Configuration-Adaptive Wireless Visual Sensing System with Deep Reinforcement Learning

Siyuan Zhou, Duc Van Le, Rui Tan, Joy Qiping Yang, and Daren Ho

Abstract—Visual sensing has been increasingly employed in various industrial applications including manufacturing process monitoring and worker safety monitoring. This paper presents the design and implementation of a wireless camera system, namely, EFCam, which uses low-power wireless communications and edge-fog computing to achieve cordless and energy-efficient visual sensing. The camera performs image pre-processing and offloads the data to a resourceful fog node for advanced processing using deep models. EFCam admits dynamic configurations of several parameters that form a configuration space. It aims to adapt the configuration to maintain desired visual sensing performance of the deep model at the fog node with minimum energy consumption of the camera in image capture, pre-processing, and data communications, under dynamic variations of the monitored process, the application requirement, and wireless channel conditions. However, the adaptation is challenging due to the complex relationships among the involved factors. To address the complexity, we apply deep reinforcement learning to learn the optimal adaptation policy when a fog node supports one or more wireless cameras. Extensive evaluation based on trace-driven simulations and experiments show that EFCam complies with the accuracy and latency requirements with lower energy consumption for a real industrial product object tracking application, compared with five baseline approaches incorporating hysteresis-based and event-triggered adaptation.

Index Terms—Wireless visual sensing, fog computing, computation offloading, deep reinforcement learning.

1 Introduction

Wireless cameras have been widely deployed for various visual sensing applications [1]. Without relying on cables for power supply and network connection, the wireless cameras can be deployed easily to monitor the locations of interest with wider coverage and/or more view angles. Specifically, they can enable event-based *ad hoc* deployments. Such features make wireless cameras promising for various monitoring tasks in the industrial context. For instance, in the scheme of *reconfigurable manufacturing system* [2] that can adjust the layout, capacity, and configuration of the production resources in response to changes in market demands or regulatory requirements, the *ad hoc* deployments of wireless cameras for configuration validation and calibration are desirable. Moreover, in a factory, the wireless cameras can be deployed for monitoring social distancing and worker safety (e.g., detecting falls) in an *ad hoc* fashion.

An visual sensing system in general involves compute-intensive image processing. Deep learning has been increasingly used for industrial computer vision applications [3]. However, the execution of deep models imposes high demand on computing resources. On the other hand, to achieve the cordless setting, the wireless cameras are often powered by batteries with finite capacities. Energy harvesting is generally infeasible in the indoor environments. Therefore, running the compute-intensive deep models on the wireless cameras is not desirable since otherwise bulky batteries or wired power supply will be needed.

Siyuan Zhou, Duc Van Le, and Joy Qiping Yang are with HP-NTU Digital Manufacturing Corporate Lab, Nanyang Technological University.
Rui Tan is with School of Computer Science and Engineering, Nanyang Technological University.
Daren Ho is with HP Inc.

In this paper, we design a system called Edge-Fog Camera (*EFCam*) that leverages wireless camera at the network edge and a resourceful fog node to achieve a cordless and energy-efficient visual sensing system. EFCam uses a battery-powered wireless camera ESP32-CAM [4] to perform image sensing,s generates smaller representations of the captured images, and offloads the advanced processing of the representations using deep models to the fog node. In EFCam, the geographical proximity of the wall-powered fog node to the data-generating wireless cameras at the network edge is in favor of the delay-critical industrial tasks, vis-à-vis the remote cloud that often suffers large jitters and long delays.

EFCam has various system parameters including the frame capture rate, image resolution, and pre-processing modes whose configurations affect the image processing accuracy and latency, and the camera's energy consumption. Existing studies [5]–[10] have designed various low-power wireless visual sensing systems. Similar to EFCam, those systems also have system parameters that need to be adaptively configured in response to the variations of the monitored process dynamics, application requirements, and wireless channel conditions for energy-efficient sensing and performance compliance. In particular, the configuration adaptation is desired in industrial settings due to the following reasons. First, the industrial application performance requirements and power saving opportunities may vary at run time. For instance, in a system for product object recognition and tracking, the frame rate should increase when the interested product objects appear in the field of view. Otherwise, the frame rate can be kept minimum to reduce image processing overheads and save power. Second, the industrial spaces typically have time-varying and noisy wireless channels due to the moving parts of production lines, vehicles and workers, as well as electromagnetic emissions from the electrified machinery. As a result, the industrial visual sensing system needs to deal with changeable data transmission performance when the wireless camera offloads its data processing to the fog node.

Adapting the configuration of the visual sensing systems is largely unexplored. The existing works have applied the hysteresis-based approach [5] and the motion detection sensors [11] to adjust the configuration of the camera parameters to save power. However, the hysteresis-based approach often suffers inferior performance under various exogenous variations, while the motion-triggered approach increases the system cost and energy consumption due to the need of additional motion sensors. Thus, in this paper, we propose a novel configuration adaptation scheme for EFCam to achieve good visual sensing performance with low camera energy consumption. We formally formulate a configuration adaptation problem that aims at maintaining the end-to-end visual sensing latency and accuracy within their respective bounds with the minimum expected energy consumption of the camera, under the dynamics of the monitored process, application performance requirements, and wireless channel conditions. The configuration includes the image frame rate, resolution, and the mode and parameters of the image pre-processing.

The key challenge in solving the formulated configuration adaptation problem is the lack of closed-form models describing the intricate relationships among the various concerned factors of the problem. To address this challenge, we apply deep reinforcement learning (DRL) to learn the optimal adaptation policy. However, the typical online training of the DRL agent takes excessive time. Instead, we develop computational models for the latency and camera energy consumption of image pre-processing and data transmissions given the wireless channel condition. Then, we use these models to drive offline training of the DRL agent. Finally, the trained agent is commissioned to adapt the configuration of EFCam at run time.

The preliminary version of this work [12] presented the design of EFCam in which a single wireless camera is supported by a fog node. As today's wireless cameras become cheaper and smaller in form factors, a number of cameras can be deployed in the interested space to achieve better monitoring coverage. Thus, a multi-camera system in which a single fog supports multiple cameras is desirable. This paper also presents the extended EFCam design to support multiple cameras. Specifically, multiple wireless cameras can transmit their image data to the fog node for advanced processing. The DRL agent running at the fog node adapts the configuration of all cameras.

We implement both the single-camera and multi-camera EFCam for an industrial product object recognition and tracking system. We perform extensive evaluation via simulations and testbed experiments. Specifically, we collect an image object dataset from the product manufacturing line to drive the design of the image pre-processing and recognition deep models. We also collect data traces of Bluetooth Low Energy transmission delay and camera energy consumption in the factory. We compare the DRL-based configuration adaptation with five baseline approaches incorporating hysteresis-based and event-triggered adaptation in the existing works. The evaluation results show that EFCam complies with the sensing performance requirements with less energy consumption for both single-camera and multi-camera systems.

In summary, the main contributions of this work are:

- We design and implement a deep learning (DL)-based image processing pipeline on an off-the-shelf wireless camera ESP32-CAM. Our design and experimental results can be useful to the developments of other visual sensing systems that use ESP32-CAM or similar camera platforms.
- We formulate the camera configuration adaptation problem and identify the relevant challenges in the industrial visual settings. Then, we propose a DRL-based approach for both the single-camera and multi-camera systems to address the identified challenges and find the efficient configuration policy.
- We conduct extensive evaluation on real-world testbeds to evaluate the effectiveness of the proposed approach for industrial visual sensing applications. We also compare the proposed approach with multiple baseline adaptation approaches.

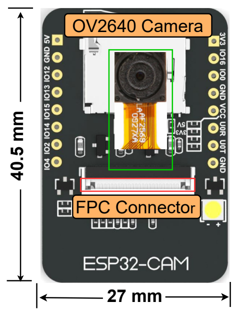
Paper organization: §2 reviews related work. §3 describes EFCam's design and implementation. §4 studies the impact of EFCam configuration on its performance. §5 and §6 present design and evaluation of single-camera EFCam. §7 presents multi-camera EFCam. §8 concludes this paper.

2 Related Work

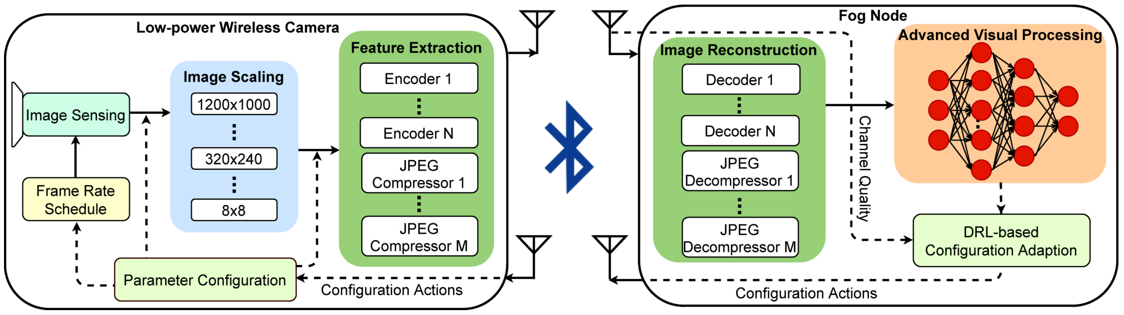
In this section, we review the related works on the low-power visual sensing systems, camera configuration, and reinforcement learning (RL)-based video quality control.

■ **Low-power visual sensing systems:** Multiple existing studies [5]–[10], [13]–[16] focused on the design of the low-power camera systems. For instance, the studies in [13]–[16] focused on developing low-power camera sensors which are equipped with early-processing capabilities to extract meaningful features of the raw images for further processing by advanced computer vision models. Such early-processing helps avoid image processing on redundant visual information, and thus reduces the camera energy consumption. For instance, Vazquez *et al.* [13] implemented a sensing front-end chip with embedded pre-processors on the focal-plane of the image sensors to extract and convert low-level features of the analog visual signal to a digital format. The use of these front-end chips leads to reduced camera energy consumption for analog-to-digital conversion (ADC). Gottardi *et al.* [14] used mixed-signal circuits to extract visual spatial-contrast and perform basic image processing at the sensor level to reduce the data output size.

The works in [5]–[10] prototyped various wireless visual sensing systems which leverage the low-power image capturing and/or local image processing to reduce the camera energy consumption. For instance, the authors in [5], [8] adopted a visual sensing approach that directly pipelines analog pixels straight from the camera to the wireless radio, which helps eliminate the need of the power-consuming hardware components (e.g., ADCs, codecs) as well as the memory for storing the images. However, the disadvantage



(a) Wireless camera



(b) Processing pipeline and configuration adaptation

Fig. 1. EFCam system overview. A fog node can support multiple low-power wireless cameras at the network edge.

of this approach is that the image frame rate of the camera is limited by the data transmission throughput of the wireless link. To address this issue, Josephson *et al.* [5] introduced a pixel-level compression technique to further reduce the size of the image data to be transmitted via the wireless link. In summary, above existing studies on smart visual sensing systems mainly focused on the design of low-power image sensors and/or local image processing techniques to reduce energy consumption. Differently, our work aims to adapt the configuration of a wireless visual sensing system to minimize the camera energy consumption while the visual sensing application requirements can be met.

■ **Camera configuration adaptation:** Several existing camera systems [5], [11], [17] can adjust the configuration of the system parameters to save power. For instance, the work in [5] proposed a hysteresis-based control approach that adjusts the frame rate and resolution of the camera to save power while maintaining a certain accuracy for a face detection task. However, as shown by our experiments in §6, the hysteresis-based approach has inferior performance under various exogenous variations. EFCam applies the model-free DRL [18] to learn the optimal configuration adaptation policy for the camera in response to the exogenous variations. The commercial security/surveillance camera products [11], [17] can also dynamically adjust their parameters based on the measurements of built-in motion sensors. For example, the Wi-Fi security camera described in [17] uses an infrared (IR) sensor to trigger video recording. Specifically, the camera stays in a sleep mode to save power. Once the IR sensor detects a motion (e.g., a moving person) in the camera's field of view, the camera wakes up and records the video at a fixed frame rate. Other security cameras on the market such as [19] are equipped with other types of sensors such as microwave and ultrasonic sensors for motion detection. Differently, EFCam applies a learning-based configuration approach to improve the camera's energy efficiency without the need of additional motion sensors. Note that the motion sensors increase the cost and power usage of the camera.

■ **RL-based video quality control:** RL has been recently applied for control of the video quality in video streaming systems [20], [21]. For instance, Pensieve [20] employs a data-driven DRL framework which aims at adapting the video bitrate with the main objective of mitigating the risk of frame stalling while maximizing the bandwidth utilization. OnRL [21] is an online RL scheme for multi-user real-time mobile video telephony. OnRL is based on a two-stage iterative RL frameworks in which the first stage characterizes individual users and the second stage aggregates the characteristics to strike a balance between individualized experience and swarm intelligence. While our work and the existing studies [20], [21] share the similar control approach based on RL for the camera configuration to maintain the high performance of visual sensing, we address different dynamics and constraints of the application requirements and wireless environment conditions.

3 EFCam Design & Problem Formulation

In this section, we describe the design to achieve a cordless and energy-efficient visual sensing system for industrial applications. Fig. 1 overviews EFCam which has two main components: the low-power wireless camera and the fog node. The camera performs low-power image sensing, local processing for image feature extraction or compression, and data transmission. The fog node reconstructs images, performs advanced visual sensing using the deep model, and a DRL-based controller for system configuration adaptation. A single fog node can support multiple low-power wireless cameras. This section describes the design and implementation details of EFCam.

3.1 Background and Hardware

For the camera, we choose ESP32-CAM [4], an off-the-shelf battery-powered wireless camera sized 40×27 mm as illustrated in Fig. 1(a). The ESP32-CAM consists of an OV2640 camera module, a low-power 32-bit MCU that can be clocked up to 240 MHz, 520KB internal SRAM memory, and 4MB external PSRAM memory. To reduce the wireless communication overheads, our design uses the deep model-based techniques to locally process the raw images at the camera before offloading the remaining computation to the fog node. Moreover, we intend to use the deep model-based local processing techniques which reduce the size of the data to be transmitted while preserving necessary information for the deep model-based processing at the fog node. Thus, it is desirable for the camera platform to support deployment of lightweight deep models. From our survey, the ESP32-CAM is a suitable platform that supports the TensorFlow Lite (TFLite) Micro, a deep learning library tailored for MCUs. Note that the OpenMV Cam platform [22] also supports the TensorFlow Lite Micro, but it does not have a radio for wireless data transmissions.

3.2 Processing Pipeline

Now, we present the detailed implementation of the end-to-end image processing pipeline of EFCam, which includes image sensing, scaling, local processing, and the image reconstruction and recognition at the fog node.

■ **Image sensing and scaling:** As shown in Fig. 1, the camera first sets a sensing schedule which defines the frame rate of the camera module. The camera uses an image sensing method that writes image frames to the SRAM memory. The ESP32-CAM supports image capture with eight resolution levels from 160×120 to 1600×1200. Our experiments on various datasets show that the image solution greatly affects the visual sensing performance and camera power usage. We implement a bilinear scaling-based image resizing method to provide the image resolutions which are not natively supported by the camera sensor.

■ **Image local processing and reconstruction:** To reduce wireless communication overheads, we implement two local image pre-processing techniques as follows. *JPEG compression*: It is a commonly used lossy image compression method [23]. EFCam supports multiple JPEG modes with the quality index from 0 to 80. A high quality index leads to better quality of decompressed images. *Autoencoder*: It is a deep learning-based image compression method. An autoencoder has two parts: encoder and decoder. The encoder is implemented in the camera to extract the high-level representation of a raw image while the decoder is deployed at the fog node to reconstruct the image based on the received data representation. The design of autoencoder is application-specific and needs training.

■ **Data transmission:** Bluetooth Low Energy (BLE) is used for the communications between the camera and the fog node due to its low energy consumption [24]. In particular, we implement a connection-based BLE mode in which the camera and the fog node perform as BLE peripheral and central, respectively. Specifically, we adopt the Generic Attribute Profile (GATT) protocol for managing the data exchange transactions at the application layer. More details can be found in Appendix B.

■ **Advanced image processing:** In the fog node, we implement convolutional neural network (CNN) models to process the images that are reconstructed by the decoder or JPEG decompressor based on the data received from the camera. EFCam's image processing framework follows an offloading approach that shifts the complex CNN-based image processing from the resource-constrained edge sensor to the resourceful fog node. Recently, the deep model compression has emerged as a promising approach to enable the deployment of the complex CNN models on the resource-constrained devices [25]. Compared with offloading, model compression may further reduce the image processing delay and energy consumption, since the data transmission is avoided. However, in general, it is non-trivial to balance the trade-off between the model size and accuracy reduction, subject to that the compressed model can fit into the device's limited memory. Thus, in this study, we adopt computation offloading, in which we perform local image processing (i.e., the JPEG compression or encoder) and use the low-energy communication (i.e., BLE) to reduce the communication energy consumption.

3.3 Formulation of Configuration Adaptation

Time is divided into intervals with identical duration of τ seconds, which is referred to as *adaptation period*. The beginning time instant of an adaptation period is called a *time step*. For the industrial product object recognition and tracking, the EFCam adapts the camera's configuration in response to the changes of the exogenous stochastic factors, including the time-varying industrial wireless quality, denoted by $\eta(t)$, and the appearance of product objects in the camera's field of view, denoted by $\xi(t)$. Note that $\xi(t) \in \{0, 1\}$. Denote $\eta_k = \eta(k\tau)$ and $\xi_k = \xi(k\tau)$, where $k \in \mathbb{Z}_{\geq 0}$. Denote by $\eta_{t=k\tau}^{(k+1)\tau}$ and $\xi_{t=k\tau}^{(k+1)\tau}$ the traces of $\eta(t)$ and $\xi(t)$ when $t \in [k\tau, (k+1)\tau]$. At the k^{th} time step, we let $\pi(\xi_k, \eta_k, \dots, \xi_0, \eta_0)$ denote the policy that determines a configuration, denoted by ω_k, based on the historical measurements of $(\xi_k, \eta_k, \dots, \xi_0, \eta_0)$. The configuration ω_k represents the combined setting of multiple parameters, including the image frame rate, resolution, local processing choice among JPEG compressors and autoencoders, BLE connection parameter, and MCU frequency, which jointly affect the camera's energy usage, sensing accuracy and latency during the next adaptation period. For a time horizon of K adaptation periods, the configuration adaptation aims to solve the following policy optimization problem:

$$\begin{aligned} \pi^* = &\operatorname*{argmin}_{\pi \in \Pi} \mathbb{E}_{\xi,\eta} \left[\frac{1}{K} \sum_{k=0}^{K-1} E_k \left(\omega_k, \xi_{t=k\tau}^{(k+1)\tau}, \eta_{t=k\tau}^{(k+1)\tau} \right) \right], \qquad (1) \\ \text{s.t. } &\mathbb{E}_{\xi,\eta} \left[A_k \left(\omega_k, \xi_{t=k\tau}^{(k+1)\tau}, \eta_{t=k\tau}^{(k+1)\tau} \right) \right] \geq A_{\text{req}}, \ \forall k \in [0, K-1], \\ &\mathbb{E}_{\xi,\eta} \left[L_k \left(\omega_k, \xi_{t=k\tau}^{(k+1)\tau}, \eta_{t=k\tau}^{(k+1)\tau} \right) \right] \leq L_{\text{req}}, \ \forall k \in [0, K-1], \end{aligned}$$

where Π represents the policy space; $\omega_k = \pi(\xi_k, \eta_k, \dots, \xi_0, \eta_0)$; $\mathbb{E}_{\xi,\eta}[\cdot]$ denotes the expectation over the two stochastic processes of $\xi(t)$ and $\eta(t)$; the $E_k(\cdot)$, $A_k(\cdot)$, and $L_k(\cdot)$ denote the camera's total energy usage, image processing accuracy and latency, respectively, in the k^{th} adaptation period; the A_{req} and L_{req} are the required accuracy and latency levels. Note that the camera may capture multiple images over an adaptation period, depending on the setting of the image frame rate. The objective is to find the optimal policy π^* that minimizes the camera's expected average energy usage per adaptation period while maintaining the expected $A_k(\cdot)$ above the A_{req} and the expected $L_k(\cdot)$ below the L_{req}.

Solving the policy optimization problem in Eq. (1) faces a basic challenge that the accuracy $A_k(\cdot)$ cannot be measured during the EFCam's online operations due to the unavailability of the sensing ground truths. Thus, in this study, we develop a learning-based configuration controller to address the optimization problem in Eq. (1). Specifically, during the offline training phase, the controller learns the optimal configuration policy based on the real data traces including the camera's power usages, image samples labelled with ground truths, and the image processing latency collected from the EFcam. Then, the learned policy is applied to adapt the configuration during the online inference phase. In §5 and §7, we will present the proposed learning-based approaches for adapting the configuration of the single-camera and multi-camera EFCam systems, respectively.

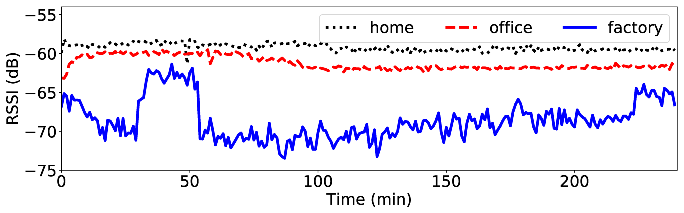
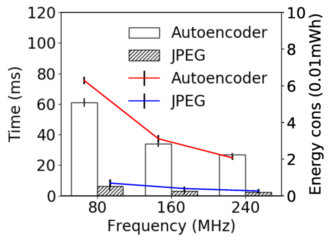
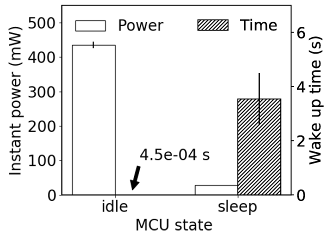


Fig. 2. BLE RSSI traces in different environments.



(a) CPU frequency.



(b) Idle and deep sleep.

Fig. 3. Impacts of MCU frequency & idle/sleep.

4 EFCam System Profiling

As discussed in §3, the proposed EFCam has multiple tunable parameters. In this section, we conduct profiling experiments to study the separate impact of these parameters on visual sensing performance and energy consumption. The results provide insights to guide the design of the configuration adaptation solution.

4.1 Profiling Methodology

We conduct the profiling based on four image datasets which are *MNIST* [26], *CIFAR-10* [27], *Casting* [28], and *Product*. Among these four datasets, we use the MNIST and CIFAR-10 datasets to understand the performance of EFCam on general visual sensing applications. The Casting and Product datasets represent the industrial visual sensing applications. Specifically, the *Casting* [28] is an industrial image dataset of 7,348 grayscale images showing top views of submersible pump impellers in a manufacturing casting process. The *Product* is a dataset containing 15,544 images of industrial product parts collected by us in a factory. Specifically, we used the ESP32-CAM to capture an image of product parts moving on a convey belt every one second. The images are labelled with four classes which indicate the number of the product parts appearing in the field of view, which is from 0 to 3. More details about these four datasets can be found in Appendix A.1.

4.2 Impacts of Image Compression and Resolution

We first conduct experiments to evaluate the impacts of the JPEG and autoencoder modes on visual sensing performance. Specifically, we design and train four convolutional autoencoder networks for image feature extraction and reconstruction for the above four datasets, respectively. Each autoencoder network contains an encoder with three convolution layers and a decoder with nine convolution layers. The rectified linear units (ReLUs) are used as the activation function for convolution layers in both the encoder and decoder, while the sigmoid activation is used at the output layers. Specifically, the encoder and decoder are deployed at the camera and fog node, respectively. The accuracy, data output size, and latency of the proposed EFCam under various configurations for the parameters of JPEG and autoencoder models are presented and analyzed in Appendix A.2. From the collected experiment results, we draw the following observations.

Observation 1: Given the same resolution, the classification accuracy and output data size increase with the compute complexity of the autoencoder network and JPEG quality index. The selection of the autoencoder or the JPEG should be adapted at run time.

We also conduct experiments to study the impacts of image resolution on the classification accuracy and compression time under the autoencoder networks and JPEG compression. The detailed results and analysis are presented in Appendix A.3. From the experiment results, we have the following observation.

Observation 2: When the image resolution is less than a certain level, the JPEG mode has higher classification accuracy than the autoencoder. Otherwise, the autoencoder achieves better accuracy. When the image resolution increases, the autoencoder results in higher output size and longer compression time.

4.3 Wireless Performance Benchmarks

We conduct experiments to investigate the BLE channel condition and its impact on the data transmission performance in various environments. In each experiment, the camera continuously transmits 400-bytes application data packets to the fog node. The BLE connection interval t_{ci} is set to 10 ms, The BLE transmitting power at the camera and fog node is set to be 0 dBm. The camera and the fog node are separated for about 2 meters. We perform the data transmission experiments in three environments which are factory, home, and office.

Fig. 2 shows the traces of the received signal strength indicator (RSSI) of the radio signal sensed by the fog node over a time duration of four hours in the three environments. We conduct experiments to evaluate the transmission delay and energy consumption of the proposed EFCam under the three environments whose wireless quality is represented by the RSSI traces in Fig. 2. More details can be found in Appendix A.4. From the experimental results, we draw the following observation.

Observation 3: The wireless channel condition in the factory environment can fluctuate significantly over time, which leads to highly variable packet transmission delay and energy consumption.

We also conduct experiments to evaluate the impacts of the configuration for the BLE connection interval and packet size on the transmission delay and energy consumption. Our experimental results show that the BLE connection interval and packet size can be configured at a fixed value to achieve the lowest transmission delay and energy consumption. More details can be found in Appendix B.

4.4 MCU Frequency and Idle/Sleep Mode

The clock frequency of the camera's MCU can be set to 80, 160, and 240 MHz. We conduct experiments to investigate the impact of the clock frequency configuration on the performance of the on-camera image processing.

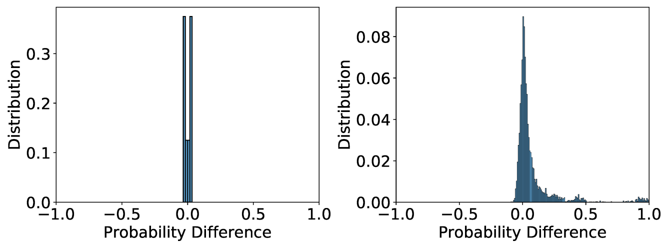
Fig. 3(a) shows the average inference time and energy consumed by the camera to process 100 CIFAR-10 images using the JPEG-10 and encoder network. As shown in Fig. 3(a), the higher clock frequency results in lower inference time and energy consumption. This is because with higher clock frequency, the MCU takes shorter time to execute the computation tasks of the image compression or feature extraction. For example, as shown in Fig. 3(a), the inference times of the autoencoder under the three clock frequency settings are 61, 34, and 27 ms, respectively. The corresponding camera powers are 440, 530, and 610 mW, respectively. As a result, the average energy consumed for executing the encoder under these three frequencies are 0.0074, 0.005, and 0.0045 mWh, respectively. Thus, EFCam sets the camera's MCU frequency to 240 MHz.

The ESP32-CAM supports a *deep sleep* mode. We consider an additional *idle* mode in which the MCU frequency is set to the lowest value of 80 MHz. Fig. 3(b) shows the camera's average power over a time duration of one second and wake-up time under the two modes. The wake-up time is defined as the time duration spent to turn on all components of the camera including MCU and BLE radio. As shown in Fig. 3(b), the idle mode consumes a higher power but has a shorter wake-up time of about 0.0005 seconds only, since every camera component is still on in this mode. The deep sleep mode consumes a lower power and needs a longer wake-up time of up to about four seconds, since every camera component is turned off in this mode.

From Fig. 3(b), when there is no pending task, the camera should be set to the idle or deep sleep mode, depending on the idle time. For example, when the idle time is less than five seconds, the idle node is selected. Otherwise, the deep sleep mode is chosen to save more power. However, the current version of ESP32-CAM cannot rebuild the BLE connection after waking up from the deep sleep. Therefore, in our current implementation of EFCam, we do not consider the deep sleep. The camera is set to the idle mode when it has no pending tasks.

4.5 Assessment of Markov Property

When EFCam is applied for industrial product object recognition and tracking, it aims to adapt the camera's configuration in response to the evolution of the appearance of the product objects and the wireless channel quality in the industrial environment. We conduct experiments to assess whether the above two stochastic processes satisfy the Markov assumption (MA), i.e., $\mathbb{P}[y_{k+1} \mid y_k] = \mathbb{P}[y_{k+1} \mid y_k, y_{k-1} \dots y_0]$, where y_k represents the process state at the k^{th} time step. The MA suggests that the probability distribution of the state transition from y_k to y_{k+1} is independent of the past states $y_{k-1}, \dots, y_0$. The MA is a basic property of the systems where RL is applicable [29]. When we assess the MA for the industrial product object recognition application, we use the probability difference, denoted by $\Delta P = \mathbb{P}[y_{k+1} \mid y_k] - \mathbb{P}[y_{k+1} \mid y_k, \dots, y_{k-N}]$,



(a) Object appearance. (b) Wireless channel quality.

Fig. 4. Compliance of state transition with the Markov assumption.

where $N \geq 0$, as an MA compliance metric. A lower absolute value of ΔP indicates better compliance. Fig. 4 shows the distributions of ΔP with $N = 1$ for the transitions of the product object's appearance in the Product dataset and the wireless quality in an industrial environment over 12,000 adaptation periods, where each adaptation period τ is five seconds. From Fig. 4, we can see that these two stochastic processes have good compliance with the MA because their values of ΔP concentrate at zero. In particular, the process of the appearance of the product object has higher MA compliance, because its state mostly remains the same over two consecutive adaptation periods.

4.6 Implications of Profiling Results

From the above profiling results and observations, the MCU frequency and BLE connection settings can be fixed at certain values that result in the least energy consumption and latency. On the other hand, the configurations for the image resolution, frame rate, and local processing method (i.e., the JPEG and autoencoder modes) should be adapted over time. Thus, EFCam adapts the configuration of these parameters. Moreover, due to the good MA compliance of the two exogenous factors, we model the configuration adaptation as a Markov decision process (MDP), and adopt the DRL to learn the optimal configuration policy based on the observed states of the product object appearance and the wireless channel quality. The detailed MDP formulations are presented in §5.1 and §7.1 for the single-camera and multi-camera EFCam systems.

5 Adaptation of Single-Camera System

This section formulates the problem of adapting the configuration of a single camera as an MDP. Then, we propose a DRL-based solution. Lastly, we present the details of the DRL agent training.

5.1 Single-Camera MDP Formulation

Since the more general policy optimization problem in Eq. (1) is reduced to MDP, we can focus on learning a simplified policy of $\omega_k = \pi(\xi_k, \eta_k)$. In addition, we apply Lagrangian relaxation [30] to convert the constrained policy optimization problem in Eq. (1) to an unconstrained one with a new objective function of

$$\mathbb{E}_{\xi,\eta}\left[\frac{\sum_{k=0}^{K-1} E_k}{K} + \beta_1\left(A_{\text{req}} - \frac{\sum_{k=0}^{K-1} A_k}{K}\right) + \beta_2\left(\frac{\sum_{k=0}^{K-1} L_k}{K} - L_{\text{req}}\right)\right],$$

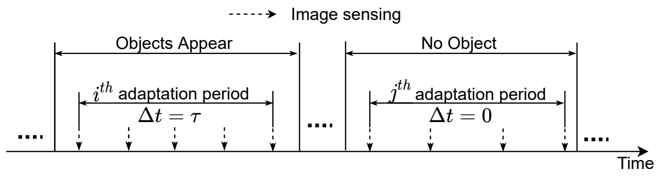


Fig. 5. Image sensing workflow for industrial product object tracking.

where $\beta_1, \beta_2 \geq 0$ are Lagrangian multipliers. The MDP addressing the above unconstrained policy optimization problem can be represented by the 4-tuple $(\mathcal{X}, \mathcal{A}, r, P)$, where $\mathcal{X}$ and $\mathcal{A}$ are the state and action spaces, respectively, r is the reward function, and P is the system transition probability. We define the immediate state, action, and reward function as follows.

System state: The system state, denoted by $\mathbf{x} \in \mathcal{X}$, is a vector $\mathbf{x}_k = [\xi_k, \eta_k]$, where $\xi_k \in \{0,1\}$ represents the appearance of the product object and η_k is the RSSI of the wireless channel at the k^{th} time step. Since the ground-truth value of ξ_k cannot be obtained during the online inference phase, we use the image processing result of whether the object is detected in the last frame captured in the previous adaptation period to represent the ξ_k. Note that the camera may capture multiple image frames in an adaptation period.

Configuration action: Let $f \in [f_{\min}, f_{\max}]$ denote the number of frames per adaptation period τ, where $f_{\min}$ and $f_{\max}$ are the minimum and maximum number of image frames required by the application, respectively. The camera performs the image scaling to resize the captured images to a resolution, denoted by $r \in [r_{\min}, r_{\max}]$, where the $r_{\min}$ and $r_{\max}$ denote the minimum and maximum resolutions, respectively. We denote c as the local image processing mode (i.e., the JPEG with a quality index and autoencoder network with a setting for its hyperparameters). The configuration action, denoted by $\mathbf{a} \in \mathcal{A}$, is a vector $\mathbf{a} = [f, r, c]$.

Reward function: When an action $\mathbf{a}_k$ is performed at the k^{th} time step with a system state of $\mathbf{x}_k$, let $e_k(\mathbf{x}_k, \mathbf{a}_k)$ denote the total energy consumed by the camera for the image sensing, scaling, local processing, and data transmission over the k^{th} adaptation period; let $l_{\max}(\mathbf{x}_k, \mathbf{a}_k)$ denote the maximum image processing latency during the period capturing f images; let ϕ_k denote the number of images that are captured and correctly recognized during the k^{th} adaptation period. We define a penalty function as follows:

$$p_k(\mathbf{x}_k, \mathbf{a}_k) = \lambda_1 \cdot \mathcal{N}(l_{\max}(\mathbf{x}_k, \mathbf{a}_k) - l_{\text{th}}) + \lambda_2 \cdot \mathcal{N}(\phi_{\text{req},k} - \phi_k), \quad (2)$$

where l_{th} is the upper bound threshold for the maximum latency, $\phi_{\text{req},k}$ is the number of correctly recognized images required over the k^{th} adaptation period, λ_1 and λ_2 are configurable weights, $\mathcal{N}(x)$ is a normalization function, i.e., $\mathcal{N}(x) = \frac{\max(x,0)}{x_{\max}}$, where $x_{\max}$ is the maximum value of x. From the definition of $p_k(\mathbf{x}_k, \mathbf{a}_k)$, if the maximum latency $l_{\max}(\mathbf{x}_k, \mathbf{a}_k)$ does not exceed the l_{th} and the $\phi_k(\mathbf{x}_k, \mathbf{a}_k)$ is higher than $\phi_{\text{req},k}$, which means that no delays exceed the threshold and enough images are captured and recognized correctly, no penalty is applied. The immediate reward, denoted by $r_k(\mathbf{x}_k, \mathbf{a}_k)$, is defined as $r_k(\mathbf{x}_k, \mathbf{a}_k) = -e_k(\mathbf{x}_k, \mathbf{a}_k) - p_k(\mathbf{x}_k, \mathbf{a}_k)$. Thus, the reward accounts for the energy consumption and the degree of violating the latency and accuracy requirements.

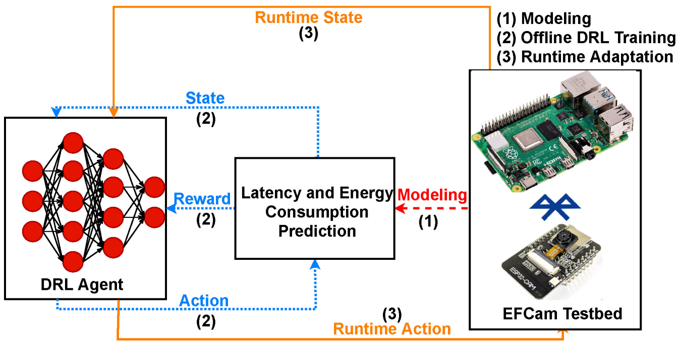


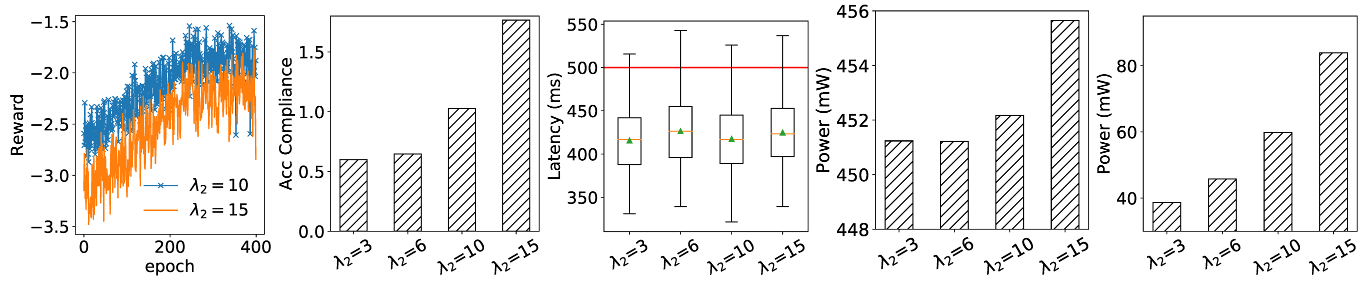
Fig. 6. Workflow of DRL-based adaptation.

We now use the case study application of industrial product object tracking as an example to illustrate the setting of the reward function. We define $\phi_{\text{req},k}$ as the required number of correctly recognized images in the current period of τ. It is calculated as $\phi_{\text{req},k} = 1 + \rho_{\text{req}} \cdot \Delta t$, where ρ_{req} is the required frame rate and Δt is a time duration that the objects appear in the camera's view during the period of τ. Fig. 5 shows an example that the Δt in the i^{th} and j^{th} adaptation periods equal to τ seconds and zero, respectively. As a result, the accuracy requirement $\phi_{\text{req},k}$ in the two adaptation periods are $1 + \rho_{\text{req}} \cdot \tau$ and 1, respectively. The $\phi_{\text{req},k}$ is equal to or higher than one to make sure that at least one image is correctly recognized in an adaptation period.

Objective: The main objective is to find an optimal policy π^* that selects action $\mathbf{a}$ based on state $\mathbf{x}$ at every time step to maximize the expected long-term reward, denoted by $\mathbb{E}_{\xi,\eta}[\sum_{k=0}^{K-1} \gamma^k r_k]$, where $\gamma \in [0,1]$ is the discount factor.

5.2 DLR-based Solution

We adopt DRL to learn the optimal configuration adaptation policy. Under the typical setting, a DRL agent learns the optimal policy during the online interactions with the controlled system. However, for the formulated configuration adaptation problem, the online DRL scheme faces the following two challenges. First, it may take a long time to converge, which may lead to the camera's excessive battery energy consumption. Second, during the online learning phase, measuring the camera's power and image processing accuracy is cumbersome or even infeasible. Specifically, the camera is not capable of metering its power in real time. Moreover, the image classification accuracy cannot be measured during online learning due to the lack of ground-truth labels. To address these challenges, we adopt an offline training approach as illustrated in Fig. 6, which consists of three steps. First, we model the image processing latency and energy consumption based on real data traces collected from the deployment environment. This step is detailed in §5.2.1. Second, we use the real data traces and models built in the first step to drive the offline training of the DRL agent. This step is detailed in §5.2.2. Third, after the completion of the offline training, the DRL agent is commissioned to adapt the configuration of EFCam for industrial visual sensing.



(a) Reward. (b) Accuracy. (c) Latency. (d) Power with idle. (e) Power with deep sleep.

Fig. 7. Training and execution results under various settings of λ_2, λ_1 = 1 and l_{th} = 500ms. (a) Traces of training reward ; (b)-(e) Execution results. The red line in (c) represents the l_{th}.

5.2.1 Modeling latency and energy consumption

This section models the image processing latency and camera energy consumption. The resulted models will be used for driving the offline training of the DRL agent. Denote by $l(\mathbf{x}, \mathbf{a})$ and $e(\mathbf{x}, \mathbf{a})$ the image processing latency and camera energy consumption, respectively, when an action $\mathbf{a}$ is taken at state $\mathbf{x}$. They can be expressed as: $l(\mathbf{x}, \mathbf{a}) = l_r + l_c + \sum_{i=1}^{n} l_{p,i}(\eta) + l_{\text{fog}}$, and $e(\mathbf{x}, \mathbf{a}) = (l_r + l_c)p_o + \sum_{i=1}^{n} e_{p,i}(\eta) + e_{\text{idle}}$, where l_r, l_c, $l_{p,i}(\eta)$, and l_{fog} are the image scaling, compression, packet transmission time for i^{th} packet, and fog image processing latency, respectively; p_o is the camera's operating power; e_{idle} is energy consumed in the idle mode; $e_{p,i}(\eta)$ is energy consumed for transmission i^{th} packet; and n is the number of packets used to deliver an image to the fog node. The l_r, l_c, and l_{fog} can be determined by offline measurements. Using real measurements, we fit Gaussian distributions to model the $l_{p,i}(\eta)$ and $e_{p,i}(\eta)$. Details can be found in Appendix C.

5.2.2 Offline training of DRL agent

We adopt the learning framework in [18] to train offline a deep Q-network (DQN) for the configuration adaptation agent to capture a good policy. Specifically, the DQN is trained through interacting with a simulated EFCam environment for several episodes, each of which consists of T adaptation periods. The simulation is driven by real data traces. An episode starts with a state chosen randomly from the training data that includes real traces of the RSSI and sequential object images. Then, at the k^{th} time step, an action $\mathbf{a}_k$ is selected for state $\mathbf{x}_k$ according to the ε-greedy algorithm. Given the selected action $\mathbf{a}_k$, the images from the image traces are fed to the compression or feature extraction module. Then, the CNNs are used to classify the reconstructed images (cf. §4). The ξ_{k+1} is set to the classification result of the last image captured in the current adaptation period, while the η_{k+1} is taken from the RSSI trace. To calculate the immediate reward r_k, the latency and power consumption are estimated using the models developed in §5.2.1. The ϕ_k and $\phi_{req,k}$ are calculated using the image classification results and real image traces, respectively. During the learning phase, two mechanisms, i.e., experience replay and target Q-network, are used to update the weights of the DQN every time step. Specifically, at every time step k, a transition tuple $(\mathbf{x}_k, \mathbf{a}_k, r_k, \mathbf{x}_k + 1)$ is stored in the experience replay memory. The weights of the primary Q-network are updated using a mini-batch of M transitions which is randomly sampled from the replay memory. The weights of the target Q-network are periodically set to the weights of the primary Q-network.

6 Evaluation of Single-Camera System

This section evaluates the DRL-based configuration adaptation for the vision-based product object tracking application, via both trace-driven simulations and testbed experiments.

6.1 Offline DRL Training and Performance

6.1.1 DRL and simulation settings

We build a fully connected deep neural network as the DQN that consists of an input layer, three hidden layers, and a linear output layer. Three hidden layers has 128, 64, and 32 ReLUs, respectively. The DRL agent takes as input the system state $\mathbf{x} = [\xi, \eta]$ and chooses the action $\mathbf{a} = [f, r, c]$ from a discrete action space: f is from $\{1, 2, 3, 4, 5\}$; r is from $\{16 \times 16, 24 \times 24, 32 \times 32\}$; c is selected from three JPEG compression modes with the image quality index of 0, 40, 80, and the three autoencoder models developed in §4. For the offline training of DQN, we use the Adam optimizer with the learning rate of 0.0001. The ϵ of the ϵ-greedy method decreases linearly from 1 to 0.1 during the learning phase. The adaptation period τ is five seconds.

We use our Product dataset to drive the evaluation of the industrial vision-based product object tracking application system. In addition, we use real traces of 15,967 RSSI samples measured in a factory over 4.5 hours, as illustrated in Fig. 2 to model the condition of the industrial wireless environment. In particular, the first 10,000 RSSI and 10,000 image samples are used for training and the remaining data for evaluating the trained DRL agent. For the performance requirements, we set the ρ_{req} to 0.8, and l_{th} to 500 ms.

6.1.2 Training of DRL agent

The offline training is conducted for 400 epochs, each of which includes $T = 500$ adaptation periods. We evaluate the convergence of the DRL agent training under various settings for λ_1 and λ_2, which affect the trade-off between the energy consumption and compliance with the latency/accuracy requirements. Fig. 7(a) shows the DQN training traces of average rewards when the λ_2 is from 10 to 15 and $\lambda_1 = 10$. Along with the training epochs, the reward always increases and then becomes flat under different settings of λ_2. We also train the DRL agent under

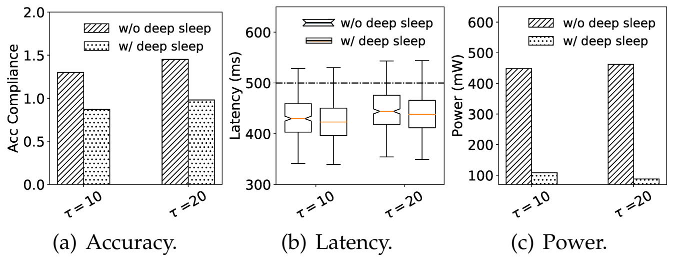


Fig. 8. Impacts of the deep sleep (λ_1=1, λ_2=10, l_{th}=500ms).

various settings for the λ_1, l_{th} and ρ_{req}. The results show that the DRL agent can converge after a certain number of training epochs (e.g., 400) with the reward learning curves similar to that shown in Fig. 7(a).

6.1.3 *DRL execution performance*

We evaluate the performance of the trained DRL agent for adapting the configuration of EFCam in trace-driven simulations over a period of 1,000 adaptation periods. Fig. 7(b)-(d) show the accuracy compliance (i.e., $\frac{\sum_{k=0}^{k=999}\phi_k}{\sum_{k=0}^{k=999}\phi_{\text{req},k}}$), box plots for the distribution of latency and average camera power over the 1,000 adaptation periods under various settings for λ_2 from 3 to 15. Higher accuracy compliance indicates that more images are correctly recognized to meet the application's accuracy requirement. From Fig. 7(b), the accuracy compliance increases with λ_2. The DRL approach can adapt the camera's configuration to meet the accuracy requirement, i.e., $\frac{\sum_{k=1}^{k=999}\phi_k}{\sum_{k=0}^{k=999}\phi_{\text{req},k}} \geq 1$ under $\lambda_2 \geq 10$. From Fig. 7(c), the latency is mostly below the l_{th} under various λ_2 settings. Moreover, as shown in Fig. 7(d) and (e), the camera power consumption increases with λ_2. This is because, with higher λ_2, the camera increases the frame rate and resolution and selects the JPEG or autoencoder mode such that more images are recognized correctly. Thus, the camera consumes more power for data processing and transmission.

As mentioned in §4.4, the ESP32-CAM cannot establish the BLE connection after waking up from deep sleep. Thus, we train the DRL agent with an assumption that the camera is set to the idle mode when it has no pending tasks. Fig. 7(d) shows the camera's power with the idle mode. Fig. 7(e) presents the power if we can set the camera to the deep sleep mode.

6.2 DRL with Deep Sleep

This section conducts simulation experiments to investigate the performance of EFCam when the camera can be configured to the deep sleep mode. In particular, the DRL agent considers an additional action variable that determines the selection of the deep sleep or the idle mode when the camera has no pending tasks. Fig. 8 compares the average accuracy compliance, latency distribution, and average camera power consumption of two trained DRL agents without and with the deep sleep over 500 adaptation periods. The setting of τ varies from 10 seconds to 20 seconds. Without the deep sleep, the camera is always set to the idle mode when it has no pending tasks. From Fig. 8(a), the deep sleep leads to the lower accuracy compliance. The reason is that when the

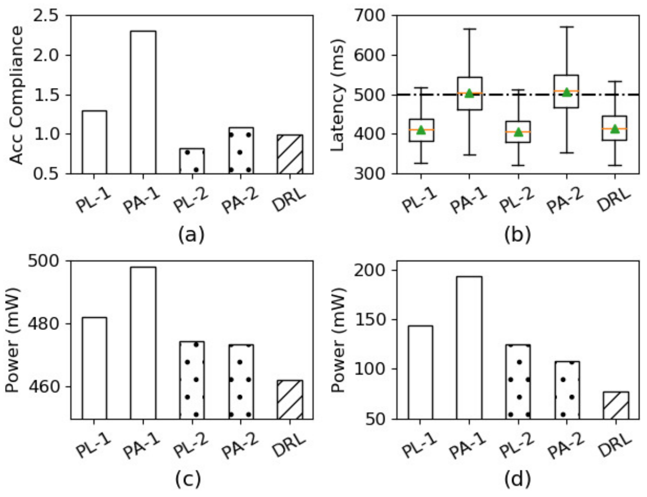


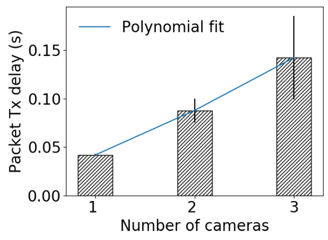
Fig. 9. Results in real experiments with the single-camera testbed. (a)-(d) show the accuracy, the latency, the average power consumptions with the idle and the deep sleep, respectively.

deep sleep is selected, the camera may not wake up in time to capture all required images over the adaptation period. Note that when the τ = 20 seconds, the DRL agent with the deep sleep can still meet the accuracy compliance requirement of 1. Moreover, from Fig. 8(c), the deep sleep results in lower energy consumption. This is because as shown in Fig. 3(b), the idle mode consumes higher power than the deep sleep mode. Thus, adapting the camera between the idle mode and the deep sleep mode can reduce the power usage while maintaining the required accuracy and latency.

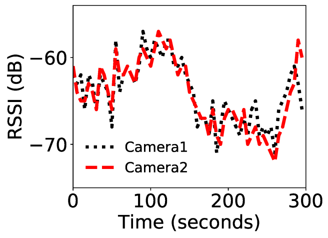
6.3 Testbed Experiments

We conduct a set of experiments to investigate the performance of EFCam in a real office environment. At the camera, we use C++ and APIs provided by the ESP32-CAM to implement the image scaling, JPEG compression, encoder of the autoencoder, BLE-based data transmission, and parameter configuration. The fog node is prototyped by a Raspberry Pi 4 single-board computer, in which the JPEG decompressor, decoder of the autoencoder, CNN image classifiers and DRL agent are implemented in Python 3.7 using TensorFlow 2.3 and PyTorch 1.6. The implementation of these models requires a total of $13.6\,\text{MB}$ memory which cannot be provided by the ESP32-CAM with $4.52\,\text{MB}$ RAM only. Thus, offloading the CNN computation to the fog is required. We perform experiments in a lab in which the wireless camera and fog node are separated for about 2 meters. We use our Product images to drive the evaluation. At the fog node, the DRL agent trained with λ_1 = 1, λ_2 = 10, and l_{th} = $500\,\text{ms}$.

To evaluate the effectiveness of the proposed EFCam, we compare our DRL-based approach with four baseline approaches which are variants of a hysteresis-based camera configuration adaptation approach proposed in [5]. Specifically, the first two baseline approaches always select the maximum resolution (i.e., $r = r_{\max}$) and the maximum number of captured images (i.e., $f = f_{max}$), and adapt the image pre-processing mode c as follows:

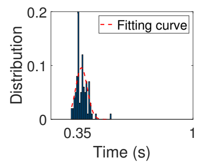


(a) Packet transmission delay.

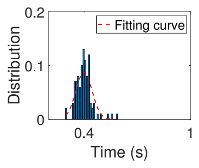


(b) BLE RSSI values.

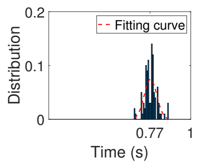
Fig. 10. Impacts of number of cameras on BLE connection performance.



(a) m =3, u = 1.



(b) m = 3, u = 2.



(c) m = 3, u = 3.

Fig. 11. Histograms and fitted Gaussian distributions for the average image processing latency of the fog node.

- The *prioritized-latency (PL-1)* baseline approach selects the c that leads to the shortest latency.
- The *prioritized-accuracy (PA-1)* baseline approach selects the c that leads to the largest number of correctly recognized images ϕ.

The remaining two baseline approaches are PL-2 and PA-2, which additionally adapt the configuration for the f. Specifically, with the PL-2 and PA-2 approaches, the initial configuration of the f is set to $f_{\max}$. At the beginning of each adaptation period, if $\xi = 0$, f is decreased by 1. Otherwise, f is increased by 1 until it reaches to $f_{\max}$.

Fig. 9 shows the accuracy compliance, box plots for the latency, and average power consumption of the camera under our DRL approach and four baseline schemes over an experiment period of one hour. The PA-1 approach achieves the highest accuracy compliance but leads to the longest latency and requires the highest power consumption. The reason is that to achieve higher accuracy, the PA-1 approach always selects the autoencoders for image compression at the camera, which results in the long compression latency. Moreover, the PA-1 and PA-2 overprovision the accuracy performance (i.e., the accuracy compliance is higher than one) at the cost of high power consumption. On the other hand, the PL-2 has the shortest latency but the lowest accuracy compliance. This is because with the PL-2 approach, the camera always selects the JPEG compression mode which leads to lower compression latency but low image quality. Compared with the four baseline approaches, the proposed DRL approach mostly meets the accuracy and latency requirements and consumes the least power. These results imply that the proposed DRL approach can strike a good trade-off to achieve more energy savings and higher sensing performance compliance levels.

Note that in our real experiments, we set the camera to the low-frequency idle model when it has no pending tasks. The averages of the power measurements are presented in Fig. 9(c). If the camera's power in the idle mode is replaced by the power of the deep sleep mode, the average power is reduced as shown in Fig. 9(c). The above results show the superiority of our DRL approach, compared with the hysteresis-based baseline approaches.

7 Adaptation of Multi-Camera System

This section presents the design and evaluation of the multi-camera EFCam, in which the fog node runs multiple CNN models to process images received from multiple cameras. The fog node also runs a DRL agent to adapt the configuration of all cameras at run time. In what follows, we formulate an MDP problem of adapting the configuration of multiple cameras. Then, we present profiling experiments to study the impact of the number of cameras on the visual sensing delay and BLE connection performance. Lastly, we present the evaluation results.

7.1 Multi-Camera MDP Formulation

Let m denote the number of cameras. We extend the MDP problem formulated in §5.1 for the configuration adaption of multiple cameras as follows.

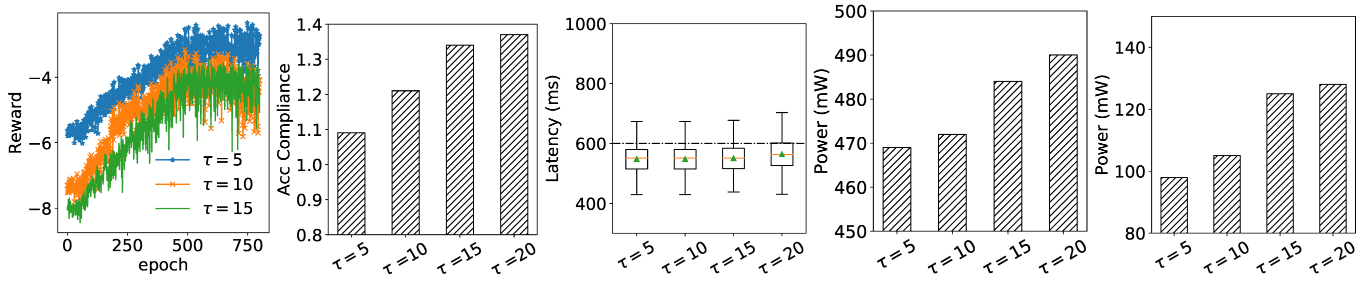
System state: The system state, denoted by $\mathbf{x}_{\text{mul}}$, is a vector $\mathbf{x}_{\text{mul}} = [\boldsymbol{\xi}_{\text{mul}}, \eta]$, where $\boldsymbol{\xi}_{\text{mul}} = [\xi_1, \ldots, \xi_m]$ represents the image processing results of m cameras, and η is the RSSI of the wireless channel. For the industrial product object tracking, the ξ_i $(i = 1, \ldots, m)$ is 1 if an object is detected in the last frame of the camera i. Otherwise, the ξ_i is zero.

Action: The action is $\mathbf{a}_{\text{mul}} = [\mathbf{a}_1, \ldots, \mathbf{a}_m]$ where $\mathbf{a}_i = [f_i, r_i, c_i]$ is a configuration action for camera i. Note that the f_i, r_i, and c_i denote the selected frame rate, image resolution, and local image processing mode of camera i.

Reward function: When an action $\mathbf{a}_{\text{mul}}$ is performed at a system state of $\mathbf{x}_{\text{mul}}$, let $e_k(\mathbf{x}_{\text{mul}}, \mathbf{a}_{\text{mul}})$ denote the total energy usage of all cameras for the image sensing, local processing and data transmission, over the k^{th} adaptation period. We define a penalty function: $p_{\text{mul}}(\mathbf{x}_{\text{mul}}, \mathbf{a}_{\text{mul}}) = \sum_{i=1}^{m} p_i$, where p_i is the camera i's penalty that is calculated using Eq. (2). Similar to the single-camera EFCam, the immediate reward r_{mul} is defined based on the total energy usage and the penalty for all cameras as follow: $r_{\text{mul}} = -e_k(\mathbf{x}_{\text{mul}}, \mathbf{a}_{\text{mul}}) - p_{\text{mul}}(\mathbf{x}_{\text{mul}}, \mathbf{a}_{\text{mul}})$.

7.2 System Profiling and Modeling

We conduct real experiments to investigate the BLE RSSI, transmission delay and energy consumption, and the fog image processing delay in the multi-camera EFCam. The measurements are used to fit the latency and energy consumption models for driving the offline training of the multi-camera DRL agent. In the experiments, each camera captures images, and then uses the autoencoder to generate the image representation data. The image data are divided into multiple 400-bytes packets for transmission to the fog node. The number of cameras varies from 1 to 3. Fig. 10(a) shows the per-packet average delay over a period during which each camera continuously transmits a total of 10,000 packets to the fog node. The packet transmission delay increases with the number of cameras. This is because multiple cameras compete for the wireless channel. Thus, with more cameras, it takes a longer time for each camera to complete the transmissions of a certain number of packets.



(a) Reward. (b) Accuracy. (c) Latency. (d) Power with idle. (e) Power with deep sleep.

Fig. 12. DRL training and execution results of 2-camera EFCam under various settings of τ. (a) Traces of training reward ; (b)-(e) Execution results. The dotted line in (c) represents the l_{th} = $600s$; λ_1 = 1 and λ_2 = 6.

From Fig. 10(a), the packet transmission delay can be modeled as a function of m, denoted by $f(m)$. We fit a linear function to model the $f(m)$, as represented by the line in Fig. 10(a). The transmission delay is also affected by the wireless channel condition. As mentioned in §5.2.1, we fit different Gaussian distributions to model the packet transmission latency $l(\eta)$ when a single camera transmits the data packet to the fog node when the RSSI is η. Let $l_{\text{mul}}(m, \eta)$ denote the packet transmission latency when m cameras simultaneously transmit the data packets to the fog node under the RSSI η. We model the $l_{\text{mul}}(m, \eta)$ as a function of m and $l(\eta)$, i.e., $l_{\text{mul}}(m, \eta) = f(m)l(\eta)$. We also adopt the similar approach to model the packet transmission energy consumption. Fig. 10(b) shows the BLE RSSI values from two different cameras to the fog node over 300 seconds. Two BLE connections have the similar RSSI value. Thus, we use the RSSI from a camera's BLE connection as an indicator for the wireless channel condition in the MDP formulation.

The fog node runs m concurrent threads to perform image reconstruction and CNN-based classification for images from m cameras. Let $l_{\text{fog}}(m, u)$ denote the average image processing latency for the fog node to simultaneously process u different images. As m cameras may have different frame rates, we have $1 \le u \le m$. Given a pair of (m, u), we conduct an experiment in which the fog node processes u images at every two seconds over a duration of 1,000. Then, we fit Gaussian distributions to model the $l_{\text{fog}}(m, u)$ based on 500 measured samples of $l_{\text{fog}}(m, u)$. Fig. 11 shows histograms and fitted Gaussian distributions for the $l_{\text{fog}}(m, u)$ with $m = 3$ and $u \in [1, 3]$. From Fig. 11, we can see that the histogram and fitted density function of $l_{\text{fog}}(m, u)$ are different under each setting of (m, u).

7.3 Performance Evaluation

In this subsection, we first present the DRL training and execution results of the multi-camera EFCam. Then, the testbed experiment results are presented.

7.3.1 DRL training and execution

Fig. 12(a) shows the training traces of the average rewards over 800 epochs when the fog node supports two cameras. The different curves correspond to τ settings of 5, 10, and 15 seconds. The accuracy requirement ρ_{req} is set to 0.8 while the latency threshold l_{th} is set to 600 ms. The λ_1 and λ_2 are set to 1 and 6, respectively. From Fig. 12(a), along with the training epochs, the reward always increases and then becomes flat over various settings of τ. The results show that the multi-camera DRL agent converges after certain training epochs.

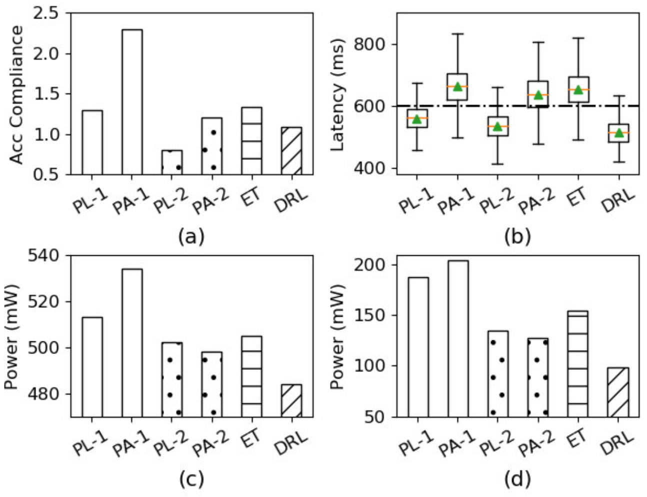


Fig. 13. Results in real experiments with two-camera testbed in office environments. (a)-(d) show the accuracy compliance, latency distribution, average powers with the idle and deep sleep modes. For DRL, λ_1 = 1, λ_2 = 6, l_{th} = 600ms.

Figs. 12(b)-(e) show the accuracy compliance, latency distribution and camera's power over 1,000 adaptation periods, where the period τ varies from 5 seconds to 20 seconds. When the τ is higher than 5 seconds, the DRL agent overprovisions the accuracy performance at the cost of higher latency and power consumption. This is because with a higher τ, the DRL may not track well the variations of the accuracy requirement and the wireless condition. In particular, upon detection of the object in the previous adaptation period, the DRL agent may select a higher frame rate and image resolution such that the appearance of the object in the next adaptation period can be accurately recognized. With long adaptation periods, these settings may be kept when the object has already disappeared in the camera's field of view. As a result, with the larger τ, the camera consumes more power to capture and process unnecessary images. More images also lead to higher transmission and fog image processing latency.

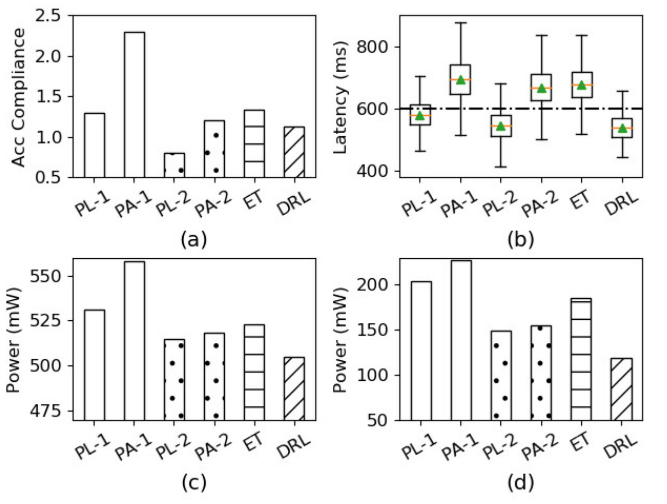


Fig. 14. Results in real factory experiments with two-camera testbed in industrial factory. (a)-(d) show the accuracy compliance, latency distribution, average powers with the idle and deep sleep modes. For DRL, λ_1=1, λ_2 = 6, l_{th} = 600ms.

7.3.2 Testbed experiments

This subsection evaluates the performance of the multi-camera DRL agent on the real testbed in both office and industrial environments. The testbed includes two ESP32-CAMs and a fog node prototyped by a Raspberry Pi. The fog node runs three concurrent Python threads. Specifically, the first two threads maintain the BLE connection, and perform image reconstruction and CNN-based classification for the images transmitted from the two cameras. The third thread runs the DRL agent for adapting the configuration of the two cameras.

In addition to the four baseline approaches of PL-1, PA-1, PL-2, and PA-2 as defined in §6.3, we also compare the performance of our multi-camera DRL approach with an event-triggered (ET) approach. Specifically, we follow the motion-triggered configuration adaptation of the existing commercial security cameras [11], [17] to design the ET approach. The camera captures an image with the highest resolution of 32 × 32 at the beginning of every adaptation period. If the object is detected in the captured image, the camera keeps capturing images at the highest frame rate for the remaining time of the period. Otherwise, the camera is set to the sleep mode to save power.

We conduct experiments in the office lab and factory. Specifically, in the factory, we deploy the testbed on an industrial production line which includes robot arms, ink-filled equipments, and automatic conveyor belts. These equipment simultaneously operates with our testbed during the experiments. Figs. 13 and 14 show the accuracy compliance, box plots for the latency, and average power usage of the two cameras under the proposed DRL and five baseline approaches over an experiment period of one hour in both environments. For the proposed approach, the DRL agent is trained with τ = 5 seconds, λ_1 = 1, λ_2 = 6, and l_{th} = 600 ms. From Figs. 13 and 14, we can see that in both office lab and factory, the PA-1 approach still achieves the highest accuracy compliance but leads to the longest latency and the highest power consumption. The ET approach overprovisions the

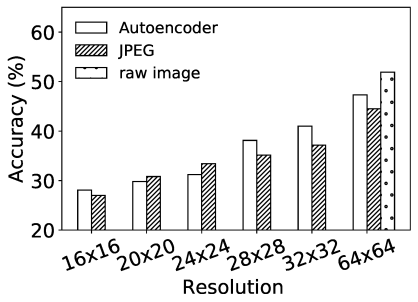


Fig. 15. Classification accuracy with the Tiny ImageNet dataset.

accuracy performance at the cost of longer latency and high power consumption, compared with the PL-2 and PA-2 approaches. Compared with the five baseline approaches, the proposed DRL approach still meets the required image processing accuracy and latency while consuming the least power.

7.3.3 Performance with a general vision application

Previous evaluation results show the performance of the proposed DRL and baseline approaches for a real industrial product object tracking application, which is the primary application focus of this paper. We also conduct simulations based on a large-scale image dataset to further evaluate the performance of all approaches for a general object detection and recognition application. In particular, we use the Tiny ImageNet dataset [31] that consists of 100,000 color images in 200 classes with 500 images per class.

We design a convolutional autoencoder network which consists of a 3-layer encoder and a 9-layer decoder for image feature extraction and reconstruction at the camera and fog node, respectively. ReLUs are used as the activation function for convolution layers in both the encoder and decoder. A CNN consisting of nine convolutional layers with ReLU neurons and a global average pooling output layer is designed to classify the reconstructed images at the fog node. We select 10,000 images in 20 classes of the Tiny ImageNet dataset to train and test the designed autoencoder and CNN. The selected images are divided into the training and testing datasets by a ratio of 9:1. Fig. 15 shows the classification accuracy on the testing images under various image resolutions. The JPEG with quality index of 10 and encoder with the size of filters in convolution and output layers s_f = s_o = 3 are used. The last bar type represents the classification accuracy on the raw Tiny ImageNet images with a resolution of 64 × 64 pixels. From Fig. 15, with the same resolution below 24 × 24 pixels, the JPEG leads to a higher accuracy than that of the autoencoder. When the resolution is higher than 24 × 24 pixels, the accuracy of the autoencoder is higher than that of JPEG. Moreover, without the image pre-processing using the autoencoder or JPEG, the designed CNN can achieve the highest accuracy of 51.4%. Note that the state-of-the-art classification accuracy for the Tiny ImageNet dataset is 67%, which is achieved by a complex CNN called UPANets [31]. Our 9-layer CNN incurs less computation overhead to the fog node and achieves a decent classification accuracy.

To evaluate the performance of the EFCam for adapting the camera configuration, we use the Tiny ImageNet dataset

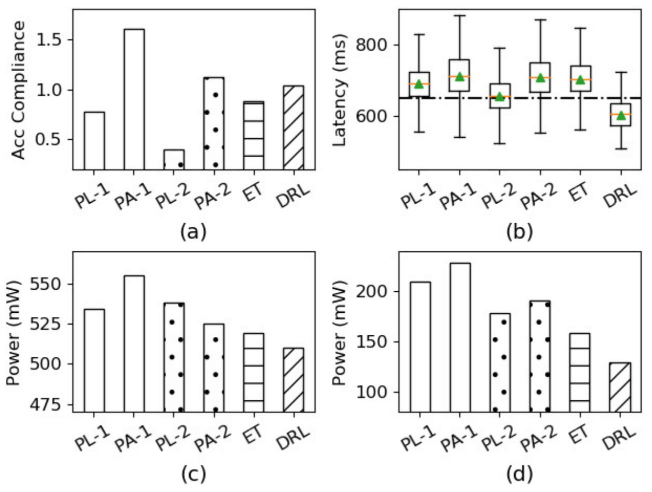


Fig. 16. Results in simulations with a two-camera system using Tiny ImageNet dataset. (a)-(d) show the accuracy compliance, latency distribution, average powers with the idle and deep sleep modes. For DRL, λ_1=1, λ_2 = 10, l_{th} = 650ms. The dotted line in (b) represents the l_{th}.

to generate a trace consisting of 4,000 image frames that simulates the appearance series of the interested objects in the camera's view over time. For each frame, an object appears with a probability of 0.4 and stays in the camera's view for a time period of T that is sampled from the Gaussian distribution with the mean of 5 s and the standard deviation of 2 s. Otherwise, with a probability of 0.6, we use a static background image to represent the camera's view. Note that the appeared object is randomly selected from the Tiny ImageNet dataset.

The traces of object appearances and RSSI values collected in the factory (cf. Fig. 2) are used to train the DRL agent for a two-camera system for 800 epochs, each of which contains 500 adaptation periods of $\tau = 5s$. Fig. 16 shows the execution results of the proposed DRL and five baseline approaches over 1,000 adaptation periods. From Fig. 16, we can see that our DRL approach still meets the accuracy and latency requirements while consuming the lowest power, compared with the five baseline approaches. These results show the effectiveness of the proposed DRL approach for a general object detection and recognition application that requires semantic understanding of more complex patterns.

8 Conclusion

This paper designed and implemented EFCam, an industrial wireless camera system which uses low-power wireless communication and edge-fog computing to achieve cordless and energy-efficient visual sensing. We formulated a configuration adaptation problem that aims to minimize the energy consumption of one or more wireless cameras, while maintaining the visual sensing performance of the deep model at the fog node, under dynamic variations of application requirement and wireless channel conditions. We applied deep reinforcement learning to learn the optimal adaptation policy for the configuration of one or more wireless cameras supported by a single fog node. Extensive evaluation shows that the DRL-based adaptation approach achieves higher accuracy and shorter latency with lower energy consumption for an industrial product object tracking application, compared with five baselines incorporating hysteresis-based and event-triggered adaptation.

Acknowledgment

This study is supported under the RIE2020 Industry Alignment Fund – Industry Collaboration Projects (IAF-ICP) Funding Initiative, as well as cash and in-kind contribution from the industry partner, HP Inc., through the HP-NTU Digital Manufacturing Corporate Lab.

References

[1] S.-H. Ou, C.-H. Lee, V. S. Somayazulu, Y.-K. Chen, and S.-Y. Chien, "On-line multi-view video summarization for wireless video sensor network," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 1, pp. 165–179, 2014.
[2] Y. Koren, X. Gu, and W. Guo, "Reconfigurable manufacturing systems: Principles, design, and future trends," *Frontiers of Mechanical Eng.*, pp. 121–136, 2017.
[3] J. Wang, Y. Ma, L. Zhang, R. Gao, and D. Wu, "Deep learning for smart manufacturing: Methods and applications," *J. Manuf. Syst.*, vol. 48, pp. 144–156, 2018.
[4] https://www.espressif.com/en/products/socs/esp32.
[5] C. Josephson, L. Yang, P. Zhang, and S. Katti, "Wireless computer vision using commodity radios," in *IEEE/ACM IPSN*, 2019, pp. 229–240.
[6] S. Naderiparizi, P. Zhang, M. Philipose, B. Priyantha, J. Liu, and D. Ganesan, "Glimpse: A programmable early-discard camera architecture for continuous mobile vision," in *ACM MobiSys*, 2017, pp. 292–305.
[7] T. Zhang, A. Chowdhery, P. V. Bahl, K. Jamieson, and S. Banerjee, "The design and implementation of a wireless video surveillance system," in *ACM MobiCom*, 2015, pp. 426–438.
[8] S. Naderiparizi, M. Hessar, V. Talla, S. Gollakota, and J. R. Smith, "Towards battery-free HD video streaming," in *NSDI*, 2018, pp. 233–247.
[9] M. Rusci, D. Rossi, M. Lecca, M. Gottardi, E. Farella, and L. Benini, "An event-driven ultra-low-power smart visual sensor," *IEEE Sensors J.*, vol. 16, no. 13, pp. 5344–5353, 2016.
[10] P. Hu, J. Im, Z. Asgar, and S. Katti, "Starfish: resilient image compression for aiot cameras," in *ACM Sensys*, 2020, pp. 395–408.
[11] https://bit.ly/3qEmYjY.
[12] S. Zhou, D. V. Le, J. Q. Yang, R. Tan, and D. Ho, "EFCam: Configuration-adaptive fog-assisted wireless cameras with reinforcement learning," in *IEEE SECON*, July 6-9, 2021, pp. 1–9.
[13] Á. R. Vázquez, R. Galán, J. F. Berni, V. M. B. Sánchez, and J. A. L. Bardallo, "In the quest of vision-sensors-on-chip: Pre-processing sensors for data reduction," *Electron. Imag.*, no. 11, pp. 96–101, 2017.
[14] M. Gottardi, N. Massari, and S. A. Jawed, "A 100μw 128 × 64 pixels contrast-based asynchronous binary vision sensor for sensor networks applications," *IEEE J. Solid-State Circuits*, vol. 44, no. 5, pp. 1582–1592, 2009.
[15] H. G. Chen, S. Jayasuriya, J. Yang, J. Stephen, S. Sivaramakrishnan, A. Veeraraghavan, and A. Molnar, "ASP vision: Optically computing the first layer of convolutional neural networks using angle sensitive pixels," in *CVPR*, 2016, pp. 903–912.
[16] S. Naderiparizi, M. Hessar, V. Talla, S. Gollakota, and J. R. Smith, "Ultra-low-power wireless streaming cameras," *arXiv:1707.08718*.
[17] https://bit.ly/3Icxk0f.
[18] V. Mnih *et al.*, "Human level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[19] https://bit.ly/3nz0gHK.
[20] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *SIGCOMM*, 2017, pp. 197–210.
[21] H. Zhang, A. Zhou, J. Lu, R. Ma, Y. Hu, C. Li, X. Zhang, H. Ma, and X. Chen, "OnRL: improving mobile video telephony via online reinforcement learning," in *ACM Mobicom*, 2020, pp. 1–14.
[22] https://openmv.io/products/openmv-cam-m7.
[23] G. K. Wallace, "The jpeg still picture compression standard," *IEEE Trans. Consum. Electron.*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[24] M. Spörk, C. A. Boano, M. Zimmerling, and K. Römer, "BLEach: Exploiting the full potential of ipv6 over ble in constrained embedded iot devices," in *ACM SenSys*, 2017, pp. 1–14.
[25] C. Alippi, S. Disabato, and M. Roveri, "Moving convolutional neural networks to embedded systems: the AlexNet and VGG-16 case," in *ACM/IEEE IPSN*, 2018, pp. 212–223.
[26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv:1409.1556*, 2014.
[27] A. Krizhevsky, "Learning multiple layers of features from tiny images," Techical Report, 2009.
[28] https://bit.ly/34QqgV9.
[29] C. Shi, R. Wan, R. Song, W. Lu, and L. Leng, "Does the markov decision process fit the data: testing for the markov property in sequential decision making," in *ICML*, 2020, pp. 8807–8817.
[30] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1999.
[31] https://paperswithcode.com/sota/image-classification-on-tiny-imagenet-2.

Rui Tan is an Associate Professor at School of Computer Science and Engineering, Nanyang Technological University, Singapore. Previously, he was a Research Scientist (2012-2015) and a Senior Research Scientist (2015) at Advanced Digital Sciences Center, a Singapore-based research center of University of Illinois at Urbana-Champaign, and a postdoctoral Research Associate (2010-2012) at Michigan State University. He received the Ph.D. (2010) degree in computer science from City University of Hong Kong, the B.S. (2004) and M.S. (2007) degrees from Shanghai Jiao Tong University. His research interests include cyber-physical systems, sensor networks, and pervasive computing systems. He is the recipient of ICCPS'22 Best Paper Award Finalist, SenSys'21 Best Paper Award Runner-Up, IPSN'21 Best Artifact Award Runner-Up, IPSN'17 and CPSR-SG'17 Best Paper Awards, IPSN'14 Best Paper Award Runner-Up, PerCom'13 Mark Weiser Best Paper Award Finalist, and CityU Outstanding Academic Performance Award. He is currently serving as an Associate Editor of the ACM Transactions on Sensor Networks. He also serves frequently on the technical program committees (TPCs) of various international conferences related to his research areas, such as SenSys, IPSN, and IoTDI. He received the Distinguished TPC Member recognition thrice from INFOCOM in 2017, 2020, and 2022. He is a Senior Member of IEEE.

Siyuan Zhou is currently a Ph.D. student in School of Computer Science and Engineering (SCSE) at Nanyang Technological University (NTU), Singapore. He received his B.Eng (2018) in communication engineering from Soochow University in China and M.S. (2019) in electrical engineering from National University of Singapore. He is also working as a Research Associate at HP-NTU Corp Lab at NTU. His research focuses on model-assisted visual sensing.

Joy Qiping Yang is a research assistant in the School of Computing at National University of Singapore. He worked as a project officer at the HP-NTU Corp Lab for two years after graduating from School of Data and Computer Science (B.S.) at Sun Yat-sen University in 2019. His recent research interests include machine learning, statistics, and distribution testing.

Daren Ho is a principal engineer at HP Inc. with strong interest in digital factory transformation. He is presently holding up a few projects within HP-NTU Corp Lab working closely with NTU professors and research staffs to introduce Industrial 4.0 to HP factory in the area of vision inspection for product quality assessment.

Duc Van Le is a Research Fellow at School of Computer and Engineering, Nanyang Technological University, Singapore. Previously, he was a Research Fellow (2016-2018) at Department of Computer Science, National University of Singapore. He received the Ph.D. (2016) degree in computer engineering from University of Ulsan, South Korea, and the B.Eng. (2011) degree in electronics and telecommunications engineering from Le Quy Don Technical University, Vietnam. His research interests include sensor networks, IoT sensing and computing in cyber-physical systems. He is a Senior Member of IEEE.

Supplementary File

Siyuan Zhou, Duc Van Le, Rui Tan, Joy Qiping Yang and Daren Ho

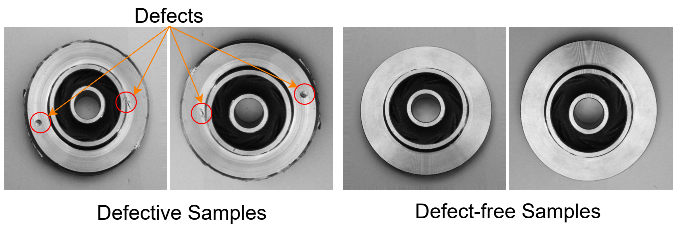


Fig. 1. Some examples from the Casting dataset [1].

This document includes the supplemental materials for the paper titled "Configuration-Adaptive Wireless Visual Sensing System with Deep Reinforcement Learning".

Appendix A System Profiling

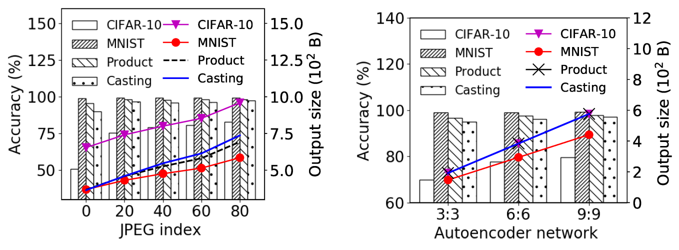
A.1 Profiling Methodology

We investigate how the configuration of EFCam affects various types of the visual sensing tasks. Thus, we conduct the profiling based on four image datasets as follows.

- *MNIST* [2] consists of 60,000 grayscale images of handwritten digits between 0 and 9.
- *CIFAR-10* [3] contains 60,000 color images in 10 classes, such as airplance, bird, car, ship, and etc.
- *Casting* [1] is an industrial image dataset of 7,348 grayscale images capturing top views of submersible pump impeller in a manufacturing casting process. The dataset has two classes: the defective class of images with various types of defects in the pump impellers (e.g., the pinholes, shrinkage and mould material defects), and the defect-free class of images. Fig. 1 shows some examples.
- *Product* is a dataset containing 15,544 images of industrial product parts collected by ourselves in a factory[1]. Specifically, we used the ESP32-CAM to capture an image of product parts moving on a convey belt every one second. The images are labelled with four classes which indicate the number of the product parts appearing in the field of view, which is from 0 to 3.

We use the MNIST and CIFAR-10 datasets to understand the performance of EFCam on general visual sensing applications. The Casting and Product datasets represent the industrial visual sensing applications which are the primary application focus of this paper. We evaluate EFCam in terms of three metrics: the camera energy consumption, visual sensing accuracy and latency. We use a Monsoon power meter to measure the camera's power.

1. Sample images of the product parts are omitted due to the confidentiality request from our industrial partner.



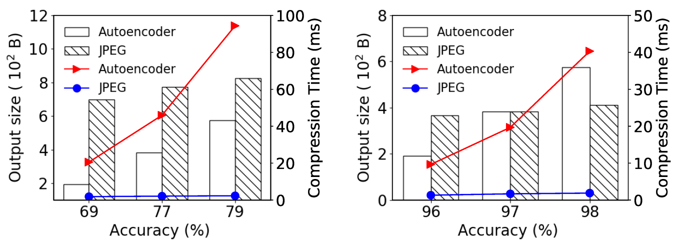
(a) JPEG modes (b) Encoders (s_f:s_o)

Fig. 2. Impacts of image compression on classification accuracy and data output size. Bars and lines present the accuracy and output size.

A.2 Impacts of Image Compression

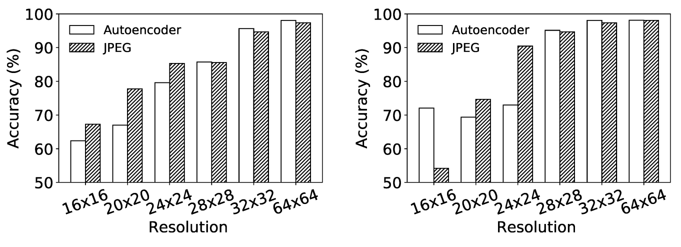
We first conduct experiments to evaluate the impacts of the JPEG and autoencoder modes on visual sensing performance. Specifically, we design and train four convolutional autoencoder networks for image feature extraction and reconstruction for the above four datasets, respectively. Each autoencoder network contains an encoder with three convolution layers and a decoder with nine convolution layers. The rectified linear units (ReLUs) are used as the activation function for convolution layers in both encoder and decoder, while the sigmoid activation is used at the output layers. Specifically, the encoder and decoder are deployed at the camera and fog node, respectively. To further reduce the data transmission overheads, we truncate each single precision floating point number in the output data of the encoder to one byte. For image classification, we train four convolution neural networks (CNNs) for the four datasets. The CNNs for CIFAR-10, MNIST, Casting, and Product datasets are trained with 50000, 60000, 6633, and 10879 samples, respectively. These trained CNNs are tested using 10000, 10000, 715, and 4665 images that are reconstructed by the respective decoder from the image presentation. We set the resolution for the Casting and Product images to be 32×32, and for CIFAR-10 and MNIST images to be $32 \times 32 \times 3$ and 28×28, respectively.

Let s_f and s_o denote the size of filters in convolution and output layers of the encoder, respectively. The s_f characterizes the amount of computing resource required to



(a) CIFAR-10. (b) Product.

Fig. 3. Performance comparison between JPEG and autoencoder. The bars and lines present the output size and compression delay.

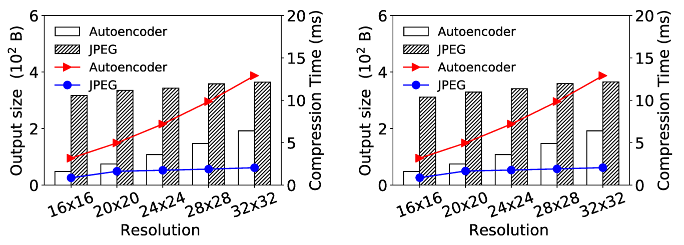


(a) Casting (b) Product

Fig. 4. Impacts of image resolution on image classification accuracy under JPEG-10 and encoder with $s_f = s_o = 3$.

extract the image feature at the wireless camera, while the s_o characterizes the amount of the data transmitted via BLE. We evaluate the impacts of the configuration for JPEG compressor and autoencoder on the classification accuracy and the size of output data. Fig. 2 shows the accuracy and output size under the JPEG with the quality index from 0 to 80 and encoder networks with the size of filters in convolution (s_f) and output (s_o) layers of the encoder from 3 to 9 on the four datasets. Note that for autoencoder network, the s_f characterizes the amount of computing resource required to extract the image feature at the wireless camera, while the s_o characterizes the amount of the data transmitted via BLE. From Fig. 2, the accuracy for the CIFAR-10 dataset and the output size for the four datasets increase with the JPEG index, and s_f and s_o of the encoders. This is because a higher JPEG index and a larger size of the encoder filters reduce the loss of information caused by the image compression and feature extraction, respectively. As a result, the reconstructed images have a higher quality which leads to a higher classification accuracy by the CNNs. Under the same dataset, the output size of the JPEG modes are mostly larger than those of the autoencoder networks.

Fig. 3 compares the performance of the JPEG and autoencoder on CIRFAR-10 and Product datasets. For CIRFAR-10, to achieve the same accuracy, the encoder always outputs a lower data size but requires a longer compression time. However, for the Product dataset, with the accuracy higher 96%, the autoencoder leads to a similar or higher data size, compared with the JPEG. Note that the large output data size usually results in increased transmission delay and energy consumption, which are also affected by the wireless channel condition. Thus, these results imply that the image local processing mode should be configured according to the required accuracy, delay, the energy budget, and the time-varying wireless channel condition.



(a) Casting (b) Product

Fig. 5. Impacts of image resolution on output size and compression time under JPEG-10 and encoder with $s_f = s_o = 3$ The bars and lines represent output size and compression time, respectively.

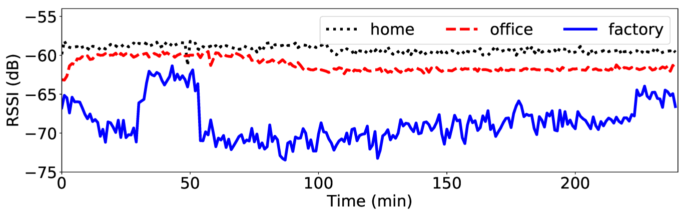


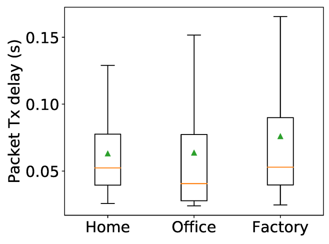
Fig. 6. BLE RSSI traces in different environments.

A.3 Impacts of Image Resolution

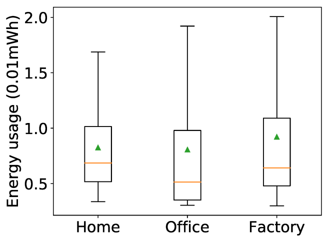
We also conduct experiments to study the impacts of image resolution on the classification accuracy and compression time under the autoencoder networks and JPEG compression. Fig. 4 shows the accuracy under various image resolutions from 16×16 to 64×64 pixels for two industrial image datasets (i.e., Casting and Product). The JPEG with quality index of 10 (denoted as JPEG-10) and encoder with $s_f = s_o = 3$ are used. We test the accuracy on the same CNN network previously trained on 32x32 original images. For images at lower resolutions, we perform image scaling before feeding them into the network. From Fig. 4, the accuracy under both JPEG and autoencoder increases with the image resolution. This is because the image with higher resolution contains more useful visual information, leading to a higher quality of the reconstructed image at the fog node. Moreover, with the same resolution below 24×24 pixels, the JPEG leads to a higher accuracy than that of the autoencoder. When the resolution is higher than 24×24 pixels, the accuracy of the autoencoder is slightly higher than that of JPEG. Fig. 5 shows output size and compression time of the JPEG and autoencoder under various image resolutions. Under the JPEG mode, the output size and compression time remain almost constant when the resolution varies. Differently, under the autoencoder mode, the output size and camera computation time increase with the resolution.

A.4 Wireless Performance Benchmarks

Fig. 6 shows the traces of the received signal strength indicator (RSSI) of the radio signal sensed by the fog node over a time duration of four hours in the three environments. During the experiments, we measure the RSSI every one second. Each data point in Fig. 6 presents the averaged RSSI over one minute. From Fig. 6, the factory environment always has lower RSSIs, compared with the home and office



(a) Packet transmission delay



(b) Packet energy consumption

Fig. 7. Performance of BLE data transmission in different environment conditions. The box, line, triangle, upper and lower whiskers represent middle 50%, median, average, ranges for the bottom 25% and the top 25% of the samples, respectively.

environments. In the factory environment, the RSSI fluctuates from -75 dB to -60 dB over the experiment period of four hours; in the home and office environments, the RSSI is mostly around -60 dB. The reason is that the factory space contains large metal objects such as machines and moving manufacturing components which may cause strong multi-path fading propagation effects. In addition, the size of the home, office, and factory spaces are about $10\,\mathrm{m}^2$, $200\,\mathrm{m}^2$, $1000\,\mathrm{m}^2$, respectively. The reflections from the surrounding walls in a smaller space can typically enhance RSSI. Fig. 6 also shows that the RSSI decreases with the indoor space size.

Fig. 7 shows box plots for the distributions of the round-trip transmission delay and energy consumption of the data packets that are transmitted from the camera to the fog node over a duration of four hours under three environments. Note that the round-trip transmission delay is the time from the camera transmits the application packet to the camera receives the acknowledgement. The packet transmission energy consumption is the total amount of energy that the camera consumes during the round-trip delay. From Fig. 7, the packet energy consumption and delay in the factory environment fluctuates in wider ranges than those under the home and office environments. Moreover, the factory environment has the highest maximum packet transmission delay and energy consumption. These results indicate that the data transmissions in the factory environment are subject to more fluctuating delay and energy consumption, compared with the office and home environments. The reason is that when the wireless channel condition is poor, the camera may need to retransmit the packet for multiple times before the successful delivery. Moreover, with the time-varying and noisy wireless channel in the factory, the number of retransmission times is more variable.

Appendix B
BLE Data Transmission

In the proposed EFCam, the BLE technology is used for the communications between the camera and the fog node due to its low energy consumption [4]. In particular, we implement a connection-based BLE mode in which the camera and the fog node perform as BLE peripheral and central, respectively, as illustrated in Fig. 8. A BLE connection begins with a setup time duration of t_{cs} in which the camera broadcasts three advertising short packets to establish a connection with the fog node. Note that multiple wireless cameras can connect to a single fog node. Then, the communication occurs in non-overlapping connection interval of t_{ci}, during which the camera and fog node exchange packets that may carry application data as shown in Fig. 8. In case no pending application data to be sent, they exchange keep-alive packets containing the mandatory link-layer header only for the maximum connection event duration of t_{ce}. In each connection interval, both devices turn off their radios after all data packets are exchanged or the time budget t_{ce} runs out.

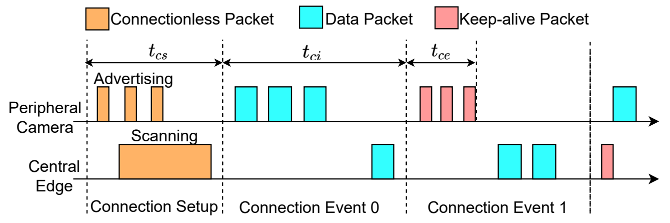
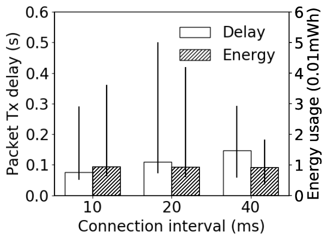
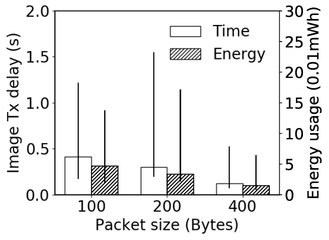


Fig. 8. Connection-based BLE Mode.



(a) Packet size = 400 Bytes.

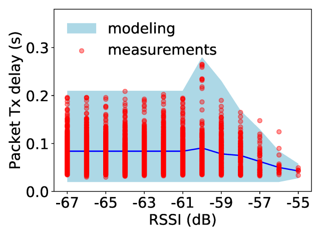


(b) t_{ci} = 10 ms.

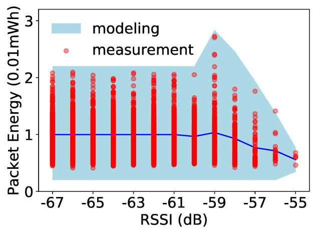
Fig. 9. Impacts of connection interval and packet size on the delay and energy usages for transmitting a packet (a) and the whole 800-byte image (b). The error bars represent minimum and maximum values.

The most important parameter of the BLE connection is the connection interval t_{ci} whose configuration affects the data transmission latency and energy consumption [4]. In general, a short t_{ci} increases the overall data transmission throughput and reduces the latency, which, however, incurs higher energy consumption due to frequent wake-ups. The long t_{ci} has opposite effects on the throughput, latency and energy consumption. Existing study [5] proposed various methods to adapt the configuration for the t_{ci} at run time to maintain a high BLE data transmission quality in dynamic wireless environments. However, the runtime adaptation of the t_{ci} is not advisable for resource-constrained wireless cameras in delay-sensitive industrial visual sensing applications. This is because to configure a new setting of t_{ci}, the camera needs to establish a new BLE connection with the fog node, which lasts for a duration t_{cs} as illustrated in Fig. 8. Our experiments show that the t_{cs} is about 4 seconds. Thus, the adaptation of the t_{ci} at run time causes significant energy consumption and latency in broadcasting advertising packets. Therefore, in EFCam, the t_{ci} is configured with a fixed value.

We conduct experiments to valuate the impacts of the configurations for the BLE connection interval t_{ci} and packet size on the transmission delay and energy consumption in the factory environment. Fig. 9(a) shows the average application packet transmission energy consumption and delay

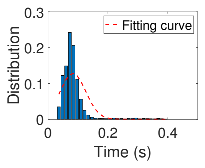


(a) Latency

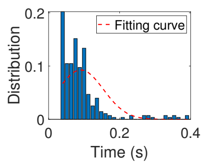


(b) Energy consumption

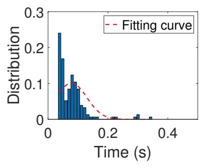
Fig. 10. Impacts of the RSSI on packet transmission latency and energy consumption. The line and belt represent the mean value and the ± 3 standard deviation of the fitted Gaussian distributions, respectively.



(a) RSSI < -60 dB

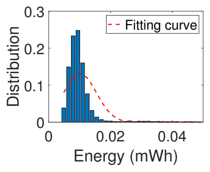


(b) RSSI $= -60$ dB

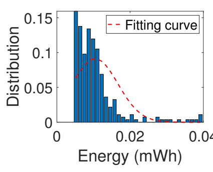


(c) RSSI $= -59$ dB

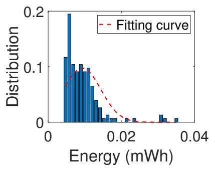
Fig. 11. Histograms and fitted Gaussian distributions for packet transmission latency under different RSSI levels. Results for the RSSI levels greater than -59 dB are not shown.



(a) RSSI < -60 dB



(b) RSSI $= -60$ dB



(c) RSSI $= -59$ dB

Fig. 12. Histograms and fitted Gaussian distributions for the camera energy consumption under different RSSI levels. Results for the RSSI levels greater than -59 dB are not shown.

of 400-byte packets continuously transmitted to the fog node over a duration of 25 minutes when the t_{ci} varies from 10 to 40 ms. As shown in Fig. 9(a), the packet transmission delay increases with the t_{ci}, while the average packet transmission energy consumption mostly remain over variation of t_{ci}. In the connection-based BLE mode, in case no pending packets at the MAC layer at the beginning of a connection event with an interval t_{ci}, the camera transmits the keep-alive packets, during which new application packets arriving at the MAC layer are stored in the buffer and will be sent to the fog node at the beginning of next connection interval. Thus, the average transmission delay of such application packets increases with the t_{ci} since the higher t_{ci} leads to longer waiting times in the buffer. As a result, the average packet transmission delay increases with the t_{ci} as shown in Fig. 9(a). On the other hand, under the GATT protocol, during the time when the camera waits for the acknowledgement of a transmitted packet, it transmits keep-alive packets if a new connection event begins. A shorter t_{ci} results in a higher number of transmitted keep-alive packets. However, the size of these packets is about 10 bytes. Thus, the average packet transmission energy consumption is almost constant over various t_{ci} setttings. Fig. 9(b) presents the image transmission delay and energy consumption when the packet size varies from 100 bytes to 400 bytes. In our experiments, each 800-byte image is divided into multiple packets for transmission. From Fig. 9(b), the image transmission delay and energy consumption decrease with the packet size. From the above results, the connection interval of 10 ms and packet size of 400 bytes are the optimal settings among the tested ones.

From these measurement results, we set the t_{ci} and packet size to 10 ms and 400 bytes in our evaluation experiments.

Appendix C
Modeling Transmission Latency and Energy Consumption

Figs. 10 (a) and (b) show the error bars of the packet transmission delay $l_{p,i}(\eta)$ and energy consumption $e_{p,i}(\eta)$, versus the RSSI η over a duration of four hours during which the wireless camera continuously transmits 400-byte data packets to the fog node in the factory. From Fig. 10, when the RSSI η is lower than -60 dB, the $l_{p,i}(\eta)$ and $e_{p,i}(\eta)$ each follows the same distribution regardless of the RSSI. When the RSSI η is above -60 dB, their distributions vary with the RSSI. Thus, we fit two distributions for the $l_{p,i}(\eta)$ and $e_{p,i}(\eta)$ using all the data with $\eta < -60$ dB. Then, for each η that is greater than or equal to 60 dB, we fit two separate distributions for the $l_{p,i}(\eta)$ and $e_{p,i}(\eta)$. Figs. 11 and 12 show the histograms of real $l_{p,i}(\eta)$ and $e_{p,i}(\eta)$ data and the fitted density functions, when $\eta < -60$ dB, $\eta = -60$ dB, and $\eta = -59$ dB. The histograms for other η levels that are greater than -59 dB are not shown. From Figs. 11 and 12, we can see that the histogram and fitted density function of $l_{p,i}(\eta)$ and $e_{p,i}(\eta)$ are different under each of the RSSI levels. For instance, under the RSSI less than -60 dB, the $l_{p,i}(\eta)$ are fitted to a Gaussian distribution of $N(0.084, 0.042^2)$. When the RSSI is -60 dB and -59 dB, the fitted Gaussian distributions are $N(0.0907, 0.063^2)$ and $N(0.0784, 0.0507^2)$, respectively.

References

[1] https://bit.ly/34QqgV9.
[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv:1409.1556*, 2014.
[3] A. Krizhevsky, "Learning multiple layers of features from tiny images," Techical Report, 2009.
[4] M. Spörk, C. A. Boano, M. Zimmerling, and K. Römer, "BLEach: Exploiting the full potential of ipv6 over ble in constrained embedded iot devices," in *ACM SenSys*, 2017, pp. 1–14.
[5] T. Lee, J. Han, M. Lee, H. Kim, and S. Bahk, "CABLE: Connection interval adaptation for ble in dynamic wireless environments," in *SECON*, 2017, pp. 1–9.