Computer Networks 191 (2021) 108019

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Task offloading in edge computing for machine learning-based smart healthcare

Mohammad Aazam [a,1,*], Sherali Zeadally [b,1], Eduardo Feo Flushing [c,1]

[a] University of Nottingham, Malaysia
[b] University of Kentucky, USA
[c] Carnegie Mellon University, Qatar

ARTICLE INFO

Keywords:
Cloud computing
Edge computing
Fog computing
Healthcare
Internet of Things (IoT)
Middleware
Machine learning
Offloading

ABSTRACT

Recent advances in networking and mobile technologies such as 5G, long-term evolution (LTE), LiFi, wireless broadband (WiBro), WiFi-Direct, Bluetooth Low Energy (BLE) have paved the way for intelligent and smart services. With an average of more than 6.5 devices per person, a plethora of applications are being developed especially related to healthcare. Although, current edge devices such as smartphone and smartwatch are becoming increasingly more powerful and more affordable, there are certain tasks such as those involving machine learning that require higher computational resources, thereby resulting in higher energy consumption in the case of edge devices. Offloading tasks to co-located edge nodes such as fog (a cloud-like localized, smaller resource pool), or a femto-cloud (integration of multiple edge nodes) is one viable solution to address the issues such as performing compute-intensive tasks, and managing energy consumption. The outbreak of coronavirus disease 2019 (COVID-19) and becoming a pandemic has also made a case for edge computing (involving smartphone, wearables, health sensors) for the detection of symptoms to quarantine potential carriers of the virus. We focus on how various forms of smart and opportunistic healthcare (oHealth) can be provided by leveraging edge computing that makes use of a machine learning-based approach. We apply k-nearest neighbors (kNN), naive Bayes (NB), and support vector classification (SVC) algorithms on real data trace for the healthcare and safety-related scenarios we considered. The empirical results obtained provide useful insights into machine learning-based task offloading in edge computing.

1. Introduction

Technology in various forms has become a de-facto part of our lives, especially as it gets more and more pervasive and intelligent which opens up to several challenges. Edge devices such as smartphone, smartwatch, smartglasses, health sensors with multi-functional capabilities are emerging in the market [1,2]. Coupled with communication technologies such as 5G, 4G, long-term evolution (LTE); or paradigms such as Internet of Things (IoT), cloud computing, and semantic web, that enable innovative applications, several IoT and intelligent applications are making their way into our lives in diverse application domains such as smart homes, smart cities, agriculture, food and healthcare, factories, green energy, safety and disaster management, emergency management, aerospace, intelligent transportation, and telecommunication [3–5]. Depending on the application domain, these applications face several challenges including scalability, privacy, security, usability, energy efficiency, computation management, and so on.

Although edge devices such as smartphones are getting more powerful in terms of computational capabilities, it still becomes necessary to offload tasks to other devices, especially for tasks that are compute intensive and consume too much energy (such as determining permutations of symptoms, e.g., in case of coronavirus disease 2019 (COVID-19)) [6]. As IoT applications become more pervasive, more intelligent services are emerging on the market. This intelligence can be achieved through learning, which heavily depends on the amount of data that the application receives [7,8]. Overloading the edge devices with too much data communication drains their energy [9]. Moreover, several tasks require higher computational power than what most current smartphones or edge devices can support. As a result, it becomes unavoidable to offload certain tasks from relatively resource-constrained edge nodes to other devices such as cloud, fog, or an aggregation of edge devices also referred to as the femto-cloud [6].

Cloud is usually the default choice to offload tasks to but not without its own compromises. First, cloud is more expensive because

* Corresponding author.
E-mail addresses: aazam@ieee.org (M. Aazam), szeadally@uky.edu (S. Zeadally), efeoflus@andrew.cmu.edu (E.F. Flushing).
[1] The authors contributed according to the sequence of their appearance on the author list.

https://doi.org/10.1016/j.comnet.2021.108019
Received 25 December 2020; Received in revised form 18 February 2021; Accepted 12 March 2021
Available online 23 March 2021

the cloud service provider has to maintain a large datacenter in order to be able to reliably provide computing and storage resources. Reaching out to the cloud also means a compromise on the privacy because it is available over the public core network [10]. Moreover, not all of the data needs to be communicated to the cloud through the public network, which will consume a large amount of network bandwidth. By performing several tasks locally, we can avoid unnecessary communications through the core network.

Offloading to local nodes such as fog or edge brings various tradeoffs as well [11]. Offloading to a fog or edge keeps the communication cost to a minimum because only necessary data is communicated beyond these devices to the cloud and privacy of the transmitted data is protected. However, fog or edge devices also have limited resources compared with the cloud. For tasks (such as extensive machine learning tasks, long term storage, data security) that need more computing or storage resources, the cloud is used.

Eventually, offloading of tasks rely on some intelligent system that can make optimal decisions about which specific tasks to be offloaded to the cloud, or to a local fog or femto-cloud. Tasks that can be executed locally must be handled locally. Tasks that require basic machine learning but are privacy and/or cost-sensitive, should be managed by aggregating the locally available resources, such as household edge devices. In the end, the tasks related to storage and extensive machine learning can be offloaded to the cloud. In this paper, we argue on the importance of such a system by providing case-studies related to healthcare. For instance, a system would be suitable for executing machine learning-based tasks that healthcare applications need to execute in different settings such as household, public transit, and globally. In this work we focus mainly on the middleware entities that reside between the IoT and the cloud. In the context of this work, such (middleware) entities are fog (as micro-datacenter or nano-datacenter), edge, femto-cloud, multi-access edge computing (MEC), and cloudlet. The basic differences among these entities are shown in Fig. 1, discussions are presented in [11].

We summarize the main contributions of this work as follows:

- We present case studies for smart healthcare, safety, and emergency response in indoor and outdoor scenarios. Through these case studies we demonstrate how opportunistic healthcare and safety can be provided.
- We describe the role of machine learning in the three use-cases of smart and opportunistic healthcare presented, and where task offloading is used.
- We evaluate three machine learning algorithms (k-nearest neighbors (kNN), support vector machine (SVM), and naive Bayes (NB)) which are used in the three case studies (smart/opportunistic healthcare, safety, emergence response). These machine learning algorithms perform data classification which is an essential element in our case-studies. Our preliminary performance results demonstrate the impact (on energy consumption) of running machine learning on individual edge node and femto-cloud.

2. Related works

In this section, we review some of the recently proposed offloading techniques in fog computing based on different factors.

2.1. Task offloading in edge computing

Emphasizing the importance of task offloading in edge environment, Wang et al. state in [12] that for the scalable IoT applications, it becomes necessary to migrate tasks from resource-constrained devices to a greater pool of resources. For mobile devices, it becomes difficult to ensure service continuity due to the mobility. Hence, offloading tasks to a more stable entity is a viable choice. The authors referred to MEC as a solution for offloading tasks in a mobile environment.

Focusing on the tasks that require high computational capacity which therefore consume more energy, Zhao et al. [13] discussed computational offloading techniques in mobile devices. Given the increasing number in applications that require high computation, mobile devices need sophisticated mechanisms to decide on which tasks to execute locally and which ones to migrate to the cloud. According to the authors, offloading usually takes place to a fog node or a cloud. Both of these entities have their tradeoffs. For example, cloud is rich in resources but is typically far away from the mobile nodes. In contrast, fog is nearby, but does not have abundant resources as the cloud. Hence, offloading to a cloud or a fog consumes different amount of energy and results in different gains in terms of computation. In this context, the authors proposed an algorithm which minimized the energy consumption when offloading a task. First they compute the energy consumption incurred when offloading the task to the fog compared to the cloud. Then, they evaluated which entity is preferable according to the computational requirements of the tasks. Based on these factors, the task is offloaded to the desired entity.

Hasan et al. [14] focused on providing incentives to mobile nodes for executing others' tasks. The authors argued that mobile and IoT devices are resource-constrained and they often require to offload tasks to other peer devices that have relatively underutilized resources if these peer devices get incentivized. In this way, local resources that are underutilized can be harvested at a much cheaper cost rather going to a cloud. The incentive discussed in the paper is provided through crypto-currencies. Pu et al. [15] also discuss incentive-based task offloading in mobile edge devices but their proposed incentive is reciprocal computation in which each device pays-back the other device that has performed tasks on its behalf by executing tasks for it when required in the future.

Meurisch et al. [16] stated that in a mobile edge environment, offloading tasks to unknown devices may not always be feasible. Hence, the proposed solution is to divide the task into micro-tasks and then probe each participant edge device. Based on the network condition and performance of task completion (which is determined by evaluating the execution of a sample micro-task), and the availability of the resources on the probed device, a decision about where and when to offload micro-tasks is made. Wang et al. [17] also proposed femto-cloud based task offloading, where a group of mobile or edge devices collect their resources to perform a task that is not possible otherwise on a standalone device.

2.2. Offloading for healthcare applications

Given the importance of privacy in healthcare applications Craciunescu et al. [18] argued that it is not always feasible to send critical health data to the cloud. Privacy-sensitive data has to remain local, making task offloading among the peer nodes the most viable choice.

Zhao et al. [4] and Aloi et al. [19] emphasized that smartphones and other smart edge devices such as smartwatch are equipped with several sensors. This enables the development and support of healthcare related applications based on the sensors in these smart devices. A femto-cloud of these smart devices can be created by leveraging the available sensors.

2.3. Role of machine learning in edge computing and healthcare

Machine learning enables systems to automatically learns programs from data, making machines intelligent, and minimizing manual input [7,20]. Machine learning can perform predictive analysis and data mining (an advanced form is called deep learning), which are essential in developing intelligent healthcare applications [21,22]. For the purpose of "learning" or making the system intelligent enough to predict correctly about the changes in the (healthcare or any) system, machine learning-based technique has to first classify the data [19,23]. Classification is performed by a component called *classifier* in the machine

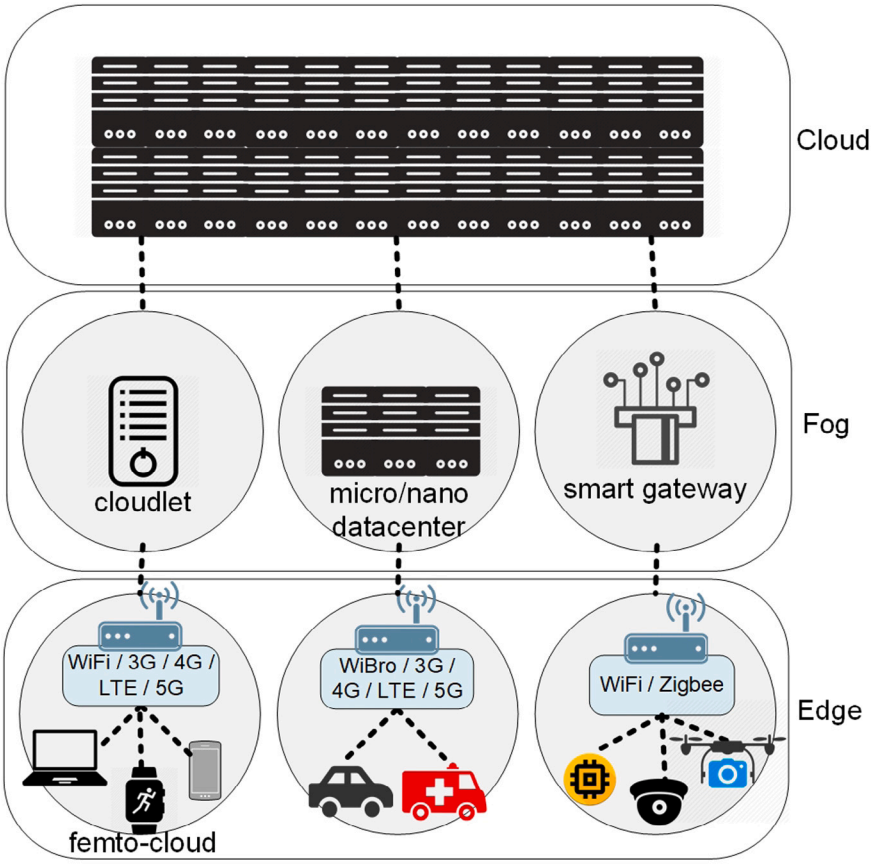

Fig. 1. Layered architecture of edge, fog, and cloud integration.

learning system [7,8]. The classifier takes the feature values as an input and gives the *class* as the output [7].

Fig. 2 shows the basic mechanism of machine learning system. The data coming from the source is first divided into two parts: training data, and test data. Depending on the type of application and size of data, the magnitude of *training data* and *test data* are decided. Machine learning classifier classifies the data according to the requirements of the application (for instance in case of healthcare, it can be activity recognition) and trains itself using the *training data*, and tests its training using *test data*. Based on the correctness of the tested data, the classifier creates the predicted value for the future events and forwards them to the relevant node (e.g. health sensor).

Hussain et al. [7] stated that machine learning will be one of the key drivers of most of the innovations in the future. Machine learning will be playing a very important role in technologies in diverse domains and in particular healthcare areas such as emergency detection, disease prediction, drug design. Data mining and predictive analysis would be the noteworthy elements in machine learning in healthcare area. According to the authors, there are different types of machine learning techniques, but the most widely adopted one is the classifier. In classifier, a vector of continuous values is input to the system, which analyzes them and classifies them by providing a discrete value for the class of the data input.

Chen et al. [20] emphasize on the importance of machine learning-based disease prediction and accurate analysis of medical data for patient care and community service. The authors stated that when the data is incomplete, it becomes hard to accurately predict disease. To counter this issue, the authors proposed latent factor model to reconstruct the missing parts of the data. The authors proposed convolutional neural network (CNN)-based model for disease risk prediction by using structured and unstructured real hospital data from year 2013 to 2015.

O'Shea and Hoydis stated the importance of deep learning in [24]. The authors point out that for computer vision, deep learning is the right choice because mathematical models does not accurately work on real-world images. Deep learning, however, can characterize the images, which is vital in case of healthcare and safety application — which is the main focus of this paper. Byrne in [8] also state that machine learning process requires neural networks through which the machine learning-based application learns the rules from the input data, which is a processing-rich task.

Table 1 summarizes the literature on task offloading.

3. Potential applications of machine learning-based smart healthcare

In this section, we present some future practical application scenarios for smart and opportunistic healthcare (oHealth) [26], and safety and emergency response.

3.1. Collaborative healthcare in the home environment

Here, we describe how we can improve the quality of healthcare services by leveraging the capabilities of smart devices. In particular, we highlight the potential of smart devices for opportunistically supporting health services that are either not possible otherwise, or lack the privacy protection they need, or are not affordable. On average, globally, each household has 5.5 people [27], who can contribute to privacy-aware healthcare services. In the most populous parts of the world, Africa and Asia, there are up to 9 people in a household [27]. Moreover, according to Statista(.com), each person possesses 6.58 devices on average. Combining the resources of these devices can lead to the emergence of several applications within a household that are only possible by using high-end computers at a much higher cost. Moreover,

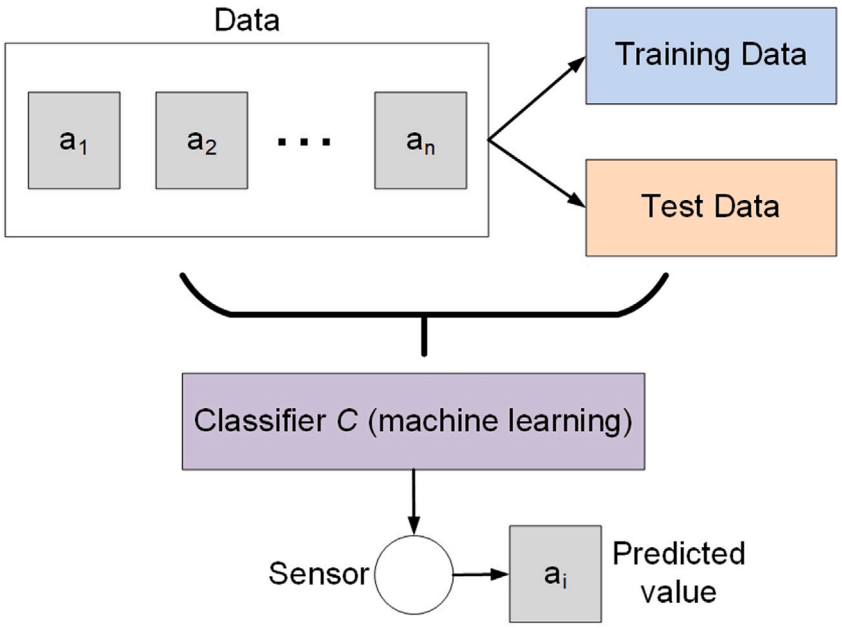

Fig. 2. The basic working mechanism of a machine learning system.

Table 1
Summary of where and why tasks are offloaded.

Task Offloading in Edge-Fog-Cloud			
Focus Area	Author	Offload Vector	Reasons for Offloading
Edge computing	Wang et al. [12]	Mobile device to MEC	Resource-constraint, mobility
Edge computing	Zhao et al. [25]	Mobile device to fog or cloud	Energy consumption
Femto-cloud	Hasan et al. [14]	Mobile device to femto-cloud	Incentives
Femto-cloud	Pu et al. [15]	Mobile device to mobile device	Incentives
Femto-cloud	Meurisch et al. [16]	Mobile device to mobile device	Resource management (micro-tasking)
Femto-cloud	Wang et al. [17]	Mobile device to mobile device	Resource management
Healthcare	Craciunescu et al. [18]	Peer nodes	Privacy
Healthcare	Zhao et al. [4]	From and to smart edge devices	Enhanced services
Healthcare	Aloi et al. [19]	From and to smart edge devices	Enhanced services

for healthcare applications, privacy remains a major concern. If the devices within a household are leveraged to enhance the capabilities (such as evolutionary intelligence) of healthcare applications, it will ensure privacy (because the privacy-sensitive health data is not sent over the public core network) to a great extent as well as customizability (because the health service is tailored according to each user's requirements).

Existing healthcare services rely on the capabilities of the device they are running on, limiting their scope in terms of their intelligence and context. This is mainly because standalone devices cannot efficiently run the tasks that make the system evolutionarily intelligent. Consequently, it becomes a challenge to extend the functionality of the healthcare applications running on resource-constrained mobile devices. For example, a diabetes monitoring application provides notifications based on the readings from the sensors installed on the person which collect information based on the lifestyle, physical activities, and food intake of the person. In such a case, the application cannot detect a certain change in the lifestyle (such as physical activity) when it happens. But by combining the nearby edge resources such as smartphone, smartwatch, body area sensors, in the form of femto-cloud, this healthcare application can provide better healthcare support opportunistically.

To illustrate our femto-cloud based oHealth, we consider the following scenario. Ariana and Ahmed are siblings who are living in the same household. Their parents have diabetes which makes both the siblings susceptible to the same disease genetically. Ahmed has a healthcare and fitness application on his smartphone that provides information about his physical activity, food intake, calories burned. The application does not consider the varying effects according to the genetic history of the family. For that, it is important to consider the healthcare data of the other family members in the household. Ahmed does not use any specific diabetes monitoring sensor, however, Ariana does. Ariana also has a fitness and healthcare application on her phone that collects information about blood glucose, blood pressure, and other vital signs. Ariana's smartphone application taking measurements from her daily routine determines the amount of physical activity that affects the diabetes. The application then shares the finding with Ahmed, who can then adjust his workout routine and food intake based on the data received. In this way, Ahmed is able to control his diabetes indirectly without having to separately purchase a diabetes sensor. Hence, resource utilization is optimized (here it should be noted that both Ahmed and Ariana are not diabetic. They just need to take precautionary measures). On the other hand, Ahmed has a body temperature monitoring sensor (say in his smartwatch) which alerts him if his body temperature has risen above a certain level. Ariana, being a patient of mild diabetes, would want to avoid any such thing that causes a rise in the body temperature or later on catch cold or other infections since diabetes slows down body's ability to fight infection. Hence detection of symptoms such as fever, headache, cold, flu becomes important. Similarly, several other genetic diseases such as Down syndrome, thalassemia, familial hypercholesterolemia, Parkinson's disease, hypertension, and so on can be monitored along with their relevant symptoms, within a household.

For any computationally intensive task a healthcare application has to perform, resources of household edge devices can be combined as

a femto-cloud [6]. One example of a compute-intensive task can be determining the permutations of various symptoms, such as the case of Crohn's disease, a combination of arthritis, anemia, skin issues. Crohn's disease is genetic, and it becomes necessary to monitor symptoms among the siblings, which may require machine learning tasks. Instead of going to the cloud for all the tasks that require more computation capabilities (at a much higher cost) that can be provided by an edge device, leveraging edge resources will help protect the privacy, minimize the cost, and avoid consuming core network bandwidth. Since each owner of the device(s) has equal access to enhanced healthcare service, the incentive is inherently there. In such a case, the device that communicates with all the other devices more often is selected by the healthcare application as the Master Edge (ME) device. The selection of ME device is dynamic and may change over time, as the frequency of communications with the other edge nodes changes. It is worth noting that the ME device itself may not necessarily be a smartphone. It can be any device that is not just meeting with the other devices, but is also capable of performing the aforementioned activities. Therefore, it can be a home computer, a laptop, an healthcare fog node in the house. The ME device is responsible for receiving health logs (Fig. 3a) from each edge device (smartphone, smartwatch, body area network). The healthcare application running on the ME device determines what to share with other devices in the household network. Later, if a device makes a request for a computationally intensive task, the ME may divide the task into subtasks based on the availability of the devices, their resources, connectivity frequency, battery consumption, and their ability to run the task (Fig. 3b).

Each subtask is then outsourced to each relevant edge node which performs the required task and returns the result to the ME device. In case the ME device does not meet the originator of the request in an expected period of time (which can be set by originator), it offloads the result to the cloud which then returns it to the relevant edge device. The cloud is also involved when data storage is required for a longer period of time or for more complex computational tasks such as those involving deep learning.

Fig. 3. (a) shows the health log sharing between household femto-cloud nodes, (b) shows the scenario where a device requires a computationally intensive task that is executed by aggregating femto-cloud resources. Deep learning tasks are performed in the cloud when required.

3.2. Safety and emergency response

In a household environment, technology around personal security and ambient assisted living (AAL) can be built using the following data sources/sensors:

- Microphone (e.g. phone's microphone or a separate one).
- Phone's location and movement traces.
- Video camera (regular cameras or smartglasses).

Assumption. The healthcare AAL application runs on the smartphone and considers the above three sources of data and detects any situation where a trigger needs to be generated to seek help. Such application is useful for home-alone people, and especially the elderly or disabled.

3.2.1. Indoor scenario

For example, Mr. Rhee, an elderly person living alone in his house, holding a glass in his hand loses his balance in the kitchen. The glass breaks and the sound of glass break (or if there is no glass break in a scenario, scream) is detected by the smartphone that is in the pocket of Mr. Rhee. The smartphone quickly tracks if Mr. Rhee moved or not and how was his movement. If the immediate movement was stable, no emergency is detected. If the immediate movement is not stable, any subsequent voice traces are analyzed (e.g. groaning in pain). Similarly, subsequent phone movement is detected as well (post-groaning) such as no movement at all means the subject is not able to move, and the subject is groaning as well means that an alert should be generated. No groaning and no movement might mean that the subject is unconscious, which also requires an alert to be generated.

The scenario is shown in Fig. 4.

In this application scenario, the existence of a video camera which is strategically located would be of great help. Due to privacy concerns, the camera only records the video locally and streams it to the client device when the client wants it (e.g. Mr. Rhee wants to see his dog that is on its own while Mr. Rhee is doing grocery). However, due to physical safety and threat to life, it becomes important at times to let the relevant part of the video stream/clip to any helpers (e.g. first responders). After all, life itself is above privacy. Hence, after the aforementioned detection of a potential alert, the video stream is generated which may then be analyzed by a dedicated fog node that is already doing other relevant jobs such as video streaming to client device when solicited, managing opportunistic healthcare, so on. Alternatively, the analysis may be performed by using a femto-cloud (Mr. Rhee's smartphone, laptop, and so on). Femto-cloud can be extended to any available participant neighbor's device(s) as well, based on the incentive of reciprocal computation.

The video analysis would show the situation of the subject (Mr. Rhee in this case) in a better way and a genuine trigger would be generated. The alert can be sent to the relevant contacts that are already stored on the smartphone or to the fog node (e.g. first responders, neighbors, relatives, ambulance, and so on). There can also be a small-scale tiny drone in the house as well, whose purpose is only to fly based on the instructions from the home-fog and take a footage of the person in need.

Challenges in indoor scenario:

- If the person in need has fallen somewhere which is hard to reach area (such as under the table) where camera footage is not possible (the smartglasses camera is also blocked) and a regular drone as well cannot reach and take footage, it will become hard to achieve the desired result.
  Machine learning approaches that take into account the first two constraints in detecting the emergency situation, without relying much on video could provide a possible solution.
  Deploying small-scale video drones is another solution (for the reachable areas).
- If the phone is not in the pocket and the person in need is away from it, it will be impossible to correctly detect (if at all) the sounds related to agony (or screaming "help"). The same applies when the person instantly becomes unconscious without any sound of distress.
  Maximizing the chances of emergency detection by increasing the number of sensors for event detection is necessary. For example, if the phone is not in the pocket, there can be a smartwatch that can detect such sounds, or strategically located microphones around the house (such as on the refrigerator, table lamp, TV, laptop's mic, etc.). Relying completely on phone's microphone will not be feasible in all the scenarios, since it will consume a large amount of battery power. However, Amazon's Alexa-like dedicated but small microphones would work.
- We assume that the first trigger is based on the received sound and the second trigger is based on the subsequent movements are the mandatory triggers. The third trigger waits for the confirmation through video data. However, if the video data is not available, the application may still generate the third trigger. Nevertheless, it depends on the accuracy of the first two triggers. This is a machine learning process that has to be intelligent enough to trigger when required.
- The first trigger gets canceled in the case of a dumb/mute person. In this case, the person has to tap his/her phone or smartwatch in a certain pattern. The role of smartglasses becomes even more

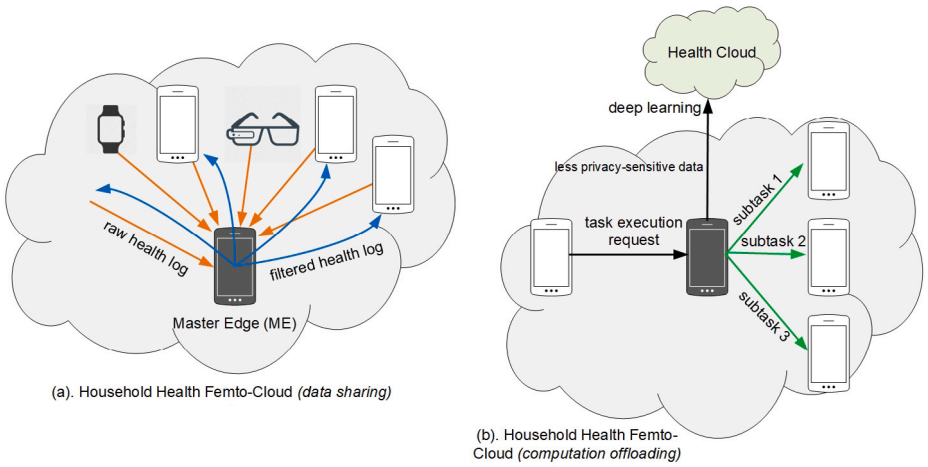

Fig. 3. (a). Health log sharing between the edge devices. (b). Task offloading among the edge devices in the femto-cloud.

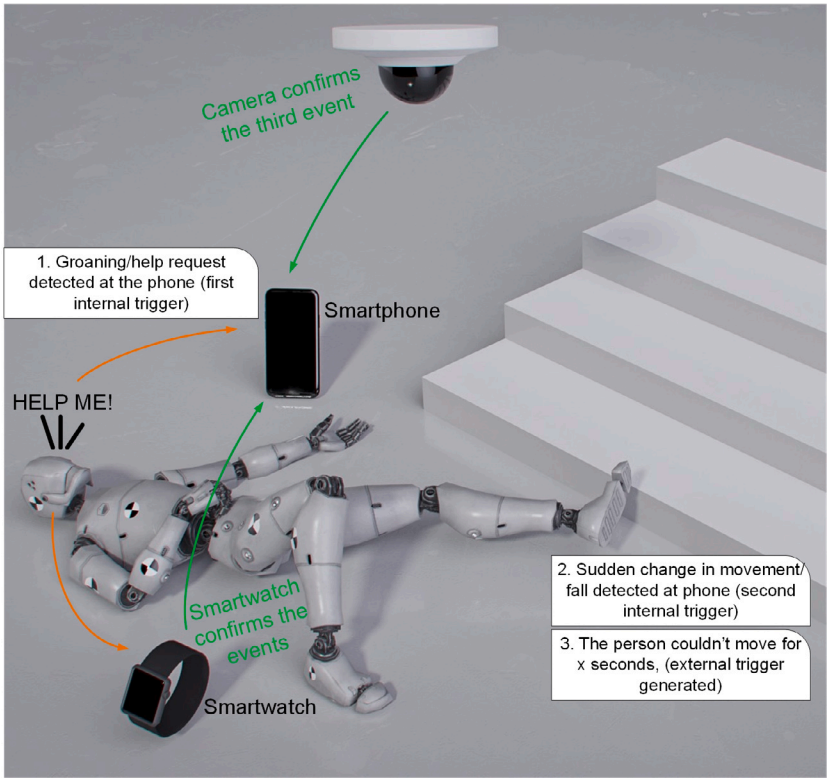

Fig. 4. Safety and emergency response in an indoor environment.

important in this case. Blinking of eyes in a certain pattern may also be used to generate a trigger, coupled with the tap pattern and sudden movement (e.g. falling) of the mute person.

3.2.2. Outdoor scenario

Mr. Rhee is walking down the street somewhere in the downtown of Suwon, Korea. In the meantime a robber approaches Mr. Rhee and tries to snatch his valuable personal belongings (such as phone, money). While the crime takes place, the emergency response application on the phone of Mr. Rhee detects unusual sound and an internal trigger is generated. With the internal trigger, the secondary component of the emergency response application gets activated in which the location of phone and sudden movement is detected. With the jerky phone snatch coupled with the previous brawl-like sounds, the application on the phone triggers an alert. If Mr. Rhee is wearing smartglasses with a camera, the video streaming (or a short clip) to the first responders (e.g. police) is started. If smartglasses do not exist (e.g. if the robber takes the smartglasses as well knowing that they can be the reason for him to get caught) then the trigger will be sent to the nearest police station that alerts the police as well as the CCTV camera operators in that region about the potential problem. Based on the CCTV footage, the police can try to locate the offender(s).

Fig. 5 shows such a scenario.

Similarly, if a person is being kidnapped, the person can scream for help that would be recorded as the first internal trigger.

Challenges in outdoor scenario: In the outdoor environment, the reliance on the first two internal triggers is a significant challenge because it depends on the scenario or situation. Hence, the availability of video resources becomes highly relevant. Some of the issues that can arise include:

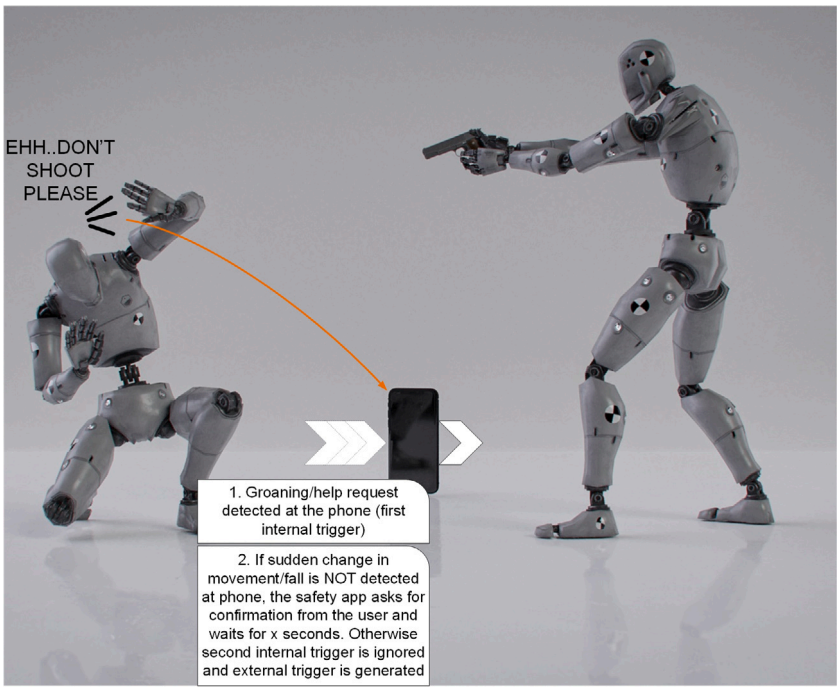

Fig. 5. Safety and emergency response in the case of a robbery.

- What if the person being kidnapped is sedated or the mouth is blocked to disallow the person from speaking? In that case, the first trigger will not be generated.
- Similarly, identifying the robbers/kidnappers is highly dependent on CCTV cameras. If the criminals can get out of the CCTV range, they can still get away. In that case, it comes down to facial recognition based identification of the criminals, which may help at a later point to apprehend them.
- In the case of kidnapping, the person himself/herself would be taken away in the vehicle. Assuming that the person's mouth is blocked and s/he is unable to speak, the application would not be able to learn whether the person took a ride or has been abducted (which is the key element in detecting the second internal trigger). In such a situation, if the person opportunistically taps on their phone/smartwatch in a certain pattern and/or blink his eyes in a certain pattern (which would be detected by the smartglasses), only then the trigger may be generated that would then notify the first responders to evaluate the video of CCTV camera.

The healthcare application has to be intelligent enough to be able to detect an unusual situation such as when the abducted person is suddenly being taken in a car and transported elsewhere. The application should be able to detect that there is a sudden unexpected speedy movement by the person who was previously taking a stroll. If the person had taken a cab, he would have stopped for a while at least. But in case of a kidnapping, the movements were too sudden and unexpected. The automatic detection of such a situation can be achieved through some machine learning task that the application has to perform. The application might also ask for time-restricted confirmation. For example, if there is no response in, say, 'x' seconds, the app might generate a "potential alert" to the first responders. The first responders from there can monitor the situation. This would be helpful and better at least than the situation where such a system does not exist at all.

3.3. Opportunistic healthcare in public transit and globally

Generally, people around the world either have easy access to healthcare resources, or are deprived of basic health. Here, for simplicity, we consider two broad categories of healthcare recipients: the privileged ones and the underprivileged ones. People in the privileged world tend to over-use the healthcare resources by visiting physicians unnecessarily on frequent occasions [28]. According to IBM Watson Health [28], up to 71% of hospital visits in America are unnecessary. In contrast, in the underprivileged parts of the world, people do not have access to healthcare resources according to their requirements. For example, in sub-Saharan Africa, at least 140% of improvement in the healthcare workforce is required in order to deliver the basic healthcare to the people. As a result, today, people end up exaggerating their initially nominal disease, or lose their life to a disease that could have been managed.

Cloud-IoT can offer some viable solutions for opportunistic healthcare (oHealth). For example, reaching out to the doctors during their available time (such as when off-duty or while they are commuting via public transit) and can answer some quick questions is one of the solutions that fit into smart healthcare definition, and enables truly pervasive healthcare. Health log and questions can be opportunistically offloaded to a nearby (such as at public transit) computing entity (such as fog server) capable of processing health logs. The fog node communicates the questions to a doctor, who provides the necessary feedback. Hence, the benefit is two-fold. First, the underprivileged patients gain access to physicians. This includes access to physicians in the developed world, through cloud computing technology. Second, unnecessary visits to the hospitals are avoided, saving monetary cost and time. In addition to that, through container migration, the offloading can be privacy-aware and stateful. The middleware technologies such as fog and edge resources will help integrate container migration with the overall service provisioning (and offload process). Container migration will help not just to containerize an entire running application (and hence enhance the privacy), but also, to maintain the state of an application and continue running it from thereon at the offloaded entity.

The notion of oHealth in this scenario is a timeserving healthcare service using wearable and smart devices. The infrastructure entities such as oHealth fog and oHealth cloud may be provided by the

government to ensure reliability, and to comply with the country's privacy related laws. However, private healthcare providers can also be involved at a later stage depending on the scale of quality assurance in a certain country or state. Fig. 6 [26] shows the communication pattern of oHealth in public transit.

Fig. 7 shows the architecture of oHealth. **oHealth edge (oHE)** represents the patient or the doctor. **oHealth fog (oHF)** is the middleware fog located at public transit stations (bus stop, train station, co-located with a 5G transceiver). In addition, 5G has the ability to locate a device with better accuracy by enhanced positioning, which can provide corresponding input to the edge computing algorithm if needed. This can be useful in those scenarios where oHF is co-located with a 5G transceiver, and additional location information is required. **oHealth cloud (oHC)** is the cloud component that manages all the other entities in the architecture, as well as the overall service.

The service request is generated by oHE as a patient (oHE_p) by contacting oHF. oHE_p sends the required health log to the oHF along with any necessary images such as pictures of wound, scar, rash, skin bump, bone lump, and so on. The patient is identified by the *profile manager* through the patient's existing account that was created during service registration on oHealth mobile application. oHF performs basic data quality checks through its module *data quality analyzer* to ensure that the received health log, questions and images comply with the service standards.

oHF then prepares the received data for executing the task which involves data trimming and preprocessing to produce an easy-to-answer questionnaire for the doctor. The preprocessing is performed by the *data processing module*. Afterwards, the necessary security and privacy measures are applied that may also include blockchain [29], through *security & privacy manager*. Blockhain would be important especially in the scenario where a long chain of entities (such as from one oHF to a series of others) is involved. The data is stored locally by the *storage controller* until not needed anymore. The oHF delegates the task through the *task manager* to the oHE as a doctor (oHE_d) as soon as a doctor gets in touch. All data communications are managed by the *communication manager*.

oHC is responsible for managing all the entities working under it. It manages all the oHFs, their identities, locations, device types, and history. It also performs real-time resource monitoring (so that resource sharing can be performed with an adjacent oHF, when required). Historical records are important in order to be able to determine the right amount of resources required at an oHF, and manage payments accordingly. In the case of oHE_d, oHC takes into consideration the identifications and license of the doctor, by using the *doctor profile and the location manager*. If it takes oHF too long to locate an oHE_d, depending on the urgency level set by oHE_p during initial service request and *service level agreement (SLA)*, then the oHF sends the data to the cloud in order to locate an oHE_d that is connected to any other oHF. Involving cloud in such a case has additional benefits as well. If such a service is standardized and launched worldwide, the cloud can reach out to doctors in other geographical regions, enabling the patients to access doctors even if there is huge difference between supply and demand. In this way, a patient sitting somewhere in sub-Saharan Africa can access a qualified doctor in Europe or America.

The oHC manages the financial aspect of the service as well. The payments may be made directly to the doctors through the patient's bank cards. Since people usually need suggestions or precautionary measures, it is assumed that the payments would be low. Doctors will give their feedback opportunistically in their underutilized time. In case the patient had visited the hospital, it would cost the hospital and the doctor more resources to handle the patient. Hence, a reasonably smaller payment is acceptable in this case. The payment can also be made through various incentives, such as transport ticket, shopping points/discounts, and so on. The payment may also involve insurance companies for which the cloud has to conduct the payment process accordingly.

The oHE manager is responsible for managing the oHE_d and oHE_p devices, their identities, location, and the historical records which describe all the contributions an oHE_d has made so far. It also takes into account the ratings oHE_d has received from the patients on each contribution. The ratings as well as the historical record on the questions/data of the oHE_p are involved when the payments/incentives are decided.

After the doctor oHE_d receives the data, s/he provides his/her feedback by answering the questions and provides suggestions according to the health conditions of the patient. The feedback is communicated to any oHF the oHE_d connects to while commuting. Depending upon the availability of the doctor, it may take from a few minutes to hours to give the required feedback. Once the feedback is received at an oHF, it is evaluated on data quality in order to make sure that the feedback is complete and complies with the standards set by the service provider (e.g. government).

We present a more in-depth discussion on oHealth in public transit in [26].

4. Evaluation

In this section, we evaluate the overall notion of healthcare IoT that includes machine learning (e.g. indoor/outdoor scenarios), and opportunistic healthcare through the amalgam of cloud-fog-IoT. We apply various machine learning algorithms on our proposed scenarios of healthcare, and discuss their performance. In this section we present a numerical performance evaluation of energy consumption in task offloading for effectively balancing consumption of resources in machine learning tasks. The goal here is to show how much energy is consumed when classification of human activity recognition (HAR) data is performed. Here, the objective is not to evaluate the accuracy of the classification of the algorithms used in this study. Rather, we want to elaborate that by utilizing the concept of task offloading, even machine learning-based evolutionary intelligent systems can be achieved locally with minimal cloud-involvement.

We selected three machine learning techniques widely used in HAR tasks, namely: k-nearest neighbors (kNN), naive Bayes (NB), and support vector classification (SVC) [30]. These three classification techniques are known for their high accuracy, and hence, we are incorporating these techniques in our evaluation. To parallelize each technique we use the map-reduce framework as presented in [23,31]. As a benchmark we used the publicly available dataset PAMAP2 [21] which maps raw data from three inertial measurement units and a heart rate monitor to 24 human activities.

In particular, we focus on online machine learning tasks. In online machine learning, patterns are extracted from continuous streams of data. As a consequence, the learned models evolve over time and are able to capture complex relationships among many different variables in non-stationary situations. This is in contrast to classical supervised learning approaches, which are trained completely from offline data and remain as static models. Online machine learning plays an important role in healthcare applications because it enables variations in the environment to be considered in order to understand the current situation and/or to make predictions about future events. For instance, online machine learning enables the capture of the dynamic nature of human activities in patient monitoring tasks [22].

Unfortunately, the increased adaptability of online machine learning comes at the cost of higher computational requirements with respect to the offline counterpart. In fact, online machine learning techniques often consume considerable amounts of computational resources. This might be an impediment to the use of online machine learning in IoT applications, because the devices in these networks typically have limited computational capabilities especially in terms of processing and storage. fog computing can enhance the capabilities of energy-constrained devices by helping to distribute some of the computing tasks among different devices.

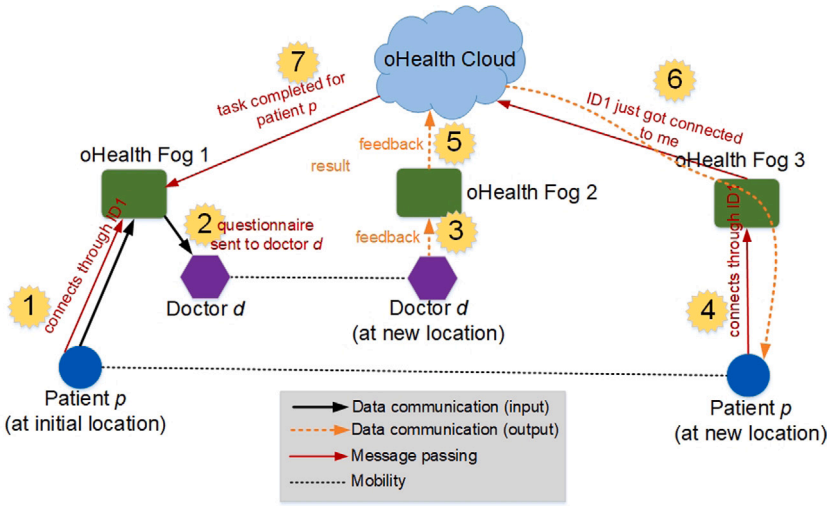

Fig. 6. oHealth communication pattern.

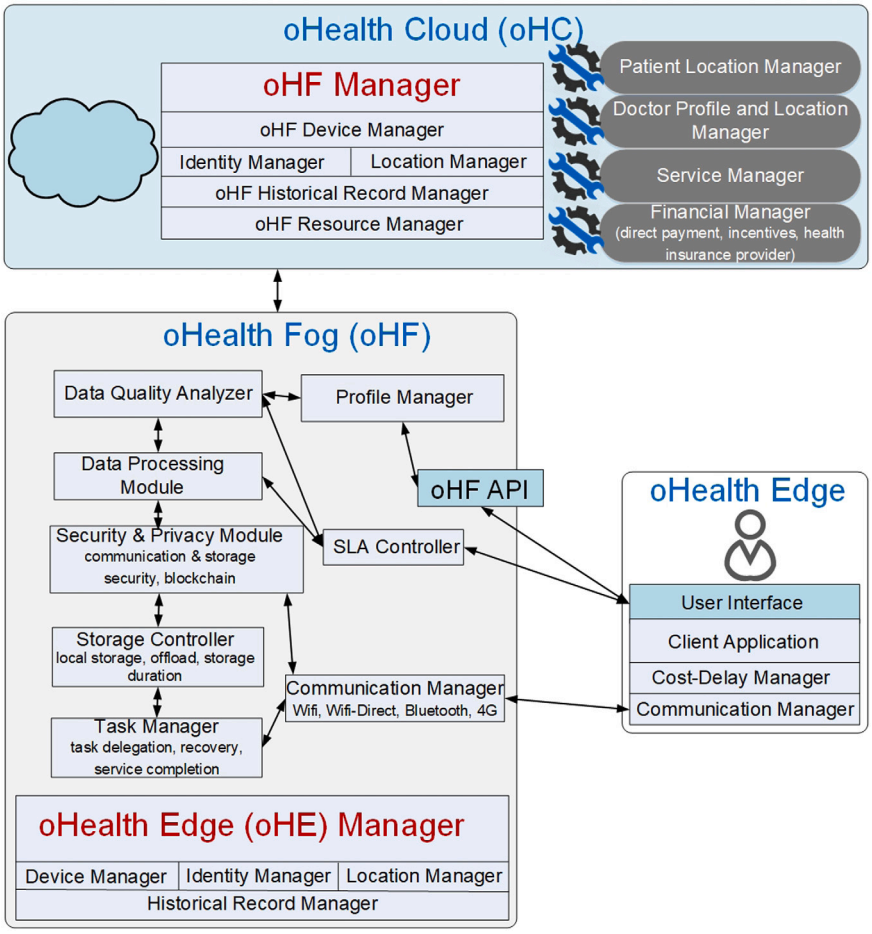

Fig. 7. oHealth architecture.

In the following, we evaluate the benefits of task offloading in scenarios where online machine learning is applied. We assume that the online implementation requires that the prediction models are re-trained at regular intervals, thus our focus is on the model training phase as a way to incorporate new data.

4.1. Experimental setup

We selected human activity recognition (HAR) [30] as the learning task. The goal of HAR is to recognize human physiological and daily activities using wearable sensor data. HAR has become widely

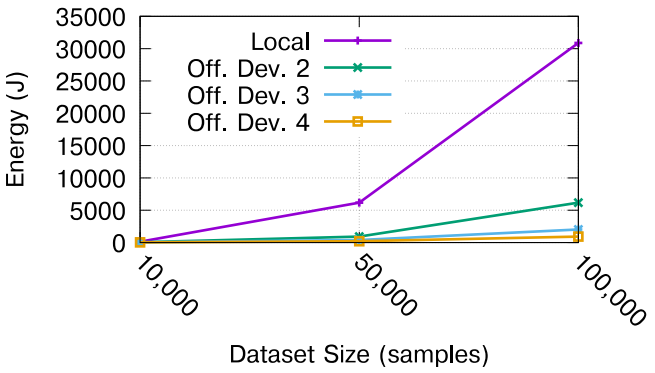

Fig. 8. Energy consumption using support vector classification.

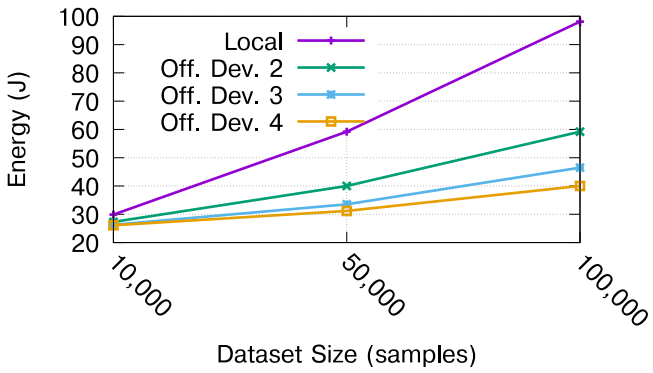

Fig. 9. Energy consumption using k-nearest neighbors classification.

popular due to the recent advances in wearable technology and many applications in healthcare, assisted living, and smart environments exist.

We identify two phases of task offloading:

First phase: When the training is done in one single device. This scenario has the minimum cost of communications involved. On the other hand, the time and energy necessary for computing is high.

Second phase: The smartphone can offload tasks to other devices (e.g., smartphones) in their vicinity using a wireless network to exchange the required data. In such scenarios, computing is distributed among different devices, but the communication overheads increase since all results must be collected back, which also results in additional energy consumption. We consider four settings in which the learning task is distributed from 1 (no task offloading) to 4 devices.

To model the energy cost incurred for computation we set the power spent for processing to be equal to 2845 mW on the smartphones [32, 33]. We compute the number of CPU cycles required by the execution of each algorithm using the Linux `perf` tool. In order to estimate the computation time in the smartphones we estimate the CPU time for a single core of an ARM Cortex-A9, 1.4 GHz using the number of CPU cycles. To model the energy cost incurred for communication, we set the power spent for communication using WiFi to 1749 mW [32]. To estimate the time spent for wireless transmission we consider IEEE 802.11g networks operating at 54 Mbps.

For each considered setting we applied the machine learning techniques over training data sampled from the PAMAP2 dataset with sizes 10,000, 50,000, and 100,000. This enables us to evaluate the benefits of task offloading with respect to the amount of collected sensor data between re-training tasks.

4.2. Performance results

Figs. 8, 9, and 10 show the energy consumption using each of the machine learning techniques for different dataset sizes. Task offloading provides the largest reward for SVC. This is due to the fact that the computational complexity of this technique is higher than the other machine learning techniques. We also observe a reduction by a factor of 2 for kNN. BN provides the least reduction in energy consumption because its computational cost is relatively very small compared to the other machine learning techniques.

5. Conclusion

With IoT devices emerging on the market, coupled with advancements in technologies related to mobile computing and communication, intelligent services are gaining momentum. As edge devices improve in their computational capabilities and become more affordable, these trends are paving the way for smart services in several domains that include healthcare, smart homes, smart cities, intelligent transportation

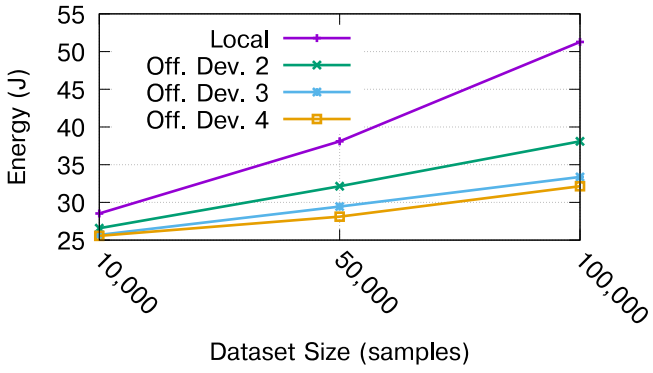

Fig. 10. Energy consumption using naive Bayes classification.

systems and so on. Intelligence in the services comes after extensive mining of the data and learning based on the outcomes. This requires a lot of processing capability and consumes energy both of which are normally beyond the capacity of a standalone edge devices such as smartphones. In this case, tasks are often offloaded to nearby devices or middlewares residing between the IoT and cloud. We have discussed how intelligent applications particularly related to healthcare that are based on machine learning can be leveraged in edge computing environment. We have discussed case-studies in indoor and outdoor environments for scenarios related to smart healthcare, safety, and emergency response. Our preliminary evaluation shows the results on the performance of machine learning algorithms from the perspective of presented scenarios. The results obtained validate the concept of intelligent and smart healthcare by offloading tasks in edge computing and femto-cloud.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] A.J. Perez, S. Zeadally, Privacy issues and solutions for consumer wearables, IT Prof. 20 (4) (2018) 46–56.

[2] E. Adi, A. Anwar, Z. Baig, S. Zeadally, Machine learning and data analytics for the IoT, Neural Comput. Appl. 32 (2020) 16205–16233.

[3] J. Muangprathub, N. Boonnam, S. Kajornkasirat, N. Lekbangpong, A. Wanichsombat, P. Nillaor, IoT and agriculture data analysis for smart farm, Comput. Electron. Agric. 156 (2019) 467–474.

[4] L. Zhao, J. Wang, J. Liu, N. Kato, Optimal edge resource allocation in IoT-based smart cities, IEEE Netw. 33 (2) (2019) 30–35.

[5] M. Aazam, X. Fernando, Fog assisted driver behavior monitoring for intelligent transportation system, in: 2017 IEEE 86th Vehicular Technology Conference, VTC-Fall, IEEE, 2017, pp. 1–5.
[6] A. Mukherjee, D. De, Low power offloading strategy for femto-cloud mobile network, Eng. Sci. Technol. Int. J. 19 (1) (2016) 260–270.
[7] F. Hussain, R. Hussain, S.A. Hassan, E. Hossain, Machine learning in IoT security: Current solutions and future challenges, IEEE Commun. Surv. Tutor. 22 (3) (2020) 1686–1721.
[8] M.D. Byrne, Machine learning in health care, J. PeriAnesthesia Nurs. 32 (5) (2017) 494–496.
[9] S. Zeadally, S.U. Khan, N. Chilamkurti, Energy-efficient networking: past, present, and future, J. Supercomput. 62 (3) (2012) 1093–1118.
[10] S. Zeadally, M. Badra, Privacy in a Digital, Networked World: Technologies, Implications and Solutions, Springer, 2015.
[11] M. Aazam, S. Zeadally, K.A. Harras, Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities, Future Gener. Comput. Syst. (2018).
[12] S. Wang, J. Xu, N. Zhang, Y. Liu, A survey on service migration in mobile edge computing, IEEE Access 6 (2018) 23511–23528.
[13] X. Zhao, L. Zhao, K. Liang, An energy consumption oriented offloading algorithm for fog computing, in: International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness, Springer, 2016, pp. 293–301.
[14] R. Hasan, M. Hossain, R. Khan, Aura: An incentive-driven ad-hoc IoT cloud framework for proximal mobile computation offloading, Future Gener. Comput. Syst. (2017).
[15] L. Pu, X. Chen, J. Xu, X. Fu, D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration, IEEE J. Sel. Areas Commun. 34 (12) (2016) 3887–3901.
[16] C. Meurisch, J. Gedeon, T.A.B. Nguyen, F. Kaup, M. Muhlhauser, Decision support for computational offloading by probing unknown services, in: Computer Communication and Networks (ICCCN), 2017 26th International Conference on, IEEE, 2017, pp. 1–9.
[17] Y. Wang, M. Sheng, X. Wang, L. Wang, J. Li, Mobile-edge computing: Partial computation offloading using dynamic voltage scaling, IEEE Trans. Commun. 64 (10) (2016) 4268–4282.
[18] R. Craciunescu, A. Mihovska, M. Mihaylov, S. Kyriazakos, R. Prasad, S. Halunga, Implementation of Fog computing for reliable E-health applications, in: Signals, Systems and Computers, 2015 49th Asilomar Conference on, IEEE, 2015, pp. 459–463.
[19] G. Aloi, G. Caliciuri, G. Fortino, R. Gravina, P. Pace, W. Russo, C. Savaglio, Enabling IoT interoperability through opportunistic smartphone-based mobile gateways, J. Netw. Comput. Appl. 81 (2017) 74–84.
[20] M. Chen, Y. Hao, K. Hwang, L. Wang, L. Wang, Disease prediction by machine learning over big data from healthcare communities, IEEE Access 5 (2017) 8869–8879.
[21] A. Reiss, D. Stricker, Introducing a new benchmarked dataset for activity monitoring, in: Proc. - Int. Symp. Wearable Comput. ISWC, No. June 2012, 2012, pp. 108–109.
[22] F.J. Ordonez, J.A. Iglesias, P. De Toledo, A. Ledezma, A. Sanchis, Online activity recognition using evolving classifiers, Expert Syst. Appl. 40 (4) (2013) 1248–1255.
[23] G. Song, J. Rochas, L.E. Beze, F. Huet, F. Magoules, K nearest neighbour joins for big data on MapReduce: A theoretical and experimental analysis, IEEE Trans. Knowl. Data Eng. 28 (9) (2016) 2376–2392, [Online]. Available: http://ieeexplore.ieee.org/document/7464884/.
[24] T. O'Shea, J. Hoydis, An introduction to deep learning for the physical layer, IEEE Trans. Cogn. Commun. Netw. 3 (4) (2017) 563–575.
[25] D. Zhao, Y. Dai, Z. Zhang, Computational intelligence in urban traffic signal control: A survey, IEEE Trans. Syst. Man Cybern. C 42 (4) (2012) 485–494.
[26] M. Aazam, X. Fernando, oHealth: Opportunistic healthcare in public transit through fog and edge computing, in: 4th IEEE International Conference on Smart Cloud (SmartCloud), 2019, IEEE, 2019.
[27] Household Size and Composition Around the World 2017, United Nations, 2017.
[28] Truven health analytics study finds most emergency room visits made by privately-insured patients are avoidable, 2018, [Online] https://tinyurl.com/ycotmu9d. (Accessed 27 August 2018).
[29] R. Sharma, Blockchain in Healthcare, France-Canada Chamber of Commerce, Ontario, Canada, 2018.
[30] O.D. Lara, M.A. Labrador, A survey on human activity recognition using wearable sensors, IEEE Commun. Surv. Tutor. 15 (3) (2013) 1192–1209, [Online]. Available: http://ieeexplore.ieee.org/document/6365160/.
[31] C.-T. Chu, S.K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, A.Y. Ng, K. Olukotun, Map-Reduce for Machine Learning on Multicore, in: NIPS, 2007, pp. 281–288.
[32] A. Carroll, G. Heiser, The systems hacker's guide to the galaxy energy usage in a modern smartphone, in: Proc. 4th Asia-Pacific Work. Syst. - APSys '13, Vol. 9300, ACM Press, New York, New York, USA, 2013, pp. 1–7, [Online]. Available: http://dl.acm.org/citation.cfm?doid=2500727.2500734.
[33] C. Ragona, F. Granelli, C. Fiandrino, D. Kliazovich, P. Bouvry, Energy-efficient computation offloading for wearable devices and smartphones in mobile cloud computing, in: IEEE Glob. Commun. Conf. GLOBECOM 2015, No. February 2018, 2015.

Mohammad Aazam (S'11, M'15, SM'16) is currently working as an associate professor at the University of Nottingham, Malaysia. Previously, he was a senior research scientist at Carnegie Mellon University in Qatar. He has also worked with Carleton University, Canada and Ryerson University, Canada as a postdoc fellow. He completed his Ph.D. Computer Engineering from Kyung Hee University, Korea, in 2015. In addition to that, he has completed a course on Data Science with R from Harvard University, USA in 2017, and a course on Internet of Things (IoT) from King's College London, UK, in 2016. He has more than 100 publications, including patents. He is also a founding member of IEEE SIG Intelligent Internet Edge (IIE). For additional details: www.aazamcs.com.

Sherali Zeadally is an associate professor in the College of Communication and Information at the University of Kentucky. He received his bachelor and doctorate degrees in computer science from the University of Cambridge, England, and the University of Buckingham, England, respectively. He is a Fellow of the British Computer Society and the Institution of Engineering Technology, England.

Eduardo Feo Flushing obtained a Ph.D. in Computer Science from the University of Lugano (USI) and the Dalle Molle Institute for Artificial Intelligence (IDSIA), in Switzerland. He is an Erasmus Mundus Master alumni and holds an MSc. in Informatics from the University of Trento (Italy) and an MSc. in Software System Engineering from RWTH-Aachen University (Germany). Currently, he is a Post-doctoral Associate in the Department of Computer Science of the Carnegie Mellon University in Qatar. His research focuses on cooperation in heterogeneous networked multi-agent systems, which encompasses topics in optimization, artificial intelligence, wireless networking, and multi-agent systems.