EFCam: Configuration-Adaptive Fog-Assisted Wireless Cameras with Reinforcement Learning

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

***Abstract*—Visual sensing has been increasingly employed in industrial processes. This paper presents the design and implementation of an industrial 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 (i.e., compression or feature extraction) and transmits 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 application requirement and wireless channel conditions. However, the adaptation is challenging due primarily to the complex relationships among the involved factors. To address the complexity, we apply deep reinforcement learning to learn the optimal adaptation policy. 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 four baseline approaches incorporating hysteresis-based adaptation.**

I. Introduction

Computer vision has become an essential component of the automated inspection processes in smart manufacturing systems. Examples include product quality inspection [1], manufacturing system fault diagnosis [2], and activity monitoring [3]. For instance, in Figs. 1(a) and 1(b), a camera is used to assess the wear and tear conditions of gears and count items for inventory checking, respectively; in Fig. 1(c), cameras are installed to monitor the settings/conditions of many legacy ovens that do not have data logging features. In this paper, we aim to design and implement an industrial wireless computer vision system. Without replying on cables for network connectivity and power supply, the wireless cameras offer several benefits such as easy deployment, mobility support, and unobtrusiveness to the ongoing industrial processes. As today's wireless cameras become smaller in form factors, they are promising for swift *ad hoc* deployments in event-based or schedule-based diagnostic tasks, such as the one-off deployment for gear condition assessment illustrated in Fig. 1(a). In the scheme of *reconfigurable manufacturing system* [4] 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* one-off deployments of wireless cameras for configuration validation and calibration are desirable.

This research was conducted in collaboration with HP Inc. and supported by the Singapore Government through the Industry Alignment Fund-Industry Collaboration Projects Grant.

978-1-6654-4108-7/21/$31.00

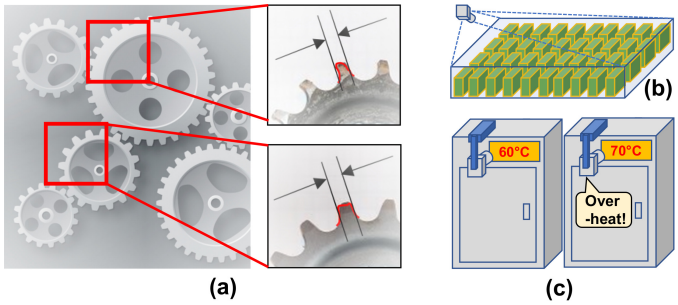


Fig. 1. Industrial vision applications. (a) Wear and tear assessment for gears; (b) Item counting; (c) Legacy equipment setting and condition monitoring. (The subfigure (a) is designed using resources from Freepik.com and Wikimedia Commons.)

An industrial visual sensing system in general involves compute-intensive image processing. Deep learning has been increasingly used for industrial computer vision applications [5]. However, the execution of deep models imposes high demand on computing resources. On the other hand, to achieve the cordless setting for agility, the wireless cameras are often powered by batteries with finite capacities. Energy harvesting is generally infeasible in the indoor environments of factories. 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. In this paper, we consider using a wall-powered wireless fog node with sufficient computing resources to support the front-end wireless cameras in facilitating deep model execution. The geographical proximity of the 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.

We design a system called Edge-Fog Camera (*EFCam*) that leverages wireless camera at the edge and fog node to achieve a cordless and energy-efficient industrial visual sensing system. EFCam uses a battery-powered wireless camera, called ESP32-CAM [6] to perform image sensing, generates smaller representations of the captured images, and transmits the representations to the fog node for advanced processing

using deep models. EFCam is capable of dynamically updating the configuration for various parameters such as frame capture rate, image resolution and the parameters of pre-processing mode of either feature extraction or image compression. The configuration affects EFCam's accuracy, latency, and energy consumption. The pre-processing at the camera can help reduce data transmission energy and latency, but may cause loss of visual sensing accuracy at the fog node.

Existing studies [7], [8] have designed various low-power wireless visual sensing systems. Similar to EFCam, those systems also have system parameters that need to be configured. However, adapting their configuration in response to the variations of application requirements and wireless channel conditions for energy-efficient sensing performance compliance is largely unexplored. On the other hand, the adaptation is desired in dynamic industrial environments. First, the 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 at the 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.

To advance wireless computer vision in industrial settings, in this paper, we propose a novel configuration adaptation scheme for EFCam to achieve good visual sensing performance with minimum 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 dynamic application performance requirements and wireless channel conditions. The configuration includes the image frame rate, resolution, and the mode and parameters of the camera's image pre-processing. The key challenge in solving this 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.

We have implemented the proposed DRL-based EFCam for an industrial product object recognition and tracking system. Specifically, we conduct various experiments to 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 extensively evaluate the performance of EFCam in real experiments and compare the DRL-based configuration adaptation with four baseline approaches incorporating hysteresis-based adaptation. The evaluation results show that EFCam complies with the sensing performance requirements with less camera energy consumption.

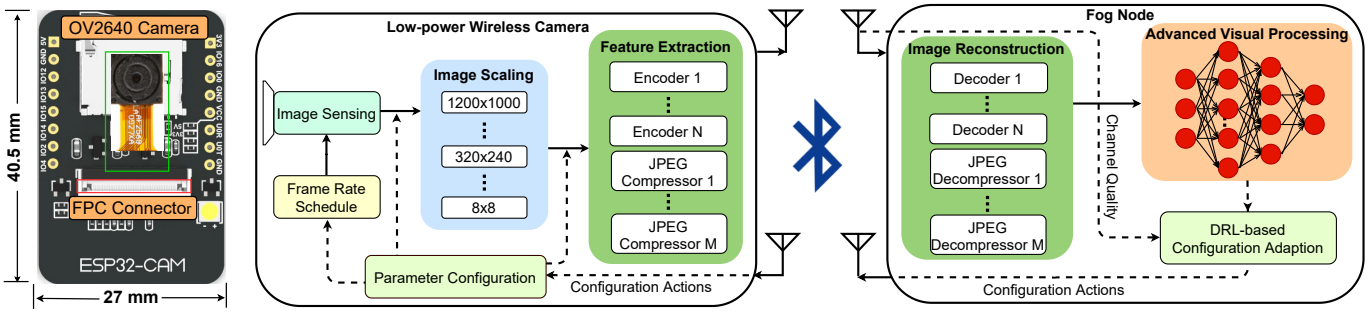
The remainder of this paper is organized as follows. §II reviews related work. §III describes EFCam's design and implementation. §IV studies the impact of EFCam configuration on the system performance. §V formulates the adaptation problem and presents our DRL-based solution. §VI presents the evaluation results. §VII concludes this paper.

II. Related Work

Low-power visual sensing systems have received increasing research attention [7]–[10]. The studies in [9]–[11] 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 capabilities help avoid image processing on redundant visual information, and thus reduce the camera energy consumption. For instance, Vazquez *et al.* [9] 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.* [10] used mixed-signal circuits to extract visual spatial-contrast and perform basic image processing such as motion extraction and background subtraction at the sensor level to reduce the amount of data output.

The works in [7], [8] 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 [7] 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 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.* [7] introduced a pixel-level compression technique to further reduce the size of the image data to be transmitted via the wireless link.

In summary, 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 low-power visual sensing system to minimize the camera energy consumption while the visual sensing application requirements can be met. Similar to our



(a) Wireless camera

(b) Processing pipeline and configuration adaptation

Fig. 2. EFCam system overview. A fog node can support multiple low-power wireless cameras at the network edge.

work, the authors in [7] also proposed to adjust the configuration of camera in terms of the frame rate and resolution to save the camera energy while maintaining high accuracy of a face detection task. However, they used a hysteresis-based control approach which may have inferior performance under dynamic application requirements and wireless channel conditions, as shown by our experiment results in §VI. In this study, we apply the model-free DRL [12] that has shown good performance for various control tasks with high-dimensional state and action spaces as well as complex system dynamics. Our proposed DRL-based approach aims to learn the optimal configuration adaptation policy for the camera under variations of application requirements and wireless channel conditions.

III. EFCam Design & Implementation

Fig. 2 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 and compression, and data transmission. The fog node reconstructs images, performs advanced visual sensing using the deep model, and a DRL-based controller for the system configuration adaptation. In what follows, we describe the design and implementation details of EFCam.

A. Background and Hardware

For the camera, we choose the ESP32-CAM [6], an off-the-shelf battery-powered wireless camera sized 40×27 mm as illustrated in Fig. 2(a). The ESP32-CAM consists of an OV2640 camera module that supports a wide range of image resolutions up to 1600×1200. It is equipped with a low-power 32-bit MCU with a clock frequency up to 240 MHz, 520KB internal SRAM memory, and 4MB external PSRAM memory. To reduce the wireless communication overheads, our design strategy is to use the deep model-based techniques to locally process the raw images at the camera before offloading the remaining computing to 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. Bluetooth Low Energy (BLE) is used for the communications between the camera and the fog node due to its low energy consumption [13].

B. Processing Pipeline and Adaptation

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. 2, the camera first sets a sensing schedule which defines the frame rate of the camera module. The OV2640 camera sensor of ESP32-CAM supports image capture with eight resolution levels between 160×120 and 1600×1200. 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 [14]. 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. For EFCam, we develop multiple deep convolutional autoencoders as well as JPEG compression modes, which can result in different processing/communication overheads and visual sensing performance.

Configuration adaptation: At the fog node, we implement a DRL-based controller which aims to adapt the configuration for the camera's parameters in response to changes of visual sensing performance requirement and industrial wireless channel condition. Specifically, at every adaptation period, the controller obtains the results of the advanced visual processing and the wireless channel condition, then takes actions based on the obtained results. The main objective of the configuration adaptation is to maintain desired visual sensing performance with the minimum camera energy consumption, subject to the application's dynamic performance requirement and time-varying and noisy industrial wireless conditions. In §V, we formally formulate the configuration adaptation problem and

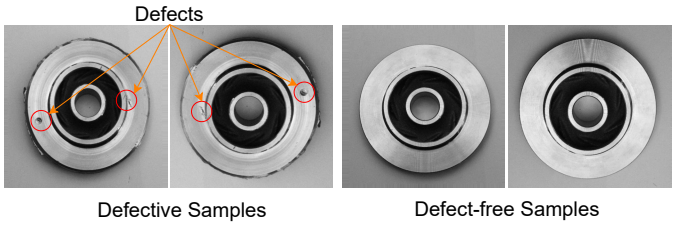


Fig. 3. Some examples from the Casting dataset [15].

present our DRL-based solution. Note that as the deep model of the DRL agent cannot fit into the limited memory of the camera, the adaptation decision is performed by the fog node.

IV. EFCam System Profiling

In this section, we conduct a set of profiling experiments to study the impact of the EFCam configuration on visual sensing performance and energy consumption. The results provide insights to guide the configuration adaptation in EFCam.

A. Profiling Methodology

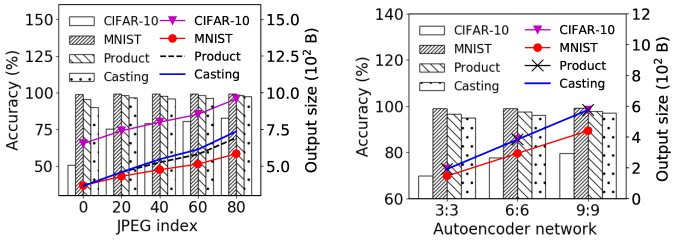
We investigate how the configuration of EFCam affects various types of the visual sensing tasks. Thus, we conduct the profiling based on four image datasets as follows.

- *MNIST* [16] consists of 60,000 grayscale images of handwritten digits between 0 and 9.
- *CIFAR-10* [17] contains 60,000 color images in 10 classes, such as airplance, bird, car, ship, and etc.
- *Casting* [15] 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. 3 shows some examples.
- *Product* is a dataset containing 15,544 images of industrial product parts collected by ourselves 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.

We evaluate EFCam in terms of three metrics: the camera energy consumption, visual sensing accuracy and latency.

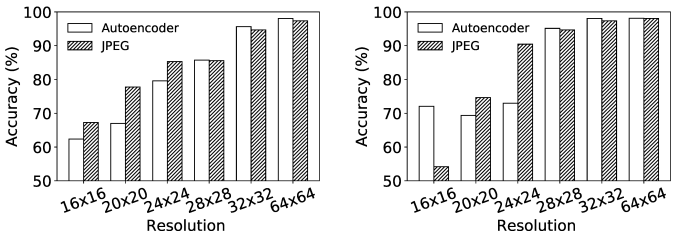
B. Impacts of Image Compression and Resolution

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 encoder and decoder, while the



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

Fig. 4. Impacts of image compression on classification accuracy and data output size. Bars and lines present the accuracy and output size, respectively.



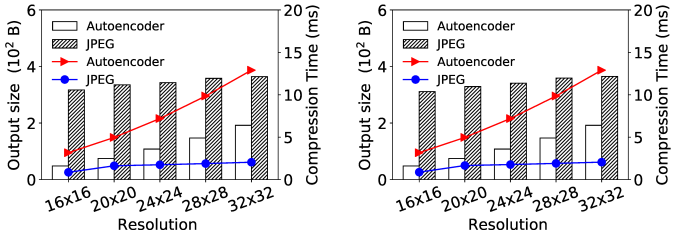
(a) Casting (b) Product

Fig. 5. Impacts of image resolution on image classification accuracy under JPEG-10 and encoder with $s_f = s_o = 3$.

sigmoid activation is used at the output layers. For image classification, we train four different 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 evaluate the impacts of the configuration for JPEG compressor and autoencoder on the classification accuracy and the size of output data. Fig. 4 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. 4, 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. Moreover, the accuracies for the MNIST, Casting, and Product are mostly constant across various JPEG and autoencoder modes. Under the same dataset, the output size of the JPEG modes are mostly larger than those of the autoencoder networks.

We also perform experiments to study the impacts of image



(a) Casting (b) Product

Fig. 6. 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.

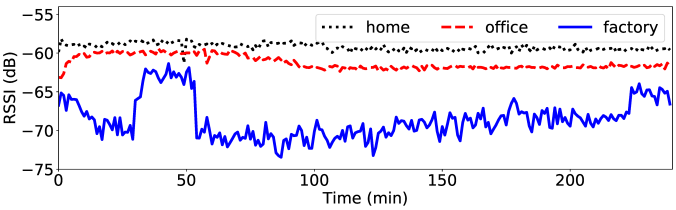


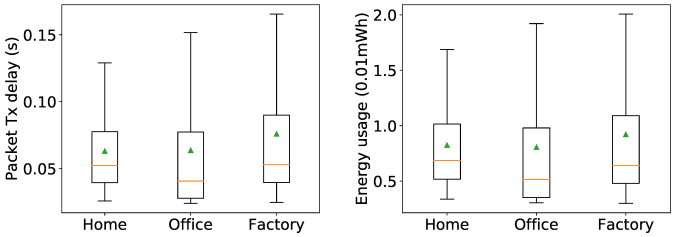
Fig. 7. BLE RSSI traces in different environments.

resolution on the classification accuracy and compression time under the autoencoder networks and JPEG compression. Fig. 5 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. From Fig. 5, the accuracy under both JPEG and autoencoder increases with the image resolution. This is because the image with a 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, 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. 6 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.

C. Wireless Link Quality and BLE 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-byte application data packets to the fog node. 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 with a line of sight. We perform the data transmission experiments in three environments which are factory, home, and office.

Fig. 7 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.



(a) Packet transmission delay (b) Packet energy consumption

Fig. 8. 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.

During the experiments, we measure the RSSI every one second. Each data point in Fig. 7 presents the averaged RSSI over one minute. From Fig. 7, the factory environment always has lower RSSIs, compared with the home and office 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 the high path loss of radio waves and strong multipath fading propagation effects. In addition, the size of the home, office, and factory spaces are about 10 m^2, 200 m^2, 1000 m^2, respectively. The reflections from the surrounding walls in a smaller space can typically enhance RSSI. Fig. 7 also shows that the RSSI decreases with the indoor space size.

Fig. 8 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. The packet transmission energy consumption is the total amount of energy that the camera consumes during the round-trip delay. From Fig. 8, 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.

V. DRL-based Adaptation

From §IV, the resolution, local processing method (JPEG compression or autoencoder networks), and wireless channel condition are the main factors affecting the image processing latency and energy consumption of EFCam. In addition, the image frame rate also greatly affects the camera energy con-

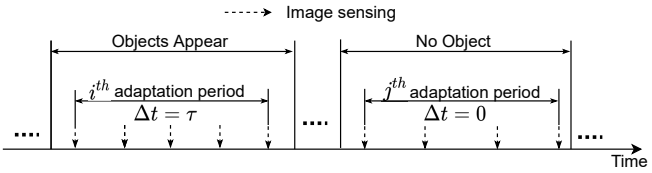


Fig. 9. Image sensing workflow for industrial product object tracking.

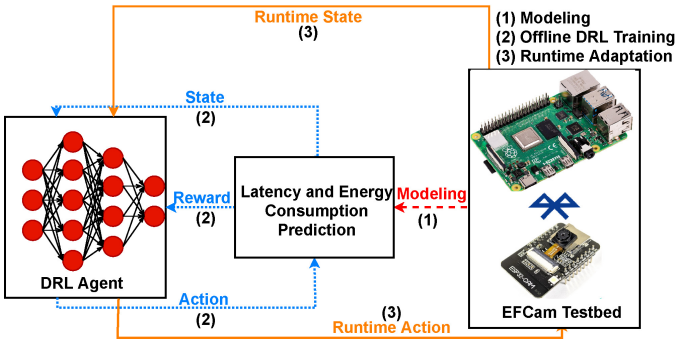


Fig. 10. Workflow of DRL-based adaptation.

sumption. In this paper, we focus on adapting the configuration for these system parameters to minimize the camera energy consumption while maintaining the desired image processing latency and visual sensing accuracy, under dynamic variations of wireless channel condition and application requirements. In what follows, we formulate the configuration adaptation problem as a Makov decision process (MDP) and propose a DRL-based solution.

A. MDP Formulation

Time is divided into intervals with identical duration of τ seconds, which is referred as *adaptation period*. The configuration adaptation is performed at the beginning of every adaptation period, called a *time step*.

System state: The system state, denoted by x, is a vector $x = [\xi, \eta]$, where ξ represents the image processing result and η is the RSSI of the wireless channel at the current time step. In our case study application of industrial product object tracking, the $\xi \in \{0, 1\}$ indicates whether an object is detected in the last frame captured in the previous 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 a, is a vector $a = [f, r, c]$.

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

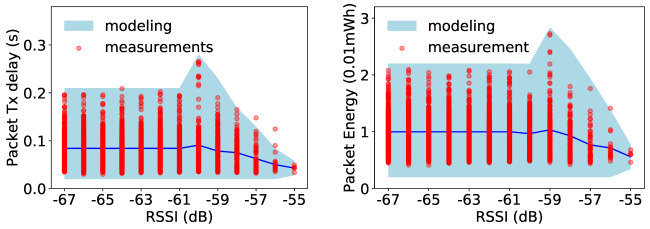
$$p(x, a) = \lambda_1 \cdot \mathcal{N}(l_{\max}(x, a) - l_{\text{th}}) + \lambda_2 \cdot \mathcal{N}(\phi_{\text{req}} - \phi), \quad (1)$$

where l_{th} is the upper bound threshold for the maximum latency, ϕ_{req} is the number of correctly recognized images required over the current period of τ, λ_1 and λ_2 are configurable weights, $\mathcal{N}(x)$ is a normalization process, i.e., $\mathcal{N}(x) = \frac{\max(x,0)}{x_{\max}}$. From the definition of $p(x, a)$, if the maximum latency $l_{\max}(s, a)$ does not exceed the l_{th} and the $\phi(x, a)$ is higher than ϕ_{req}, which means that no delays exceed the threshold and enough images are captured and recognized correctly, no penalty will be applied. The immediate reward, denoted by $r(x, a)$, is defined as $r(x, a) = -e_\tau(x, a) - p(x, a)$. Thus, the reward is defined based on the weighted sum of the energy consumption and the degree of violating of the latency and accuracy requirements.

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 ϕ_{req} as the required number of correctly recognized images in the current period of τ. It is calculated as $\phi_{\text{req}} = 1 + \rho_{\text{req}} \cdot \Delta t$, 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. 9 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 ϕ_{req} in the two adaptation periods are $1 + \rho_{\text{req}} \cdot \tau$ and 1, respectively. The ϕ_{req} is equal to or higher than one to make sure that at least one image is correctly recognized in an adaptation period.

B. Approach Overview

The main objective of the above MDP problem is to find an adaptation policy that determines a based on x to maximize the expected reward over a long run, i.e., $\mathbb{E}[r]$. In general, it is difficult to design a closed-form adaptation policy to maximize $\mathbb{E}[r]$ because the state evolution of the system (i.e., ξ and η) has complex dynamics. To deal with this challenge, we adopt the DRL to learn the optimal configuration adaptation. Typically, DRL agent learns the configuration adaptation policy during the online interactions with the camera system. However, for the formulated configuration adaptation problem, the online DRL learning scheme has the following two issues. 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 infeasible. The camera is not capable of metering its power in real time, which requires an external power meter. Moreover, the image classification accuracy cannot be obtained during the online learning due to the lack of ground truth labels. To



(a) Latency (b) Energy consumption

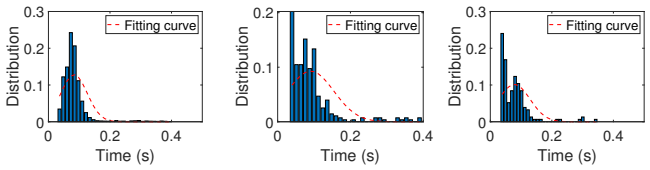
Fig. 11. Impacts of the RSSI on packet transmission latency and energy consumption. The line and belt represent the mean value and the ± 3 standard deviation of the fitted Gaussian distributions, respectively.

address these issues, we adopt an offline training approach as illustrated in Fig. 10, which consists of three steps. First, we model the image processing latency and energy consumption based on real data traces collected from the deployment field. Second, we use the real data traces and models built in the first step to drive the offline training of the DRL agent. Third, after the completion of the offline training, the DRL agent is commissioned to adapt the configuration of EFCam for industrial visual sensing tasks.

C. Modeling Latency and Energy Consumption

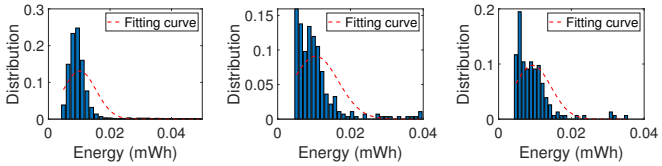
This section models the image processing latency and camera energy consumption. The resulted models will be used for driving the offline training of the DRL agent. Denote by $l(x,a)$ and $e(x,a)$ the image processing latency and camera energy consumption, respectively, when an action a is taken at state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

Our measurements are shown in Fig. 11. When the RSSI η is lower than -60 dB, the $l_p(\eta)$ and $e_p(\eta)$ 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 $l_p(\eta)$ and $e_p(\eta)$ 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 $l_p(\eta)$ and $e_p(\eta)$. Figs. 12 and 13 show the histograms of real $l_p(\eta)$ and $e_p(\eta)$ 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. 12 and 13, we can see that the histogram and fitted density function of $l_p(\eta)$ and $e_p(\eta)$ are different under each of the RSSI levels. Fig. 11 also shows the mean values and the ranges of ± 3 standard deviations of the fitted distributions under various RSSI levels.



(a) RSSI < -60 dB (b) RSSI $= -60$ dB (c) RSSI $= -59$ dB

Fig. 12. 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. 13. 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.

D. Offline Training of DRL Agent

We adopt the learning framework in [12] to train offline a deep Q-network (DQN) for the configuration adaptation agent to capture a good policy to address the problem formulated in §V-A. Specifically, the DQN is trained through interacting with a simulated EFCam environment for N episodes, each of which consists of T time steps. The simulation is driven by real data traces. An episode starts with a state chosen randomly from the training data that include real traces of the RSSI and time-series object images. Then, at the k_{th} time step, an action $a[k]$ is selected for state $x[k]$ according to the ε-greedy algorithm. Given the selected action $a[k]$, the images from the image traces are fed to compression or feature extraction module. Then, the CNNs are used to classify the reconstructed images (cf. §IV). The $\xi[k+1]$ is set to the classification result of the last image captured in the current period, while the $\eta[k+1]$ 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 §V-C. The ϕ and ϕ_{req} 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.

VI. Evaluation

This section evaluates the DRL-based configuration adaptation for the vision-based product object tracking application. In what follows, we present the performance of the proposed DRL approach using simulations driven by real data traces. Lastly, we evaluate the performance of the trained DRL agent in real experiments.

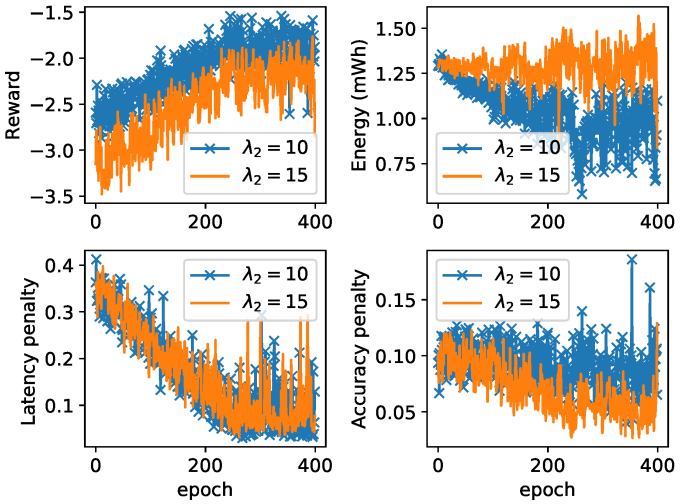
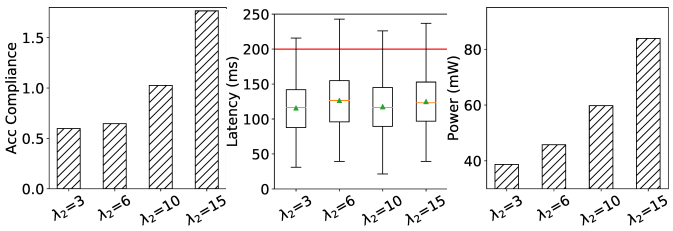


Fig. 14. DQN training results with various λ_2 settings (λ_1=1, l_{th}=200ms).



(a) Accuracy. (b) Latency. (c) Power.

Fig. 15. Execution results under various settings of λ_2 and λ_1 = 1 and l_{th} = 200ms. The red line in (b) represents the l_{th}.

A. Offline DRL Training and Performance

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 $a = [\xi, \eta]$ and chooses the action $a = [f, r, c]$ 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 §IV. 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 5 seconds.

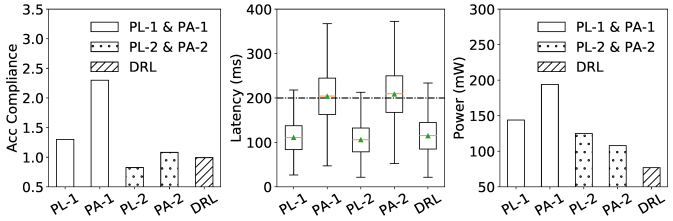
We use our Product dataset to drive the evaluation of the industrial vision-based product object tracking application system. In addition, we use real traces of 15,967 RSSI samples measured in a factory over 4.5 hours, as illustrated in Fig. 7 to model the condition of the industrial wireless environment. 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 200 ms.

2) Training of DRL agent: The offline training is conducted for $N = 400$ 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. 14 shows the DQN training traces of average rewards, average energy, average latency and accuracy penalties when the λ_2 is from 10 to 15 and λ_1 = 10. Along with the training epochs, the reward always increases and then becomes flat under different settings of λ_2. With λ_2 = 10, all the energy consumption, latency, and accuracy penalties have increasing variances but decreasing averages. Differently, when λ_2 = 15, the penalties are close to zero and the energy consumption remains stable. This is because with a higher weight λ_2, the DRL agent is trained toward minimizing the accuracy penalty. We also train the DRL agent under 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 Fig. 14.

3) DRL 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. 15 shows the accuracy compliance (i.e., $\frac{\sum \phi}{\sum \phi_{\text{req}}}$), 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. 15(a), the accuracy compliance increases with λ_2. The DRL approach can adapt the camera's configuration to meet the accuracy requirement, i.e., $\frac{\sum \phi}{\sum \phi_{\text{req}}} \geq 1$ under $\lambda_2 \geq 10$. From Fig. 15(b), the latency is mostly below the l_{th} under various λ_2 settings. Moreover, the camera power consumption increases with λ_2 as shown in Fig. 15(c). 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 processing and data transmission.

B. Real-World Experiments

We conduct a set of experiments to investigate the performance of EFCam in real environments. 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 MB memory which cannot be provided by the ESP32-CAM with 4.52 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 model the



(a) Accuracy. (b) Latency. (c) Power.

Fig. 16. Results in real experiments. For DRL, λ_1=1, λ_2=10, l_{th}=200ms.

product object tracking system. At the fog node, the DRL agent trained with $\lambda_1 = 1$, $\lambda_2 = 10$, and $l_{th} = 200$ ms is used to adapt the configuration for the camera.

We compare our DRL-based approach with four baseline approaches which are variants of a hysteresis-based camera configuration adaptation approach in an existing study [7]. 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 (i.e., feature extraction or JPEG compression modes) as follows:

- 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 the product objects do not appear in the last frame of the previous period (i.e., $\xi = 0$), f is decreased by 1. Otherwise, f is increased by 1 until it reaches to $f_{\max}$.

Fig. 16 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 lowest power. These results imply that the proposed DRL approach can strike good trade-off points to achieve more energy savings and higher sensing performance compliance levels. Specifically, over a period of one hour, our DRL approach can achieve power saving of about 61.1%, compared with the PA-1 approach. The above results show the superiority of our DRL approach, compared with the hysteresis-based baseline approaches.

VII. 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 the wireless camera, 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. Extensive evaluation shows that the DRL-based adaptation approach can achieve better accuracy and latency with lower energy consumption for an industrial product object tracking application, compared with four baselines incorporating hysteresis-based adaptation.

References

[1] N. Neogi, D. Mohanta, and P. K. Dutta, "Review of vision-based steel surface inspection systems," *J. Image Video Process.*, 2014.

[2] V. Chauhan and B. Surgenor, "Fault detection and classification in automated assembly machines using machine vision," *Intl. J. Adv. Manuf. Technol.*, vol. 90, 06 2017.

[3] T. Hussain, K. Muhammad, J. D. Ser, S. W. Baik, and V. H. C. de Albuquerque, "Intelligent embedded vision for summarization of multiview videos in iiot," *IEEE Trans. Ind. Informat.*, vol. 16, no. 4, pp. 2592–2602, 2020.

[4] Y. Koren, X. Gu, and W. Guo, "Reconfigurable manufacturing systems: Principles, design, and future trends," *Frontiers of Mechanical Eng.*, pp. 121–136, 2017.

[5] 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.

[6] https://www.espressif.com/en/products/socs/esp32.

[7] C. Josephson, L. Yang, P. Zhang, and S. Katti, "Wireless computer vision using commodity radios," in *IPSN*, 2019.

[8] 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.

[9] 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.*, 2017.

[10] M. Gottardi, N. Massari, and S. A. Jawed, "A 100 μ w 128 $\times$ 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.

[11] 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.

[12] V. Mnih *et al.*, "Human level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[13] 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 *SenSys*, 2017.

[14] G. K. Wallace, "The jpeg still picture compression standard," *IEEE Trans. Consum. Electron.*, vol. 38, no. 1, 1992.

[15] https://bit.ly/34QqgV9.

[16] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[17] A. Krizhevsky, "Learning multiple layers of features from tiny images," Techical Report, 2009.