MVSAS: Sematic-Aware Scheduling for Low Latency and High Precision in Wireless Multi-View Application

Xiuxian Guan*, Zekai Sun*, Shengliang Deng*, Shixiong Zhao*,
Tszon Li*, Tianxiang Shen*, Yuexuan Wang*, Rui Wang† and Heming Cui*‡
* *Dept. of Computer Science*
The University of Hong Kong, Hong Kong, China
Email: guanxx@connect.hku.hk, zekaisun@cs.hku.hk, sldeng@cs.hku.hk
† *Dept. of Electrical & Electronic Engineering*
The Southern University of Science & Technology, Shenzhen, China
Email: wangr@sustech.edu.cn

***Abstract*—Multi-view models for various multi-view applications (e.g., pose recognition, facial recognition) achieve higher accuracy when more sensing data (views) from different sensors are flexibly collected via wireless networks and combined into inference input. However, when the view number scales up, the application suffers a long latency to collect all the latest views before inference (vanilla workflow). We observed that collecting all the latest views before inference is unnecessary, because different views are often not equally important and important views have major contribution to the output. In this paper, we present a Multi-View Semantic-Aware Scheduling (MVSAS) system that automatically prioritizes views according to their importance and schedules the early transmission of the important views. We tackled the challenge to infer view importance by analyzing the inference intermediates and extracting the semantics (e.g., number of persons) of each view. Once important views are collected, needless to wait for other less important views, the important views are combined with stale version of less important views as inference input, so as to retain high accuracy while reducing the latency to collect views. Evaluation shows that MVSAS achieved at most 36.9% latency reduction while retaining at most 98.7% accuracy compared to the vanilla workflow.**

***Index Terms*—multi-view learning, scheduling, IoT, low latency**

I. INTRODUCTION

With the fast development of multi-view learning [8], [10], [12] and wireless technologies [2], [6], wireless multi-view applications are becoming increasingly pervasive and important in real life recognition problems, such as pose recognition [3], [9], facial recognition [3], video surveillance [7], etc. In the real world, data for these applications are often collected from different sensors (each referred to as a different view), and they contain semantics (i.e., information extracted from views) of the application from different perspectives. Multi-view applications extract the shared semantics across different views for better performance (e.g., higher accuracy of recognition). When the views can be gathered via wireless networks, the multi-view applications can flexibly leverage data from even more diverse views and further extend the performance gain.

We summarize a typical vanilla workflow of wireless multi-view applications as follows. A device (host device) hosting a multi-view model and multiple sensors are communicating with each other via a shared wireless channel. Periodically, each sensor captures data for the application and transmits the data to the host device via the wireless channel. In order to utilize up-to-date and complete sensor data, a synchronization barrier among views is set and requires the application to wait until all views from the same period are gathered. Then all views from the same period are combined and fed to the multi-view model to produce output.

However, such a workflow for wireless multi-view applications suffers from a scalability issue: with the increasing number of sensors, their communication volume increases linearly, while the network capacity is limited. On the one hand, each view takes longer to be transmitted because sensors have to compete with each other for access to the wireless channel. Contention among sensors for the channel gets more intensive and the time cost for sensors to acquire the channel (access cost) increases. On the other hand, the existing workflow requires the application to wait for a larger number of views to be received.

We refer to the time between the start of a period and the production of the corresponding output as *output latency* and the aforementioned two effects severely deteriorate the output latency of a wireless multi-view application (e.g., when view number scales from 3 to 10 in our evaluation VI, the time for inference did not increase evidently, but the output latency of the vanilla workflow scales 6.5x, from 0.79s to 5.13s)

Considering the semantics of multi-view applications, we observe that not all views are equally important. When certain views are gathered (e.g., views containing most details of a certain objective, referred to as important views), the multi-view model is able to extract the key information for the application that can be used for early prediction. On the

‡Corresponding Author

contrary, less important views (e.g., views that fail to capture the objective) have minor contributions to the application (also reported in similar areas in [1], [11]). It is notable that less important views are very common in the real world captured multi-view data, because not all sensors can always keep track of the mobile objective, and there are often sensors capturing more details than others.

Our another observation is that for real-world multi-view applications, wireless sensors are collecting data continuously. The importance of incoming views is closely related to views received in previous periods and can be predicted based on analyzing the received views.

In this paper, we present Multi-View Semantic-Aware Scheduling (MVSAS), a scheduling system for wireless multi-view applications to reduce output latency and improve scalability. MVSAS resides both in the host device and sensors and improves multi-view applications at two layers, the view transmission layer and the application layer. At the view transmission layer, based on the buffered views of previous periods on the host device, MVSAS predicts the future importance of each view and schedules their transmission so that important views can be received earlier. At the application layer, once all latest important views are gathered, needless to wait for other less important views, MVSAS completes all the view input with the latest important views and the stale version of less important views, and then feeds them to the multi-view model to produce output earlier.

To this end, the key challenge for MVSAS would be how to infer the priority of each view. The importance of each view highly depends on the semantics of view data, which is difficult to infer through heuristic methods, and the importance of each view will vary as the application progresses. After researching various multi-view applications, we find that inference intermediates of a multi-view model contain the semantics of each view. For example, MVPOSE [3], a multi-view pose recognition model published in 2019 CVPR, first extracts 2D poses of each view and then correlates the extracted poses to each actor to produce 3D pose recognition result. The number of extracted poses of each view implies its contribution to the final output. We develop specified heuristics to leverage such intermediates to reflect view importance without interfering the inference process.

Specifically, how to configure a proper number of high priority views is also challenging, which is a dominant parameter about the performance of MVSAS. An inadequate number may cause the multi-view application to miss urgent and important information about the application, influencing the accuracy of the output, and an unnecessarily large one one will deteriorate the output latency. Also, we believe less important views that are generated too long time ago will interfere the inference accuracy. We empirically inspect the relationship between the importance distribution of all views and the staleness (i.e., number of versions behind the latest views) of views, and develop an scheduling algorithm that dynamically adjusts view importance and the number of high priority views, so as to balance between accuracy and output latency.

We implemented MVSAS with ROS1-kinetic [] on real Wi-Fi and we chose Voxelpose [9], a powerful multi-view 3D pose recognition model as our main evaluation workload, and Panoptic [5], a high-resolution real-world captured multi-view video dataset, as the main dataset. We compared MVSAS with raw Wi-Fi that uses the vanilla workflow of multi-view application and a strawman approach that has fixed importance under the same pre-trained Voxelpose model. The vanilla workflow ensures that views in each inference are all latest and from the same period (0 staleness), and thus is expected to achieve the highest accuracy. Evaluation shows that

- When the view number increased from 3 to 10, MVSAS achieved an average 30.5% output latency reduction compared with raw WiFi while maintaining average 88.5% accuracy of raw WiFi. MVSAS at most achieved 36.9% output latency reduction and 98.7% accuracy.
- MVSAS had zero influence on the time for model inference.
- MVSAS retained a low level of overall staleness (average 0.224 version behind, comparable to that of raw Wi-Fi, much lower than average 1.58 of fixed importance in 10 views case), assuring minimum influence on accuracy.

Our contribution is a semantic-aware scheduling method for wireless multi-view applications to reduce output latency. To the best of our knowledge, it takes the first step to automatically prioritize views by extracting the semantics of inference intermediates of multi-view models for scheduling view transmission. We also develop a scheduling algorithm to balance between view importance and accuracy for better scalability and accuracy.

The rest of this paper is organized as follows. Sec. II introduces background of multi-view applications; Sec. III describes system overview; Sec. IV presents detailed design of MVSAS; Sec. VI evaluates MVSAS and Sec. VII concludes.

II. Background

A. Multi-View Learning

Many real world data are often collected from different sensors from different perspectives or different measuring methods, since single-view data cannot comprehensively reflect useful information for a particular task. For example, for images and videos, color, texture and depth are all different features that can be regarded as multi-view data; images or videos taken from different angles about same objectives are also multi-view data. They can complement details or important information that cannot be captured from a certain single view.

Multi-view learning is a paradigm that given multi-view data as input, one function models each view to generate view modelling output and then jointly optimizes the view modelling output to exploit the redundant views of the same objectives and improve the learning performance. This is different from a naive solution that simply concatenating all multi-view data to a single view and then applying single-view machine learning, where the over-fitting problem can easily occur and the statistical property of each view is ignored [8], [12].

Scientists are making various progresses in multi-view learning. From co-training methods [], to co-regulation methods [] and recently margin-consistency methods [], they are pushing multi-view learning to higher performance and more general usages. However, it is notable that they are mostly optimizing the methods to jointly optimize all view modelling output and the basic framework of multi-view learning is not changed: modelling output of each view is generated in the middle of multi-view learning. And the modelling output of each view is the inference intermediates that we are making use of in MVSAS to lightweightly extract semantics about view importance, needless to infer directly from the raw view data.

B. View Importance

Many works in related areas have discussed that different views in a multi-view application have different importance or contribution to the application output. While many existing works on multi-view applications assume views are of equal importance, in [11] the authors explored the relationship among different visual tasks and developed a computational approach to estimate how much a view modelling output supplies useful information to another visual task. In an application where multiple tasks together facilitate better performance of another task on the edge, in [1] researchers observed that merely 12.72% of tasks have a high contribution of over 80% to the final performance. They exploited different importance of the tasks and schedules early execution of those important tasks to mitigate computation burden of such an application and reported a significant reduction of the execution time of the application.

III. Overview

A. System Model

We assume there are N wireless sensors that capture images of certain objects from different angles, and one host device that hosts a multi-view model capable of extracting useful information about the objects when fed with the captured images. The time t of the sensors and the host device are coordinated and they start working at time t_0 in the same time period p. When the $i^{th}(i = 0, 1, 2...)$ period starts at $t_0 + p \times i$, the sensors capture images and start transmitting the captured images via a shared wireless channel; the host device starts collecting the images at the same time. When the required views are collected at t^c_i, the host device will feed the images to the multi-view model for inference and get the inference result at time t'_i. The *output latency* is defined as $l^o_i = t'_i - (t_0 + p \times i)$, which consists of two major parts, the *input latency* $l^c_i = t^c_i - (t_0 + p \times i)$ for collecting the required views for input and the *inference latency* $l^{inf}_i = t'_i - t^c_i$. Typically, input latency and inference latency both increase as the number of required views increase.

We assume that once the required input views are collected, the device will feed the input to the model immediately so as to produce output as fast as possible; thus the input batch size is always one and only one inference process is running at a time.

B. System Overview

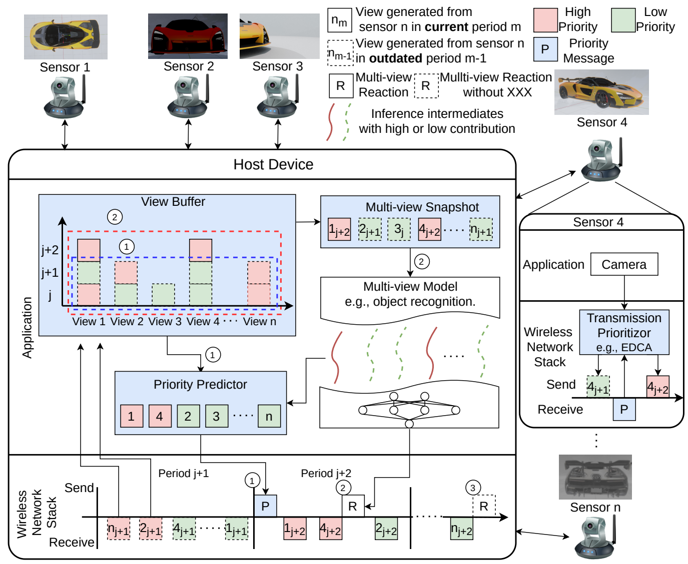


Fig. 1. Architecture of MVSAS (shaded in blue). MVSAS prioritizes the views collected from sensors according to their importance (e.g., inference intermediates inflecting quality of photographs captured in period j+1; photographs from sensor 1 and 4 were of high quality and were given high priority in period j+2) and generates priority messages every period as shown in ①. With MVSAS, the important views can be transmitted earlier and once the important views are collected, MVSAS completes the n views for the multi-view model to produce reaction at ②, both contributing to the latency reinforcement compared with high tail latency at ③.

The architecture of MVSAS is shown in Fig. 1. The multi-view model resides at the host device. The view buffer that collects the transmitted views from other sensors and their corresponding period resides at the host device. To analyse the importance of the latest buffered views and reach a proper semantic-aware priority schedule, the priority predictor inspects the buffered views at the view buffer and the corresponding intermediate semantics when the views are fed to the model. According to the priority schedule, when important views are collected, a multi-view snapshot collects the latest important views and the buffered less important views to form a complete N view input for the model, so that important views can be processed earlier with little loss in inference accuracy.

When the sensors receive the priority message, they will enforce the priority schedule with the transmission prioritizor that resides on them. It is notable that we adopt not only the contention-based techniques such as enhanced distributed channel access (EDCA [4]) for the transmission prioritizor, which are common in wireless networks, but also the application level scheduling. That is due to the observation that views for computer vision tasks are typically large in size (e.g., 1

MB per view) and they will be divided into packets during transmission. Although techniques like EDCA can enhance the possibility for packets from important views to acquire the channel in contention, one possible failed contention of a packet will lead to latency increase. In complementary to EDCA, we also choose to use ACK-based scheduling at application level for the transmission prioritizor: the host device will send an ACK message when all important views are collected and less important views are not allowed to transmit until the ACK message is received. The increased latency of less important views does not interfere the output latency of MVSAS and the cost for an ACK message is comparatively small to the transmission of views. In this way, the prioritized transmission of views as shown in Fig. 1 can be better enforced and important views can be collected earlier.

IV. Design

A. *Priority Predictor*

As discussed in Section III-A, the smallest permitted working period should be bounded by the inference latency, otherwise the output latency will gradually increase as the application progresses without an upper bound. In this case we will have output latency $l_i^o = T_i + T_i'$, where T_i is the collection latency, and T_i' is inference latency. T_i' is comparatively constant, and we consider minimizing T_i with minimized impact on the inference accuracy.

Assume all the sensors form a set C and in last period i, views $V_i = \{v_{i'}^j\}, j \in C, i' \leq i$ were scored $S_i = \{s_i^j\}$, where i' stands for the period that the view was generated, j represents the number of the sensor that the view comes from and s_i^j is the importance score. Staleness of each view is quantified as $o_i^j = i_j - i_j'$. We believe too large o_i^j (e.g., $o_i^j > 1$) (i.e., less important views generated too long ago may contain wrong information about the objectives and lead to more error) will deteriorate the inference accuracy and treat it as another parameter with a factor R to quantify the view's importance for staleness control:

$$I_i^j = \begin{cases} s_i^j & , o_i^j \leq 1 \\ s_i^j + R \times \frac{\sum_j s_i^j}{N} \times (o_i^j - 1) & , otherwise \end{cases} \tag{1}$$

Assume a subset of the sensors $C' \subset C$ and its portion of the sum of importance against all the sensors is

$$P_{C'} = \frac{\sum_{j \in C'} I_i^j}{\sum_{j \in C} I_i^j} \tag{2}$$

Empirically given a threshold P_e, if $P_{C'} \geq P_e$, $0 < P_e < 1$ and we treat views in C' as important views, we can reach a adequate inference accuracy. To also minimize T_i, we need to minimize the number of views in C', which is a combinatorial optimization problem formalized as follows.

$$\begin{aligned} \min_{C' \subset C} \; & len(C') \\ s.t. \quad & P_{C'} \geq P_e \end{aligned} \tag{3}$$

We greedily select views of largest importance score until the importance score of the chosen views exceed P_e as shown in Algorithm 2 The returned C' contains the important and least views that contribute to the output most.

```
function PRIORITY SCHEDULING({I_i^j}, P_e, C)
    P_C ← 0
    C' ← {}
    {I_i^j} ← sorted({I_i^j})
    for I in I_i^j do
        if P_C ≥ P_e then
            break
        end if
        P_C ← P_C + I
        C' ← C' + getView(I)
    end for
    return C'
end function
```

Fig. 2. Priority Scheduling Algorithm

B. *Transmission Prioritizor*

After prioritizing the views from the Priority Predictor, the transmission of views from each sensor should enforce the priority scheduling: all views in C' in period $i+1$ should be transmitted prior to any other view, so that the host device can collect them early. To this end, we consider view transmission in two different conditions: views transmitted in the period they are generated and views transmitted later than the period they are generated. As discussed in Section III-B, we use ACK-based scheduling from the application level to make sure less important views will be transmitted later than the important views generated in the same period, as shown in Algorithm 3 line 2-5. Considering a dense scenario that collecting the important views from period i' takes too long (e.g., longer than a period), when the collection ACK is received, the current latest period might have been $i > i'$. Continuing transmitting less important views from i' can block other views from the following periods and increase the total staleness, because old views naturally have higher staleness and newer less important views are blocked. Thus as a complementary staleness control method, we chose to get the latest period number and transmit the latest less important views as shown in line 7-9.

Besides, once less important views are handed to the underlying network stack, no scheduling from application level is able to control them and they may be transmitted in the following periods and increase the collection latency of the following periods. As a best effort, we use contention-based techniques (e.g., EDCA) to support a higher chance of important views to acquire the wireless channel when contending with less important views as shown in highPriority and lowPriority in Algorithm 3 line 3 and line 9.

C. *Multi-view Snapshot*

With the returned subset C' of all sensors that need to have high priority, MVSAS can inspect each view being received and form an effective input for the multi-view model once

```
function TRANSMISSION PRIORITIZOR(v_i^j, I_i^j)
    if isImportant(I_i^j) then
        transmit(v_i^j, highPriority)
    else
        wait(receiveCollectionACK())
        wait(importantViewsTransmitting())
        i' ←updateCurrentPeriod()
        if i' = i then
            transmit(v_i^j, lowPriority)
        end if
    end if
end function
```

Fig. 3. Transmission Prioritizor

all views in C' are collected. In this way, the multi-view application can proceed to the inference process with only views in C' collected and the collection latency is roughly decreased by $1 - \frac{len(C')}{len(C)}$. To cooperate with the ACK-based scheduling of Transmission Prioritizor, once views in C' are collected, a collection ACK will also be broadcast to the sensors for them to start transmitting less important views. The algorithm of multi-view snapshot is show in Algorithm 4.

```
function MULTI-VIEW SNAPSHOT(C')
    procedure ON RECEPTION(v_i^j)
        if v_i^j ∈ C' then
            C'.remove(v_i^j)
            add(input, v_i^j)
            if isEmpty(C') then
                broadcastCollectionACK()
                completeInput ← completeBuffer(input)
                return completeInput
            end if
        end if
    end procedure
end function
```

Fig. 4. Multi-view Snapshot

D. Discussion

Here we are going to discuss the inference staleness (i.e., $\{o_i^j\}$ of each inference) guarantee of MVSAS when the view number scales up to a high level. Assume in each time period p, at most n views of the multi-view application can be transmitted and we treat p as n slots for view transmission. At the start of each p, m views are generated by the sensors and we have $m > n$. In this case, the wireless network throughput is exhausted and collection latency can become higher and higher as the time progresses. We consider three scenarios, raw WiFi, MVSAS with certain random views chosen as important and disabled staleness control (referred to as fixed importance) and MVSAS. In raw WiFi, views are transmitted at best effort but inference will be made only when views from the coordinated period are all received. Thus all views in each inference are latest and $\forall i, \forall j, o_i^j = 0$, which guarantees no error is introduced from staleness.

In fixed importance scenario, we assume m' out of m ($m' < m$) views are transmitted in high priority and we also assume $m' < n$, otherwise no less important views can be transmitted ($o_i^j \to \infty$). In period i, $i \times (m-n)$ less important views generated from previous periods and $m - m'$ less important views generated from this period will contend for the $n-m'$ slots with equal priority. The probability that a view generated in period i gets transmitted after k periods is $P_{i,k}^j = (\prod_{l=i}^{i+k-1}(1-\frac{\binom{m-m'-1+l\times(m-n)}{n-m'-1}}{\binom{m-m'+l\times(m-n)}{n-m'}}))\times\frac{\binom{m-m'-1+(i+k)\times(m-n)}{n-m'-1}}{\binom{m-m'+(i+k)\times(m-n)}{n-m'}}$. As i increases, $P_{i,k}^j$ will gradually decrease towards 0 and apparently staleness will increase without an upper bound.

In MVSAS, as shown in Equation 1, assume a subset $C'' = \{j\}$, $o_i^j > 1$ and $\forall j' \in C - C'', o_i^{j'} \leq 1$, then

$$P_{C''} = \frac{s_i^j + R \times \frac{\sum_j s_i^j}{N} \times (o_i^j - 1)}{\sum_j s_i^j + R \times \frac{\sum_j s_i^j}{N} \times (o_i^j - 1)} \geq \frac{\frac{R}{N} \times (o_i^j - 1)}{1 + \frac{R}{N} \times (o_i^j - 1)}$$

We can learn if $o_i^j > \frac{P_e \times N}{(1-P_e)\times R} + 1$, we will have $P_{C''} > P_e$ and views in C'' will be treated as important, which means the staleness of each view has a solid lower bound $\frac{P_e \times N}{(1-P_e)\times R} + 1$ guaranteed by MVSAS's importance score calculation in Equation 1. Actually MVSAS can achieve a tighter lower bound of staleness with the staleness control mechanism IV-B (discussed in Sec. VI-C).

V. IMPLEMENTATION

We implemented MVSAS with python based on robot operating system kinetic (ROS-kinetic), which is broadly adopted by wireless sensors and robots. We adopted and implemented EDCA as the contention-based scheduling method in transmission prioritizor. We make less important views be transmitted in the lowest priority of EDCA [4] (i.e., AC_BK) and important views, priority scheduling and the collection ACK be transmitted in the highest priority (i.e., AC_VO).

VI. EVALUATION

A. Settings

Since MVSAS mainly focuses on transmission scheduling and multi-view inference, we evaluated MVSAS with real wireless networks, real workload and simulated sensors: sensors are simulated by laptops periodically transmitting images from real-world captured datasets through wireless networks; a server working as the host device is connected to the laptops in the same wireless networks and is equipped with four RTX2080 GPUs. The inference latency against view numbers on this host device is shown in Table I. These devices are each equipped with an MT76 wireless NIC that runs in Wi-Fi 5, 5 Ghz and has a roughly 210 Mbps peak throughput. The distance between sensors and the server is 5 meters.

Our chosen workload, Voxelpose [9], is a powerful multi-view 3D multi-human pose recognition model. It first samples a heatmap of humans from each input view with a backbone model (e.g., resnet), then forms the combined projection of

TABLE I
INFERENCE LATENCY AGAINST VIEW NUMBERS

View Number	3	4	5	6	7	8	9	10
Inference Latency/s	0.50	0.51	0.52	0.57	0.61	0.66	0.73	0.74

the heatmaps into a 3D space and finally builds up 3D human pose from the 3D projection. To infer the importance score of each view, we chose to cluster the hot spots in each heatmap with Sci-kit Learn, which represents the number of recognized joints of people in each view. Note that the accuracy in 3D pose recognition task is defined as the ratio of pose recognition results that have position errors within an error range compared with the ground truth. The error range is called the threshold of the accuracy. Another important metric of the recognition is the mean per joint position error (MPJPE), which reflects the average position error of the pose recognition result.

We chose Panoptic [5] as the evaluation dataset of Voxelpose, where high quality videos captured by over 30 coordinated cameras surrounding a room about certain human activities are provided together with the ground-truth action 3D model annotations of the humans. We randomly chose videos from 10 different cameras from the dataset in our evaluation. The size of each image extracted from the videos to be transmitted by each sensor is roughly 2 MB.

We compare MVSAS with raw WiFi and MVSAS with random fixed important views and disabled staleness control (referred to as fixed importance) on the same pre-trained voxelpose model. The number of important views in the fixed importance scenario is the ceiling of $0.6 \times viewNumber$. And the period grows as the view number scales up to avoid exhausting the wireless network throughput and the period follows $p = 0.8 + viewNumber \times 0.2$ (seconds). In our evaluation, we aim to find out

- When the view number scales up, whether MVSAS reduces the output latency of wireless multi-view applications while maintaining high accuracy?
- Where does the latency reduction gain come from?
- How is the high accuracy of MVSAS maintained?

B. Output Latency and Accuracy

TABLE II
MPJPE AGAINST VIEW NUMBER

View Number	Raw WiFi/mm	MVSAS/mm	Fixed Importance/mm
3	69.29	71.66	76.00
4	46.76	48.51	51.24
5	22.26	24.58	28.37
6	19.91	21.53	24.08
7	16.26	17.92	19.66
8	15.10	16.70	19.38
9	14.54	16.23	18.16
10	**13.05**	**14.04**	18.10

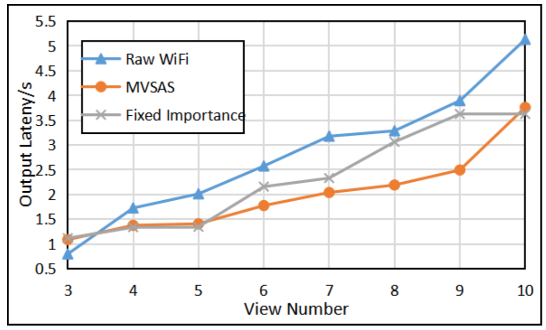


Fig. 5. Average output latency against view number.

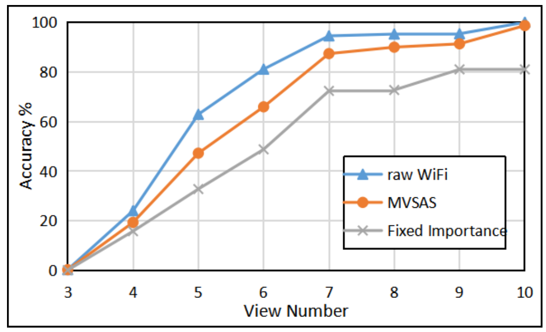


Fig. 6. Average accuracy against view number.

Fig. 5 and Fig. 6 show the average output latency and accuracy of 3D pose prediction result of each system under the threshold of 25 mm. Table II shows the mean per joint position error against view numbers. Note that in the raw WiFi case, inference will only be made when views from the same period are all received and all views in each inference have zero staleness (number of version of a stale view behind the latest views). Thus it has the highest accuracy that the multi-view model can achieve with the corresponding view number. It can be observed from Fig. 6 and Table II that in all the three systems, the inference accuracy increases as the view number increases. This phenomenon was also reported in [9]. With more views from different perspectives are combined into the input, the inference model can reduce the ambiguity in 3D pose estimation and reach higher accuracy.

As the view number increases, MVSAS retains similar accuracy as raw WiFi: average **88.5%** (at most **98.7%** in the 10 view case) accuracy of raw WiFi and average 1.08x MPJPE, while fixed importance reaches a lower accuracy (average 67.3% accuracy and 1.23x MPJPE compared with raw WiFi). However, although from Table I we can learn that inference latency doesn't evidently increase as view number increases, the average output latency of raw WiFi still increases from $0.79s$ to $5.13s$, which means the input latency of raw WiFi scales from $0.27s$ to $4.36s$ when the view number scales up. It is notable that the growth rate $\frac{4.36}{0.27} = 16.15 > \frac{10}{3}$, higher than the growth rate of view numbers, reflecting that the intensive contention among views further deteriorates the input latency.

On the contrary, from Fig. 5, MVSAS achieves a lower growth rate of output latency against view number: average **30.5%** (at most **36.9%** in the 9 views case) output latency reduction and average 38.4% input latency reduction compared with raw WiFi. Fixed importance reaches an average 20.2% output latency reduction and 25.1% input latency reduction. In conclusion, MVSAS retains a comparable accuracy to the highest accuracy of raw WiFi while achieves the highest output latency reduction.

C. Breakdown

To inspect the factors that build up MVSAS's advantages, we carried out a case study of 10 views as shown in Fig. 7. We define the *collection latency* as the time between the start of a period and the time that the views generated in this period are all collected. It can be learned from the cumulative distribution function (CDF) of collection latency in Fig. 7a that the usage of MVSAS or fixed importance would cause the collection latency to increase, compared with raw WiFi. This is because less important views from the period must wait until important views are received before they can be transmitted and they may contend with important views with a lower EDCA priority according to our design. However, in terms of input latency as shown in Fig. 7b, raw WiFi had the same input latency as collection latency, while both MVSAS and fixed importance achieved a lower input latency than raw WiFi. The low input latency of MVSAS and fixed importance benefit from the fact that they can start inference once important views are collected, accounting for their output latency reduction compared with raw WiFi.

To find out the reason for MVSAS's accuracy gain over fixed importance, we recorded the CDF of average staleness of each view at each inference as shown in Fig. 7c. Raw WiFi had zero staleness as discussed in Sec. IV-D. MVSAS managed to retain a low level of staleness (average 0.224, lower that 0.8 the whole time) while fixed importance suffered from a much higher staleness level (average 1.581, up to 2.2). The high staleness level can also be inferred from the fact that the fixed importance scenario had a much higher upper bound of collection latency from Fig. 7a (roughly 18 second compared to 10 second of the other two).

We also captured the CDF of P_C value of each inference in Fig. 8 during the evaluation in the 10 view case. P_C reflects the percentage of importance score that the selected important views or rather, the latest views take up from the total importance score in an inference process. It can be observed that P_C in raw WiFi is always 1 and MVSAS maintained a P_C close to the empirical $P_e = 0.6$. Lower P_C of MVSAS and similar accuracy compared with raw WiFi imply that there exist many less important views and MVSAS efficiently completes the view input to retain high accuracy. Although in the fixed importance system in the 10 view case, there were always 6 views selected as the important views according to our configuration, its P_C is evidently lower than MVSAS due to the fact that its important views were randomly selected. Lower level of staleness and higher but moderate P_C both help MVSAS achieve higher accuracy than fixed importance while maintains low output latency.

D. Limitations

As shown in Sec. VI-A, the usage of MVSAS relies on the knowledge of the exact multi-view model to distill the correct inference intermediates related to the output accuracy. How to automatically infer the correct inference intermediates and calculate the best-performing importance score remains a future work. Besides, the usage of MVSAS is limited to the multi-view applications that take continuous input, such as real world video streaming, as discussed in Sec. I. If the input data is not continuous, completing late views with their stale version can cause more error even when the staleness level is under control.

VII. Conclusion

In this paper, we presented MVSAS, a multi-view semantic-aware scheduling system. We showed that the contribution of each view to the output can be automatically and precisely inferred by analyzing the inference intermediates that contain semantics of each view. Being aware of the contribution (or semantic-aware), MVSAS managed to schedule the transmission and inference of the multiple views to significantly reduce the end-to-end output latency while retaining high inference accuracy.

References

[1] Q. Chen, Z. Zheng, C. Hu, D. Wang, and F. Liu, "Data-driven Task Allocation for Multi-task Transfer Learning on the Edge," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, Jul. 2019, pp. 1040–1050, iSSN: 2575-8411.

[2] D.-J. Deng, Y.-P. Lin, X. Yang, J. Zhu, Y.-B. Li, J. Luo, and K.-C. Chen, "IEEE 802.11ax: Highly Efficient WLANs for Intelligent Information Infrastructure," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 52–59, Dec. 2017, conference Name: IEEE Communications Magazine.

[3] J. Dong, W. Jiang, Q. Huang, H. Bao, and X. Zhou, "Fast and Robust Multi-Person 3D Pose Estimation From Multiple Views," 2019, pp. 7792–7801. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2019/html/Dong_Fast_and_Robust_Multi-Person_3D_Pose_Estimation_From_Multiple_Views_CVPR_2019_paper.html

[4] J. Hui and M. Devetsikiotis, "A unified model for the performance analysis of IEEE 802.11e EDCA," *IEEE Transactions on Communications*, vol. 53, no. 9, pp. 1498–1510, Sep. 2005, conference Name: IEEE Transactions on Communications.

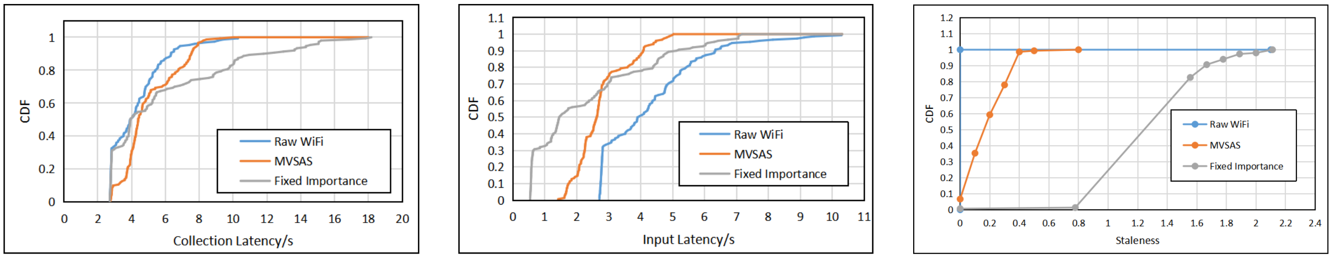
[5] H. Joo, T. Simon, X. Li, H. Liu, L. Tan, L. Gui, S. Banerjee, T. S. Godisart, B. Nabbe, I. Matthews, T. Kanade, S. Nobuhara, and Y. Sheikh, "Panoptic studio: A massively multiview system for social interaction capture," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[6] D. Kandris, C. Nakas, D. Vomvas, and G. Koulouras, "Applications of Wireless Sensor Networks: An Up-to-Date Survey," *Applied System Innovation*, vol. 3, no. 1, p. 14, Mar. 2020, number: 1 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2571-5577/3/1/14

[7] R. Panda and A. K. Roy-Chowdhury, "Multi-View Surveillance Video Summarization via Joint Embedding and Sparse Optimization," *IEEE Transactions on Multimedia*, vol. 19, no. 9, pp. 2010–2021, Sep. 2017, conference Name: IEEE Transactions on Multimedia.

[8] S. Sun, "A survey of multi-view machine learning," *Neural Computing and Applications*, vol. 23, no. 7, pp. 2031–2038, Dec. 2013. [Online]. Available: https://doi.org/10.1007/s00521-013-1362-6

[9] H. Tu, C. Wang, and W. Zeng, "VoxelPose: Towards Multi-Camera 3D Human Pose Estimation in Wild Environment," *arXiv:2004.06239 [cs]*, Aug. 2020, arXiv: 2004.06239. [Online]. Available: http://arxiv.org/abs/2004.06239



(a) Collection latency CDF of each inference in the 10 views case. Both MVSAS and Fixed Importance have an increase in collection latency than Raw WiFi

(b) Input latency CDF of each inference in the 10 views case. Although taking longer to collect all views from a period, MVSAS and Fixed Importance were able to input and start inference earlier

(c) Staleness CDF of each inference in the 10 views case. MVSAS maintains a low level of staleness compared to Fixed Importance to retain high inference accuracy when the view number scales up.

Fig. 7. Breakdown on the 10 view case.

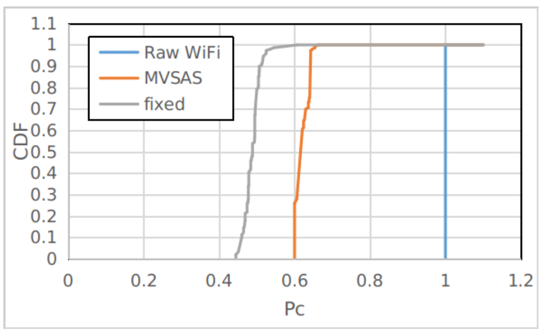


Fig. 8. P_C value of the three systems in the 10 view case.

[10] C. Xu, D. Tao, and C. Xu, "A Survey on Multi-view Learning," *arXiv:1304.5634 [cs]*, Apr. 2013, arXiv: 1304.5634. [Online]. Available: http://arxiv.org/abs/1304.5634

[11] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese, "Taskonomy: Disentangling Task Transfer Learning," 2018, pp. 3712–3722. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2018/html/Zamir_Taskonomy_Disentangling_Task_CVPR_2018_paper.html

[12] J. Zhao, X. Xie, X. Xu, and S. Sun, "Multi-view learning overview: Recent progress and new challenges," *Information Fusion*, vol. 38, pp. 43–54, Nov. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253516302032