` [18] have also been implemented in the existing systems to operate and control different sensors. Although the DSP modules have been decoupled at the bare-metal layer, the current approach to modality sensing still relies on a unified, end-to-end pipeline [19], [20]. This approach lacks interfaces for upper applications to manage the sensing phase, which hinders the ability to optimize the sensing phase for energy efficiency.

In addition, throttling the energy drain during the sensing

phase necessitates using raw data before the DSP to select the appropriate modalities for different tasks, which has not been extensively explored in prior work. Existing approaches rely on empirical statistics or hand-crafted heuristics, progressively computing more modality data based on feedback accuracy during MMC task execution [21], [22]. However, these methods may not be suitable for progressively sensing scenarios, as performing DSP based on feedback information can significantly increase response latency. Recently, a few works have utilized machine learning methods to pre-analyze modality data and select important modalities for different MMC tasks [8]. Nevertheless, these methods rely on pre-analysis of high-quality input data, rather than raw data. Pre-analysis of raw data requires more precise methods, as prediction errors can lead to higher remediation overhead. Additionally, the pre-analysis should not incur excessive overhead in the overall MMC task.

In this paper, we propose *SMG*, a ***s***ystem-level ***m***odality ***g***ating facility that can be easily integrated into existing AES without any additional hardware modifications. We implement *SMG* using application-architecture co-designs. At the architecture layer, we devise a software-controlled DSP gating technique that allows the upper application to control modality sensing phases. Its core module is a raw data buffer that separates the energy-hungry DSP from the frontend sensor array within the modality sensor pipeline. This buffer serves a crucial role in enabling the upper-level application to efficiently operate the DSP based on its specific task requirements. Additionally, it also consists of a set of interfaces to control the data flow of modality sensors. At the application layer, we propose a raw data-activated MMC mechanism. To bypass unnecessary modalities, we implement a fast modality tester to select the modalities for different tasks based on the raw data. Then, we leverage an adaptive modality executor to extract the features of the selected modalities and make the final decision. We integrate *SMG* into the mainstream industry AES board of `Jetson Xavier NX` and show the great potentiality of *SMG* in enabling low-latency, high-performance and energy-efficient AES. In all, we make the following contributions:

1) We characterize the features of MMC and identify the unexploited opportunities to improve the latency and energy efficiency of MMC in the modality sensing phase.
2) We propose *SMG*, a novel system-level modality gating facility that eliminates the DSP of unimportant modality data for different MMC tasks by a fast and accurate pre-analysis of raw modality data, thus reducing the latency and improving energy efficiency.
3) We implement SMG with a software-controlled DSP gating architecture without any need to upgrade the existing hardware. Also, our design approach considers the application-architecture co-designs, which optimizes the system for situation-aware, energy-efficient MMC.
4) We build a prototype of *SMG* and verify it with extensive real-world multimodal video recognition applications. Our results demonstrate that *SMG* can significantly reduce their latency and improve energy efficiency.

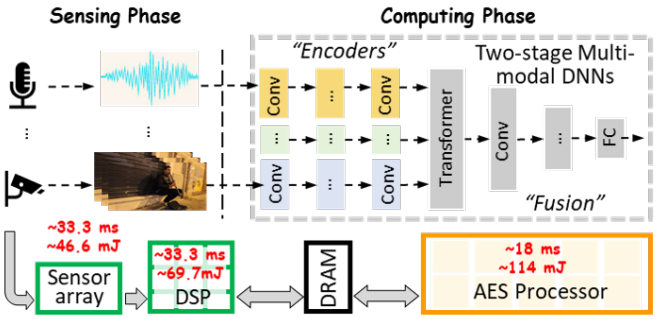


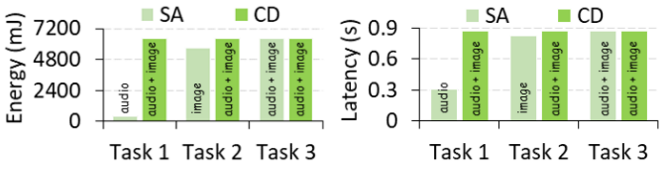

Figure 2: HW/SW features of multimodal AES.

## II. Motivations

### A. Characterization of Multimodal Computing

**MMC Hardware/Software Components:** MMC combines well-developed sensors and multimodal DNNs (MMDNNs) to achieve high-performance edge AI computing. It has recently attracted tremendous attention in a wide variety of AES applications [4], [8], [23]. For instance, video processing applications often rely on multiple modalities, such as RGB frames and audio [8], while autonomous machines such as unmanned aerial vehicles (UAVs) leverage multiple cameras, GPS sensors, and Lidars [5], [11], [23] to improve their operational correctness. As a result, MMC typically includes a sensing phase and a computing phase, making it a more complex process that goes beyond the conventional ones [1], [5], [11]. In the sensing phase, it leverages a wide variety of sensors to capture different environmental data and stores the data in DRAM. In the computing phase, it feeds the sensing data into perception modules that run multimodal deep neural networks to generate insightful results. Figure 2 illustrates the key hardware and software components of MMC. At the hardware layer, MMC consists of a set of sensors that collect different modality information and a powerful AES board that processes the data. At the software layer, the key component of MMC is two-stage MMDNNs [24]. In the first stage, parallel "encoders" are employed to extract distinct features from multi-modal data. The ensuing features are subsequently transmitted to the second stage of the framework to perform feature "fusion" and generate enhanced accuracy results.

**MMC Features:** We present a quantitative analysis and comparison of the performance and energy characteristics of MMC based on multimodal video recognition applications. These applications commonly encompass prevalent modalities, such as image and audio, and are frequently employed in various AES, including in-vehicle infotainment systems and online translation robots. Notably, image and audio modalities make up over 71% of the current modality data [25]. To conduct the following analysis, we leverage a well-known video recognition dataset called `UCF101` [26] and its details can be found in Section IV. To examine the features of MMC tasks, we first analyze three tasks with distinct situational requirements under conventional design (`CD`) without any optimization, existing situation-aware approaches (`SA`) that skip computing




(a) Energy consumption (b) Execution latency

Figure 3: Energy and latency under different MMC tasks.

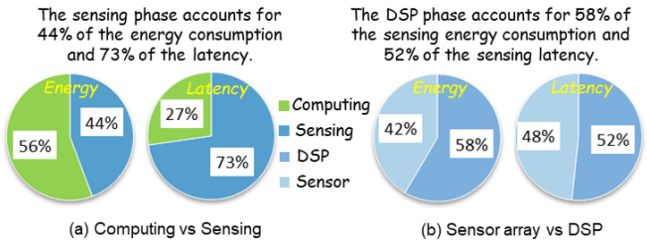



Figure 4: Breakdown of energy consumption and execution latency for different components.

phases of unimportant modalities, as depicted in Figure 3. Our results indicate that different tasks require different modalities, with some tasks solely relying on audio, some exclusively utilizing images, and others necessitating data from both audio and images. This highlights the significance of situational features in MMC task performance and presents opportunities to optimize the system efficiency. Thus, adopting situation-aware MMC can notably reduce energy consumption and execution delays, ultimately enhancing the overall efficiency of AES.

### B. Analysis of Modality Sensing

**Sensor Decoupling Architecture:** The modality sensing phase is an indispensable part of the MMC application, which is responsible for real-time acquisition of information from different modalities in the environment and providing the input sources for the AES applications. This phase typically consists of two main counterparts: 1) a sensor array that perceives the environmental signals and transfers them into digital signals; 2) a DSP (e.g., `ISP` for image [12] and `CODEC` for audio [13]) that pre-processes the digital signals and generates formatted data for the back-end MMDNN inference. Among them, the DSP plays a critical role in the modality sensing phase by performing essential operations such as noise reduction, filtering, and compression to improve the quality and accuracy of the input data. Traditionally, both the sensor array and DSP are placed in a unified, static sensor architecture. In recent years, the trend and benefits of modality pipeline reorganization have been recognized. There is an increasing trend of sensor decoupling architecture where the DSP component is moved from the sensor part to the compute board [27]. Doing so provides the AES applications with much more control over their energy consumption and latency.

To further understand the modality sensors in MMC applications, we analyze the energy consumption and performance characteristics of the modality sensing phase, which has been notably overlooked in previous studies. The results of our

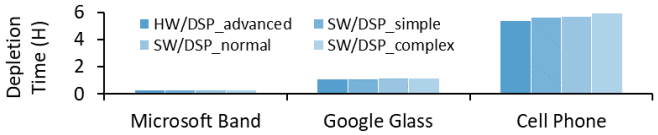



Figure 5: Battery depletion time with only DSP applied.

analysis, illustrated in Figure 4, reveal that the sensing phase (the blue area in the figure) can consume up to 44% of the energy and contribute to 73% of the task latency as shown in Figure 4-(a), significantly impacting the overall performance of the MMC system. We further divide the modality sensing results into two components: the sensor array and DSP. Our findings reveal that the DSP accounts for more than half of the energy consumption and latency as shown in Figure 4-(b). Additionally, in Figure 5, we measure the battery depletion time of three popular commercial AES platforms with only DSP applied. We consider one advanced hardware DSP method and three software DSP methods. The results show that the DSP alone can drain the batteries of these commercial AES platforms quickly within a few hours. The results highlight the critical importance of optimizing the sensing phase to enhance the efficiency and improve the latency of AES applications.

### C. Design Considerations

In Figure 1, we compare three MMC methods that optimize different phases: `CD` that does not employ any optimizations, `SA` that skips computing phases of unimportant modalities, and modality gating (`MG`) that throttles the DSP parts to reduce DSP computing. Compared with `CD`, `SA` can save 34.7% of energy consumption and improve 13.9% of latency. It is remarkable that throttling the DSP parts (`MG`) can save more energy by 37.7% and improve 41% latency compared with `SA`. The above analysis suggests that throttling the DSP part appears to be a promising approach to enhance the energy efficiency of AES.

While it is intuitive that one can turn off sensors dynamically to conserve energy and improve response time, applying this approach to MMC is challenging because doing so causes MMC tasks to become blind to the associated modality information, i.e., the modality data will be lost in the current MMC task. In the absence of complete modality information, one can hardly determine whether or not a typical MMC task is to be executed. Additionally, in some unexpected situations, for instance, if a task has already calculated a portion of modalities but finds that it still does not meet performance requirements, turning on the sensors of other modalities can lead to incorrect data timestamps. Therefore, it remains a non-trivial task that requires architectural support and system management to achieve optimal efficiency, accuracy, and latency. In the following sections, we provide an overview of our design considerations and the overall system architecture.

1) **A new sensing architecture is necessary to provide control interfaces for MMC applications to bypass the DSP**. To reduce both the computing and sensing energy of unimportant modalities for different MMC tasks, deactivating the DSP for those modalities is desirable. However, directly

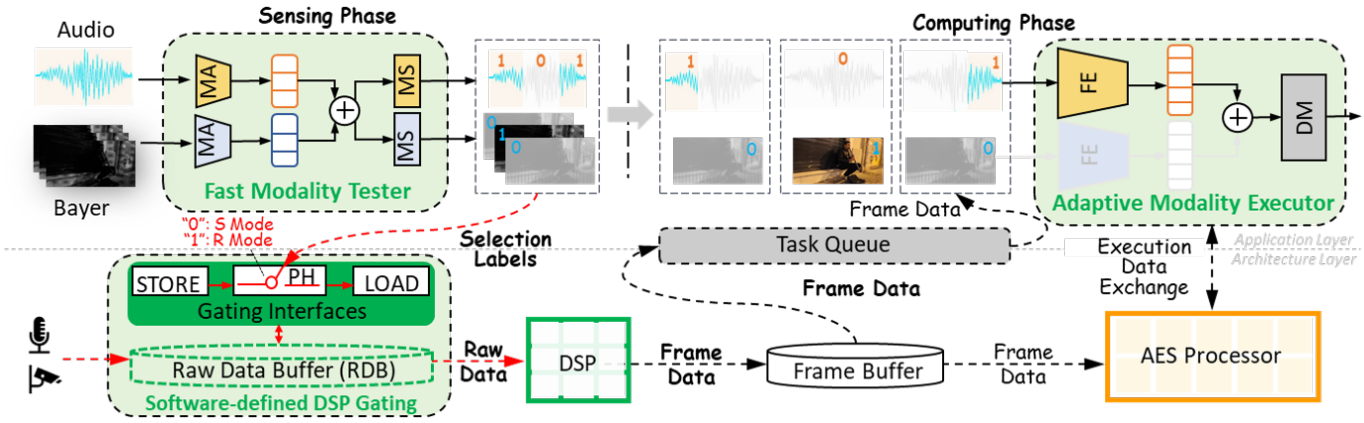



Figure 6: Technical overview of *SMG*, demonstrating the dataflow between the three modules.

turning off the sensors of unimportant modalities can cause missing information and incorrect timestamps. Therefore, it is essential to implement a sensing architecture that can deactivate the DSP for unimportant modalities while maintaining task situation awareness (raw data streaming).

2) **A raw data-aware MMC mechanism that adapts to the throttling energy drain of the DSP is necessary**. In the multimodal sensing phase, modalities must be selected and determined for different MMC tasks based on the raw data before DSP. This selection process should respond actively to different tasks and lead to negligible overhead. In the multimodal computing phase, the computing system should be robust to different task scenarios with dynamically varying numbers of modality inputs.

## III. The Design of *SMG*

### A. Overview

In this paper, we propose *SMG*, a system-level modality gating facility that enables highly energy-efficient AES. We present *SMG* with several optimizations at both the application and architecture layers to enhance the existing multimodal AES. By carefully controlling the sensing and computing phases for every data sample, *SMG* can improve the energy efficiency of AES while maintaining the performance benefits of MMC. Figure 6 provides an overview of *SMG*, which includes the following key features:

- **Software-Defined Gating Architecture**: At the architecture layer, we incorporate a *Software-defined DSP Gating* architecture to support adaptive modality sensing. This architecture consists of a raw data buffer (RDB) that separates the energy-hungry DSP in the modality sensor pipeline from the frontend sensor array. This allows the deactivation of the backends of unnecessary modalities while temporarily reserving the associated raw data to maintain situation awareness and data correctness. It also implements several gating interfaces including a pipeline handler (PH) and a set of RDB operators. These gating interfaces provide upper-level applications with the necessary control over the sensing phase. thus, this architecture design enables partial throttling of the modality sensor pipeline and adaptive modality sensing.
- **Raw Data-Activated MMC Mechanism**: At the application layer, we propose a raw data-activated MMC mechanism to manage the sensing and computing phases for different MMC tasks. This mechanism ensures optimal performance while reducing the sensing and computing energy of unimportant modalities for different tasks. It comprises two key modules: ① *Fast Modality Tester*: We implement a fast modality tester to select an appropriate number of modalities for different MMC tasks based on raw data collected from sensors. It utilizes a lightweight policy network that contains feature analysis layers and fully-connected (FC) layers to pre-analyze raw data from different modalities. This enables the selection of appropriate modalities for different MMC tasks to achieve accuracy while suspending the DSP of the unselected modalities. ② *Adaptive Modality Executor*: We also design an adaptive modality executor to support elastic and adaptive multimodal computing. This executor can elastically control and activate feature extractors that adapt to a varying number of selected modalities. It is also responsible for making the final decisions.

With a focus on practicality and ease of integration, we implement *SMG* as a system-level facility that does not require any additional hardware modifications as shown in Figure 7. This allows for seamless integration into existing AES, providing a cost-effective solution for enhancing their energy efficiency. To demonstrate the versatility of *SMG*, we deploy and implement an instance of the system based on an AES application for multimodal video analysis. We show that the implementation is straightforward and that *SMG* can be easily applied to various AES applications.

### B. Software-Defined Gating Architecture

We propose a software-defined DSP gating architecture. Figure 8 illustrates the differences between our proposed architecture and traditional architecture. As shown in the figure, in the traditional architecture, the sensor data interface receives raw data from the sensor and transfers it directly to the DSP

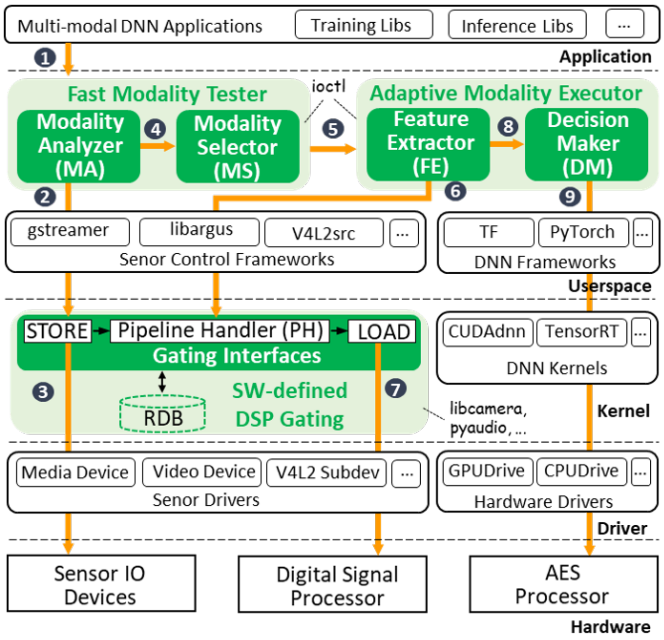



Figure 7: The hardware-software stack of AES with *SMG*.

for preprocessing via the bus. Then, the DSP preprocesses the raw data and produces high-quality data that is finally stored in the modality data buffer. This process is holistic and cannot be interrupted by higher-level applications. Instead, the application can ultimately decide and control whether to read the modality data from the buffer and execute it. In this architecture, all modality data is preprocessed into high-quality data, which is then selected and processed by the applications. For some unimportant modality data, the energy and latency of preprocessing are undoubtedly wasted.

In our proposed architecture, we choose to save the raw modality data before the DSP instead of saving the high-quality modality data after the DSP. This design enables AES to provide flexible control over the sensing phases of different modalities, thus improving energy efficiency while maintaining situational awareness of all modalities. As shown in Figure 7, we design the software-defined DSP gating architecture with two key modules and deploy it into the existing kernel with system-level interfaces such as `libcamera` [17] and `pyaudio` [28]. Firstly, we introduce a raw data buffer before the DSP to temporarily store the raw data of each modality, allowing upper-level applications to decide whether to perform the DSP. Secondly, we implement a set of gating interfaces that include a pipeline handler module and some RDB operators. These interfaces provide the necessary functionality for upper-level applications to control the DSP. Notably, both the RDB and the gating interface are implemented as system functions at the kernel layer, ensuring efficient and seamless integration within the overall system architecture.

We implement a set of raw data buffer functions that enable users to define and control their raw data buffers for different modalities. Among them, the `creatRDB(...)` function is the most critical function, which is used to define and create the raw data buffer with user-defined size. By default, the

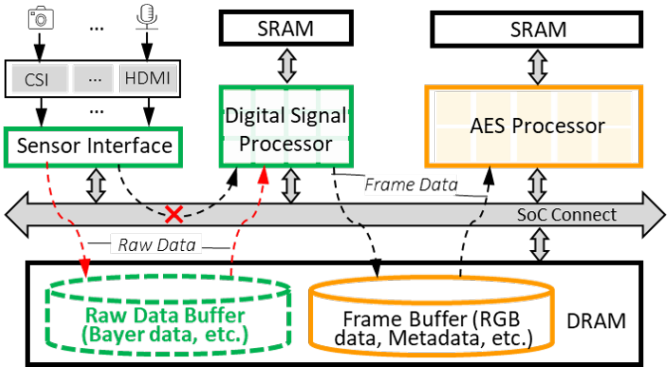



Figure 8: The DSP gating architecture.

function supports a maximum raw data buffer size of 20 MB in its function stack, which can store raw data for at least 10 modalities. The available raw data buffer size for each modality is set to 2 MB, which is sufficient to store the raw data of a color image with a resolution of 1920*1080 pixels. This is suitable for most modalities in real-world applications since image data is often larger than the data of other modalities. In the `creatRDB(...)` function, we call the existing system interface `alloc` to allocate buffer addresses in DRAM. To make efficient use of memory space, we recommend that users define their own raw data buffer size based on their application requirements. Furthermore, we also realize a series of RDB operators such as `storeRDB()` and `loadRDB()` to support other functions to store and load raw data from the buffer. These interface functions provide users with a convenient way to handle the raw data buffer.

The PH of the gating interfaces are several functions to manage the operation modes of different hardware and software components as well as control the data flow in AES. Its most prominent function is `switchSMGMode(...)`, in which we define two modality gating modes, including suspending the DSP (`S`) and resuming the DSP (`R`). Among them, the `S` mode means that the raw data of the modality is stored in the raw data buffer and temporarily bypasses the DSP to save energy, while the `R` mode means resuming the raw data of the modality from the raw data buffer and activating the DSP for preprocessing. By setting the parameters of the `switchSMGMode(...)` function, the user can control the energy-saving modes in the modality sensing phase. In addition, the PH provides several interfacing functions to manipulate the data flow among different hardware components. For example, it leverages the `graspRAW()` function to read raw data from the sensors.

### C. Raw Data-Activated MMC Mechanism

After developing the architecture and system support for the modality gating mechanism, the next step is to determine how to execute situation-aware MMC tasks. This involves allowing the upper layer application to choose the appropriate modalities for the given task, pre-processing the associated raw data using DSP, and analyzing and fusing features from the selected modalities to make accurate decisions for the task while reducing the DSP and computing energy consumption

of the overall AES application. While previous research has proposed situation-aware multimodal computing optimization methods, they cannot be directly applied to the modality gating scenario as they mainly rely on high-fidelity modality data after DSP. In this study, we propose implementing situation-aware MMC tasks based on pre-analysis of the raw data before DSP, which has not been previously explored. This approach will allow for more energy-efficient MMC applications by bypassing the DSP of unimportant modalities for each task.

To achieve this goal, we propose a raw data-activated MMC mechanism at the application layer as shown in Figure 7 and we deploy it into the existing kernel with system-level interface `ioctl` [29]. This mechanism consists of two key modules: a fast modality tester and an adaptive modality executor. The fast modality tester module analyzes raw modality data before DSP for each MMC task and selects the most important modalities to achieve optimal task performance. This approach allows for the efficient use of DSP and computing resources by prioritizing the most relevant modalities. Meanwhile, the adaptive modality executor module is designed to dynamically adapt to varying numbers of appropriate modalities and handle unexpected cases of performance penalty issues. This ensures that the system is able to perform effectively in a variety of situations and can handle any unexpected challenges that may arise.

In real-world AES applications, the appropriate modalities can change stochastically with the tasks, making it difficult to select modalities using empirical methods due to their inefficiency in dynamic scenarios. To address this challenge, we propose a lightly fast modality tester to dynamically select appropriate modalities for different MMC tasks, as shown in Figure 7. The fast modality tester consists of two key components. Firstly, a set of modality analyzers (`MA`) is used to quickly analyze the raw data of different modalities and extract elementary modality features. Then, the extracted elementary features are fed into the modality selector (`MS`). It consists of a single modality fusion layer to federate the elementary features and decides whether to preprocess or suspend an input modality. It should be noted that in the fast modality tester, we leverage shadow modality feature extraction networks to implement the `MA` module according to the features of the modality or the original encoders in the MMDNNs. For example, for the image modality, we can use `MobileNet` [30] to conduct the pre-analysis. Meanwhile, we use simple fusion operators, such as the `concatenate` [31] to implement the single modality fusion layer, and fully-connected layers to implement the `MS` module. Therefore, the fast modality tester is lightweight and will not lead to large overheads. Through the utilization of this approach, the system is capable of attaining more precise and adaptable modalities selection for diverse MMC tasks, thereby enhancing the overall performance of the system.

The adaptive modality executor is responsible for computing the data of the modalities selected by the fast modality tester and obtaining the prediction results that satisfy the application performance. Like the original MMDNNs, it contains a feature extractor (`FE`) and a decision maker (`DM`), the former mainly using the feature extraction networks to extract the modality

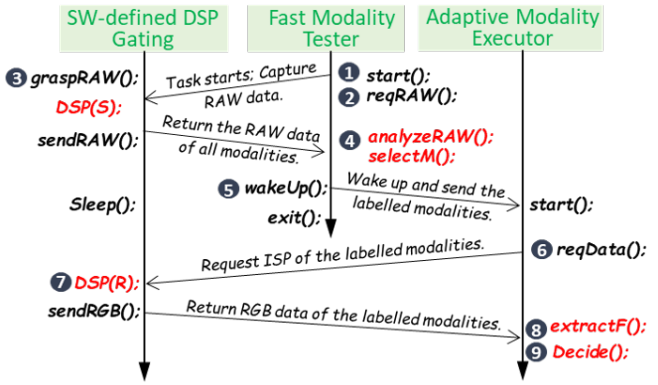



Figure 9: Function calls between the three modules.

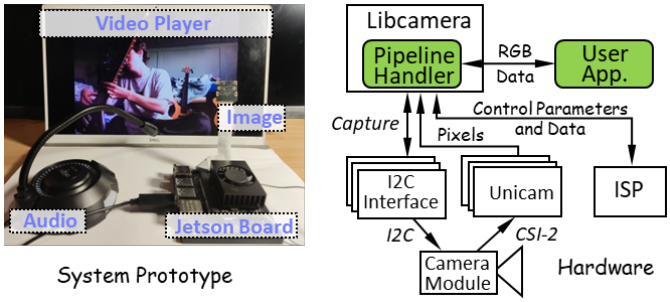



Figure 10: Prototype platform and system integration of *SMG*.

features and the latter mainly using the decision network to execute the task results. However, it differs from the original MMDNN in two aspects in order to perform situation-aware multimodal computing. Firstly, it supports dynamically turning on the feature extraction networks of the selected modalities and turning off the ones of the unselected modalities. Secondly, it can dynamically fuse the features of different numbers of modalities and handle unexpected cases of performance penalty issues. We show the technical details of the adaptive modality executor in Figure 6. In Figures 7 and 9, we show the complete flow of an MMC task under the proposed *SMG* approach: the task ❶ starts, and it ❷ calls the `reqRAW(...)` function to ❸ collect the raw data of the task modalities, which is sent to the fast modality tester to ❹ select the appropriate modalities; the fast modality tester ❺ sends the selected modality information to the adaptive modality executor and exits; after that, the adaptive modality executor ❻ requests the DSP for the selected modalities and ❼ obtain the preprocessed modality data from the software-defined gating module; finally, the adaptive modality executor ❽ computes the dynamic multimodal DNNs on the AES processor and ❾ generates the final prediction results.

### *D. A Case Study: SMG for Multimodal Video Processing*

Our proposed *SMG* approach is a versatile, hardware-independent tool at the system level that can be easily deployed on a variety of AES platforms. Due to constraints on page length, here we use multimodal video processing applications as a case study to demonstrate the effectiveness of *SMG*. We choose video processing because it is one of the most common

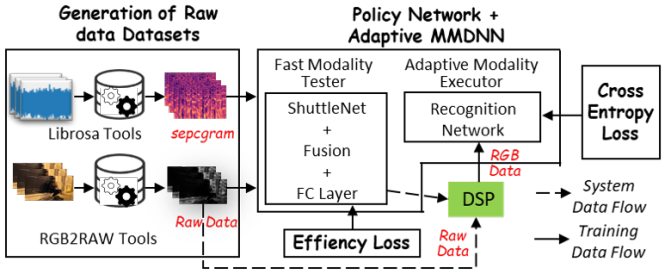



Figure 11: Training platform for *SMG*.

multimodal computing tasks in real-world scenarios [8], [32]–[34], involving image and audio modalities. Video processing is widely used in numerous AES applications, such as online translation robots and in-vehicle entertainment systems [24]. This technology facilitates real-time analysis of information obtained from images and speech.

To validate our approach, we establish an AES platform, as illustrated in Figure 10, and utilize the most popular commercial AES board, `Jetson Xavier NX` [14], to process different video processing tasks. `Jetson Xavier NX` is widely used in edge AES scenarios, from AIoT to embedded platforms, and we leverage the system-level interfaces such as `libcamera` [17], [18] to deploy *SMG* on the `linux4tegra` system, optimizing the energy efficiency of various video processing applications. The video processing system we have established uses a combination of hardware and software components to enable real-time processing of video images and audio data. The camera and microphone capture data in real-time and transmit it to the Jetson onboard system through the `MIPI CSI` [14] interface on the board. The software-defined gating module receives the raw data of the modalities and stores it in the raw data buffer while also forwarding it to the application layer for selection. The fast modality tester in the application layer selects the best modality for each segment of the video based on input conditions, such as lighting conditions, motion, and audio quality. The selected modalities are sent back to the gating module, which calls the DSP to preprocess the raw data of the selected modalities and transmits the preprocessed modality data to the application layer for computing. Through the use of *SMG*, we are able to achieve high performance and low power consumption in video processing.

In the fast modality tester, we use `ShuffleNetV2` [35] to implement the image and audio modality analyzers. To enhance the dynamics of the multimodal computing network and enable it to adapt to varying numbers of modalities, we integrate `ResNet50` [36] and `MobileNetV2` [30] into the original video processing MMDNNs. Figure 11 depicts the training framework used to train the proposed fast modality tester and adaptive modality executor. The training process involves two key factors. The first factor is the training and testing datasets of raw data samples. However, there are currently no raw data-based datasets for video processing. To overcome this, we use conversion tools, such as `CycleISP` [37], from prior works to transform the original video datasets into raw data-based datasets. It is worth noting that previous research has established

**Algorithm 1:** SMG Training Algorithm

**Input:** Multimodal raw dataset $RD$, fast modality tester $PN$, adaptive modality executor $MMC$, modality number $M$, segment number $T$.
**Output:** $PN$'s weights and $MMC$'s weights

```
1 Initialize MMC with {θ_m}_{m=1}^M.
2 for epoch=1 to N do
3     for multi modal raw data {rd_m}_{m=1}^M in RD do
4         Get PN predictions as s = PN({rd_m}_{m=1}^M, T).
5         Use DSP to process selected modalities as pred.
6         Get MMC prediction with selected modalities as
            MMC(prd, T, s).
7         Calculate the loss with equation (1).
8         Backpropagate and update the parameters.
9 return PN, MMC.
```

raw data-based datasets, thus, we don't worry about the lack of available training and testing datasets. Data collection is not the focus of this paper, thus, we utilize existing tools to directly convert existing datasets. The second factor is an appropriate loss function. In our approach, we train the fast modality tester and the adaptive modality executor simultaneously. Therefore, we define the loss function using the efficiency loss of the fast modality tester and the performance loss of the adaptive modality executor. We add the loss of both parts to get the final loss depicted as in equation (1).

$$\mathcal{L} = -\sum y \log(p) + \sum_{m=1}^{M} w_m (\frac{1}{T} \sum_{t=1}^{T} s_{t_m})^2 C + \gamma(1-C) \quad (1)$$

where $y$ is the one-hot encoded ground truth label and $p$ represents predicted probability. $M$ represents the number of modalities and $w_m$ represents the weight of each modality. $T$ represents the segment number of the video we split and $s_{t_m}$ is the output (0 or 1) of the fast modality tester representing whether to select the $m$-th modality of the $t$-th segment in the video. $C$ represents the indicator of the correctness of the prediction and $\gamma$ is the penalty when the prediction is not correct ($C = 0$). However, the discrete nature of the fast modality tester's decisions makes the algorithm non-differentiable. To address this issue, we adopt the Gumbel-Softmax trick [38] to train the fast modality tester. The training process is outlined in Algorithm 1. Firstly, training samples are obtained from the converted raw data training set (`Line 3`). Next, each training sample is inputted into the fast modality tester (`Line 4`). Subsequently, DSP is performed on the modality data selected by the fast modality tester (`Line 5`). The adaptive modality executor is then leveraged to compute the modality data (`Line 6`). Further, the loss is computed and parameters are updated accordingly (`Lines 7,8`). The aforementioned steps are repeated until the network fits the data well.

## IV. Experimental Methodologies

### A. Multimodal Video Processing Workloads

We use two well-known datasets namely `Kinetics-Sounds` [39] and `UCF101` [26] that consist of extensive video recognition workloads to train and validate

*SMG*. Among them, Kinetics-Sounds is a subset of Kinetics [40] and comprises 28,268 videos for training and 1,512 videos for testing, across 31 action classes. UCF101 is an action recognition dataset of realistic action videos collected from YouTube, containing 101 action categories, of which we use 51 action categories in our study as we can only extract audio from those categories. The 51 categories have a total of 6,837 videos, and the authors provide three ways to split the dataset. We select the first method, which uses 4,941 videos for training and 1,896 videos for testing. To present the results more clearly, we manually split the Kinetics-Sounds test set into 15 workload groups and split the UCF101 test set into 11 workload groups. Each group consists of several video recognition tasks with similar features, allowing for a comprehensive evaluation of the performance of *SMG* in various task scenarios.

### B. Prototype Platform Setup

**Warm-up Settings:** Before deploying a system prototype, we first develop well-trained video recognition models. To achieve this, we initially train the image encoder network, i.e., ResNet50, and the audio encoder network, i.e., MobileNetV2, for 60 epochs. We use the weights of the two pre-trained modality encoders to initialize the adaptive modality executor and "warm it up" for 5 epochs while fixing the fast modality tester. Subsequently, we alternately train both the fast modality tester and adaptive modality executor for 20 epochs. Finally, we fine-tune the adaptive modality executor for 10 epochs while fixing the fast modality tester. During training, each video is segmented into 5 segments ($T$ in Equation 1), and the penalty is set to be 10 ($\gamma$ in Equation 1). Furthermore, we respectively use the Adam and SGD methods from AdaMML [8] to train the fast modality tester and adaptive modality executor.

**Prototype Platform Configurations:** As shown in Figure 10, we implement a prototype platform for *SMG* to validate its effectiveness. We conduct experiments on NVIDIA's Jetson Xavier NX board, which is a widely-used AES board with rich sensor interfaces. We utilize a Laptop as our primary device for video playback and we use Sony's IMX219-77 camera and aigo K2 USB microphone to collect the raw data of the image and audio modalities respectively. We collect raw Bayer images with a resolution of $1920 \times 1080$ at 30fps. The Bayer images are converted to RGB images using fast-openISP [41] based on the decisions of *SMG* or other methods. Audio signals are captured at a sample rate of 20KHz, and corresponding spectrograms are extracted using the Librosa tool [42]. We utilize software tools integrated into the existing system to obtain the desired results, such as using the INA3221 power monitor at I2C address 0x40 [43] and python time module [44] to obtain the system power and task execution latency respectively.

### C. The State-of-the-Art Baselines

We compare *SMG* with the SOTA energy-efficient approaches for AES, as shown in Table I. Among them, OriginMMC represents the traditional MMC method without any optimization [8]. LiteEval [45] is one of the most representative efforts [45], [47]–[49] to reduce the computing overhead of edge AI by skipping image frames. In contrast to LiteEval, which reduces the modality data, OrderedMMC and AdaMMC decrease the multimodal computing overheads by reducing the number of modalities. OrderedMMC represents the traditional heuristic-driven modality selection method [46], while AdaMMC is a more advanced method that uses machine learning [8]. To cover all traditional methods that use different techniques such as quantization and pruning to optimize the computation phases of MMC, we implement IdealMMC, which simulates an ideal but impossible scenario where computing consumption is optimized to zero, with only sensor overhead considered. The key limitation of the above methods is that they all overlook the overheads of the modality sensing phase while only optimizing the inference phase. *SMG* is the first approach to utilize sensor gating for energy-efficient edge MMC. We also develop LargeSMG, an SMG-derived approach to validate *SMG*. It utilizes a larger backbone (MobileNetV2) to implement the fast modality tester, while *SMG* uses a smaller backbone (ShuffleNetV2) to do this.

Table I: The compared state-of-the-art baselines.

| Scheme | Description |
|---|---|
| OriginMMC [8] | Traditional MMC method that fuses all modalities directly. |
| LiteEval [45] | SOTA situation-aware method that skip unimportant frames. |
| OrderMMC [46] | SOTA method using heuristic-driven modality selection. |
| AdaMMC [8] | SOTA method that only optimizes the computing phase. |
| IdealMMC | Simulating an ideal scenario where computing overhead is 0. |
| LargeSMG | Selecting modalities with a larger fast modality tester. |
| *SMG* | Our method with a lightweight fast modality tester. |

Table II: The performance effect of *SMG* on the two datasets.

| | Methods | | AudioUni | ImageUni | MultiMMC | SMG |
|---|---|---|---|---|---|---|
| Kinetics-Sounds | Selection Ration(%) | None | 0 | 0 | 0 | 8.21 |
| | | Audio | 100 | 0 | 0 | 46.34 |
| | | Image | 0 | 100 | 0 | 7.07 |
| | | Both | 0 | 0 | 100 | 38.38 |
| | Energy(mJ) | Sense | 2663.04 | 8946.61 | 9368.86 | 6421.78 (**-31.4%**) |
| | | Compute | 580.36 | 7935.45 | 9165.51 | 3184.99 (**-65.3%**) |
| | Latency(s) | Sense | 2.67 | 5.23 | 5.25 | 3.79 (**-27.8%**) |
| | | Compute | 0.29 | 1.34 | 1.44 | 0.74 (**-48.6%**) |
| | Accuracy(%) | | 52.82 | 61.29 | 84.1 | 82.47 |
| UCF101 | Selection Ration(%) | None | 0 | 0 | 0 | 19.57 |
| | | Audio | 100 | 0 | 0 | 19.57 |
| | | Image | 0 | 100 | 0 | 26.06 |
| | | Both | 0 | 0 | 100 | 31.86 |
| | Energy (mJ) | Sense | 2521.86 | 9026.61 | 9301.51 | 7379.89 (**-20.7%**) |
| | | Compute | 639.81 | 8099.81 | 9120.45 | 4271.55 (**-53.2%**) |
| | Latency(s) | Sense | 2.67 | 5.24 | 5.32 | 4.29 (**-19.4%**) |
| | | Compute | 0.28 | 1.35 | 1.44 | 0.92 (**-36.1%**) |
| | Accuracy(%) | | 30.33 | 83.91 | 87.76 | 87.76 |

## V. Evaluation Results

### A. Effectiveness Validation

**Demonstration of Situation-Aware Modality Gating:** We examine the ability *SMG* to perform situation-aware modality gating for various MMC task scenarios. Figure 12 provides a demonstration of four different MMC task scenarios in the playing instruments workload in the Kinetics-Sounds dataset. In Scenario 1, no information from either the image or audio modalities is required since no instruments are being played. Scenario 2 requires analysis of the image information because the sound features are not obvious. In Scenario 3, the audio modality alone is relied upon to determine the category of the instrument being played. Scenario 4 is a more complex task that requires both

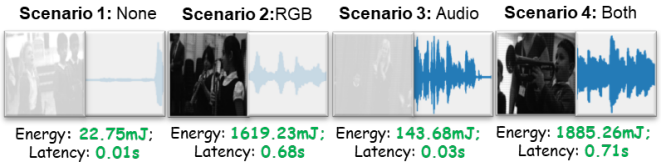



Figure 12: Demonstrating *SMG* 's ability in situation-aware modality sensing and computing for various MMC tasks.

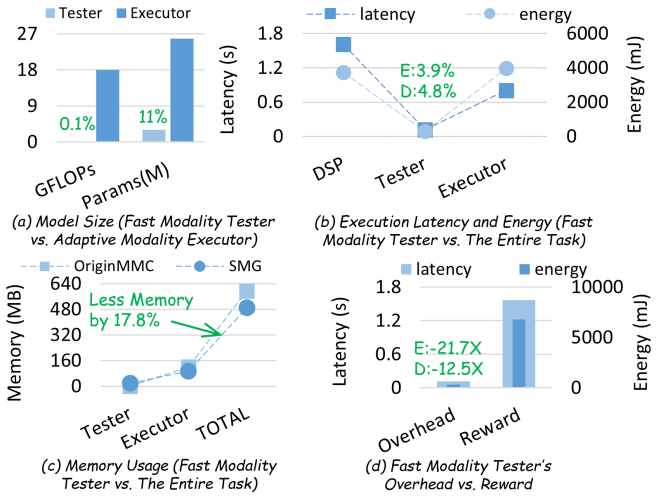



Figure 13: Overheads vs. benefits of the fast modality tester. TOTAL means the memory usage of the whole end-to-end process; Reward means the energy and latency reduction of SMG compared to the original MMC model.

sound and image analysis to make accurate judgments. We observe that *SMG* adapts well to different task scenarios to preprocess and compute the appropriate modalities, thus reducing task processing delay and energy consumption without any degradation in accuracy.

**Performance Evaluation:** Table II presents the performance of *SMG* in optimizing the accuracy, execution delay, and energy consumption of traditional uni-modal (`AudioUni` & `ImageUni`) and (`MultiMMC`) AES applications. We analyze the modality selection ratio and the optimization of associated metrics for the evaluated datasets. Our results show that *SMG* achieves significant reductions in energy consumption of the modality sensing and computing phases by 31.4% and 65.3% under the `Kinetics-Sounds` dataset, while reducing the modality sensing and computing delay by 27.8% and 48.6%, respectively. Similar performance improvements are observed for the `UCF101` dataset. These findings illustrate that *SMG* can effectively select different modalities for various task scenarios, perform adaptive modality gating and computing mechanisms, and improve the overall performance of the applications.

**Overhead Analysis:** To evaluate the effectiveness of *SMG*, it is crucial to ensure that its benefits outweigh the costs incurred by its overhead. The design of *SMG* may introduce two types of overhead: the overhead of the raw data buffer (RDB) and the overhead of the fast modality tester. As the size of the raw data is only one third of the frame data size, it is intuitive that *SMG* results in a reduction of the AES buffer size for storing modality data. In this context, we focus on evaluating the overheads and benefits of the fast modality tester, as shown in Figure 13. Our results illustrate that the fast modality tester increases the floating point operations per second (FLOPs) and parameter size of the MMC model lightly (Figure 13-(a)). Nevertheless, this overhead contributes no more than 5% of the task execution delay and energy consumption compared to the entire task (Figure 13-(b)) while it reduces the task energy consumption and execution latency by a factor of 21.7 and 12.5 respectively (Figure 13-(d)). Furthermore, *SMG* effectively reduces the memory usage overhead by 17.8% compared to original MMC model ( Figure 13-(c)).These findings demonstrate that *SMG* can effectively improve the energy efficiency and reduce the latency of MMC tasks with negligible overhead.

### *B. Comparison with SOTA*

We then compare the accuracy, task execution latency, and energy consumption of *SMG* with the state-of-the-art (SOTA) energy-efficient AES methods in different scenarios.

**Accuracy:** It is important that the energy-efficient design does not compromise the accuracy benefits of MMC. Therefore, we compare the accuracy of different workloads in the two evaluated datasets under *SMG* and the SOTA approaches in Figure 14. We consider the average accuracy across all methods as the baseline value for fair comparison. This approach is appropriate since *SMG* primarily aims to optimize the system and architecture, rather than tuning parameters for each MMC task model individually. As shown in the figures, *SMG* achieves higher accuracy than the `Baseline` for all the workloads. Although there are a few instances where the accuracy under *SMG* is lower than that of certain individual methods, such as `LiteEval` exhibiting higher accuracy than *SMG* on a few specific workloads like `guitar` and `stomp`, the differences observed are minimal and inconsistent. Moreover, our following analysis results from Figure 15 to Figure 16 show that *SMG* achieves better execution performance and efficiency than other methods including `LiteVal`. Furthermore, it is worth mentioning that the accuracy of *SMG* can be improved through meticulous algorithmic optimization, such as extensive parameter tuning. However, in the scope of this paper, we have primarily focused on enhancing the system and architecture, and algorithmic optimizations have not been extensively explored. Future work could delve into these algorithmic optimizations to maximize the accuracy potential of *SMG*.

**Execution Latency:** An important advantage of *SMG* is its ability to reduce the latency of the modality sensing phase, which decreases the execution latency of the entire MMC task. Figure 15 displays the average processing latency of an MMC task in the three main components - the sensor array, the DSP, and inference (including all networks' inference; for instance, the inference of fast modality tester and adaptive modality executor in *SMG* ) - under *SMG* and the SOTA methods using the `UCF101` dataset. To facilitate a clear comparison, we normalize the latency reduction of different methods relative to `OriginMMC`. The results show that traditional schemes


(a) Accuracy of different workloads in the `Kinetics-Sounds`




(b) Accuracy of different workloads in the `UCF101`

Figure 14: Comparison of accuracy for *SMG* and the SOTA approaches.

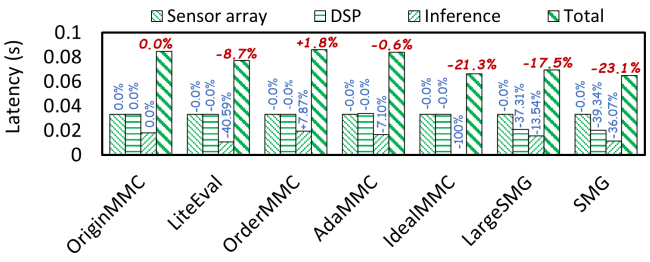



Figure 15: Latency comparison under the `UCF101` dataset.

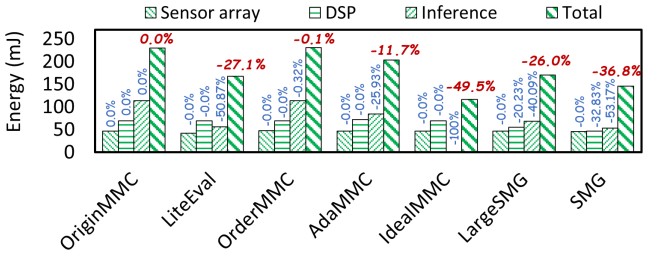



Figure 16: Energy comparison under the `UCF101` dataset.

exhibit zero, i.e., 0.0% reduction in DSP latency due to their neglect of the sensing phase. In contrast, *SMG* excels in optimizing both computing and sensing latency, resulting in a substantially lower average end-to-end task latency compared to all other schemes. Impressively, *SMG* achieves a remarkable reduction of up to 23.1% compared to other methods. While it is worth mentioning that `LiteEval` exhibits a slightly greater reduction in inference latency than *SMG* due to its utilization of fewer image frames as inputs, our method still achieves significantly greater latency reduction in DSP and total task latency. Additionally, it is important to note that the latency of the sensor array remains the same for different scenarios, as all scenarios require the collection of raw data to ensure that the task is fully informed about the environment. Overall, *SMG* effectively reduces the average modality inference latency per task by up to 36.07% and the overall latency by up to 23.1% compared to the existing schemes.

**Energy Consumption:** Figure 16 presents a detailed analysis of how the modality gating technique reduces task-level energy consumption. We compare energy consumption at different stages of the task under various schemes using the `UCF101` dataset. It is important to note that the measurements solely consider dynamic execution energy consumption, as idle energy is not accounted for due to the unavailability of appropriate measurement tools. Nevertheless, it is reasonable to assume that *SMG* consumes less idle energy compared to other methods due to its shorter sensing latency, as demonstrated in Figure 15. In Figure 16, the energy results are consistent with the latency results, with one exception: `IdealMMC` consumes less total energy than *SMG*, as it assumes an ideal but impossible state where computing consumption is optimized to zero and only sensor overhead is considered. Nevertheless, our findings reveal that *SMG* consumes less DSP energy than `IdealMMC`. Furthermore, *SMG* can respectively achieve 32.81%, 2.3% and 9.7% more energy consumption reduction in DSP, inference and total end-to-end task compared with `LiteEval`. Overall, *SMG* reduces the average sensing energy consumption of the task by up to 32.83%, the inference energy consumption by up to 53.17%, and the overall energy consumption by up to 36.8% except for `IdealMMC`.

### C. Real-world Verification

All the experiments conducted thus far have been carried out with an adequate energy budget. However, it is important to validate the benefits of *SMG* in real-world AES scenarios where energy budgets are often constrained. To validate this, we conduct experiments using three different battery capacities commonly found in AES applications: the `Microsoft Band Battery` (200 mAh, equivalent to 2520 Joules) [50], the `Google Glass Battery` (600 mAh, equivalent to 7560 Joules) [51], and a `20%-Cell Phone Battery` (3000 mAh, equivalent to 37,800 Joules) [52]. Due to the lack of programmability in the target systems of these commercial platforms, we are unable to directly deploy *SMG* onto them. Instead, we connect batteries with the aforementioned capacities to our experimental platform. This measurement gives an impression of the relative power consumption, even though the power distribution in AES target systems will somewhat deviate from `Jetson Xavier NX`. In the results, we compare the battery depletion time, throughput (i.e., the number of processed frames), frame rate (i.e., frames per second or FPS), and energy per frame (EPS) for the AES system built in our experiment

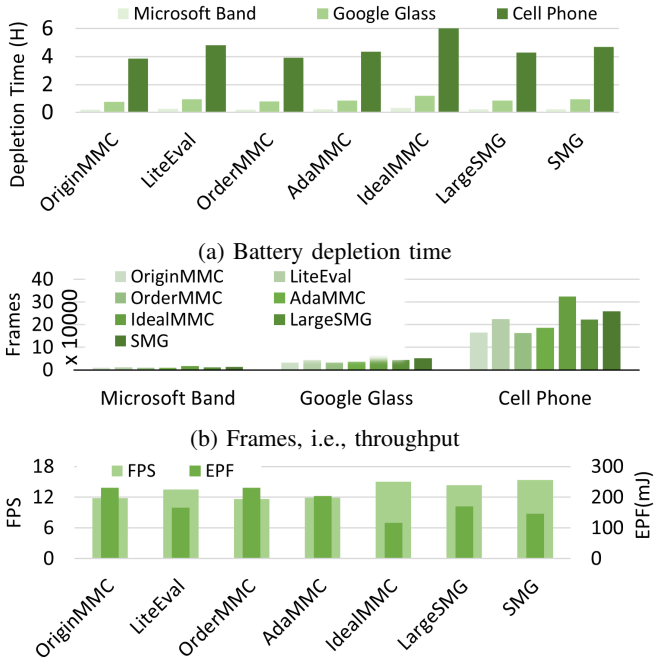



Figure 17: Comparison of the performance for *SMG* and the SOTA approaches in three real-world AES energy budgets.

under *SMG* and other schemes using the `UCF101` dataset. Our results demonstrate that *SMG* significantly reduces energy consumption and extends battery depletion time by 20% to 22% across different batteries (Figure 17-(a)) while increasing system throughput by 57% to 59% (Figure 17-(b)) and frame rate by 3% to 32% (Figure 17-(c)) except for `IdealMMC`. It is worth noting that the results of `IdealMMC` outperform those of *SMG* for it ignores the computation overhead. However, our results show that *SMG* almost approaches the `IdealMMC`'s.

## VI. Related Work

**Multimodal Deep Neural Networks:** Multimodal DNNs [1], [34] have been demonstrated to outperform the unimodal networks in many application fields [4], [34] recently. Most of the current studies focus on finding more effective fusion or representation methods of different modalities [1], [53], which can be categorized into two main classes, namely early fusion methods [31], [54] and late fusion methods [55], [56]. For example, Zhou et al. use a multiple discriminant analysis scheme to implement an early fusion approach that concatenates the features from different modalities [31]. Although multimodal DNN algorithms have been extensively studied, how to apply them for practical AES has not been explored well. In particular, they have excessive modality sensing and processing demand that far exceeds the performance and energy constraints of embedded systems. The objective of our approach is to enable the deployment of multimodal DNNs in production applications through cross-stack optimizations of data structures, algorithms, systems, and architectures.

**Sensor Pipeline Optimizations:** With the increasing importance of sensors, many research works have focused on enhancing the capability of sensor pipelines with more powerful algorithms [9], [57] and specialized accelerators [33], [58], [59]. For example, DeepISP [9] is an end-to-end camera image processing pipeline that leverages DNN model to improve image quality. However, the higher energy consumption of more powerful sensors has become a headache issue, particularly for resource-constrained embedded systems. More recently, energy-aware schemes are used to adaptively control the sensing pipeline. They either reduce data streaming lines [60], [61] or alleviate data computations [57], [62] at a high level. For example, Kodukula et al. propose to only capture the updated pixels with rhythmic pixel regions rather than the new frame or region of interest (ROI), thus reducing the data volume [62]. Recently, few works have focused on streamlining the sensor pipelines themselves by adaptive data pre-processing capabilities [63], [64]. Our approach is orthogonal to the existing works that focus on architecture optimizations for adaptive sensor pipelines.

**Design of Energy-efficient AES:** There is an increasing interest in applying DNN models for practical autonomous embedded systems (AES) to achieve ubiquitous intelligence. Due to the resource constraints of AES, a number of energy-efficient designs have been proposed to better trade-off performance and efficiency in different scenarios. Typically, researchers have studied several DNN compression approaches such as quantization [65], and pruning [66]. These approaches perform adaptive DNN execution by increasing or decreasing the accuracy under various energy constraints. A more general method is conditional computing which progressively computes a complicated DNN model. For example, early exit [67], SkipNet [68], etc., are representative works that dynamically find the optimal trade-offs between computation and performance. There have emerged several adaptive multimodal neural architectures [8], [21] to enable adaptive multimodal learning and fusion for different applications. For example, FrameExit applies conditional early exiting for efficient video recognition based on the complexity of frame context [21]. Inspired by these works, our approach presents a holistic execution engine that facilitates adaptive multimodal sensing and computing at both the system and architecture levels.

## VII. Conclusion

Multimodal computing (MMC) holds significant potential for a wide range of AES applications. In this paper, we propose *SMG*, a novel modality gating technique that enables low-latency and high-efficiency MMC. We integrate *SMG* into the existing AES as a system-level facility. It can pre-analyze task scenarios based on raw data before DSP and adaptively determine the DSP and computing of appropriate modalities for different tasks. *SMG* is a versatile, hardware-independent system-level tool that is applicable to a variety of multimodal computing applications and AES platforms. Our work aims to facilitate the development of future high-performance edge intelligent systems, e.g., edge micro data centers.

## VIII. Acknowledgements

We sincerely thank our shepherd and all the anonymous reviewers for their valuable comments that helped us to improve the paper. This work is supported by the National Natural Science Foundation of China (No. 62122053). The corresponding author is Chao Li.

## References

[1] C. Zhang, Z. Yang, X. He, and L. Deng, "Multimodal intelligence: Representation learning, information fusion, and applications," in *IEEE Journal of Selected Topics in Signal Processing (JSTSP)*, 2020.

[2] S. Krishnan, Z. Wan, K. Bhardwaj, P. N. Whatmough, A. Faust, S. M. Neuman, G.-Y. Wei, D. M. Brooks, and V. J. Reddi, "Automatic domain-specific soc design for autonomous unmanned aerial vehicles," *International Symposium on Microarchitecture (MICRO)*, 2022.

[3] T. Baltrušaitis, C. Ahuja, and L. Morency, "Multimodal machine learning: A survey and taxonomy," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2018.

[4] P. Liang, Y. Lyu, X. Fan, Z. Wu, Y. Cheng, J. Wu, L. Chen, P. Wu, M. Lee, and Y. Zhu, "Multibench: Multiscale benchmarks for multimodal representation learning," *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[5] D. Roy, Y. Li, T. Jian, P. Tian, K. R. Chowdhury, and S. Ioannidis, "Multi-modality sensing and data fusion for multi-vehicle detection," *IEEE Transactions on Multimedia (TOM)*, 2022.

[6] X. Hou, J. Liu, X. Tang, C. Li, K.-T. Cheng, L. Li, and M. Guo, "Mmexit: Enabling fast and efficient multi-modal dnn inference with adaptive network exits," in *European Conference on Parallel Processing (Euro-par)*, 2023.

[7] J. Arevalo, T. Solorio, M. Montes-y Gómez, and F. González, "Gated multimodal units for information fusion," in *International Conference on Learning Representations (ICLR)*, 2017.

[8] R. Panda, C.-F. Chen, Q. Fan, X. Sun, K. Saenko, A. Oliva, and R. S. Feris, "Adamml: Adaptive multi-modal learning for efficient video recognition," *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[9] E. Schwartz, R. Giryes, and A. M. Bronstein, "Deepisp: Toward learning an end-to-end image processing pipeline," *IEEE Transactions on Image Processing (TIP)*, 2018.

[10] R. Panda, C.-F. Chen, Q. Fan, X. Sun, K. Saenko, A. Oliva, and R. Feris, "AdaMML: Adaptive Multi-Modal Learning for Efficient Video Recognition," in *International Conference on Computer Vision (ICCV)*, 2021.

[11] B. Yu, W. Hu, L. Xu, J. Tang, S. Liu, and Y. Zhu, "Building the computing system for autonomous micromobility vehicles: Design constraints and architectural optimizations," *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2020.

[12] R. Likamwa, B. Priyantha, M. Philipose, L. Zhong, and P. Bahl, "Energy characterization and optimization of image sensing toward continuous mobile vision," *International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2013.

[13] C.-T. Chiang, C.-H. Wang, and C.-Y. Wu, "A cmos mems audio transducer implemented by silicon condenser microphone with analog front-end circuits of audio codec," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2012.

[14] Nvidia, "Nvidia jetson nano developer kit." https://developer.nvidia.com/zh-cn/embedded/learn/get-started-jetson-nano-devkit, 2020.

[15] R. P. T. Ltd, "Raspberry pi 4 model b." https://tinyurl.com/w5hyyhf4, 2020.

[16] Gitlab, "Gstreamer: a flexible, fast and multiplatform multimedia framework." https://gstreamer.freedesktop.org/documentation/?gi-language=c, 2023.

[17] Nvidia, "Libargus camera api." https://docs.nvidia.com/jetson/l4t-multimedia/group__LibargusAPI.html, 2023.

[18] L. API, "v4l2src." https://thiblahute.github.io/GStreamer-doc/video4linux2-1.0/v4l2src.html?gi-language=c, 2023.

[19] M. Urbina, T. Acosta, J. Lázaro, A. Astarloa, and U. Bidarte, "Smart sensor: Soc architecture for the industrial internet of things," in *IEEE Internet of Things Journal (IoTJ)*, 2019.

[20] I. Cevik, X. Huang, H. Yu, M. Yan, and S. U. Ay, "An ultra-low power cmos image sensor with on-chip energy harvesting and power management capability," *Sensors*, 2015.

[21] A. Ghodrati, B. E. Bejnordi, and A. Habibian, "Frameexit: Conditional early exiting for efficient video recognition," *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[22] Z. Weng, Z. Wu, H. Li, J. Chen, and Y.-G. Jiang, "Hcms: Hierarchical and conditional modality selection for efficient video recognition," *ACM Transactions on Multimedia Computing, Communications and Applications (TMCCA)*, 2021.

[23] Y. Gan, Y. Bo, B. Tian, L. Xu, W. Hu, S. Liu, Q. Liu, Y. Zhang, J. Tang, and Y. Zhu, "Eudoxus: Characterizing and accelerating localization in autonomous machines industry track paper," *IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2020.

[24] X. Hou, C. Xu, J. Liu, X. Tang, L. Sun, C. Li, and K.-T. Cheng, "Characterizing and understanding end-to-end multi-modal neural networks on gpus," *IEEE Computer Architecture Letters (CAL)*, 2022.

[25] W. C. Sleeman IV, R. Kapoor, and P. Ghosh, "Multimodal classification: Current landscape, taxonomy and future directions," *ACM Computing Surveys*, vol. 55, no. 7, pp. 1–31, 2022.

[26] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[27] X. Hou, J. Liu, X. Tang, C. Li, J. Chen, L. Liang, K.-T. Cheng, and M. Guo, "Architecting efficient multi-modal aiot systems," *International Symposium on Computer Architecture (ISCA)*, 2023.

[28] Python, "Pyaudio," 2023.

[29] T. kernel development community, "Ioctl based interfaces," 2023.

[30] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

[31] Z. Xiaoli and B. Bir, "Feature fusion of side face and gait for video-based human identification," *Pattern Recognition*, 2008.

[32] R. Xu, J. Koo, R. Kumar, P. Bai, S. Mitra, S. Misailovic, and S. Bagchi, "Videochef: Efficient approximation for streaming video processing pipelines," in *USENIX Annual Technical Conference (Usenix ATC)*, 2018.

[33] Z. Song, Z. Yu, N. Jing, and X. Liang, "E2sr: an end-to-end video codec assisted system for super resolution acceleration," *ACM/IEEE Design Automation Conference (DAC)*, 2022.

[34] Z. Sun, P. Sarma, W. Sethares, and Y. Liang, "Learning relationships between text, audio, and video via deep canonical correlation for multimodal language analysis," *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[35] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 116–131, 2018.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[37] S. W. Zamir, A. Arora, S. Khan, M. Hayat, F. S. Khan, M.-H. Yang, and L. Shao, "Cycleisp: Real image restoration via improved data synthesis," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2696–2705, 2020.

[38] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[39] R. Arandjelovic and A. Zisserman, "Look, listen and learn," *CoRR*, vol. abs/1705.08168, 2017.

[40] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, *et al.*, "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.

[41] QiuJueqin, "A faster re-implementation of the openisp project," 2020.

[42] Python, "Librosa: Audio and music processing in python," 2023.

[43] Nvidia, "Nvidia jetson linux developer guide," 2021.

[44] Python, "Python time module," 2023.

[45] Z. Wu, C. Xiong, Y.-G. Jiang, and L. S. Davis, "Liteeval: A coarse-to-fine framework for resource efficient video recognition," *Advances in neural information processing systems*, vol. 32, 2019.

[46] Z. Weng, Z. Wu, H. Li, and Y.-G. Jiang, "Hms: Hierarchical modality selection for efficient video recognition," *arXiv preprint arXiv:2104.09760*, 2021.

[47] Z. Wu, C. Xiong, C.-Y. Ma, R. Socher, and L. S. Davis, "Adaframe: Adaptive frame selection for fast video recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1278–1287, 2019.

[48] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei, "End-to-end learning of action detection from frame glimpses in videos," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2678–2687, 2016.

[49] R. Gao, T.-H. Oh, K. Grauman, and L. Torresani, "Listen to look: Action recognition by previewing audio," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10457–10467, 2020.

[50] Wiki, "Microsoft band," 2023.

[51] Wiki, "Google glass," 2023.

[52] Apple, "Cell phone battery," 2023.

[53] W. Wang, D. Tran, and M. Feiszli, "What makes training multi-modal classification networks hard?," *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[54] N. Natalia, W. Christian, W. Graham, and N. Florian, "Multi-scale deep learning for gesture detection and localization," *European Conference on Computer Vision (ECCV)*, 2014.

[55] Y. Wu, E. Chang, K. Chang, and J. Smith, "Optimal multimodal fusion for multimedia data analysis," *ACM International Conference on Multimedia (ACMMM)*, 2004.

[56] F. Bach, G. Lanckriet, and M. Jordan, "Multiple kernel learning, conic duality, and the smo algorithm," *International Conference on Machine Learning (ICML)*, 2004.

[57] A. Mosleh, A. Sharma, E. Onzon, F. Mannan, N. Robidoux, and F. Heide, "Hardware-in-the-loop end-to-end optimization of camera image processing pipelines," *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[58] M. Buckler, P. Bedoukian, S. Jayasuriya, and A. Sampson, "Eva²: Exploiting temporal redundancy in live computer vision," *International Symposium on Computer Architecture (ISCA)*, 2018.

[59] A. Vasilyev, N. Bhagdikar, A. Pedram, S. Richardson, S. Kvatinsky, and M. Horowitz, "Evaluating programmable architectures for imaging and vision applications," *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2016.

[60] S. Naderiparizi, P. Zhang, M. Philipose, B. Priyantha, J. Liu, and D. Ganesan, "Glimpse: A programmable early-discard camera architecture for continuous mobile vision," *International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2017.

[61] M. Xu, T. Xu, Y. Liu, and F. X. Lin, "Video analytics with zero-streaming cameras," *USENIX Annual Technical Conference (Usenix ATC)*, 2019.

[62] V. Kodukula, A. Shearer, V. Nguyen, S. Lingutla, Y. Liu, and R. Likamwa, "Rhythmic pixel regions: multi-resolution visual sensing system towards high-precision visual computing at low power," *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2021.

[63] X. Fu, T. Ghaffar, J. C. Davis, and D. Lee, "Edgewise: A better stream processing engine for the edge," *USENIX Annual Technical Conference (Usenix ATC)*, 2019.

[64] F. Zhang, L. Yang, S. Zhang, B. He, W. Lu, and X. Du, "Finestream: Fine-grained window-based stream processing on cpu-gpu integrated architectures," *USENIX Annual Technical Conference (Usenix ATC)*, 2020.

[65] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "Haq: Hardware-aware automated quantization with mixed precision," *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[66] Y. Wu, Z. Wang, Z. Jia, Y. Shi, and J. Hu, "Intermittent inference with nonuniformly compressed multi-exit neural network for energy harvesting powered devices," *ACM/IEEE Design Automation Conference (DAC)*, 2020.

[67] S. Teerapittayanon, B. McDanel, and H. T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," *International Conference on Pattern Recognition (ICPR)*, 2016.

[68] X. Wang, F. Yu, Z.-Y. Dou, and J. Gonzalez, "Skipnet: Learning dynamic routing in convolutional networks," *European Conference on Computer Vision (ECCV)*, 2017.