IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 8 September 2022, accepted 26 September 2022, date of publication 5 October 2022, date of current version 13 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3212077

RESEARCH ARTICLE

Interactive Machine Learning on Edge Devices With User-in-the-Loop Sample Recommendation

TIANYI LIU AND YUSUKE SUGANO
Institute of Industrial Science, The University of Tokyo, Tokyo 153-8505, Japan
Corresponding author: Yusuke Sugano (sugano@iis.u-tokyo.ac.jp)

This work was supported by JST CREST, Japan, under Grant JPMJCR1781.

ABSTRACT Interactive machine learning (IML) aims to make machine learning an easy-to-use tool for novice users to solve personalized tasks. However, despite the recent popularity of edge AI, research into interactive machine learning on edge devices has not been conducted actively. Existing IML designs cannot be directly applied to small edge devices due to interface and computational resource limitations. In this paper, we propose a method for efficient model personalization on a small interactive object recognition camera device by combining sample recommendations with an IML workflow. The proposed method recommends training data candidates from unlabeled samples in addition to the usual annotation operations. Our method interactively trains a noise filter to handle a noisy sample pool obtained while using the device. The user can indicate whether the recommended sample corresponds to 1) the recommended class; 2) other classes; or 3) noise unrelated to the recognition task by providing ternary feedback. Our system is designed to gradually update both the target classifier and the noise filtering recommendation modules on the basis of feedback. We show that our feedback design achieves more efficient model training while improving system usability through a systematic evaluation and user study using a prototype device.

INDEX TERMS Computer vision, edge machine learning, interactive machine learning, user interface.

I. INTRODUCTION

Since the widespread adoption of deep learning, the domain of machine learning applications has grown in a variety of ways. However, making machine learning accessible to novice users remains a difficult task despite its enormous potential. One of the main goals of interactive machine learning (IML) research is to create environments in which users can interact with machine learning models and create their personalized models [1]. The following features are typically provided by an IML system to allow novice users to build their machine learning models. First, it provides user interfaces and visualization tools to browse and label samples, allowing users to design their training data for the target task [2], [3]. Interactive model validation features are frequently included to allow users to examine the behavior of the trained model and provide additional feedback and training data [4], [5], [6]. The benefits of providing such interactions have been verified by user experiments in previous studies.

However, these IML studies rely heavily on the assumption of rich graphical user interfaces (GUIs), which has some serious drawbacks when applied to a variety of devices. Most previous work, for example, has relied on desktop and object metaphors for intuitive annotation and operations (e.g., drag-and-drop operation to assign training samples to the target class), which necessitates a large screen and pointing interfaces [7], [8], [9], [10]. They also employ rich visualization techniques, such as sample distribution in a high-dimensional feature space, to help users comprehend the model [4], [11], [12], [13]. Furthermore, frequent updates and validation of models are prerequisites for user intervention in IML, and many methods necessitate a certain level of high-performance computational resources. As a result,

The associate editor coordinating the review of this manuscript and approving it for publication was Charalambos Poullis.

designing the interaction and feedback loop for IML systems with limited interaction modalities remains a challenge.

Machine learning applications in edge computing devices have, on the contrary, received a lot more attention in recent years, such as small robots and smart home systems. While most real-world applications use centralized and/or pre-trained machine learning models, allowing users to freely customize the models has a lot of potentials. Consider the case of a device that responds to the user's selection of an arbitrary object category. When using a pre-trained model, the user can only choose an object from a predetermined category, whereas with the IML design, the user can completely customize the target. Furthermore, physical hardware-based IML systems can be more intuitive than GUI-based ones, especially for users unfamiliar with PC operations. If interactive personalization of ML models could be achieved using only physical hardware, it would result in an accessible system that is open to more users and allows for diverse applications. However, in the context of IML research, achieving efficient data annotation and model inspection on small edge devices remains an open challenge. Most of these devices have only a few basic input modalities and a small information display. Therefore, as discussed above, it is not a trivial task to apply existing IML techniques to such devices.

To introduce an IML feedback loop on an edge device with limited interaction modality, we investigate the design of an interactive image recognition camera that allows users to register target categories for classification. In this scenario, adding a feature that allows users to register training images one by one with one of the target category IDs is relatively simple (Fig. 1a). However, due to the input and visualization limitations, it is difficult to extend the interaction to allow for more efficient training data registration. Our basic concept is to suggest training data candidates from the sample pool obtained during device use (e.g., during inference). Although this approach is similar to active learning strategies [14], we must assume that the sample pool will contain significant amounts of noise, such as frames that are irrelevant to the task and frames that have been degraded by motion blur. It is difficult to filter out such noisy frames and find suitable data candidates for user feedback because the classification target task is entirely up to the user.

In this paper, we propose the idea of using user feedback to update not only the classification but also the noise filtering sample recommendation modules. As illustrated in Fig. 1b, the system displays a candidate image with an expected category ID and asks for user feedback. To minimize user burden with limited input, our proposed system uses ternary feedback; 1) *Yes*, the sample belongs to the recommended category, 2) *No*, the sample belongs to other categories, and 3) the sample is *Noise* and not relevant to the classification task. While the *Yes*/*No* feedback can be used to update the classification module, the *Noise* feedback can be used to train a supervised noise filter and helps with the performance of further sample recommendations. In this way, we adapt the IML paradigm to interaction-limited edge devices. Our design requires significantly fewer interaction and visualization modalities than existing IML systems designed for desktop environments. We built a hardware prototype of a Raspberry Pi-based camera device with the proposed feedback design (Fig. 1c). We demonstrate the proposed system's annotation effectiveness and improved user experience through a real-world user study. The user study data is further used to systematically evaluate how our feedback loop design can improve model performance with a limited number of interactions.

The contribution of this work is summarized as follows. First, we propose a novel design of an edge IML system that allows users to iteratively train and inspect user-defined recognition models. To achieve an efficient training loop, we propose to incorporate both user-active sample annotation and machine-active sample recommendation. This work is the first to bring the IML workflow that includes model validation to small edge devices. Second, we proposed an algorithm to update both the classification and noise filtering modules using user ternary feedback. In our IML setting, the additional noise filtering module significantly improves the sample recommendation process. Third, we implemented and used a prototype device to conduct a user study that includes both qualitative and quantitative evaluations of the user experience and algorithm performance. Through the experiments, we demonstrate the advantage of our system design over baseline systems.

II. RELATED WORK

Our research is based on previous work on IML while bridging the gap between machine-centric active learning and human-centric approaches. We also build on previous work on edge machine learning by incorporating an IML perspective.

A. INTERACTIVE MACHINE LEARNING

IML aims to help users create personalized machine learning models to solve personalized tasks [1]. IML is expected to provide easy-to-understand information on model behavior and an intuitive user interface to allow communication between human insight and the ML model for non-expert users without experience and knowledge of programming and machine learning [9], [15]. IML systems have been studied in a variety of application scenarios, including interactive image classification [8], [9], interactive image editing [16], [17], [18] and interactive recommendation system [19], [20], to name a few.

The majority of the previous research has focused on GUI design in order to achieve fast user feedback loops and involve users in the machine learning process [9], [21]. As examples of how to use a rich GUI to enable efficient annotation of large amounts of data [3], [7], [8], [12], [22]. Cui et al. proposed Easyalbum, an interactive photo annotation system based on facial clustering and reranking [7].

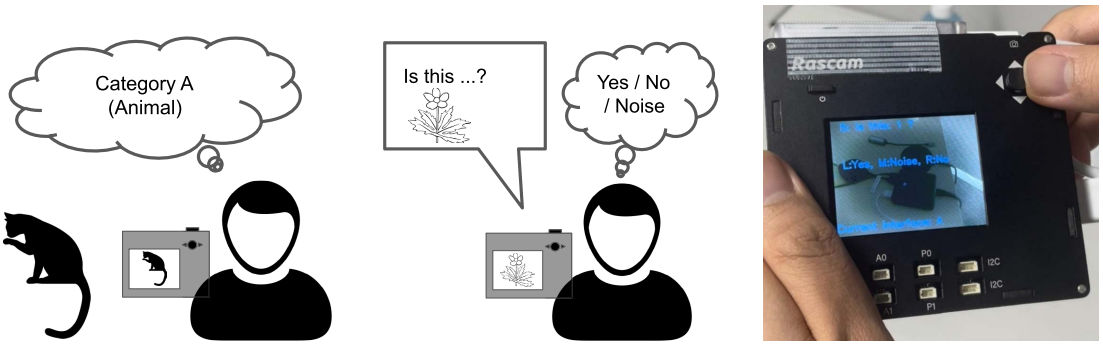


FIGURE 1. The goal of this work is to introduce an IML feedback loop on an edge device with limited interaction modality. The basic idea is to incorporate user feedback to update not only the classification module but also the sample recommendation module with a noise filter.

SHARKZOR is a personalized image classification system proposed by Pirrung et al. featuring a canvas-like interface that allows users to browse and annotate images in the 2D feature space [8]. Ishibashi et al. proposed a similar interactive sound recognition system that displayed sound data using multiple visualization techniques [12]. To achieve a rapid and accurate update of the model, Arendt et al. proposed an interface that communicates with the user through a small set of recommended instances for each class [3]. GUI can also be used to provide visualization for a more intuitive model inspection [2], [4], [5], [10]. Mishra et al. proposed an interactive transfer learning system that provides various design elements in the user interface that visualize the behavior and structure of the model [4]. Jiang et al. proposed their GPU-accelerated GUI for training that allows users to directly manipulate the model parameters [2]. However, as previously stated, these approaches rely heavily on the interaction modalities available on desktop computers, and thus cannot be directly applied to small devices.

As IML continues to evolve, its intersection with image processing and computer vision techniques also attracts increasing attention. In the computer vision community, many interactive systems have emerged with a novel interaction design to improve system performance and usability [23], [24], [25], [26]. Zhang et al. proposed their interactive image segmentation system with inside and outside guidance to achieve efficient segmentation mask generation with a simple interaction design [23]. Zheng et al. proposed a continual learning framework that uses uncertainty maps to provide the more informative guidance and achieve more effective segmentation performance with minimal user effort [24]. Zeng et al. optimized existing interactive image editing that requires an extra mask. The mask-free partial sketch paradigm provides a more intuitive interaction design for the sketch-based image manipulation task [25]. Still, these examples are also designed for desktop GUI environments and are not applicable to small edge devices.

Since such a GUI-rich approach also restricts the use of IML in accessibility applications, some previous work proposed simpler interactions [27], [28]. Kacorri et al. proposed a system in which blind users can take pictures of personal objects for recognition [27]. Ahmetovic et al. proposed Recog, an interactive assistance system that helps blind people capture their personalized data to train their personal object recognition model [28]. Although these approaches allow users to register their own target classes, the scenario is limited to personal object recognition and does not achieve the full IML workflow.

In addition, some systems are proposed to deal with partial user feedback [29], [30], [31]. Joshi et al. introduced the active sample selection and binary feedback (matched or unmatched) method to reduce user effort in the multiclass active learning process [29]. Ngo et al. assumed that full feedback would be too taxing for humans, so they made the human-robot interaction binary. Upper confidence weighted learning (UCWL) was proposed to allow their system to work with binary feedback that may not even be very reliable [30]. However, in the context of edge machine learning, such partial feedback designs have not been thoroughly investigated.

B. ACTIVE LEARNING AND HUMAN-IN-THE-LOOP MACHINE LEARNING

Active learning, in contrast to the human-centric IML approach, can be seen as a machine-centric approach to involving users in the machine learning process. The goal of active learning algorithms is to make the annotation and training process more efficient under a predefined recognition task by querying an annotation candidate from the *oracle* user who knows the ground truth [14]. Therefore, active learning research focuses mainly on the method of finding the most informative annotation candidate [32], [33], [34], [35], [36]. Lewis et al. proposed uncertainty sampling, which is the most common query strategy, querying samples with the highest prediction uncertainty in the sample pool to the oracle and asking for annotations [32]. Expected model change [34] and the expected error reduction [35] are also based on the same idea, and they find the most unfamiliar data by comparing the changes in the model or the behavior of the model after

adding a certain sample to the training set. Active learning algorithms, in contrast, frequently make assumptions about the underlying knowledge about recognition tasks, sample distribution, and user behavior. Simple active learning algorithms cannot handle the recognition task and the distribution of unlabeled samples if they are completely user-dependent.

Several other human-in-the-loop machine learning methods involve the user in the learning process. A representative example is reinforcement learning based on human demonstrations, which replaces the predefined expert reward with human rewards [37], [38], [39], [40]. For example, Liu et al. proposed a deep reinforcement active learning method in which users not only annotate the query sample but also provide the reward to update the reinforcement learning policy [40]. Putting humans in the training loop is also one of the main goals of the crowd-sourcing system to achieve better performance in recent years [41], [42], [43] Rahmanian et al. investigated the design of human-in-the-loop crowd-sourcing, where the user interface significantly affects the performance of workers [43]. These approaches are still machine-centric in the sense that the algorithm is in charge of task definition, which is insufficient in the context of this study.

C. MACHINE LEARNING ON MOBILE/EDGE DEVICES

With the growth of edge computing technology, the concept of edge machine learning promotes the development of research and the emergence of consumer devices, bringing machine learning capability to edge devices [44], [45], [46], [47], [48], [49], [50], [51]. Previous research on edge machine learning focused mainly on improving inference performance by improving inference speed and reducing memory usage [46]. To better accommodate the property of edge machine learning with limited computing resources, recent studies investigate neural network architectures [52], [53], [54] and model compression techniques [55], [56], [57], [58] to provide fast and lightweight models for the edge device.

There have also been many application studies using edge machine learning devices. For example, using surveillance cameras to perform edge video analysis for real-time traffic monitoring is an important application scenario [59], [60], [61], [62], [63]. The machine learning approach is also applied for autonomous driving to perform pedestrian detection [64]. For personal use cases, Liu et al. proposed a food recognition system [65] on edge devices based on cloud training. Weng et al. proposed their optical character recognition system [66] for mobile computing devices. However, most of the applications simply apply pre-trained models locally or in the cloud, and on-device interaction with the machine learning process is still a problem that has not received much attention.

As the range of applications grows, several consumer devices have emerged to improve edge machine learning performance, such as the Canaan Kendryte K series neural network processor (KPU)[1] and the Google Coral edge TPU.[2] Some vision sensors have also been developed that allow developers to register training images for on-device model training, such as an M5StickV[3] and HUSKYLENS.[4] However, their interaction design is still limited to the data annotation step. They allow only the operation of adding training data one by one and do not allow users to inspect the behavior of the model and give feedback. To the best of our knowledge, this is the first scientific study to examine the design of edge IML workflows that include both data annotation and model inspection loops.

III. PROPOSED SYSTEM

Our proposed system combines IML-like interactive data annotation and active learning-like sample recommendation approaches, as illustrated in Fig. 2. The system is made up of two main modules to accomplish this. One is *classification module*, which learns from user-provided data and makes inferences to solve the user's personalized classification task. The other is *recommendation module*, which recommends potentially useful samples from the inference history to users to obtain feedback from the user. The system is designed to operate in two phases, where users can freely switch between them. In *input phase*, users can input new training samples in real-time into the target class to train their personalized classifier and then input live test images into the system to check the performance of the classifier. Meanwhile, the frames recorded in this phase are stored and used in *feedback phase* where users provide *Yes*/*No*/*Noise* ternary feedback. In the following, we describe the process of each phase in detail.

A. INTERACTION PHASES

1) INPUT PHASE

In the first interaction phase, starting from a classifier that makes the prediction at random, the user can first perform **label assignment**. We assume that the user watches the live frame captured by the camera and assigns this frame to one of the user-defined target classes. The captured image $\boldsymbol{X}_i$ and its class ID y_i are given to the classification module as a result of this annotation process.

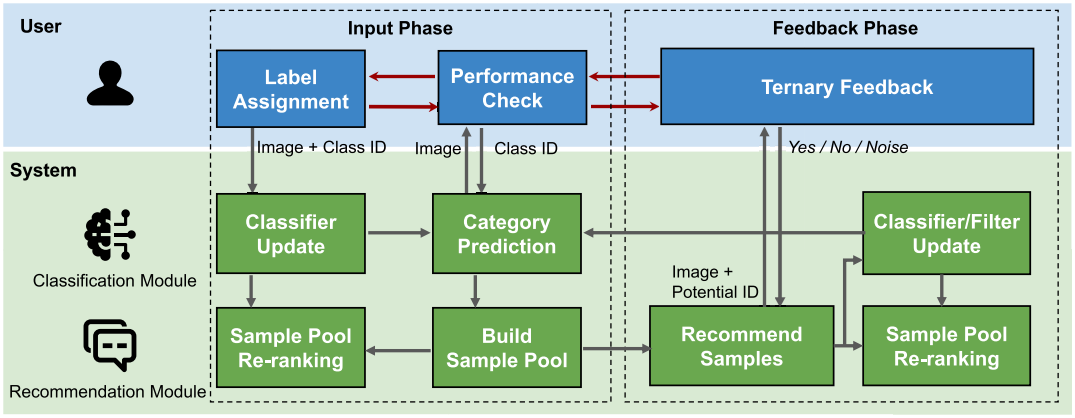
Based on the training data obtained from the user annotations, the classification module performs **classifier update**. We assume that the user can add an unlimited number of classes to the system, and the classifier must be trained from scratch every time to accommodate new classes. The model update is triggered every time N new training data are given in this phase to reduce computational cost. The classification module consists of a feature extractor based on EfficientNet [52] pre-trained on ImageNet [67] and a small classifier using non-linear SVM with the RBF kernel [68]. For each training image $\boldsymbol{X}_i$, the feature extractor first extracts a 1280-dimensional feature vector $\boldsymbol{f}_i$ from EfficientNet before the last fully connected layer. Using the training data obtained

[1] https://canaan.io/product/kendryteai

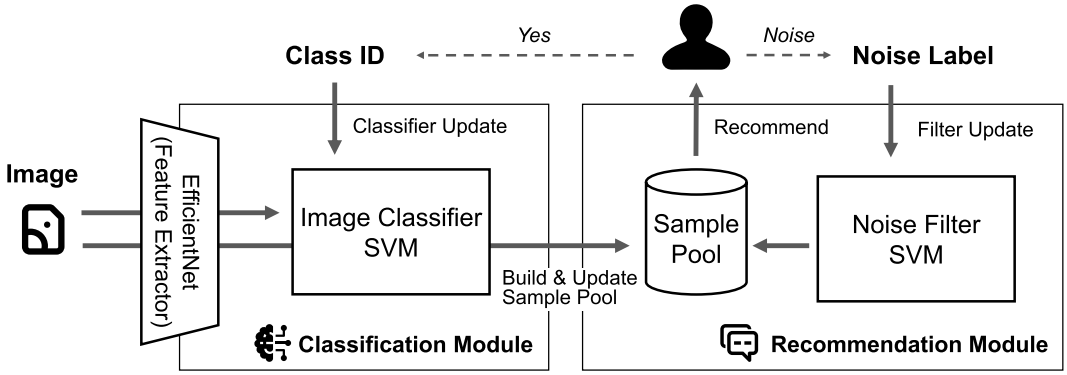
[2] https://coral.ai/products/accelerator/

[3] https://shop.m5stack.com/products/stickv

[4] https://wiki.dfrobot.com/HUSKYLENS_V1.0_SKU_SEN0305_SEN0336



(a) The overview of the system.



(b) The detailed illustration of the system with classification and recommendation module.

FIGURE 2. The overview of the proposed system (a) with the detailed system illustration of the technical blocks (b). The overview of the proposed system (a) with the detailed system illustration of the technical blocks (b). It combines IML-like interactive data annotation and active learning-like sample recommendation approaches as shown in this diagram. The system is divided into two main modules, classification and recommendation modules, and it is designed to work in two phases, the input and feedback phases.

up to that point, the SVM classifier is updated with a fixed regularization parameter C and hinge loss.

After the classifier is trained and starts making predictions with the knowledge learned from the training data, the user can perform the classifier's **performance check** by feeding new images to the system. We assume that the user can use the camera to recognize the input images. Users can subjectively check if the current output is in line with their intention by freely pointing the camera in different directions and observing the live images alongside the classifier's prediction results. Consequently, the classification module must perform **category prediction**. It takes the current image $\boldsymbol{X}_i$ and extracts its deep feature $\boldsymbol{f}_i$ in real-time. The feature is fed into the SVM classifier, and then the SVM outputs its predicted class probability $\hat{\boldsymbol{p}}_i$. The class probability is then used to return the predicted class ID $\hat{y} = \arg\max(\hat{\boldsymbol{p}}_i)$ so that it is shown to the user on the screen.

Through this iterative annotation and inference process, some of the inferred images are stored in **build the samples pool** to be recommended. The recommendation module can rank the sample pool according to the informativeness of each sample. In the proposed system, we treat images with greater prediction uncertainty and task relevance as informative samples, which can be helpful for faster model optimization. The classification module calculates the uncertainty score of each image in the sample pool after the classifier is updated, which is defined as the inverse of the maximum class probability [32]:

$$u_i = \frac{1}{\max(\hat{\boldsymbol{p}}_i)}, \tag{1}$$

where $\hat{\boldsymbol{p}}_i$ indicates the predicted class probability for the i-th image. In addition to the uncertainty score, the ranking further takes into account the task relevance score r_i estimated by the recommendation module. r_i is initialized as 1.0 when the noise filter in the recommendation module has not been trained prior to the first feedback phase. In other words, **sample pool rank** at the beginning only refers to u_i. Once the noise filter is trained, r_i is updated by the normalized prediction probability of the model. Based on these two scores, the module defines the recommendation weight w_i for the i-th sample as

$$w_i = u_i r_i. \tag{2}$$

As stated above, the higher the weight w_i, the more informative the samples are for the current user-defined task.

Ideally, the sample pool would be dynamically ranked based on the progress of model training. However, to save time and resources, we only rerank the sample pool when the user enters the feedback phase. Based on the deep feature of the image, we calculate uncertainty and relevance using the most recent classifier and noise filter.

2) FEEDBACK PHASE

Users are asked if an image belongs to the category of potential ID shown on the text interface when entering this phase. The **sample recommendation** is done by the recommendation module by selecting the first-rank image from the sample pool and showing its potential class ID with the highest probability. The user responds to the recommendation by choosing one answer from *Yes*, *No*, and *Noise*. When the current question user answers with *Yes*, the image is annotated with the current class ID as a label and added to the training set to update the classifier. Following the previous work on active learning with binary oracle [29], [69], the module repeats the recommendation using the class ID with the next highest probability if the user's answer is *No*. If the answer is *Noise*, the image is marked as noise and used to update the noise filter within the recommendation module. Triggered by the update of the classifier or its noise filter, the recommendation module can perform **sample pool re-ranking** in the feedback phase.

The noise filter is implemented as another binary support vector machine using the same feature extractor as the classification module. The filter is trained with valid training samples as positive samples and noise samples as negative samples. Its output is applied to calculate the task relevance score r_i in Eq. (2), which indicates that the current image is closely relevant to the current user-defined task. Specifically, the relevance score of $\boldsymbol{X}_i$ can be calculated as a normalized probability.

$$r_i = \sigma(\hat{q}_i) = \frac{1}{a + e^{-b\hat{q}_i + c}}, \tag{3}$$

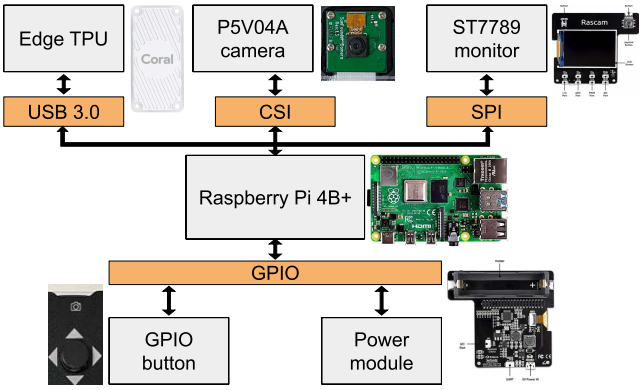
where $\hat{q}_i$ indicates the probability that the image $\boldsymbol{X}_i$ belongs to the positive sample category. $\sigma(\cdot)$ is a sigmoid function parameterized with a, b, and c.

The noise filter is updated every time M new noise samples are given to reduce computational costs. Similarly, the classifier is updated when L new sample images are annotated as valid samples. If the user exits the feedback phase and returns to the input phase before specifying L sample images or M noise images, the system updates both the classifier and the filter with all available training data.

B. PROTOTYPE DEVICE

Based on the system design described above, we build a hardware prototype as shown in Fig. 3a. Our prototype is built based on the SunFounder Rascam,[5] a camera kit designed for Raspberry pi 4B+.[6] Since our system utilizes a CNN-based feature extractor, we also use the Coral USB Accelerator[7] for fast deep feature extraction.

(a) The hardware implementation of our prototype. Our prototype is built based on a camera kit designed for Raspberry pi 4B+ and the Coral USB Accelerator.



(b) System Diagram of our prototype hardware

FIGURE 3. The hardware appearance and diagram of our prototype.

1) HARDWARE DESIGN

Figure 3b shows the detailed diagram of the system. The Raspberry pi 4B+ features a Broadcom BCM2711 1.5 GHz quad-core 64-bit CPU with Cortex-A72 processors, as well as a boosted GPU that is capable of running lightweight machine learning algorithms such as SVMs and random forests. A USB 3.0 port that connects the Coral Edge TPU to the Raspberry Pi. The shutter button and joystick on the Rascam PCB board are controlled via a GPIO interface. A 2.4-inch TFT screen powered by ST7789 with a resolution of 320 × 240 resolution is also connected to the development board via the SPI protocol and the corresponding ports. The CSI port is used to connect the P5V04A camera module. The power supply module is powered by a 18650 battery, which feeds power to the development board through a 5V DC header with a minimum current of 3A.

2) USER INTERFACE

We created the user interface using only Rascam hardware, which consists of one shutter button and one joystick, to achieve the required interactions. The flow of user

[5]https://www.sunfounder.com/products/rpi-camera-kit

[6]https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

[7]https://coral.ai/products/accelerator/

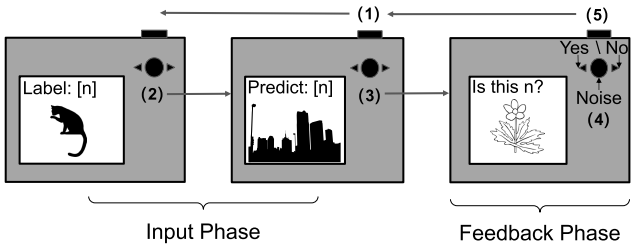


FIGURE 4. The overall usage flow of our IML prototype on the edge device. During the input phase, (1) By pressing the shutter button, users can capture a training sample, and (2) assign the label by selecting the target ID with the joystick. (3) Users can enter the feedback phase by pressing the joystick, (4) where users provide feedback by operating the joystick. (5) They can leave the feedback phase by pressing the shutter button.

interaction in our prototype system is depicted in Figure 4. When the system is first turned on, users are taken to the input phase, where the system attempts to make predictions. Users can capture a training sample by pressing the shutter button (Fig. 4 (1)). Label assignment is accomplished by using the joystick to select the target ID (left / right) and then pressing the joystick to confirm the decision (Fig. 4 (2)).

After the input phase, users can choose to press the joystick (Fig. 4 (3)) to enter the feedback phase. As illustrated in Fig. 4 (4), users can provide answers to the system's question by operating the joystick to the left, right, or down, which corresponds to *Yes*, *No*, and *Noise*, respectively. Finally, users can press the shutter button (Fig. 4 (5)) to exit the feedback phase.

3) SOFTWARE IMPLEMENTATION DETAILS

We built our system based on Python with the default Raspberry Pi OS. The TPU unit is used by the CNN feature extractor, where the quantized model was obtained from pre-trained weights using the TensorFlow Lite converter. The classifier and noise filter were implemented using the scikit-learn library [70] with the same hyperparameter settings. The penalty coefficient for the SVM classifier was set to $C = 1$ and the tolerance for stopping the criterion is set to $1e - 3$. The coefficient γ for the RBF kernel is set to $\gamma = 1/(1280\sigma)$, where σ denotes the variance of 1280-dimensional feature elements from all training data. The default settings were used for the other parameters. The parameters of the sigmoid function are set as $a = 1$, $b = 10$, $c = 3$ to calculate the relevance score. The threshold parameters N, M, L for model updates are set differently for each experiment and are described in the following sections.

IV. USER STUDY

In this section, we assess the usability of our ternary feedback design. The additional option of *Noise* to the ternary feedback design increases the complexity of the operation, potentially resulting in a poorer user experience. We conducted a real-world user study to compare the usability of our system with the baseline system to investigate this issue. We also examine whether interactive noise filtering improves the recommendations and user experience.

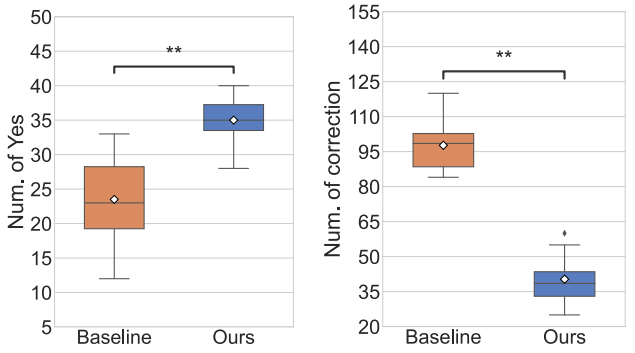
We assume a simplified use case scenario in which the user first defines all target objects and assigns labels in one room. The user moves the device to a different room to perform a qualitative performance check before switching to the feedback phase to review the system recommendations. As a result, the sample pool only contains images taken in environments other than those used in the initial training data.

We created a baseline system using active learning with binary feedback for comparison. In the same way that our proposed system receives the image and the associated uncertainty information in the input phase, the baseline system does as well. However, it interacts with users in binary form (*Yes* or *No*), where the classification module can receive new training data if the user answers the question with *Yes*. As a result, in the baseline system, noise filtering and sample pool re-ranking based on task relevance do not occur, and the weight w is always equal to u. Therefore, even for task-irrelevant samples, this baseline continues to request binary feedback (if ranked higher in the recommendation phase). The sample is ignored only if the user did not accept any of the potential IDs.

A. PROCEDURE

Participants in the user study were recruited on the university campus by posters and mailing lists. We provided only a brief overview of these announcements, and none of them knew the details of the experiment in advance. Seven male and three female participants aged 23 to 28 years (M = 25, SD = 1.99) participated in the study. Nine of them had a basic knowledge of machine learning and/or computer vision, but none had experience in developing edge devices or interactive machine learning systems. A detailed tutorial on the system was given to each participant before the user study, and they were familiarized with the system and the basic concept of IML through hands-on experience. During the experiment, the main objective of the participants was to identify three categories of objects: *Electronic device*, *Daily necessity*, and *Stationery*. The participants' judgments were used to determine which category a particular object belongs to.

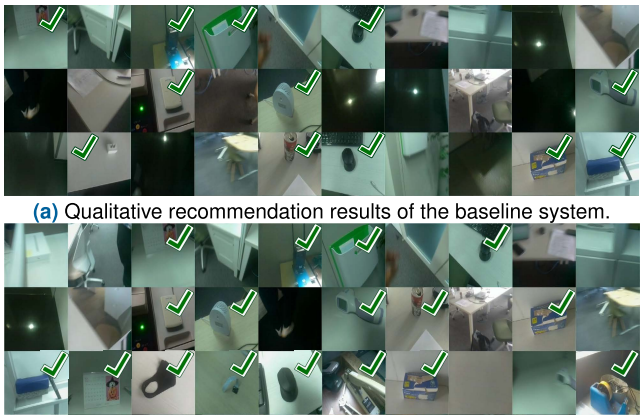
Because the only difference between the proposed and baseline designs is in the feedback phase, each participant only went through the training and inference phases once. The participants used the device to freely collect initial training data in one room for 5 minutes as a first step. The system operates in the input phase, and the classifier is updated with these annotations. Subsequently, the participants were moved to another room and given 10 minutes to perform inference by pointing the device to some new objects. The sample pool is newly constructed during this inference step, and participants then provide feedback using the two interfaces, in turn, with the same data. Each participant was instructed to provide feedback on 50 images using both interfaces, where the order of using two interfaces was randomized. During



(a) Number of *Yes* operations during 50 recommendations

(b) Number of *No*/*Noise* operations during 50 recommendations

FIGURE 5. The comparison between different feedback designs in terms of the number of *Yes*/*No*/*Noise* operations users made to 50 recommended images. Each box plot corresponds to each design that denotes a summary of the data. The star annotation indicates statistical significance.



(a) Qualitative recommendation results of the baseline system.

(b) Qualitative recommendation results of our system.

FIGURE 6. Example recommendation results in the user study. Each image shows an image recommended to one participant in the user study, and the green check mark indicates that the image was confirmed as a valid sample.

this step, we only update the noise filter with threshold $M = 3$ to strictly evaluate the feedback scheme. The classifier is updated only after the feedback phase ($L = 50$). After that, as the final step, participants return to the inference phase to assess the model performance qualitatively.

We saved all time-stamped interaction logs, input data, and the sample pool they created throughout the session. After completion of the task, participants were asked to complete a questionnaire with the following questions on a 5-point Likert scale to compare the two interfaces and provide their feedback (**Q1** - **Q5**) and comment with as many descriptive expressions as possible on subjective questions (**Q6** - **Q9**):

- **Q1.** The usage flow was intuitive and easy.
- **Q2.** The recommendation in the feedback interface helped annotate some ignored potential data.
- **Q3.** I could quickly answer the questions raised by the system.
- **Q4.** Too many useless recommendations occurred, which influence my interaction efficiency.
- **Q5.** Such an IML system with a feedback interface can make model refinement more convenient.
- **Q6.** Was there a difference between the two interfaces and what was your impression?
- **Q7.** Do you think that the model performance was improved after you provide feedback?
- **Q8.** Was this device more efficient than the existing GUI-based IML system as we introduced, and what was your impression?
- **Q9.** Was there anything inconvenient about this system?

Furthermore, they were asked to provide their 5-point Likert scale preference score and complete the questionnaire based on the NASA-TLX [71] evaluation for each design.

B. USER BEHAVIOR

Fig. 5 shows the difference between the two systems in terms of the number of user operations. Statistics of the number of responses *Yes* made during the feedback phase are shown in Fig. 5a. Each box plot corresponds to each design of the system. The median is represented by the middle line of each box, and the mean value of the data is represented by the white marker on each plot. The whisker visualizes the minimum and maximum, while the length of the box represents the range of interquartile. On average, the number of responses *Yes* for the proposed and baseline systems is 35.0 and 23.5, respectively. There is a statistically significant difference between the two systems ($p < 0.01$, Wilcoxon signed-rank test). Furthermore, the data has a shorter interquartile range and a shorter whisker range, indicating a better result with a less scattered data distribution. On the contrary, the statistics of *No* and *Noise* operation required by different systems to handle the 50 recommended images are shown in Fig. 5b. On average, the baseline system required 97.7 corrections, while our system required only 40.3 corrections in total to handle the 50 recommended images. The mean value of our system is significantly lower than that of the baseline system ($p < 0.01$, Wilcoxon signed-rank test). Significantly lower operating demand and higher user acceptance (*Yes*) demonstrate that our system provides better recommendations with higher quality.

Figure 6 shows some qualitative recommendation results obtained from a participant. Each image in the figure represents an image recommended to the participant. Images confirmed as valid samples by the participant are indicated by the green check mark. Images were chosen at random and arranged in the order in which the recommendations were given (from left to right, from top to bottom). Using the baseline system (Fig. 6a), 12 valid samples were recommended from these 30 images. The baseline system, for example, provides more irrelevant images, such as the floor and corridors. These recommended noise images require users to provide many responses *No* to skip. On the contrary, 17 of them were valid samples using our proposed system (Fig. 6b). The diversity of the valid recommendation also improves as the number of noise images decreases. As shown in the third

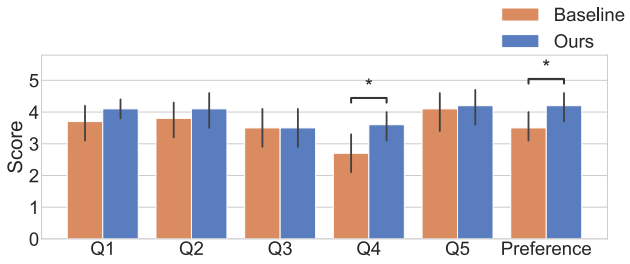


FIGURE 7. Average rating between 2 feedback design and their preference score. Each bar shows the average rating of all participants, and the error bars show the standard deviations. The star annotation indicates statistical significance.

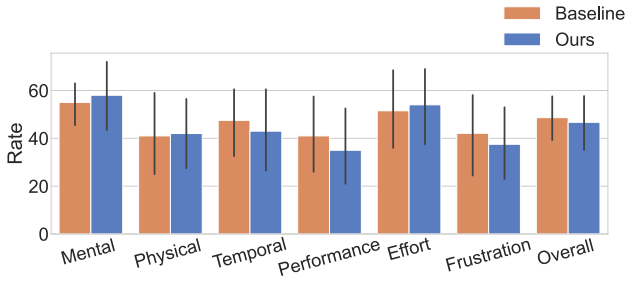


FIGURE 8. NASA-TLX results. Each bar shows the average rating of all participants, and the error bars show the standard deviations.

row, the recommended images from our system contain more types of items than the baseline, such as masks, white staplers, large staplers, and tape tools.

C. EVALUATION SCORES

Figure 7 shows the difference in subjective user evaluation of the two interfaces. The bar plot visualizes the 5-point Likert scale preference scores for each design of participants. The mean score of all participants is represented by each bar in the figure, while the error bars show their standard deviations. Our system scored higher on most of the questions, although the difference was not significant. **Q4** resulted in significantly different ratings between the two interfaces ($p = 0.03$, Wilcoxon signed-rank test), and our proposed interface is capable of reducing the frequency of useless data in the recommendations. There also exists a significant difference in preference scores between the two interfaces ($p = 0.04$, Wilcoxon signed-rank test), and our proposed interface received a higher overall preference score.

Furthermore, Fig. 8 shows the results of the NASA-TLX-based questionnaire that illustrates the summary of workload assessment. Similarly to Fig. 7, each bar in the figure shows the mean score of all participants, and the error bars show their standard deviations. Although our method occasionally shows a slightly higher workload than the baseline system, no significant differences were found in any of the dimensions.

D. SUBJECTIVE FEEDBACK

In **Q7**, most of the participants (8 of 10) stated that the option of noise increases their efficiency and the interaction becomes more intuitive and easier to comprehend. One of them commented that *"The operation of the (baseline) is more straightforward than (the proposed method). But I think (the proposed method) is more efficient"*. And another participant commented that *"(The proposed method) is great, while on (baseline) interface too much incorrect and noise were recommend."* Two participants expressed their mixed opinion that our proposed method has better functional integrity, while the baseline has better simplicity. One of them said: *"From a practitioner's view, (the proposed method) is better because I can actively control which ones should be background class, but it sometimes takes time to decide which image should be regarded as noise or not. I prefer the simplicity of (baseline), and it might be easy for novice users."*. Another participant commented that *"I would prefer the binary feedback on the operation side, it's easier. The noise annotation (background class) poses a high mental demand"*.

In general, participants were more receptive to edge IML devices. Most of the participants (9 out of 10) have observed an improvement in model performance in **Q8** and expressed positive impressions about tangible IML devices in **Q9**. Seven participants simply agree that the tangible device is more convenient. One of them left the comment with a mixed opinion that such an IML system working on edge devices cannot directly replace desktop IML as *"I think if we seriously need a good model I may have no hesitation towards using GUI-based IML system, but the experience of immediately receiving the feedback was fairly good, and I enjoyed it in general."* Another participant also did not like the edge IML device and said *"GUI is easier to understand how to use it"*

Furthermore, some participants expressed their dissatisfaction with the experiment and pointed out some improvements. Five participants said that the operation was uncomfortable with the functional button and that they could not find how to cancel the misoperation. One of them said: *"The part I found most inconvenient was remembering the commands for each option. In addition, I was under pressure not to make a human error while the annotation was done, as there was no back or undo button."*. Another common annoyance mentioned by the three participants is mapping the numerical label to the real object category. One participant said that *"Mapping the index numbers to an actual label"* was the biggest inconvenience. The supervised noise filtering method takes some time to update the filter, which makes two participants unsatisfied and commented: *"response time is slow"*, *"latency is not good"*, and *"The second interface was better in my opinion as it took into consideration the samples which did not belong to any of the 3 classes. But the first interface required lesser time to think."*

V. PERFORMANCE EVALUATION

In terms of category definition, the user study mentioned above contains some subjectivity. It was impossible to establish ground truth labels a priori because it was up to the participants to decide which objects each category should include. As a result, we conducted a second experiment to evaluate the performance of the back-end algorithm to supplement the quantitative evaluation of the first.

In the following experiments, we evaluate recommendation algorithms assuming an *oracle* user who always provides correct feedback based on ground-truth annotations. In other words, we label the images in the sample pool with ground-truth category labels and compare how the feedback algorithms extract information from them. To accomplish this, we create a fully re-annotated dataset made up of images gathered during the user study.

To reflect real-world use cases, we simulate two usage patterns for each part of the evaluation. The first is *fixed category definition*, where all target categories are defined in the beginning as in the user study. The second is *incremental category definition*, where the user adds another class after completing the first feedback phase. We compare the performance under different noise ratios in each use case to evaluate their performance under different conditions. In the following experiments, the system is run with the threshold parameters $M = 3$, $N = 3$, and $L = 3$ and the model's performance is recorded every 20 (oracle) user feedback.

Our proposed algorithm is compared with three baseline algorithms, which are listed below.

- **Binary (Random Sampling)** This baseline uses binary feedback from the oracle user and chooses the feedback candidate completely randomly.
- **Binary (Uncertainty Sampling)** This baseline also employs binary feedback, selecting candidates' feedback based on the uncertainty sampling strategy. It selects data solely based on the ranking of prediction uncertainty u.
- **Binary (Auto-filter)** This baseline uses the result of an unsupervised noisy sample filter for sample recommendation. Specifically, features in the sample pool are projected onto a 2D distribution using the t-SNE algorithm [72], and noisy samples are identified by fitting the Gaussian Mixture Model [73].

A. RESULTS

We created a fully annotated dataset for quantitative evaluation using the data collected from all participants in the user study. We manually annotated the recorded images following these three basic categories, rather than relying on user-defined training samples obtained through the study. Specifically, we used 2,574 images collected during the user study. If a single dominant object in the center of the image is clearly captured without motion blur, we label it as one of the target categories. To avoid overlap, we manually split the object images into a sample pool and test data in this experiment after annotating them. As a result, the sample pool contains 405, 275, 223 valid images for *Electronic devices*, *Stationery*, and *Daily necessities*, respectively. For these three categories, the test set contains 77, 87, and 81 valid images, respectively. The remaining 1,426 images in the sample pool were classified as noise images. We also controlled the ratio of noisy images in the sample pool using these data by randomly removing some images. Table 1 summarizes the noise ratio in each setting. Each column shows the target noise ratio, the total size of the sample pool, the number of noises, and the number of valid samples in the sample pool, in order from left to right.

TABLE 1. Summary of the sample pools with different noise ratios. From left to right, each column indicates the target noise ratio, the total size of the sample pool, the number of noises, and the number of valid samples in it.

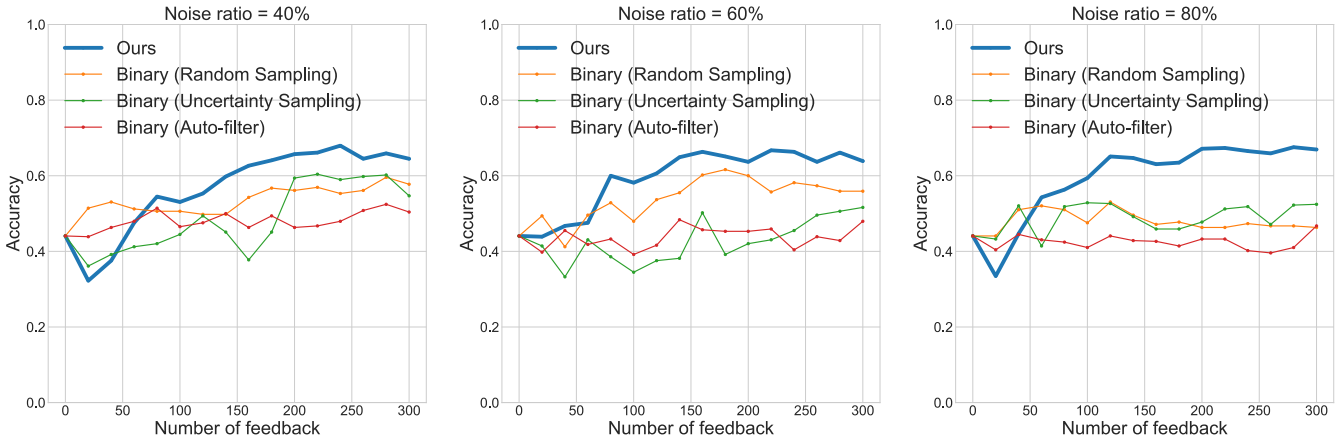
Noise ratio	Pool size	Num. noise	Num. valid image
40%	1505	602	903
60%	2257	1355	903
80%	1783	1426	357

1) FIXED CATEGORY DEFINITION

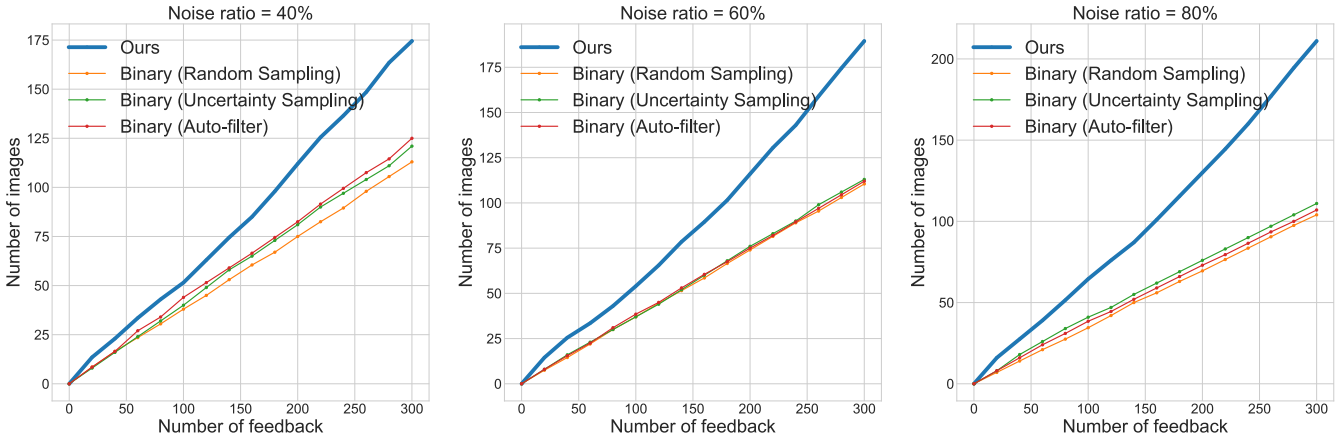
In this section, we evaluate the performance in a simple setting with a fixed category definition. We assume that, before the feedback phase, the user had already defined all object categories with some training images and had gathered a sufficient amount of candidate images in the sample pool. We randomly selected three images from each of the three object categories as initial training data, and these nine images were used to train a shared initial model under all conditions. The accuracy of this initial model on the test set was 0.44. Using this as a starting point, we assess how performance changes with (oracle) user feedback.

Fig. 9a depicts the performance of each algorithm in different noise ratios. The horizontal axis of each graph represents the number of oracle user feedbacks, while the vertical axis represents the accuracy of the classification in the test data. The number of images recommended by each algorithm is shown in Fig. 9b. The number of images encountered by users during the same 400 feedbacks is shown on the vertical axis in each plot.

Our proposed algorithm increased the accuracy to 0.65 after 300 feedbacks, when the noise ratio is 40%. Similarly, our proposed algorithm increased the accuracy to 0.64 when the noise ratio is 60%. In contrast, none of the baseline methods performs as well as the proposed method, with an accuracy below 60%. In more difficult situations with higher noise ratios, this tendency is more pronounced. With the 80% noise ratio, the baseline algorithms almost failed to improve the model performance, while the proposed algorithm still improved the accuracy to 0.67. As illustrated in Fig. 9b, one of the reasons is that our algorithm results in more training data with respect to feedback.



(a) Change in classification performance with respect to the number of feedbacks given by the user.



(b) Number of images recommended by the system relative to the number of feedbacks given by the user.

FIGURE 9. **The comparison of different systems in terms of classification performance and image handling efficiency under various noise ratios. Each plot shows the system performance evolution using the sample pool with a certain noise ratio, and horizontal axes show the number of feedbacks. The vertical axis corresponds to (a) the accuracy and (b) the number of images the system recommended.**

2) INCREMENTAL CATEGORY DEFINITION

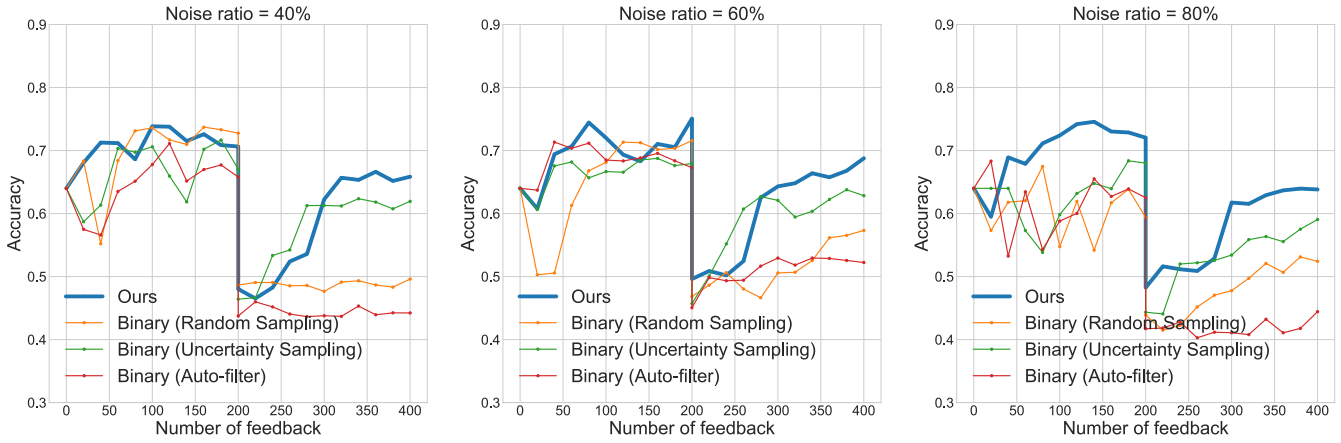
In this section, we assess the performance in a more complicated task with incremental category definition. Here, we assume that the user only defined two object categories at the beginning, and then we used half of the images in the sample pool to provide feedback. After the first feedback phase, it is assumed that the user defines another object category and gives another feedback using the rest of the images in the sample pool. The category and its corresponding samples are the same as in the fixed category definition experiment. In this case, we evaluate the class average accuracy using only the classes that are currently available, ignoring the categories that have not yet been defined. At the beginning, the test accuracy for the two-category classification was 0.64.

Fig. 10a illustrates the performance of the algorithms in different noise ratios, which is similar to the previous experiment. After the 200th feedback, a new category is defined, and we can see a clear performance drop. Fig. 10b also shows the number of images recommended by each algorithm in different noise ratios. When the noise ratio is 40%, our proposed algorithm increased the accuracy to 0.71 after the first 200 feedbacks and finally reaches 0.66 after the total 400 feedbacks. Similarly, when the noise ratio is 60%, our proposed algorithm increased the accuracy to 0.75 after the first 200 feedbacks and finally reached 0.69 after 400 feedbacks. In more difficult situations with a noise ratio of 80%, the proposed algorithm still used 200 feedbacks to improve the accuracy to 0.72, and then achieved 0.64 after another 200 feedbacks. As can be seen, none of the other baseline methods performs as well as the proposed method, with the exception of random sampling, which slightly outperforms our algorithm in the first half of the case of 40% noise ratio. Similarly to the previous experiment, Fig. 10b indicates that our algorithm results in more training data.

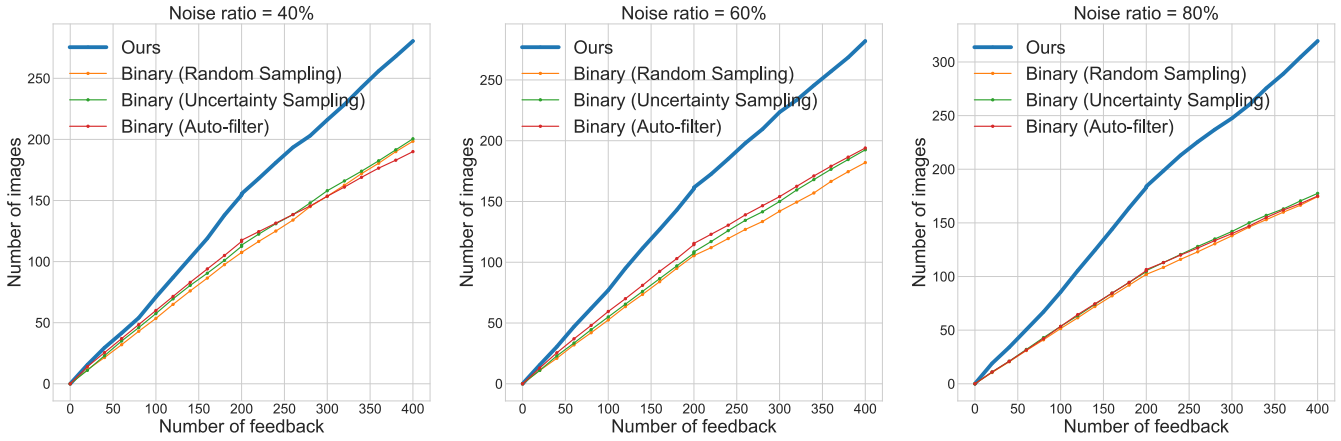
VI. DISCUSSIONS

A. ADVANTAGES OF THE PROPOSED APPROACH

We discovered that the recommendation module can benefit both the user and the system during our tests. Edge IML devices can learn more efficiently from users thanks to the feedback scheme and improve recognition accuracy. Users



(a) Transition of the class average accuracy along with the increasing number of feedbacks with sample pools of different noise ratios.



(b) The number of processed images along with the number of feedback the algorithm gave.

FIGURE 10. The comparison between different systems in terms of both average class accuracy and image handling efficiency under different noise ratios. The number of feedback is represented by the horizontal axis in each graph. The vertical axis represents the recognition accuracy when comparing performance in (a). In the comparison of the number of images the user meets in (b), the vertical axis represents the number of images.

can also actively intervene and take complete control of the training process by providing feedback.

The user study also revealed that the ternary feedback system has no discernible impact on user experience. Participants were not confused by the additional noise option during the feedback phase. Instead, it provides users with a simple way to provide more information about the system. The ternary feedback design has its own advantages in terms of intuitiveness and efficiency compared to binary feedback.

Furthermore, we found that using ternary feedback in conjunction with our proposed algorithm improves performance. Our proposed system consistently outperformed the baseline algorithms without user-defined noise in quantitative testing. In our IML scenario, existing active learning strategies do not work well in the noisy sample pool, and unsupervised noise filtering is also challenging. Human feedback can make the recommendation module more efficient and accurate than the baseline systems, according to our findings. In addition, the baseline system without noise filtering tended to perform poorly as the noise rate in the sample pool increased. It shows that binary user feedback on the target class alone is not sufficient for sample recommendation, and noise filtering is an important part of the feedback loop design.

B. LIMITATIONS AND FUTURE WORK

We discovered that our system still has many limitations after analyzing the behavior of the participants and their subjective feedback in the user study, as well as the data from each part of the experiment. These drawbacks are primarily related to operational efficiency, ease of use, and versatility of the system.

We found that there is still room for improvement in interaction design. Our prototype relies heavily on the joystick for interaction and lacks undo and modification features. This design imposes a relatively higher mental load on the user, as indicated by subjective feedback. Although the limited interaction modality was the driving force behind this research, users should benefit from more discriminative buttons. In particular, usability could be greatly improved by designing custom hardware. Buttons with more intuitive shapes and colors, such as green check, red cross, and gray

question, could be used to represent ternary feedback options in a more intuitive way.

The user experience was also influenced by the latency of the model update during the input phase. To improve the efficiency of the system, the current training strategy should also be changed. When the target category does not change, one possible method is to perform incremental learning and only retrain the model from scratch when a new category is defined, which can further save computational resources.

Due to the processing capabilities of edge devices, when the prediction is not stable, users tend to stay in front of the object longer to achieve a stable result. As a result of this common behavior, the sample pool contains an excessive number of similar images. As a result, the sample pool contains a lot of redundant information, which is another reason for inefficiency. Integration of smarter ways to reduce redundant images in the sample pool is also important for future work. One possible way is to implement simple or fast algorithms to compute image similarities. An efficient sample pool construction would be possible if similar images could be removed in real-time on the device.

The advantage of our system over active learning is not as obvious in the incremental category definition experiment as in the fixed category definition. In this case, the filter may have saved some outdated information during the first feedback phase. To improve the filter, it should be able to forget some old information, such as removing images from its training set that are relatively similar to newly defined category images.

Besides, one participant wondered how the system would interpret the current frame if there were multiple objects in the same frame. Since the current system is specifically designed for the basic image recognition task, this question made us realize that more generic feedback loops designed for various AI tasks are also worth considering. More specific feedback and annotation interfaces are required for other tasks. One of our future research directions is to apply the concepts of sample recommendation and noise filtering to other visual recognition tasks on edge devices equipped with interactive machine learning systems. For example, the recommendation module should be able to suggest and ask user feedback on bounding boxes in order to apply our approach to object detection and tracking tasks.

VII. CONCLUSION

In this paper, we present a novel system design of an edge IML device. We propose introducing user feedback based on ternary feedback to achieve efficient sample annotation with the limited interaction modality on small devices. We added an option for users to specify noisy samples unrelated to the target task to use an active learning-like sample recommendation strategy with the noisy sample pool available during interactive system usage. Despite allowing users complete control over the target task definition, the system updates both the classifier and the noise filtering modules. We built a prototype device and conducted a qualitative user study as well as a quantitative performance assessment. Throughout the experiments, we found that our feedback design reduces the annotation burden without affecting the user experience while improving the classification performance over the baseline algorithms.

REFERENCES

[1] S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza, "Power to the people: The role of humans in interactive machine learning," *AI Mag.*, vol. 35, no. 4, pp. 105–120, 2014.

[2] B. Jiang and J. Canny, "Interactive machine learning via a GPU-accelerated toolkit," in *Proc. 22nd Int. Conf. Intell. User Interfaces (IUI)*, 2017, pp. 535–546.

[3] D. Arendt, E. Saldanha, R. Wesslen, S. Volkova, and W. Dou, "Towards rapid interactive machine learning: Evaluating tradeoffs of classification without representation," in *Proc. 24th Int. Conf. Intell. User Interfaces (IUI)*, Mar. 2019, pp. 591–602.

[4] S. Mishra and J. M. Rzeszotarski, "Designing interactive transfer learning tools for ML non-experts," in *Proc. CHI Conf. Hum. Factors Comput. Syst. (CHI)*, 2021, pp. 1–15.

[5] T. Spinner, U. Schlegel, H. Schäfer, and M. El-Assady, "Explainer: A visual analytics framework for interactive and explainable machine learning," *IEEE Trans. Vis. Comput. Graphics*, vol. 26, no. 1, pp. 1064–1074, Jan. 2019.

[6] S. Teso and K. Kersting, "Explanatory interactive machine learning," in *Proc. AAAI/ACM Conf. AI, Ethics, Soc.*, Jan. 2019, pp. 239–245.

[7] J. Cui, F. Wen, R. Xiao, Y. Tian, and X. Tang, "EasyAlbum: An interactive photo annotation system based on face clustering and re-ranking," in *Proc. CHI Conf. Hum. Factors Comput. Syst. (CHI)*, 2007, pp. 367–376.

[8] M. Pirrung, N. Hilliard, N. O'Brien, A. Yankov, C. D. Corley, and N. O. Hodas, "SHARKZOR: Human in the loop ML for user-defined image classification," in *Proc. 23rd Int. Conf. Intell. User Interfaces Companion*, Mar. 2018, pp. 1–2.

[9] J. A. Fails and D. R. Olsen, "Interactive machine learning," in *Proc. the 8th Int. Conf. Intell. User Interfaces (IUI)*, 2003, pp. 39–45.

[10] M. Kabra, A. A. Robie, M. Rivera-Alba, S. Branson, and K. Branson, "JAABA: Interactive machine learning for automatic annotation of animal behavior," *Nature Methods*, vol. 10, pp. 64–67, Dec. 2013.

[11] J. Blaas, C. P. Botha, and F. H. Post, "Interactive visualization of multi-field medical data using linked physical and feature-space views," in *Proc. 9th Joint Eurographics IEEE VGTC Symp. Vis.*, 2007, pp. 123–130.

[12] T. Ishibashi, Y. Nakao, and Y. Sugano, "Investigating audio data visualization for interactive sound recognition," in *Proc. 25th Int. Conf. Intell. User Interfaces*, Mar. 2020, pp. 67–77.

[13] H. Doleisch, M. Gasser, and H. Hauser, "Interactive feature specification for focus+ context visualization of complex simulation data," in *Proc. IEEE Trans. Vis. Comput. Graphics Symp. Vis.*, vol. 3, Oct. 2003, pp. 239–248.

[14] B. Settles, "Active learning literature survey," Comput. Sci. Dept., Univ. Wisconsin-Madison, Madison, Madison, WI, USA, Tech. Rep. 1648, 2009.

[15] J. J. Dudley and P. O. Kristensson, "A review of user interface design for interactive machine learning," *ACM Trans. Interact. Intell. Syst.*, vol. 8, no. 2, pp. 1–37, Jun. 2018.

[16] P. Musialski, M. Cui, J. Ye, A. Razdan, and P. Wonka, "A framework for interactive image color editing," *Vis. Comput.*, vol. 29, no. 11, pp. 1173–1186, Nov. 2013.

[17] Y. Cheng, Z. Gan, Y. Li, J. Liu, and J. Gao, "Sequential attention GAN for interactive image editing," in *Proc. 28th ACM Int. Conf. Multimedia*, 2020, pp. 4383–4391.

[18] X. Guo, H. Wu, Y. Cheng, S. Rennie, G. Tesauro, and R. S. Feris, "Dialog-based interactive image retrieval," 2018, *arXiv:1805.00145*.

[19] J. O'Donovan, B. Smyth, B. Gretarsson, S. Bostandjiev, and T. Höllerer, "PeerChooser: Visual interactive recommendation," in *Proc. CHI Conf. Hum. Factors Comput. Syst. (CHI)*, 2008, pp. 1085–1088.

[20] H. Wang, Q. Wu, and H. Wang, "Factorization bandits for interactive recommendation," in *Proc. AAAI Conf. Artif. Intell.*, vol. 31, no. 1, 2017, pp. 2695–2702.

[21] M. Carney, B. Webster, I. Alvarado, K. Phillips, N. Howell, J. Griffith, J. Jongejan, A. Pitaru, and A. Chen, "Teachable machine: Approachable web-based tool for exploring machine learning classification," in *Proc. CHI Conf. Hum. Factors Comput. Syst. (CHI)*, 2020, pp. 1–8.

[22] A. Agassi, H. Erel, I. Y. Wald, and O. Zuckerman, "Scratch nodes ML: A playful system for children to create gesture recognition classifiers," in *Proc. Extended Abstr. CHI Conf. Hum. Factors Comput. Syst.*, May 2019, pp. 1–6.

[23] S. Zhang, J. H. Liew, Y. Wei, S. Wei, and Y. Zhao, "Interactive object segmentation with inside-outside guidance," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 12234–12244.

[24] E. Zheng, Q. Yu, R. Li, P. Shi, and A. Haake, "A continual learning framework for uncertainty-aware interactive image segmentation," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, vol. 35, no. 7, 2021, pp. 6030–6038.

[25] Y. Zeng, Z. Lin, and V. M. Patel, "SketchEdit: Mask-free local image manipulation with partial sketches," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 5951–5961.

[26] Q. Liu, M. Zheng, B. Planche, S. Karanam, T. Chen, M. Niethammer, and Z. Wu, "PseudoClick: Interactive image segmentation with click imitation," 2022, *arXiv:2207.05282*.

[27] H. Kacorri, K. M. Kitani, J. P. Bigham, and C. Asakawa, "People with visual impairment training personal object recognizers: Feasibility and challenges," in *Proc. CHI Conf. Hum. Factors Comput. Syst. (CHI)*, 2017, pp. 5839–5849.

[28] D. Ahmetovic, D. Sato, U. Oh, T. Ishihara, K. Kitani, and C. Asakawa, "ReCog: Supporting blind people in recognizing personal objects," in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, Apr. 2020, pp. 1–12.

[29] A. J. Joshi, F. Porikli, and N. Papanikolopoulos, "Breaking the interactive bottleneck in multi-class classification with active selection and binary feedback," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 2995–3002.

[30] H. Ngo, M. Luciw, J. Nagi, A. Forster, J. Schmidhuber, and N. A. Vien, "Efficient interactive multiclass learning from binary feedback," *ACM Trans. Interact. Intell. Syst.*, vol. 4, no. 3, pp. 1–25, Nov. 2014.

[31] J. Nagi, H. Ngo, A. Giusti, L. M. Gambardella, J. Schmidhuber, and G. A. Di Caro, "Incremental learning using partial feedback for gesture-based human-swarm interaction," in *Proc. 21st IEEE Int. Symp. Robot Hum. Interact. Commun. (IEEE RO-MAN)*, Sep. 2012, pp. 898–905.

[32] D. D. Lewis and W. A. Gale, "A sequential algorithm for training text classifiers," in *Proc. 17th Annu. Int. ACM-SIGIR Conf. Res. Dev. Info. Retriv.*, 1994, pp. 3–12.

[33] H. S. Seung, M. Opper, and H. Sompolinsky, "Query by committee," in *Proc. 5th Annu. Workshop Comput. Learn. Theory (COLT)*, 1992, pp. 287–294.

[34] B. Settles, M. Craven, and S. Ray, "Multiple-instance active learning," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 20, 2007, pp. 1289–1296.

[35] N. Roy and A. McCallum, "Toward optimal active learning through Monte Carlo estimation of error reduction," in *Proc. ICML*, vol. 2, 2001, pp. 441–448.

[36] K. Dimitriadou, O. Papaemmanouil, and Y. Diao, "AIDE: An active learning-based approach for interactive data exploration," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 11, pp. 2842–2856, Nov. 2016.

[37] C. L. Isbell, M. Kearns, S. Singh, C. R. Shelton, P. Stone, and D. Kormann, "Cobot in LambdaMOO: An adaptive social statistics agent," *Auto. Agents Multi-Agent Syst.*, vol. 13, no. 3, pp. 327–354, Nov. 2006.

[38] A. L. Thomaz and C. Breazeal, "Teachable robots: Understanding human teaching behavior to build more effective robot learners," *Artif. Intell.*, vol. 172, nos. 6–7, pp. 716–737, Apr. 2008.

[39] W. B. Knox and P. Stone, "Reinforcement learning from human reward: Discounting in episodic tasks," in *Proc. IEEE RO-MAN: 21st IEEE Int. Symp. Robot Human Interact. Commun.*, Sep. 2012, pp. 878–885.

[40] Z. Liu, J. Wang, S. Gong, H. Lu, and D. Tao, "Deep reinforcement active learning for human-in-the-loop person re-identification," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6122–6131.

[41] L. Zhao, G. Sukthankar, and R. Sukthankar, "Incremental relabeling for active learning with noisy crowdsourced annotations," in *Proc. IEEE 3rd Int. Conf. Privacy, Secur., Risk Trust IEEE 3rd Int. Conf. Social Comput.*, Oct. 2011, pp. 728–733.

[42] M. Buhrmester, T. Kwang, and S. D. Gosling, "Amazon's mechanical turk: A new source of inexpensive, yet high-quality data?" *Perspect. Psychol. Sci.*, vol. 6, no. 1, pp. 3–5, 2011.

[43] B. Rahmanian and J. G. Davis, "User interface design for crowdsourcing systems," in *Proc. Int. Work. Conf. Adv. Vis. Intf.*, 2014, pp. 405–408.

[44] Y.-L. Lee, P.-K. Tsung, and M. Wu, "Techology trend of edge AI," in *Proc. Int. Symp. VLSI Design, Autom. Test (VLSI-DAT)*, Apr. 2018, pp. 1–2.

[45] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge AI: On-demand accelerating deep neural network inference via edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 447–457, Jan. 2019.

[46] M. G. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, "Machine learning at the network edge: A survey," *ACM Comput. Surv.*, vol. 54, no. 8, pp. 1–37, Nov. 2021.

[47] S. Lee, B. Islam, Y. Luo, and S. Nirjon, "Intermittent learning: On-device machine learning on intermittently powered system," *Proc. ACM Interact., Mobile, Wearable Ubiquitous Technol.*, vol. 3, no. 4, pp. 1–30, Dec. 2019.

[48] F. Ott, M. Wehbi, T. Hamann, J. Barth, B. Eskofier, and C. Mutschler, "The OnHW dataset: Online handwriting recognition from IMU-enhanced ballpoint pens with machine learning," *Proc. ACM Interact., Mobile, Wearable Ubiquitous Technol.*, vol. 4, no. 3, pp. 1–20, Sep. 2020.

[49] T. Zhan, W. Li, X. Chen, and S. Lu, "MegaLight: Learning-based color adaptation for barcode stream recognition over screen-camera links," *Proc. ACM Interact., Mobile, Wearable Ubiquitous Technol.*, vol. 3, no. 2, pp. 1–23, Jun. 2019.

[50] B. Islam and S. Nirjon, "Zygarde: Time-sensitive on-device deep inference and adaptation on intermittently-powered systems," *Proc. ACM Interact., Mobile, Wearable Ubiquitous Technol.*, vol. 4, no. 3, pp. 1–29, Sep. 2020.

[51] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, and R. Boyle, "In-datacenter performance analysis of a tensor processing unit," in *Proc. 44th Annu. Int. Symp. Comput. Architect.*, 2017, pp. 1–12.

[52] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 6105–6114.

[53] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[54] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 6848–6856.

[55] C. Gupta, A. S. Suggala, A. Goyal, H. V. Simhadri, B. Paranjape, A. Kumar, S. Goyal, R. Udupa, M. Varma, and P. Jain, "ProtoNN: Compressed and accurate kNN for resource-scarce devices," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 1331–1340.

[56] K. Pradeep, K. Kamalavasan, R. Natheesan, and A. Pasqual, "EdgeNet: SqueezeNet like convolution neural network on embedded FPGA," in *Proc. 25th IEEE Int. Conf. Electron., Circuits Syst. (ICECS)*, Dec. 2018, pp. 81–84.

[57] A. Kumar, S. Goyal, and M. Varma, "Resource-efficient machine learning in 2 KB RAM for the Internet of Things," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 1935–1944.

[58] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50X fewer parameters and <0.5 MB model size," 2016, *arXiv:1602.07360*.

[59] G. Ananthanarayanan, P. Bahl, P. Bodík, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha, "Real-time video analytics: The killer app for Edge computing," *Computer*, vol. 50, no. 10, pp. 58–67, 2017.

[60] J. Wang, Z. Feng, Z. Chen, S. George, M. Bala, P. Pillai, S.-W. Yang, and M. Satyanarayanan, "Bandwidth-efficient live video analytics for drones via edge computing," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2018, pp. 159–173.

[61] G. Kar, S. Jain, M. Gruteser, F. Bai, and R. Govindan, "Real-time traffic estimation at vehicular edge nodes," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, Oct. 2017, pp. 1–13.

[62] S. Wan, S. Ding, and C. Chen, "Edge computing enabled video segmentation for real-time traffic monitoring in Internet of Vehicles," *Pattern Recognit.*, vol. 121, Jan. 2022, Art. no. 108146.

[63] Z. Xu, H. Gupta, and U. Ramachandran, "STTR: A system for tracking all vehicles all the time at the edge of the network," in *Proc. 12th ACM Int. Conf. Distrib. Event-Based Syst.*, Jun. 2018, pp. 124–135.
[64] M. Hahnle, F. Saxen, M. Hisung, U. Brunsmann, and K. Doll, "FPGA-based real-time pedestrian detection on high-resolution images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2013, pp. 629–635.
[65] C. Liu, Y. Cao, Y. Luo, G. Chen, V. Vokkarane, M. Yunsheng, S. Chen, and P. Hou, "A new deep learning-based food recognition system for dietary assessment on an edge computing service infrastructure," *IEEE Trans. Services Comput.*, vol. 11, no. 2, pp. 249–261, Apr. 2017.
[66] Y. Weng and C. Xia, "A new deep learning-based handwritten character recognition system on mobile computing devices," *Mobile Netw. Appl.*, vol. 25, no. 2, pp. 402–411, Apr. 2020.
[67] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 25, 2012, pp. 1097–1105.
[68] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Jul. 1995.
[69] S. R. Kulkarni, S. K. Mitter, and J. N. Tsitsiklis, "Active learning using arbitrary binary valued queries," *Mach. Learn.*, vol. 11, no. 1, pp. 23–35, 1993.
[70] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, and J. Vanderplas, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12 no. 10, pp. 2825–2830, 2011.
[71] S. G. Hart and L. E. Staveland, "Development of NASA-TLX (task load index): Results of empirical and theoretical research," *Adv. Psychol.*, vol. 52, pp. 139–183, Jan. 1988.
[72] D. Ulyanov. (2016). *Multicore-tSNE*. GitHub Repository. [Online]. Available: https://github.com/DmitryUlyanov/Multicore-TSNE
[73] C. Rasmussen, "The infinite Gaussian mixture model," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 12, 1999, pp. 554–560.

TIANYI LIU received the dual B.E. degree from the University of Electronic Science and Technology of China and the University of Glasgow, in 2019, and the M.S. degree from Nanyang Technological University, in 2020. During the project of this article, he was a Graduate Research Student at The University of Tokyo. His research interests include computer vision and human–computer interaction.

YUSUKE SUGANO received the Ph.D. degree in information science and technology from The University of Tokyo, in 2010. He was previously an Associate Professor at the Graduate School of Information Science and Technology, Osaka University, a Postdoctoral Researcher at the Max Planck Institute for Informatics, and a Project Research Associate at the Institute of Industrial Science, The University of Tokyo, where he is currently an Associate Professor. His research interests include computer vision and human–computer interaction.